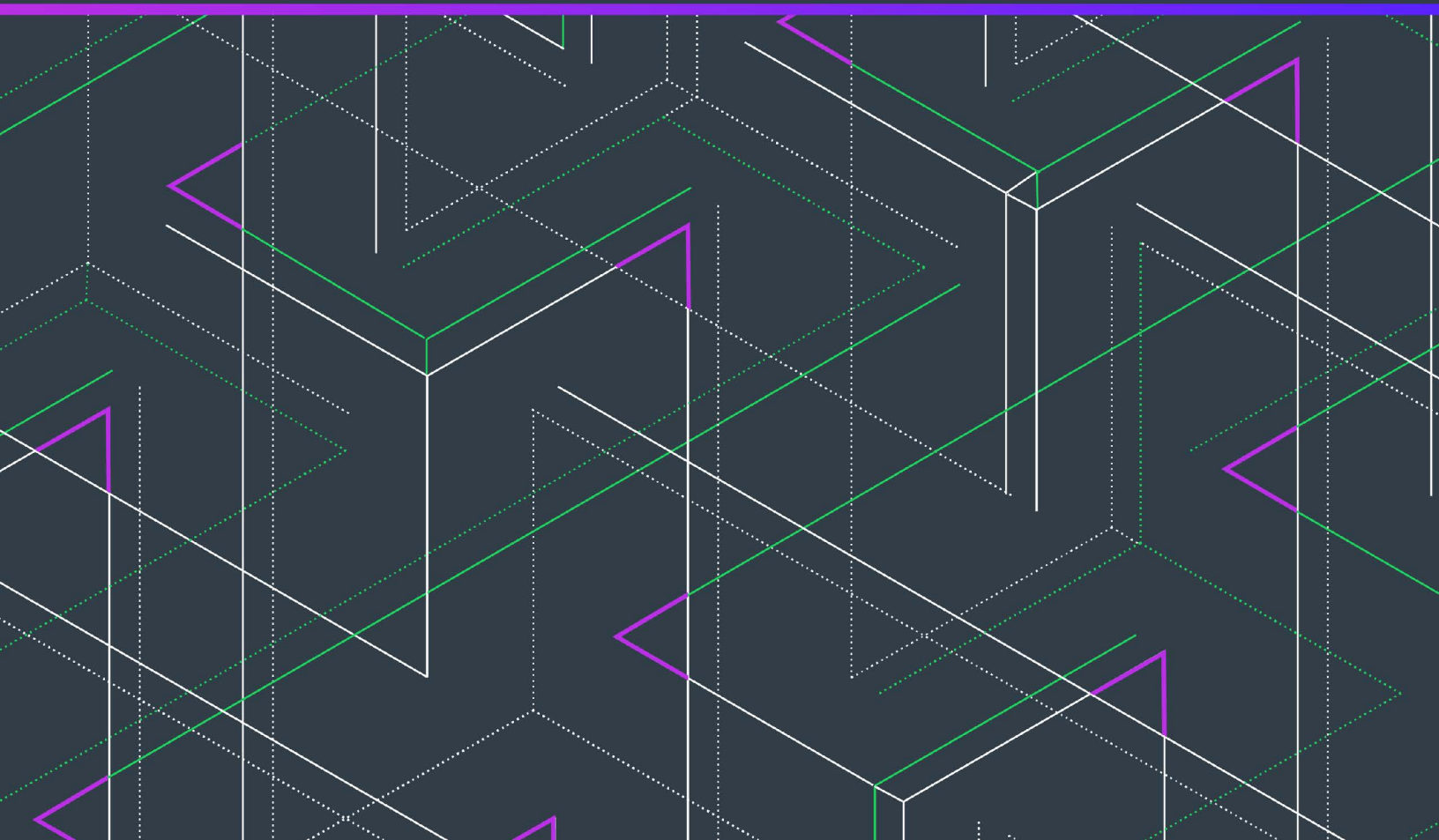
revenera

InstallShield 2025

ユーザー ガイド



法的情報

文書名: InstallShield 2025 ユーザー ガイド

部品番号: ISP-3100-UG00

製品のリリース日: 2025 年 12 月

著作権情報

Copyright © 2025 Flexera.All Rights Reserved.

この出版物には、Flexera およびそのライセンサーによって所有されている機密情報、創造的な製作物が含まれています。本出版物の一部または全部を、Flexera からの事前の書面による明示的許可なしに、使用、複製、出版、配布、表示、改変または転載することはいかなる形態または手段を問わず厳重に禁止いたします。Flexera によって書面で明示されている場合を除き、この出版物の所有は、禁反言、黙示などによっても、Flexera が所有するいかなる知的財産権の下、ライセンスまたは権利を一切付与するものではありません。

本技術およびそれに関する情報のすべての複製は、Flexera より許可されている場合に限り、著作権および所有権に関する通知を完全な形で表示しなければなりません。

知的財産

Flexera が所有する商標および特許の一覧は、https://www.flexera.com/producer/company/about/intellectual-property/ を参照してください。Flexera 製品、製品ドキュメント、およびマーケティング資料で言及されているその他すべてのブランドおよび製品名は、各社の商標または登録商標です。

(米国内向け) 制限付権利に関する表示

本ソフトウェアは商業用コンピュータ ソフトウェアです。本ソフトウェアのユーザーまたはライセンス許可対象者が米国政府の代理、部署、その他の関連機関の場合、ソフトウェアまたは技術データおよびマニュアルを含むすべての関連文書の使用、複写、複製、開示、変更、公開、または譲渡に関して、ライセンス契約または本契約の条項ならびに民生機関については連邦調達規則第 12.212 条または軍事機関については国防連邦調達規則補遺第 227.7202 条による制限が適用されます。本ソフトウェアは完全に自費で開発されたものです。その他一切の使用は禁止されています。

目次

1 InstallShield 2025 ヘルプ ライブラリ 69

InstallShield 2025 の新しい機能 71

InstallShield 2025 R2 の新しい機能 72

InstallShield 2025 R1 の新しい機能 73

InstallShield の以前のバージョンの新しい機能 74

InstallShield 2024 の新しい機能 75

InstallShield 2023 の新しい機能 76

InstallShield 2022 の新しい機能 80

InstallShield 2021 の新しい機能 81

InstallShield 2020 の新しい機能 85

InstallShield 2019 の新しい機能 93

InstallShield 2018 R2 の新しい機能 101

InstallShield 2018 SP1 の新しい機能 101

InstallShield 2018 の新しい機能 104

InstallShield 2016 SP2 の新しい機能 114

InstallShield 2016 SP1 の新しい機能 116

InstallShield 2016 の新しい機能 116

InstallShield 2015 SP1 の新しい機能 128

InstallShield 2015 の新しい機能 129

InstallShield 2014 SP1 の新しい機能 144

InstallShield 2014 の新しい機能 144

InstallShield 2013 SP1 の新しい機能 157

InstallShield 2013 の新しい機能 162

InstallShield 2012 Spring SP1 の新しい機能 175

InstallShield 2012 Spring の新しい機能 177

InstallShield 2012 SP1 の新しい機能 192

InstallShield 2012 の新しい機能 192

InstallShield 2011 の新しい機能 200

InstallShield 2010 の新しい機能 220

InstallShield 2010 SP1 の新しい機能 222

InstallShield 2010 の新しい機能 224
InstallShield 2009 SP2 の新しい機能 244
InstallShield 2009 SP1 の新しい機能 245
InstallShield 2009 の新しい機能 246
InstallShield 2008 の新しい機能 262
InstallShield 12 SP1 の新しい機能 283
InstallShield 12 の新しい機能 284
ターゲット システムの要件 290
64 ビット オペレーティング システムをターゲットにする 291
64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする 292
64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする 295
64 ビット オペレーティング システムをスイート/アドバンスト UI およびアドバンスト UI インストールでターゲットにする 296
32 ビットのオペレーティング システムを 64 ビットInstallScript セットアップ ランチャーを使ってターゲットする 297
InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い 298
32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い 299
ヘルプの使い方 302
お問い合わせ先 305

2 スタート ガイド 307
インストールの基本 308
アプリケーション ライフサイクル 309
InstallShield の起動 310
InstallShield スタート ページ 311
プロジェクトについて 312
インストール プロジェクトの概要 312
ニーズに合ったインストール プロジェクトを選択する 313
プロジェクトの種類 314
アドバンスト UI およびスイート/アドバンスト UI プロジェクト 317
MSIX プロジェクト 318
パッケージ情報 318
パッケージ ペイロード 318
メディア 319
基本の MSI インストール プロジェクト 319
InstallScript カスタム アクションを使った、基本の MSI インストールのランタイム動作 319
DIM プロジェクト 320
InstallScript インストール プロジェクト 321
InstallScript MSI インストール プロジェクト 322
InstallScript MSI インストールのランタイム動作 323
マージ モジュール プロジェクト 324
MSI データベースおよび MSM データベース プロジェクト 325
InstallScript オブジェクト プロジェクト 325
QuickPatch プロジェクト 325

トランスフォーム プロジェクト . . . 326
プロジェクトの使用 . . . 326
新しいプロジェクトの作成 . . . 326
プロジェクトを開く . . . 327
ソース コード管理からプロジェクトを開く . . . 327
ダイレクト編集モードでパッチ作成プロパティ ファイルを開く . . . 327
Windows Installer パッケージを開く . . . 328
マージ モジュールを開く . . . 328
オブジェクト プロジェクトを開く . . . 329
[再配布可能ファイル] ビューにマージ モジュールを配置する . . . 329
プロジェクトの保存 . . . 329
新しい名前と場所でプロジェクトを保存する . . . 329
プロジェクトを別の種類のプロジェクトに変換する . . . 330
基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する . . . 331
InstallScript MSI プロジェクトから InstallScript プロジェクトへ変換する . . . 332
GUID . . . 332
デフォルトのプロジェクトの場所を変更する . . . 333
プロジェクト要素を再利用する . . . 334
リポジトリを使用してプロジェクトの要素を共有する . . . 335
ネットワーク リポジトリを設定する . . . 335
プロジェクト テンプレート . . . 335
プロジェクト テンプレートの作成 . . . 336
テンプレートを使用した新規プロジェクトの作成 . . . 337
プロジェクト テンプレートをリポジトリにパブリッシュする . . . 337
サンプル プロジェクト . . . 338
プロジェクト アシスタント . . . 338
プロジェクト アシスタントを使用する . . . 339
プロジェクト アシスタントの [他のオプション]、[他の場所]、および [ヘルプ リンク] を使用する . . . 340
プロジェクト アシスタント内を移動する . . . 340
インストール デザイナーを開く . . . 341
プロジェクト アシスタントの表示または非表示 . . . 341
[アプリケーション情報] ページ . . . 342
コントロール パネルの [プログラムの追加と削除] . . . 342
インストールの会社名と製品名 . . . 342
[インストール要件] ページ . . . 343
プロジェクト アシスタントでオペレーティング システムを指定する . . . 343
インストールによる要件確認のタイミング . . . 344
ソフトウェア要件のランタイム メッセージを変更する . . . 344
カスタム インストール要件の作成 . . . 344
[インストール アーキテクチャ] ページ . . . 345
プロジェクト アシスタントで機能を追加する . . . 345
複数機能インストールを作成するかどうかを判断する . . . 346
複数の機能を持つインストールを作成する . . . 346
デフォルトの機能 . . . 347
機能の階層を定義する . . . 348
[アプリケーション ファイル] ページ . . . 348

プロジェクト アシスタントで機能にファイルを追加する . . . 348
プロジェクト アシスタントで機能からファイルを削除する . . . 349
固定のフォルダーの場所へファイルを追加する . . . 349
追加の定義済みフォルダーを表示する . . . 350
[アプリケーションのショートカット] ページ . . . 350
ファイル拡張子 . . . 351
インストールに含まれていないファイルへのショートカットを作成する . . . 351
プロジェクト アシスタントでデフォルトのショートカットを変更する . . . 352
プロジェクト アシスタントでショートカットのターゲットにファイル拡張子を関連付ける . . . 352
[アプリケーション レジストリ] ページ . . . 353
レジストリの更新 . . . 353
プロジェクト アシスタントでレジストリ データを構成する . . . 354
プロジェクト アシスタントでレジストリ データの値を変更する . . . 354
機能にレジストリ データを関連付ける . . . 355
レジストリ データで変数データ型を使用する . . . 355
アプリケーション パス . . . 356
[インストール インタビュー] ページ . . . 356
プロジェクト アシスタントでインストールに使用するダイアログを指定する . . . 356
エンドユーザーによるインストール先の変更を許可する . . . 357
使用許諾契約書 . . . 357
インストールが選択できるインストールを作成する . . . 358
[インストール ローカリゼーション] ページ . . . 359
プロジェクト アシスタントから文字列データをローカライズする . . . 359
インストールでローカライズされた文字列データの使用のされ方 . . . 360
[インストールのビルド] ページ . . . 360
プロジェクト アシスタントからインストールをビルドする . . . 360
プロジェクト アシスタントの完了後: 次のステップ . . . 362
InstallShield インターフェイスを使って作業する . . . 362
ビュー リストを表示する . . . 363
InstallShield ユーザー インターフェイスでビューを開く . . . 363
様々なビューで、[グループ ボックス] 領域を使って作業する . . . 363
ツールバーの表示または非表示 . . . 365
ツールバーにボタンおよびメニューを追加する . . . 366
ツールバーからボタンおよびメニューを削除する . . . 366
カスタム ツールバーの作成 . . . 366
[出力] ウィンドウを固定する/取り外す . . . 367
様々なビューで、[スクリプト エディター] ペインを使って作業する . . . 367
スクリプト エディターでブックマークを使用する . . . 368
スクリプト エディターでオートコンプリート機能を有効または無効にする . . . 368
スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う . . . 369
[スクリプト エディター] 内で InstallScript 関数の呼び出しについてのヒントを有効または無効にする . . . 370
スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する . . . 371
スクリプト エディターで構文ハイライト機能を有効または無効にする . . . 372
スクリプト エディターで構文ハイライトの色を変更する . . . 372
スクリプト エディターでテキストを表示するためのフォントを変更する . . . 373
スクリプト エディターで行番号を表示または非表示にする . . . 374

スクリプト エディター内で表示されているスクリプトの行番号へジャンプする *375*
スクリプト エディターで自動インデント機能を有効または無効にする 375
スクリプト エディターでタブ幅を設定する 375
スクリプト エディターで構文の折りたたみ機能を有効または無効にする 376
スクリプト エディターで空白スペースを表示または非表示にする 377
スクリプト エディター内でテキストをドラッグ アンド ドロップする 377
[スクリプト エディタ] 内からスクリプト ファイルを印刷する 378
スクリプト エディターのキーボード ショートカット 378
InstallShield の詳細設定を構成する 379
デジタル署名のタイムスタンプ サーバーを変更する 380
Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する 382
.cab ファイルの最大サイズを構成する 383
仮想メディア メモリの最大サイズを構成する 385
Standalone Build のポータブル実行可能ファイルの一覧を変更する 386
XML エンコード オプションのサポートを追加する 387
すべての仮想パッケージをビルドする場所を指定する 389
セットアップのデフォルト Clone_wait パラメーターを変更する 390
デジタル署名のプラットフォーム アーキテクチャを構成する 391
ランタイム中の MSI の検証を構成する 392
イベント設定コマンドの実行失敗に対するエラー表示の管理 393
リリースの配布用の仮想マシン設定を共有する 394
以前のバージョンの InstallShield からアップグレードする 395
InstallShield 2014 以前のプロジェクトをアップグレードする 396
InstallShield 2013 以前のプロジェクトをアップグレードする 399
InstallShield 2012 Spring 以前のプロジェクトをアップグレードする 399
InstallShield 2012 以前のプロジェクトをアップグレードする 403
InstallShield 2011 以前のプロジェクトをアップグレードする 404
InstallShield 2010 以前のプロジェクトをアップグレードする 407
InstallShield 2009 以前のプロジェクトをアップグレードする 411
InstallShield 2008 以前のプロジェクトをアップグレードする 422
プロジェクトを InstallShield 12 以前からアップグレードする 433
InstallShield 11.5 以前のプロジェクトをアップグレードする 436
InstallScript カスタム アクションを含む InstallShield 11.5 または以前の基本の MSI プロジェクトをアップグレードする 439
基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール *441*
InstallShield 11.5 以前の InstallScript MSI プロジェクトをアップグレードする 444
InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール *447*
InstallShield 11.5 以前の InstallScript プロジェクトをアップグレードする 452
InstallScript カスタム アクションを含む InstallShield 11.5 または以前の QuickPatch プロジェクトをアップグレードする 454
InstallShield 11.5 以前の InstallScript MSI プロジェクトに標準のパッチを作成する 454
InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトまたはこの種類のオブジェクトを含むプロジェクトをアップグレードする 454
InstallShield Express で作成したプロジェクトをアップグレードする 455
InstallShield Professional で作成したプロジェクトをアップグレードする 456

InstallShield Professional 6.x からの移行 456
InstallShield Professional 5.x からの移行 460
スクリプトの変更: 語彙変換 466
プロジェクト アシスタント ダイアログ サポートを InstallShield Professional プロジェクトからアップグレードされたプロジェクトに追加する 470
InstallShield—Windows Installer Edition で作成したプロジェクトをアップグレードする 470
プロパティ マネージャーでのアップグレード後の変更 471
InstallShield の他のエディションからアップグレードする 471
Express Edition から Professional Edition または Premier Edition にアップグレードする 476
Professional Edition から Premier Edition にアップグレードする 477
Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする 477
InstallShield のアップデートを取得する 481
InstallShield 実行時の言語サポート 481
言語 サポートのコード ページ要件 482
ビルド マシン上に補足言語サポートをインストールする 482
サポートされているアプリケーション プログラミング言語 483

3 チュートリアル 485

InstallScript プロジェクト チュートリアル 487
ステップ 1: プロジェクトの作成、ビルドおよびテスト 487
プロジェクト アシスタントを使ったプロジェクトの作成 488
アプリケーション情報を入力する 489
インストール アーキテクチャのカスタマイズ 489
プロジェクトへファイルを追加する 490
ショートカットの作成 490
レジストリ データを構成する 490
インストール インタビューを使ってダイアログを選択する 491
インストール用の言語を選択する 491
インストールをビルドする 491
インストールを実行する 492
InstallShield インターフェイスを使って作業する 492
機能のプロパティを設定する 492
"セットアップの種類" プロパティの設定 493
コンポーネントおよびファイルリンクを作成する 493
リリースのビルド 494
リリースに名前を付ける 494
メディアタイプと一般オプションの選択 495
パスワードとサポートするプラットフォームを指定する 495
セットアップ言語と含める機能を指定する 495
メディア レイアウトとダイアログの外観を定義する 496
インターネット オプションの指定およびアプリケーションのデジタル署名 496
アップデートとポスト ビルド情報を指定する 497
設定を確認する 497
インストールのトラブルシューティング 497
ステップ 2: ショートカットとレジストリ データ 498

ショートカットの作成 498
レジストリ データを作成する 499
ステップ 3: COM サーバーの登録 500
ステップ 4: 条件とプロパティ 501
ステップ 5: スクリプトでの作業 502
ステップ 6: ユーザー インターフェイスの変更 504
ユーザー入力の処理 504
表示ダイアログを変更する 505
ダイアログ エディターを使用する 505
基本の MSI プロジェクト チュートリアル 507
ステップ 1: プロジェクトの作成、ビルドおよびテスト 507
新規基本の MSI プロジェクトの作成 508
アプリケーション情報を入力する 509
インストール要件の設定 510
インストール アーキテクチャのカスタマイズ 511
プロジェクトへファイルを追加する 511
ショートカットの作成 512
レジストリ データを構成する 512
インストール インタビューを使ってダイアログを選択する 512
インストール用の言語を選択する 513
インストールをビルドする 513
インストールを実行する 514
IDE で作業する 515
機能のプロパティを設定する 515
コンポーネントおよびファイルリンクを作成する 516
リリースのビルド 516
製品構成とリリースに名前を付ける 517
フィルター設定と言語の指定 517
メディアの種類とディスク分割のオプションを選択する 518
圧縮の設定とセットアップランチャーのオプションを指定する 518
デジタル署名とパスワード保護を追加する 519
.NET Framework のサポートを含める/詳細設定の選択 519
設定を確認する 520
インストールのトラブルシューティング 520
ステップ 2: ショートカットとレジストリ データ 521
ショートカットの作成 521
レジストリ データを作成する 522
ステップ 3: COM サーバーの登録 524
ステップ 4: 条件とプロパティ 525
ステップ 5: エンドユーザー インターフェイスを変更する 527
新しいダイアログの追加 527
ダイアログ エディターでダイアログのレイアウトを変更する 528
グローバリゼーション チュートリアル 529
プロジェクトファイルを開く 529
ターゲット言語の選択 529
言語固有の文字列エントリを編集する 530

文字列エントリの作成 531
言語別ファイルとコンポーネントを含める 532
コンポーネント インストール条件を指定する 532
文字列の翻訳 533
インストールのビルド 534
インストールの実行 535
インストールのテスト 535

4 インストールの作成 537

まず始めに 538
Windows ロゴ プログラムの要件 539
Windows Installer 入門 540
インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する 540
アプリケーションのディスクの使用量を設定する 546
NSTALLDIR と TARGETDIR の違い 546
現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ 547
非管理者パッチのインストールを準備する 548
プラットフォームとプラットフォーム スイートの設定 550

インストール情報を指定する 553
一般的なプロジェクト設定を構成する 553
InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存する 554
製品名の指定 554
製品バージョンを指定する 555
InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおける製品バージョン番号 555
Windows Installer ベースのプロジェクトで製品コードを設定する 557
InstallScript ベースのプロジェクトで製品コードを設定する 557
アップグレード コードを設定する 558
製品の条件を設定する 558
デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定 560
ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護 561
プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する 564
Windows Installer インストールをログ記録するかどうかを指定する 565
InstallScript インストールを複数回実行する 566
"メンテナンスの有効化" 設定を構成する 567
InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い 568
InstallScript MSI インストールの InstallScript ユーザー インターフェイス タイプを指定する 574
概要情報ストリーム データを入力する 574
[概要情報ストリーム パネル] へのアクセス 575
概要情報ストリームのプロパティを設定する 575
"テンプレート概要" プロパティを使用する 576
[プログラムの追加と削除] 情報を構成する 577
Readme ファイルを指定する 578
製品のソフトウェア識別タグを含める 579
製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める 580

インストールのファイルを編成する 583
インストールをデザインする 583
アプリケーションをコンポーネントに分ける 584
アプリケーションを機能に分ける 585
パス変数を使用する 585
パス変数の種類 586
定義済みパス変数 587
標準パス変数 588
レジストリ パス変数 589
環境パス変数 590
カスタム パス変数の作成と定義 591
リリースのカスタム パス変数の値をオーバーライドする 592
プロジェクト内の絶対パスの参照をパス変数に変換する 593
パス変数の削除 594
ファイルとフォルダーを含める 594
プロジェクト ファイルに追加してターゲット システムにインストールする 595
コンテキスト メニューを使ってファイルをドラッグ アンド ドロップする 597
64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する 598
新しいインストール先フォルダーの作成 599
空フォルダーの作成 599
ハードコード化されたインストール先ディレクトリを指定する 600
ターゲット システムからファイルとフォルダーを削除する 601
プロジェクトでファイルとフォルダーを管理するときのヒント 602
プロジェクトでファイルとフォルダーを検索する 603
[ファイルとフォルダー] ビューにコンポーネントを表示する 604
[ファイルとフォルダー] ビューを使ってコンポーネントを変更する 604
ファイルの設定を構成する 604
ファイルの関連付け 605
新しいファイル拡張子の追加 606
ファイル拡張子の削除 606
ファイル拡張子のプロパティを設定する 607
コマンド動詞の指定 607
ファイル拡張子から動詞を削除する 608
新しい MIME タイプを追加する 608
MIME タイプを削除する 608
ProgID の作成 609
ProgID の削除 610
ダイナミック ファイル リンク 610
ダイナミック ファイル リンクの制限事項 611
ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する 612
ダイナミック リンクの作成 614
ダイナミック ファイル リンクをコンポーネントに追加する 615
ダイナミック ファイル リンクにキー ファイルを設定する 616
ターゲット システム上でファイルとコンポーネントを上書きする 617
[ファイル] ビューで定義済みフォルダーを表示する 618
コンパニオン ファイル 619
ファイルとフォルダーのアクセス許可を構成する 620

ファイルの追加時に COM データを抽出する 621
自己登録ファイルのインストール 621
.NET アセンブリのプロパティおよび依存関係を識別する 622
依存関係の 64 ビット .NET アセンブリをスキャンする 624
.NET 依存関係スキャナー結果の確認 625
InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする 626
Windows Fonts Folder にフォントをインストールする 626
グローバル フォント登録を有効または無効にする 627
フォント ファイルの登録を有効または無効にする 627
コンポーネントを使用する 628
セットアップ ベスト プラクティス 628
コンポーネントの作成 630
[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使ってコンポーネントを作成する 630
コンポーネント ウィザードのベスト プラクティス オプションを使用する 632
コンポーネント作成におけるベスト プラクティス規則 632
InstallShield による COM サーバー コンポーネントの作成 633
InstallShield によるフォント コンポーネントの作成 634
コンポーネント ウィザードの [コンポーネント タイプ] オプションを使用する 634
COM サーバーをインストールする 635
フォントのインストール 636
他のプロジェクトへコンポーネントをエクスポートする 636
プロジェクトのコンポーネントを削除する 637
コンポーネントと機能の関連付け 638
新しいコンポーネントを機能に関連付ける 638
既存コンポーネントを機能に関連付ける 639
コンポーネントと機能の関連付けを解除する 639
データをコンポーネントに追加する 640
ファイルをコンポーネントに追加する 640
コンポーネントのファイルの [リンク] 列で値を変更する 641
コンポーネントからファイルを削除する 642
レジストリ データをコンポーネントに追加する 642
[コンポーネント] ビューでショートカットを作成する 643
静的にリンクされたコンポーネントにサブフォルダーを追加する 643
コンポーネントのキーファイル 643
コンポーネントのキーファイルを設定する 644
コンポーネントからキー ファイルをクリアする 645
コンポーネントの設定 645
コンポーネントのインストール先と機能のインストール先の違い 646
コンポーネントのファイルのインストール先フォルダーを設定する 647
スクリプトからコンポーネントのインストール先を指定する 649
大文字のディレクトリ識別子とコンポーネントのインストール先 650
アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する 650
コンポーネント条件を構成する 651
再インストール中にコンポーネント条件を再評価する 652
共有ファイルの参照カウントを管理する 653
インストール前にファイルのバージョンを確認する 655
同じ名前のファイルをインストールする 655

コンポーネントの "リモート インストール" 設定を構成する *656*
コンポーネントの "リモート インストール" 設定と機能の "リモートインストール" 設定の違い 657
ビルド時に COM 登録データを抽出する *657*
.NET 依存関係とプロパティを確認するためのスキャン *658*
.NET アプリケーション ファイルを指定する *660*
.NET Installer クラスへ渡されたプロパティを読み込む *660*
レジストリ リフレクションの有効化と無効化 *662*
コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する *663*
コンポーネントの詳細設定 665
COM 登録の設定を手動で構成する *666*
COM 登録の COM クラスを手動で構成する 667
COM 登録の ProgID を手動で構成する 668
COM 登録のタイプ ライブラリを手動で構成する 669
ファイル拡張子を登録する *669*
アセンブリのインストール *670*
アセンブリを追加する 671
アセンブリを削除する 672
ターゲット システムで .NET アセンブリ サポートをテストする 672
コンポーネントのアプリケーションパスを指定する *673*
デバイスドライバー設定を構成する *674*
コンポーネントをパブリッシュする *674*
パブリッシュ情報を指定する 675
パブリッシュされるコンポーネントのコンポーネント ID を追加する 676
ComponentID を削除する 676
componentID へ修飾子を追加する 677
componentID から修飾子を削除する 678
修飾子の設定を構成する 678
機能の定義 679
機能の作成 680
機能の設定を構成する 681
機能のインストール先を設定する 681
機能の条件を設定する 682
機能をエンドユーザーへ表示する 684
機能の条件付き選択 685
機能の条件付き非表示 686
条件付きで実行時にスイート機能の脆弱性を設定する 687
機能のインストールを必須にする 688
機能のアドバタイズ 688
機能の "インストール レベル" 設定を構成する 689
機能の "リモート インストール" 設定を設定する 690
機能にリリース フラグを使用する 691
リリース フラグを機能に割り当てる *691*
リリース フラグを削除する *691*
機能の並べ替え 692
"必要な機能" 設定を使用する 692
セットアップの種類について 693
セットアップの種類を追加する 694
セットアップの種類を編集する 694
セットアップの種類の名前を変更する 695

セットアップの種類を削除する . . . 695
インストールに再配布可能ファイルを含める . . . 696
再配布可能ファイルの出荷 . . . 698
再配布可能ファイルギャラリーを管理する . . . 700
再配布可能ファイルをコンピューターにダウンロードする . . . 702
再配布可能ファイルギャラリーに InstallShield 前提条件を追加する . . . 704
再配布可能ファイルギャラリーから InstallShield 前提条件を削除する . . . 704
マージ モジュールを参照する . . . 705
マージ モジュールを参照した場合に起きること . . . 705
再配布可能ファイル ギャラリーにマージ モジュールを追加する . . . 706
再配布可能ファイルギャラリーからマージ モジュールを削除する . . . 707
InstallScript プロジェクトでオブジェクトを登録する . . . 707
InstallShield 前提条件、マージ モジュール、およびオブジェクトをプロジェクトに組み込む . . . 708
InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する . . . 708
InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する . . . 710
InstallShield 前提条件をプロジェクトから削除する . . . 711
プロジェクトからマージ モジュールとオブジェクトを削除する . . . 712
InstallShield 前提条件、マージ モジュールおよびオブジェクトのファイルを決定する . . . 712
インストール プロジェクトに含まれている InstallShield 前提条件を利用する . . . 713
セットアップ前提条件と機能前提条件の違い . . . 714
基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける . . . 715
基本の MSI プロジェクトで、機能から InstallShield 前提条件の関連付けを解除する . . . 716
InstallShield 前提条件のインストール順を指定する . . . 717
前提条件を含むリリースを構成する . . . 718
InstallShield 前提条件を含むディレクトリを指定する . . . 719
特定の InstallShield 前提条件の実行時の場所を指定する . . . 720
リリース フラグを InstallShield 前提条件に割り当てる . . . 721
InstallShield 前提条件を含むリリースをビルドする . . . 721
InstallShield 前提条件を含むインストールの実行時の動作 . . . 722
インストールに InstallShield 前提条件が含まれていたアプリケーションをアンインストールする . . . 726
インストール プロジェクトに含まれているマージ モジュールを使用する . . . 726
マージ モジュールを含むディレクトリを指定する . . . 727
マージ モジュールのインストール先をオーバーライドする . . . 728
マージ モジュールに関するトラブルシューティング . . . 729
マージ モジュールをインストール プロジェクト内からリポジトリにパブリッシュする . . . 729
Windows Installer 再配布可能ファイルをプロジェクトに追加する . . . 730
Microsoft Windows Installer の前提条件を含める . . . 731
.NET Framework 再配布可能ファイルをプロジェクトへ追加する . . . 733
Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める . . . 734
.NET Framework 再配布可能ファイルをプロジェクトへ追加する . . . 735
Microsoft .NET Core 前提条件を含める . . . 736
MySQL Connector ODBC 前提条件を含める . . . 737
Oracle 11g Instant Client の InstallShield 前提条件を含める . . . 738
DirectX 9.0 オブジェクトを含める . . . 738
アプリケーションの依存関係を識別する . . . 739

スタティック スキャン . . . 740
ダイナミック スキャン . . . 741
64 ビット依存関係のスキャン . . . 742
依存関係スキャナー結果の確認 . . . 742
依存関係スキャナーでファイルをフィルターする . . . 743
COM サーバーの登録 . . . 745
従来型の COM 登録 . . . 745
COM サーバーが自己登録をサポートするかどうかを判断する . . . 746
COM 情報を COM サーバーから抽出する . . . 746
COM 抽出のレジストリ変更をフィルターする . . . 748
COM サーバーの自己登録 . . . 750
自己登録メソッド . . . 751
InstallShield 自己登録 (ISSelfReg) . . . 752
レジストリ フリー COM 登録 . . . 752
インストールに Reg-Free COM ファイルを作成および変更する . . . 753
Ref-Free COM ファイルのサンプル マニフェスト ファイル . . . 754
プロジェクトに DIM を追加する . . . 754
インストール プロジェクトに DIM を統合するための適切な方法を判別する . . . 755
プロジェクトで DIM ファイルを参照する . . . 756
DIM リファレンスを機能に関連付ける . . . 757
ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する . . . 757
DIM リファレンスのビルドの手順を表示する . . . 759
DIM リファレンスのインストール先のオーバーライド . . . 759
DIM リファレンスからのカスタム アクションとダイアログをスケジュールする . . . 760
インストール プロジェクト内から参照された DIM プロジェクトを開く . . . 760
DIM をプロジェクトにインポートする . . . 761
DIM を基本の MSI プロジェクトにインポート中、デザイン時の競合を解決する . . . 761
インストール プロジェクトで DIM の要素を識別する . . . 762
インストール プロジェクトで使用する DIM プロジェクト内のパス変数をオーバーライドする . . . 762
ターゲット システムの構成 . . . 765
ショートカットおよびプログラム フォルダーの作成 . . . 765
ショートカットの種類 . . . 766
ショートカットの作成 . . . 768
ショートカットの設定を構成する . . . 769
ショートカットのアイコンを指定する . . . 769
インターネット ショートカットの作成 . . . 771
フォルダーにショートカットを作成する . . . 771
ショートカットにアクセスできるキーボード ショートカットを指定する . . . 773
ショートカット名の変更 . . . 773
ショートカットのシェル プロパティを設定する . . . 774
Windows 8 スタート画面への初回のショートカットのピン留めを抑制する . . . 775
エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する . . . 776
[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ . . . 778
カスタム アクション シェルのプロパティを設定する . . . 780
基本の MSI プロジェクトのアンインストール ショートカットを作成する . . . 781

InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットを作成する 782
スタート画面上のデスクトップ アプリのタイルの外観を構成する 783
レジストリの編集 784
コンポーネントまたは機能によるレジストリ エントリのフィルタリング 785
[レジストリ] ビューのリフレッシュ 786
レジストリ キーの作成 787
レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する 788
レジストリ キーの削除 789
レジストリ値の作成 790
レジストリ値データの変更 791
レジストリ値の削除 792
レジストリ フラグ 793
プロジェクト内のレジストリ エントリの検索 794
レジストリ キーのアンインストール動作を設定する 795
レジストリ値で環境変数を使用する 796
レジストリにプロパティ値を書き込む 797
レジストリ ファイルをインポートする 797
レジストリ ファイルのエクスポート 798
コンポーネントのキーパスを設定する 798
レジストリ キーのアクセス許可を構成する 799
レジストリ テーブルのプライマリ キー 800
Registry のプライマリ キーを指定する *800*
レジストリ キーに複数行の文字列値を入力する 801
ユーザーごとのインストールで HKCU の下にエントリを書き込む 802
レジストリで複数行文字列の設定または取得を行う 802
レジストリ関数の使い方 803
レジストリからデータを読み込む 804
.ini ファイル データの変更 805
.ini ファイルのリファレンスを作成する 806
既存の .ini ファイルをインポートする 807
.ini ファイルのセクションを指定する 807
.ini ファイルのキーワードを指定する 808
.ini ファイルからデータを読み込む 809
.ini ファイルからすべてのキー名を取得する 810
ODBC リソースの構成 810
ODBC リソースのインストール 811
追加の ODBC リソースを含める 811
ODBC リソースを機能に関連付ける 812
ODBC リソース属性の設定 813
環境変数を使用する 813
環境変数の設定 814
環境変数の例 815
XML ファイルの変更 816
XML ファイルの変更の概要 817
XML ファイルの変更の実行時の要件 818
XML ファイル リファレンスの作成 819

XML ファイルの場所を指定する 820
XML ファイルの変更のリファレンスを機能に関連付ける 821
ルート要素の追加 822
新しい要素の追加 822
要素を .NET 構成ファイルに追加する 823
属性を要素に追加する 824
属性の値を編集する 825
コンテンツを要素に追加する 825
XPath 式を使って、XML ファイル内の XML データを検索する 826
Windows Installer のプロパティを使用して、XML ファイルを動的に変更する 830
InstallScript テキスト置換を使用して、XML ファイルを動的に変更する 832
要素内で予約されている文字 (<、>、&、'、") を使用する 833
XML ファイルで名前空間を使用する 834
XML ファイルに名前空間のマッピングを宣言する *834*
名前空間プレフィックスを要素に追加する *835*
名前空間プレフィックスを属性に追加する *836*
名前空間プレフィックスを属性から削除する *836*
名前空間プレフィックスを要素から削除する *837*
XML ファイルから名前空間のマッピングを削除する *837*
XML ファイルに加えられるインストール時の変更をテストする 838
XML ファイルに加えられるアンインストール時の変更をテストする 839
[XML ファイルの変更] ビューから要素または XML ファイルを削除する 840
テキスト ファイルの変更 841
テキスト 変更セットを作成する 842
テキスト ファイルの変更を指定する 843
テキスト ファイルの変更が行われる順番を変更する 844
Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する 844
ANSI テキスト ファイルを開く時に使用するコード ページを指定する 847
[テキスト ファイルの変更] ビューから変更アイテムまたは変更セットを削除する 848
スケジュール タスク 848
スケジュール タスクの追加 849
Windows Installer プロパティを使って、スケジュール タスクをダイナミックに構成する 849
スケジュール タスクの削除 851
Windows サービスのインストール、制御、および構成 851
実行時に既定のユーザー アカウントを作成する 853
ユーザーごとのインストールとマシンごとのインストールの違い 854
管理者権限なしに InstallScript インストールを実行する 857
インストール動作のカスタマイズ 859
InstallScript を使用する 859
ISSetup.dll の概要 860
スクリプト ファイル 861
InstallScript ファイルの作成 *861*
InstallScript ファイルを開く *862*
スクリプト ファイルの挿入とインポート *862*
スクリプトのコンパイル *864*
スクリプトのデバッグ *864*

プリプロセッサ ステートメントを使用してスクリプトをデバッグする....865
InstallScript ファイルの名前を変更する....866
スクリプト中で文字列エントリを使用する....866
プロジェクトから InstallScript ファイルを削除する....866
スクリプト ライブラリ (.obl ファイル) を作成する....867
InstallScript ファイル (.rul および .h) をリポジトリにパブリッシュする....868
InstallScript リスト....869
リストの作成と破棄....869
リストへ要素を追加する....870
空白リストへ要素を追加する....871
現在の要素の前に要素を追加する....871
現在の要素の後に要素を追加する....872
現在の要素の前後に要素を追加する....872
リスト内にある既存の要素を変更する....873
リストから要素を削除する....874
リストの特定要素を検索する....875
リストの 1 番目と 2 番目の要素を取得する....875
リストへファイルを読み込む....876
リストのインデックスを設定する....876
リストのスキャン....877
ファイルへリストを書き込む....878
InstallScript ファイルの保存....878
システムの復元....878
プロパティの取得と設定....879
ビット フラグの使い方....881
文字列比較....882
ヌル区切り文字列の使用....882
相対パス....883
長いファイル名....883
長いファイル名と 16 ビット アプリケーション....883
長いファイル名と二重引用符....884
長いファイル名の形式....884
コンパイラで定数を定義する....884
スクリプトで Windows 定数を使用する....884
長い文字列リテラルのコーディング....885
絶対パス....885
関数のビルド....885
関数の呼び出し....886
DLL ファイル関数の呼び出し....886
DLL ファイル関数に配列を渡す....888
関数の宣言....888
関数から値を戻す....889
サポートされていない関数....890
エントリポイント関数の書き込み....890
関数ウィザードで関数の呼び出しを追加する....891
宣言済みの Windows API 関数....891
デフォルト以外の機能イベント ハンドラー関数を指定する....892
グローバル変数へのアクセス....892
初期化 (.ini) ファイル エントリのアンインストール....893

初期化 (.ini) ファイル エントリのアンインストールについての一般制限事項 *893*
AddProfString の初期化 (.ini) ファイル エントリをアンインストールする *894*
ReplaceProfString の初期化 (.ini) ファイル エントリをアンインストールする *895*
WriteProfString の初期化 (.ini) ファイル エントリをアンインストールする *895*
COM オブジェクトを使用してインストールを拡張する 896
関数呼び出しで機能やサブ機能を指定する 899
Windows API 関数を呼び出す 899
カスタム ファイルの転送操作を埋め込む 900
InstallScript で大きい数値を表現する 902
デバイス ドライバーのインストール 903
コンパイラ バージョンの確認 903
_ISCRIPT_ISDEV と _ISCRIPT_ISPRO を使用してオーサリング環境を確認する 904
別の InstallScript インストールからインストールを起動する 904
InstallScript の言語サポート 905
色化けの防止 907
カスタム アクションを使用 908
カスタム アクションウィザードを使用する 909
[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する 910
カスタム アクションのクローン 910
カスタム アクションを別のプロジェクトにエクスポートする 911
カスタム アクションの設定を構成する 912
カスタム アクションの種類についての概要 912
カスタム アクションの戻り値 913
スクリプト内実行 914
カスタム アクションの動作をドキュメント化する 916
Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン 917
カスタム アクションをリリース フラグに基づいて条件付きで起動する 918
.msi データベースにファイルを配置し、実行時に抽出する 918
遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う 920
InstallScript カスタム アクション 921
標準 DLLI ファイルの関数を呼び出す 922
Windows Installer DLL ファイル内の関数を呼び出す 923
カスタム アクションで DLL ファイル関数にパラメーターを渡す 925
マネージ アセンブリのパブリック メソッドを呼び出す 926
マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す *926*
マネージコード カスタム アクションの実行時要件 927
32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い 928
.NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す *928*
.NET Core カスタム アクションの実行時要件 929
32 ビットと 64 ビット .NET Core カスタム アクションの違い 929
アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する *930*
プロセスの強制終了カスタム アクションの呼び出し 932
PowerShell カスタム アクションの呼び出し 933
実行可能ファイルを起動する 935
Msiexec.exe を使用して 2 番目の Windows Installer インストールを起動する 938
ネスト インストール 940

ネスト インストール カスタム アクションを作成する *941*
ネストされたインストール カスタム アクションをシーケンスへ挿入する *942*
子製品を削除する *942*
インストールで MessageBoxA を呼び出す 944
ターゲット システム上のファイルを検索する 946
インストールが完了した後にアプリケーションを起動する 948
InstallScript カスタム アクション内部からインストールを終了する 948
スクリプトを通して ODBC プロパティを変更する 948
VBScript カスタム アクションで INSTALLDIR プロパティを使用する 949
アクション テキストを使う 949
アクションの説明とアクション データのテンプレートを指定する 950
進行状況ダイアログにアクションの説明を表示する 951
進行状況ダイアログにアクション データを表示する 952
アクションの説明またはアクション データのテンプレートを削除する 953
シーケンスを定義する 954
[インストール] シーケンス 954
[アドバタイズ] シーケンス 967
[管理] シーケンス 969
[ユーザー インターフェイス] シーケンス 971
[実行] シーケンス 971
アクションをシーケンスに挿入する 972
カスタム アクションを別のシーケンスへコピーする 973
シーケンスの順序を変更する 974
アクションをシーケンスから削除する 975
ロールバック カスタム アクションのシーケンス 976
.exe ファイルを起動するカスタム アクションをシーケンスする 976
DLL 内の関数を呼び出すカスタム アクションをシーケンスする 977
スクリプトを呼び出すカスタム アクションをシーケンスする 979
プロパティまたはディレクトリのプロパティを設定するカスタム アクションのシーケンス 980
2 番目の .msi パッケージを起動するカスタム アクションをシーケンスする 981
ISSetAllUsers カスタム アクション 982
サポート ファイルを使用する 983
サポート ファイルを追加する 983
ライセンス ファイルの追加 984
サポート ファイルの並べ替え 985
Disk1 フォルダーへファイルとフォルダーを追加する 985
Disk1 フォルダーからファイルまたはフォルダーを削除する 986
最後の Disk フォルダーにファイルとフォルダーを追加する 986
最後の Disk フォルダーからファイルとフォルダーを削除する 987
その他の Disk フォルダーにファイルとフォルダーを追加する 987
他の Disk フォルダーからファイルとフォルダーを削除する 988
サポート ファイルを削除する 988
エンドユーザー インターフェイスを定義する 989
ダイアログの使い方 989
プロジェクトでダイアログを使用する 990
ダイアログ ウィザードを使用して新しいダイアログを作成する *990*

既存のユーザー アカウントを作成または設定する機能を追加する *991*
他のプロジェクトで使用するダイアログを .isd ファイルへエクスポートする *993*
.isd ファイルからダイアログをインポートする *993*
ダイアログ ファイル (.isd) をリポジトリにパブリッシュする *994*
すべてのダイアログを .rc ファイルへエクスポートする *994*
リソース .dll ファイルからダイアログをインポートする *996*
他のプロジェクトへダイアログをエクスポートする *996*
プロジェクトからダイアログを削除する *997*
カスタム ダイアログの作成と構成 *998*
ダイアログ エディターで変更を元に戻す *998*
InstallScript および InstallScript MSI プロジェクトでダイアログを使用する 999
InstallScript および InstallScript MSI インストールでダイアログを表示する *1000*
InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるテキストを変更する *1001*
InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるデフォルト イメージについての背景情報 *1002*
InstallScript および InstallScript MSI プロジェクトで新しいカスタム ダイアログを作成する *1002*
新規ダイアログ ウィザードを使って、InstallScript または InstallScript MSI プロジェクトに新しいカスタム ダイアログを追加する *1003*
InstallScript または InstallScript MSI プロジェクトのダイアログ レイアウトを編集する *1003*
InstallScript または InstallScript MSI プロジェクトのダイアログにコントロールを追加する 1004
コントロールのプロパティを設定する 1005
ビットマップの上部にコントロールを表示する 1006
InstallScript を使用してカスタム ダイアログを実装する *1006*
InstallScript を利用してダイアログ コントロールを処理する *1009*
InstallScript または InstallScript MSI プロジェクトでダイアログの動作を編集する *1012*
"Sd ダイアログ スタティック テキスト" フィールドに製品名、製品バージョン、または、インストール済みバージョンを含むフィールドを追加する *1012*
ダイアログで HTML コントロールを使用する *1013*
デフォルトのダイアログに戻す *1015*
リソース コンパイラとリソース リンカ *1016*
ダイアログ サンプラー *1016*
ダイアログ スキン *1017*
ダイアログ スキンの指定 1018
基本の MSI プロジェクトでダイアログを使用する 1019
基本の MSI インストール中にダイアログを表示する *1019*
基本の MSI プロジェクトで新しいダイアログを作成する *1020*
基本の MSI プロジェクトでダイアログ レイアウトを編集する *1021*
ダイアログ エディターを開く 1021
基本の MSI プロジェクトでダイアログにコントロールを追加する 1022
コントロールの設定を構成する 1022
ビットマップ上にコントロールを表示する 1023
基本の MSI プロジェクトでダイアログ レイアウトを編集する *1024*
基本の MSI ダイアログでコントロール イベントを使用する *1024*
基本の MSI ダイアログでサブスクリプションを使用する *1028*
基本の MSI ダイアログでコントロール イベントをトリガーする *1030*
ダイアログ コントロールに新しいイベントを追加する 1030
ダイアログ コントロールのイベントを並べ替える 1031
イベントをダイアログ コントロールから削除する 1031
ダイアログ ボタンからカスタム アクションを起動する *1032*

[カスタム アクションとシーケンス] ビューでエンドユーザー ダイアログの順序の表示する (基本の MSI プロジェクト) *1032*
CustomSetup ダイアログのオプション *1034*
ダイアログのテーマ *1036*
幅の広いダイアログ テーマにおける実行時の要件 1036
ダイアログ テーマのプレビュー 1036
ダイアログ テーマの選択または変更 1037
テーマをカスタム外部ダイアログに適用する 1038
ロゴまたは他のイメージを外部ダイアログを追加する 1039
提供されているテーマと対応するダイアログのサイズ 1040
Circles テーマ (大) 1041
Classic テーマ 1042
Cooperation テーマ (大) 1043
Filmstripテーマ (大) 1044
Global テーマ 1046
InstallShield Blue テーマ 1047
InstallShield Blue テーマ (大) 1048
InstallShield Silver テーマ 1049
Monitor テーマ 1050
Pastel Wheat テーマ 1051
Theater テーマ (大) 1053
右から左に記述される言語のダイアログ サポート *1054*
[ファイルを開く] ダイアログを起動する *1055*
LicenseAgreement ダイアログでエンドユーザーが EULA を最初から最後までスクロールするのを必須にする *1058*
[印刷] ボタンをダイアログに追加する *1061*
Windows Vista 以降のシステムの再起動を最小限にする *1063*
ダイアログのコントロール 1064
ビルボード コントロール *1068*
ビットマップ コントロール *1070*
[チェック ボックス] ボタン *1075*
コンボ ボックス コントロール *1080*
ダイアログ コントロール *1085*
[ディレクトリ コンボ] コントロール *1088*
[ディレクトリ リスト] コントロール *1091*
[編集フィールド] コントロール *1094*
[グループ ボックス] コントロール *1099*
ハイパーリンク コントロール *1102*
アイコン コントロール *1106*
行コントロール *1109*
リスト ボックス コントロール *1112*
リスト ビュー コントロール *1117*
マスク編集コントロール *1122*
パス編集コントロール *1124*
[進行状況バー] コントロール *1127*
[プッシュ ボタン] コントロール *1130*
[ラジオ ボタン] コントロール *1135*
[ラジオ ボタン グループ] コントロール *1138*
[スクロール可能テキスト] コントロール *1143*
[選択ツリー] コントロール *1145*

[テキスト領域] コントロール *1149*
[ボリューム コスト リスト] コントロール *1154*
[ボリューム選択コンボ] コントロール *1157*
コントロールのコピーと貼り付け *1160*
コントロールの切り取りと貼り付け *1160*
ウィザード インターフェイスを使って作業する 1160
ウィザード インターフェイスの要素 1161
スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする 1163
カスタム スタイルをウィザード インターフェイスに定義する *1164*
テキスト スタイルをウィザード インターフェイスのテキストに適用する *1165*
カスタム ブラシをウィザード インターフェイスに定義する *1166*
ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する *1167*
ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする *1169*
カスタム コントロール テーマをウィザード インターフェイスに定義する *1169*
テーマをウィザード インターフェイスのコントロールに適用する *1170*
最小ウィザード ページのカスタマイズ *1172*
ウィザード インターフェイスのフォーマットを選択する 1173
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白のウィザード ページを作成する 1174
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、定義済みのウィザード ページを追加する 1175
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい 2 番目のウィンドウを作成する 1176
シーケンス ウィザード ページ 1177
ウィザード ページまたは 2 番目のウィンドウのレイアウトおよび動作を編集する 1177
ウィザード ページまたは 2 番目のウィンドウにコントロールを追加する *1178*
ウィザード ページまたは 2 番目のウィンドウに含まれるコントロールのタブの順番を変更する *1179*
ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用 *1179*
ウィザード ページでナビゲーション ボタンを使う *1181*
ウィザード ページのナビゲーション ボタンのデフォルト動作をオーバーライドする 1183
ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する *1184*
ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する *1186*
ウィザード インターフェイス内の要素のアクションを構成する *1189*
ウィザード インターフェイスにおけるイメージ、テキスト ファイル、その他のオブジェクトの使用 1195
ウィザード インターフェイスにおける DPI 対応 1195
エンドユーザー インターフェイスのローカライズ 1196
複数言語をサポートするプロジェクトで文字列エントリを処理する 1197
InstallShield で文字列エントリを使用する 1198
文字列エントリを追加する 1200
文字列エントリを編集する 1201
文字列 ID のインスタンスを検索する 1202
文字列エントリの検索と置換 1203
文字列エントリを削除する 1204
ローカライズ可能な設定から文字列 ID を削除する 1204
ビルボードを表示する 1205
基本の MSI インストールでビルボードを表示する 1205
基本の MSI プロジェクトにおけるビルボード ファイルの種類 *1206*
基本の MSI プロジェクトにおけるビルボードの種類 *1206*
基本の MSI プロジェクトで使用するビルボードの種類を指定する 1209

基本の MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する *1209*
基本の MSI プロジェクトにイメージ ビルボードを追加する *1210*
基本の MSI プロジェクトでビルボードの設定を構成する *1210*
リリースをビルドまたは起動せずにビルボードをプレビューする *1211*
基本の MSI プロジェクトでビルボードの順序を設定する *1211*
ビルボードを含む基本の MSI インストールにおける実行時の動作 *1212*
基本の MSI プロジェクトからビルボードを削除する *1213*
InstallScript および InstallScript MSI インストールでビルボードを表示する 1213
InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類 *1213*
InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前 *1215*
InstallScript または InstallScript MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する *1216*
InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する *1217*
InstallScript または InstallScript MSI プロジェクトでコードを追加または変更してビルボードを表示する *1218*
InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する *1218*
InstallScript または InstallScript MSI インストール中に特殊効果を持つビルボードを表示する *1219*
InstallScript または InstallScript MSI プロジェクトでビルボードを画面上の異なる場所に移動させる *1219*
InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する *1220*
InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する 1220
実行時にリスト ボックスへデータを挿入する 1221
ファイル参照ダイアログを表示する 1222
InstallScript インストールでネットワーク参照ダイアログを表示する 1222
サーバーの構成 1223
SQL サポートの構成 1223
SQL ログインの設定に Windows Installer プロパティを使用する 1224
新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する *1226*
InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する 1227
新しい SQL 接続の追加 1228
SQL Server データベースのデフォルトの TCP/IP ネットワーク ライブラリを別のプロトコルで上書きする *1228*
SQL Server Express LocalDB のインスタンスに接続するときの要件 *1229*
新しい SQL スクリプトの追加 1230
SQL スクリプトの挿入およびインポート 1231
SQL Server データベースのインポートと SQL スクリプト ファイルの生成 1232
SQL スクリプト ファイルの編集 1233
SQL スクリプト ファイルのバージョン番号を指定する 1234
接続に関連付けられている複数 SQL スクリプトの実行順序を指定する 1235
デフォルトの SQL ランタイム動作をオーバーライドする 1236
SQL ランタイム エラーの処理 1238
SQL スクリプトにデータベース サーバー タイプの条件を設定 1238
Windows Installer ベースのインストールで SQL スクリプトを条件付きで起動する 1240
InstallScript インストールで SQL スクリプトを条件付きで起動する 1241
SQL スクリプトが完全 SQL サーバーに対してのみ実行されるように要求する 1242
Windows Installer プロパティを使って SQL スクリプトの文字列を動的に置換する 1242
InstallScript テキスト置換を使って SQL スクリプトの文字列を動的に置換する 1244
Windows Installer ベースのインストールで SQL サーバー側インストールを要求する 1246
InstallScript プロジェクトのサーバー側インストールを要求する 1247

SQL スクリプト ファイルをリポジトリにパブリッシュする 1247
MySQL ODBC ドライバーをインストールする 1248
アンインストール中に、SQL データベースを削除する 1249
Microsoft SQL Azure Database サーバーへの接続と SQL スクリプトの実行 1250
Oracle のインスタンスに接続して SQL スクリプトを実行する 1250
Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する *1251*
Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する *1253*
Oracle 11g Instant Client をインストールする *1256*
Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する *1257*
カスタマイズした SQL スクリプトを実行して SQL Server データベースを作成するサンプル プロジェクトを作成する 1258
カスタマイズした SQL スクリプトを実行して Oracle スキーマを作成するサンプル インストールを作成する 1261
スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する 1263
COM+ アプリケーションとコンポーネントの管理 1265
COM+ サーバー アプリケーションの管理 1265
COM+ アプリケーション プロキシの管理 1266
サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める 1268
インターネット インフォメーション サービス 1268
InstallShield における IIS サポートのバージョン固有情報 1269
インストールする IIS のバージョンを判別する *1270*
IIS サポートの実行時要件 1270
Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する 1271
Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加 1272
IIS Web サイトのスキャンをして、その設定を InstallShield プロジェクトにインポートする *1273*
Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする1275
ネスト仮想ディレクトリの作成 *1277*
TCP ポート番号とサイト番号の構成 1278
Web サイトの IIS ホスト ヘッダー名を指定する 1281
Web サイトの SSL 証明書を指定する 1281
実行時に IIS 証明書ファイルを設定する 1282
ファイルを IIS 仮想ディレクトリに追加する 1283
[IIS 構成] ビューからアプリケーションと仮想ディレクトリを削除する 1284
アプリケーション プールの追加 1284
[IIS 構成] ビューからアプリケーション プールを削除する 1285
Web サービス拡張の追加 1286
[IIS 構成] ビューから Web サービス拡張を削除する 1286
IIS サポートの機能とコンポーネントの関連付け 1287
Web サイト、アプリケーション、および仮想ディレクトリのアンインストール 1288
Web サービス拡張とアプリケーション プールのアンインストール 1289
Web サイトまたはアプリケーションの ASP.NET バージョンを設定する 1290
繰り返しまたは 1 回限りの ASP.NET 登録を行うかどうかを指定する 1291
64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮 1291
Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する 1294
Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する 1295
ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する 1295

IIS の詳細設定を構成する 1297
Web サイト、アプリケーション、または仮想ディレクトリのカスタム エラー メッセージを定義する 1298
Windows Installer のプロパティを使用して、IIS 設定を動的に変更する 1298
InstallScript テキスト置換を使用して、IIS 設定を動的に変更する 1300
ターゲット マシン上で Web サービスを配布する 1301
Web アプリケーションのフォーム認証を有効化する 1302
IISROOTFOLDER サポートの追加 1302
IIS_WEBSITE_NAME プロパティ 1303
IIS_PORT_NUMBER プロパティ 1303
メンテナンスおよびアンインストールのためのインストールを作成する 1305
アンインストールのための InstallScript インストールを作成 1306
アンインストール時またはインストール時のみにスクリプト コードを実行する 1307
アンインストール用にログされた InstallScript 関数 1308
メンテナンス/アンインストール機能 1309
製品が作成したファイルの削除 1309
製品が作成したレジストリデータの削除 1310
トランザクション処理を使って複数パッケージ インストールを構成する 1313
トランザクション処理を使った複数パッケージ インストールについての概要 1313
新しい連鎖 .msi パッケージをプロジェクトに追加する 1314
リリース フラグを連鎖 .msi パッケージに割り当てる 1316
プロジェクトから連鎖 .msi パッケージを削除する 1316
インストールのビルド、テスト、および配布 1319
リリースの作成とビルド 1319
製品構成の使用 1320
ビルドに適切なアーキテクチャ検証の種類を選択する 1320
リリースに関する作業 1323
リリース ウィザードを使用して、リリースの作成およびビルドを行う 1324
[リリース] ビューを使用して、リリースの作成およびビルドを行う 1324
リリースの場所の設定 1326
コマンドラインからのリリースのビルド 1326
ISCmdBld.exe を使用して、コマンドラインからリリースをビルドする 1327
.ini ファイルでコマンドライン ビルド パラメーターを渡す 1327
ISBuild.exe を使用してコマンドラインからのビルドする 1333
コマンドラインからの自己展開型実行可能ファイルをビルドする 1333
リリースの再ビルド 1333
クイック ビルドを実行する 1334
バッチ ビルドを実行する 1335
ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する 1336
ファイルの準備イベントを使って、署名されていない InstallShield ファイルに署名を行う 1338
ビルドエラーと警告を解決する 1339
コマンドラインからプロジェクトの設定を変更する 1340
ビルドのキャンセル 1340
Standalone Build 1341
Standalone Build をビルド マシンにインストールする 1341
Standalone Build 用に再配布可能ファイルをビルド マシンへ追加する 1344
スタンドアロン コマンドライン ビルド 1346
スタンドアロン オートメーション インターフェイス 1346
Standalone Build と InstallShield を同一のマシン上にインストールする 1346

Microsoft ビルド エンジン (MSBuild) *1347*
MSBuild を使用して、コマンドラインからリリースをビルドする 1355
リリースの設定を構成する 1356
配布する単一実行可能ファイルの作成 *1356*
ディスク イメージにファイルを非圧縮のまま残す *1357*
Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する *1358*
InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する *1360*
インストールのパスワード保護 *1362*
リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する *1362*
リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する *1363*
スイート でパッケージのアンインストール順を指定する *1365*
デジタル署名とセキュリティ 1366
ビルド時にリリースとそのファイルにデジタル署名を行う *1368*
コマンドラインからビルドされたリリースにデジタル署名を行う *1369*
リリース フラグ 1370
製品構成フラグとリリース フラグの違い *1371*
リリース フラグに基づくフィルター *1372*
1 つのプロジェクトから複数の製品構成のインストールを作成する 1374
標準 Web インストールと One-Click Install の違い 1374
InstallScript プロジェクトの One-Click Install インストール *1375*
One-Click Install オブジェクト *1376*
Player オブジェクト *1377*
LastError メソッド 1377
Open メソッド 1378
Ether オブジェクト *1379*
GetLastError メソッド 1379
IsPlaying メソッド 1381
Play メソッド 1381
SetProperty メソッド 1382
One-Click Install インストールのデジタル署名 *1382*
古いプロパティとメソッドの置換 *1383*
デフォルトの Web ページ (Setup.htm) *1384*
One-Click Install インストール *1385*
Setup Player の別の呼び出し方法 *1385*
HTTPS と One-Click Install インストール *1385*
One-Click Install インストールの作成 *1386*
Setup Player を使用する 1387
起動された ClickOnce Install インストールにデータを渡す 1387
インターネット インストールをサイレントで実行する 1387
ランタイム言語の設定 1387
デバッグ モードで One-Click Install インストールを実行する 1388
InstallScript からユーザーを認証する 1388
Setup.ini 1389
[Startup] セクション *1390*
[Mif] セクション *1392*
リリースのクローンを作成する 1393
Windows アプリ パッケージの作成 1393
インストールのテストと実行 1395

インストールのテスト実行 1395
InstallShield インターフェイスからインストールを実行する 1396
サイレント モードでインストールを実行する 1396
InstallScript MSI と InstallScript のサイレント インストール *1397*
サイレント インストールの作成 *1398*
応答ファイルを作成する *1398*
応答ファイルのサイレント ヘッダー 1400
応答ファイルのアプリケーション ヘッダー 1400
応答ファイルのダイアログ シーケンス 1401
応答ファイルのダイアログ データ 1402
サンプル応答ファイル 1411
サイレント インストールの実行 *1412*
Setup.log ファイルを使用してエラーを確認する *1413*
InstallScript MSI プロジェクトの MSI サイレント モード インストール 1415
コマンドラインを使ったサイレント アンインストール (InstallScript MSI のみ) 1416
スイート インストールをコマンドラインから最小ユーザー インターフェイスを使って実行する 1416
コマンドラインからアップデートのダウンロードを行わずにスイートのサイレント インストールを実行する 1417
プロジェクトの検証 1417
インストール パッケージまたはマージ モジュールを検証する 1418
ビルド時に検証を行うかどうかを指定する 1419
検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する 1419
インストール パッケージまたはマージ モジュールをオン デマンドで検証する 1420
検証結果を表示する 1420
ICE 1421
ISICE 1421
ISICE01 . . . . . *1424*
ISICE02 . . . . . *1425*
ISICE03 . . . . . *1426*
ISICE04 . . . . . *1426*
ISICE05 . . . . . *1427*
ISICE06 . . . . . *1428*
ISICE07 . . . . . *1429*
ISICE08 . . . . . *1430*
ISICE09 . . . . . *1431*
ISICE10 . . . . . *1431*
ISICE11 . . . . . *1432*
ISICE12 . . . . . *1434*
ISICE16 . . . . . *1434*
ISICE17 . . . . . *1435*
ISICE18 . . . . . *1436*
ISICE19 . . . . . *1437*
ISICE20 . . . . . *1439*
ISICE21 . . . . . *1440*
ISICE22 . . . . . *1441*
ISICE23 . . . . . *1441*
ISICE24 . . . . . *1442*
ISICE25 . . . . . *1443*

ISICE26 . . . . . *1443*
InstallShield 仮想化整合性スイート 1444
ISVICE01 . . . . . *1447*
ISVICE02 . . . . . *1447*
ISVICE03 . . . . . *1448*
ISVICE04 . . . . . *1449*
ISVICE05 . . . . . *1450*
ISVICE06 . . . . . *1450*
ISVICE07 . . . . . *1451*
ISVICE08 . . . . . *1452*
ISVICE09 . . . . . *1452*
ISVICE10 . . . . . *1453*
ISVICE11 . . . . . *1454*
ISVICE12 . . . . . *1454*
ISVICE13 . . . . . *1455*
ISVICE14 . . . . . *1456*
ISVICE15 . . . . . *1457*
ISVICE16 . . . . . *1457*
ISVICE17 . . . . . *1458*
ISVICE18 . . . . . *1459*
ISVICE19 . . . . . *1459*
InstallShield MSIX/UWP アプリ適合性スイート 1460
ISUWP01 . . . . . *1462*
ISUWP02 . . . . . *1462*
ISUWP03 . . . . . *1463*
ISUWP04 . . . . . *1463*
ISUWP05 . . . . . *1463*
ISUWP06 . . . . . *1464*
ISUWP07 . . . . . *1464*
ISUWP08 . . . . . *1465*
ISUWP09 . . . . . *1465*
ISUWP10 . . . . . *1466*
ISUWP11 . . . . . *1466*
ISUWP12 . . . . . *1467*
ISUWP13 . . . . . *1467*
ISUWP14 . . . . . *1468*
ISUWP15 . . . . . *1468*
ISUWP16 . . . . . *1468*
ISUWP17 . . . . . *1469*
ISUWP18 . . . . . *1469*
ISUWP19 . . . . . *1470*
ISUWP20 . . . . . *1470*
InstallShield ベスト プラクティス スイート 1470
ISBP01 . . . . . *1472*
ISBP02 . . . . . *1473*
ISBP03 . . . . . *1473*

ISBP04 1474
ISBP05 1475
ISBP06 1476
ISBP07 1477
ISBP08 1477
ISBP09 1478
ISBP10 1479
ISBP11 1480
ISBP12 1480
ISBP13 1481
ISBP14 1482
ISBP15 1483
ISBP16 1483
ISBP17 1484
ISBP18 1485
ISBP19 1485
ISBP20 1486
インストールまたはアンインストールがターゲット システムを再開するタイミングについて 1487
Windows Installer ベースのインストールのデバッグ 1488
MSI デバッガーを開始する 1488
MSI デバッガーでブレークポイントを設定する 1489
ブレークポイントを削除する 1490
MSI デバッガーでプロパティの表示と設定を行う 1490
MSI デバッガーでアクションにステップインする 1491
MSI デバッガーでアクションを順番に実行する 1491
MSI デバッガーを停止する 1492
インストールを複数のディスクまたは CD に分割する 1492
複数ディスク インストールを圧縮する 1493
ディスク分割規則 1493
カスタム ディスク分割 1494
セットアップランチャーの作成 1495
セットアップランチャーのファイルのプロパティをカスタマイズする 1497
セットアップランチャーのファイルのアイコンを指定する 1504
インストールを配布する 1505
フォルダーまたは FTP サイトにリリースを自動的に配布する 1505
ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する 1506
インターネットで配布するインストールを準備する 1508
InstallScript インストールと共に配布される再配布可能ファイル 1509
64 ビット セットアップ ランチャーのビルド 1510
Windows アプリ パッケージの配布 1512
トランスフォームの作成 1515
2 つの .msi パッケージを比較してトランスフォームを作成する 1516
単一の .msi パッケージを基にトランスフォームを作成する 1516
トランスフォームの適用 1517
トランスフォームの編集 1518
既存のトランスフォーム ファイルを InstallShield トランスフォーム ウィザードで開く 1518

.msi ファイルをトランスフォームとして保存する 1519
トランスフォームを .msi ファイルとして保存する 1519
応答トランスフォーム 1520
応答トランスフォームの作成 1520
応答トランスフォームの変更 1521
応答トランスフォームのテスト 1521
サーバーの場所の構成 1521
サーバーの場所の追加 1522
サーバーの場所の変更 1522
サーバーの場所の削除 1523
サーバーの場所の順番を変更する 1523

5 アドバンスト UI およびスイート/アドバンスト UI インストールの作成 1525

アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い 1527

アドバンスト UI またはスイート/アドバンスト UI インストールで機能、パッケージ、前提条件、およびファイルを編成する 1530

パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン 1532
.msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する 1534
InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する 1536
アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮 1537
スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する 1539
実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する 1540
Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する 1541
サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する 1542
サイドローディング Windows アプリ バンドル (.appxbundle | .msixbundle) をスイート/アドバンスト UI プロジェクトに追加する 1544
Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する 1545
アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い 1546
アドバンスト UI またはスイート/アドバンスト UI プロジェクトでダイナミック リンクを作成する 1547
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する 1549
InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める 1550
アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い 1552
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、パッケージとトランザクションのインストール順序を指定する 1553
アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する 1553
アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける 1554
異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する 1555
アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する 1557
アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージ操作を構成する 1558
アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する 1560

スイート/アドバンスト UI インストールの動作をカスタマイズする 1561

スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する 1561

スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する 1564
スイート/アドバンスト UI インストールに含まれるアクションの .exe ファイルでの作業について 1565
スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について 1566
スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について 1570
スイート/アドバンスト UI インストールに含まれるプロパティを設定するアクションでの作業について 1572
スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について 1573
スイート/アドバンスト UI インストールでマネージ コード アクションを使用する 1575
スイート/アドバンスト UI インストールで .NET Core アクションを使用する 1578
アクションをスイート/アドバンスト UI プロジェクトに追加する 1582
スイート/アドバンスト UI アクションの設定を構成する 1582
スイート/アドバンスト UI アクションをスケジュールする 1583
スイート/アドバンスト UI インストールにおけるイベントの種類 1585
アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する 1587
アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする 1589
アドバンスト UI またはスイート/アドバンスト UI インストールの [プログラムの追加または削除] エントリ 1590
コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す 1591
アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する 1594
アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスを定義する 1596
アドバンスト UI およびスイート/アドバンスト UI インストールでハイコントラストモードをサポート 1597
アドバンスト UI またはスイート/アドバンスト UI プロジェクトの UI で機能ステータスおよびその他の機能データを参照する 1598
アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する 1600
アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する 1602
アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える 1603
アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動 1605
再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する 1606
アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート 1609
[新しいアップデート利用可能] ダイアログのカスタマイズ 1615
Windows タスク スケジューラでのアプリケーション更新スケジュール タスクの管理 1616
アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する 1617
アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング 1618
アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート 1623
64 ビット セットアップ EXE のビルド 1625

6 InstallShield 前提条件およびその他の再配布可能ファイルをデザインする 1629

InstallShield 前提条件を定義する 1633
InstallShield 前提条件を作成する 1634
既存の InstallShield 前提条件を変更する 1634
InstallShield 前提条件のプロパティを設定する 1634
nstallShield Windows Package Manager 前提条件の作成 1635

.prq ファイルの代替 URL を指定する 1635
InstallShield 前提条件のインストール条件を設定する 1636
InstallShield 前提条件の新しいインストール条件を追加する 1637
InstallShield 前提条件のオペレーティング システム条件を追加する 1637
InstallShield 前提条件のインストール条件を変更する 1638
InstallShield 前提条件からインストール条件を削除する 1639
InstallShield 前提条件のファイルを指定する 1639
InstallShield 前提条件をダウンロードする URL を指定する 1640
InstallShield 前提条件をインストールするためのパラメーターを指定する 1641
InstallShield 前提条件のコマンドライン パラメーターを指定する 1642
InstallShield 前提条件のインストールでターゲットマシンを再起動する 1645
InstallShield 前提条件のインストール動作を指定する 1645
InstallShield 前提条件に管理者権限が必要であることを指定する 1646
InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする 1646
ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する 1646
実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する 1648
InstallShield 前提条件のインストールに伴う可能性のある問題の対処方法について 1649
再起動を必要とする InstallShield 前提条件の動作を指定する 1649
InstallShield 前提条件の依存関係を指定する 1649
InstallShield 前提条件の依存関係を追加する 1650
InstallShield 前提条件の依存関係を変更する 1650
InstallShield 前提条件の依存関係を削除する 1650
InstallShield 前提条件を保存する 1650
マージ モジュールのデザイン 1653
マージ モジュール プロジェクトを作成する 1654
マージ モジュールのデフォルトのインストール先フォルダーを指定する 1654
マージ モジュール言語を選択する 1655
マージ モジュールに依存関係を追加する 1655
マージ モジュールへ除外を追加する 1656
マージ モジュールをマージ モジュール プロジェクト内からリポジトリにパブリッシュする 1657
InstallScript オブジェクトを作成する 1659
オブジェクトの設計 1660
オブジェクト用機能の設計 1662
オブジェクト ファイルの管理 1663
オブジェクトのウィザードの設計 1663
カスタム オブジェクト ウィザード 1664
オブジェクトのデザインタイム環境のローカライズ 1665
インターナショナル化のためのオブジェクトの作成 1666
実行時に、オブジェクトとインストールとの互換性を持たせる 1667
プロパティとメソッド 1668
プロパティの作成 1668
構造評価プロパティの作成 1670
オブジェクトとインストールの間で文字列配列を渡す 1672
オブジェクトのプロパティにアクセスする 1673
メソッドの作成 1674

メソッドを使用する *1675*
オブジェクトステータスのプロパティ *1675*
オブジェクトの実行時のユーザー インターフェイスを作成する 1679
オブジェクトでサポートされていない機能 1680
オブジェクトのビルド 1681
ユーザー インターフェイスからオブジェクトをビルドする 1681
コマンドラインからオブジェクトのコンパイルとビルドを行う 1682
オブジェクトのテストとデバッグ 1682
オブジェクトの配布 1684
オブジェクトのスクリプト 1685
オブジェクト プロジェクトでサポートされていない関数 1685
オブジェクト プロジェクトでサポートされていない定数 1685
オブジェクト プロジェクトでのみサポートされている関数 1686
ScriptDefinedVar プロパティをオブジェクトに追加する 1686
オブジェクトでシステム変数を使用する 1688
スクリプトを使用してオブジェクトのインストール先を設定する 1688

7 開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化 1689

DIM のインストール情報を指定する 1690
プロジェクト ファイルを XML またはバイナリ形式で保存する 1690
DIM のデフォルトのインストール先フォルダーを指定する 1690
DIM でパス変数を使用する 1691
DIM のファイルを編成する 1695
DIM のターゲット システムの構成 1695
DIM のインストール動作をカスタマイズする 1696
DIM のサーバーを構成する 1696

8 アプリケーションのアップデート 1697

アップグレードの概要 1697
メジャーアップグレード 1699
マイナーアップグレード 1700
スモール アップデート 1700
パッチの適用 1701
QuickPatch プロジェクト 1701
差分リリース 1702
完全リリース 1703
最適なアップグレード ソリューションの決め方 1703
メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い 1706
自動アップグレード 1709
パッチと QuickPatch プロジェクトの違い 1709
アップグレードのパッケージ オプション 1712
差分リリースと完全リリース 1713
アップグレード、パッチ、および QuickPatch プロジェクトを使用する 1714
ファイルの上書き規則について 1715
アップグレード関する考慮事項 1715

InstallShield を構成してアップグレードの種類を自動判別する 1717
自動アップグレード アイテムの追加 1717
自動アップグレード アイテムの設定を構成する 1717
自動アップグレードからメジャーまたはマイナーアップグレードへ変換する 1718
自動アップグレード アイテムを削除する 1718
メジャー アップグレードを作成する 1718
メジャー アップグレード アイテムを追加する 1719
メジャー アップグレードの設定を構成する 1719
メジャー アップグレード アイテムを削除する 1720
マイナー アップグレードを作成する 1720
マイナー アップグレード アイテムを追加する 1721
マイナー アップグレードの設定を構成する 1721
マイナー アップグレードを構成してインストール済みのデータを削除する 1722
マイナー アップグレード アイテムを削除する 1723
マイナー アップグレードの実行時の動作 1723
スモール アップデートを作成する 1724
パッチ時の考慮事項 1725
アップデート起動ツールのファイルのプロパティをカスタマイズする 1727
アップデート起動ツールのアイコンを指定する 1728
パッチ シーケンス 1729
パッチのアンインストール 1732
非管理者パッチ 1733
パッチの作成と適用 1734
新しいパッチ構成アイテムを追加する 1735
新しい最新セットアップ アイテムを追加する 1735
新しい以前のセットアップ アイテムを追加する 1735
外部ファイルを追加する 1736
パッチの設定を構成する 1736
パッチの識別情報を指定する 1737
パッチのアンインストールを有効にする 1737
パッチのシーケンス 1738
パッチ パッケージに署名する 1739
パッチ パッケージをパスワードで保護する 1739
パッチに Update.exe アップデート ランチャをビルドをするかどうかを指定する 1740
最新セットアップ アイテムを別のパッチ構成にコピーする 1740
パッチ構成アイテムを削除する 1741
最新セットアップ アイテムを削除する 1741
以前のセットアップ アイテムを削除する 1741
パッチのビルド 1742
パッチの適用 1742
QuickPatch プロジェクトを作成する 1743
既存の QuickPatch をパッチする QuickPatch プロジェクトを作成する 1744
QuickPatch パッケージを簡素化するかどうかを指定する 1744
パッチのターゲット リリースを指定する 1745
QuickPatch を実行するカスタム アクションを指定する 1746
QuickPatch ヘファイルを追加する 1746

QuickPatch の識別情報を指定する 1747
QuickPatch のアンインストールを有効にする 1747
QuickPatch パッケージをシーケンスする 1747
QuickPatch パッケージに署名する 1748
QuickPatch パッケージをパスワードで保護する 1748
QuickPatch パッケージに Update.exe アップデート起動ツールをビルドをするかどうかを指定する 1749
[パッチするファイル] エクスプローラーからファイルを削除する 1750
QuickPatch を使用したインストール済みファイルの変更と削除 1750
QuickPatch を使用したレジストリデータの追加、変更および削除 1750
QuickPatch プロジェクトのファイル形式を指定する: XML またはバイナリ 1751
アップグレード、パッチ、および QuickPatch パッケージを検証する 1751
検証ツール 1752
Val0001 . . . . . *1753*
Val0002 . . . . . *1754*
Val0003 . . . . . *1755*
Val0004 . . . . . *1757*
Val0005 . . . . . *1758*
Val0006 . . . . . *1759*
Val0007 . . . . . *1760*
Val0008 . . . . . *1760*
Val0009 . . . . . *1762*
Val0011 . . . . . *1763*
Val0012 . . . . . *1764*
Val0013 . . . . . *1764*
Val0014 . . . . . *1765*
Val0015 . . . . . *1766*
グローバル アセンブリ キャッシュのアセンブリをパッチする 1768
差分リリースと完全リリース 1769
以前のバージョンをアップデートする InstallScript リリースを作成する 1769

9 追加のインストール オプション 1773

複数言語インストールの作成 1775
グローバリゼーションのヒント 1776
デフォルトのプロジェクト言語の設定 1777
言語の設定 1778
インストール言語の選択 1781
コンポーネントを言語依存としてマークする 1782
言語基づいてコンポーネントをインストールする 1783
リリースへの言語の組み込み 1784
文字列エントリの翻訳 1785
文字列エントリのエクスポートとインポート 1785
各言語ごとのダイアログを変更する 1787
インストールがユーザー インターフェイスで使用する言語を判別する方法 1789
言語サポートのカスタマイズ 1790
言語識別子 1790

製品の複数のインスタンスをインストールする 1795

複数インスタンス サポートの実行時要件 1795
InstallShield で複数インスタンスを構成する 1796
製品構成にインスタンスを追加する 1797
インスタンスの名前付け 1797
1 つ以上のインスタンスの製品コードを更新する 1798
インスタンスのプロパティを設定します 1799
製品構成からインスタンスを削除する 1800
複数インスタンス サポートに関する特別考慮 1800
複数インスタンスのサポートを含むリリースの構成とビルド 1801
複数インスタンスのサポートを含むメジャー アップグレードの構成とビルド 1802
製品の複数インスタンスをインストールするためのサポート 1803
製品のインスタンスを複数インストールするときの実行時の動作 1804
ターゲット システムの条件を検出する 1807
条件ステートメントのビルド 1807
条件ステートメントの構文 1810
管理者権限と昇格された権限を検出する 1813
初回インストール、メンテナンス モード、およびアンインストールを検出する 1814
初回インストール時にのみ動作をトリガーする 1815
エンド ユーザーが特定の機能を選択したかどうかを検出する 1815
エンド ユーザーが特定のオペレーティング システムを実行しているかどうか検出する 1816
インストールが仮想マシン上で実行されているかどうかを検出する 1817
アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド 1820
アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類 1823
"プラットフォーム" 条件の設定 1827
"ファイルの存在" 条件の設定 1830
"ファイルの比較" 条件の設定 1832
"レジストリの存在" 条件の設定 1836
"レジストリの比較" 条件の設定 1837
"プロパティの比較" 条件の設定 1840
"MSI パッケージ" 条件の設定 1843
"MSI アップグレード" 条件の設定 1845
"対象パッケージ" 条件の設定 1847
"InstallScript パッケージ" 条件の設定 1848
"ロケール" 条件の設定 1850
"インストール済み" 条件の設定 1851
「Windows アプリ パッケージの条件" の設定 1854
"UWP タイプの存在" 条件の設定 1855
"パッケージの操作" 条件の設定 1856
"機能の操作" 条件の設定 1857
"拡張条件" の設定 1858
"モード条件" の設定 1859
アドバンスト UI またはスイート/アドバンスト UI プロジェクトで条件を定義するときのガイドライン 1860
特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する 1862
アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する 1863
アドバンスト UI またはスイート/アドバンスト UI インストールのカスタム条件を作成する 1865

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する *1866*
アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する *1869*
インストールされたデータの検索 **1871**
定義済みシステム検索の追加 1871
システム検索の追加 1872
システム検索の変更 1873
実行時にレジストリを検索する場合の特別考慮 1873
システム検索をリポジトリにパブリッシュする 1875
システム検索を削除する 1876
XML データの検索 1876
インストールおよびプロジェクトのテーブルを編集する **1879**
プロジェクト テーブルを編集する 1880
ダイレクト エディターで新しいテーブルを追加する 1881
テーブルへレコードを追加する 1881
ダイレクト エディターでの検索と置換 1882
テーブル レコードを編集する 1883
テーブルのフィールド編集 1884
ダイレクト エディターからテーブルをエクスポートする 1885
ダイレクト エディターを使ってテーブルをインポートする 1885
テーブルからレコードを削除する 1886
ダイレクト エディターでテーブルを削除する 1887
Windows Installer データベースとプロジェクト ファイルのインプレース差分比較 1887
テーブルへバイナリデータを入力する 1889
親のないディレクトリ エントリ 1889
InstallShield ツールの使用 **1891**
InstallShield MSI ツール 1891
InstallShield MSI Diff 1892
トランスフォーム (.mst ファイル) によって行われる変更を判別する *1892*
パッチが .msi パッケージに対して有効であるかどうかを判別する *1893*
パッチによる実行時の影響を判別する *1893*
2 つの .msi パッケージ間の違いを判別する *1894*
InstallShield MSI Diff をコマンドラインから実行して、差分のログ ファイルを生成する *1894*
2 つの InstallShieldプロジェクト間の違いを判別する *1894*
InstallShield MSI Query 1895
特定の Windows Installer ベースに対して SQL Query を実行する *1895*
InstallShield MSI Sleuth 1896
ターゲット システムで実行された .msi パッケージについての詳細を表示する *1896*
特定のコンポーネントを含むすべてのインストール済みのパッケージを検索する *1896*
InstallShield MSI Grep 1897
.msi パッケージで特定の文字列を検索する *1897*
InstallShield キャビネット & ログ ファイル ビューアー 1898
InstallScript キャビネット ファイル (.cab) およびヘッダー ファイル (.hdr) を表示する 1898
InstallScript .cab および .hdr ファイルの概要 *1899*
InstallScript .cab および .hdr ファイルを開く *1900*
InstallScript .cab または .hdr ファイルからファイルを抽出する *1901*
InstallScript .cab または .hdr ファイルからファイルについてのレポートを生成する *1901*

InstallScript ログ ファイル (.ilg) を参照する 1902
InstallScript ログ ファイルの概要 *1902*
InstallScript ログ ファイルを開く *1903*
InstallScript ログ ファイルを検索する *1904*
InstallScript ログ ファイルをテキスト ファイルに変換する *1904*
InstallShield MSIX バンドル ユーティリティ 1904
Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方 1907
Windows Installer プロパティの概要 1908
Windows Installer プロパティ リファレンス 1909
SETUPEXEDIR 1934
アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス 1935
Windows Installer ベースのプロジェクトにおけるプロパティの作成 1942
アドバンスト UI およびスイート/アドバンスト UI プロジェクトでプロパティを作成する 1944
既存プロパティを変更する 1945
ローカライズ可能なプロパティを作成する 1945
既存プロパティをローカライズ可能にする 1946
プロパティ値が Windows Installer ログ ファイルに記録されないように防ぐ 1947
アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ 1947
パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する 1948
InstallScript で Windows Installer プロパティを取得または設定する方法 1949
プロパティから値を削除する 1949
プロパティを削除する 1950
.msi および .msm データベースを直接編集する 1953
Windows Installer パッケージを開く 1953
ダイレクト編集モードで .msi/.msm データベースを編集する 1954
ダイレクト編集モードでファイルを追加する 1954
ダイレクト編集モードでマージ モジュールを追加する 1955

10 InstallShield と外部アプリケーションの統合 1957
ソース コード管理を使用する 1957
ソース コード管理統合を使用する 1958
プロジェクトをソース管理へ追加する 1958
ソース管理からプロジェクトをチェックアウトする 1959
ソース管理にプロジェクトをチェックインする 1960
ソース管理からのプロジェクトのリンクの解除 1960
Microsoft Visual Studio との統合 1960
Microsoft Visual Studio で InstallShield プロジェクトを作成する 1961
Microsoft Visual Studio で InstallShield プロジェクトを開く 1962
Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する 1962
Visual Studio ソリューションにリファレンスを追加する 1963
ツールボックスを使ってダイアログ コントロールを追加する 1963
InstallShield ツールバーまたはコマンドを Visual Studio ツールバーに追加する 1964
Microsoft Visual Studio でリリースをビルドする 1965
.NET アセンブリをインストール プロジェクトに追加する 1967
Web サービスまたは Web アプリケーション プロジェクトからプロジェクト出力を追加する 1968

Microsoft Visual Studio Team Foundation Server との統合 1968
InstallShield プロジェクトを Team Explorer に追加する 1969

11 インストールの開発およびビルド プロセスの自動化 1971
VBScript を使用したサブフォルダーの機能へのリンク 1971
オートメーション インターフェイスを使用した文字列エントリのインポートとエクスポート 1974
オートメーション インターフェイスを使用したプロジェクト レポートの実行 1976
コマンドラインでソース コード管理を使用する 1978
ビルドのステータス イベント 1980
64 ビット システム上でオートメーション インターフェイスを使用する 1981

12 InstallShield プロジェクトを DevOps でビルドする 1983
Azure DevOps ビルド拡張 1983
InstallShield StandAlone Build と Docker 1984

13 リファレンス 1985
メニュー、ツールバー、および ウィンドウのリファレンス 1987
メニュー 1987
[ファイル] メニュー 1988
[ホーム] メニュー 1989
[編集] メニュー 1991
[表示] メニュー 1993
[移動] メニュー 1994
[プロジェクト] メニュー 1997
[ビルド] メニュー 2000
[ツール] メニュー 2005
[ウィンドウ] メニュー 2008
[ヘルプ] メニュー 2009
ツールバー 2010
標準ツールバー 2011
[コントロール] ツールバー 2014
レイアウト ツールバー 2016
[MSI デバッグ] ツールバー 2018
出力ウィンドウ 2018
ダイアログ ボックス リファレンス 2021
[.NET インストーラー クラス引数] ダイアログ ボックス 2025
[引数/値ペアの追加] ダイアログ ボックス 2025
[既存のメディアを追加] ダイアログ ボックス 2026
[このパッケージのファイルを追加する] ダイアログ ボックス 2026
[MIME の種類を追加] ダイアログ ボックス 2028
[新しいメソッドの追加] ダイアログ ボックス 2029
[新しいプロパティの追加] ダイアログ ボックス 2029
[定義済み検索の追加] ダイアログ ボックス 2030
[ソース管理に追加] ダイアログ ボックス 2031

[アプリケーション拡張子マッピング] ダイアログ ボックス 2031
[アプリケーションのマッピング] ダイアログ ボックス 2032
[コンポーネントの関連付け] ダイアログ ボックス 2033
[DIM の関連付け] ダイアログ ボックス 2033
[バッチ ビルド] ダイアログ ボックス 2033
[ディレクトリの参照] ダイアログ ボックス 2034
ディレクトリ/ショートカットのターゲットを参照するダイアログ ボックス 2035
[ファイルの参照] ダイアログ ボックス 2037
[実行するファイルの参照] ダイアログ ボックス 2038
[ショートカット アイコンの参照] ダイアログ ボックス 2039
[タイトル ターゲットの参照] ダイアログ ボックス 2040
[証明書の選択] ダイアログ ボックス 2040
[チェックイン] ダイアログ ボックス 2043
[チェックアウト] ダイアログ ボックス 2043
[コンポーネントのプロパティ] ダイアログ ボックス 2043
[全般] タブ 2044
[ファイルのリンク] タブ 2046
[機能] タブ 2048
[条件ビルダー] ダイアログ ボックス 2048
[コンテンツ ソース パス] ダイアログ ボックス 2049
[InstallScript MSI の変換] ダイアログ ボックス 2050
[新しいコンポーネントの作成] ダイアログ ボックス 2052
[新しい機能の作成] ダイアログ ボックス 2052
[カスタム エラー処理] ダイアログ ボックス 2053
[カスタム エラー] ダイアログ ボックス 2053
[検証設定のカスタマイズ] ダイアログ ボックス 2054
[データ言語] ダイアログ ボックス 2054
[プロパティの削除] ダイアログ ボックス 2054
[依存関係] ダイアログ ボックス 2055
[ダイアログのイメージ] ダイアログ ボックス 2055
[セットアップのデジタル署名] ダイアログ ボックス 2056
[差分] ダイアログ ボックス 2058
[ダイナミック ファイル リンクの設定] ダイアログ ボックス 2059
[データの編集] ダイアログ ボックス 2061
[IIS Metabase 値の編集] ダイアログ ボックス 2061
[オプション リストの編集] ダイアログ ボックス 2062
[レジストリ データの編集] ダイアログ ボックス 2063
[仮想マシンの構成を編集] ダイアログ ボックス 2064
[エラー マッピングのプロパティ] ダイアログ ボックス 2067
[言語の除外] ダイアログ ボックス 2067
[テーブルのエクスポート] ダイアログ ボックス 2068
[機能条件ビルダー] ダイアログ ボックス 2068
[ファイルの詳細] ダイアログ ボックス 2070
[ファイルのプロパティ] ダイアログ ボックス 2070
[ファイルのプロパティ] ダイアログ ボックス (InstallScript インストール ロジェクト) 2074
ファイル 削除の [プロパティ] ダイアログ ボックス 2075

[検索/置換] ダイアログ ボックス 2077
[プロジェクト内の文字列 ID を検索する] ダイアログ ボックス 2079
[フォルダーのプロパティ] ダイアログ ボックス 2079
[全般] タブ 2080
[共有] タブ 2080
[関数シグネチャ] ダイアログ ボックス 2081
[一般オプション – 詳細] ダイアログ ボックス 2084
[一般オプション – 別のディスク ファイル] ダイアログ ボックス 2086
[履歴] ダイアログ ボックス 2086
[ダイアログのインポート] ダイアログ ボックス 2086
[InstallScript ファイルのインポート] ダイアログ ボックス 2087
[SQL スクリプト ファイルのインポート] ダイアログ ボックス 2087
[アクションの挿入] ダイアログ ボックス 2088
[InstallShield前提条件のプロパティ] ダイアログ ボックス 2090
[言語] ダイアログ ボックス 2092
[リンク タイプ] ダイアログ ボックス 2092
[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックス 2094
小文字のコンポーネントディレクトリ – 警告 2095
[メディア ファイルのプロパティ] ダイアログ ボックス 2095
[マージ モジュールの構成可能な値] ダイアログ ボックス 2096
[マージ モジュールのプロパティ] ダイアログ ボックス 2096
[メディア] タブ 2097
[インストール先] タブ 2097
[メソッドの参照] ダイアログ ボックス 2098
[メソッド シグネチャ] ダイアログ ボックス 2098
[MIME の種類] ダイアログ ボックス 2100
[ダイナミック リンクの変更] ダイアログ ボックス 2101
[プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル 2102
[モジュール依存関係] ダイアログ ボックス 2104
[除外モジュール] ダイアログ ボックス 2105
[MSI 値] ダイアログ ボックス 2105
[MST SIS の設定] ダイアログ ボックス 2105
[複数行文字列値] ダイアログ ボックス 2107
[新規依存関係] ダイアログ ボックス 2109
[新規ファイル] ダイアログ ボックス 2109
[新規プロジェクト] ダイアログ ボックス 2109
[オペレーティング システム] ダイアログ ボックス 2112
[オプション] ダイアログ ボックス 2112
[全般] タブ 2113
[ファイルの場所] タブ 2114
[パス変数] タブ 2115
[ユーザー インターフェイス] タブ 2115
[プリファレンス] タブ 2117
[品質] タブ 2118
[.NET] タブ 2119
[ファイル ビュー] タブ 2120

[ファイルの拡張子] タブ 2122
[前提条件] タブ 2124
[ソース管理] タブ 2124
[ディレクトリ] タブ 2126
[リソース] タブ 2126
[検証] タブ 2127
[リポジトリ] タブ 2128
[SQL スクリプト] タブ 2128
[マージ モジュール] タブ 2130
その他の IIS プロパティ ダイアログボックス 2131
[その他のウィンドウ スタイル] ダイアログ ボックス 2131
ダイアログのその他のウィンドウ スタイル 2132
ビットマップ、アイコン、およびテキスト領域コントロールのその他のウィンドウ スタイル 2134
ボタンコントロールのその他のウィンドウスタイル 2135
コンボ ボックスコントロールのその他のウィンドウ スタイル 2137
リスト ビュー コントロールのその他のウィンドウ スタイル 2139
編集フィールドとリスト ボックス コントロールのその他のウィンドウ スタイル 2141
行コントロールのその他のウィンドウ スタイル 2143
進行状況バー コントロールのその他のウィンドウ スタイル 2145
選択ツリー コントロールのその他のウィンドウスタイル 2146
[出力] ダイアログ ボックス 2147
[上書き] ダイアログ ボックス 2148
[パッチ シーケンス] ダイアログ ボックス 2150
[パス変数のオーバーライド] ダイアログ ボックス 2150
[パス変数の推奨] ダイアログ ボックス 2151
ファイルとディレクトリの [アクセス許可] ダイアログ ボックス 2152
レジストリ キーの [アクセス許可] ダイアログ ボックス 2155
[プラットフォーム スイート] ダイアログ ボックス 2158
[プラットフォーム] ダイアログ ボックス 2158
[ビルド後のイベント] ダイアログ ボックス 2161
[ビルド前のイベント] ダイアログ ボックス 2162
[圧縮前のイベント] ダイアログ ボックス 2162
[ファイルの準備イベント] ダイアログ ボックス 2163
[前提条件設定] ダイアログ ボックス 2163
[特権] ダイアログ ボックス 2172
[製品条件ビルダー] ダイアログ ボックス 2173
[プロジェクト設定] ダイアログ ボックス 2175
[プロジェクト] タブ 2175
[アプリケーション] タブ (Windows Installer ベースのプロジェクト) 2175
[アプリケーション] タブ (InstallScript プロジェクト) 2177
[プラットフォーム] タブ 2179
[言語] タブ 2180
[メンテナンス] タブ 2180
[言語非依存オブジェクトのプロパティ] タブ 2181
[言語固有オブジェクトのプロパティ] タブ 2182
[キー名の変更] ダイアログ ボックス 2182

[必要な機能] ダイアログ ボックス 2183
[競合の解決] ダイアログ ボックス 2183
[名前を付けて保存] ダイアログ ボックス 2184
[スクリプト変換] ダイアログ ボックス 2185
[スクリプト エディターのプロパティ] ダイアログ ボックス 2186
[SCM シャットダウンの遅延] ダイアログ ボックス 2190
[セキュリティ記述子] ダイアログ ボックス 2190
[ファイルの選択] ダイアログ ボックス 2191
[メディアの場所選択] ダイアログ ボックス 2191
[SQL Server の選択] ダイアログ ボックス 2191
[文字列の選択] ダイアログ ボックス 2192
シリアル番号違反ダイアログ ボックス 2193
[サーバーの場所] ダイアログ ボックス 2194
自動開始サービスの遅延に関する [サービスのオプション] ダイアログ ボックス 2195
回復操作の実行のための [サービスのオプション] ダイアログ ボックス 2195
[サービス SID] ダイアログ ボックス 2196
[ファイルの URL の設定] ダイアログ ボックス 2197
[設定] ダイアログ ボックス 2197
[コンパイル/リンク] タブ 2198
[実行/デバッグ] タブ 2202
[MSI ログファイル] タブ 2203
[セットアップ言語] ダイアログ ボックス 2203
[SQL Server 要件] ダイアログ ボックス 2205
[優先] ダイアログ ボックス 2205
[UI 言語] ダイアログ ボックス 2206
[マージ モジュール検索パスのアップデート] ダイアログ ボックス 2206
ウィザード リファレンス 2207
コンポーネント ウィザード 2208
[ようこそ] パネル 2209
セットアップ ベスト プラクティス 2210
[インストール先] パネル 2211
[ファイル] パネル 2212
[概要] パネル 2213
[コンポーネント タイプ] パネル 2213
COM サーバー コンポーネントの種類 2214
[COM サーバー ファイル] パネル 2215
[クラス] パネル 2217
[コンテキストの種類] パネル 2219
[タイプ ライブラリ] パネル 2219
[AppID 一般情報] パネル 2220
[AppID 詳細情報] パネル 2220
[概要] パネル 2222
サービスのコントロール コンポーネントの種類 2222
[サービスの指定] パネル 2223
[インストール イベント] パネル 2224
[アンインストール イベント] パネル 2225
[待機のタイプ] パネル 2226
[概要] パネル 2226

サービスのインストール コンポーネントの種類 *2227*
[サービス実行可能ファイル] パネル 2228
[サービスの種類の情報] パネル 2229
[サービス開始タイプ情報] パネル 2230
[サービスのロード順] パネル 2231
[エラー コントロール] パネル 2231
[サービス ログオン] パネル 2232
[概要] パネル 2233
フォント コンポーネント タイプ *2233*
[インストールされているフォントの追加] パネル 2234
[新規フォントの追加] パネル 2235
[概要] パネル 2235
コンポーネント ウィザード [一般概要] パネル 2236
ソース パス変換ウィザード 2237
[ようこそ] パネル 2237
[検索] パネル 2237
[検索結果と推奨] パネル 2237
[プロジェクトのアップデート] パネル 2238
[アップデート完了] パネル 2238
新規 QuickPatch 作成ウィザード 2238
[ようこそ] パネル 2239
[QuickPatch プロジェクト ベース] パネル 2239
[元ののセットアップ パッケージ] パネル 2240
[既存の QuickPatch の場所] パネル 2241
テーブル作成ウィザード 2242
[ようこそ] パネル 2242
[列のプロパティ] パネル 2243
[概要] パネル 2244
カスタム アクション ウィザード 2244
[ようこそ] パネル 2246
[基本情報] パネル 2246
[アクションの種類] パネル 2247
[関数定義] パネル 2253
[アクション パラメーター] パネル 2256
[マネージ メソッドの定義] パネル 2261
[シーケンス内スクリプト] パネル 2263
[追加のオプション] パネル 2263
[応答オプション] パネル 2266
[シーケンスに挿入] パネル 2267
[概要] パネル 2268
データベース インポート ウィザード 2268
[ようこそ] パネル 2269
SQL Server パネル 2269
[SQL データベース] パネル 2269
[データベース テーブル] パネル 2270
[スクリプト化するオブジェクト] パネル 2270
[スクリプト作成オプション] パネル 2271
[スクリプト作成詳細オプション] パネル 2273

[概要] パネル 2274
デバイス ドライバー ウィザード 2275
[ようこそ] パネル 2275
[デバイス ドライバー パッケージ] パネル 2276
[デバイス ドライバー ファイル] パネル 2276
[デバイス ドライバー情報] パネル 2276
デバイス ドライバー タイプ *2277*
プロジェクト全体の [デバイスドライバー情報] パネル 2278
[概要] パネル 2279
ダイアログ ウィザード 2279
[ようこそ] パネル 2279
[ダイアログ テンプレート] パネル 2280
[ユーザー インターフェイス シーケンス] パネル 2282
[ダイアログの位置と条件] パネル 2283
DirectX オブジェクト ウィザード 2283
[ようこそ] パネル 2284
[オブジェクトの設定] パネル 2284
[概要] パネル 2285
ダイナミック スキャン ウィザード 2285
[ようこそ] パネル 2286
[ファイルのフィルター] パネル 2286
[実行可能ファイルの指定] パネル 2287
[アプリケーション ファイルの指定] パネル 2287
アプリケーションの起動 2288
[アプリケーションが実行中です] パネル 2288
[ファイル選択] パネル 2289
[スキャン結果] パネル 2289
[ダイナミック スキャン ウィザード完了] パネル 2290
コンポーネントのエクスポート ウィザード 2290
[ようこそ] パネル 2291
[コンポーネントの選択] パネル 2291
[ターゲット ファイルの選択] パネル 2292
[ターゲット ファイル情報] パネル 2292
[概要] パネル 2293
関数ウィザード 2293
[関数名] パネル 2293
[関数のパラメーター] パネル 2294
DIM のインポート ウィザード 2294
[ようこそ] パネル 2295
[ソース ファイルの選択] パネル 2295
[インポート] パネル 2295
REG ファイルのインポート ウィザード 2296
[ようこそ] パネル 2296
[レジストリ ファイルのインポート] パネル 2297
[競合オプションのインポート] パネル 2297
[インポート進行状況] パネル 2297

XML 設定のインポート ウィザード 2297
[ようこそ] パネル 2298
[XML ファイル] パネル 2299
[XML 要素] パネル 2299
[XML 進行状況] パネル 2299
[XML 結果] パネル 2299
Microsoft .NET Framework オブジェクト ウィザード 2300
[ステップ 1] パネル 2300
[ステップ 2] パネル 2301
MSI からMSIX への変換ウィザード 2302
[ようこそ] パネル 2303
新しい言語ウィザード 2305
[ようこそ] パネル 2305
プロジェクト言語パネル 2306
[プロジェクトファイル] パネル 2306
[概要] パネル 2307
MSI/MSM オープン ウィザード 2308
[ようこそ] パネル 2308
[ファイル の場所] パネル 2309
MSP オープン ウィザード 2309
[ようこそ] パネル 2309
[基本の MSI パッケージ] パネル 2309
トランスフォーム オープン ウィザード 2310
[ようこそ] パネル 2311
[トランスフォーム情報] パネル 2311
[追加トランスフォーム] パネル 2311
[応答トランスフォームの作成] パネル 2312
パブリッシュ ウィザード 2313
[ようこそ] パネル 2314
[リポジトリ] パネル 2314
[パブリッシュ情報] パネル 2314
[概要] パネル 2314
再配布可能ファイル ダウンローダー ウィザード 2314
Reg-Free COM ウィザード 2315
[ようこそ] パネル 2315
[EXE ファイルの選択] パネル 2316
[マニフェスト ファイルの選択] パネル 2316
[ファイルの選択] パネル 2316
[概要] パネル 2316
リリース ウィザード 2317
[ようこそ] パネル 2317
[製品構成] パネル 2318
[リリースの指定] パネル 2318
[フィルターの設定] パネル 2318
[マージ モジュールのオプション]パネル 2319
[セットアップ言 語] パネル 2319

[メディアの種類] パネル 2321
[ディスク分割オプション] パネル 2322
[ディスク分割詳細設定] パネル 2323
[Web タイプ] パネル 2324
[ダウンローダーのオプション]パネル 2325
[Web からインストールする オプション] パネル 2326
One-Click Install パネル 2327
[ローカル マシン] パネル 2328
[リリース構成] パネル 2328
[カスタム圧縮の設定] パネル 2329
[セットアップランチャ] パネル 2330
[InstallShield 前提条件] パネル 2331
[デジタル署名] パネル 2332
[デジタル署名のオプション] パネル 2334
[パスワードと著作権] パネル 2338
[詳細設定] パネル 2339
[リリース設定の概要] パネル 2343
[一般オプション] パネル 2343
[機能] パネル 2345
[メディア レイアウト] パネル 2345
[カスタム メディア レイアウト] パネル 2346
[ユーザー インターフェイス] パネル 2347
[インターネット オプション] パネル 2348
[アップデート] パネル 2349
[オブジェクト差分] ダイアログ ボックス 2351
[ポストビルドのオプション] パネル 2351
セットアップ ベスト プラクティス ウィザード 2353
[ようこそ] パネル 2353
[準拠] パネル 2354
スタティック スキャン ウィザード 2354
[ようこそ] パネル 2355
[ファイルのフィルター] パネル 2355
[スキャンの進行状況] パネル 2355
[ファイル選択] パネル 2356
[スキャン結果] パネル 2357
[スタティック スキャン ウィザードの完了] パネル 2357
システム検索ウィザード 2357
[ようこそ] パネル 2358
検索する対象を指定してくださいパネル 2358
検索方法パネル (システム検索メソッドの定義) 2359
[ファイルの詳細の指定] パネル (ファイル検索オプション) 2361
検索方法パネル (フォルダー検索オプション) 2362
検索方法パネル(特定のフォルダーオプション) 2362
検索方法パネル (レジストリ検索オプション) 2363
検索方法パネル (.ini ファイル検索 オプション) 2364
検索方法パネル (コンポーネント検索オプション) 2365

[XML ファイル] パネル (XML ファイル オプション) で検索するデータを指定します。 . . . 2365
この値の処理方法を指定してくださいパネル . . . 2366
この値の処理方法を指定してくださいパネル . . . 2367
トランスフォーム ウィザード . . . 2368
[ようこそ] パネル . . . 2369
[ファイルの指定] パネル . . . 2369
[検証設定] パネル . . . 2369
[エラー条件の抑制] パネル . . . 2371
[出力ファイル名の指定] パネル . . . 2372
[概要] パネル . . . 2372
[トランスフォーム ウィザードの完了] パネル . . . 2373
アップグレード検証ウィザード . . . 2373
[ようこそ] パネル . . . 2373
[設定] パネル . . . 2374
[概要] パネル . . . 2374
ユーザー インターフェイス ウィザード . . . 2375
[定義済みタスク ページ] パネル . . . 2375
[タスク構成] パネル . . . 2378
Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザード . . . 2379
[ようこそ] パネル . . . 2380
[ソリューション] パネル . . . 2380
Visual Studio デプロイメント プロジェクト インポート ウィザード . . . 2380
[ようこそ] パネル . . . 2381
[プロジェクトファイル] パネル . . . 2381
[オプション] パネル . . . 2381
[概要] パネル . . . 2387
ビュー リファレンス . . . 2389
[インストール情報] ビュー . . . 2389
[一般情報] ビュー . . . 2390
[一般情報] ビューの設定 . . . 2391
[パッケージ情報] ビュー . . . 2435
[一般情報] ビュー . . . 2435
[一般情報] ビューの設定 . . . 2436
[ビジュアル資産] ビュー . . . 2439
パッケージ ロゴの設定 . . . 2439
既定のビジュアル資産の設定 . . . 2440
ユーザー定義の設定 . . . 2445
[アプリ機能] ビュー . . . 2445
デバイスのアプリ機能 . . . 2445
全般アプリ機能 . . . 2446
制限付きアプリ機能 . . . 2449
[宣言] ビュー . . . 2451
[アプリケーションの宣言セット] . . . 2451
パッケージの宣言 . . . 2501
[コンテンツ URI] ビュー . . . 2505
[編成] ビュー . . . 2506

[セットアップのデザイン] ビュー 2508
[機能] ビュー 2509
機能の設定 *2510*
[コンポーネント] ビュー 2524
コンポーネントの設定 *2525*
コンポーネントの詳細設定 *2547*
アプリケーション パスの設定 2549
アセンブリの設定 2549
COM 登録の設定 2551
ファイル タイプの設定 2557
サービスの設定 2559
パブリッシュの設定 2560
デバイス ドライバーの設定 2561
その他のデータの設定 2564
[セットアップの種類] ビュー 2565
[パッケージ] ビュー 2566
共通 *2567*
Features *2600*
ダイナミック リンクの設定 *2601*
[DIM リファレンス] ビュー 2603
[全般] タブ *2603*
[手順] タブ *2604*
[ビルド オプション] タブ *2604*
[機能] タブ *2605*
[シーケンス] タブ *2606*
[アプリケーション データ] ビュー 2606
[ファイルとフォルダー] ビュー 2607
インストール先フォルダー *2610*
[再配布可能ファイル] ビュー 2614
[前提条件] ビュー 2618
[オブジェクト] ビュー 2620
[システム検索] ビュー 2621
[ショートカット] ビュー 2625
ショートカットの設定 *2627*
フォルダーの設定 *2646*
[タイル構成] の設定 *2648*
[MSIX アプリケーション] ビュー 2652
アプリケーションの設定 *2652*
[レジストリ] ビュー 2658
[ODBC リソース] ビュー 2659
ODBC リソースの設定 *2659*
[INI ファイルの変更] ビュー 2662
.ini ファイルの設定 *2663*
.ini ファイルのセクションの設定 *2664*
.ini ファイルのキーワードの設定 *2664*
[環境変数] ビュー 2665
環境変数の設定 *2666*
[XML ファイルの変更] ビュー 2668

XML ファイルの [全般] タブ *2669*
XML ファイルの [詳細] タブ *2670*
XML ファイルの [名前空間] タブ *2672*
XML 要素の [全般] タブ *2672*
XML 要素の [詳細] タブ *2674*
[テキスト ファイルの変更] ビュー 2675
変更セットの設定 *2675*
変更の設定 *2677*
[スケジュール タスク] ビュー 2681
スケジュール タスクの設定 *2681*
[サービス] ビュー 2685
[サービス] ビューの設定 *2686*
[サーバー構成] ビュー 2700
[IIS 構成] ビュー 2701
Web サイトの設定 *2702*
"アプリケーション" と "仮想ディレクトリ" の設定 *2711*
アプリケーション プールの設定 *2720*
Web サービス拡張の設定 *2726*
[コンポーネント サービス] ビュー 2727
[インストール] タブ *2728*
[SQL スクリプト] ビュー 2731
[全般] タブ *2732*
[要件] タブ *2735*
[詳細] タブ *2737*
SQL スクリプト レベル *2740*
[全般] タブ 2741
[スクリプト] タブ 2742
[ランタイム] タブ 2742
[データベースのインポート] タブ 2745
[テキスト置換] タブ 2745
[動作とロジック] ビュー 2746
[InstallScript] ビュー 2749
[InstallScript] ビューのスクリプト エディター *2750*
[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) 2751
カスタム アクションの種類 *2754*
カスタム アクションの設定 *2758*
アクション テキストの設定 *2773*
[サポート ファイル] ビュー (アドバンスト UI、基本の MSI、InstallScript オブジェクト、およびスイート/アドバンスト UI プロジェクト) 2775
[サポート ファイル/ビルボード] ビュー (InstallScript および InstallScript MSI プロジェクト) 2777
[システム検索] ビュー 2778
[プロパティ マネージャー] ビュー 2779
[イベント] ビュー 2782
[イベント] ビューの設定 *2783*
[ユーザー インターフェイス] ビュー 2791
[ダイアログ] ビュー (InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクト) 2793
ダイアログ ビュー (基本の MSI、その他の Windows Installer ベースのプロジェクト) 2794

[ウィザード インターフェイス] ビュー 2794
[ウィザード インターフェイス] ビューのツールバー *2797*
[ビルボード] ビュー 2801
ビルボード設定 *2802*
Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定 *2803*
[文字列エディター] ビュー 2807
[メディア] ビュー 2810
[パス変数] ビュー 2811
[アップグレード] ビュー 2816
[共通] タブ *2818*
[詳細] タブ *2820*
メジャー アップグレード アイテム: [共通] タブ *2822*
メジャー アップグレード アイテム: [詳細] タブ *2824*
マイナー アップグレード/スモール アップデート アイテムの設定 *2828*
自動アップグレード アイテムの設定 *2828*
[リリース] ビュー 2829
[製品構成] の設定 *2830*
製品構成の [全般] タブ 2830
製品構成の [複数インスタンス] タブ 2842
リリースの設定 *2842*
リリースの [ビルド] タブ 2843
リリースの [Setup.exe] タブ 2856
リリースの [署名] タブ 2872
リリースの [インターネット] タブ 2880
リリースの [イベント] タブ 2883
リリースの [Appx/MSIX] タブ 2896
リリースの [アップデート] タブ 2907
MSIX リリースの [全般] タブ 2910
MSIX リリースの [アプリ インストーラー] タブ 2921
連鎖 .msi パッケージの設定 *2924*
[パッチのデザイン] ビュー 2929
パッチの構成 *2930*
[共通] タブ 2930
[識別] タブ 2932
[デジタル署名] タブ 2933
[シーケンス] タブ 2935
[詳細] タブ 2936
最新セットアップ *2945*
以前のセットアップ *2946*
[共通] タブ 2946
[詳細] タブ 2947
追加の外部ファイル *2949*
外部ファイル 2949
[追加ツール] ビュー 2949
依存関係スキャナー 2951
MSI デバッガー 2951
ダイレクト エディター 2952
Windows Installer テーブル リファレンス *2958*
InstallShieldテーブル リファレンス *2958*
ISAlias テーブル 2960

ISCOMCatalogAttribute テーブル 2961
ISCOMCatalogCollection テーブル 2962
ISCOMCatalogCollectionObject テーブル 2962
ISCOMCatalogObject テーブル 2963
ISCOMPlusApplication テーブル 2963
ISCustomActionReference テーブル 2964
ISIISItem テーブル 2965
ISIISProperty テーブル 2965
ISProductConfigurationInstance テーブル 2967
ISSelfReg テーブル 2968
InstallShield標準 Windows Installer テーブルの列 2969
QuickPatch プロジェクト 2971
[パッチの設定] ビュー 2971
[一般情報] ビュー 2972
製品のプロパティ 2972
ビルドの設定 2973
[共通] タブ 2973
[識別] タブ 2974
[デジタル署名] タブ 2974
[詳細] タブ 2977
履歴 2985
カスタム アクション 2985
[ファイル] ビュー 2985
新規ファイルの設定 2985
変更/削除されたファイルの設定 2986
[レジストリ] ビュー 2988
InstallShield 前提条件エディター リファレンス 2991
[プロパティ] タブ 2992
[Windows パッケージ マネージャー] タブ 2992
[条件] タブ 2994
[含めるファイル] タブ 2995
[実行するアプリケーション] タブ 2995
[動作] タブ 2998
[依存関係] タブ 3003
エラーと警告 3005
ビルド エラーと警告 3006
メディア ビルド エラーと警告 3103
エラー 118 3105
エラー 129 3105
警告 -5000 3106
警告 -5006 3106
警告 -5020 3106
警告 -5021 3107
警告 -5032 3107
警告 -5050 3107
警告 -5051 3107
エラー -7040 3108
エラー -7041 3108
エラー -7043 3108

エラー –7044 . . . 3109
エラー –7045 . . . 3109
エラー –7126 . . . 3109
エラー –7127 . . . 3109
エラー –7275 . . . 3110
エラー –7276 . . . 3110
エラー –7380 . . . 3110
エラー –9008 . . . 3110
MSI/MSM 変換エラー . . . 3111
Windows Installer 実行時のエラー . . . 3115
Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト) . . . 3125
Setup.exe 戻り値および実行時のエラー (基本の MSI および InstallScript MSI プロジェクト) . . . 3128
Setup.exe 戻り値および実行時のエラー (アドバンスト UI およびスイート/アドバンスト UI) . . . 3133
Visual Studio プロジェクトのインポート エラーと警告 . . . 3143
アップグレード エラー (InstallShield Professional からのアップグレード) . . . 3162
アップグレードの警告 (InstallShield Professional からのアップグレード) . . . 3170
更新の警告 289: 動的にリンクされたファイルのファイルリンクが存在しません . . . 3170
更新の警告 –305: InstallShield Professional コンポーネントの "ビルドに含める" プロパティを直接移行できませんでした . . . 3171
更新の警告 –480: ファイル グループ中のファイルの共有 .dll 参照カウントのインクリメント . . . 3171
更新の警告 –481: ファイルグループ内のダイナミックリンクファイルの共有 DLL 参照カウントのインクリメント . . . 3172
更新の警告 –486: ショートカット アイコンの指定 . . . 3172
更新の警告 –487: インストール時のファイルの上書き . . . 3173
更新の警告 –488: スクリプト ファイルに古い形式のイベントがあります . . . 3174
更新の警告 495: ODBC コアがデフォルトでインストールされていません . . . 3175
アップグレード エラーと警告 (InstallShield—Windows Installer Edition からのアップグレード) . . . 3175
Windows Installer ランタイムエラーの HRESULT 値 . . . 3177
DIFxAPI エラー (InstallScript プロジェクト) . . . 3178
"文字列 PRODUCT_NAME が文字列テーブルで見つかりません" エラー . . . 3180
InstallScript エラー情報 . . . 3181
InstallScript 構文またはコンパイラ エラー . . . 3181
エラー C8001 . . . 3185
エラー C8002 . . . 3186
エラー C8003 . . . 3186
エラー C8004 . . . 3186
エラー C8005 . . . 3187
エラー C8006 . . . 3187
エラーC8007 . . . 3188
エラー C8008 . . . 3188
エラー C8009 . . . 3188
エラー C8010 . . . 3189
エラー C8011 . . . 3189
エラー C8012 . . . 3189
エラー C8013 . . . 3189
エラー C8014 . . . 3190
エラー C8015 . . . 3190

エラー C8016. 3191
エラー C8017. 3191
エラー C8018. 3191
エラー C8019. 3192
エラー C8020. 3192
エラー C8021. 3193
エラー C8022. 3193
エラー C8023. 3193
エラー C8024. 3193
エラー C8025. 3194
エラー C8026. 3195
エラー C8027. 3195
エラー C8028. 3195
エラー C8031. 3196
エラー C8032. 3196
エラー C8033. 3196
エラー C8034. 3197
エラー C8035. 3197
エラー C8036. 3197
エラー C8037. 3198
エラー C8038. 3198
エラー C8039. 3198
エラー C8040. 3199
エラー C8042. 3199
エラー C8043. 3199
エラー C8044. 3200
エラー C8045. 3200
エラー C8046. 3200
エラー C8047. 3201
エラー C8048. 3201
エラー C8049. 3201
エラー C8050. 3202
エラー C8051. 3202
エラー C8052. 3202
エラー C8053. 3202
エラー C8054. 3203
エラー C8055. 3203
エラー C8057. 3203
エラー C8058. 3204
エラー C8059. 3204
エラー C8062. 3204
エラー C8063. 3204
エラー C8064. 3205
エラー C8065. 3205
エラー C8066. 3205
エラー C8067. 3206

エラー C8068. . . . . *3206*
エラー C8069. . . . . *3206*
エラー C8070. . . . . *3207*
エラー C8071. . . . . *3207*
エラー C8072. . . . . *3207*
エラー C8073. . . . . *3208*
エラー C8074. . . . . *3208*
エラー C8075. . . . . *3208*
エラー C8076. . . . . *3209*
エラー C8077. . . . . *3209*
エラー C8078. . . . . *3209*
エラー C8079. . . . . *3210*
エラー C8080. . . . . *3210*
エラー C8081. . . . . *3210*
エラー C8082. . . . . *3211*
エラー C8083. . . . . *3211*
エラー C8084. . . . . *3211*
エラー C8085. . . . . *3212*
エラー C8086. . . . . *3212*
エラー C8087. . . . . *3212*
エラー C8088. . . . . *3212*
エラー C8089. . . . . *3213*
エラー C8090. . . . . *3213*
エラー C8091. . . . . *3213*
エラー C8092. . . . . *3214*
エラー C8093. . . . . *3214*
エラー C8097. . . . . *3214*
エラー C8098. . . . . *3215*
エラー C8099. . . . . *3215*
エラー C8100. . . . . *3215*
エラー C8101. . . . . *3216*
エラー C8112. . . . . *3216*
エラー C8113. . . . . *3217*
エラー C8114. . . . . *3217*
エラー C8115. . . . . *3217*
エラー C8126. . . . . *3217*
エラー C8127. . . . . *3218*
エラー C8128. . . . . *3218*
エラー C8522. . . . . *3218*
InstallScript の致命的エラー. 3219
エラー F8501. . . . . *3220*
エラー F8502. . . . . *3221*
エラー F8503. . . . . *3221*
エラー F8504. . . . . *3222*
エラー F8505. . . . . *3222*
エラー F8506. . . . . *3223*

エラー F8508. 3223
エラー F8509. 3223
エラー F8510. 3224
エラー F8511. 3224
エラー F8512. 3225
エラー F8513. 3225
エラー F8514. 3225
エラー F8515. 3225
エラー F8516. 3226
エラー F8517. 3226
エラー F8518. 3226
エラー F8519. 3227
InstallScript 内部エラー 3227
エラー C9001. 3227
InstallScript 警告 3227
警告 C7501 3228
警告 C7502 3228
Warning C7503. 3229
Warning C7505. 3229
仮想化変換のエラーと警告 3231
エラー -9000: 不明な例外 3231
エラー -9001: 不明な COM. 3231
エラー -9002: パッケージを開くときにエラーが発生しました 3231
エラー -9003: パッケージを保存するときにエラーが発生しました 3232
エラー -9004: ユーザーによってプロセスがキャンセルされました 3232
エラー -9005: 一時フォルダーの作成中にエラーが発生しました 3233
エラー -9006: パッケージの圧縮中にエラーが発生しました. 3233
エラー -9007: 拡張子を持つファイルが見つかりませんでした 3234
エラー -9008: アイコンの抽出中にエラーが発生しました 3234
エラー -9009: 不明のプロバイダー 3234
エラー -9010: ターゲット ファイル名が正しくありません 3235
エラー -9011: MSI テーブルの読み込み中にエラーが発生しました. 3235
エラー -9012: メソッドで予期しないエラーが発生しました 3236
エラー -9013: タイプ ライブラリが見つかりませんでした 3236
エラー -9014: ShellExecute が失敗しました 3236
エラー -9015: ドライバーの完全パスを判別できませんでした 3237
警告 -9016: テーブルのコンテンツが無視されました 3238
警告 -9017: .NET 1.x アセンブリはサポートされていません 3238
警告 -9018: カスタム アクションが無視されました 3238
警告 -9019: 条件付きコンポーネント. 3239
エラー -9020: ディレクトリの親がヌルです 3240
エラー -9021: COM データを抽出できませんでした 3241
エラー -9022: Complus テーブル 3241
エラー -9024: FileSFPCatalog 3242
警告 -9026: LaunchCondition テーブル 3242
警告 -9027: LockPermissions テーブル 3243

エラー -9028: MoveFile テーブル 3244
エラー -9029: MsiDriverPackages テーブル 3244
警告 -9030: ODBCTranslator テーブル 3245
警告 -9031: RemoveFile テーブル 3245
警告 -9032: RemoveIniFile テーブル 3246
警告 -9033: RemoveRegistry テーブル 3246
エラー -9036: ISCEInstall テーブル 3247
エラー -9037: ISComPlusApplication テーブル 3247
エラー -9038: ISPalmApp テーブル 3248
エラー -9039: ISSQLScriptFile テーブル 3248
エラー -9040: ISVRoot テーブル 3249
エラー -9041: ISXmlFile テーブル 3250
エラー -9051: パッケージの圧縮中がキャンセルされました 3250
エラー -9100: パッケージ オブジェクトのインスタンスの作成が失敗しました 3250
エラー -9101: パッケージ オブジェクトの作成操作が失敗しました 3251
Error -9102: ヘッダー情報の書き込みに失敗しました 3251
エラー -9103: Citrix の最終処理が失敗しました 3252
エラー -9104: Citrix の保存に失敗しました 3252
エラー -9105: Citrix ライターの初期化中にエラーが発生しました 3252
エラー -9106: Citrix パッケージの初期化中にエラーが発生しました 3253
エラー -9107: Citrix ファイル エントリの書き込み中にエラーが発生しました 3253
エラー -9108: ソース ファイル パスの判別中にエラーが発生しました 3254
エラー -9109: Citrix フォルダー エントリの書き込み中にエラーが発生しました 3254
エラー -9110: Citrix レジストリ エントリの書き込み中にエラーが発生しました 3254
エラー -9113: Citrix INI ファイル エントリの書き込み中にエラーが発生しました 3255
エラー -9114: Citrix ショートカットの書き込み中にエラーが発生しました 3255
エラー -9115: Citrix プロファイルを保存するときにエラーが発生しました 3256
エラー -9116: 空の Citrix プロファイルを作成するときにエラーが発生しました 3256
エラー -9117: 中間フォルダーの作成中にエラーが発生しました 3256
エラー -9118: Citrix プロファイルの初期化中にエラーが発生しました 3257
エラー -9119: Citrix プロファイルにデフォルト ターゲットを作成するときにエラーが発生しました 3257
エラー -9120: プロファイルからファイルを削除中にエラーが発生しました 3258
エラー -9121: ファイルを Citrix プロファイルにコピーできませんでした 3258
エラー -9122: ターゲットが Citrix プロファイルに存在しません 3258
エラー -9124: このプロファイルに作成されたショートカットがありません 3259
エラー -9125: Citrix ファイルの種類の関連付けの書き込み中にエラーが発生しました 3259
エラー -9126: 証明書を使ってプロファイルを署名できませんでした 3260
エラー -9127: スクリプトの実行を作成できませんでした 3260
警告 -9128: ショートカットが重複しています 3260
警告 -9129: ショートカットの名前が重複しています 3261
警告 -9130: ショートカットのターゲットが重複しています 3261
警告 -9131: インストーラーの変数を解決できませんでした 3262
警告 -9132: 16 色ショートカット アイコンが見つかりませんでした 3262
警告 -9133: ショートカット アイコンが見つかりませんでした 3262
警告 -9134: 実行可能ファイルからアイコンを抽出できませんでした 3263
エラー -9135: ショートカットのターゲットが 16 ビットです 3263

警告 -9136: 一部のファイルが圧縮されていない可能性があります 3264
警告 -9137: 対象となるディレクトリが見つかりませんでした 3264
Warning -9138: DuplicateFile テーブル エントリを無視しています 3265
警告 -9150: Windows Installer パッケージをビルド中に警告が発生しました 3265
エラー -9151: Windows Installer パッケージをビルド中にエラーが発生しました 3266
Error -9200: ThinApp のインストールが必要です 3266
警告 -9201: ショートカット ファイルの拡張子は".exe" である必要があります 3267
エラー -9202: アプリケーションが作成されませんでした 3267
Error -9203: ThinApp ツールがありません 3267
エラー -9204: ショートカットが重複しています 3268
エラー -9205: 同じ名前のショートカットが既に存在しますが、パラメーターが異なります 3268
エラー -9206: 同じ名前のショートカットが既に存在しますが、ターゲットが異なります 3269
エラー -9207: ビルド プロセス中にエラーが発生しました (vregtool.exe) 3269
エラー -9208: ビルド プロセス中にエラーが発生しました (vftool.exe) 3269
エラー -9209: ビルド プロセス中にエラーが発生しました (tlink.exe) 3270
エラー -9300: AdviseFile 操作で、未処理の例外が発生しました 3270
エラー -9301: AdviseRegistry 操作で、未処理の例外が発生しました 3271
エラー -9302: コマンド操作で、未処理の例外が発生しました 3271
エラー -9303: ファイルの変更操作で、未処理の例外が発生しました 3271
エラー -9304: レジストリの変更操作で、未処理の例外が発生しました 3272
エラー -9305: 作成操作で、未処理の例外が発生しました 3272
エラー -9306: 規則エンジンの実行時に、未処理の例外が発生しました 3272
エラー -9401: App-V ライターの初期化中に、エラーが発生しました 3273
エラー -9402: App-V パッケージの初期化中に、エラーが発生しました 3273
エラー -9403: App-V ファイル エントリを書き込み中にエラーが発生しました 3273
エラー -9404: App-V フォルダー エントリを書き込み中にエラーが発生しました 3274
エラー -9405: App-V レジストリ エントリを書き込み中にエラーが発生しました 3274
エラー -9406: App-V INI ファイル エントリを書き込み中にエラーが発生しました 3274
エラー -9407: App-V ショートカットを書き込み中にエラーが発生しました 3275
エラー -9408: App-V ファイル型のデータを書き込み中にエラーが発生しました 3275
エラー -9409: App-V データの保存中にエラーが発生しました 3275
エラー -9410: ソース ファイル パスの判別中にエラーが発生しました 3276
エラー -9411: OSD ファイルのテンプレートを抽出できませんでした 3276
エラー -9412: OSD ファイルを保存できませんでした 3276
エラー -9413: App-V OSD の保存 3277
警告 -9414: プライマリ アプリケーションの依存関係として指定されたローカル App-V アプリケーション 3277
エラー -9415: 依存関係のアプリケーションが見つかりませんでした 3277
警告 -9416: プライマリ アプリケーション ディレクトリが無効です 3278
エラー -9417: 依存アプリケーションの OSD ファイルに、無効な HREF の値が含まれています。 3278
エラー -9418: サイドバイサイド アセンブリをプライベート化中にエラー 3279
エラー -9419: ウォーターマークを挿入中に、エラーが発生しました 3279
エラー: -9424: App-V 5.x パッケージをビルドする 3279
警告 -9500: ショートカットが不足しています 3280
エラー -10000: ユーザーによってプロセスがキャンセルされました 3280
警告 -10001: スイート ファイルが不足しています 3280
警告 -10002: スイート ファイルが重複しています 3281

警告 -10003: アプリケーション ファイルが不足しています 3281
警告 -10004: INI ファイルが不足しています 3281
修正 11000: Excluding TCPIP レジストリ エントリを除外しています 3282
致命的エラー 11001: VMware が失敗しました 3282
警告 11003: コントロール パネル アプレット - Citrix 3283
修正 11004: コントロール パネル アプレット - ThinApp 3283
致命的エラー 11005: QuickTime 7.4.1 が原因で、致命的エラーが発生しました 3283
修正 11006: Adobe Distiller によって AdobePDFSettings が除外されました 3284
修正 11007: URL ショートカットの除外 3284
テクニカル サポートにお問い合わせの前に 3284
変換前および後の操作が必要なアプリケーションの機能 3284
オートメーション インターフェイス 3287
オートメーション オブジェクト 3287
ISWiProject オブジェクト 3288
AddAdvancedFile メソッド 3305
AddAutomaticUpgradeEntry メソッド 3306
AddChainPackage メソッド 3307
AddComponent メソッド 3308
AddCustomAction メソッド 3309
AddFeature メソッド 3310
AddLanguage メソッド 3312
AddPatchConfig Method 3312
AddPathVariable メソッド 3313
AddProductConfig メソッド 3314
AddProperty メソッド 3314
AddSetupFile メソッド 3315
AddSetupType メソッド 3316
AddSQLServerConnection メソッド 3317
AddUpgradeTableEntry メソッド 3318
BuildPatchConfiguration メソッド 3319
BuildPCPFile メソッド 3321
CloseProject メソッド 3322
CreatePatch メソッド 3322
CreateProject メソッド 3323
DeleteAdvancedFile メソッド 3324
DeleteChainPackage メソッド 3325
DeleteComponent メソッド 3325
DeleteCustomAction メソッド 3326
DeleteMergeModule メソッド 3327
DeletePatchConfig メソッド 3327
DeletePathVariable メソッド 3328
DeleteProductConfig メソッド 3329
DeleteProperty メソッド 3329
DeleteSetupFile メソッド 3330
DeleteSetupType メソッド 3331
DeleteSQLConnection メソッド 3332

DeselectLanguage メソッド 3333
ExportProject メソッド 3334
ExportStrings メソッド 3334
ForceUpgrade メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3335
GenerateGUID メソッド 3336
ImportProject メソッド 3337
ImportStrings メソッド 3337
OpenProject メソッド 3339
RemoveFeature メソッド 3341
SaveProject メソッド 3342
ISWiAdvancedFile オブジェクト 3342
ISWiAutomaticUpgradeEntry オブジェクト 3344
Delete メソッド 3344
ISWiChainPackageData オブジェクト 3345
ISWiChainPackage オブジェクト 3346
AddChainPackageData メソッド 3347
DeleteChainPackageData メソッド 3348
ISWiCustomAction オブジェクト 3349
ISWiComponent オブジェクト 3350
AddComponentSubFolder メソッド 3368
AddDynamicFileLinking メソッド 3369
AddEnvironmentVar メソッド 3370
AddFile メソッド 3371
AddRemoveFile メソッド 3372
DeviceDriverFlagsEx プロパティ 3373
DeviceDriverSetRedist プロパティ 3374
ImportINIFile メソッド 3375
ImportRegFile メソッド 3376
ISWiRemoveFiles メソッド 3377
RemoveComponentSubFolder メソッド 3378
RemoveDynamicFileLinking メソッド 3379
RemoveEnvironmentVar メソッド 3380
RemoveFile メソッド 3380
RemoveRemoveFile メソッド 3381
ISWiComponentSubFolder オブジェクト 3382
DeleteFile メソッド 3383
ISWiCondition オブジェクト 3384
ISWiDynamicFileLinking オブジェクト 3384
ISWiEnvironmentVar オブジェクト 3387
ISWiFeature オブジェクト 3389
AddCondition メソッド 3396
AddMergeModule メソッド 3397
AddObject メソッド 3397
AttachComponent メソッド 3398
DeleteCondition メソッド 3399
RemoveComponent メソッド 3399

RemoveMergeModule メソッド 3400
RemoveObject メソッド 3401
ISWiFile オブジェクト 3401
ISWiFolder オブジェクト 3407
AddShortcut メソッド 3410
AddSubFolder メソッド 3410
DeleteShortcut メソッド 3411
DeleteSubFolder メソッド 3412
ISWiLanguage Object 3412
AddStringEntry メソッド 3413
DeleteStringEntry メソッド 3414
ISWiLatestSetup オブジェクト 3415
AddPreviousSetup メソッド 3417
DeletePreviousSetup メソッド 3417
ISWiMsiPatchSequence オブジェクト 3418
ISWiObject オブジェクト 3420
ISWiPatchConfig オブジェクト 3420
AddLatestSetup メソッド 3432
DeleteLatestSetup メソッド 3433
ISWiPatchExternalFile オブジェクト 3433
ISWiPatchWholeFile オブジェクト 3434
ISWiPathVariable オブジェクト 3435
ISWiPreviousSetup オブジェクト 3437
ISWiProductConfig オブジェクト 3438
AddRelease メソッド 3443
DeleteRelease メソッド 3444
ISWiProperty オブジェクト 3444
ISWiRelease オブジェクト 3445
ビルド メソッド 3505
BuildTablesOnly メソッド 3505
BuildTablesRefreshFiles メソッド 3506
CubFile プロパティ 3506
SignMedia プロパティ 3507
ISWiRemoveFile オブジェクト 3508
ISWiSequenceRecord オブジェクト 3509
ISWiSetupFile オブジェクト 3510
ISWiSetupType オブジェクト 3512
ISWiShellProperty オブジェクト 3513
ISWiShortcut オブジェクト 3513
AddShellProperty メソッド 3520
DeleteShellProperty メソッド 3521
ISWiSISProperty オブジェクト 3522
ISWiSQLConnection オブジェクト 3523
AddSQLRequirement メソッド 3526
AddSQLScript メソッド 3527
AddSQLScriptEx メソッド 3527

DeleteSQLScript メソッド 3528
GetDatabaseServer メソッド 3529
InsertSQLScript メソッド 3529
RemoveSQLRequirement メソッド 3530
ISWiSQLDatabaseServer オブジェクト 3530
ISWiSQLReplace オブジェクト 3532
ISWiSQLRequirement オブジェクト 3533
SetPredefinedRequirement メソッド 3534
ISWiSQLScript オブジェクト 3536
AddSQLScriptError メソッド 3539
AddSQLScriptReplacement メソッド 3539
DeleteSQLReplace メソッド 3540
DeleteSQLScriptError メソッド 3540
ISWiSQLScriptError オブジェクト 3541
ISWiStringEntry オブジェクト 3541
ISWiUpgradeTableEntry オブジェクト 3542
Delete メソッド 3543
オートメーション コレクション 3544
ISWiAdvancedFiles コレクション 3544
ISWiAutomaticUpgradeEntries コレクション 3545
ISWiChainPackageDatas コレクション 3546
ISWiChainPackages コレクション 3547
ISWiComponents コレクション 3548
ISWiComponentSubFolders コレクション 3550
ISWiConditions コレクション 3551
ISWiCustomActions コレクション 3552
ISWiDynamicFileLinkings コレクション 3553
ISWiEnvironmentVars コレクション 3554
ISWiFeatures コレクション 3554
ISWiFiles コレクション 3556
ISWiFolders コレクション 3557
ISWiLanguages コレクション 3558
ISWiLatestSetups コレクション 3559
AddMsiPatchSequence メソッド 3560
DeleteMsiPatchSequence メソッド 3560
ISWiMsiPatchSequences コレクション 3561
BuildPatch メソッド 3562
ISWiObjects コレクション 3564
ISWiPatchConfigs コレクション 3565
ISWiPatchExternalFiles コレクション 3566
AddPatchWholeFile メソッド 3567
DeletePatchWholeFile メソッド 3568
ISWiPatchWholeFiles コレクション 3568
ISWiPathVariables コレクション 3569
ISWiPreviousSetups コレクション 3570
AddPatchExternalFile メソッド 3571

DeletePatchExternalFile メソッド 3572
ISWiProductConfigs コレクション 3572
ISWiProperties コレクション 3574
ISWiReleases コレクション 3575
ISWiRemoveFiles コレクション 3576
ISWiSequence コレクション 3577
InsertCustomAction メソッド 3579
RemoveSequenceRecord メソッド 3579
ISWiSetupFiles コレクション 3580
ISWiSetupTypes コレクション 3581
ISWiShellProperties コレクション 3582
ISWiShortcuts コレクション 3583
ISWiSISProperties コレクション 3584
ISWiSQLConnections コレクション 3585
ISWiSQLDatabaseServers コレクション 3586
ISWiSQLReplaces コレクション 3587
ISWiSQLRequirements コレクション 3588
ISWiSQLScriptErrors コレクション 3589
ISWiSQLScripts コレクション 3590
ISWiStringEntries コレクション 3591
ISWiUpgradeTableEntries コレクション 3592
» アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション オブジェクト 3595
ISWiProject オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3595
AddExitCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3603
AddLanguage メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3604
AddPathVariable メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3605
AddProperty メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3605
AddSetupFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3606
AddSuiteAction メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3607
AddSuiteFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3608
AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3610
AddSuiteRelease メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3612
AddSuiteTransaction メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3612
CloseProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3613
CreateProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3614
DeleteExitCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3615
DeleteLanguage メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3615
DeletePathVariable メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3616
DeleteProperty メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3617
DeleteSetupFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3618
DeleteSuiteAction メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3619
DeleteSuiteFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3619
DeleteSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3620
DeleteSuiteRelease メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3621
ForceUpgrade メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3621
GenerateGUID メソッド (アドバンスト UI およびスイート/アドバンスト UI) 3622

ISWiSuiteEvent メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3622*
OpenProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3625*
SaveProject メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3626*
ISWiLanguage オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3627
AddStringEntry メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3628*
DeleteStringEntry メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3629*
ISWiPathVariable オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3630
ISWiProperty オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3633
ISWiSetupFile オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3634
ISWiSuiteAction オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3635
ISWiSuiteActionRef オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3643
MoveTo メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3644*
ISWiSuiteCondition オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3645
AddCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3647*
AddExtensionCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3651*
AddGroup メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3651*
AddParameter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3652*
ChangeGroupType メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3653*
DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3654*
DeleteParameter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3655*
Move メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3656*
ISWiSuiteConditionParameter オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3656
ISWiSuiteDFLFilter オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3657
CanMove メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3658*
MoveDown Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3659*
MoveUp Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3660*
ISWiSuiteDynamicFileLink オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3660
AddFilter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3661*
DeleteFilter メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3662*
ISWiSuiteEvent オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3663
AddSuiteActionRef メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3664*
DeleteSuiteActionRef メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3665*
ISWiSuiteExitCondition オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3666
ISWiSuiteFeatureオブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3666
AddCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3670*
AddSuiteSubFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3671*
AttachPackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3671*
DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3672*
DeleteSuiteSubFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3672*
RemovePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3673*
ISWiSuiteFile オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3674
ISWiSuiteFolder オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3675
AddDynamicFileLink メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3676*
AddFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3677*
AddFolder メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3678*
DeleteDynamicFileLink メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3679*

DeleteFile メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3679*
DeleteFolder メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3680*
ISWiSuiteOperation オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3681
ISWiSuitePackage オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3686
AddDetectCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3708*
AddEligibleCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3709*
AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3709*
DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3711*
MoveDown Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3712*
MoveUp Method メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3713*
ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI) 3713
Build メソッド (アドバンスト UI およびスイート/アドバンスト UI) *3733*
» アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション コレクション 3734
ISWiLanguages コレクション 3735
ISWiPathVariables コレクション 3736
ISWiProperties コレクション 3736
ISWiSetupFiles コレクション 3737
ISWiSuiteActionRefs コレクション 3738
ISWiSuiteActions コレクション 3738
ISWiSuiteConditionParameters コレクション 3739
ISWiSuiteConditions コレクション 3740
ISWiSuiteDFLFilters コレクション 3740
ISWiSuiteDynamicFileLinks コレクション 3741
ISWiSuiteEvents コレクション 3742
ISWiSuiteExitConditions コレクション 3743
ISWiSuiteFeatures コレクション 3744
ISWiSuiteFiles コレクション 3745
ISWiSuiteFolders コレクション 3746
ISWiSuitePackages コレクション 3747
ISWiSuiteReleases コレクション 3748
アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス 3749
Feature オブジェクト 3750
File オブジェクト 3752
Parcel オブジェクト 3753
Platform オブジェクト 3755
Registry オブジェクト 3757
InstallShield カスタム アクション リファレンス 3761
コマンドライン ツール 3773
Compile.exe 3775
ISCmdBld.exe 3781
ISBuild.exe 3791
IISscan.exe 3792
iSign.exe 3793
ReleasePackager.exe 3794
SetupIni.exe 3795
MsiExec.exe コマンドラインのパラメーター 3796

Setup.exe および Update.exe コマンドライン パラメーター 3801
アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター 3821
Setup.exe (InstallScript プロジェクト) 3826

14 よく寄せられる質問 (FAQ) 3827

用語集 3831

1 索引 1

1

InstallShield 2025 ヘルプ ライブラリ

InstallShield を使って、Windows ベースのシステムをターゲットにするインストールを素早くビルド、テストおよび配布することができます。

InstallShield ヘルプ ライブラリには、InstallShield の機能性および機能についての情報が掲載されています。ヘルプ ライブラリは、次のセクションに分かれています。

テーブル 1-1 • ヘルプ ライブラリのセクション

セクション	説明
InstallShield 2025 の新しい機能	InstallShield 2025 の変更点について案内します。
InstallShield の以前のバージョンの新しい機能	InstallShield の以前のバージョンで加えられた変更に関する情報です。
ターゲット システムの要件	ターゲット システムの要件を一覧にします。
InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い	管理者権限を持たずに InstallShield を実行している場合に利用できない機能についてアラートします。また、管理者と非管理者コンテキストを切り替えた場合で、プロジェクトでマップされたドライブを使用した際に発生する可能性のある問題についても説明します。
32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い	InstallShield を 32 ビット システムで使用している場合に現れる可能性がある、64 ビット システムとの相違点をハイライトします。
ヘルプの使い方	InstallShield ドキュメントに関する情報が提供されています。
スタート ガイド	InstallShield に慣れていただき、インストール プロジェクトの作成の開始、および InstallShield ユーザー インターフェイスのカスタマイズに役立つ情報が掲載されています。

テーブル 1-1 • ヘルプ ライブラリのセクション (続き.)

セクション	説明
チュートリアル	InstallScript および基本の MSI インストール プロジェクトの処理手順、およびグローバル インストールの作成方法について、順を追って説明します。
インストールの作成	ユーザーフレンドリーで信頼できるインスールの作成方法、およびその作成手順がステップごと ([プログラムの追加と削除] のための情報指定からインストールのビルド、テストおよび配布まで) に説明されています。
アドバンスト UI およびスイート/アドバンスト UI インストールの作成	スイート/アドバンスト UI およびアドバンスト UI プロジェクトについての概要および詳しい使用方法が説明されています。 InstallShield Premier Edition で提供されているスイート/アドバンスト UI プロジェクト タイプでは、複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installern トランザクションだけでなく、複数の InstallShield 前提条件も、モダンでカスタマイズ可能なユーザー インターフェイスを持つ単一のアプリケーションにパッケージできます。 InstallShield Professional Edition で提供されているアドバンスト UI プロジェクト タイプでは、単一の .msi パッケージ、.msp パッケージ、または InstallScript パッケージに対して、最新のカスタマイズ可能なユーザー インターフェイスを提供することができます。
InstallShield 前提条件およびその他の再配布可能ファイルをデザインする	他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布できる InstallShield 前提条件、マージ モジュール、InstallScript オブジェクトを自分自身で初めて設計するときに役に立つ基本的な概念が紹介されています。
開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化	デベロッパー インストール マニフェスト (DIM) の紹介。DIMは、製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、およびインストール パッケージの論理的に分かれている部分を別個に構成している要素など、関連するアイテムを集めたものです。サイズは、機能とほぼ同じです。
アプリケーションのアップデート	製品アップデートのために用意されている様々なタイプのアップグレードおよびパッチの計画および実装の仕方をステップごとに見ていくことができます。また、FlexNet Connect を利用して、どのようにエンドユーザーに入手可能なアップグレードおよびパッチについて通知するかについても説明されています。
カスタム仮想アプリケーションの作成	InstallShield を使ってカスタマイズされた仮想アプリケーションを作成する方法について説明します。
追加のインストール オプション	InstallShield で幅広く提供されているオプションの説明で、多言語インストールの作成、1 つの製品の複数インスタンスのインストール、条件ステートメントのビルド、インストール済みデータの検索、インストール テーブルの編集、その他多くのトピックが掲載されています。

テーブル 1-1 • ヘルプ ライブラリのセクション (続き.)

セクション	説明
InstallShield と外部アプリケーションの統合	InstallShield と ソース コード管理ソフトウェア、Microsoft Visual Studio や Microsoft Visual Studio Team Foundation Server (TFS) などのサードパーティ ツールとの統合について詳細を説明します。
インストールの開発およびビルド プロセスの自動化	InstallShield ユーザー インターフェイスを直接開くことなくインストール プロジェクトの作成プロセスを自動化することができる InstallShield オートメーション インターフェイスについての詳細が載っています。
リファレンス	次の項目に関する総合リファレンス情報です: InstallShield ユーザー インターフェイス、InstallScript 言語、 インストールの作成、コンパイル、ビルド、実行時に発生する可能性があるエラーおよび警告、リリースのビルドおよびインストールの実行等のタスクを実行するためにコマンドラインから利用することができるツール、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの様々な設定にオブジェクト式を埋め込んでターゲット システムを検索するときに使用できるオブジェクト、プロジェクトに追加される InstallShield カスタム アクション、オートメーション インターフェイスを使ってインストール プロジェクトを変更するときに使われるオブジェクト、メソッド、プロパティ およびコレクション。
よく寄せられる質問 (FAQ)	InstallShield およびプロジェクトの作成について頻繁に寄せられる質問に答えるヘルプ トピックを見つけることができます。
用語集	用語およびその解説を集めたものです。

メモ • InstallShield ヘルプライブラリは、InstallShield とインタラクトするよう設計されているので、InstallShield 内からヘルプを開くことをお勧めします。ヘルプファイルを別のフォルダーやシステムにコピーすると、多くの機能が正常に機能しないことがあります。

InstallShield に関してよく寄せられる質問や、ドキュメントに記載されていない新規の情報については ナレッジベースを参照してください。

InstallShield 2025 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2025 R2 の新しい機能
- InstallShield 2025 R1 の新しい機能

InstallShield 2025 R2 の新しい機能

InstallShield 2025 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- 仮想マシンのサポートの強化
- InstallShield 前提条件ダイアログに追加された新しいレジストリ条件タイプ

仮想マシンのサポートの強化

InstallShield 2025 R 2は、インストールを実行できる次の仮想環境をサポートすることで、仮想マシン検出機能を強化します:

- Xen
- Amazon EC 2 インスタンス
- QEMU
- Parallels Desktop
- Google Compute
- Microsoft Azure
- EverRun

仮想マシンのこの拡張されたサポート（前述）により、InstallShield はインストールが実行されている環境を正確に識別できるため、より信頼性が高く、予測可能なインストール動作を実現できます。これにより、幅広い仮想プラットフォームで徹底的な互換性チェックが可能になり、インストール エラーのリスクが軽減され、基礎となるシステムにかかわらず、ユーザーにとってシームレスでスムーズなエクスペリエンスが保証されます。

InstallShield 前提条件ダイアログに追加された新しいレジストリ条件タイプ

InstallShield 2025 R 2 は、[**前提条件**]ダイアログボックスに **レジストリ エントリがワイルドカード パターンに一致し、指定されたバージョン 値を持つ**という名前の新しい条件タイプを導入しました。この条件タイプを選択して構成すると、InstallShield はターゲットシステム上で、指定または入力されたレジストリ サブキー名と一致する各レジストリキー内の値データとして格納されているバージョン番号が、指定または入力されたバージョン番号よりも大きいか、小さいか、または等しいかをチェックできます。

この新しい条件タイプを選択すると、レジストリ サブキー名の予測可能な部分とワイルドカード文字 (%) を指定できます。具体的には、**"確認するレジストリキー名を指定します"** フィールドでキー名、**"確認するレジストリ値名を指定します"** フィールドで値名、**"確認するレジストリ値データを指定します"** フィールドで値データをそれぞれ指定します。

予測可能なレジストリ サブキー名パターンを指定する場合、レジストリ サブキー名と共にワイルドカード文字 (%) を使用する必要があります。

例:

HKEY_LOCAL_MACHINE¥SOFTWARE¥WOW6432Node¥Microsoft¥Updates¥.NET Core¥%Microsoft .NET Core%

この場合、InstallShieldは、Microsoft .NET Core を含む名前を持つターゲット システム上の .NET Core ノードの下にあるすべてのレジストリ サブキーを検索し、一致する各レジストリ サブ キー内で入力または指定されたレジストリ値と値データを評価します。

***メモ・** ワイルドカード文字 (%) は、非常に柔軟かつ正確な検索であるために、レジストリ サブキー名内で複数回使用することができます。たとえば、"**%Microsoft% %.NET% %Core%**" のようなパターンにより、InstallShield は幅広い関連レジストリ サブキーを識別することができます。*

この条件タイプでは、**[以下の場合に前提条件を実行します]** ドロップダウンから適切な条件を選択して、このダイアログボックスで指定したレジストリ エントリとターゲット マシン上のレジストリ エントリとを比較できます。また、64 ビット システム上でチェックを行う適切なレジストリの場所 (**32 ビット** または **64 ビット**) を選択することもできます。**[デフォルト]** ラジオボタンは、特定のオプションが選択されていないことを示し、32 ビット レジストリの場所がチェックされます。

詳細については、「[前提条件設定] ダイアログ ボックス」を参照してください。

InstallShield 2025 R1 の新しい機能

InstallShield 2025 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- アドバンスト UI およびスイート/アドバンストUIインストールでハイコントラストモードをサポート
- 64 ビット InstallScript ベースのセットアップ ランチャーのビルド サポート

アドバンスト UI およびスイート/アドバンストUIインストールでハイコントラストモードをサポート

以前のバージョンでは、アドバンスト UI およびスイート/アドバンスト UI インストーラーのインストール中に、ウィザード ページ上の本文テキストやヘッダーテキスト、ナビゲーション ボタンのラベルなどのテキスト要素は、ターゲット システムでハイコントラスト モードが有効になっている場合には読み取れませんでした。

InstallShield 2025 R 1では、アドバンスト UI およびスイート/アドバンストUIインストールが、ターゲット システムのハイコントラスト モードをサポートするように拡張されています。この機能強化により、ターゲットシステムにハイコントラスト モードを設定すると、各ウィザードページの本文テキストやヘッダー テキスト、ナビゲーション ボタンのラベルなどのテキスト要素が明確に読み取れるようになり、アクセシビリティとユーザー エクスペリエンスが向上します。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストールでハイコントラストモードをサポート」を参照してください。

64 ビット InstallScript ベースのセットアップ ランチャーのビルド サポート

***プロジェクト・** この情報は、次のプロジェクトの種類に適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield 2025 R 1では、**[メディア]** ビューの **[リリース]** エクスプローラーにあるリリースの **[ビルド]** タブに、**[アーキテクチャ]** という名前の新しい設定が導入されています。この設定により、デプロイメント要件に応じて、32 ビット (x 86) および 64 ビット (x 64) Windowsプラットフォームの両方、または 64 ビット (x 64) Windows プラットフォーム専用の InstallScript ベースのセットアップ ランチャーをビルドできるようになり、柔軟性が向上します。

この設定を導入することで、適切なシステム アーキテクチャを簡単にターゲットにして、アプリケーションのランタイム環境に合わせることができます。次のオプションを選択できます:

- **x86**—このオプションを選択すると、32 ビットと 64 ビットの両方の Windows プラットフォームで動作する 32 ビット セットアップ ランチャーがビルドされます。
- **x64**—このオプションを選択すると、64 ビット Windows プラットフォーム上で動作する 64 ビット セットアップ ランチャーがビルドされます。

デフォルトで、この設定は **x86** になっています。

InstallShield の以前のバージョンの新しい機能

このセクションでは、InstallShield の以前のバージョンでリリースされた機能と強化点が説明されています。

- InstallShield 2024 の新しい機能
- InstallShield 2023 の新しい機能
- InstallShield 2022 の新しい機能
- InstallShield 2021 の新しい機能
- InstallShield 2020 の新しい機能
- InstallShield 2019 の新しい機能
- InstallShield 2018 R2 の新しい機能
- InstallShield 2018 SP1 の新しい機能
- InstallShield 2018 の新しい機能
- InstallShield 2016 SP2 の新しい機能
- InstallShield 2016 SP1 の新しい機能
- InstallShield 2016 の新しい機能
- InstallShield 2015 SP1 の新しい機能
- InstallShield 2015 の新しい機能
- InstallShield 2014 SP1 の新しい機能
- InstallShield 2014 の新しい機能
- InstallShield 2013 SP1 の新しい機能
- InstallShield 2013 の新しい機能
- InstallShield 2012 Spring SP1 の新しい機能
- InstallShield 2012 Spring の新しい機能
- InstallShield 2012 SP1 の新しい機能
- InstallShield 2012 の新しい機能
- InstallShield 2011 の新しい機能

- InstallShield 2010 の新しい機能
- InstallShield 2010 SP1 の新しい機能
- InstallShield 2010 の新しい機能
- InstallShield 2009 SP2 の新しい機能
- InstallShield 2009 SP1 の新しい機能
- InstallShield 2009 の新しい機能
- InstallShield 2008 の新しい機能
- InstallShield 12 SP1 の新しい機能
- InstallShield 12 の新しい機能

InstallShield 2024 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2024 R2 の新しい機能
- InstallShield 2024 R1 の新しい機能

InstallShield 2024 R2 の新しい機能

InstallShield 2024 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- InstallScript パッケージで ARM をサポート

InstallScript パッケージで ARM をサポート

InstallShield 2024 R2 には、ARM プロセッサ アーキテクチャを使用した Windows 11 上での InstallScript インストールのサポートが含まれています。

InstallShield 2024 R1 の新しい機能

InstallShield 2024 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- .NET Core カスタム アクションをサポート
- SQL Scripts の データベース インポート ウィザード機能の強化

.NET Core カスタム アクションをサポート

InstallShield 2024 R1 では、プロジェクトの **[カスタム アクションとシーケンス]** ビューに .NET Core タイプのカスタムアクションを追加して、.NET Core ベースのアセンブリ ファイルをサポートすることができます.NET Core タイプのカスタム アクションは、Visual Basic .NET または C#のようなマネージ コードで書かれた .NET アセンブリ内のパブリック メソッドを呼び出します。

この機能は、基本の MSI、InstallScript MSI、および スイート/アドバンスト UI プロジェクトで使用できます。

詳細は、次を参照してください:

- カスタム アクションの種類

- カスタム アクションの設定
- .NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- .NET Framework 再配布可能ファイルをプロジェクトへ追加する
- Microsoft .NET Core 前提条件を含める
- スイート/アドバンスト UI インストールで .NET Core アクションを使用する

SQL Scripts の データベース インポート ウィザード機能の強化

[SQL スクリプト] ビューの **データベース インポート ウィザード**を使ってデータベース設定をインポートして、構成された設定およびオプションに基づいて SQL スクリプト ファイルを生成することができます。

InstallShield 2024 R1 の**データベース インポート ウィザード**の機能強化として、SQL Server 管理オブジェクト (SMO) が含まれています。SQL Server 管理オブジェクトを組み込むことで、データベース オブジェクトのインポートおよび SQL スクリプト ファイルの生成がより正確で効率的になります。
この強化機能によって、カスタマーから報告されたレガシー スクリプト問題を解決することができます。

InstallShield 2023 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2023 R2 の新しい機能
- InstallShield 2023 R1 の新しい機能

InstallShield 2023 R2 の新しい機能

InstallShield 2023 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- 内部ファイルにデジタル署名を行う機能
- デジタル署名のカスタム署名ソリューションを構成できる機能
- 新しいスケジュール タスクの時間単位のスケジューリング設定
- Standalone Build をビルド中に前提条件をダウンロードするコマンドラインをサポート
- EV デジタル署名用トークン パスワードの保存機能

内部ファイルにデジタル署名を行う機能

InstallShield 2023 R2 Premier Edition では、**[メディア]** ビューの **[リリース]** エクスプローラーに表示される **[イベント]** タブに **"ファイルの準備イベント"** および **"ファイルの準備でバッチファイルを実行する"** という設定が追加されています。こららの設定を使って、バッチ ファイル (.bat または . cmd) または実行可能ファイル (.exe) を使ったカスタム スクリプト/ コマンドを指定し、署名されていない InstallShield 内部ファイルにデジタル署名を行うことができます。

InstallShield のビルトイン署名ステップを使用する場合、内部ファイルを含むすべてのバイナリの署名はデフォルトの署名動作で管理されているため、これらの設定は不要です。

これらの設定は、InstallShield の標準署名ソリューションの代わりにカスタム署名ソリューションをご利用のお客様向けに提供されています。

詳細は、次を参照してください:

- リリースの [イベント] タブ
- [ファイルの準備イベント] ダイアログ ボックス

デジタル署名のカスタム署名ソリューションを構成できる機能

InstallShield 2023 R2 では、ビルドで生成されたファイルにデジタル署名を行うカスタム署名ソリューションを選択および構成できる新しい "署名の種類" 設定が導入されました。

これらの設定は、InstallShield の標準署名ソリューションの代わりにカスタム署名ソリューションをご利用希望のお客様向けに提供されています。**[カスタム]** 署名タイプを選択すると、カスタム署名ユーティリティ パスおよび引数を構成できる追加フィールドが有効化されます。カスタム署名ソリューションを構成するには、**[メディア]** ビューの **[リリース]** エクスプローラーに表示されているリリースの **[署名]** タブで次の新しい設定を使用してください::

- **署名の種類** – この設定を使って、ビルドが生成したファイルにデジタル署名行う方法を指定します。この設定では、次のオプションを使用できます:
  - **標準** – InstallShield デフォルト署名ツールを使って、ビルドが生成したファイルに署名を行います。
  - **カスタム** – 独自のカスタマイズ署名ツールを使って、ビルドが生成したファイルに署名を行います。このオプションを設定すると、**"パス"** および **"引数"** 設定が有効になります。
- **パス** – ビルドが生成したファイルにデジタル署名行うために使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。
- **引数** – 署名ツール構成のコマンドライン引数を指定します。たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。

  sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename]

  ***メモ・** カスタム署名ツールへのパスは、<ProgramFilesFolder>¥Windows Kits¥10¥bin¥<WinSDKVer>¥x86¥signtool.exe を使用します。*

  [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクトで定義されている任意のパス変数および環境変数を使用することができます。

詳細は、次を参照してください:

- リリースの [署名] タブ
- [デジタル署名] タブ ([パッチのデザイン] ビューのパッチ構成用)
- [デジタル署名] タブ (QuickPatch プロジェクト)

***メモ・** デフォルトで、**"署名の種類"** 設定は **[標準]** になっています。*

新しいスケジュール タスクの時間単位のスケジューリング設定

InstallShield 2023 R2 では、1 時間単位のスケジュール タスクの繰り返し頻度と、各スケジュール タスクの期間を指定できるようになりました。以下の設定は、**[システム構成]** ビューの **[スケジュールされたタスク]** エクスプローラで、新しいタスクの **"スケジュール"** 設定に追加されています:

- **繰り返し間隔 (分)** – この設定を使って、指定した時間の間に連続するタスクを実行する頻度を決定づける時間間隔を分数で指定します。この間隔は、前のスケジュール タスクの開始時点から計算されます。

  たとえば、タスクが 8:00 AM から 5:00 PM まで 1 時間間隔で実行する場合、このフィールドには 60 と入力します。

- **時間間隔 (分)** – この設定を使って、アクティブなトリガでスケジュール タスクを繰り返す期間を決定づける時間間隔を分数で指定します。ここに指定する間隔は、**"繰り返し間隔 (分)"** 設定で指定された間隔値と等しいか、それよりも長く無くてななりません。

  たとえば、8:00 AM から5:00 PM までスケジュール タスクを実行するには、540 と入力します。

詳細については、「スケジュール」を参照してください (**[スケジュール タスク]** ビューのスケジュール タスクの設定)。

メモ・この設定は、"繰り返し間隔 (分)" 設定に間隔値が指定されているときに、"時間間隔 (分)" 設定が有効化されます。

Standalone Build をビルド中に前提条件をダウンロードするコマンドラインをサポート

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- 基本の MSI
- InstallScript
- InstallScript MSI

InstallShield 2023 R2 では、-prqdownload という新しいコマンドラインパラメーターが導入されました。このパラメーターは、ISCmdBld.exe を使用してコマンドラインからプロジェクトをビルドするときに、不足している InstallShield の前提条件バイナリ ファイルを該当するプロジェクトのディレクトリに自動的にダウンロードするように Standalone Build に指示します。
詳細については、「ISCmdBld.exe」を参照してください。

メモ・Iscmdbld.exe を使ったスイート/アドバンスト UI プロジェクトがこのパラメーター無しでビルドされると、ISM パッケージに必要な InstallShield 前提条件のバイナリ ファイルが自動的にプロジェクト ファイルにダウンロードされます。そのため、このパラメーターはスイート/アドバンスト UI プロジェクトには適用しません。

EV デジタル署名用トークン パスワードの保存機能

InstallShield 2023 R2 では、EV トークンのパスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。InstallShield の以前のバージョンでは、**[証明書ストアを使用する]** オプションを使った EV 証明書がサポートされていました。ただしこのオプションを使用すると、ファイルが EV 証明書で署名されるたびにトークン パスワードが要求されます。

また、EV ベンダーに付属する [シングルログオンを有効にする] というオプションもあります。このオプションでは、1 回のトークン パスワード要求でセッションごとのユーザー介入を制限できます。InstallShield の [署名] タブでトークン パスワードを構成すると、パスワードの有効期限が切れるまで機能し続けます。トークンのパスワードは、InstallShield IDE または InstallShield オートメーション レイヤーを使って変更することができます。

詳細は、次を参照してください:

- リリースの [署名] タブ
- MSIX リリースの [全般] タブ

InstallShield 2023 R1 の新しい機能

InstallShield 2023 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- スイート/アドバンスト UI プロジェクトにおける Windows パッケージ マネージャー パッケージのサポート
- スイート/アドバンスト UI プロジェクトに新しく導入された、アプリのアップデート確認機能
- InstallShield 2022 R1 における Visual Studio 2022 の統合

スイート/アドバンスト UI プロジェクトにおける Windows パッケージ マネージャー パッケージのサポート

InstallShield 2023 R1 では、スイート/アドバンスト UI プロジェクトで新しく Windows パッケージ マネージャー (WPM) パッケージがサポートされています。Windows パッケージ マネージャー は、Windows 10 と Windows 11 上にアプリケーションをインストールするためのコマンドライン ツールおよび一連のサービスで構成される総合的なパッケージ マネージャー ソリューションです。

[編成] ビューの **[パッケージ]** エクスプローラーを使って、Windows パッケージ マネージャー (WPM) パッケージをスイート/アドバンスト UI プロジェクトに追加できます。

Windows パッケージ マネージャー (WPM) パッケージをインストールするには、ターゲット システム上に Windows パッケージ マネージャー (WPM) ソリューションがインストールされている必要があります。Windows パッケージ マネージャー(WPM) ソリューションは、アプリ インストーラー MSIX バンドルでインストールされます。Windows パッケージ マネージャー (WPM) パッケージが スイート/アドバンスト UI プロジェクトに追加されると、InstallShield は、アプリ インストーラー MSIX バンドル パッケージを依存ファイルとして追加します。ターゲット マシン上でアプリ インストーラーが使用できない場合、InstallShield はアプリ インストーラー MSIX バンドル パッケージをインストールします。Windows パッケージ マネージャー は、Windows 10 1809 からアプリ インストーラーを通してサポートされています。

詳細については、次を参照してください。

- Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する
- [パッケージ] ビュー

スイート/アドバンスト UI プロジェクトに新しく導入された、アプリのアップデート確認機能

InstallShield 2023 R1 では、スイート/アドバンスト UI セットアップ ランチャーのアップデートの設定を構成する **[アプリのアップデートを確認する]** という名前の新しい機能が、**[メディア]** ビューの **[リリース]** エクスプローラーにある **[アップデート]** タブに追加されました。

ビルドを行う前にこの新しい機能を使って、アプリケーションがターゲットマシン上に配置される時に使用可能なアプリケーションのアップデートを定期的にチェックして表示する追加設定を構成できます。最初のアップデートの確認ともない、後に続くアップデートがインストール開発環境 (IDE) で定義された間隔で表示されます。

たとえば、ビルド前に **[アプリのアップデートを確認する]** 機能の追加設定が構成されている場合、任意のアプリケーションで使用可能なアップデートがあるとき、バージョン アップデート情報と共に次の 3 つのボタンを含むダイアログ ボックスが表示されます。

- **後で通知** – 最近のアップデート情報を設定された間隔でもう一度ダイアログボックスに表示するには、このボタンをクリックします。
- **アップデートをスキップ** – 特定のアップデート情報を再び表示するダイアログ ボックスを抑制するには、このボタンをクリックします。
- **ダウンロード** – アップデートをダウンロードするには、このボタンをクリックします。

詳細については、「リリースの [アップデート] タブ」を参照してください。

InstallShield 2022 R1 における Visual Studio 2022 の統合

InstallShield 2023 R1 から、Visual Studio 2022 内で InstallScript プロジェクトの作成、編集およびビルドがサポートされています。

InstallShield 2022 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2022 R2 の新しい機能
- InstallShield 2022 R1 の新しい機能

InstallShield 2022 R2 の新しい機能

InstallShield 2022 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

InstallShield 2022 R2 における Visual Studio 2022 の統合

InstallShield Express 2022 R2 では Visual Studio 2022 でプロジェクト作成、編集、アップグレードおよびビルドを行うことができます。

Visual Studio 2022 と InstallShield 2022 R2 との統合についての概要は、つぎのとおりです:

- Visual Studio 2022 内で新しい InstallShield プロジェクトを作成することができます。
- プロジェクトを Visual Studio 2022 内で編集することができます。
- プロジェクトを最新版の InstallShield にアップグレードすることができます。
- Visual Studio 2022 IDE および x64 MSBuild からのプロジェクトをビルドすることができます。

***メモ・** Visual Studio 2022 からの InstallScript プロジェクトのサポートは、InstallShield の将来的なリリースで予定されています。*

InstallShield 2022 R1 の新しい機能

InstallShield 2022 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX でカスタム宣言をサポート

- InstallShield 2022 R1 における Visual Studio 2022 の統合

MSIX でカスタム宣言をサポート

MSIX プロジェクトにカスタム宣言のサポートが追加されました。

MSIX プロジェクトの [宣言] ビューで、多くの組み込みアプリケーションおよびパッケージ拡張タイプがデフォルトでサポートされています。また、カスタム宣言機能を使って、InstallShield では提供されていない MSIX 拡張機能を定義することもできます。[カスタム宣言] メニューでは、名前空間およびコンテンツを指定して必要な拡張タイプを定義できるオプションが提供されています。InstallShield ビルド プロセスは、MSIX マニフェスト ファイルの準備中にこれらのカスタマイズを考慮に入れます。詳細については、InstallShield 製品マニュアルを参照してください。

InstallShield 2022 R1 における Visual Studio 2022 の統合

InstallShield は Visual Studio 2022 との統合をサポートします。この統合は段階的な提供が予定されています。InstallShield 2022 R1 では Visual Studio 2022 でプロジェクトを開き、アップグレードおよびビルドを行うことができます。

Visual Studio 2022 と InstallShield 2022 R1 との統合についての概要は、つぎのとおりです:

- 既存するプロジェクトを Visual Studio 2022 で開くことができます。
- プロジェクトを最新版の InstallShield にアップグレードすることができます
- Visual Studio 2022 IDE および x64 MSBuild からのプロジェクトをビルドすることができます。

***メモ・** Visual Studio 2022 は以前のリリースの 32 ビットとは異なり、64 ビット アプリケーションとしてリリースされているため、段階的な統合が行なわれています。*

***メモ・** InstallScript プロジェクトおよび Visual Studio 2022 から新しい InstallShield プロジェクトを作成できる機能は、InstallShield の将来的なリリースでサポートされます。。*

InstallShield 2021 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2021 R2 の新しい機能
- InstallShield 2021 R1 の新しい機能

InstallShield 2021 R2 の新しい機能

InstallShield 2021 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

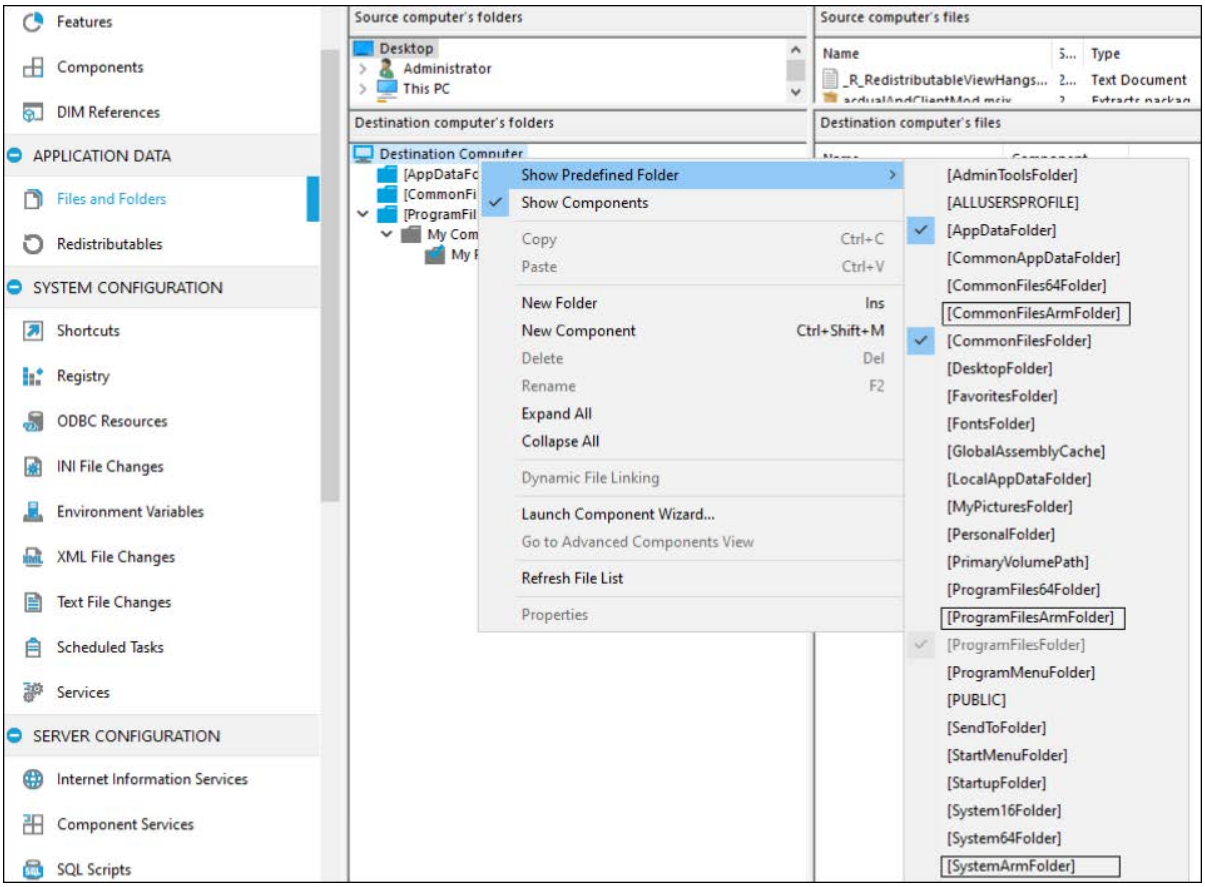
- Windows 11 および Windows Server 2022 のサポート強化
- 完全な ARM サポート

Windows 11 および Windows Server 2022 のサポート強化

Windows 11 および Windows Server 2022 をターゲットに特定してインストール条件を構成することができます。

完全な ARM サポート

ARM64 ベースのデバイスをターゲットとするソフトウェア インストーラで、今回より ARM マシン上で実行中の Windows 10 のネイティブ ARM の場所にファイルをインストールすることができます。InstallShield を使って、ARM 固有のフォルダーを定義済みフォルダーから InstallShield プロジェクトに追加することができます。実行時に ARM 固有のフォルダーにファイルをコピーするためには、Windows Installer バージョン 5.0 が必要で、またテンプレート概要が Arm64 をポイントしなくてはなりません。このため、[一般情報] ビューでスキーマ バージョンを 500、およびテンプレート概要を Arm64 に変更する必要があります。



InstallShield 2021 R1 の新しい機能

InstallShield 2021 R1 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- Cloud License Server (CLS) を使ったライセンシング
- 前提条件における Windows Package Manager のサポート
- MSIX アップデート向けのアプリ インストーラー ファイルのサポート
- シンプルでモダンなインストーラー
- Windows 11 のサポート

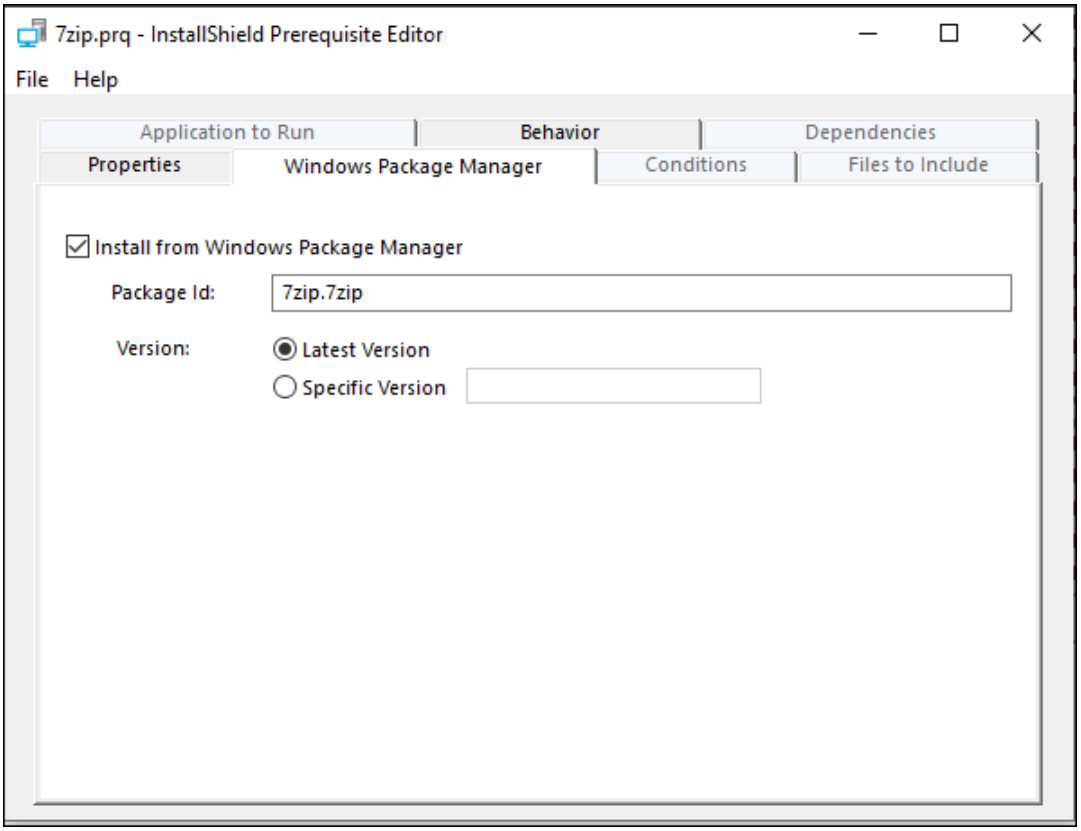
Cloud License Server (CLS) を使ったライセンシング

InstallShield は、Cloud License Server (CLS) を使った同時接続ライセンスを提供します。

Cloud License Server (CLS) は、Revenera FlexNet Cloud Licensing Service でホストされ、InstallShield が使用するライセンス権利を保管します。Cloud License Server (CLS) を使用することで、ローカル ライセンス サーバーをセットアップおよび管理するのに比べて時間と労力が削減されるだけでなく、複数のマシンからライセンスをチェックアウトできる同時接続ライセンスを使用でき、また DepOps ビルド パイプラインからのビルドを簡単に構成することができます。

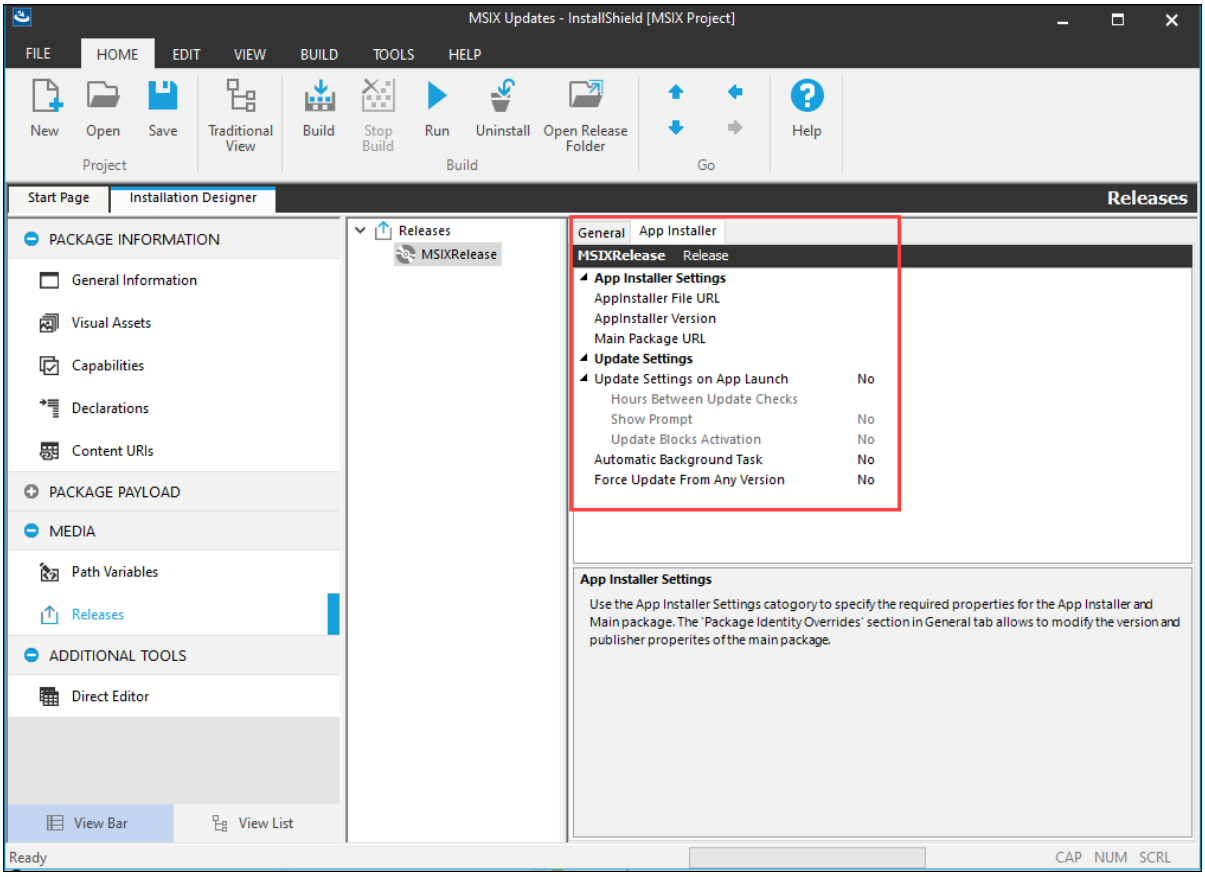
前提条件における Windows Package Manager のサポート

InstallShield 2021 R1 では、InstallShield 前提条件エディターに [Windows Package Manager] タブが追加されています。[Windows Package Manager] タブを使って、製品に前提条件としてインストールしたい Windows Package Manager (WPM) パッケージを指定することができます。



MSIX アップデート向けのアプリ インストーラー ファイルのサポート

今回より、アプリ インストーラー ファイルを使って MSIX アップデートを構成することができます。



シンプルでモダンなインストーラー

[ウィザード インターフェイス] ビューの強化機能として [最小ウィザード ページ] が追加されました。このページを使ってフレームを使わない画面でエンド ユーザーにモダンなインストール/アンインストール エクスペリエンスを提供することができます。

制限

スイート プロジェクトにおける最小ウィザード ページのサポートに関する制限事項:

- 最小ウィザード ページのレイアウトのカスタマイズはサポートされていません。ページおよびコントロールの追加/削除を行うことはできません。
- 最小ウィザード ページ上のボタン クリックのアクションを指定することはできません。
- インストールのパスワード保護は、現在サポートされていません。
- セットアップのインストール中、タスクバーに表示される進行状況は、このリリースではサポートされていません。
- Windows 7 上の InstallShield IDE で最小ウィザード ページをプレビューすることはできません。

Windows 11 のサポート

InstallShield 2021 R1 で作成されたセットアップは、Windows 11 バージョン上で実行可能です。

InstallShield 2020 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2020 R3 SP1 の新しい機能
- InstallShield 2020 R3 の新しい機能
- InstallShield 2020 R2 の新しい機能
- InstallShield 2020 の新しい機能

InstallShield 2020 R3 SP1 の新しい機能

- InstallShield 前提条件で MSIX パッケージおよびバンドルをサポート

InstallShield 前提条件で MSIX パッケージおよびバンドルをサポート

InstallShield 前提条件エディターは、次の MSIX パッケージおよびバンドルをサポートします。

- InstallShield 前提条件に MSIX パッケージおよびバンドルを追加または構成する。
- 検出条件に基づいて、選択された MSIX パッケージまたはバンドルをインストールする。

InstallShield 2020 R3 の新しい機能

InstallShield 2020 R3 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- InstallShield Azure DevOps ビルド拡張
- [ファイルとフォルダー] ビューのショートカット オプション
- MSIX フォント

InstallShield Azure DevOps ビルド拡張

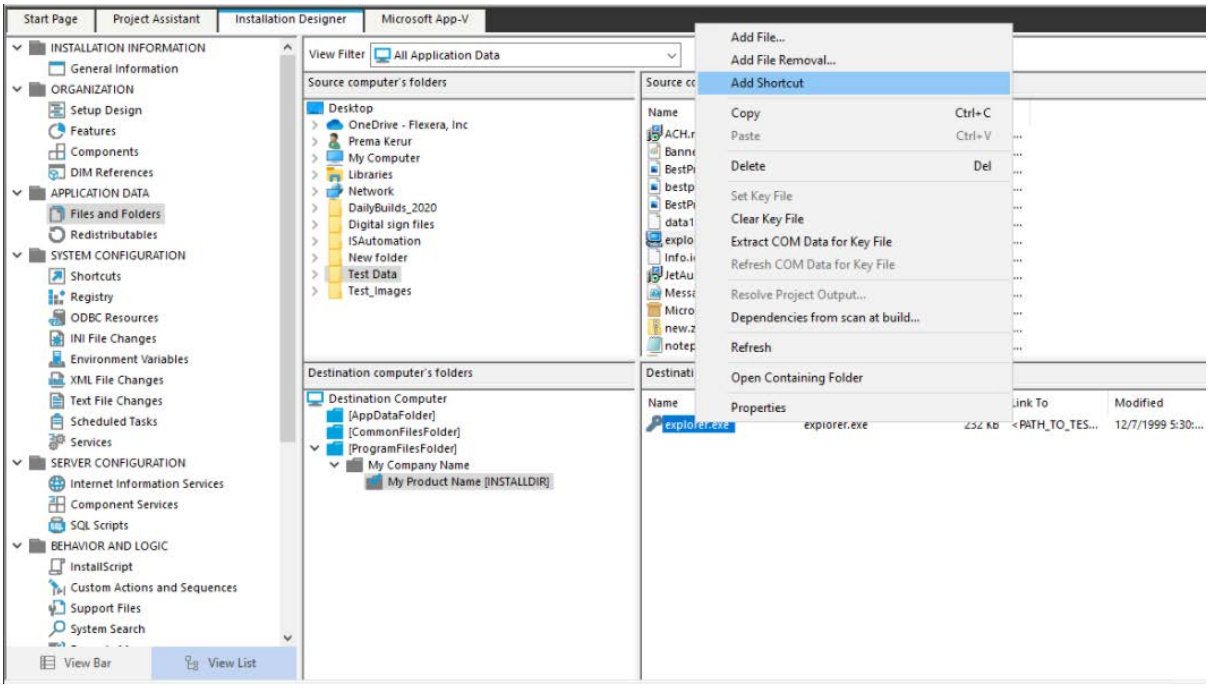
InstallShield 2020 R3 では、Azure DevOps Pipelines で InstallShield プロジェクトをビルドするビルトイン拡張が追加されました。タスクを構成して InstallShield プロジェクトをビルドするには、「InstallShield Azure DevOps Build Extension (InstallShield Azure DevOps ビルド拡張)」ナレッジベース記事 (英語) を参照してください。

[ファイルとフォルダー] ビューのショートカット オプション

基本の MSI プロジェクトで、[ファイルとフォルダー] ビューにショートカットの作成オプションが追加されました。

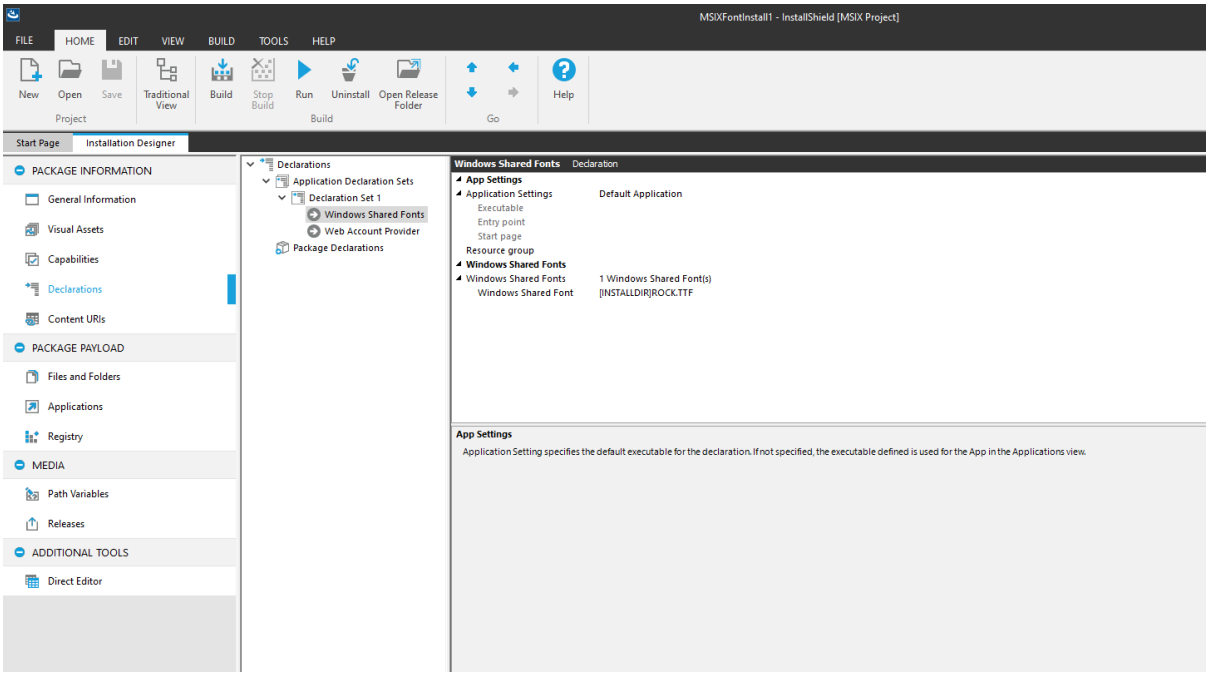
[ファイルとフォルダー] ビューからショートカットを作成するには、次の手順に従います。

1. 実行可能ファイルを右クリックします。
2. [ショートカットの追加] オプションをクリックします。



MSIX フォント

InstallShield では、MSIX プロジェクトで「Windows 共通フォント」宣言がサポートされています。この宣言は、[宣言] ビューから MSIX プロジェクトに追加することができ、カスタム フォントをインストールし、システム上のその他のアプリケーションとそれを共有する事ができます。



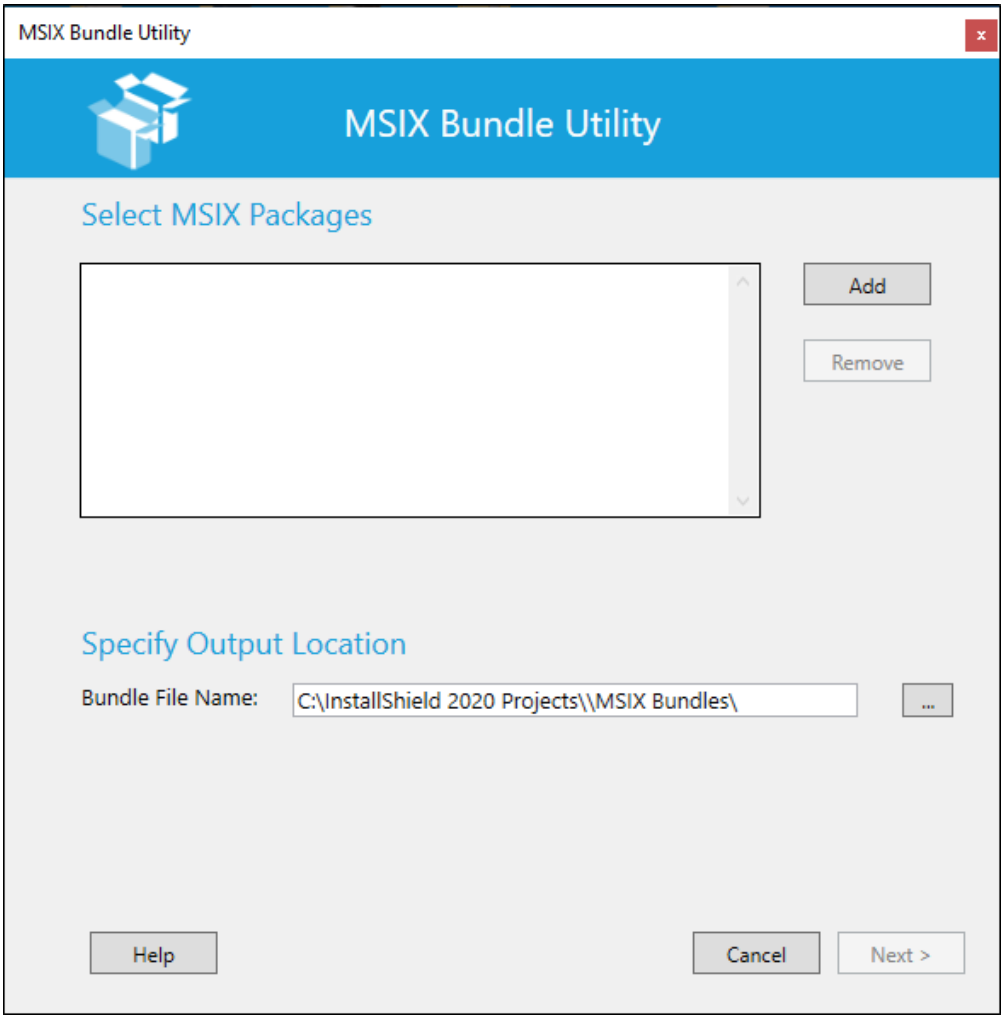
InstallShield 2020 R2 の新しい機能

InstallShield 2020 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX バンドル
- MSIX – サービスのサポート
- スイート/アドバンスト UI 向けの MSIX バンドルまたは AppX バンドル パッケージ

MSIX バンドル

新しい MSIX バンドル ユーティリティを使って MSIX バンドルを作成できます。アーキテクチャ固有のパッケージをユーティリティに追加し、署名情報を入力して MSIX バンドルを作成します。



MSIX – サービスのサポート

今回より、新しい宣言をパッケージに追加することで、サービスを含む MSIX パッケージを作成できるようになりました。また、基本の MSI プロジェクトに既存するサービスを MSIX パッケージに変換することもできます。

スイート/アドバンスト UI 向けの MSIX バンドルまたは AppX バンドル パッケージ

MSIX バンドルを作成できる機能に加えて、それをスイート/アドバンスト プロジェクトに追加し、他のパッケージと共にインストールすることも可能になりました。

InstallShield 2020 の新しい機能

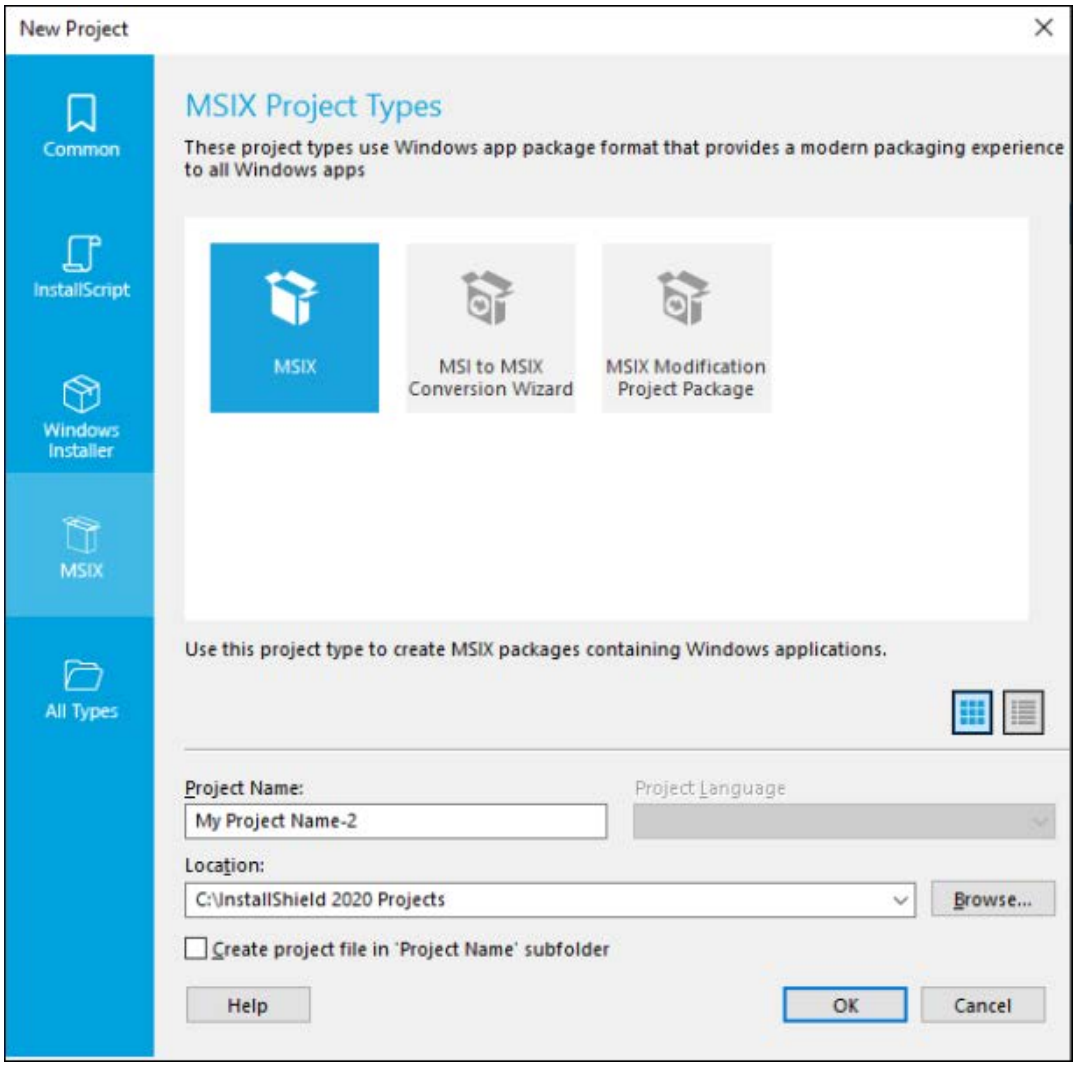
InstallShield 2020 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX 変更パッケージ
- MSIX コアのサポート
- ピュア 64 ビット インストーラー
- AWS CloudHSM ベースのデジタル署名をサポート

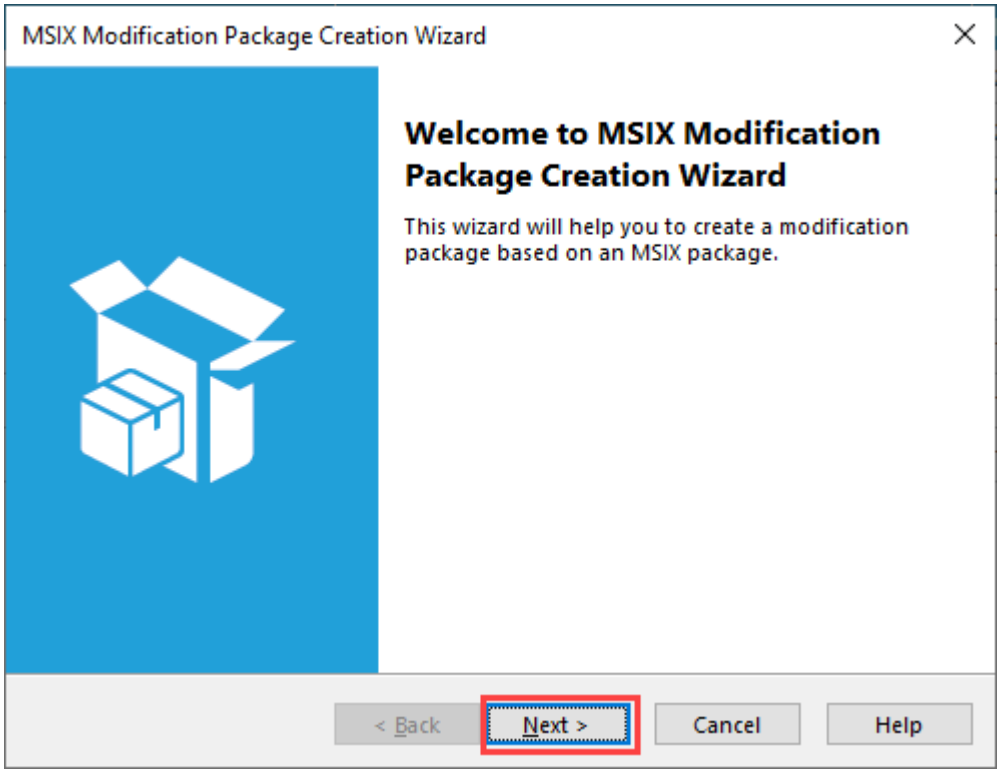
MSIX 変更パッケージ

今回より InstallShield を使って、任意の MSIX パッケージの変更パッケージを作成することができます。次の方法で変更パッケージを作成します:

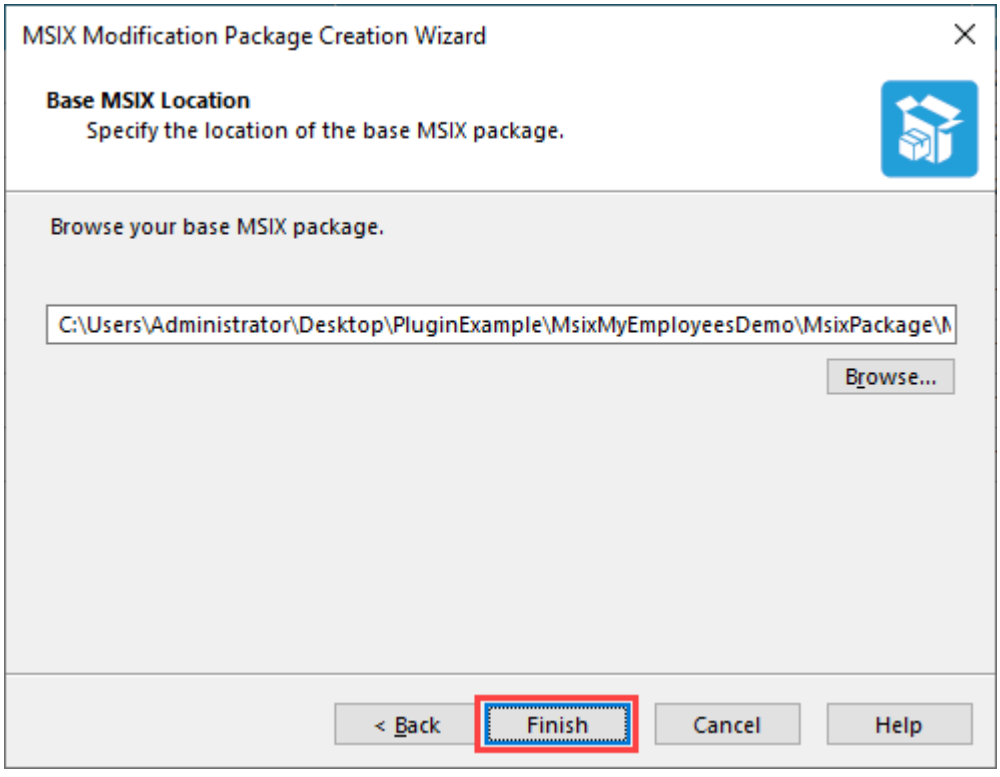
1. [新規プロジェクト] ダイアログから新しい変更パッケージを作成する。



2. MSIX 変更パッケージ作成ウィザードで [次へ] をクリックします。



3. ベース MSIX パッケージを参照します。



4. [ファイルとフォルダー] ビューで、更新パッケージに追加したい新しいファイルをすべて追加します。
5. [レジストリ] ビューでは、変更パッケージに含める任意のエントリを追加または変更することができます。

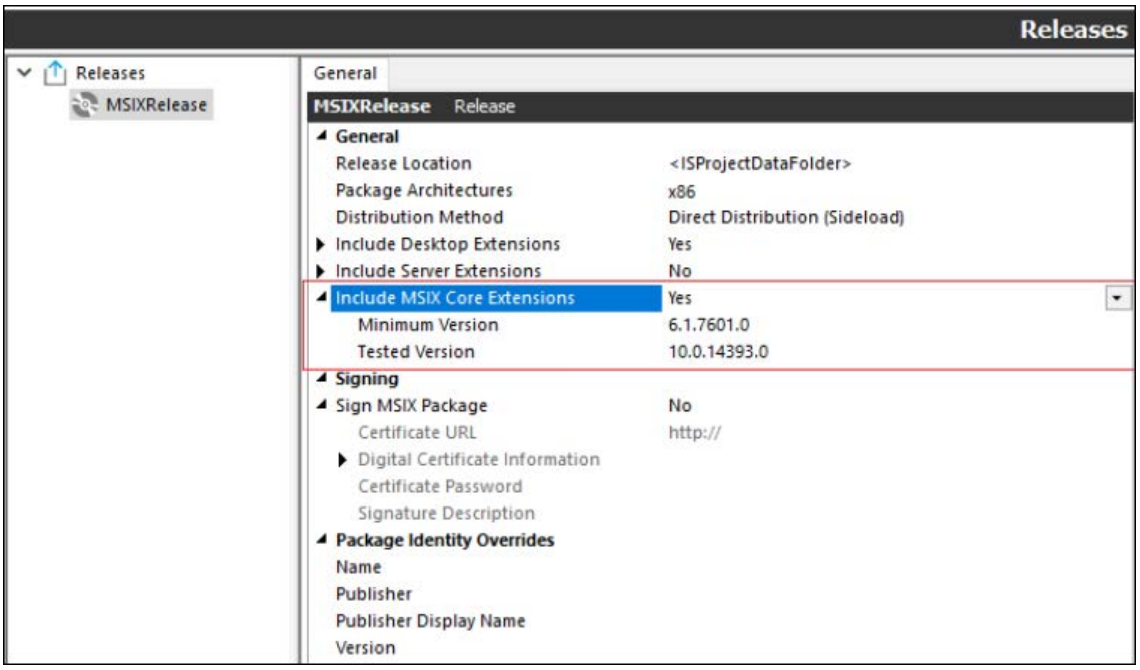
6. [リリース] ビューでは、デジタル署名情報を追加します。

7. プロジェクトをビルドします。
これは変更パッケージを生成します。

MSIX コアのサポート

InstallShield 2020 R1 より、Windows 7 および Windows 10 の古いバージョン (1709 以前) をターゲットとする MSIX パッケージを作成およびインストールすることができます。さらに、サポートされているすべてのプラットフォームに MSIX パッケージをインストールする単一のスイート インストーラーを作成することも可能です。

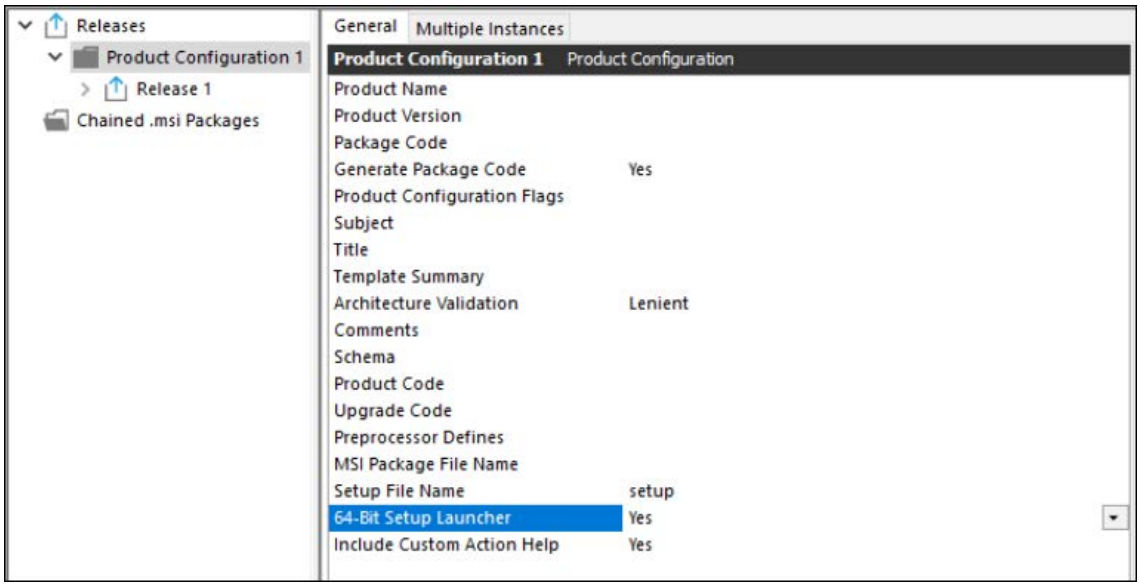
基本の MSI プロジェクトまたは MSIX プロジェクトで MSIX パッケージを作成するとき、[リリース] ビューの値を次のように設定してください:



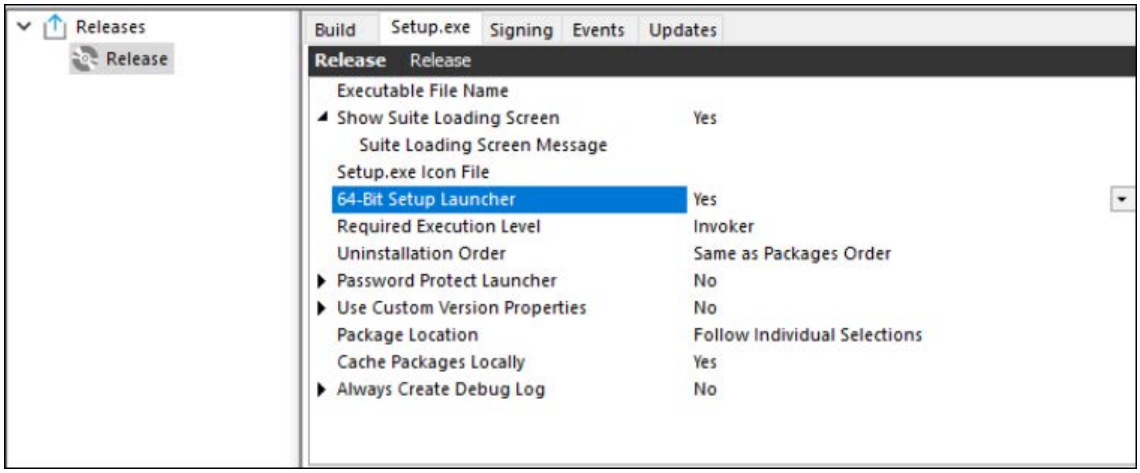
ピュア 64 ビット インストーラー

2020 R1 より、基本の MSI プロジェクトおよびスイート プロジェクトで 64 ビット ランチャーを使ったインストーラーを作成することができます。

1. 基本の MSI プロジェクトでピュア 64 ビット インストーラーを作成するには、[製品の構成] ビューに移動して [64 ビット セットアップ ランチャー] を選択します。



2. スイート プロジェクトでピュア 64 ビット インストーラーを作成するには、[リリース] ビューに移動して [64 ビット セットアップ ランチャー] を選択します。



AWS CloudHSM ベースのデジタル署名をサポート

InstallShield を使って、今回より AWS CloudHSM ベースのデジタル署名を使ってインストーラーにデジタル署名を行うことができます。この機能を有効化するには、次のプロパティを <<InstallShield_Location>/Support/<0409¥0411>Settings.xml ファイルに追加してください。

```
<!-- Specify Platform = X86 | X64 for Digital Signing -->
<DigitalSignature Platform="X64"/>
```

InstallShield 2019 の新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

- InstallShield 2019 R3 の新しい機能
- InstallShield 2019 R2 の新しい機能
- InstallShield 2019 の新しい機能

InstallShield 2019 R3 の新しい機能

InstallShield 2019 R3 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- Docker コンテナ上の Standalone Build
- ファイル リダイレクト fixup のサポート
- OLE DB Driver for SQL Server の使用
- データベース サーバーの SQL ドライバーを選択する
- オートメーション インターフェイスを使って [パッチ デザイン] のパッチを構成できる機能

Docker コンテナ上の Standalone Build

今回より、InstallShield Standalone Build (SAB) を Docker コンテナ内にインストールおよび実行することが可能です。

Docker Hub 上に InstallShield 2019 R3 Standalone Build をセットアップする方法について、または Docker 上にセットアップ済みの InstallShield 2019 R3 Standalone Build をダウンロードする方法については、「InstallShield StandAlone Build with Docker (Docker を使った InstallShield スタンドアロン ビルド)」ナレッジベース記事に掲載されています。

ファイル リダイレクト fixup のサポート

以前のリリースで、InstallShield に MSIX のファイル リダイレクトおよびカスタム fixup を含むパッケージ サポート フレームワークのサポートが追加されました。

InstallShield 2019 R3 リリースでは、ファイル リダイレクト fixup パラメーターの実行可能ファイル名をカスタマイズすることができるように機能が強化されています。この強化は、複数プロセスの起動を行う実行可能ファイルに適用します。

以前のリリースでは、リダイレクト fixup パラメーターの実行可能ファイルは、プライマリ プロセス名と同じ名前が使用されました。InstallShield 2019 R3 には、MSIX プロジェクト タイプの **[アプリケーションの設定]** ビューにある **[パッケージ サポート フレームワーク]** 領域に、新しい **"実行可能ファイル"** フィールドが追加されました。ここでリダイレクト fixup パラメーターの実行可能ファイル名を指定することができます。デフォルトで、値としてプライマリ プロセス名が入力されます。

OLE DB Driver for SQL Server の使用

以前の Microsoft OLE DB Provider for SQL Server (SQLOLEDB) および SQL Server Native Client OLE DB provider (SQLNCLI) は、引き続き使用できません。

現在、Microsoft から Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) と呼ばれる新しい OLE DEB プロバイダーがリリースされています。

InstallShield 2019 R3 では今回より、Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) を使って TLS 1.2 のみの環境をサポートします。

***上級者向けメモ・** Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) は、InstallShield 2019 R3 に前提条件として含まれています。*

データベース サーバーの SQL ドライバーを選択する

InstallShield 2019 R3 では今回より、使用するデータベース サーバー用のドライバー一覧から、ドライバーを選択することができます。

デフォルトで、新しい SQL 接続には、[要件] タブで Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) が表示されます。

オートメーション インターフェイスを使って [パッチ デザイン] のパッチを構成できる機能

以前は、オートメーション インターフェイスを使って [パッチ ザイン] のパッチを構成することができませんでした。オートメーションでは、既に作成済みのパッチのみをビルドすることができました。

InstallShield 2019 R3 リリースから、[パッチ デザイン] のパッチをオートメーション インターフェイスを使って構成することができます。

InstallShield 2019 R2 の新しい機能

InstallShield 2019 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX – パッケージ サポート フレームワーク

MSIX – パッケージ サポート フレームワーク

パッケージ サポート フレームワークは、ソース コードにアクセスできない場合でも既存する win32 アプリケーションに修正を適用できるキットなので、MSIX コンテナ内でこれを実行することができます。

これを使って、アプリケーションにモダンな実行環境のベスト プラクティスを適用できます。パッケージ サポート フレームワークには、実行可能ファイル、ランタイム マネージャー DLL、およびランタイム修正プログラムのセットが含まれます。

ユーザーがアプリケーションを起動すると、パッケージ サポート フレームワーク ランチャーが最初に実行されます。ランチャーによって構成ファイルが読み取られ、ランタイム修正プログラムとランタイム マネージャー DLL がアプリケーション プロセスに挿入されます。ランタイム マネージャーは、MSIX コンテナー内で実行するアプリケーションで必要になったときに修正プログラムを適用します。

InstallShield 2019 R2 では、ファイル リダイレクト fixup およびカスタム fixup などのランタイム修正を含むパッケージ サポート フレームワークを MSIX で使用できます。

- ファイル リダイレクト fixup – ファイル リダイレクト fixup を使って、MSIX コンテナ内で実行するアプリケーションからアクセスできないディレクトリ内にデータを書き込みまたは読み取りしようとする場合にリダイレクトを行います。
- カスタム fixup – カスタム リダイレクト fixup を使って、アプリケーションに適用する修正を指定する構成ファイルとアプリケーションの互換性問題を解決できます。

InstallShield 2019 の新しい機能

InstallShield 2019 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- MSIX プロジェクト タイプを使って MSIX パッケージをビルドする
- MSI からMSIX への変換ウィザード
- 基本の MSI の MSIX 検証スイートをビルド
- スイートへの MSIX の追加
- 新しいプロジェクトウィザード

MSIX プロジェクト タイプを使って MSIX パッケージをビルドする

MSIX は Windows アプリ パッケージ形式で、すべての Windows アプリにモダンなパッケージ エクスペリエンスを提供します。MSIX パッケージ形式は、既存アプリ パッケージおよび/またはファイルのインストール機能をはそののままに、Win32、WPF、および WinForm アプリに新しくモダンなパッケージ作成およびデプロイメント機能を追加します。

InstallShield 2019 では、 MSIX プロジェクト タイプを使って MAIX パッケージをビルドすることができます。サイドロードまたは Windows ストア経由で配布を行う Windows アプリケーションが含まれます。

- パッケージ情報
- パッケージ ペイロード
- メディア

パッケージ情報

今回より、次を使ってプロジェクトを説明および識別することができます:

- **[一般情報]** ビューでは、プロジェクト ファイルの名前やフォーマットなどの詳細を指定します。
- **[MSIX ビジュアル資産]** エクスプローラーでは、統合的に視覚的な方法で、MSIX アプリのビジュアル的な側面を説明することができます。
- **[アプリ機能]** ビューは、特定の API または画像、音楽、またはカメラやマイクなどのデバイスといったリソースにアクセスできるよう、アプリのパッケージ マニフェスト内で有効化さている必要があります。
- **[宣言]** ビューは、アプリケーション、パッケージ レベルの宣言を作成および管理できるビジュアル ツールを提供し、それらのプロパティを構成することができます。
- **[コンテンツ URI]** ビューでは、window.external.notify を使って Script-Notify イベントをアプリに送る URI を指定します。

パッケージ ペイロード

今回より、MSIX アプリケーションをデザインするファイルを指定することができます:

- **[ファイルとフォルダー]** ビューで、InstallShield プロジェクトにファイルを追加することができます。ターゲット上にあるフォルダーへこれらのファイルを整理することができます。
- **[アプリケーション]** ビューを使って、パッケージで提供される機能の一部またはすべてを構成する Windows アプリのプロパティを視覚的および統合的にデザインすることができます。
- **[レジストリ]** ビューでは、インストールが作成するレジストリキーおよびレジストリ値を定義することができます。

メディア

今回より、インストールを配布するのに利用するファイルとフォルダーをカスタマイズできます:

- **[パス変数]** ビューでは、変数を使って開発システム間で簡単にインストールを移動させることができます。
- **[リリース]** ビューには、プロジェクトに含まれる各製品のビジュアルツールがあります。

MSI からMSIX への変換ウィザード

MSI からMSIX への変換ウィザードを使って、既存の Windows Installer ファイル（.msi ファイル）を基本の MSI プロジェクトを通して MSIX パッケージに変換することができます。

基本の MSI の MSIX 検証スイートをビルド

今回より、Microsoft MSIX 標準に準拠する必要がある既存の InstallShield プロジェクトを識別することができます。また、MSIX で動作するために必要な MSI プロジェクト要素の変換に関する情報や手順も提供します。

今回より InstallShield には、InstallShield MSIX 適合性スイートという名前の 1 セットの検証ツールが含まれています。

このスイートに含まれる InstallShield MSIX 適合性検証ツール は、.msi パッケージ内で MSIX パッケージ (..msix) フォーマットに適さないアイテムの存在をスキャンします。

スイートへの MSIX の追加

InstallShield 2019 では、スイートに MSIX パッケージを追加して、他のデプロイメント パッケージ タイプと共にインストールすることが可能です。

新しいプロジェクトウィザード

新しいプロジェクト ウィザードを使って、異なるプロジェクト タイプを容易に選択できます。

新しいプロジェクト ウィザードには、次の異なるタブがあります:

- 良く使うプロジェクト タイプ – よく使用されるプロジェクト タイプを使用します。
- InstallScript プロジェクト タイプ – InstallShield Professional で InstallShield セットアップ エンジンを使ってビルドされたプロジェクトを使用します。
- Windows Installer プロジェクト タイプ – InstallShield で Microsoft Windows Insatller（MSI）セットアップ エンジンを使ってビルドされたプロジェクトを使用します。
- MSIX プロジェクト タイプ – Windows アプリ パッケージ形式を使用します。
- すべてのプロジェクト タイプ – 使用可能なすべてのプロジェクト タイプおよび任意のカスタム テンプレートを使います。

InstallShield 2019 R2 強化機能

InstallShield 2019 R2 には、次の強化機能が含まれています:

- 連続する署名の遅延
- IsCmdBld.exe パラメーターを使ってデジタル証明書パスワードを指定する
- スイート プロジェクトで、ISM パッケージへのリンクに相対パスを使用可能にする
- SQL 2012 Native Client 前提条件のアップデート

- PRQ 条件に Windows Server 2019 を追加

連続する署名の遅延

InstallShield 2019 R2 では、連続するデジタル署名の間に遅延を設定することができます。この機能は、タイムスタンプ サーバーが連続する署名要求の処理に失敗した場合のみ必要になります。

Settings.xml の <DevStudio/Build> ノードの下にある <DelayBetweenSigning default="1500" /> ノードで、ミリ秒単位で指定する必要があります。

InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語バージョンに応じて、次のいずれかの場所にインストールされています:

- 英語—InstallShield Program Files Folder¥Support¥0409
- 日本語—InstallShield Program Files フォルダー¥Support¥0411

IsCmdBld.exe パラメーターを使ってデジタル証明書パスワードを指定する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*
- *アドバンスト UI*

InstallShield 2019 R2 には、デジタル証明書パスワードを指定するための新しいパラメーター (cert_password) が含まれています。これはオプション パラメーターで、コマンドライン ビルド (IsCmdBld.exe) で使用できます。

指定されなかった場合、ファイルの署名にはプロジェクトで構成されたパスワードが使用されます。

スイート プロジェクトで、ISM パッケージへのリンクに相対パスを使用可能にする

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

InstallShield 2019 R2 では今回より、変更済みのプロジェクトをビルドするとき、相対パスを使って ISM パッケージにリンクすることができます。

SQL 2012 Native Client 前提条件のアップデート

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

- *InstallScript MSI*

InstallShield 2019 R2 には、最新バージョンの 2012 Native Client 用の Microsoft SQL Server 2012 Native Client 前提条件（x86 および x64）が含まれています。

PRQ 条件に Windows Server 2019 を追加

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します：*

- *基本の MSI*
- *InstallScript*

InstallShield 2019 R2 では、前提条件の条件を Windows Server 2019 の特定のプラットフォーム上で実行するように設定できます。オプションを選択/選択解除して、前提条件エディターで適切な PRQ 条件を設定することができます。

InstallShield 2019 の強化機能

InstallShield 2019 には、以下の強化機能が含まれています:

- 選択された証明書の詳細を表示する
- オペレーティング システム要件として Windows Server 2019 を追加
- 定義済みインストール条件の追加
- デジタル署名におけるタイムスタンプ サーバーの更新
- MSBuild を使ったスイート プロジェクトのサポート
- デバッグ ログ ファイル名のカスタマイズ
- カスタム リターン コードのカスタマイズ
- セットアップ ファイル名のカスタマイズ
- InstallShield Standalone Build のサポート

選択された証明書の詳細を表示する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します：*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*
- *QuickPatch*
- *MSIX*

InstallShield 2019 では、証明書の全般的な情報や、セキュリティ、証明書パスなどの情報が **[証明書の選択]** ダイアログ ボックスの **[詳細表示]** オプションに一覧表示されます。

オペレーティング システム要件として Windows Server 2019 を追加

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2019 では、オペレーティング システム要件セクションに Windows Server 2019 のオプションを追加できるようになりました。オプションを選択/選択解除して、プロジェクトに適切なインストール条件を設定することができます。

定義済みインストール条件の追加

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

InstallShield に新しい定義済みシステム検索が追加されました:

- Microsoft .NET Framework 4.6.2

インストールで上記が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、このシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

デジタル署名におけるタイムスタンプ サーバーの更新

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield 2019 で

- SHA-1 を使ってパッケージに署名を行うと、パッケージは次を使ってタイムスタンプを取得します:

  <DigitalSignature Timestamp="http://timestamp.verisign.com/scripts/timstamp.dll"/>

- SHA-256 を使ってパッケージに署名を行うと、パッケージは次を使ってタイムスタンプを取得します:

  <DigitalSignature TimestampRFC3161="http://sha256timestamp.ws.symantec.com/sha256/timestamp"/>

MSBuild を使ったスイート プロジェクトのサポート

***プロジェクト・**この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

InstallShield では、.NET Framework に含まれている Microsoft ビルド エンジン (MSBuild) がサポートされています。MSBuild サポートを使って、Visual Sutudio がビルド ラボ環境にインストールされていない場合でも、InstallShield で Visual Studio ソリューションをビルドすることができます。

InstallShield 2019 では今回より、MSBuild を使用することで、Visual Studio で作成されたスイート プロジェクト (.sln) をビルドすることができます。

デバッグ ログ ファイル名のカスタマイズ

***プロジェクト・**この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで **[リリース]** ビューの **[Setup.exe]** タブに新しいオプション **[常にデバッグ ログを作成する]** が追加されました。

InstallShield 2019 では今回より、生成されるデバッグ ログ ファイル名を指定することができます。

カスタム リターン コードのカスタマイズ

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *InstallScript*

InstallShield 2019 では今回より、実行時に Setup.exe のリターン コードを独自のカスタム値で上書きすることができます。

セットアップ ファイル名のカスタマイズ

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *InstallScript*
- *スイート/アドバンスト UI*

InstallShield 2019 では、Setup.exe の元のファイル名プロパティをカスタマイズすることができます。

InstallShield Standalone Build のサポート

今回より、InstallShield Standalone Build をインストールした場合、オートメーション インターフェイス機能がデフォルトで選択されています。

これによって、手動で選択せずに、オートメーション インターフェイス機能をインストールすることができます。

追加された前提条件

InstallShield 2019 には、以下の前提条件が追加されています:

- Microsoft .Net Core 2.1 ランタイム 前提条件

Microsoft .Net Core 2.1 ランタイム 前提条件

InstallShield の [再配布可能ファイル] ビューには、今回より Microsoft .Net Core 2.1 ランタイム 前提条件が追加されています。

InstallShield 2018 R2 の新しい機能

InstallShield 2018 R2 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- InstallShield IDE を使った MSIX パッケージのビルド

InstallShield IDE を使った MSIX パッケージのビルド

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 R2 では、InstallShield IDE を使って MSIX パッケージをビルドすることができます。

InstallShield 2018 SP1 の新しい機能

InstallShield 2018 SP1 で追加された強化機能についての詳細は、次のセクションを参照してください:

- スイート/アドバンスト UI インストールでダウンロード可能なアップデートを構成する
- SHA1 & SHA256 を使用するデュアル署名のサポート
- スイート読み込み画面のメッセージを構成する
- アップデート ランチャー名のカスタマイズ
- PowerShell カスタム アクションで文字スペースを指定する
- 定義済みパブリック フォルダーの指定

スイート/アドバンスト UI インストールでダウンロード可能なアップデートを構成する

InstallShield 2018 SP1 では今回より、ソフトウェアのダウンロード可能なアップデートを有効化および構成することができます。ユーザーはより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行することができます。

- ダウンロード可能なアップデートのサポートを有効化する
- ダウンロード可能なアップデートのより新しいバージョンを構成する
- 確認メッセージを構成する
- ダウンロード可能なアップデートのサポートを追加する

ダウンロード可能なアップデートのサポートを有効化する

今回より、エンド ユーザーがアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのダウンロード可能なアップデートのサポートを確認できるようにアップデートを有効化することができます。

ダウンロード可能なアップデートのより新しいバージョンを構成する

- エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するかどちらかのオプションを受け取ります。
- エンド ユーザーはスイート パッケージ (使用可能な場合) の新しいバージョンを強制的にダウンロードおよびインストールすることができます。

確認メッセージを構成する

エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するためのメッセージを入力できるようになりました。

新しいメッセージを作成するか、文字列一覧からローカライズ済みのメッセージを選択できます。

***メモ・** より新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するための [はい] または [いいえ] オプションを表示する確認メッセージがエンド ユーザーに表示されます。*

ダウンロード可能なアップデートのサポートを追加する

以前のリリースでは、インストーラーの **[Setup.exe]** タブでターゲット システムへのダウンロードを可能にするアップデート セットアップ ランチャーのパスとして今後使用する、アドバンスト UI およびスイート/アドバンスト UI インストールの**絶対パス**を入力しました。

InstallShield 2018 SP1 では、アドバンスト UI またはスイート/アドバンスト UI インストールと同じ**絶対パス**を、インストーラーの **[アップデート]** タブに入力することができます。

SHA1 & SHA256 を使用するデュアル署名のサポート

以前のリリースでは、署名ダイジェスト ハッシュ アルゴリズムの選択肢は次に限られていました:

- 証明書ハッシュ
- SHA-1
- SHA-256

InstallShield 2018 SP1 では、これらの署名ダイジェスト ハッシュ アルゴリズム以外にもデュアル署名 – (SHA-1 および SHA-256) ダイジェストを選択できます。

***メモ・** 両方のダイジェスト (SHA1 およびSHA256) の使用は、.msi のデジタル署名でサポートされていません。*

スイート読み込み画面のメッセージを構成する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート読み込み画面にメッセージ追加することができます。

スイート読み込み画面メッセージの長さは 35 文字までです。

アップデート ランチャー名のカスタマイズ

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、アップデート ランチャーの名前をカスタマイズするための新しい設定が追加されています。InstallShield はアップデート ランチャーの名前として、デフォルトで Update.exe を使用します。今回より、名前を指定してアップデート ランチャーを作成することができます。

PowerShell カスタム アクションで文字スペースを指定する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield では、PowerShell カスタム アクションで前後に移動させる場合の文字スペース数を指定するための新しい設定が追加されています。

定義済みパブリック フォルダーの指定

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、ユーザーのパブリック フォルダーへの完全パスを保持する新しい定義済みフォルダーが追加されました。

InstallShield 2018 の新しい機能

InstallShield 2018 で追加された新しい機能についての詳細は、次のセクションを参照してください:

- スイート プロジェクトでパッケージのアンインストール順を指定する
- 最小 UI を使ったスイート インストールを実行するメソッド
- 条件付きで実行時に機能の表示/非表示を設定する
- 再帰的または非再帰的な IIS 登録を行う
- Web アプリケーションのフォーム認証を設定する
- アプリケーション プール エンティティのユーザー プロファイルを読み込むかどうかを制御する新しいオプション
- トランスフォーム プロジェクトにプロセスの強制終了および PowerShell カスタム アクションを追加する
- 強化機能

スイート プロジェクトでパッケージのアンインストール順を指定する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、**[リリース]** ビューの **[Setup.exe]** タブにある新しい **"アンインストール順"** プロパティを使って、スイート プロジェクトに含まれるパッケージのアンインストール順を指定することができます。

この設定を使って、次のオプションから一つを選択し、スイート プロジェクトに含まれるパッケージのアンインストール順を指定できます。

- **パッケージ順と同じ** – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
- **パッケージ順と逆** – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

オートメーション インターフェイスにおける新しい UninstallOrder メソッド

オートメーション インターフェイスで UninstallOrder メソッドを使って **"アンインストール順"** プロパティを設定することができます。次の値のうち 1 つを指定します。

- euoForward(0) – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
- euoReverse(1) – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

最小 UI を使ったスイート インストールを実行するメソッド

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、進行状況パネルを表示するのみの最小 UI モードでスイートインストールを実行する新しいコマンドライン パラメーターを使用できます。

スイート インストールを最小 UI モードで実行するには、コマンドラインで /passive パラメーターを使用します:

Setup.exe /passive

最小 UI モードを使ってアンインストールするには、次のコマンドを使用します :

Setup.exe /passive /remove

条件付きで実行時に機能の表示/非表示を設定する

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *アドバンスト UI*
- *スイート/アドバンスト UI*

以前のリリースでは、アドバンスト UI またはスイート/アドバンスト UI インストールで機能の **"表示"** プロパティを **[はい]**または **[いいえ]** に設定して、インストーラーの **InstallationFeatures** ウィザード ページで表示するかどうかを指定することができました。

InstallShield 2018 では、インストール デザイナーの **[機能]** ビューで **"表示"** プロパティの下にある **[条件]** オプションを使って、実行時のプロパティに基づいて条件付きで機能を表示または非表示することができます。

"条件" 設定を使って、**InstallationFeatures** ウィザード ページで機能をインストール可能としてデフォルトで表示するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。

たとえば、Windows の特定バージョンが搭載されたターゲット システム上では特定の機能をデフォルトで表示する場合、その Windows バージョンを指定する条件を作成します。

タスク

アドバンスト UI またはスイート/アドバンスト UI インストールで条件付きで機能を表示する場合、以下の手順に従います:

1. **[機能]** ビューで **"表示"** プロパティの下にある **[条件]** 行をクリックします。緑色の + 記号 (**[新しい条件]** ボタン) が行の終わりに表示されます。
2. **[新しい条件]** ボタンをクリックします。**[条件]** 行の下に新しい行が追加されます。
3. **[新しい条件]** ボタンの横にある下矢印をクリックして、一覧から適切なオプション (**[All]**、**[Any]**、または **[None]**) を選択します。
4. 次に同じ行で、**[新しい条件]** ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。

1 つ以上の条件ステートメントが構成されると、**"条件"** プロパティには **(条件)** と表示されます。何も構成されていない場合、**"条件"** プロパティには **(空白)** と表示されます。

条件付きの表示をサポートするための、オートメーション インターフェイスに追加された新しいメソッド

実行時に条件付きで機能の表示を設定するため、次の新しいメソッドがオートメーション インターフェイスに追加されました:

メソッド	構文
AddVisibleCondition	AddVisibleCondition() As ISWiSuiteCondition
DeleteVisibleCondition	DeleteVisibleCondition()
VisibleCondition	読み取り専用オブジェクト プロパティ

再帰的または非再帰的な IIS 登録を行う

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューの **"アプリケーション"** 設定に **[ASP.NET 登録]** という名前の新しいオプションが追加され、再帰的または非再帰的な ASP .NET 登録を行うことができます。この機能を使って、同じ Web サイトに ASP.NET アプリケーションおよび ASP.NET コア アプリケーションの両方をインストールすることができます。

ASP.NET アプリケーション登録オプションを IIS (インターネット インフォメーション サービス) を使って設定するには、**"ASP.NET 登録"** プロパティを次のオプションの 1 つに設定します:

- **再帰的** – 指定されたアプリケーションおよびすべてのサブアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。
- **非再帰的** – 特定のアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。サブアプリケーションは変更されません。

Web アプリケーションのフォーム認証を設定する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 に、Web アプリケーションでフォーム認証を設定するための新しいオプションが追加されました。この新しいオプション **[フォーム認証]** は、Web サイトの **[IIS 構成]** ビューで **[認証済みアクセス]** セクションの下に表示されます。

[フォーム認証] オプションを **[はい]** に設定してフォーム認証を有効化します。ASP.NET フォーム ベースの認証は、多くの要求を受け取るパブリック Web サーバー上にあるサイトまたはアプリケーションに適しています。この認証モードでは、オペレーティング システムが提供している認証方法に頼ることなく、アプリケーション レベルでのクライアント登録および認証を管理することができます。

重要・*フォーム認証は、Web サーバーにユーザー名とパスワードをテキスト形式で送信します。ホームページ以外のアプリケーション内のすべてのページ、およびログオン ページには、Secure Sockets Layer (SSL) 暗号化を使用してください。*

アプリケーション プール エンティティのユーザー プロファイルを読み込むかどうかを制御する新しいオプション

プロジェクト・*この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 では、アプリケーション プール エンティティのユーザー プロファイルをロードするかどうかを制御する新しい **"アプリケーション プール"** 設定が **[IIS 構成]** ビューに追加されました。

"ユーザー プロファイルの読み込み" プロパティに次のオプションの 1 つを設定します:

- **はい** – IIS がアプリケーション プールのユーザー プロファイルをロードします。
- **いいえ** – IIS がアプリケーション プールのユーザー プロファイルをロードしません。これは IIS 6.0 のときと同じ動作です。

トランスフォーム プロジェクトにプロセスの強制終了および PowerShell カスタム アクションを追加する

プロジェクト・*この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *トランスフォーム*

以前のリリースでは、トランスフォーム プロジェクトにプロセスの強制終了または PowerShell カスタム アクションを追加することができませんでした。InstallShield 2018 では今回より、**[カスタム アクションとシーケンス]** ビューでトランスフォーム プロジェクトに **[新しいプロセスの強制終了]** または **[新しい PowerShell]** カスタム アクションを追加することができます。

強化機能

InstallShield 2018 には、以下の強化機能が含まれています:

- QuickPatch プロジェクトを XML 形式で保存する
- スイート プロジェクトにおける製品名プロパティのローカライズ
- 製品構成のセットアップ ファイル名にプロパティの値を含める
- 概要情報ストリーム コメントおよびパッケージ ファイル名を設定するための新しい MSBuild パラメーター
- テキスト ファイルの変更で改行およびタブ文字の指定
- スイート読み込み画面を削除または非表示にする
- スイート インストールで常にデバッグ ログを作成する設定
- 実行時に SSL 証明書の IIS 証明書ファイルを設定する新しい簡易ダイアログ
- XML ファイルで新しい子要素を作成するときに絶対パスまたは相対パスを指定する
- スイート プロジェクトで、ダイアログ ボックス コントロールのデフォルト キーボード フォーカスを設定する
- 基本の MSI プロジェクトの PowerShell スクリプト エディター
- InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開く新しいオプション
- 追加された前提条件

QuickPatch プロジェクトを XML 形式で保存する

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *QuickPatch*

InstallShield 2018 では、QuickPatch プロジェクトを XML 形式で保存することができます。また XML 形式で保存されているプロジェクトから QuickPatch プロジェクトを作成することもできます。以前のリリースでは、QuickPatch プロジェクト はバイナリ形式でのみ保存することが可能でした。

スイート プロジェクトにおける製品名プロパティのローカライズ

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、今回より、スイート プロジェクトで **"製品名"** プロパティのローカライズをサポートします。

スイート プロジェクトで "製品名" プロパティをローカライズする場合、次の手順に従います。

タスク *スイート プロジェクトでプロパティをローカライズするには:*

1. スイート プロジェクトを開いて**[ユーザー インターフェイス > 文字列エディター]** ビューに移動します。
2. スイート プロジェクトでサポートされている各言語の翻訳済みテキストを含む新しい文字列を作成します。例、ID_STRING2
3. **[インストール情報 > 一般情報]** ビューを開きます。
4. **"製品名"** フィールドの横にある参照ボタンをクリックして、**[文字列の選択]** ダイアログ ボックスを開きます。
5. 翻訳済みテキストを含む作成した文字列の名前を選択します。

製品構成のセットアップ ファイル名にプロパティの値を含める

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 では今回より、製品リリース構成のセットアップ名およびパッケージ ファイル名に Property テーブルからのプロパティの値を含めることができます。

たとえば、**[リリース > 製品構成]** ビューの **[全般]** タブにある "セットアップ ファイル名" または "MSI パッケージ ファイル名" フィールドに次の任意のプロパティを入力することができます:

setup[ProductVersion]
setup[CustomVersion]
setup[ProductCode]
setup[ProductCode][ProductVersion]

たとえば、**"セットアップ ファイル名"** フィールドに **setup[ProductVersion]** と入力した場合、セットアップ名が setup14.10.1234.exe となります。

概要情報ストリーム コメントおよびパッケージ ファイル名を設定するための新しい MSBuild パラメーター

InstallShield 2018 で今回追加された MSBuild パラメーターを使って、インストーラーにコメントの追加を設定、およびインストーラーのパッケージ ファイル名の設定を行うことができます。

- 概要情報ストリーム コメントを設定する新しいパラメーター
- パッケージ ファイル名を設定する新しいパラメーター

概要情報ストリーム コメントを設定する新しいパラメーター

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *マージ モジュール*

[一般情報] ビューの **"概要情報ストリーム コメント"** フィールドでインストーラーにコメントを追加することができます。

InstallShield 2018 ではまた、ビルド時にコメントを入力できるオプションも追加されています。MSBuild.exe タスクに SummaryInfoComments という名前の新しいパラメーターが追加されました。これを使って、次の例のようにビルド番号を含めるなど、ビルド時に **[概要情報ストリーム コメント]** を設定することができます。

MSBuild.exe c:¥installers¥Setup.sln /Property:SummaryInfoComments="ここにコメントを入力"

SummaryInfoComments プロパティを使って追加されたコメントは、ビルド済みのインストーラーの **[プロパティ]** ダイアログ ボックスで参照することができます。

パッケージ ファイル名を設定する新しいパラメーター

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

[リリース] ビューで **"製品構成"** フィールドの **[全般]** タブにある **"MSI パッケージ ファイル名"** フィールドで、インストーラーのパッケージ ファイル名を指定できます。

InstallShield 2018 ではまた、パッケージ ファイル名を設定できるオプションも追加されています。MSBuild.exe タスクに MSIPackageFileName という名前の新しいパラメーターが追加されました。これを使って、次の例のようにビルド時にインストーラーのパッケージ ファイル名を設定することができます。

MSBuild.exe c:¥installers¥Setup.isproj /Property:MSIPackageFileName="MySetup"

MSIPackageFileName パラメーターの値を入力するとき、InstallShield が .msi ファイルに使用する (ピリオドまたはファイル拡張子を含まない) ファイル名を入力する必要があります。

テキスト ファイルの変更で改行およびタブ文字の指定

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

ターゲット システム上で実行時に変更するテキスト ファイルの内容について、インストーラーの検索/置換処理を構成することができます。そのためには、**[システム構成 > テキスト ファイルの変更]** ビューを開いて実行時に検索するテキスト ファイルを識別するためのテキスト **[変更セット]** を追加し、さらに検索するテキスト (**検索後の文字列**) および置換するテキスト (**置換後の文字列**) を指定します。

InstallShield 2018 では、テキストの変更セットを追加したとき、改行またはタブを指定するために **"置換後の文字列"** フィールドにエスケープ シーケンスを入力できるようになりました。

文字	エスケープシーケンス
改行	¥r¥n
タブ	¥t

***メモ・** Windows オペレーティング システムの場合、改行には **¥r¥n** の両方を入力する必要があります。*

実行時に検索/置換が行われた場合、**"置換後の文字列"** フィールドの **¥r¥n** が入力された場所に改行が、**¥t** が入力された場所にタブが挿入されます。

これらの文字がエスケープ シーケンスとして認識されるように、**[エスケープ シーケンスの解析]** オプションを **[はい]** に設定する必要があります。

スイート読み込み画面を削除または非表示にする

***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、インストール中にスイート読み込み画面を表示するかどうかを制御する機能が追加されました。

この画面が表示されるかどうかを制御するため、**[リリース]** ビューの **[Setup.exe]** タブに **[スイート読み込み画面を表示]** という名前の新しいプロパティが追加されました。アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート 読み込み画面を非表示にする場合は、このプロパティを **[いいえ]** に設定します。

スイート読み込み画面のメッセージを構成する

InstallShield 2018 では、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート読み込み画面にメッセージ追加することができます。

スイート読み込み画面メッセージの長さは 35 文字までです。

オートメーション インターフェイスにおける新しい ShowSuiteLoadingScreen メソッド

オートメーション インターフェイスで ShowSuiteLoadingScreen メソッドを使って、**[リリース]** ビューの **[Setup.exe]** タブにある **"スイート読み込み画面を表示"** 設定を指定することができます。デフォルト値は True です。

スイート インストールで常にデバッグ ログを作成する設定

***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、コマンドラインで debuglog を渡さずにスイート プロジェクトのログ記録を有効にするオプションを選択できます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで **[リリース]** ビューの **[Setup.exe]** タブに新しいオプション **[常にデバッグ ログを作成する]** が追加されました。

アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのデバッグ ログを常に作成する場合は **[常にデバッグ ログを作成する]** オプションに **[はい]** を選択します。

オートメーション インターフェイスにおける新しい CreateDebugLog メソッド

オートメーション インターフェイスで CreateDebugLog メソッドを使って、**[リリース]** ビューの **[Setup.exe]** タブにある **"常にデバッグ ログを作成する"** 設定を指定することができます。デフォルト値は False です。

実行時に SSL 証明書の IIS 証明書ファイルを設定する新しい簡易ダイアログ

プロジェクト・この情報は、基本の MSI プロジェクトに適用します。

InstallShield 2018 には、エンド ユーザーが SSL 証明書に提供する IIS 証明書ファイルを参照し、インストール実行時にパスワードを入力できる、インストーラー用の新しい簡易ダイアログ (IISBrowseSSLCertificate) が含まれています。

インストーラーに **[IIS の SSL を構成する]** ダイアログ を追加するには、次の手順に従います:

タスク ***インストーラーに [IIS の SSL を構成] ダイアログ を追加するには:***

1. **[サーバー構成]** の下の **ビュー リスト** にある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトの追加]** をクリックします。InstallShield が新しい Web サイトを追加します。
3. 新しい Web サイトを選択して **[セキュリティ > セキュリティで保護された通信]** の下にある **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティに移動します。
4. **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティを次の値に設定します:

Property	Value
SSL 証明書	[IS_IIS_WEBCERTPATH]
SSL 証明書のパスワード	[IS_IIS_WEBCERTPASSWORD]

5. **[ユーザー インターフェイス > ダイアログ]** ビューを開いて、ダイアログ シーケンスに **IISBrowseSSLCertificate** ダイアログを追加します。

   [編集] ボックス (**IISWebCertPassword** と **IISWebCertPath**) およびプッシュボタン (**BrowseCertificate**) イベントの **IISBrowseSSLCertificate** ダイアログで、ユーザーが構成した SSL 証明書およびパスワードのプロパティ名がプロパティ名を更新する必要があります。

XML ファイルで新しい子要素を作成するときに絶対パスまたは相対パスを指定する

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

以前のリリースでは、**[システム構成 > XML ファイルの変更]** ビューを使って既存する親要素内の子要素と同じ名前を持つ XML ファイルに新しい子要素を追加すると、XML ファイルの変更が失敗しました。

XML ドキュメント内のノードのパスには、絶対パスまたは相対パスの両方が使用できます。絶対パスはルートから始まります。既存の親要素内にある子要素と同じ名前を持つ XML ファイルに新しい子要素を追加するときは、絶対パスを使用する必要があります。

InstallShield 2018 では、子要素を作成するときに絶対パスを使用することを指定するための "絶対 XPath を使用する" という名前の新しい設定が **[XML ファイルの変更]** ビューに追加されました。

"絶対 XPath を使用する" 設定によって異なる子要素作成時の動作:

- **選択** — このオプションが選択されている場合、子要素が追加されるときに絶対 XPath が使用されます。
- **非選択** — このオプションが選択されていない場合、子要素が追加されるときに汎用 XPath が使用されます。デフォルトでは、**"絶対 Xpath を使用する"** オプションは選択されていません。

スイート プロジェクトで、ダイアログ ボックス コントロールのデフォルト キーボード フォーカスを設定する

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では今回より、スイート プロジェクトでウィザード ページを定義するとき、ウィザード ページ上のどのコントロールにデフォルトのキーボード フォーカスを配置するのかを指定することができます。

[ウィザード インターフェイス] ビューで、**[外観]** の下に **"デフォルト フォーカス"** という名前の新しいプロパティが追加されていて、そのウィザード ページで定義されているすべてのコントロールが一覧表示されます。コントロールを選択して、デフォルト キーボード フォーカスをそのコントロールに設定します。

基本の MSI プロジェクトの PowerShell スクリプト エディター

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2018 では、基本の MSI プロジェクトで **[カスタム アクションとシーケンス > カスタム アクション]** ビューの新しい **[スクリプト]** タブで PowerShell スクリプト エディターを使用できます。以前のリリースでは、PowerShell スクリプト エディターはスイート/アドバンスト のみ使用可能でした。

InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開く新しいオプション

プロジェクト・この情報は、次のプロジェクトの種類に適用します :

- *トランスフォーム*

InstallShield 2018 では今回より、InstallShield トランスフォーム ウィザードで、既存のトランスフォーム ファイルを開くことができます。トランスフォーム ウィザードで初めて開いたときと同様に、トランスフォーム ファイルのベースとなる MSI パッケージを選択するためのプロンプトが表示されます。これによって、複数の MSI パッケージに共通の汎用トランスフォーム ファイルを使用することができます。

InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開くには、Windows エクスプローラーでトランスフォーム ファイルを右クリックしてから、コンテキスト メニューから **[InstallShield トランスフォーム ウィザードで開く]** を選択します。

追加された前提条件

InstallShield 2018 には、以下の前提条件が追加されています:

- Visual C++ 2017 x86 および x64 前提条件
- Microsoft SQL Server 2014 SP1 および SP2 前提条件
- Microsoft .NET Framework 4.7 前提条件

Visual C++ 2017 x86 および x64 前提条件

Microsoft Visual Studio 2017 がリリースされたため、InstallShield は今回より、Visual C++ 2017 x86 および x64 の前提条件を含みます。

Microsoft SQL Server 2014 SP1 および SP2 前提条件

Microsoft SQL Server 2014 には 2 つのサービス パックがリリースされているため、InstallShield は今回より、Microsoft SQL Server 2014 SP1 および SP2 両方の前提条件を含みます。

Microsoft .NET Framework 4.7 前提条件

InstallShield は今回より、Microsoft .NET Framework 4.7 の前提条件を含みます。

InstallShield 2016 SP2 の新しい機能

FlexNet Code Aware との統合

InstallShield には、今回より FlexNet Code Aware との統合が含まれています。FlexNet Code Aware は、製品のセキュリティおよび知的財産 (IP) コンプライアンス リスクを迅速にスキャンする、自動オープン ソース リスク評価およびパッケージ検出ソリューションです。

FlexNet Code Aware の現在のリリースでは、次のファイル分析がサポートされています:

- Java パッケージ
- Node パッケージ
- Nuget パッケージ
- RPM パッケージ
- Ruby パッケージ
- EXE & DLL ファイル

セキュリティ脆弱性は、National Vulnerability Database (NVD) と照合されます。

FlexNet Code Aware を実行する

FlexNet Code には InstallShield とは別のライセンスが必要です。トライアル/評価版もあります。詳細については、Revenera Web サイトの [FlexNet Code Aware 製品] ページ を参照してください。

InstallShield 内部から FlexNet Code Aware を実行するには、InstallShield の **[プロジェクト]** メニューから **[FlexNet Code Aware を使ってプロジェクトをスキャン]** をクリックします。このメニュー オプションは、InstallShield プロジェクトが現在開いていない場合は無効です。FlexNet Code Aware アイコンも、InstallShield 標準ツールバーに表示されます。

FlexNet Code Aware がプロジェクトのスキャンを完了したとき概要にはキャン済みファイルの数、オープン ソース パッケージ、ならびに検出された脆弱性の数が表示されます。FlexNet Code Aware のフル ライセンス版では、**[View report (レポートの表示)]** ボタンが使用できます。このレポートで提供される詳細については、「FlexNet Code Aware レポートの読み方」を参照してください。

FlexNet Code Aware レポートの読み方

***メモ・** FlexNet Code Aware レポートは、トライアル/評価版モードでは使用できません。FlexNet Code Aware のフル ライセンス版が必要です。*

FlexNet Code Aware レポートを参照するためには、FlexNet Code Aware がプロジェクトをスキャンした後に表示される概要ダイアログで **[View repoert (レポートの表示)]** をクリックします。

FlexNet Code Aware レポートは、いくつかのセクションから構成されています:

- 最初の [Summary (概要)] ビューには、**Scan Summary (スキャン概要)**、**Operational Risk (運用リスク)評価**、**Security Vulnerability Exposure (セキュリティ脆弱性の公開)**、および **License Exposure (ライセンスの公開)** が提示されます。
  - **[Scan Summary (スキャン概要)]** セクションは、ファイル タイプの内訳、分析済みファイルの割合、および検出数を含む、スキャン済みのコードベースについての詳細が表示されます。
  - **[Operational Risk (運用リスク)]** セクションには、パッケージと知的財産 (IP) との問題、およびパッケージとセキュリティ脆弱性の組み合わせに基づいた合成リスク評価が表示されます。
  - **[Security Vulnerability Exposure (セキュリティ脆弱性の公開)]** および **[License Exposure (ライセンスの公開)]** セクションには、識別された問題の種類とカテゴリーの内訳が表示されます。
- [Package Inventory (パッケージ インベントリ)] ビューは **[Scan Summary (スキャン概要)]** セクションで **[view full package inventory (フル パッケージ インベントリの表示)]** をクリックすると表示され、検出されたオープンソースおよびサードパーティ パッケージおよび関連ライセンス、セキュリティ脆弱性、依存関係、ならびに検出された著作権ステートメントの完全リストを表示します。

[Package Inventory (パッケージ インベントリ)] ビューには、ターゲットのクエリを実行して、一覧を様々なパッケージの種類に絞り込むためのフィルターが用意されています。

パッケージの詳細を確認する

[Package Inventory (パッケージ インベントリ)] レポート ページの [Vulnerabilities (脆弱性)] 列にある脆弱性カウントをクリックして、各パッケージの詳細を確認します: パッケージ インベントリ レポートの一部が含まれた **[Vulnerabilities detail (脆弱性の詳細)]** が表示されます:

InstallShield 2016 SP1 の新しい機能

Microsoft Visual Studio 2017 のサポート

InstallShield は、Visual Studio 2017 をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

InstallShield 2016 の新しい機能

InstallShield 2016 には、以下のよう新機能が追加されています:

- Windows オペレーティング システムの最新版リリースをサポート
- Microsoft SQL Server 2016 サポート
- Windows アプリ パッケージを作成するためのサポート
- スイートに追加された SQL サポート
- タイルの構成
- 新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件
- Adobe Reader、Microsoft Office、および .NET Framework の前提条件システム検索

Windows オペレーティング システムの最新版リリースをサポート

InstallShield 2016 は、Windows オペレーティング システムの最新版リリースをサポートします。

- Windows 10 Anniversary Update
- Windows Server 2016

InstallShield をこれらのオペレーティング システムにインストールできるだけでなく、これらのオペレーティング システムをターゲットにするインストールを作成することができます。

Microsoft SQL Server 2016 サポート

プロジェクト・*Microsoft SQL Server 2016 サポートは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*

InstallShield に、SQL Server 2016 Database 上で SQL スクリプトを実行できるサポートが追加されました。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2016 が含まれています。

インストールで SQL Server 2016 をターゲットするとき、エンドユーザーがデータベース サーバーを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、SQL Server 2016、SQL Server 2016 Express、および SQL Server 2014 Express LocalDB のインスタンスが表示できるようになりました。また、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、指定された SQL Server 2016 サーバー上のカタログが表示されるようになりました。

InstallShield に追加された、新しい InstallShield 前提条件の完全リストは「新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件」を参照してください。

メモ・*Microsoft SQL Server 2016 は 64-bit (x64) のみです。*

Windows アプリ パッケージを作成するためのサポート

プロジェクト・*Windows アプリの作成機能は、基本の MSI プロジェクトで使用できます。*

重要・*デスクトップ拡張 (デスクトップ ブリッジ) を含む Windows アプリ パッケージ (.appx) のインストールおよびテストを行うには、Windows 10 Anniversary Update が必要です。Windows アプリ パッケージにデジタル署名を行う場合、InstallShield を Windows 10 または Windows 10 SDK がインストールされているマシン上にインストールする必要があります。*

Windows 8.x および 10 上にアプリを配布およびインストールする為に使用される Windows アプリ パッケージ (.appx) は、シンプルでセキュリティ保護されたパッケージ フォーマットで、UWP (ユニバーサル Windows プラットフォーム) で使用可能な唯一のフォーマットです。Windows アプリ パッケージの利点:

- 高い可用性、信頼性、および耐久性によって、アプリケーションが長期間にわたってエラーなしで継続的に動作し続けます。
- 必要最小限の構成とカスタマイズ不要な UI によるスタティック ビルドを使ったスムーズなインストール経験
- Windows ストアを使ってアプリケーションを販売または提供できるオプション
- UWP API を使用できる機能だけでなく、ライブ タイルなどの UWP 機能を活用
- Windows Nano Server 上で サポートを持つ唯一のパッケージ フォーマット

InstallShield は今回より、代替ビルド出力を通して Windows アプリ パッケージ フォーマット (.appx) およびそのデスクトップ / サーバー拡張機能の作成をサポートし、Windows アプリ パッケージ フォーマットに適合しないアイテムを識別するための適合性テストを提供します。Windows アプリ パッケージの作成をサポートする InstallShield に新しく追加された機能の詳細については、次のサブセクションをご覧ください。

- [リリース] ビュー内の Windows アプリの設定
- [ショートカット] ビューにおける Windows アプリ ロゴのカスタマイズ
- MSIX/UWP アプリ適合性のテスト

- スイートに追加された新しい UWP 条件のチェック

[リリース] ビュー内の Windows アプリの設定

[リリース] ビューでリリースを選択するとき、**[Appx/MSIX]** という名前の新しいリリースごとに提供されるタブに、Windows アプリ パッケージを作成する設定が追加されました。ここで、Windows アプリ パッケージのビルド プロセスに影響する様々な主要な設定を指定できます。特に **[配布方法]**、**[デスクトップ拡張を含む]**、または **[サーバー拡張を含む]** オプションは、特定の種類のインストーラー プロジェクト データにどの警告またはエラーが発生するかに影響します。

これらの新しい設定についての総合情報は、「リリースの [Appx/MSIX] タブ」を参照してください。

[ショートカット] ビューにおける Windows アプリ ロゴのカスタマイズ

[リリース] ビューに追加された新しい Windows アプリ設定以外にも、Windows アプリ パッケージで作成されたタイルを構成するための新しい設定があります。これらの設定は、[ショートカット] ビューの **[Windows アプリ パッケージ タイルのオーバーライド]** 領域にあります。

これらの新しい設定についての総合情報は、「ショートカットの設定」を参照してください。

MSIX/UWP アプリ適合性のテスト

InstallShield には、.msi パッケージ内で Windows アプリ パッケージ フォーマットに適さないアイテムの存在をスキャンする、新しい **InstallShield MSIX/UWP アプリ適合性スイート**が追加されました。アクセスするには、**[ビルド]** メニューから **[検証]** をポイントしてから、**[InstallShield MSIX/UWP アプリ適合性スイート]** をクリックします。

InstallShield MSIX/UWP アプリ適合性スイートは、問題が見つかったすべてのテストを [リリース] ビューに表示し、各問題に関連付けられた列に既知の Windows アプリ形式への適用性を示します。従来型の CUB の場合、これらの列は空白のままです。このレポートは、[リリース] ビューで、リリースの下にある [検証] フォルダーを選択すると表示されます。新しい ISUWP 検証の説明その他の総合情報は、「InstallShield MSIX/UWP アプリ適合性スイート」を参照してください。

スイートに追加された新しい UWP 条件のチェック

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な異なる種類から選択できます。スイートに次の条件チェックが追加されました。

- **Windows アプリ パッケージの対象**—ターゲット システムで Windows アプリ パッケージの実行時に依存ファイルの存在を確認して、これをサポートしない Windows または Windows Server バージョンに Windows アプリ パッケージがインストールされることのないように防ぎます。

  ***メモ・**この条件は、Windows アプリ パッケージの対象条件のみで使用可能です。別のパッケージ タイプで使用した場合、正しく機能しません。*

- **UWP タイプの存在**—ターゲット システムをチェックして、UWP 機能の存在を確認します。たとえば、デスクトップ ブリッジの存在を確認する条件ステートメントを作成するには、Windows.ApplicationModel.FullTrustProcessLauncher タイプを確認します。これは、条件付でインストールをブロックする、または .msi および Windows アプリ パッケージ (.appx) のどちらをインストールするかを選択する場合に使用できます。

メモ・これは、Windows 10 以前のオペレーティング システムで常に False 評価されます。"Windows.ApplicationModel.FullTrustProcessLauncher" タイプ名サブ設定を使用するには、Windows 10 Anniversary Update 以降が必要です。

スイートに追加された SQL サポート

***プロジェクト・**この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

***エディション・**スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。*

SQL サーバーは、特に InstallShield スイート インストールで提供される複数パッケージ サポートを活用する多くのアプリケーションで不可欠です。以前、InstallShield SQL サポートが使用できるのは基本の MSI、InstallScript、および InstallScript MSI プロジェクトのみでした。今回より、SQL サポートがスイート/アドバンスト UI プロジェクトに追加されており、以下の機能を使用することができます:

- 新しい SQLLogin 定義済みウィザード ページを追加
- SQL ステートメントを直接スイートから実行

詳しくは、次のトピックを参照してください:

- スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する
- ウィザード インターフェイス内の要素のアクションを構成する
- [定義済みタスク ページ] パネル

新しい SQLLogin 定義済みウィザード ページを追加

InstallShield では、新しい**SQLLogin** 定義済みウィザード ページを使ってスイート/アドバンスト UI プロジェクトに SQL サポートを追加します。以前、スイート/アドバンスト UI プロジェクトに SQL サポートを含む .msi パッケージを追加すると、アドバンスト UI またはスイート/アドバンスト UI セットアップ起動プログラムが Windows Installer のユーザー インターフェイスを自動的に抑制しました。スイート/アドバンスト UI プロジェクト インストールにカスタム SQLLogin ウィザード ページを手動で作成する必要がありました。

新しい定義済みのページをプロジェクトに追加するとき、**[データベース サーバーのログイン情報を入力]** タスク ページを選択して、必要に応じてウィザードのパネルを完了させます。その後、SQLLogin 定義済みウィザード ページを、プロジェクトに追加します。この SQLLogin ウィザード ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージがターゲットとするデータベース サーバーへの接続を設立することができます。

SQLLogin ウィザード ページをプロジェクトに追加すると、以下が可能となります:

- スイート プロジェクト内の SQL ログインの設定を識別するプロパティを指定してから、これらのプロパティを受け取る .msi パッケージを選択する
- .msi パッケージ内で SQL ログインの設定を識別するプロパティを指定する
- データベース テクノロジ (Microsoft SQL Server、Microsoft Windows Azure、MySQL、または Oracle) を選択して、ターゲットにする ODBC ドライバーを選択する

SQL ステートメントを直接スイートから実行

スイート/アドバンスト UI プロジェクトでは今回より、ユーザー インターフェイスから SQL データベース サーバー上の SQL ステートメントを直接実行することができます。これによって、インストール続行前に SQL データベース サーバーを調査することができます。

これによって、スイート プロパティで SQL クエリの結果にアクセスが可能となります。このサポートを使用するための **[SQL 文字列を実行]** オプションが、UI イベントの [新しいアクション] メニューに追加されています。SQL ステートメントは、UI イベントの [新しいアクション] メニューで使用できる新しい追加オプション ([**データベース メタデータの構成**] および [**SQL ログイン プロパティのオーバーライド**]) を使って指定されたプロパティおよびデータベース メタデータ を使って実行されます。

タイルの構成

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- 基本の MSI
- InstallScript MSI
- InstallScript

Windows 8 からアプリケーション タイルのグリッドをスタート画面に表示できるようになりました。これは、今までのショートカットの一覧に取って代わるもので、ショートカットの代わりにタイルを配置します。
InstallShield は、スタート画面上のデスクトップ アプリのタイルの外観をカスタマイズすることができます。次のタイル構成設定が使用できます:

- アプリケーション名を中サイズ (150x150) のタイルに含めるとき、明色または暗色のテキストを切り替える
- タイル背景色を選択
- カスタム タイル イメージ (小: 70x70、中:150x150) の使用オプション
- アプリケーション名を中サイズ タイルに表示または非表示を選択

[タイルの構成] ノードは、メインの **[ショートカット]** ビューおよび各コンポーネントの **[ショートカット]** サブビューに表示されます。適用するすべてのタイル構成が一覧表示されます。

詳しくは、次のトピックを参照してください:

- スタート画面上のデスクトップ アプリのタイルの外観を構成する
- [タイル構成] の設定

新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件

***プロジェクト・**InstallShield 前提条件は、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加できます。*

InstallShield には、以下の InstallShield 前提条件が含まれています。

- Microsoft .NET Framework 4.6.2 (完全版)
- Microsoft .NET Framework 4.6.1 (完全版)

- Microsoft .NET Framework 4.6.1 (Web インストーラー)
- Microsoft ReportViewer 2015
- Microsoft SQL Server 2014 Express System CLR Types (x86)
- Microsoft SQL Server 2016 Express RTM (x64)
- Microsoft SQL Server 2016 Express RTM LocalDB (x64)
- Microsoft Visual C++ 2015 Update 3 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2015 Update 3 再配布可能パッケージ (x64)
- Windows Management Framework 4.0 for Windows 7 SP1 および Server 2008 R2 SP1 (x64)
- Windows Management Framework 4.0 for Windows Server 2012 (x64)
- Windows Management Framework 5.0 for Windows 7 SP1 (x86)
- Windows Management Framework 5.0 for Windows 7 SP1 および Server 2008 R2 SP1 (x64)
- Windows Management Framework 5.0 for Windows 8.1 (x86)
- Windows Management Framework 5.0 for Windows 8.1 および Server 2012 R2 (x64)
- Windows Management Framework 5.0 for Windows Server 2012 (x64)

これらの前提条件は、サポートされているターゲット システムに適切なテクノロジをインストールします。

***メモ・** .NET Framework の* **Web** *前提条件には、インターネット接続が必要です。この前提条件は、必要に応じて、必須の再配布可能ファイルをダウンロードします。.NET Framework の* **完全** *な前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。*

Adobe Reader、Microsoft Office、および .NET Framework の前提条件システム検索

***プロジェクト・** 前提条件システム検索は、基本の MSI および InstallScript MSI プロジェクトに適用します。*

InstallShield に新しい定義済みシステム検索が追加されました:

- Adobe Reader 11
- Adobe Reader DC
- Microsoft Office 2013
- Microsoft Office 2016
- Microsoft .NET Framework 4.5.1
- Microsoft .NET Framework 4.5.2
- Microsoft .NET Framework 4.6
- Microsoft .NET Framework 4.6.1
- Microsoft .NET Framework 4.6.2

インストールでこれらの 1 つまたは両方が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

強化機能

InstallShield 2025には、以下のような強化機能が搭載されています。

- [ダイレクト エディター] ビューの強化機能
- スイート – UI 機能強化
- [プロセスの強制終了] カスタム アクションの強化内容
- コンポーネントの属性に使用するデフォルト値を設定できる機能
- 追加のビューで、機能ごとに項目をフィルターできる機能
- デジタル署名の更新

[ダイレクト エディター] ビューの強化機能

***プロジェクト** - ダイレクト エディターは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

InstallShield には今回より、テーブル、スキーマ情報、および検証エラーの詳細を確認できるいくつかのダイレクト エディター強化内容が追加されています。これにより、ダイレクト エディターを使って高度な問題を識別および解決するためにトラブルシューティングを行うセットアップ作成者またはパッケージ作成者の生産性が飛躍的に高まります。これらの強化内容について、次のセクションに分けて説明されています:

- Directory テーブルに、解決されたターゲット ディレクトリ パスが表示される
- 列ヘッダー スキーマ情報のツールヒント
- テーブル レコード参照の追跡
- 破損した参照のインジケータ

総合的な情報は、「ダイレクト エディター」を参照してください。

Directory テーブルに、解決されたターゲット ディレクトリ パスが表示される

Directory テーブルを表示するとき、InstallShield は各行についてディレクトリの場所の解決済みパスを表示する読み取り専用の灰色列を表示します。この列は実際、プロジェクト ファイルには保存されません。表示されるテキストごとに並べ替えることができますが、その値を挿入、更新、または削除することはできません。

列ヘッダー スキーマ情報のツールヒント

InstallShield は今回より、使用可能な列データの種類を示すスキーマ情報を表示する列ヘッダーに、ツールヒントを表示します。

- **ヌル可能**—列は空白のままに残すことが可能です。
- **必須**—列には、必ず空白以外の値が必要です。
- **文字(*nn*)**—固定文字数 *nn* の文字列。
- **小さい整数**—整数値 (小数点なし)、-32767 から +32767 までの値を含む。
- **長い整数**—整数値 (小数点なし)、-2147483647 から ++2147483647 までの値を含む。
- **ローカライズ可能**—翻訳可能な文字列を含む列。このマーカーが付いていない列は、ローカライズ不可能です。
- **ストリーム**—ファイルのコンテンツなどの、バイナリ ストリーム。

***ヒント・Directory、Binary、および CustomAction** ダイレクト エディター テーブルには、これらの種類の列のいくつかが表示されます。*

テーブル レコード参照の追跡

ダイレクト エディターには今回より、テーブル レコードの関係性を簡単に確認することができる [参照の追跡] ペインが追加されています。ダイレクト エディター上部に追加された **[参照の追跡を表示]** ボタンを使って、ペインの表示/非表示を切り替えることができます。

各レコードは、1 つ以上のレコードを参照、または 1 つ以上のレコードによって参照されている可能性があります。レコードが強調表示されている場合、別のレコードを参照しているか、別のレコードから参照されていることを示し、[参照の追跡] ペインには、参照が存在するテーブルを表示する **[参照テーブル]** セクション、および実際のレコードの参照を表示する追加セクションが含まれます。レコードの参照セクションには、参照の方向を示す矢印アイコンが表示されます。

- 右向きの緑色矢印は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードを**参照する**ことを示します。
- 左向きの青色矢印は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードによって**参照されている**ことを示します。
- 両方向を指す 2 つの矢印 (右向きの緑色矢印と左向きの青色矢印) は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードを**参照する**、*および***参照されている**ことを示します。

***メモ・複数のダイレクト エディター レコードが選択されている場合、フォーカスされているレコードのみの参照が表示されます。さらに、複数のテーブルが [参照テーブル] セクションに表示された場合、ダイレクト エディター テーブルで選択されたレコードが参照する、または複数のテーブルのレコードによって参照されることを示**

*します。関連するレコードの参照を表示するには、**[参照テーブル]** セクションにある任意のテーブルをクリックします。*

***ヒント・**[参照の追跡] ペインでは、セル内をダブルクリックしてレコードの参照間を簡単に移動することができます。*

破損した参照のインジケータ

ダイレクト エディターのテーブル レコードが、もう存在しない外部キー レコードを参照している場合があります。InstallShield は今回より、このような破損した参照を持つセルに注意が向くように、赤い背景色を使用します。

たとえば、Component テーブルの **Directory_** 列が、Directory テーブルで見つからないディレクトリ名を参照する場合、**Directory_** 列が赤い背景色で表示されます。

***メモ・**ダイレクト エディターの破損した参照インジケータは、[オプション] ダイアログ ボックスの [プリファレンス] タブにある **[参照の整合性を維持]** チェックボックスには関連していません。**"参照の整合性を維持"** 設定は、プライマリ キーを変更したときに、外部キーを更新することを目的とし、破損した参照インジケータは、親の無いレコードを簡単に識別できるよう、破損した参照を表示します。このため、破損した参照は **"参照の整合性を維持"** 設定で行った選択に関係なく表示されます。*

スイート – UI 機能強化

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・**アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

様々な用途を広くサポートするために、InstallShield ではアドバンスト UI およびスイート/アドバンスト UI プロジェクトに次の機能が追加されています:

- 新しい [Windows を閉じる] および [イベントの停止] UI アクション
- スイートでスプラッシュ スクリーンのロード追加

新しい [Windows を閉じる] および [イベントの停止] UI アクション

InstallShield には、以下のような新しい UI アクションが搭載されています。

テーブル 1-2 ・ 新しいコントロール アクション

アクションの種類	説明
ウィンドウを閉じる	この種類のアクションは、メインのウィザード ページまたは 2 番目のウィンドウを閉じるか、場合によっては 2 番目のウィンドウを条件付きで閉じます。 [ウィンドウを閉じる] アクションには、次の定義済みリターン コード ID に対応するパラメータを使用できます: IDOK、IDCANCEL、IDABORT、IDRETRY、IDIGNORE、IDYES、IDNO、および IDCLOSE [ウィンドウを閉じる] アクションの動作は、ウィザード ページおよび 2 番目のウィンドウで次のように多少異なります: ・ ウィザード ページの場合、[ウィンドウを閉じる] アクションはそのリターン コード パラメータが IDCANCEL に設定されている場合、エンド ユーザーがウィザードをキャンセルできるプロンプトを表示 (また、エンド ユーザーが [はい] を指定した場合にウィザードを中断します)。その他のリターン コード ID の場合、ウィザードがすぐに閉じます。 ・ 2 番目のウィンドウの場合、[ウィンドウを閉じる] アクションは 2 番目のウィンドウを閉じます。また特殊な場合、たとえば 2 番目のウィンドウが ISRMFilesInUse および ISRMFileInUse の場合は指定されたリターン コード値が戻されます。 InstallShield では現在、指定されたリターン コード ID によって異なるカスタム動作を含む、次の 2 番目のウィンドウが提供されています: ・ ISDownloadProgress ・ ISPromptForSourceMedia ・ ISFilesInUse ・ ISRMFilesInUse ・ ISUpgradeParcel ・ ISSQLBrowse
イベントの停止	この種類のアクションは、後に続くアクションの処理を条件付きで停止します。たとえば、このアクションを使ってボタンのデフォルト動作を抑制することができます。

詳細については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

スイートでスプラッシュ スクリーンのロード追加

ロード処理に 0.5 秒以上を要するスイート インストール中、InstallShield は今回より、[インストールようこそ] ダイアログが表示される前に、プログラムが起動済みで、ロード処理を完了する必要があることを示す、スプラッシュ スクリーンを表示します。スプラッシュ スクリーンに、InstallShield は提供されている中から一番大きい setup.exe アイコンを使用し、進行状況バーも含みます。

[プロセスの強制終了] カスタム アクションの強化内容

__プロジェクト__・この情報は、次のプロジェクトの種類に適用します :

- *基本の MSI*
- *InstallScript MSI*

[プロセスの強制終了] カスタム アクションの設定に新しい *__"プロセス"__* 設定が追加されました。この設定を使って、[プロパティ マネージャー] を使用してプロパティを作成、およびアクションが正しく動作するようその値を正しくフォーマットする必要なしに、強制終了する実行可能ファイルの名前またはプロセスの PID を直接入力することができます。

詳細については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。

__ヒント__・__"プロセス"__ 設定の値は、ISTerminateProcesses プロパティに書き込まれる場合があります。InstallShield 2015 以前から移行したプロジェクトなど、__"プロセス"__ 設定に値が指定されていない追加の kill-process カスタム アクションがある場合、ISTerminateProcesses プロパティを共有して使用すると予期しない動作が発生する可能性があります。

コンポーネントの属性に使用するデフォルト値を設定できる機能

InstallShield テーブルで、コンポーネント属性に使用されるデフォルト値を設定するための新しいプロパティのサポートが追加されました。InstallShield テーブルに MsiComponentAttributes のプロパティが存在する場合、その値は Component テーブル内のデフォルト Attributes 列値を 8 から指定の値にオーバーライドします。

たとえば、新しいコンポーネントを 64 ビットとする場合、MsiComponentAttributes 値に 256 を追加します。264 (64 ビット共有の場合)、または256 (64 ビット非共有の場合) を指定できます。これによって、"64 ビット コンポーネント" および "共有" 設定 ([コンポーネント] ビューの [全般] 領域) がそれぞれ [はい] または [いいえ] に更新されます。

Component テーブルの Attributes 列を計算する時に使用されるビット値についての詳細は、MSDN ライブラリの「Component Table」ページを参照してください。

__メモ__・コンポーネント属性に使用されるデフォルト値を設定するには、各プロジェクトの __ダイレクト エディタ__ にある InstallShield テーブルに、手動で MsiComponentAttributesプロパティを更新する必要があります。この場合、製品構成の __"テンプレートの概要"__ 設定は無視されます。

追加のビューで、機能ごとに項目をフィルターできる機能

次のビューには今回より、プロジェクトに含まれる任意の機能ごとにビュー リストをフィルターすることができる [ビュー フィルター] が用意されています。

- **[環境変数] ビュー**—このビューの上部にある [ビュー フィルター] リストを使って、プロジェクトに含まれる特定機能に関連付けられた環境変数を表示/非表示にすることができます。[ビューリスト] から機能を選択して、その機能のみを後に続くイベント (たとえば、環境変数の作成、変更、または削除) に関連付けることができます。最後に、プロジェクトに含まれるすべての環境変数を表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。詳細については、「[環境変数] ビュー」を参照してください。
- **[テキスト ファイルの変更] ビュー**—このビューの上部にある [ビュー フィルター] リストを使って、プロジェクトに含まれる特定機能に関連付けられたテキスト ファイルの変更セットを表示/非表示にすることができます。[ビューリスト] から機能を選択して、その機能のみを後に続くイベント (たとえば、変更セットの作成、変更、並べ替え、または削除」) に関連付けることができます。結果となる変更は、機能がインストールされるときにターゲット システム上で実行時に行われます。最後に、プロジェクトに含まれるすべてのテキスト ファイルの変更セットを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。詳細については、「[テキスト ファイルの変更] ビュー」を参照してください。
- **[INI ファイルの変更] ビュー**—このビューの上部にある [ビュー フィルター] リストを使って、プロジェクトに含まれる特定機能に関連付けられた初期化 (.ini) ファイルを表示/非表示にすることができます。[ビューリスト] から機能を選択して、その機能のみを後に続くイベント (たとえば、.ini ファイルの作成、インポート、変更、または削除」) に関連付けることができます。結果となる変更は、機能がインストールされるときにターゲット システム上で実行時に行われます。最後に、プロジェクトに含まれるすべての .ini ファイルを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。詳細については、「[INI ファイルの変更] ビュー」を参照してください。

デジタル署名の更新

InstallShield 2015 より、インストールおよびファイルをビルド時に署名する際、SHA-256 ハッシュ アルゴリズムを使ったデジタル証明書を使用できるサポートが追加されました。

InstallShield 2016 では、Windows Installer および InstallScript プロジェクトの SHA-256 デジタル証明書サポートが次のように強化されています:

- **[証明書の選択]** ダイアログ ボックスの新しい [署名ダイジェスト] ドロップダウンを使って、ダイジェストの種類を指定できる機能
- 今回より、RFC3161 タイムスタンプがサポートされていて、settings.xml で次の記述を含んで指定できます:
  - .msi、.exe、および .dll ファイルでは、**DigitalSignature/@Timestamp** は **Authenticode** または **RFC3161 サーバー**
  - Windows アプリ パッケージファイルに使用される **DigitalSignature/@TimestampRFC3161** ファイルは **RFC3161 サーバー** でなくてはなりません
- 証明書ストアにある類似した名前の証明書も処理します

***重要・**2016 年 1 月以降に作成またはタイムスタンプが付けられたすべての新しい署名は、SHA-256 に基づく必要があります。SHA-1 証明書を使って署名されているすべてのファイルを継続してサポートするためには、2016 年 1 月以前の日時を使ったタイムスタンプを含める必要があります。これらのファイルは、すべての現在のバージョンの Windowsですべての SHA-1 サポートが停止される 2020 年 1 月 14 日まで、MOTW (Mark of the web) システムを使って引き続き使用することができます。*

InstallShield 2015 SP1 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows 10 のサポート

InstallShield は、Windows 10 をサポートします。

Microsoft Visual Studio 2015 のサポート

InstallShield は、Visual Studio 2015 をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

Microsoft App-V 5.1 のサポート

InstallShield および InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5.1 クライアント上で実行できる仮想アプリケーションを作成するためのサポートが含まれています。

新しい App-V 5.1 用 InstallShield 前提条件

InstallShield には、生成された App-V 5.x パッケージに setup.exe を含めるときに使用できる、新しい InstallShield 前提条件が含まれています。InstallShield 前提条件をリリースに含める必要がある場合、Setup.exe セットアップランチャーが必要になることに注意してください。

- Microsoft App-V 5.1 Desktop Client

この InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft から再配布可能ファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。必要な前提条件についての詳細は、https://technet.microsoft.com/ja-jp/library/mt346482.aspxを参照してください。

App-V アシスタントの [オペレーティング システム] オプションについて、App-V 5.x バージョン用に Windows 10 を追加

App-V 5.1 より、Windows 10 32 ビット版および 64 ビット版のオペレーティング システムがサポートされています。これに伴い、Microsoft App-V アシスタントの [パッケージ情報] ページで、App-V 5.x が選択されている際に選択可能なオペレーティングシステムとして次が追加されました:

- Windows 10 (32 ビット版)
- Windows 10 (64 ビット版)

Microsoft Visual C++ 2015 および .NET Framework 4.6 用の新しい InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい前提条件が含まれています:

- Microsoft Visual C++ 2015 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2015 再配布可能パッケージ (x64)
- Microsoft .NET Framework 4.6 Full
- Microsoft .NET Framework 4.6 Web

これらの前提条件は、サポートされているターゲット システムに適切なテクノロジをインストールします。

DialogSetInfo 関数の新しい定数

InstallScript 関数 DialogSetInfo の nInfoType パラメーターに新しい定数が使用できます:

DLG_INFO_ALTIMAGE_HIDPI—この定数は、ダイアログで高 DPI イメージを指定します。高 DPI イメージは BMP、GIF、JPEG、PNG、および TIFF などをサポートします。透明化が必要な場合、それをサポートする PNG などのイメージ タイプを使い、ダイアログ内で szInfoString が表示するイメージの名前 (オプションでパスを含むことが可能) を指定します。このパラメーターは、ダイアログの左側にある標準インストール イメージを表示する ダイアログすべてに適用されます。

DLG_INFO_ALTIMAGE_HIDPI が nInfoType で渡される場合、次のパラメーター値が必要です:

- szInfoString—表示するイメージ ファイルの名前。オプションでパスを含めることもできます。ファイルが指定されなかった場合、SUPPORTDIR に存在するものと見なされます。ファイルが存在しない場合は、DialogSetInfo が ISERR_FILE_NOT_FOUND を返します。
- nParameter—DPI 拡大/縮小率。たとえば、200% 拡大する場合 200、150% の場合 150 など。サポートされている最小縮小値は 25 です。この値に 0 が渡された場合、イメージは何も表示されません。If DLG_INFO_ALTIMAGE_REVERT_IMAGE が渡されると、以前に使用されたイメージが表示されます。

この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。

詳細については、「DialogSetInfo」を参照してください。

アーキテクチャの検証を無効化するオプションの追加

[リリース] エクスプローラーの [アーキテクチャの検証] オプションに **[なし]** オプションが追加され、ビルド時にアーキテクチャの検証をバイパスすることを指定できるようになりました。

詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

InstallShield 2015 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows 10 ベース システムのサポート

InstallShield は、Windows 10 をサポートします。

Windows 10 をターゲットにする

Windows 10 が搭載されているシステム上では、Windows Installer プロパティ VersionNT および VersionNT64 が 603 を示します。これは、もともと Windows 8.1 のバージョン番号として使用されたものです。従って、Windows 10 をターゲットに特定して .msi パッケージの条件を作成することは不可能です。

Windows Installer 5.0 および Windows 7 より、.msi パッケージの DLL アクションには shim が適用されるため、オペレーティング システム バージョンの取得が阻止されます。API の **GetVersion**、**GetVersionEx**、および **RtlGetVersion** は、もともと Windows Vista のバージョン番号である 6.0.6000 を返します。従って、DLL カスタム アクション、または InstallScript カスタム アクション (DLL として実装される) から実際の Windows のバージョン番号を取得することはできません。

前述の Windows Installer 動作のため、.msi パッケージが実行中のバージョンを検出することは容易ではありません。ターゲット システムの OS 要件を指定できる領域、たとえば基本の MSI および InstallScript MSI プロジェクトでは、プロジェクト アシスタントの [インストール要件] ページで、新しい実行時の動作を反映するように、[Windows 8.1] オプションの名前が **「Windows 8.1 または Windows 10」** に変更されました。

InstallScript、アドバンスト UI、およびスイート/アドバンスト UI プロジェクトでは、ターゲット システム上に存在する Windows のバージョン (Windows 10 を含む) を正しく検出できる機能が提供されています。したがって、インストールが Windows 10 をターゲットとする、または除外する必要がある場合、これらのプロジェクトの種類を使って、実際のターゲット システム プラットフォームに基づいて .msi パッケージを実行する条件を作成できます。

Windows 10 にインストール可能な InstallShield 前提条件は、必要に応じて、これらのシステムにインストールされるように更新されています。以前これらのシステムでは、前提条件がデフォルトで実行されない場合がありました。

Windows 10 の InstallScript 言語サポート

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- SYSINFO.WINNT.bWin10 – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 10 の場合、この値は TRUE です。(これは、InstallScript イベント ドリブン型コードに適用し、カスタム アクションには適用しません。)
- ISOSL_WIN10 – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 10 を実行中であることを示します。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

Microsoft Visual Studio 2015 のサポート

InstallShield は、Visual Studio 2015 プレビュー版をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

デジタル署名の強化機能

InstallShield には、ビルド時にインストールおよびファイルにデジタル署名を行うための、いくつかの強化機能が含まれています。

SHA-256 デジタル証明書のサポート

InstallShield では、インストールおよびファイルをビルド時に署名する際、HA-256 ハッシュ アルゴリズムを使ったデジタル証明書を使用できます。

SHA-1 はセキュリティの脆弱性があるため、SHA-256 の使用が推奨されます。Microsoft は、Windows では 2016 年 1 月以降に SHA-1 証明書を使って署名およびタイムスタンプが追加されているアイテムを信頼しないことを発表しました。さらに、証明書を発行する組織である証明機関では、SHA-1 証明書が段階的に廃止されます。したがって、InstallShield プロジェクトに含まれる任意の SHA-1 証明書は、SHA-256 証明書と差し替えることが推奨されます。最新情報および特定の詳細については、証明機関にお問い合わせください。

InstallShield でリリースに署名するための SHA-1 証明書を SHA-256 証明書に置き換えるには、[リリース] ビューの [署名] タブを使って、現在の証明書への参照を SHA-256 証明書と置き換えます。

プロジェクトで、SHA-256 証明書を使った署名が構成されている場合、InstallShield はビルド時に署名を行うファイルの署名に SHA-256 ハッシュを使用します。プロジェクトで SHA-1 証明書を使った署名が構成されたままになっている場合、InstallShield は SHA-1 ハッシュを使用します。また、SHA-1 証明書を使用すると、今回より SHA-1 の使用についてアラートするビルド警告 -7346 が発生します。

InstallShield の以前のバージョンでは、SHA-1 または SHA-256 証明書のいずれかを使って署名が行なわれる際、ファイルの署名に SHA-1 ハッシュが使用されました。

詳細については、「デジタル署名とセキュリティ」を参照してください。

証明書を参照するための証明書ストアを使用できる機能

ファイルおよびインストールの署名に使用するデジタル署名情報を指定するとき、InstallShield では今回より、使用する証明書を含む証明書ストアを参照することができます。このサポートは、マシン上の .pfx 証明書ファイルを指定する方法の代替として提供されています。

証明書ストアまたは .pfx 証明書を使うかどうかを指定するには、[リリース] ビューの [署名] タブにある "デジタル証明書ファイル" 設定を使用します。この設定で省略記号ボタン (...) をクリックすると、新しい [証明書の選択] ダイアログ ボックスが開きます。このダイアログ ボックスを使って、ストア名 (Personal、Trusted Root Certification Authorities、Enterprise Trust、Intermediate Certification Authorities)、ストアの場所 (ユーザー、マシン)、および特定の証明書を識別するためのサブジェクトといった情報を指定することができます。別の方法として、このダイアログ ボックスで使用する .pfx ファイルの名前とパスを指定することもできます。

ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。

証明書ストアを、パッチおよび QuickPatch パッケージの署名に使用することもできます。

- パッチの証明書ストアまたは .pfx 証明書情報を指定するには、[パッチのデザイン] ビューでパッチの構成の [デジタル署名] タブを使用します。
- QuickPatch パッケージで証明書ストアまたは .pfx 証明書情報を指定するには、QuickPatch プロジェクトで [一般情報] ビューの [ビルドの設定] 領域を使用します。この領域には、新しいサポートを含む [デジタル署名] タブがあります。

さらに、ビルド済みの InstallScript リリースの署名に使用できるコマンドライン ツール iSign.exe が更新され、証明書ストアにある証明書を使用できるようになりました。

詳しくは、次を参照してください:

- デジタル署名とセキュリティ
- [証明書の選択] ダイアログ ボックス
- リリースの [署名] タブ ([リリース] ビュー内のリリース)
- [デジタル署名] タブ ([パッチのデザイン] ビューのパッチ構成用)
- [デジタル署名] タブ (QuickPatch プロジェクト)
- iSign.exe

InstallShield では、今回より .spc および .pvk ファイルを使った署名がサポートされていません。これらのファイルを .pfx ファイルに変換する方法については、「デジタル署名とセキュリティ」を参照してください。

UAC ダイアログ ボックスにプログラム名を指定できる機能

[リリース] ビューの [署名] タブには、"署名の説明" 設定がります。この設定を使って、ビルド時に InstallShield が署名を行なう Setup.exe ファイル、.msi ファイル、その他のインストール ファイルの UAC ダイアログ ボックスで "プログラム名:" ラベルの右側に表示するテキストを指定します。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。

"署名の説明" 設定を空白のままに残すと、InstallShield は UAC ダイアログ ボックスのテキストとして、ファイル名を拡張子なしで使用します。

詳細については、「リリースの [署名] タブ」を参照してください。

.pfx ファイルまたは証明書ストアにある証明書を使って、メディア ヘッダー ファイルにデジタル署名できる機能

InstallShield では今回より、.pfx ファイルを使って、メディア ヘッダー ファイル (.hdr ファイル) に署名することができます。.hdr ファイルは、InstallScript プロジェクトの One-Click Install インストールに使用されます。別の方法として、証明書ストア内の証明書を使って、メディア ヘッダー ファイルに署名することもできます。

以前は、.pfx ファイルの代わりに .spc および .pvk ファイルを使って署名しなくてはなりませんでした。

デジタル署名情報を指定できるオートメーション インターフェイス サポート

オートメーション インターフェイスには、.pfx ファイルおよび署名ストア情報の指定がサポートが含まれています。また、署名の説明を指定することもサポートされています。

基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクトには、ISWiReleases オブジェクトに 2 つの読み書き文字列プロパティが含まれています。

- **DigitalCertificateInfo**—このプロパティは、マシン上の .pfx ファイルへのパスまたは証明書ストアを示す文字列を取得または設定します。
- **SignatureDescription**—このプロパティは、署名の説明を取得または設定します。

詳細については、「ISWiRelease オブジェクト」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの ISWiSuiteReleases オブジェクトは、これらの同じプロパティを含みます。詳細については、「ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)」を参照してください。

64 ビット開発システム上で、ソースマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示できる機能

InstallShield を 64 ビット開発システム上で使用する場合、InstallShield が表示する [レジストリ] ビューは、使用中のマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示します:

- HKEY_LOCAL_MACHINE¥Software
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node

このサポートによって、このビューのインストール先ペインの適切な領域ににソース領域からのエントリをドラッグ アンド ドロップできるようになるため、64 ビット マシン上でのインストールの開発が容易になります。

以前、64 ビット開発システム上で InstallShield を使用した場合、InstallShield の [レジストリ] ビューのソース ペインにはレジストリの HKLM¥Software 部分にある 64 ビットデータが表示されませんでした。さらに、ソース ペインには、マシンの HKLM¥Software¥Wow6432Node 領域の 32 ビットデータが HKLM¥Software 領域に表示されました。

インストールがレジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット ターゲット システム上のレジストリの 64 ビット領域にインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。[レジストリ] ビューのソースペインから 64 ビット データを、ビュー内のインストール先ペインにドラッグするだけでは、そのコンポーネントが 64 ビットであるとマークされません。

この機能は、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- 32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い
- レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する
- [レジストリ] ビュー

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで形式化された式を埋め込んで実行時に解決できる機能

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域では、ファイル、レジストリ エントリ、オペレーティング システム、その他の詳細情報をターゲット システムで照会するオブジェクト式を埋め込むことができます。これによって、ターゲット システム固有の条件に基づいて、実行時にアドバンスト UI またはスイート/アドバンスト UI の多くの設定をダイナミックに構成することができます。オブジェクト式には、この規則を使用します:

[@Object(Parameters, ...).Property(Parameters, ...)]

各オブジェクト式には、オブジェクト固有プロパティの集まりである、オブジェクトへの参照が含まれています。オブジェクトとプロパティにはパラメーターを含めることができます。

たとえば、次の Platform オブジェクト式は、アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンのアーキテクチャ (x86、x64、IA64、ARM、または不明) を取得します:

[@Platform.Architecture]

次の Registry オブジェクト式は、HKLM¥Software¥My Company Name¥My Product Name レジストリ キーの RegisteredOwner 値のデータを取得します:

[@Registry(HKLM¥Software¥[COMPANY]¥[PRODUCT], true).KeyValue(RegisteredOwner)]

プロパティ式その他のオブジェクト式のような、別の形式化された式内にオブジェクト式を埋め込むことができます。次の式では、Registry オブジェクト式が File オブジェクト式にパラメーターの一部として埋め込まれています。

[@File([@Registry(HKLM¥Software¥MyProduct).KeyValue(MyProductPath)]¥MyProduct.exe).Version]

MyProduct.exe ファイルが MyProductPath 値データに指定された場所にあるとき、File オブジェクト式はファイルのバージョンを返します。ファイルがその場所に見つからなかった場合、またはレジストリ値が存在しなかった場合、File オブジェクト式は空白文字列を返します。

InstallShield ではまた、アドバンスト UI またはスイート/アドバンスト UI インストール内のパッケージにコマンドラインを使って、文字列の一部としてリテラル角括弧を渡すこともできます。たとえば、[¥[]Text[¥]] のようなコマンドラインは実行時に [Text] として解決され、角括弧と共にパッケージに渡されます。以前、角括弧で囲まれた文字列はプロパティとして扱われたため、インストールが実行時に解決しようとしました。唯一のワークアラウンドは、実行時に角括弧を含むプロパティ値に解決する形式化されたプロパティ式 (たとえば、[PropertyForSquareBracketString]) を使う方法でした。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス

- Feature オブジェクト
- File オブジェクト
- Parcel オブジェクト
- Platform オブジェクト
- Registry オブジェクト

共通パッケージを異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有できる機能

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージを構成するとき、今回より、これを共有とマークすることができます。共有パッケージ機能を使って、2 以上のアドバンスト UI またはスイート/アドバンスト UI インストールが 1 つのパッケージを共有している場合に、アドバンスト UI およびスイート/アドバンスト UI 製品のすべてが削除されるまで、ターゲット システムにそのパッケージが保持されるようにします。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビュで選択されているパッケージをマークするには、新しい ″共有″ 設定に [はい] を選択します。さらに、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで必ず同じ パッケージ GUID を使用してください。[パッケージ] ビューの ″パッケージ GUID″ 設定を使って、パッケージの GUID を確認および変更することができます。

このビルトイン共有サポートが有効になる前には、様々な状況下で予期しない結果が発生する可能性がありました。たとえば、一部の状況下で、ターゲットシステム上に 2 つの異なるアドバンスト UI またはスイート/アドバンスト UI インストールが共有パッケージをインストールした場合、1 つだけがアンインストールされたとき、共有パッケージも削除されました。共有パッケージが不足すると、残りのアドバンスト UI またはスイート/アドバンスト UI 製品が使用不可能となる可能性があります。別の状況下では、アドバンスト UI またはスイート/アドバンスト UI 製品が無いのにもかかわらず、共有パッケージが誤ってターゲット システムに残りました。

この機能は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに含まれるパッケージの種類 (.msi、.msp、.exe、.appx、InstallScript、基本の MSI および InstallScript MSI プロジェクト) で使用できます。

詳細については、次を参照してください。

- 異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する
- [パッケージ] ビュー

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのリリースで、製品名、製品バージョン、およびスイート GUID をオーバーライドできる機能

InstallShield では今回より、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [一般情報] ビューで指定されたスイート GUID 値を、選択されたリリースの新しい値でオーバーライドすることができます。[一般情報] ビューの値をオーバーライドするには、[リリース] ビューでリリースの [ビルド] タブで使用できる新しい設定 (製品名、製品バージョン、およびスイート GUID) を使用します。

詳細については、「リリースの [ビルド] タブ」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのリリースで、パス変数の値をオーバーライドできる機能

InstallShield では今回より、プロジェクト内の各リリースでプロジェクトのカスタムパス変数 (標準パス変数、環境パス変数、およびレジストリ パス変数) の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、この機能を使って、プロジェクト内の異なるリリースでイメージや EULA などの UI 要素を置換することができます。これによって、プロジェクトの異なるエディションまたは異なるブランド バージョンのインストールを簡単に生成することができます。

プロジェクトで 1 つ以上のパス変数をオーバーライドする場合、[リリース] ビュー内のリリースの [ビルド] タブに追加された "パス変数のオーバーライド" 設定を使います。

詳しくは、次を参照してください:

- リリースのカスタム パス変数の値をオーバーライドする
- "パス変数のオーバーライド" 設定

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション インターフェイス サポート

InstallShield では、InstallShield インターフェイスを直接開いて異なるビューで変更を加えることなく、オートメーション インターフェイスを利用してアドバンスト UI およびスイート/アドバンスト UI プロジェクト (.issuite ファイル) のほとんどの開発およびビルド プロセスを自動化することができます。オートメーション インターフェイスを通して、アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域に、プログラムによるアクセスが可能です。オートメーション インターフェイスでは、多くの言語から呼び出してアドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成、編集およびビルドできる COM インターフェイスが公開されます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのトップ階層のオートメーション オブジェクトは、ISWiProject オブジェクトです。プロジェクトを開いて、変更、保存およびこれをと閉じるサンプル VBScript コードの始まりと終りは次の通りです:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("ISWiAutoSuite22.ISWiProject")
pProj.OpenProject "C:¥InstallShield 2015 Projects¥Project1.issuite"
' 変更をここで行う
pProj.SaveProject
pProj.CloseProject
```

メソッドを呼び出し、プロパティを取得および設定、およびコレクションにアクセス、またプロジェクトに機能やパッケージを追加、条件の構成、その他を行うことができます。詳細については、ドキュメントの次のセクションを参照してください:

- » アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション オブジェクト
- » アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション コレクション

アドバンスト UI およびスイート/アドバンスト UI オートメーション インターフェイスの ProgID は ISWiAutoSuiteAutomation Interface Version.ISWiProject です。その他のプロジェクト タイプの ProgID は ISWiAutoAutomation Interface Version.ISWiProject です。

新しい Microsoft Visual C++ 2015、.NET Framework 4.6 その他用の InstallShield 前提条件

InstallShield には、プロジェクトに追加することができる新しい InstallShield 前提条件が含まれています:

- Microsoft Visual C++ 2015 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2015 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2013 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2013 再配布可能パッケージ (x64)
- Microsoft .NET Framework 4.6 Full
- Microsoft .NET Framework 4.6 Web
- Microsoft .NET Framework 4.5.2 (完全版)
- Microsoft .NET Framework 4.5.2 (Web インストーラー)
- Microsoft SQL Server 2012 Express SP2 (x86)
- Microsoft SQL Server 2012 Express SP2 (x86 & x64Wow)
- Microsoft SQL Server 2012 Express SP2 (x64)
- Microsoft SQL Server 2012 Express SP2 LocalDB (x86)
- Microsoft SQL Server 2012 Express SP2 LocalDB (x64)
- Microsoft SQL Server 2012 Express SP2 Management Objects (x86)
- Microsoft SQL Server 2012 Express SP2 Management Objects (x64)
- Microsoft SQL Server 2012 Express SP2 System CLR Types (x86)
- Microsoft SQL Server 2012 Express SP2 System CLR Types (x64)
- Windows 7 (x86) 用 Internet Explorer 11.0
- Windows 7 と Windows Server 2008 R2 (x64) 用の Internet Explorer 11.0
- Microsoft ReportViewer 2012

これらの前提条件は、サポートされているターゲット システムに適切なテクノロジをインストールします。

Microsoft SQL Server 2012 Express SP2 前提条件は、Microsoft SQL Server 2012 Express SP1 前提条件にとって代わります。

Internet Explorer 10 および 11 用の新しい定義済みシステム検索

InstallShield には、ターゲット システム上で Internet Explorer 10 または Internet Explorer 11を確認する新しい定義済みのシステム検索が含まれています。インストールまたは製品でこれらのバージョンのどちらかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索の 1 つをプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この変更は、基本の MSI および InstallScript MSI プロジェクトに含まれています。

Microsoft App-V 5.0 SP3 の仮想アプリケーションを作成するためのサポート、追加の App-V パッケージの強化機能

InstallShield および InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5.0 SP3クライアント上で実行できる仮想アプリケーションを作成するためのサポートが含まれています。また、Microsoft App-V アシスタントには、App-V 5.0 SP3 または App-V の以前のバージョンに適用する、新しい設定と機能が含まれています。

新しい pp-V 5.0 SP3 用 InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft App-V 5.0 SP3 Desktop Client (x86)
- Microsoft App-V 5.0 SP3 Desktop Client (x64)

これらの InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft から再配布可能ファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。

プライマリ アプリケーション ディレクトリの代わりに、仮想ファイル システムにファイルをマップできる機能

Microsoft App-V アシスタントを使って、今回より、仮想ファイル システム (VFS) にファイルをマップするように App-V パッケージを構成することができます。このサポートは、App-V 4.x および 5.xパッケージで使用できます。

VFS にファイルをマップするか、プライマリ アプリケーション ディレクトリを使用するかを指定するには、Microsoft App-V アシスタントの [ファイル] ページを使用します。このページのその他の [オプション] 領域には、新しいファイルマッピング リンクが含まれています。この新しいリンクをクリックすると、新しい [ファイル マッピング] ダイアログ ボックスが開き、適切なオプションを選択することができます。

[ファイル マッピング] リンクとダイアログ ボックスは、[プライマリ アプリケーション ディレクトリ] リンクとダイアログ ボックスを置換します。ここでは、プライマリ アプリケーション ディレクトリを指定することができます。

詳細については、[ファイル マッピング] ダイアログ ボックス」を参照してください。

仮想ファイル システムに完全な書き込みアクセス権を持つ App-V 5.x パッケージを作成できる機能

Microsoft App-V アシスタントを使って、今回より、作成中の App-V 5.x パッケージに仮想ファイル システム (VFS) の完全な書き込みアクセス権を持たせるかどうかを指定することができます。このサポートを使用するかどうかを指定するには、新しい [VFS の完全な書き込みアクセス権を許可する] チェックボックスを使います。このチェックボックスは、[詳細オプション] 領域にある [ファイル マッピング] リンクをクリックすると表示される Microsoft App-V アシスタントの [ファイル] ページから使用できる [ファイル マッピング] ダイアログ ボックスにあります。

詳細については、[ファイル マッピング] ダイアログ ボックス」を参照してください。

App-V5.x の詳細 COM 分離の設定を構成できる機能

Microsoft App-V アシスタントを使って、今回より、COM 分離の詳細設定を構成できます。このサポートは、App-V 5.x パッケージで使用できます。

新しい設定を構成するには、Microsoft App-V アシスタントの [パッケージ情報] ページを使用します。このページの [詳細オプション] 領域には、新しい [分離の設定] リンクが含まれています。この新しいリンクをクリックすると、新しい [分離の設定] ダイアログ ボックスが開きます。このダイアログ ボックスを使って、ローカル シ

ステムから COM オブジェクトを分離するかどうか、またはCOM オブジェクトとローカル システムの対話を許可するかどうかを指定できます。このダイアログ ボックスを使って、ローカル システムから名前付きオブジェクトを分離するかどうか、または名前付オブジェクトとローカル システムの対話を許可するかどうかを指定できます。

詳細については、パッケージの [分離オプション] ダイアログ ボックスを参照してください。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

強化機能

[ファイルとフォルダー] ビューのパフォーマンス強化

サイズが大きいプロジェクトの [ファイルとフォルダー] ビューををより短時間でロードできるように、InstallShield が強化されています。

機能ごとに [ファイルとフォルダー] ビューのアイテムをフィルターできる機能

[ファイルとフィルター] ビューには今回より、[すべてのアプリケーション データ] オプションの他に、プロジェクト内の各機能を一覧表示するビュー フィルターが追加されました。このフィルターを使って、このビューのインストール先ペインのファイルとフォルダーを表示または非表示にします。

- このビューで特定の機能に属するファイルとフォルダーのみを表示するには、[ビュー フィルター] リストで機能を選択します。
- 特定の機能にファイルまたはフォルダーを追加するには、[ビュー フィルター] リストで機能を選択します。次に、**[インストール先コンピューターのフォルダー]** ペインの適切な場所にファイルまたはフォルダーを追加します。
- プロジェクトに含まれるすべてのファイルとフォルダーを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。

新しいビュー フィルターは、以前の **[機能に新しいコンポーネントを追加する]** フィルターの代わりとなります。これまでは、選択された機能によってビュー内のファイルとフォルダーを表示または非表示にする機能がありませんでした。

詳細については、「[ファイルとフォルダー] ビュー」を参照してください。

基本の MSI インストールおよび InstallScript MSI インストールにおける PowerShell スクリプトからインストーラー セッションにアクセスできる機能

PowerShell カスタム アクション サポートが強化されました。今回より、実行中の基本の MSI インストールまたは InstallScript MSI インストールとの対話操作を可能にする、いくつかの cmdlet をサポートします: cmdlet を使って Windows Installer プロパティを取得および設定し、形式化された式の値を展開し、情報をログ ファイルに書き込むことができます。

この改訂された実装により、Windows Installer プロパティ IS_CLR_VERSION を利用して、カスタム アクションがダウンロードして PowerShell スクリプトを実行するセミコロンで区切られた .NET Framework のバージョン一覧を識別することができます。

詳細については、「PowerShell カスタム アクションの呼び出し」を参照してください。

トランスフォーム プロジェクトおよび MSI データベースでメジャー アップグレードを構成できる機能

トランスフォーム プロジェクトと MSI データベース プロジェクトには、今回より、メジャー アップグレードの作成に使用できる [アップグレード] ビューが含まれています。トランスフォーム プロジェクトまたは MSI データベース プロジェクトの間接編集モードでアップグレード エントリを追加するには、[アップグレード] ビューを開きます。[Windows Installer セットアップのアップグレード] ノードを右クリックしてから、[メジャー アップグレード アイテムの追加] を選択します。必要に応じて、右側のペインで設定を構成します。

詳しくは、次を参照してください:

- [アップグレード] ビュー
- メジャー アップグレード アイテムを追加する
- メジャーアップグレード
- アップグレード関する考慮事項
- 現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ

アドバンスト UI およびスイート/アドバンスト UI ログ ファイルへのパスワードの書き込みを防ぐ ISHiddenProperties プロパティの強化

ISHiddenProperties プロパティは、大文字と小文字を区別するプロパティ名のリストをセミコロン区切りで格納し、それらの値をデバッグ ログ ファイルに書き込まないようにできます。このプロパティを使って、パスワードその他の機密情報を含むプロパティのログ記録を防ぐことができます。ISHiddenProperties プロパティが強化されました。このプロパティを使って、次の状況下で値がログ記録されるのを防ぐのことができます:

- アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動したときに、エンド ユーザーが、コマンドラインを使ってプロパティの値を設定する場合。
- アドバンスト UI またはスイート/アドバンスト UI インストールがパッケージに渡すコマンドラインを使ってプロパティが構成される場合。これは [パッケージ] ビューの [共通] タブ、[操作] 領域で構成できます。

以前、ISHiddenProperties は、プロパティ値を変更するとログ記録される値に限ってログ記録を防ぎました。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールのウィザード インターフェイスで使用されているアンパサンドの解釈に関する強化

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード インターフェイスの特定の領域で、アンパサンドの解釈が更新されました。

インストールでウィザード インターフェイスの次の任意の領域でアンパサンド (&) を使用する場合、今回より、インストールはアンパサンドをリテラル文字として表示します。以前はアンパサンドがキーボード ショートカットの先頭の文字として解釈されました。この変更はまた、これらの文字列にアンパサンドを含む値に解決するプロパティが含まれている場合にも適用します。

- ウィザード ページまたは 2 番目のウィンドウのヘッダー領域にある文字列 – これは [ウィザード インターフェイス] ビューでウィザード ページまたはウィンドウの "タイトル" 設定で構成されます。
- ウィザード ページのキャプション バー – これは [ウィザード インターフェイス] ビューで [ウィザード ページ] ノードの "ウィザード キャプション" 設定で構成されます。
- コマンド リンク コントロールの [補足説明] 領域 – これはウィザード ページまたはウィンドウの "メモ" 設定で構成されます。

- イメージ コントロールの代替テキスト — これはウィザード ページまたはウィンドウにあるイメージ コントロールの "代替テキスト" 設定で構成されます。

ほとんどのラベル コントロールでは、今回より、"スタイル" 設定の下にある SS_NOPREFIX サブ設定のデフォルトの選択が True となりました。以前はデフォルトで False が選択されていました。このサブ設定が True 値に設定されていることで、これらのコントロールの文字列エントリに含まれるアンパサンドが誤ってキーボード ショートカットとして解釈されないようにし、コントロールの文字列に含まれるアンパサンドをウィザード インターフェイスのアンパサンドとして正確に表示します。これは、定義済みウィザード ページだけでなく、ビルトイン デフォルト ウィザード ページとウィンドウにも適用します。この SS_NOPREFIX サブ設定の変更に関する唯一の例外は、キーボード ショートカットを含むラベル コントロールです。たとえば、デフォルトの定義済み [Web 配布] ウィザード ページには、対応するラベルにキーボード ショートカットが含まれている可能性があるいくつかのテキスト ボックスがあります。したがって、このようなコントロールの SS_NOPREFIX サブ設定のデフォルト値は False のままとなります。

InstallShield ヘルプ ライブラリには、ウィザード インターフェイスにおけるアンパサンドおよびキーボード ショートカットの使用をサポートするためのプロジェクト構成方法について説明する、新しい「ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用」ヘルプ トピックが追加されました。

InstallShield 2014 以前のアドバンスト UI または スイート/アドバンスト UI プロジェクトを InstallShield 2025 にアップグレードすると、アンパサンドの解釈に関する前述の変更の多くが自動的に組み込まれますが、すべてが変更されるわけではありません。詳細については、「InstallShield 2014 以前のプロジェクトをアップグレードする」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールにおけるファイル連関連タイプの条件のチェックで複数のターゲット システム パスを検索できる機能強化

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、次のファイル関連タイプの条件チェックを使用できます:

- **ファイルの存在**—ターゲット システムをチェックして、特定のファイルの有無を確認します。
- **ファイルの比較**—ターゲット システムをチェックして、特定のファイルの特定の情報 (日付、バージョン番号、またはコンポーネント) を確認します。

これら両方の条件チェックが拡張され、ターゲット システム上の複数のパスでファイルを確認する条件を定義できるようになりました。以前、各条件はターゲット システム上にある特定の場所を 1 つだけ確認することができました。

このサポートを有効にするため、これらの条件に既存する "パス" 設定がファイルの名前だけを入力できるように拡張されました (つまり、オプションでパスが不要となりました)。パスを入力しない場合、新しい "検索パス" 設定を使って、インストールが実行時に特定のファイルを確認するセミコロン区切りのパス一覧を指定することができます。この設定にプロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールがこれらの式の値を拡張します。

たとえば、ターゲット システム上でレジストリで定義されたパスだけでなく、PATH 環境変数で定義されたすべてのディレクトリで指定されたファイルを検索するには、"検索パス" 設定に次の値を入力します:

```
[%PATH];[@Registry(HKLM¥SOFTWARE¥MyPath).KeyValue(MyValue)]
```

詳細については、次を参照してください。

- "ファイルの存在" 条件の設定

- "ファイルの比較" 条件の設定

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで、特定の種類のファイル関連条件のチェックと共に InstallShield 前提条件をインポートする際の機能強化

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、今回より、InstallShield 前提条件でサポートされている次の条件がサポートされています:

- ある前提条件で、[**ファイルが存在しているかしていないか**] または [**特定の日付を持つファイルが存在する**] という条件を使用していて、かつ、そのファイルのパスの一部に対して、角括弧で囲まれれたレジストリ キーを参照している場合、基本の MSI、InstallScript、InstallScript MSI インストールでは、レジストリ キーがレジストリ値で解決されます。アドバンスト UI およびスイート/アドバンスト UI インストールにおける条件の形式化された式のサポートが拡張されたため、これらの種類の前提条件は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに適切にインポートされ、実行時に解決されます。以前、角括弧に囲まれたレジストリ キーが実行時に解決されず、これらの条件が常に False と評価されました。
- ある前提条件で、[**ファイルが存在しているかしていないか**] または [**特定の日付を持つファイルが存在する**] という条件を使用していて、そのファイルのパスを省略した場合、基本の MSI、InstallScript、InstallScript MSI インストールでは、ターゲット システムで PATH 環境変数で定義されているすべてのディレクトリで、指定されたファイルが検索されます。アドバンスト UI およびスイート/アドバンスト UI インストールにおける条件の形式化された式のサポートが拡張されたため、これらの種類の前提条件は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに適切にインポートされ、実行時に解決されます。以前、ファイルへのパスが解決されなかったため、この種類の条件は常に False と評価されました。

Oracle Instant Client 用の InstallShield前提条件は、ファイル関連の条件を使用する前提条件の例です。これらの種類の InstallShield 前提条件をアドバンスト UI またはスイート/アドバンスト UI プロジェクトにインポートすると、今回より、[パッケージ] ビューの "検出条件" 設定でファイル関連の条件が正しく構成されます。

詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

オートメーション インターフェイスを使ってプロジェクトをアップグレードできる機能

オートメーション インターフェイスには、InstallShield の以前のバージョンから現在のバージョンへのプロジェクトのアップグレードがサポートされています。オートメーション インターフェイスを使ってプロジェクトをアップグレードするには、ISWiProject オブジェクトの ForceUpgrade メソッドを使用します。

この強化機能は、アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、および スイート/アドバンスト UI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- ISWiProject オブジェクト
- ISWiProject オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

オートメーション インターフェイスを使って、リリースに仮想マシン構成を選択できる機能

オートメーション インターフェイスでは、指定のリリースに仮想マシン構成を選択することができます。これによって、ビルド時に選択されたリリースを仮想マシン (VM) に配布する際に使用する構成を指定することができます。

基本の MSI、InstallScript、および InstallScript MSI プロジェクトでは、ISWiReleases オブジェクトに新しい読み書きプロパティが含まれています。スイート/アドバンスト UI プロジェクトでは、ISWiSuiteReleases オブジェクトに同じ新しい読み書きプロパティが含まれています。

- VMConfig—このプロパティは、リリースを VM に配布するときに使用する VM 構成設定グループの名前を取得または設定します。
- VMSnapShot—このプロパティはオプションで、リリースの配布に使用するスナップショットの名前を取得または設定します。

  VM 構成にスナップショットが指定されていない場合、InstallShield は特定のスナップショットには戻りません。InstallShield は特定のスナップショットに戻さずに VM の電源をオンにして、VM にインストールをコピーします。
- VMMachineUserName—このプロパティは VM 構成のユーザー名を取得または設定します。
- VMMachinePassword—このプロパティは VM 構成のパスワードを取得または設定します。
- VMStageMachineCopyPath—このプロパティは、リリースの配布先となる VM 上の場所を取得または設定します。パスの最後のサブフォルダーとして、InstallShield が VM 上に作成するパスを使用できます。その他のパスは既存していなくてはなりません。
- VMStagePostBuild—このブール型プロパティは、ビルドが成功するたびに選択されたリリースを、InstallShield で自動的に配布するかどうかを示します。

この機能強化は、InstallShield の Premier Edition で使用できます。

詳しくは、次を参照してください:

- ISWiRelease オブジェクト
- ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

ISICE 11 に追加された強化機能

InstallShield 内部性合成評価プログラム 11 (ISICE11) が改良されました。プロジェクトに含まれる実行可能ファイルに trustInfo 要素の asm.v2 エントリが含まれている有効なマニフェストが含まれているとき、検証中に ISICE11 が発生することがなくなりました。以前、asm.v3 エントリはチェックされましたが、asm.v2 エントリはチェックされませんでした。

詳細については、「ISICE11」を参照してください。

基本の MSI、DIM、およびマージ モジュールプロジェクトで強化された [ダイアログ] ビュー: 各コントロールのすべての "動作" 設定を単一のグリッドで表示

[ダイアログ] ビューが強化されています。実行時ダイアログ上の各ユーザー インターフェイス コントロールの動作 (イベント、サブスクリプション、および条件) を構成できるすべての設定が、今回より、単一のグリッドで表示されます。以前は、ビューの右下に表示される個別の [イベント]、[サブスクリプション]、および [条件] タブに設定が表示されました。

イベント、サブスクリプション、または条件をコントロールに追加するには、[ダイアログ] ビューで、コントロールを含むダイアログの下にある [動作] ノードを選択します。ダイアログで使用されているコントロールが中央ペインに表示されます。ここで構成するコントロールを選択します。次に、右側ペインに表示される "イベント"、"サブスクリプション"、および "条件" 設定を使って適切な動作を構成します。

この強化機能は、基本の MSI、DIM、およびマージ モジュール プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- 基本の MSI プロジェクトでダイアログ レイアウトを編集する
- 基本の MSI ダイアログでコントロール イベントを使用する

- 基本の MSI ダイアログでサブスクリプションを使用する

強化されたフォルダー ビューで、現在のプロジェクトに含まれる各アイテムの合計数を表示

InstallShield のフォルダー ビューでは、今回より、プロジェクトに含まれるコンテンツの概要を確認することができます。

たとえば、一部のプロジェクト タイプで [ファイルとフォルダー] ビューと [再配布可能ファイル] ビューを含むフォルダービューである [アプリケーション データ] ビューは、現在のプロジェクトに含まれるファイルの数、マージ モジュールの数、および InstallShield 前提条件の数を示します。[ショートカット] や [レジストリ] などのビューを含むフォルダー ビューである [システム構成] ビューは、ショートカットの数、レジストリ キーの数、およびレジストリ値の数といった概要データを表示します。

未使用のコンポーネントを自動的に削除する、新しいマシン全体の設定

InstallShield の [オプション] ダイアログにある [ファイル ビュー] タブに、新しく **[未使用のコンポーネントをクリーンアップする]** チェックボックスが追加されました。このチェック ボックスを使って、InstallShield でプロジェクトから未使用のコンポーネントを自動的に削除するかどうかを指定することができます。

このチェックボックスが選択されているときに、コンポーネントのすべてのファイルを削除して、そのコンポーネントが別の領域で必要でない場合、そのコンポーネントは自動的に削除されます。

この新しいチェックボックスは、マシン全体に影響する設定です。このチェックボックスはデフォルトでクリアの状態です。したがって、デフォルトで未使用のコンポーネントは自動的に削除されません。

詳細については、「[ファイル ビュー] タブ」を参照してください。

Binary テーブルのファイルを使用するカスタム アクションを削除する強化された機能

Binary テーブルのファイルを使用するカスタム アクションをプロジェクトから削除するとき、InstallShield は、Binary テーブルのファイルがプロジェクト内の別のカスタム アクションで参照されているかどうか判別します。別のカスタム アクションで参照されていない場合、InstallShield はカスタム アクションを削除すると同時に、Binary テーブルからエントリを削除するかどうかを指定できるプロンプトを表示します:

- デフォルトの選択である [いいえ] ボタンは、カスタム アクションのみを削除できます。
- [はい] ボタンは、カスタム アクションと Binary テーブル エントリの両方を削除できます。
- [キャンセル] ボタンは、カスタム アクションと Binary テーブル エントリの両方が削除されないようにします。

プロジェクト内の 1 つ以上のアクションが Binary テーブル内のファイルを参照する場合、InstallShield は削除が指定されているカスタム アクションのみを削除します。Binary テーブル エントリは削除されません。このシナリオでは、InstallShield はカスタム アクションを削除するときにプロンプトを表示しません。

以前、前述の状況下でカスタム アクションを削除すると、Binary テーブルのエントリがプロジェクト内の別のエントリで参照されているかどうかにかかわらず、常に Binary テーブルのエントリを削除するかどうかを問い合わせるプロンプトが表示されました。このプロンプトのデフォルト ボタンは [はい] で、カスタム アクションと Binary テーブルのエントリの両方が削除されます。このデフォルトの動作によって、**Binary** テーブルのエントリの削除が誤って選択されることがありました。

この強化機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール および トランスフォーム プロジェクト タイプに適用します。

新しいスタンドアロン ビルド レジストリ エントリが InstallShield バージョン情報を表示

スタンドビルド インストールは、今回より、InstallShield インストールが HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥*VersionNumber*¥Professional の下にインストールするのと同じ InstallShield バージョン関連レジストリ値をインストールします。これによって、使用中の環境内のすべての開発マシンとビルド マシンで、製品の同じバージョンを使用しているかどうかを素早く検証することができます。関連するレジストリ値:

- **製品コード**—このレジストリ値は、InstallShield IDE またはスタンドアロン ビルドのどちらがインストールされているかによって異なります。これらのツールのメジャー アップグレードの際に更新されます。
- **MPIndicator**—適切な場合、このレジストリ値はメンテナンスパックを示します。たとえば、*サービス パック 1*。
- **SP**—適切な場合、このレジストリ値はサービス パックの番号を示します (たとえば、*1*)。

InstallShield 2025 および InstallShield 2025 スタンドアロン ビルドの場合、レジストリ値は次のキーの下にインストールされます:

HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥Automation Interface Version¥Professional

InstallShield 2014 SP1 の新しい機能

InstallShield には、以下のような強化機能が搭載されています。

ポルトガル実行時文字列の更新

ポルトガル語 (ブラジル) および ポルトガル語 (ポルトガル) のデフォルトの実行時文字列は、1990 年のポルトガル語新正書法に基づいて更新されています。この新正書法は、2009 年以来 6 年間の移行期間を経て、2014 年の終りに完全実施されます。

InstallShield 2014 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

新しい .NET Framework 4.5.1 用 InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft .NET Framework 4.5.1 (完全版)
- Microsoft .NET Framework 4.5.1 (Web インストーラー)

これらの前提条件は、サポートされているターゲット システムに、.NET Framework 4.5.1 をインストールします。

Microsoft SQL Server 2014 サポート

InstallShield に、SQL Server 2014 Database 上で SQL スクリプトを実行できるサポートが追加されました。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2014 が含まれています。

インストールで SQL Server 2014 をターゲットするとき、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、SQL Server 2014、SQL Server 2014 Express、および SQL Server 2014 Express LocalDB のインスタンスが表示できるようになりました。また、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、指定された SQL Server 2014 サーバー上のカタログが表示されるようになりました。

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

Microsoft SQL Server 2014 の前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい SQL Server 2014 関連の InstallShield 前提条件が含まれています:

- Microsoft SQL Server 2014 Express RTM (x64)
- Microsoft SQL Server 2014 Express RTM (x86 & x64Wow)
- Microsoft SQL Server 2014 Express RTM (x86)
- Microsoft SQL Server 2014 Express RTM LocalDB (x64)
- Microsoft SQL Server 2014 Express RTM LocalDB (x86)

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

新しい App-V 5.0 SP2 用 InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft App-V 5.0 SP2 Desktop Client (x86)
- Microsoft App-V 5.0 SP2 Desktop Client (x64)

これらの InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft から再配布可能ファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。

Microsoft Visual C++ 2012 Update 4 用の新しい InstallShield 前提条件

InstallShield には、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加可能な新しい InstallShield 前提条件が含まれています:

- Microsoft Visual C++ 2012 Update 4 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 Update 4 再配布可能パッケージ (x64)

これらの前提条件は、サポートされているターゲット システムに様々なテクノロジをインストールします。

アドバンスト UI およびスイート/アドバンスト UI インストールのメンテナンス モードからアップデートをチェックできる機能

アドバンスト UI およびスイート/アドバンスト UI インストールでは、ユーザーがメンテナンス ウィザード ページにある新しい [アップデート] ボタンをクリックして製品のアップデートをチェックできる機能がサポートされています。Web サイトでアップデートが利用可能なとき、エンド ユーザーがそのアップデートの取得を選択した場合、インストールがそれをダウンロードし、デジタル署名を検証してから起動します。

この機能を使用するには、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで自動アップデートがサポートされていなくてはなりません。ビルド時に InstallShield は isupdate.xml という名前のメタ ファイルを作成して、プロジェクトのビルド フォルダー内の UpdateMetadata サブフォルダーに配置します。このメタ ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるバージョンとそのダウンロード場所を識別します。製品のアップデートのリリース準備が完了したら、アップデート済みのアドバンスト UI またはスイート/アドバンスト UI インストールと共に、この isupdate.xml ファイルもサイトにアップロードします。

実行時、エンド ユーザーがメンテナンス モードを実行すると、スイート エンジンがメタデータ ファイルのアップデートをチェックします。ダウンロード可能なアップデートがある場合、スイート エンジンが新しい ISUpdateAvailable プロパティを 1 に設定して、[アップデート] ボタンが含まれるメンテナンス ウィザードの新しいページ MaintenanceUpdateWelcome を表示します。また、スイート エンジンは新しい ISUpdateVersion プロパティをダウンロード可能なスイート/アドバンスト UI またはアドバンスト UI インストールのバージョンと等しく設定します。

アップグレードが使用できないとき、スイート エンジンは [アップデート] ボタンが含まれていない標準の MaintenanceWelcome ウィザード ページを表示します。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

サブフォルダーにサポート ファイルを含める機能

InstallShield では、今回より、サポート ファイル用のカスタム フォルダー構造を指定することができます。[サポート ファイル] ビューの語固有ノードの 1 つまたは言語非依存ノードの 1 つの下にサブフォルダーを追加するには、そのノードを右クリックしてから [新しいフォルダー] をクリックします。サブフォルダーが追加され、必要に応じてその名前を変更することができます。ネストされたフォルダー構造を追加することもできます。ファイルをサブフォルダーに追加するには、右側のペインを選択してから、[ファイルの挿入] をクリックします。実行時、インストールがサポート フォルダーとファイルをターゲット システム上の一時ディレクトリである SUPPORTDIR にコピーして、製品のインストール処理中に使用できるようにします。サポート フォルダーとファイルは、インストールが完了すると削除されます。

強化された [サポート ファイル] ビューは、アドバンスト UI、基本の MSI、スイート/アドバンスト UI、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

詳細については、「サポート ファイルを追加する」を参照してください。

オートメーション インターフェイスには、サポートファイルのサブフォルダー機能も含まれています。ISWiSetupFile オブジェクトには、SUPPORTDIR 内でサポートファイルのターゲット ディレクトリのパスを指定または取得するのに使われる新しい読み取り-書き込み Target プロパティが含まれています。詳細については、「ISWiSetupFile オブジェクト」を参照してください。

IIS サーバーおよびクラウドに Web 配置パッケージを配置するためのサポート

スイート/アドバンスト UI インストールには、今回より、IIS サーバーおよびクラウドに Web 配置パッケージを展開するためのビルトイン サポートが含まれています。Web 配置パッケージは、Visual Studio などの IIS または Web アプリケーション開発環境を使って作成できます。

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加するには、[パッケージ] ビューを使います。[パッケージ] ビューにパッケージを追加および選択すると、右側のペインにパッケージに関連する設定が表示されます。右側のペインの [共通] タブにある [構成] 領域には、Web 配置の構成関連の設定、たとえばインストール先 (多くの場合はリモート サーバー) や資格情報などが含まれています。さらに、[構成] 領域には Web 配置パッケージのパラメーター XML ファイルで定義されるパラメーターの設定も含まれています。デフォルトでは、スイート/アドバンスト UI プロパティが構成設定すべての構成設定に使用され、エンド ユーザーはウィザード インターフェイスまたはコマンドラインを使って実行時にそれらを設定することができます。

Web 配置パッケージをプロジェクトに追加すると、そのパッケージ用の定義済みの Web 配置ウィザード ページがプロジェクトに追加されます。Web 配置ウィザード ページを使って、エンド ユーザーは特定のパッケージをローカル IIS サーバー、リモート サーバー、またはクラウドベースのサーバーのいずれに配置するのかを指定することができます。また、パブリッシャー プロファイル ファイル (.publishsettings) から構成設定をロードできるようにもなります。エンド ユーザーが Web 配置パッケージのパラメーター XML ファイルで定義されているパラメーターを構成できるようにするには、必要に応じてウィザード インターフェイスにコントロールを追加して、それらのコントロールを [パッケージ] ビューの対応するパラメーター設定に指定されているプロパティに関連付けます。

Web 配置パッケージは一般的に、初回インストールと同様、または既存の古いバージョンあるいは同一バージョンを上書きする場合と同様に動作するように設計されています。その結果、Web 配置パッケージではアンインストールを行うことができません。つまり、スイート/アドバンスト UI インストールはこの種類のパッケージのメンテナンスを行うことができません。1 つ以上の Web 配置パッケージ、または 1 つ以上の従来型パッケージ (メンテナンス オプションが含まれている) をプライマリ パッケージとして含むスイート/アドバンスト UI インストールの作成を避けることが推奨されます。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する
- [パッケージ] ビュー

スイート/アドバンスト UI プロジェクトに InstallShield プロジェクトをパッケージとして含める機能のサポート

新しい InstallShield では、InstallScript インストールを、パッケージとして、スイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加することができます。InstallShield プロジェクトとして、基本の MSI プロジェクトまたは InstallScript プロジェクトのどちらかを使用できますが、これらのプロジェクトはスイート/アドバンスト UI プロジェクトと同じバージョンの InstallShield で保存されている必要があります。

1 つ以上の InstallShield プロジェクト パッケージを含むスイート/アドバンスト UI プロジェクトのリリースをビルドすると、関連する InstallShield プロジェクト内の指定されたリリースが最初にビルドされて、生成されるスイート/アドバンスト UI インストールにそれらがパッケージとして含まれます。InstallShield は基本の MSI プロジェクトおよび InstallScript プロジェクトの出力を .msi パッケージおよび .hdr パッケージとしてスイート/アドバンスト UI インストールに追加します。

[パッケージ] ビューを使って、スイート/アドバンスト UI プロジェクトに InstallShield プロジェクトをパッケージとして追加できます。このビューではまた、スイート/アドバンスト UI プロジェクトにパッケージとして含めるのが、どの製品構成 (基本の MSI プロジェクト パッケージ) か、またはどのリリース (基本の MSI パッケージと InstallScript プロジェクト パッケージの両方) かを指定します。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する
- [パッケージ] ビュー

スイート/アドバンスト UI インストールにおける InstallScript アクションのサポート

スイート/アドバンスト UI インストールでは、今回より、インストールに含めるパッケージの範囲を超えた様々な実行時タスク行うための InstallScript アクションを起動できるビルトインサポートが提供されています。スイート/アドバンスト UI ベースの InstallScript アクションは、基本の MSI プロジェクトの InstallScript カスタム アクションとほぼ同じ要領で機能します。

スイート/アドバンスト プロジェクトで今回から使用可能な [InstallScript] ビューを使って、プロジェクトに InstallScript ファイル (.rul) を追加し、InstallScript 言語を使って関数を書き込むことができます。スイート/アドバンスト UI アクションで InstallScript 関数を呼び出す場合、基本の MSI プロジェクトで InstallScript カスタム アクションをプロトタイプ化するときと同様に、export キーワードを使って関数をプロトタイプ化します。

InstallScript コードをプロジェクトに追加してから、[イベント] ビューを使ってコードを呼び出す InstallScript を追加します。つぎに、実行時にアクションを起動するタイミングをスケジュールします。たとえば [イベント] ビューを使って、スイート/アドバンスト UI エンジンが管理する 1 つ以上のビルトイン イベントの実行中に InstallScript アクションをスケジュールできます。

スイート/アドバンスト UI プロジェクトで InstallScript を使用するとき、いくつかの制限があります。たとえば、InstallScript UI 関連の関数と InstallScript 実行時パス変数 (PROGRAMFILES および WINDIR) をスイート/アドバンスト UI プロジェクトで使用することはできません。

この機能は、InstallShield の Premier Edition で提供されています。

詳しくは、次を参照してください:

- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する
- スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について
- スイート/アドバンスト UI アクションの設定を構成する
- スイート/アドバンスト UI アクションをスケジュールする
- スイート/アドバンスト UI およびアドバンスト UI の対話関数。

スイート/アドバンスト UI インストールにおける マネージコード アクションのサポート

スイート/アドバンスト UI インストールでは、今回より、インストールに含めるパッケージの範囲を超えた様々な実行時タスク行うためのマネージコード アクションを起動するビルトイン サポートが提供されています。このアクションの種類は、Visual Basic .NET または C# などのマネージ コードで書かれた、。NET アセンブリのパブリック メソッドを呼び出します。これらのスイート/アドバンスト UI ベースの InstallScript アクションは、ス

イート/アドバンスト UI インストールに含まれる DLL アクションと同じ機能を果たします。つまり、スイート エンジンは呼び出しを行うマネージ関数で 1 つのパラメーターを見つけます。このパラメーターは ISuiteExtension インターフェイスで、System.Object として渡されます。戻り値の型は 32 ビット整数型でなくてはなりません。

マネージコード アクションをスイート/アドバンスト UI プロジェクトに追加するには、[イベント] ビューを使います。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する
- スイート/アドバンスト UI インストールでマネージ コード アクションを使用する
- スイート/アドバンスト UI アクションの設定を構成する
- スイート/アドバンスト UI アクションをスケジュールする

ウィザード インターフェイス コントロールを使用したときにエンド ユーザーがトリガするスイート/アドバンスト UI イベント アクションを起動できる機能

スイート/アドバンスト UI インストールでは、ウィザード インターフェイス コントロール上でエンド ユーザーがクリックその他の操作をおこなったときに起動される、[イベント] ビューで定義される実行時のアクションがサポートされています。たとえばこの機能を使って、ウィザード インターフェイス上のボタンをエンド ユーザーがクリックしたときに、インストールで実行可能ファイルを実行、InstallScript コードを実行、マネージ アセンブリのマネージ メソッドを呼び出し、PowerShell スクリプトを実行、またはスイート/アドバンスト UI プロパティの設定を行うことができます。

プロジェクトにアクションを追加するには、[イベント] ビューを使います。

ウィザード インターフェイス コントロールがクリックまたは使用されたときに実行するアクションをスケジュールするには、[ウィザード インターフェイス] ビューで構成するコントロールを選択してから、"クリック" 設定または "イベント" 設定の下にあるアクション関連の設定を使って起動するアクションを選択します。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- ウィザード インターフェイス内の要素のアクションを構成する
- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

ウィザード ページとウィンドウが表示/非表示のときにスイート/アドバンスト UI アクションを起動できる機能

スイート/アドバンスト UI インストールは、ウィザード ページまたは 2 番目のウィンドウが開いているまたは閉じているときに起動された実行時アクションをサポートします。この機能によって、コンボボックスの値を動的に取得するなど、インストールで任意の初期設定処理が可能となります。また、実行時にウィザード インターフェイスが表示される前に実行されるアクションをスケジュールすることも可能です。

プロジェクトにアクションを追加するには、[イベント] ビューを使います。

ウィザード ページが開くまたは閉じるときに実行するアクションをスケジュールするには、[ウィザード インターフェイス] ビューで構成するウィザード ページを選択してから "開始ページ" 設定または "終了ページ" 設定で適切なアクションを選択します。

2 番目のウィンドウが開くまたは閉じるときに実行するアクションをスケジュールするには、[ウィザード インターフェイス] ビューで構成する 2 番目のウィンドウを選択してから "開始ウィンドウ" 設定または "終了ウィンドウ" 設定で適切なアクションを選択します。

この機能は、InstallShield の Premier Edition で提供されています。

詳細については、次を参照してください。

- ウィザード インターフェイス内の要素のアクションを構成する
- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

アドバンスト UI およびスイート/アドバンスト UI インストールのログ ファイルへのパスワードの書き込みを防ぐ新しいプロパティ

/debuglog パラメーターを使って アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動すると、スイート エンジンがデバッグ ファイルを生成します。デフォルトで、デバッグ ログ ファイルには、アドバンスト UI またはスイート/アドバンスト UI プロパティの値が含まれます。

場合によって、スイート エンジンが特定のプロパティの値をデバッグ ログ ファイルに書き込むことを防ぐ必要があります。たとえば、パスワードその他の機密情報を含むプロパティのログ記録を防ぐ必要があります。この状況に対応するため、アドバンスト UI およびスイート/アドバンスト UI インストールでは ISHiddenProperties という新しいプロパティがサポートされています。このプロパティに大文字と小文字を区別するプロパティ名のリストをセミコロン区切りで設定して、それらの値をデバッグ ログ ファイルに書き込まないようにできます。

ISHiddenProperties は、プロパティ値を変更するとログ記録される値に限ってログ記録を防ぐのに使用できます。その他の方法でプロパティがログ記録される場合 (ISuiteExtension::LogInfo を使う場合など)、スイート エンジンでログ記録を防ぐことはできません。したがって、ログ ファイルにプロパティ値を書き込むために作成するすべてのコードは、ISHiddenProperties を読み込んで、そのプロパティ値がログ記録されるべきかを判別する必要があります。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ」を参照してください。

強化されたアドバンスト UI およびスイート/アドバンスト UI プロジェクトの置換構文

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域では、プロパティ名を角括弧で囲むことができます。角括弧で囲まれたプロパティ名は、実行時に適切な値で置換されます。実行時の構文では、新しい種類の置換がサポートされています:

- **特殊文字** – 特殊文字の前に円記号を付けて、結果の文字列を角括弧で囲むと、実行時に特殊文字が解決されます。たとえば、[¥[]Text[¥]] と入力すると [Text] に置換されます。
- **ヌル文字** – 角括弧で囲まれたチルダ (つまり [~]) は、文字列に埋め込まれたヌル文字として解決します。
- **環境変数** – 環境変数の名前の前にパーセント記号を付けて、結果の文字列を角括弧で囲むと、実行時に環境変数が解決されます。たとえば、[%PATH] は PATH 環境変数の値として解決します。
- **再帰的なプロパティの解決** –プロパティの解決をさらに角括弧で囲むと、解決済みのプロパティの値が解決されます。たとえば [[PROPERTY1]] は、PROPERTY1 にその名前が格納されているプロパティの値に解決し、PROPERTY1 が PROPERTY2 を含む場合、解決された値は PROPERTY2 に格納されている値となります。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

ファイルとフォルダーの削除に関するサポート

InstallShield では、実行時にターゲット システムから削除するファイルとフォルダーを簡単に指定することができるビルトイン サポートが提供されています。このファイルとフォルダーの削除機能は、アプリケーションによって作成されるファイルの削除など、インストールが追跡を行わない処理に使用すると便利です。

ファイルまたはフォルダーの削除は、次のイベントの 1 つにスケジュールできます:

- ファイルまたはフォルダーのコンポーネントがインストールされるとき
- ファイルまたはフォルダーのコンポーネントがアンインストールされるとき
- ファイルまたはフォルダーのコンポーネントがインストールまたはアンインストールされるとき

削除されるアイテムがフォルダーの場合、そのフォルダーが空の場合のみ削除されます。

InstallShield では、プロジェクトからのファイルまたはフォルダーのを削除を構成するさまざまな方法が提供されています:

- [ファイルとフォルダー] ビューで、削除するファイルまたはフォルダーを含むフォルダーを選択します。次に、**[インストール先コンピューターのファイル]** ペインを右クリックしてから [ファイルの削除を追加] をクリックします。
- [セットアップのデザイン] ビュー (インストール プロジェクト) または [コンポーネント] ビューで、削除するファイルまたはフォルダーが含まれているコンポーネントのノードを拡張してから、[ファイル] サブノードをクリックします。次に、[ファイル] ペインを右クリックしてから [ファイルの削除を追加] をクリックします。

どちらの方法も、使用可能な削除の設定を構成できる [プロパティ] ダイアログ ボックスが表示されます。

以前は、削除するファイルとフォルダーを示す方法は、[ダイレクト エディター] ビューを使って RemoveFile テーブルに手動でエントリを入力する方法のみでした。

この機能は、基本の MSI、DIM、 InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- ターゲット システムからファイルとフォルダーを削除する
- ファイル 削除の [プロパティ] ダイアログ ボックス

オートメーション インターフェイスに、ファイルとフォルダーを削除するための新しい ISWiRemoveFile オブジェクトと新しい ISWiRemoveFiles コレクションが追加されました。また、ISWiComponent オブジェクトで新しい 3 つのメソッドが使用できます: AddRemoveFile は現在のコンポーネントにファイルの削除エントリを追加し、RemoveRemoveFile は現在のコンポーネントからファイルの削除エントリを削除し、ISWiRemoveFiles は現在のコンポーネントと関連付けられたファイルの削除エントリすべてを含む ISWiRemoveFiles コレクションを返します。

これらのオートメーション インターフェイスの強化機能は、基本の MSI、InstallScript、InstallScript MSI、およびマージ モジュール プロジェクト タイプで利用できます。

詳細については、次を参照してください。

- ISWiRemoveFile オブジェクト
- ISWiRemoveFiles コレクション
- AddRemoveFile メソッド
- RemoveRemoveFile メソッド

- ISWiRemoveFiles メソッド
- ISWiComponent オブジェクト

DPI 対応インストールのサポート

InstallShield では、今回より、様々な高 DPI ディスプレイ環境のターゲット システムにおいて、実行時のユーザー インターフェイス要素を適切に調整できる DPI 対応インストールを作成して、インストールのユーザー インターフェイスにより統一性のある魅力的な外観を実現できます。使用可能なサポートおよび強化内容は、使用するプロジェクトの種類によって異なります。

アドバンスト UI およびスイート/アドバンスト UI プロジェクト

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける DPI 対応には、2 つの基本事項があります:

- エンジンが使用するアイテムは、システムの DPI 設定にしたがって調整されます。つまり、ターゲット システムで DPI 200% が設定されている場合、チェックボックス コントロールも同様に調整されます。
- エンジンは、ウィザード インターフェイスに含まれたイメージその他のリソースを表示するとき、スケール因子と言語を考慮します。

たとえば、エンジンがスケール因子を 150 と判断したとき、適切な言語が英語である場合は、ファイル名またはパス名に *scale-150* の文字列を含むリソースを探します。ターゲット システム上で image.png を検索するとき、次にリストされている順番でパスを検索します:

1. [SUPPORTDIR]0409¥**scale-150**¥image.png
2. [SUPPORTDIR]0409¥image.**scale-150**.png
3. [SUPPORTDIR]0409¥image.png
4. [SUPPORTDIR]**scale-150**¥image.png
5. [SUPPORTDIR]image.**scale-150**.png
6. [SUPPORTDIR]¥image.png

正しい DPI よりも一致する言語が優先します。

InstallShield はアドバンスト UI およびスイート/アドバンスト UI プロジェクトに含まれるビルトイン デフォルト イメージに、scale-150 および scale-200 イメージを含みます。プロジェクトにカスタム リソースを含める場合、[サポート ファイル] ビューを使って適切なスケールのイメージを追加します。

InstallScript および InstallScript MSI プロジェクト

InstallScript プロジェクトと InstallScript MSI プロジェクトにおける DPI 対応には、次の点が含まれています:

- ビルトイン デフォルト ダイアログのすべてのイメージは、200% に上手く調整可能なイメージに差し替えられています。
- 今回より、コントロール ID 1200 を持つスタティック ダイアログ コントロールは、透明色の使用が可能な .png イメージをサポートします。これは、ウェルカム ダイアログのような、外部スキン ダイアログの左側にデフォルトで表示されるコンピューター イメージに適用します。内部ダイアログのバナー イメージにも適用します。

  さらに、これらの同じコントロールは、各イメージとそれに関連するスケール因子を識別するための新しいフォーマットをサポートします。

@@ResourceID;ScaleFactor

ResourceID は、.png イメージ (リソース タイプ PNG として格納される) またはビットマップ イメージ (リソース タイプ RT_BITMAP として格納される) のいずれかを検出するためのリソース識別番号を示します。*ScaleFactor* は、イメージが意図されている DPI スケール率を示します。

たとえば、100% (96 DPI)、125% (120 DPI)、150% (144 DPI)、200% (192 DPI) となります。イメージに指定されているスケール因子が 200 の場合、200% DPI スケール以下で実行中のターゲット システム上で、イメージは縮小表示されます。200% ターゲットシステム上で、これは 1:1 の割合で表示されます。イメージに指定されているスケール因子が 100 の場合、200% DPI スケールで実行中のターゲット システム上で、イメージは拡大表示されます。

ビットマップ イメージ (.bmp) を識別する以前のフォーマットも使用できますが、スケーリングまたは .png イメージがサポートされていません:

@BitmapResourceID;TransparentFlag;3-DFlag;<unused>;TransparentColorKey

- ダイアログ ヘッダーのバナーおよび外部パネルのアイコン イメージ用のすべての ID 1200 スタティック コントロールには、今回より新しいイメージ文字列フォーマットが使用されます。InstallShield プロジェクトまたは InstallScript MSI プロジェクトを InstallShield 2025 にアップグレードするとき、以前のバージョンの InstallShield でダイアログを編集した場合、プロジェクトでサポートされているすべての言語のイメージ文字列を新しいフォーマットに編集する必要があります。
- InstallScript 関数 **SdOptionsButtons** は、イメージを識別する新しい文字列フォーマットをサポートするようアップデートされています:

  @@ResourceID;ScaleFactor

  以前は、次のフォーマットが使用されていました:

  @BitmapResourceID;BitmapIconFlag;TransparentColor

- InstallScript 関数 **GetSystemInfo** の nItem パラメーターは、2 つの新しい定数をサポートします:
  - **SYSTEM_DPI** — nvResult パラメーターのシステム DPI 値を返します。
  - **SYSTEM_DPI_SCALING** — nvResult パラメーターの DPI スケール値 を返します。

詳細は、次を参照してください:

- GetSystemInfo
- SdOptionsButtons
- InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるデフォルト イメージについての背景情報

基本の MSI プロジェクト

Setup.exe 起動プログラムが実行時に表示するダイアログに表示されるビルトイン イメージが、より大きな領域に上手く調整されるイメージに差し替えられています。これには、初期設定ダイアログに表示されるイメージも含まれています。

ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにインストールを配布できる機能

インストールのビルドが成功するたびに InstallShieldが仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態になるようプロジェクトを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。このテスト準備機能により、手作業で行う処理を減らし、テストにかかる時間を短縮できます。

VM は Microsoft Hyper-V Server、VMware ESX または ESXi Server、または VMware Workstation を指定できます。

VM 情報を指定するには、[リリース] ビューにあるリリースの [イベント] タブにある [仮想マシン] 領域を使用します。設定のうちの 1 つを使って、ビルドが成功するたびに指定の VM にリリースを配布するかどうかを指定します。

必要なときにリリースを VM に配布するには、[リリース] ビューで該当するリリースを右クリックしてから、[VM に配布] をクリックします。

[イベント] タブにある "構成" 設定を使用して VM の詳細を構成すると、InstallShield によって 指定されたデータが VMConfigurations.xml という名前のファイルに書き込まれます。この VMConfigurations.xml ファイルは、マシン全体に適用するファイルで、一度構成を行うと他の InstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルを Standalone Build で使用することも可能です。

サポートされている VM の1 つに配布する場合、InstallShield では必要な仮想化テクノロジとの対話が必要です。VMWare 仮想化テクノロジ (VMware ESX、ESXi Server oまたは VMware Workstation) を使用する場合、InstallShield と同じマシン上に VMware VIX API が インストールされていなくてはなりません。マシン上に VMware Workstation をインストールするか、http://www.vmware.com/support/developer/vix-api から VMware VIX API をダウンロードおよびインストールすることができます。

VMware Workstation を使用する場合、InstallShield と同じマシン上に VMware Workstation をインストールすることが推奨されます。これによって、InstallShield が特定の VMware Workstation バージョン用に設計された VIX API のバージョンを使用します。VIX API の新しいバージョンも使用できる場合がありますが、現在使っている VMware Workstation にバンドルされている VIX API のバージョンを InstallShield で使用する方法が最も推奨されます。

この機能は、基本の MSI、InstallScript、InstallScript MSI、および スイート/アドバンスト UI プロジェクト タイプで使用できます。

この機能は、InstallShield の Premier Edition で提供されています。

詳しくは、次を参照してください:

- ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する
- リリースの [イベント] タブ
- [仮想マシンの構成を編集] ダイアログ ボックス
- リリースの配布用の仮想マシン設定を共有する

強化機能

InstallShield には、以下のような強化機能が搭載されています。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで、コンボボックスおよびリストボックス コントロールの設定におけるユーザビリティーの向上

InstallShield では、コンボボックス コントロールとリストボックス コントロールをアドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード ページおよび 2 番目のウィンドウに追加することができます。これらのどちらかの種類のコントロールを作成するプロセスでは、2 つのプロパティ (コントロールで表示するオプションのリストを定義するプロパティ、および実行時にエンド ユーザーが選択するオプションを格納するためにインストールが使用するプロパティ) を使用します。これまでは [ウィザード インターフェイス] ビューと [プロパティ マネージャー] ビューを使用する必要がありましたが、これらのプロパティの両方を InstallShield の [ウィザード インターフェイス] ビュー内部から構成することができます。

[ウィザード インターフェイス] ビューで、コンボボックスまたはリストボックス コントロールの "コンテンツ プロパティ" 設定には省略記号ボタン (...) が含まれていて、これをクリックしてコントロールに表示するオプションのリストを定義することができます。このボタンをクリックすると、シンプルな [リスト オプションを編集] ダイアログ ボックスが開いて、選択されたコントロールに表示するオプションのリストおよび関連する値を指定できます。

以前は、[プロパティ マネージャー] ビューを使ってオプションのリストが含まれるプロパティおよび対応するプロパティ値を定義する必要がありました。プロパティを定義するのに必要なフォーマット (例、ID_Option1¥rValue1¥nID_Option2¥rValue2¥nID_Option3¥rValue3) は覚えにくく、間違いやすいものでした。

詳細は、次を参照してください:

- ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する
- [オプション リストの編集] ダイアログ ボックス

アドバンスト UI およびスイート/アドバンスト UI インストールでファイルまたはフォルダーの参照における実行時の機能強化

アドバンスト UI およびスイート/アドバンスト UI インストールでは、今回より、エンド ユーザーがファイルまたはフォルダーを参照できる機能が強化されています:

- ウィザード ページと 2 番目のウィンドウのコントロールで使用できる [新しいファイルを参照] アクションが追加されました。このアクションをボタンなどのコントロールの Click イベントに追加すると、エンド ユーザーがそのボタンをクリックしたときに参照ダイアログ ボックスが開き、新しいファイルとその場所を指定することができます。

  この種類のアクションをプロジェクト内のコントロールに追加するとき、ダイアログ ボックスのタイトル バーにテキストを指定できます。また、選択可能なファイル拡張子のフィルター オプションや、ファイル作成時に使用するデフォルトのファイル拡張子を指定することもできます。

- 既存の [フォルダーの参照] アクションが強化されました。このアクションが起動する [フォルダーの参照] ダイアログ ボックスは、対応するファイル システムを持つ場所のみに使用することができます。つまり、エンド ユーザーが [この PC] または [マイコンピューター] といった場所を選択した場合、このダイアログ ボックスの [OK] ボタンは無効化されます。以前、エンド ユーザーがこれらの場所を選択することが可能で、その場合に関連プロパティの値がクリアされました。

  この種類のアクションをプロジェクトに追加するとき、この [フォルダーの参照] ダイアログ ボックスに表示する支持を指定することができます。

- 既存の [ファイルの参照] アクションが強化され、[ファイルの参照] ダイアログ ボックスで選択可能なファイル拡張子のフィルター オプションを指定できるようになりました。

詳細については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

実行時に LicenseAgreement ダイアログで [印刷] の選択肢を提供できる機能

LicenseAgreement ダイアログの [印刷] ボタンの機能が強化されました。エンド ユーザーが [印刷] ボタンをクリックしたときにデフォルト プリンターに直接印刷する代わりに、今回より、印刷選択ダイアログ ボックスが開きます。

この強化は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、以前からこの機能の使用が可能です。

テキスト ファイルにテキストを挿入できる機能

[テキスト ファイルの変更] ビューでは、ターゲット システム上で実行時にテキスト ファイルに行う変更を構成することができます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。[テキスト ファイルの変更] ビューには、1 つ以上のファイルの最初または最後にテキストを挿入できる機能が追加されています。また今回より、指定したテキストの前後にテキストを挿入することもできます。このビューでテキスト ファイルの変更を構成するとき、既存テキストを置換するのか、既存テキストの前後にテキストを挿入するのか、またはファイルの最初または最後にテキストを挿入するのかを指定するための新しい "アクション" 設定を使用します。

以前、[テキスト ファイルの変更] ビューでは、実行時にテキスト ファイル内の既存テキストを置換するためのサポートが含まれていました。

この強化機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- テキスト ファイルの変更
- [テキスト ファイルの変更] ビュー

エンド ユーザーがメジャー アップグレードを適用する製品の以前のバージョンのインストール済みインスタンスを選択できる機能をサポート

基本の MSI プロジェクトにおける複数インスタンス サポートは、今回より、メジャー アップグレードのサポートを含みます。実行時、Setup.exe セットアップ ランチャーが表示するインスタンスの選択ダイアログを通して、エンドユーザーは製品の現在インストール済みのインスタンスのすべてをリストで参照できます。リストには、インストール済みの製品の異なるメジャーバージョンも含まれます。エンド ユーザーは、今回より、この強化されたインスタンス ダイアログを使用して新しいインスタンスのインストール、既存インスタンスの保持、またはインスタンスのメジャー アップグレードの適用を行うことができます。

以前、インスタンスの選択ダイアログには、実行中の複数インスタンス インストールと同じ製品コードを持つインスタンスのみがリストされました。

詳しくは、次を参照してください:

- 複数インスタンスのサポートを含むメジャー アップグレードの構成とビルド
- インスタンスのプロパティを設定します
- 製品のインスタンスを複数インストールするときの実行時の動作

追加プラットフォームで Microsoft App-V 5x パッケージを作成できる機能

InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5 フォーマットで仮想アプリケーションを作成するためのサポートが含まれています。このアシスタントの [パッケージ情報] ビューを使って、App-V 5 または 4.x のどちらをターゲットにするかを指定できます。InstallShield がサポートされている任意の Windows バージョンで、App-V 5x パッケージをビルドすることができます。以前は Windows 8 または Windows Server 2012 が必要でした。

詳細については、次を参照してください:

- App-V パッケージのコンポーネント
- App-V パッケージのビルド出力

基本の MSI インストールの [ファイルを開く] 実行時ダイアログの機能強化

InstallShield は、基本の MSI インストールのダイアログの 1 つから [ファイルを開く] ダイアログを起動するサポートを含みます。エンド ユーザーがダイアログの 1 つから [参照] ボタンをクリックすると、[ファイルを開く] ダイアログが起動します。[ファイルを開く] ダイアログを使って、エンド ユーザーはファイルを参照できます。今回よりオプションで、新しいプロパティ IS_BROWSE_INITIALDIR を使用して、実行時にこのダイアログに表示するデフォルト パスを指定できます。以前、ダイアログにはエンド ユーザーが最後に参照したパスが表示されました。

詳細については、「[ファイルを開く] ダイアログを起動する」を参照してください。

新しい FlexNet Connect 13.06 再配布可能ファイル

InstallShield は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 13.06 をサポートします。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 13.06 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。

InstallShield 2013 SP1 の新しい機能

InstallShield 2013 Service Pack 1 (SP1) には、Windows 8.1、Windows Server 2012 R2、および Visual Studio 2013 をサポートするための変更が含まれています。

Windows 8.1 および Windows Server 2012 R2 システムをターゲットにできる機能

InstallShield では、インストールに Windows 8.1 または Windows Server 2012 R2 が必要であることを指定できます。また、これらのオペレーティング システムに対する機能条件およびコンポーネント条件をビルドすることができます。

Windows 8.1 および Windows Server 2012 R2 にインストール可能な InstallShield 前提条件は、必要に応じて、これらのシステムにインストールされるように更新されています。以前これらのシステムでは、前提条件がデフォルトでは実行されませんでした。これは、次の InstallShield 前提条件に適用します:

- FSharp Redistributable Package 2.0
- JRE_SE 1.7.0_02 (x64)
- JRE_SE 1.7.0_02 (x86)
- Microsoft .NET Framework 3.0 OS Component
- Microsoft .NET Framework 3.5 SP1 (Windows 機能)
- Microsoft .NET Framework 4.5 Full
- Microsoft .NET Framework 4.5 Web
- Microsoft App-V 5.0 SP1 Desktop Client (x64)
- Microsoft App-V 5.0 SP1 Desktop Client (x86)
- Microsoft ReportViewer 2010
- Microsoft SQL CE 3.5 SP2
- Microsoft SQL Server 2005 Express SP3 (x86 & x64Wow)

- Microsoft SQL Server 2005 Express SP3 (x86)
- Microsoft SQL Server 2008 Express SP1 (x64)
- Microsoft SQL Server 2008 Express SP1 (x86 & x64Wow)
- Microsoft SQL Server 2008 Express SP1 (x86)
- Microsoft SQL Server 2008 Management Objects 10.00.2531 (IA64)
- Microsoft SQL Server 2008 Management Objects 10.00.2531 (x64)
- Microsoft SQL Server 2008 Management Objects 10.00.2531 (x86)
- Microsoft SQL Server 2008 Native Client 10.00.2531 (IA64)
- Microsoft SQL Server 2008 Native Client 10.00.2531 (x64)
- Microsoft SQL Server 2008 Native Client 10.00.2531 (x86)
- Microsoft SQL Server 2008 R2 Express RTM (x64)
- Microsoft SQL Server 2008 R2 Express RTM (x86 & x64Wow)
- Microsoft SQL Server 2008 R2 Express RTM (x86)
- Microsoft SQL Server 2008 R2 Express SP2 (x64)
- Microsoft SQL Server 2008 R2 Express SP2 (x86 & x64Wow)
- Microsoft SQL Server 2008 R2 Express SP2 (x86)
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1 (IA64)
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1 (x64)
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1 (x86)
- Microsoft SQL Server 2012 Express LocalDB RTM (x64)
- Microsoft SQL Server 2012 Express LocalDB RTM (x86)
- Microsoft SQL Server 2012 Express RTM (x64)
- Microsoft SQL Server 2012 Express RTM (x86 & x64Wow)
- Microsoft SQL Server 2012 Express RTM (x86)
- Microsoft SQL Server 2012 Native Client (x64)
- Microsoft SQL Server 2012 Native Client (x86)
- Microsoft SQL Server Compact 4.0 (x64)
- Microsoft SQL Server Compact 4.0 (x86)
- Microsoft SQL Server Native Client 9.00.4035 (IA64)
- Microsoft SQL Server Native Client 9.00.4035 (x64)
- Microsoft SQL Server Native Client 9.00.4035 (x86)
- Microsoft SQL Server System CLR Types 10.00.2531 (IA64)
- Microsoft SQL Server System CLR Types 10.00.2531 (x64)

- Microsoft SQL Server System CLR Types 10.00.2531 (x86)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538242(x64)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538242(x86)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538243(x64)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2538243(x86)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2008 SP1 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2010 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2010 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2010 RTM 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2467173 (x64)
- Microsoft Visual C++ 2010 RTM 再配布可能パッケージ MFC のセキュリティ更新プログラム KB2467173 (x86)
- Microsoft Visual C++ 2010 SP1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2010 SP1 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2012 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x86)
- Microsoft VSTO 2010 Runtime (x64)
- Microsoft VSTO 2010 Runtime

Windows 8.1 および Windows Server 2012 R2 用の InstallScript 言語の強化内容

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- SYSINFO.WINNT.bWin81 – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 8.1 または Windows Server 2012 R2 の場合、この値は TRUE です。
- ISOSL_WIN81 – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 8.1 または Windows Server 2012 R2 を実行中であることを示します。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

オートメーション インターフェイスの強化機能: Windows 8.1 と Windows Server 2012 R2 用の OSFilter プロパティの値

オートメーション インターフェイスで、ISWiComponent と ISWiRelease オブジェクトの OSFilter メンバーと共に、以下の定数を使用できます:

eosWin81 = &H8000000 (134217728)—これらの定数は Windows 8.1 および Windows Server 2012 R2 用です。

また、eosAll 定数の値が &7D100D0 (131137744) から &HFD100D0 (265355472) に変更されました。

OSFilter メンバーは、InstallScript、および InstallScript MSI プロジェクトで ISWiComponent オブジェクトに適用します。OSFilter メンバーは、InstallScript プロジェクトで ISWiReleaseISWiRelease オブジェクトに適用します。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

Microsoft Visual Studio 2013 のサポート

InstallShield は、Visual Studio 2013 をサポートします。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

新しい Microsoft SQL Server 2012 Express SP1 用の InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、いくつかの新しい SQL Server 関連の InstallShield 前提条件が含まれています:

- Microsoft SQL Server 2012 Express SP1 LocalDB (x64)
- Microsoft SQL Server 2012 Express SP1 LocalDB (x86)
- Microsoft SQL Server 2012 Express SP1 (x64)
- Microsoft SQL Server 2012 Express SP1 (x86 & x64Wow)
- Microsoft SQL Server 2012 Express SP1 (x86)
- Microsoft SQL Server 2012 Express SP1 Management Objects (x64)
- Microsoft SQL Server 2012 Express SP1 Management Objects (x86)
- Microsoft SQL Server 2012 Express SP1 System CLR Types (x64)
- Microsoft SQL Server 2012 Express SP1 System CLR Types (x86)

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

スイート/アドバンスト UI およびアドバンスト UI インストールで、ウィザード ページおよびウィンドウのコントロールに使用するカスタム テーマを定義できるサポート

スイート/アドバンスト UI およびアドバンスト UI プロジェクトでは、今回より、ウィザード インターフェイスのコントロールに使用するテーマを定義できます。コントロールのテーマは、コントロールの様々な部分の色 (テキストの色、背景の色、および境界線の色) や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトで、ユーザー インターフェイスのコントロールに使用するコントロールのテーマを指定できます。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

次の種類のコントロールで、コントロールのテーマを使用できます:

- ボタン (ナビゲーション ボタンを含む)

- チェック ボックス
- ラジオ ボタン
- コマンド リンク

プロジェクトにコントロールのテーマを追加するには、[ウィザード インターフェイス] ビューで [コントロールのテーマ] ノードを右クリックしてから [コントロールのテーマを追加] を選択します。必要に応じて、テーマの設定を構成します。

ユーザー インターフェイスのコントロールに、デフォルトで使用するテーマを指定するには、[ウィザード ページ] ノードが選択されているときに [ウィザード インターフェイス] ビューに表示される、新しい "デフォルト コントロールのテーマ" 設定で適切なテーマを選択します。デフォルトで、この設定は空白です。

特定のコントロールについて、コントロールのテーマをオーバーライドするには、そのコントロールの "コントロールのテーマ" 設定で適切なテーマを選択します。

詳細については、次を参照してください:

- スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする
- カスタム コントロール テーマをウィザード インターフェイスに定義する
- テーマをウィザード インターフェイスのコントロールに適用する

新しい、Visual Studio ソリューション フォルダーの定義済みパス変数

ハイレベルなベース ディレクトリを参照する、VSSolutionFolder と呼ばれる新しい定義済みパス変数をプロジェクトで使用できます。このサポートを使うと、InstallShield プロジェクトで Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルへのスタティック リンクを含めることができます。異なるマシン上のプロジェクトで作業を行う場合、VSSolutionFolder パス変数を使用するスタティック リンクは、姉妹プロジェクトのファイルへの正しいパスを参照することができます。

VSSolutionFolder パス変数は、InstallShield が Visual Studio ソリューション内で開かれたときに自動的に定義されます。また、MSBuild を使って InstallShieldプロジェクトを含むソリューションをビルドするときにも、自動的に定義されます。ただし、Visual Studio ソリューションなしで InstallShieldプロジェクトを開いた場合、VSSolutionFolder が自動的に定義されることはありません。たとえば、InstallShieldプロジェクトを、Visual Studio を開かずに InstallShieldインターフェイスで直接開いた場合、VSSolutionFolder は定義されません。同様に、コマンドライン ツール IsCmdBld.exe や、MSBuild で .isproj ファイルを使用する場合、VSSolutionFolder は定義されません。IsCmdBld.exe を使って InstallShield プロジェクトのリリースをビルドするには、-L コマンドライン パラメーターを使って、VSSolutionFolder の値を設定します。MSBuild を使う場合は、PathVariables パラメーターを使って VSSolutionFolder の値を設定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。

InstallShieldプロジェクトで VSSolutionFolder パス変数を含むパスを持つソース ファイルを含み、それを VSSolutionFolder パス変数がサポートされていない環境でビルドすると、次のようなビルド エラーが発生する可能性があります:

- -6103: ファイル <VSSolutionFolder>¥MyFile.exe が見つかりません
- -6271: ファイル <VSSolutionFolder>¥MyFile.exe が見つかりませんでした。このファイルの MsiFileHash テーブルをビルド中にエラーが発生しました。指定した場所にファイルが存在することを確認します。

この機能は、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、およびスイート/アドバンスト UI プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する
- 定義済みパス変数
- Microsoft ビルド エンジン (MSBuild)
- ISCmdBld.exe

複数インスタンス プロジェクトでインスタンスの製品コードを簡単に更新できる機能

複数インスタンス サポートを含むプロジェクトのメジャー アップグレードを作成している場合、各インスタンスの製品コードを更新する必要があります。InstallShield では、今回より、プロジェクトで定義されている特定インスタンスの製品コードを簡単に更新することができます。InstallShield ではまた、プロジェクト内の製品構成で定義されているすべてのインスタンスの製品コードを簡単に更新することもできます。

特定インスタンスの製品コードを更新するには、[リリース] ビューの製品構成で [複数インスタンス] タブで定義されているインスタンスの製品コード設定の値を右クリックすると使用できる、[新しい GUID を生成する] コマンドを使用します。

プロジェクトに含まれる製品構成で定義されている各インスタンスについて、その製品コードを更新するには、[リリース] ビューにある製品構成で、[複数インスタンス] タブにある [インスタンス] ノードを右クリックしたときに使用できる、新しい [各インスタンスの ProductCode を変更する] コマンドを使用します。

この機能は、基本の MSI プロジェクトで使用できます。

詳細については、「1 つ以上のインスタンスの製品コードを更新する」を参照してください。

InstallShield 2013 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

スイート/アドバンスト UI インストールの動作を拡張する実行時アクションを作成できる機能

スイート/アドバンスト UI インストールで、インストールに含めるパッケージの範囲を超えた様々な実行時タスクが必要な場合があります。たとえば、スイート/アドバンスト UI インストールで、以下の 1 つ以上の操作が必要な状況が考えられます。

- スイート/アドバンスト UI インストールのユーザー インターフェイスが表示される前にアプリケーションをインストールする。

  この状況が必要な例として、スイート/アドバンスト UI インストール内のパッケージの 1 つが Oracle データベースをインストールする SQL スクリプトを実行する時があります。このパッケージを実行する前に、スイート/アドバンスト UI インターフェイスで、エンド ユーザーがネットワーク上で使用可能なサーバーの一覧から適切なサーバーを選択できるウィザード ページを表示したい場合があります。この種類の UI サポートを使用するには、ターゲット システム上に ODBC ドライバーがインストールされていることが必要です。

- 特定の製品、テクノロジ、フォルダー、ファイル、レジストリ エントリ、その他のアイテムの存在の有無をターゲット システム上で検索する。

- スイート/アドバンスト UI インストール内のパッケージを実行する前後にターゲット システムを構成する。

スイート/アドバンスト UI インストールの機能を拡張して、前述のタスクその他を実行できるようにするには、スイート/アドバンスト UI プロジェクトに新しく追加された [イベント] ビューを使って、実行時に実行可能ファイルの実行、DLL 関数の呼び出し、PowerShell スクリプトの実行、またはスイート/アドバンスト UI プロパティの設定を行うアクションを作成できます。

プロジェクトにアクションを追加するとき、それを実行時に起動するタイミングをスケジュールできます。

- [イベント] ビューにある [イベント] エクスプローラーを使って、スイート/アドバンスト UI が管理する 1 つ以上のビルトイン イベントの実行中にアクションをスケジュールできます。ビュー内のこの領域には、各イベント中に発生するイベントとアクションが時系列で上から下にリストされます。
- [パッケージ] ビューの [イベント] 領域にある設定を使って、スイート/アドバンスト UI エンジンのインストール、削除、変更、またはパッケージの修復の前後に実行するアクションをスケジュールします。

アクションをスケジュールするとき、アクションを実行するかどうかを制御する条件ステートメントをビルドできます。新しく追加された、次の 2 種類の条件チェック以外にも、スイート/アドバンスト UI プロジェクトの別の領域で、同じ種類の条件チェックを利用できます:

- **パッケージの操作**—スイート/アドバンスト UI インストール内の特定のパケージの実行状態をチェックします (例、インストールまたは削除)。
- **機能の操作**—スイート/アドバンスト UI インストール内の特定の機能の実行状態をチェックします (例、インストールまたは削除)。

これらの条件の種類は、[イベント] ビューのイベント、または [パッケージ] ビューのパッケージに関連付けられたアクションでのみ使用可能です。

PowerShell アクションをサポートするには、ターゲット システム上に PowerShell 2.0 以降が必要です。

この機能は、InstallShield Premier Edition で提供されています。

詳細については、次を参照してください。

- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する
- スイート/アドバンスト UI インストールに含まれるアクションの .exe ファイルでの作業について
- スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について
- スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について
- アクションをスイート/アドバンスト UI プロジェクトに追加する
- スイート/アドバンスト UI アクションの設定を構成する
- スイート/アドバンスト UI アクションをスケジュールする
- スイート/アドバンスト UI インストールにおけるイベントの種類
- [イベント] ビュー
- [パッケージ] ビュー
- "パッケージの操作" 条件の設定
- "機能の操作" 条件の設定

スイート/アドバンスト UI インストール中に Windows の役割と機能を有効化できるサポート

スイート/アドバンスト UI インストール内の特定のパッケージが、ターゲット システム上で 1 つ以上の Windows の役割と機能が有効化されていることを必要とする場合、スイート/アドバンスト プロジェクトでパッケージを構成中に、[パッケージ] ビューに追加された新しい "Windows 機能" 設定を使って、この要件を指定できます。実行時、インストールされる対象のパッケージが無効化されている 1 つ以上の Windows の役割と機能を必要とする場合、スイート/アドバンスト UI インストールは、パッケージを起動する前にこれらの役割と機能を有効化します。

たとえば、スイート/アドバンスト UI プロジェクトで Windows のインターネット インフォメーション サービス機能が有効であることを必要とする IIS Web サイトをインストールパッケージがあります。同じプロジェクト内の別のパッケージでは、PowerShell 機能が有効であることが必要です。プロジェクトのこれらのパッケージの設定を構成するとき、必要な Windows 機能を指定します。実行時、パッケージがターゲット システム上でインストールの対象となっているが、必要な Windows 機能のいずれかが無効である場合、スイート/アドバンスト UI インストールは、起動前にこれらの無効化されている Windows 機能を有効にします。パッケージがインストールの対象ではない場合、必要な Windows 機能は有効化されません。

InstallShield には、いくつかの Windows 機能のビルトインサポートが搭載されています:

- インターネット インフォメーション サービス
- PowerShell
- .NET Framework 3.x

InstallShield ではまた、パッケージに必要な上記以外の Windows 役割と機能を指定することもできます。

この機能は、InstallShield Premier Edition で提供されています。

詳しくは、次を参照してください:

- スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する
- [パッケージ] ビュー

アドバンスト UI およびスイート/アドバンスト UI プロジェクト テンプレートを作成できる機能

InstallShield では、アドバンスト UI およびスイート/アドバンスト UI プロジェクトをテンプレートとして使用する機能がサポートされています。プロジェクト テンプレートは、新しいアドバンスト UI またはスイート/アドバンスト UI プロジェクトの作成を開始するときに使用できる、デフォルトの設定とデザイン要素を含みます。

任意のアドバンスト UI またはスイート アドバンスト UI プロジェクトをテンプレートに指定できます。プロジェクト ファイルとテンプレートファイルは、同じファイル拡張子 (.issuite) を持ちます。

.issuite プロジェクト ファイルがテンプレート ファイルであり、[新規プロジェクト] ダイアログ ボックスでこのテンプレートの選択を可能にするためには、[新規プロジェクト] ダイアログ ボックスの [すべてのタイプ] タブ内で右クリックして [新規テンプレートの追加] を選択します。テンプレートとして使用する .issuite ファイルを参照して、[すべてのタイプ] タブに新しく追加したテンプレートのアイコンが表示されます。

.issuite ファイルをテンプレートとして新しいアドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成するには、新しいプロジェクトを作成する際に [新規プロジェクト] ダイアログ ボックスの [すべてのタイプ] タブでテンプレートのアイコンを選択します。

手順については、次を参照してください:

- プロジェクト テンプレートの作成
- テンプレートを使用した新規プロジェクトの作成

多層アプリケーションをインストールするためのインストール作成用の新しいビルトイン テンプレート

InstallShield には、クラウド ベースの多層アプリケーション向けのスイート/アドバンスト UI インストールの作成に役立つ、新しい多層アプリケーション用のテンプレートが含まれています。エンド ユーザーが Web サイト、データベース、およびアプリケーションをインストールするインストール階層を簡単に管理できる、この種類のインストールを作成できます。エンド ユーザーは、スイート/アドバンスト UI インストールのウィザード ページを使って、インストールしたい層を選択するか、コマンドライン サポートを使って、適切な機能を選択できまうす。

新しい多層アプリケーション テンプレートには、「機能」として表示される 3 つの層があります。これらの機能をカスタマイズして、製品の層を含みます。機能は必要に応じて追加および削除が可能です。テンプレートにはまた、[リリース] ビューに 4つのリリース (3 つの定義済み機能のそれぞれにフラグが付いているテンプレートと、すべての機能を含むテンプレート) が含まれています。これによって、各層に分かれたインストールと、エンド ユーザーがすべての使用可能な層をインストールできるインストールを個別にビルドすることができます。

この機能は、InstallShield の Premier Edition で提供されています。

.NET Framework 3.5 SP1、Microsoft Visual C++ 2012、および SQL Server 2008 R2 Express SP2 用の新しい InstallShield 前提条件

InstallShield には、「アドバンスト UI」、「基本の MSI」、「InstallScript」、「InstallScript MSI」、および「スイート/アドバンスト UI」 プロジェクトに追加可能な次の前提条件が含まれています:

- Microsoft .NET Framework 3.5 SP1 (Windows 機能)
- Microsoft SQL Server 2008 R2 Express SP2 (x64)
- Microsoft SQL Server 2008 R2 Express SP2 (x86 & x64Wow)
- Microsoft SQL Server 2008 R2 Express SP2 (x86)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x64)
- Microsoft Visual C++ 2012 Update 1 再配布可能パッケージ (x86)

これらの前提条件は、サポートされているターゲット システムに様々なテクノロジをインストールします。

新しいアプリケーション仮想化適合性テスト

InstallShield では、製品を仮想化するための準備が整っているかどうかを判断するのに役立つ、新しい検証スイートが提供されています。これらのスイートで提供されている InstallShield 仮想化内部整合性検証ツール (ISVICE) を使って、Microsoft App-V、Microsoft Server App-V、VMware ThinApp、および Citrix XenApp の適合性を確認できます。検証スイートを使用することで、顧客に仮想バージョンの提供を考慮する場合に、製品をどのようにビルドするかについて、豊富な情報に基づいて意思決定を行うことができます。

InstallShield を使って、基本の MSI リリースが正しくビルドされるたびに、これらの検証スイートを実行するように構成する場合: [ツール] メニューから [オプション] を選択します。[検証] タブで、適切なチェック ボックスを選択します。

ビルド処理とは別に検証を行う場合: [ビルド] メニューから [検証] をポイントして、適切な新しいスイートを選択します。

仮想化スイートは、InstallShield Premier Edition で使用できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- InstallShield 仮想化整合性スイート
- プロジェクトの検証

Microsoft App-V 5 パッケージを作成するためのサポート

InstallShield の Microsoft App-V アシスタントには、Microsoft App-V 5 フォーマットで仮想アプリケーションを作成するためのサポートが含まれています。このアシスタントの [パッケージ情報] ビューを使って、App-V 5 または 4.x のどちらをターゲットにするかを指定できます。

App-V 5.x をターゲットにする場合、App-V 5.x パッケージを配布サーバーへ移動する前に、新しいバージョンの App-V ランチャー ツールを使って、それらをテストできます。このツールの新しいバージョンは、App-V パッケージの仮想環境内からコマンド プロンプト ウィンドウを起動することができます。そのため、App-V 5.x 以降の場合、Cmd.exe または Regedit.exe の診断ツール ショートカットを App-V パッケージに直接挿入する必要がなくなりました。

InstallShield が App-V パッケージをビルド時に保存するディレクトリは、ターゲットとする App-V のバージョンによって異なります。App-V 5 パッケージの場合、ディレクトリは ProductName で、App-VPackage という名前のフォルダーに配置されます。App-V 4.x パッケージの場合、InstallShield が ProductName フォルダーに製品バージョン番号を ProductName_v*N*（ここで *N* は製品バージョン番号）の形式で含みます。

InstallShield がサポートする任意の Windows バージョンで App-V 5.x パッケージの設定を構成できますが、InstallShield 内部から App-V 5.x パッケージをビルドするには Windows 8 または Windows Server 2012 が必要です。以前の Windows バージョンで App-V 5.x パッケージをビルドしようとすると、ビルド エラー -9424 が発生します。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

詳細については、次を参照してください:

- App-V パッケージのコンポーネント
- App-V パッケージのビルド出力

条件チェックの新しいプロパティの比較の種類 – スイート/アドバンスト UI および アドバンスト UI プロジェクトのインストール モードを判別するための新しいプロパティ

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクションの条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。条件チェックに新しく追加された [プロパティの比較] 型を使って、特定のビルトイン アドバンスト UI またはスイート/アドバンスト UI プロパティ、あるいは [プロパティ マネージャー] ビューで定義されたプロパティの値をチェックできます。

さらに、ISInstallMode という新しい読み取り専用プロパティも使用できます。このプロパティは、スイート/アドバンスト UI またはアドバンスト UI インストールを実行しているモードを示す値を格納します（初回インストール、メンテナンス/UI メンテナンスの選択、変更、削除、修復、またはステージングのみ）。このプロパティは、プロジェクトで構成する条件ステートメントで使用できます。

詳細については、次を参照してください。

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- "プロパティの比較" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

64 ビット専用 .msi および .msm パッケージをビルドできる機能 – 新しいビルド時のアーキテクチャの検証

Windows Server Core は、32 ビットの Windows-on-Windows (WOW64) サポートの無効化をサポートします。この構成がより一般化するのに伴い、64 ビットのアプリケーションが 32 ビットの機能に全く依存しないでインストールできるようにすることが必要となります。これを実現するため、InstallShield では 64 ビット専用 .msi パッケージのビルドが可能です。これらは、WOW64 機能を搭載していない 64 ビット版の Windows ベースのシステム上で実行します。また、ピュア 64 ビット マージ モジュールをビルドして、それを 64 ビット サポートを含む InstallShieldプロジェクトに含めることができます。インストールまたはマージ モジュール プロジェクトが 64 ビット専用環境をターゲットとし、任意のビルトイン InstallShield カスタム アクション DLL を必要とするサポートを含む場合、InstallShield はビルド時に、リリースにこれらの DLL の新しい 64 ビット バージョンを含めます。

今回より、InstallShield でリリースをビルドする時に使用できるアーキテクチャの検証が搭載されています。アーキテクチャの検証を使って、インストールがターゲット システムのアーキテクチャと一致しない製品ファイルをインストールしようとしたり、ランタイム バイナリを使用したりする問題を引き起こす可能性のある状況を検出することができます。

たとえば、エンド ユーザーが WOW64 サポートが搭載されていない 64 ビットのターゲット システムでインストールを実行する場合、アーキテクチャの検証を使って、インストールに含まれるすべての 32 ビット の製品ファイルおよび 32 ビットのカスタム アクション ファイルを識別できます。また、単一のプロジェクトに 64 ビットと 32 ビットの製品ファイル (または 64 ビットと32 ビットのカスタム アクション) が混在し、個別の 32 ビット .msi パッケージと 64 ビット .msi パッケージを生成する場合、アーキテクチャの検証を使って、32 ビット リリースに含まれる任意の 64 ビット製品またはカスタム アクション ファイルを識別することができます。これらのファイルを 32 ビット ターゲット システムでロードすることはできません。

使用するアーキテクチャの検証の種類を指定し、32 ビット専用パッケージ、または 64 ビット専用パッケージのどちらをビルドするのかを識別するには、[リリース] ビューの製品構成を選択したときに [全般] タブに表示される新しい "アーキテクチャの検証" 設定を使用します。2 種類の検証タイプ (厳密な検証をする、厳密な検証をしない) があります。

厳密な検証を行うオプションは、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガする場合があります。この種類の検証ではまた、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合にビルド警告をトリガする場合もあります。

デフォルトの検証の種類である、アーキテクチャの厳密な検証をしない場合、"テンプレートの概要" プロパティで指定されたアーキテクチャとリリースに含まれる 1 つ以上の製品ファイルまたはカスタム アクション ファイルのアーキテクチャが一致しないとき、ビルド エラーまたは警告がトリガされません。

オートメーション インターフェイスには、新しいアーキテクチャの検証サポートが含まれています。ISWiProductConfig オブジェクトに含まれている、新しい読み書きプロパティ ArchitectureValidation を使って、製品構成で使用するアーキテクチャの検証の種類を指定できます。

この機能は、基本の MSI およびマージ モジュール プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- ビルドに適切なアーキテクチャ検証の種類を選択する
- 64 ビット オペレーティング システムをターゲットにする
- "テンプレート概要" プロパティを使用する
- ISWiProductConfig オブジェクト

Windows 8 ロゴ プログラムの検証

InstallShield では、新しい 2 つの検証スイート (InstallShield 検証スイート – Windows 8 および InstallShield マージ モジュール検証スイート - Windows 8) が提供されています。これらの検証スイートには、いくつかの新しい InstallShield 内部これらの検証スイートには、新しい ISICE (内部整合性検証ツール) ISICE21 から ISICE26 までが含まれています。いれらのスイートは、Windows Installer ベースのインストール、またはマージ モジュールが Windows 8 デスクトップ アプリケーション認証プログラムに対応するインストール要件を満たすかどうかの検証に役立ちます。Windows 8 ロゴを使用する場合、アプリケーションのインストールがプログラムの要件を満たしている必要があります。これらのスイートはまた、インストールまたは マージ モジュールが Windows Server 2012 システム上で動作することを検証するのにも役立ちます。

InstallShield を使って、リリースが正しくビルドされるたびに、これらの検証スイートを実行するように構成する場合: [ツール] メニューから [オプション] を選択します。[検証] タブで、適切なチェック ボックスを選択します。

ビルド処理とは別に検証を行う場合: [ビルド] メニューから [検証] をポイントして、適切な新しいスイートを選択します。

詳細は、次を参照してください:

- プロジェクトの検証
- ISICE
- Windows ロゴ プログラムの要件

最新プラットフォームにおけるショートカットの構成に関する強化内容 – ショートカット作成における InstallScript 言語の強化

InstallShield では、プロジェクトのショートカットを構成するための強化が提供されています。

Windows 8 スタート画面にショートカットをピン留めしないように防ぐサポート

InstallShield では、Windows 8 ターゲット システム上で、インストールに含まれる各ショートカットをデフォルトでスタート画面にピン留めするかどうかを指定できます。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。ショートカットのピン留めを無効化した場合でも、システム上のすべてのショートカットを含むアプリケーション一覧には、そのショートカットが表示されます。

ショートカットをスタート画面にピン留めするのを防ぐには、[ショートカット] ビューでショートカットの新しい "Windows 8 スタート画面にピン留めする" 設定を使用します。

この機能は、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

オートメーション インターフェイスでは、Windows 8 スタート画面にショートカットをピン留めするかどうかを指定できるサポートを提供します。ISWiShortcut オブジェクトには、Windows 8 ターゲット システム上で、デフォルトでスタ—ト画面にショートカットをピン留めするように構成されているかどうかを示すブール型 EnableWin8StartPin プロパティが含まれています。

詳細は、次を参照してください:

- Windows 8 スタート画面への初回のショートカットのピン留めを抑制する
- [ショートカット] ビュー
- ISWiShortcut オブジェクト

エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めするのを防ぐサポート

インストールに含まれる各ショートカットについて、エンド ユーザーがショートカットをタスクバーまたは [スタート] メニューにピン留めできるようにするかどうかを指定できます。この種類のピン留めを防ぐ場合、インストールはショートカットをタスクバーおよび [スタート] メニューにピン留めするコンテキスト メニュー コマンドを非表示にするプロパティを設定します。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを防ぎたい場合があります。

エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできないように防ぐサポートは、使用するプロジェクトの種類によって異なります。

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクトの場合: ショートカットのタスクバーおよび [スタート] メニューへのピン留めを防ぐには、[ショートカット] ビューでショートカットの "シェル プロパティ" 設定を使用します。この設定には、選択されたショートカットのピン留めを無効化できる、新しい [ピン留めを防ぐ] オプションが追加されました。Windows Installer 5 で、このショートカットの設定がサポートされています。以前、同じ動作を構成するためには、適切なプロパティ名と値を手入力する必要がありました。

InstallScript プロジェクトの場合: ショートカットのタスクバーおよび [スタート] メニューへのピン留めを防ぐには、[ショートカット] ビューに新しく追加された "ピン留めを防ぐ" 設定を使用します。Windows 7 以降で、このショートカットの設定がサポートされています。以前、InstallScript プロジェクトで、この機能の構成はサポートされていませんでした。

オートメーション インターフェイスでは、エンド ユーザーが製品をインストールした後、ショートカットをタスクバーおよび [スタート] メニューにピン留めするためのコンテキスト メニューを表示するかどうかを指定できるサポートが提供されています。ISWiShortcut オブジェクトには、ショートカットのピン留めを行うコンテキスト メニュー コマンドが非表示となるように構成されているかどうかを示す、読み取り専用のブール型 PreventPinning プロパティが含まれています。

詳細については、次を参照してください。

- エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する
- [ショートカット] ビュー
- ISWiShortcut オブジェクト

[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐサポート

オプションで InstallShield では、エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐことができます。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。インストールの一部であるツールまたは従属的な製品のショートカットのプロパティを設定したい場合があります。

強調表示の動作は、使用するプロジェクトの種類によって異なります。

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合: ショートカットの強調表示を防ぐには、[ショートカット] ビューでショートカットの "シェル プロパティ" 設定を使用します。この設定には、選択されたショートカットの強調表示を無効化できる、新しい [新規として強調表示しない] オプションが追加されました。Windows Installer 5 で、このショートカットの設定がサポートされています。以前、同じ動作を構成するためには、適切なプロパティ名と値を手入力する必要がありました。

InstallScript プロジェクトの場合: ショートカットの強調表示を防ぐには、[ショートカット] ビューで新しい "新規として強調表示しない" 設定を使います。Windows 7 以降で、このショートカットの設定がサポートされています。以前、InstallScript プロジェクトで、この機能の構成はサポートされていませんでした。

オートメーション インターフェイスには、エンド ユーザーが製品をインストールした後に、[スタート] メニューで新規項アイテムとして強調表示するかどうかを指定できるサポートが含まれています。ISWiShortcut オブジェクトには、この強調表示を阻止するかどうかを示す新しい読み取り専用のブール型 DoNotHighlightAsNew プロパティが含まれています。

詳しくは、次を参照してください:

- [スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ
- [ショートカット] ビュー
- ISWiShortcut オブジェクト

ショートカットの追加 Windows シェル プロパティのビルトイン サポート

InstallShield では、実行時に Windows シェルによる設定が必要な 1 つ以上の追加ショートカット プロパティを指定できます。シェルが設定できるプロパティは、Windows SDK に含まれている propkey.h で定義されています。

ショートカットの追加プロパティを構成するには、[ショートカット] ビューでショートカットの "シェル プロパティ" 設定を使用します。この設定には、選択されたショートカットに追加プロパティとそれに対応する値を指定できる、新しい [カスタム プロパティ] オプションが追加されています。

Windows Installer 5 では、シェル プロパティの構成がサポートされています。

オートメーション インターフェイスには、ショートカットにカスタム Windows シェル プロパティ用の ISWiShellProperty オブジェクトが含まれています。さらに、ISWiShortcut オブジェクトには、ショートカットにシェル プロパティを追加する (AddShellProperty)メソッド、シェル プロパティを削除する (DeleteShellProperty) メソッド 、およびシェル プロパティのコレクションを取得する (ISWiShellProperties) が含まれています。

このサポートは、基本の MSI、DIM、 InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- カスタム アクション シェルのプロパティを設定する
- [ショートカット] ビュー
- ISWiShellProperty オブジェクト
- AddShellProperty メソッド
- DeleteShellProperty メソッド
- ISWiShellProperties コレクション
- ISWiShortcut オブジェクト

ショートカット プロパティの設定、その他のショートカット機能における InstallScript 言語の強化

SetShortcutProperty という名前の新しい InstallScript 関数が追加されました。InstallScript コードでこの関数を使って、ショートカットの Windows シェル プロパティを設定します。この関数は、エンド ユーザーが Windows 8 スタート画面にショートカットをピン留めすることを防ぎ、Windows 7 以降のシステムでは、タスクバーまたは [スタート] メニューにショートカットをピン留めすることを防ぎ、Windows 7 以降のシステム上では [スタート] メ

ニューのショートカットが新しくインストールされたアイテムとして強調表示されないように防ぎます。またこの関数を使って、Windows SDK に含まれている propkey.h で定義されている追加シェル プロパティを指定することもできます。

InstallShield キャビネット & ログ ファイル ビューアは、前述のシェル プロパティの状態を反映して更新されています。

現在のオペレーティング システムに関する用語を反映して、既存のショートカット関数の名前が変更されています。古い関数名も、InstallShield の以前のバージョンと同様にコンパイルおよびビルドすることが可能です。新しい InstallScript 関数は、次の通りです:

- **CreateShortcut**—この関数は AddFolderIcon に優先します。**CreateShortcut** は、**AddFolderIcon** と同じパラメーターを受け付けますが、nFlag パラメーターの以下のオプションを追加または置換します:
  - CS_OPTION_FLAG_REPLACE_EXISTING
  - CS_OPTION_FLAG_RUN_MAXIMIZED
  - CS_OPTION_FLAG_RUN_MINIMIZED
  - CS_OPTION_FLAG_PREVENT_PINNING
  - CS_OPTION_FLAG_NO_NEW_INSTALL_HIGHLIGHT
  - CS_OPTION_FLAG_NO_STARTSCREEN_PIN
- **CreateShortcutFolder** — この関数は **CreateProgramFolder** に優先します。どちらの関数も、同じパラメーターを使用します。
- **DeleteShortcut** — この関数は **DeleteFolderIcon** に優先します。どちらの関数も、同じパラメーターを使用します。
- **DeleteShortcutFolder** — この関数は **DeleteProgramFolder** に優先します。どちらの関数も、同じパラメーターを使用します。
- **GetShortcutInfo** — この関数は **QueryProgItem** に優先します。どちらの関数も、同じパラメーターを使用します。
- **ReplaceShortcut** — この関数は **ReplaceFolderIcon** に優先します。**ReplaceShortcut** は、**ReplaceFolderIcon** と同じパラメーターを受け付けますが、nFlag パラメーターに **CreateShortcut** と同じオプションを追加または置換します。

詳細については、次を参照してください。

- CreateShortcut
- CreateShortcut の例
- CreateShortcutFolder
- CreateShortcutFolder の例
- DeleteShortcut
- DeleteShortcut の例
- DeleteShortcutFolder
- DeleteShortcutFolder の例
- GetShortcutInfo

- GetShortcutInfo の例
- ReplaceShortcut
- ReplaceShortcut の例
- SetShortcutProperty
- SetShortcutProperty の例

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで、ウィザード インターフェイスのカスタマイズにおけるユーザビリティーの向上

混乱しやすい構文要件の削除

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューにある様々な UI 設定が改良され、アドバンスト UI およびスイート/アドバンスト UI インストールのウィザード インターフェイスのカスタマイズがより簡単になりました。たとえば、[ウィザード インターフェイス] ビューでウィザード ページまたはウィザード コントロールを選択したときに表示される次の設定が改良されています:

- **表示**—この設定を使って、選択されたページまたはコントロールを表示するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。
- **有効**—この設定を使って、選択されたコントロールを有効化するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。
- **検証**—この設定を使って、エンド ユーザーによるコントロールへの応答を評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の検証ステートメントを指定できます。たとえば、この設定を使ってエンド ユーザーがテキスト ボックス コントロールにシリアル番号を入力するのに使用するフォーマットを指定できます。
- **アクション**—この設定の各インスタンスは、選択されたコントロールに適切な特定の種類のトリガを反映した、より相応しい名前に変更されています。たとえば、リスト ボックス コントロールの "アクション" 設定の名前は、"選択範囲の変更" に変更されています。ボタンの "アクション" 設定の名前は、"クリック" に変更されています。名前が変更されている設定を使って、エンド ユーザーが選択されたコントロールを使ったときにトリガする 1 つ以上のアクションを選択できます。

これらの設定には、今回より、アドバンスト UI およびスイート/アドバンスト UI プロジェクトのその他の領域で使用できるドロップダウン リストとボタンと同じような要素が含まれていて、素早く条件ステートメントをビルドすることができます。これらの設定を使って、UI の様々な領域で状態の表示/非表示、状態の有効化/無効化、検証、およびアクションの応答を素早く構成して、適切な場合、その動作に対する条件ステートメントをビルドできます。以前、これらの設定は、覚え難い構文を使ったステートメントを手作業で入力しなくてはならない編集フィールドでした。

詳細については、次を参照してください。

- ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する
- ウィザード インターフェイス内の要素のアクションを構成する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に単一の背景を使用できるビルトイン サポート

[ウィザード インターフェイス] ビューの [ウィザード ページ] ノードに、新しい "完全なウィザード背景" 設定が追加されました。この設定を使って、ウィザード ページのヘッダー、本文、およびナビゲーション領域のすべてにわたって使用する単一の背景を指定できます。以前は、InstallShield で背景を指定するためのビルトインされた方法は、"デフォルトの本文背景"、"ヘッダー背景"、および "ナビゲーション背景" 設定で個別の背景スタイルを使用する方法しかありませんでした。ウィザード インターフェイスの 3 つの領域すべてで単一の背景を使用したい場合、InstallTalk ブログ記事に記載されているように、テキスト エディターで InstallShield プロジェクト ファイル (.issuite) を変更する処理を行う必要がありました。

この新しい "完全なウィザード背景" 設定で背景を選択する場合、その他の背景設定 ("デフォルトの本文背景"、"ヘッダー背景"、および "ナビゲーション背景") からすべての値を削除しなくてはなりません。

詳細については、「ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する」を参照してください。

すべてのナビゲーション ボタンにグローバル テキスト スタイルを指定できる機能

[ウィザード インターフェイス] ビューの [ウィザード ページ] ノードに、新しい "ナビゲーションのテキスト スタイル" 設定が追加されました。この設定を使って、すべてのナビゲーション ボタン上のテキストに使用する単一のテキスト スタイルを選択できます。必要な場合、個別のナビゲーション ボタンのグローバル テキスト スタイルをオーバーライドできます。

論理的に編成された設定グリッド

[ウィザード インターフェイス] ビューでは、ウィザード インターフェイスをカスタマイズするのに必要な設定を見つけやすくするため、ウィザード ページのグリッドの設定、2 番目のウィンドウ、およびウィザード コントロールが再編成されています。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでフォント セットを定義し、オプションで言語固有の選択を行う機能のサポート

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、フォント セットという、新しい種類のウィザード インターフェイス スタイルがサポートされています。フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。

フォント セットは、テキスト スタイルと組み合わせて使用します。テキスト スタイル (テキストの色や配置などのテキスト属性を定義するスタイル) は、今回より、フォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。

たとえば、ウィザード インターフェイスのテキスト本文にフォント情報を指定する時、BodyFonts と呼ばれるフォント セットを使用して、プロジェクトでサポートされている各言語に異なるフォントを設定できます。その後、様々なテキスト スタイル (本文、ヘッダー、および BodyBold) に BodyFonts フォントセットを選択できますが、必要に応じて小さいサイズや太字を特別にオーバーライドすることができます。これによって、可能なフォントの大きなセットを一度指定しておいて、特定のウィザード インターフェイスが使用するテキスト サイズや色などの属性を変更することができます。

詳しくは、次を参照してください:

- スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする
- カスタム スタイルをウィザード インターフェイスに定義する

強化機能

InstallShield には、以下のような強化機能が搭載されています。

新しい [サービス] ビュー

新しい [サービス] ビュー ([ビュー] リストの [システム構成] ノードの下) では、実行時にインストール、開始、構成、停止、または削除する Windows サービスについて必要な情報を指定できます。

この新しい [サービス] ビューは、新しいユーザーがサービスを構成しやすいように追加されました。このビューは次の既存領域と同じ機能を提供します:

- (インストール プロジェクトの) [セットアップのデザイン] ビューでコンポーネントの [詳細設定] ノードの下にある [サービス] 領域
- [コンポーネント] ビューでコンポーネントの [詳細設定] ノードの下にある [サービス] 領域

3 つのビュー ([サービス] ビュー、[セットアップのデザイン] ビュー、または [コンポーネント] ビュー) のいずれかでサービス情報を構成すると、対応するビューも自動的に更新されます。

[サービス] ビューは、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- Windows サービスのインストール、制御、および構成
- [サービス] ビュー

スイート/アドバンスト UI およびアドバンスト UI ウィザード インターフェイスの新しいウィザード フォーマット

スイート/アドバンスト UI およびアドバンスト UI インストールのウィザード インターフェイスで使用できる新しい Glass ウィザード フォーマットこの新しいフォーマットは、既存の Aero フォーマットに似ています。どちらのフォーマットでも、インストールのユーザーが (プロジェクトで定義されたブラシ スタイルの代わりに) ユーザー選択のシステム色を使ってウィザード インターフェイスのキャプション バー、ヘッダー、およびナビゲーション領域を表示できます。Windows Vista から使用できる透明感サポートが有効なターゲット システム上では、ウィザード インターフェイスのこれらの同じ領域にガラス効果 (半透明) を使用することもできます。

一部の状況下で、インストールは Glass または Aero フォーマットの代わりに Wizard 97 フォーマットを使用します:

- Windows 8 および Windows Server 2012 システムは、透明感をサポートしません。これらのシステムでは、Glass フォーマットを持つウィザード インターフェイスは Wizard 97 フォーマットを使用します。ただし、Aero フォーマットを持つウィザード インターフェイスは (ユーザー選択のシステム色を含む) Aero を使用しますが、ガラス効果はありません。
- Windows 7、Windows Vista、Windows Server 2008 R2、および Windows Server 2008 では、エンド ユーザーが有効化/無効化できる透明感サポートが使用できます。これらのシステムでは、透明感が無効化されている場合、Glass フォーマットを使ったウィザード インターフェイスと Aero フォーマットを使ったウィザード インターフェイスでは、Wizard 97 フォーマットが使用されます。また、これらのウィザード インターフェイスでは、ターゲット システム上のデスクトップ コンポジション機能が無効化されている場合、Wizard 97 フォーマットが使用されます。
- Windows XP および Windows Server 2003 上では、Glass フォーマットのウィザード インターフェイスと Aero フォーマットのウィザード インターフェイスは Wizard 97 フォーマットを使用します。

フォーマットを構成するには、[ウィザード ページ] ノードを選択したときに [ウィザード インターフェイス] ビューに表示される "ウィザード フォーマット" 設定を使用します。すべての新しいスイート/アドバンスト UI およびアドバンスト UI プロジェクトでは、Glass フォーマットがデフォルト オプションです。

詳細については、「ウィザード インターフェイスのフォーマットを選択する」を参照してください。

オートメーション インターフェイスの強化内容: マイナー アップグレードおよびスモール アップデートで使用できる新しい OnUpgrade プロパティ

オートメーション インターフェイスで、新しい読み書きプロパティ OnUpgrade が ISWiProject オブジェクトに追加されました。このプロパティをサポートされている値の 1 つに設定して、マイナー アップグレードまたはスモール アップデートをインストールする前にプロンプトを表示するかどうかを指定できます:

このプロパティは、[アップグレード] ビューの [共通] タブにある "スモール/マイナー アップグレード" 設定オプションに対応します。

詳細については、「ISWiProject オブジェクト」を参照してください。

InstallShield 2012 Spring SP1 の新しい機能

InstallShield 2012 Spring Service Pack 1 (SP1) には、Windows 8、Windows Server 2012、および Visual Studio 2012 の最終版をサポートするための変更が含まれています。また、その他の変更も含まれています。

重要 *• InstallShield 2012 SP1 で InstallShield 2012 Spring アドバンスト UI または スイート/アドバンスト UI プロジェクト (.issuite) を開くとき、プロジェクトを InstallShield 2012 Spring SP1 にアップグレードすることを許可する必要があります。InstallShield 2012 SP1 アドバンスト UI およびスイート/アドバンスト UI プロジェクトには、InstallShield 2012 Spring でこれらのプロジェクト タイプでは使用できなかった機能のサポートが提供されていますが、このサポートはアップグレード中に追加する必要があります。InstallShield 2012 Spring SP1 アドバンスト UI またはスイート/アドバンスト UI プロジェクトを以前のバージョンの InstallShield (SP1 を適用する前の InstallShield 2012 Spring を含む) で開くことはできませんので、ご注意ください。このため、複数のユーザーが InstallShield プロジェクトを開いて編集する場合、すべてのユーザーが同時に SP1 パッチを適用するようにしてください。*

InstallShield 2012 Spring アドバンスト UI またはスイート/アドバンスト UI プロジェクトを InstallShield Spring SP1、InstallShield 2012 Spring SP1 で開くと、そのプロジェクトを新しいバージョンに変換するかどうかをたずねるメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前にプロジェクトのバックアップ コピーが作成されます。

サイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加するためのサポート

InstallShield では、今回よりサイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加できます。その他の種類のパッケージをプロジェクトに追加する処理と同様に、このパッケージ タイプをスイート アドバンスト UI の [パッケージ] ビューを使って追加できます。

アプリのサイドローディングとは、Windows Store を介さずにアプリケーションをインストールするプロセスです。この種類のアプリケーションは、企業環境で配布されることがあります。Windows 8 および Windows Server 2012 では、サイドローディング アプリケーションがサポートされています。

このサポートは、InstallShield の Premier Edition で提供されています。

サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。

詳細については、次を参照してください。

- サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する
- [パッケージ] ビュー

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける、条件チェックに追加された Windows アプリ パッケージ タイプ

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。新しい Windows アプリ パッケージ タイプの条件チェックを使うと、ターゲット システムで特定 Windows アプリ パッケージの存在をチェックできます。条件は特定のアプリケーション名をチェックし、製品バージョンなどのその他の情報もチェックできます。

詳細については、次を参照してください。

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- 「Windows アプリ パッケージの条件" の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

Visual Studio 2012、.NET Framework 4.5、および Visual C++ 2012 のサポート

InstallShield Visual Studio 2012 の最終版をサポートするための変更が含まれており、このバージョンの Visual Studio インターフェイス内部でインストールおよび製品の開発が可能です。

さらに、InstallShield には 2 つの .NET Framework 用 InstallShield 前提条件および新しい 2 つの Visual C++ 用 InstallShield 前提条件が提供されています:

- Microsoft .NET Framework 4.5 Full
- Microsoft .NET Framework 4.5 Web
- Microsoft Visual C++ 2012 再配布可能パッケージ (x86)
- Microsoft Visual C++ 2012 再配布可能パッケージ (x64)

これらの前提条件は、アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクトに追加できます。

.NET Framework の Web 前提条件には、インターネット接続が必要です。この前提条件は、必要に応じて、必須の再配布可能ファイルをダウンロードします。完全な前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。

追加の変更

InstallShield 2012 Spring SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2012 Spring の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows 8 および Windows Server 2012 システムをターゲットできる機能

InstallShield では、インストールに Windows 8 または Windows Server 8 が必要であることを指定できます。また、これらのオペレーティング システムに対する機能条件およびコンポーネント条件をビルドすることができます。

Windows 8 および Windows Server 2012 にインストール可能な InstallShield 前提条件は、必要に応じて、これらのシステムにインストールされるように更新されています。以前これらのシステムでは、前提条件がデフォルトでは実行されませんでした。これは、次の InstallShield 前提条件に適用します:

- FSharp Redistributable Package 2.0
- Microsoft ReportViewer 2010
- Microsoft SQL CE 3.5 SP2
- Microsoft SQL Server 2005 Express SP3
- Microsoft SQL Server 2008 Express SP1
- Microsoft SQL Server 2008 Management Objects 10.00.2531
- Microsoft SQL Server 2008 Native Client 10.00.2531
- Microsoft SQL Server 2008 R2 Express RTM
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1
- Microsoft SQL Server Native Client 9.00.4035
- Microsoft SQL Server System CLR 型 10.00.2531
- Microsoft Visual C++ 2005 SP1 Redistributable MFC セキュリティ更新プログラム KB2538242
- Microsoft Visual C++ 2005 SP1 Redistributable Package
- Microsoft Visual C++ 2008 SP1 Redistributable MFC セキュリティ更新プログラム KB2538243
- Microsoft Visual C++ 2008 SP1 Redistributable Package
- Microsoft Visual C++ 2010 再配布可能パッケージ
- Microsoft Visual C++ 2010 RTM Redistributable MFC セキュリティ更新プログラム KB2467173
- Microsoft Visual C++ 2010 SP1 Redistributable Package
- Microsoft VSTO 2010 Runtime

InstallShield Profesioanl Edition で最新のカスタマイズ可能なエンド ユーザー インターフェイスをサポート

新しいアドバンスト UI プロジェクト タイプでは、Windows Installer パッケージまたは InstallScript パッケージ用に、デザインが一新された最新のウィザード ページを持つ新しいエンドユーザー インターフェイスを作成することができます。InstallShield Professional Edition で提供されているこの新しいプロジェクト タイプでは、以前 InstallShield Premier Edition で紹介されたスイートプロジェクト タイプ (現在はスイート/アドバンスト UI プロジェクト タイプと呼ばれています) と同じテクノロジーが使用されています。

新しいアドバンスト UI プロジェクト タイプでは、アドバンスト UI インストールに含めることができるカスタマイズ可能なビルトイン ウィザード ページが使用されています。このプロジェクト タイプで使用できるウィザード ページ エディターでは、必要に応じてページを追加、シーケンス、または削除することができます。また、様々な異なる種類のコントロールを追加または削除することで任意のページのレイアウトを編集することもできます。以前 Premier Edition でのみ使用可能だったすべての新しい UI 機能は、アドバンスト UI プロジェクトで使用可能になっています。

スイート/アドバンスト UI プロジェクト同様、アドバンスト UI プロジェクトでは、ターゲット システムでパッケージを起動する次世代セットアップ ランチャー (Setup.exe) を使用しています。また、スイート/アドバンスト UI プロジェクト同様、アドバンスト UI プロジェクトでは、アドバンスト UI インストールに含まれるパッケージのランタイム ソースの場所を指定できる柔軟なオプションが提供されています。以下は、選択可能な場所です:

- Web 上で、必要に応じて Setup.exe によるダウンロードが可能
- Setup.exe に埋め込み、必要に応じてターゲット システムに展開
- 非圧縮ファイルでスイートのアドバンスト UI ソース メディアに格納

エンド ユーザーはコンパクトなアドバンスト UI Setup.exe ファイルを短時間でダウンロードすることができ、Setup.exe ファイルが必要に応じて、Windows Installer ベースまたは InstallScript のパッケージをダウンロードおよび起動します。

アドバンスト UI プロジェクト タイプでは、プライマリ パッケージとして含めることができるパッケージは、.msi パッケージ、.msp パッケージ、または InstallScript パッケージの中からいずれか 1 つのみです。InstallShield Premier Edition で提供されているスイート/アドバンスト UI プロジェクト タイプでは、.exe パッケージを含む複数のプライマリ インストールを単一インストールにパッケージ化できる機能が提供されています。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI インストールの作成
- アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い
- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- ウィザード インターフェイスを使って作業する

Microsoft Windows Azure SQL データベース サポート

InstallShield では、Microsoft Windows Azure SQL データベース サーバーで SQL スクリプトを実行できるサポートが提供されています。さらに、InstallShield の [SQL スクリプト] ビューには、製品がサポートするターゲットのデータベース サーバーを指定するときに選択可能な定義済みのデータベース サーバーのリストに Microsoft Windows Azure が含まれています。

インストールが Windows Azure をターゲットとする場合、エンド ユーザーがデータベース カタログの参照を選択したときに表示される SQLBrowse ランタイム ダイアログに、SQL データベース サーバーで指定されたカタログの一覧が含まれています。

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

詳細は、次を参照してください:

- Microsoft SQL Azure Database サーバーへの接続と SQL スクリプトの実行
- SQL サポートの構成
- [SQL スクリプト] ビュー

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

Microsoft Visual Studio 2012 のベータ サポート

InstallShield には Visual Studio 2012 ベータのサポートが含まれています。このバージョンの Visual Studio 内部から InstallShield プロジェクトを作成できます。

Microsoft .NET Framework 4.5 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい 2 つの InstallShield 前提条件が含まれています:

- Microsoft .NET Framework 4.5 Full
- Microsoft .NET Framework 4.5 Web

これらの InstallShield 前提条件は、サポートされているターゲット システムに、ベータ バージョンの .NET Framework 4.5 をインストールします。

Web 前提条件には、インターネットへの接続が必要です。この前提条件は、必要に応じて、必須の再配布可能ファイルをダウンロードします。完全な前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。

Microsoft SQL Server 2012 サポート

InstallShield に、SQL Server 2012 Database 上で SQL スクリプトを実行できるサポートが追加されました。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2012 が含まれています。

インストールで SQL Server 2012 をターゲットするとき、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、SQL Server 2012、SQL Server 2012 Express、および SQL Server 2012 Express LocalDB のインスタンスが表示できるようになりました。また、エンドユーザーがデータベース カタログを参照する選択をしたときに表示される SQLBrowse ランタイム ダイアログで、指定された SQL Server 2012 サーバー上のカタログが表示されるようになりました。

詳細は、次を参照してください:

- SQL Server Express LocalDB のインスタンスに接続するときの要件
- SQL サポートの構成
- [SQL スクリプト] ビュー

このサポートは、基本の MSI、DIM、InstallScript、および InstallScript MSI プロジェクト タイプで利用できます。

Microsoft SQL Server 2012 の前提条件

InstallShield に、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加できる SQL Server 2012 関連の新しい InstallShield 前提条件がいくつか追加されました:

- Microsoft SQL Server 2012 Express
- Microsoft SQL Server 2012 Express LocalDB
- Microsoft SQL Server 2012 Native Client

InstallShield には、Microsoft SQL Server 2012 Express の依存関係である Microsoft .NET Framework 3.5 SP1 Update KB956250 をインストールする InstallShield 前提条件も含まれています。

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

App-V 4.6 SP1、SQL Server Compact 4.0 および JRE SE 1.7 の新しい InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、新しい InstallShield 前提条件が含まれています:

- Java Runtime Environment Second Edition (JRE SE) 1.7
- Microsoft App-V 4.6 SP1 Desktop Client (これらの InstallShield 前提条件の再配布可能ファイル (32 ビット バージョンおよび 64 ビット バージョン) は、InstallShield 内部からはダウンロードできません。これらは、Microsoft から取得する必要があります。Microsoft からそれらのファイルを入手した後、InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。)
- SQL Server Compact 4.0

これらの InstallShield 前提条件は、サポートされているターゲット システムにテクノロジーがインストールされます。

Configuring System Center 2012 Configuration Manager のアプリケーション データの構成のサポート

配置メタデータを正確に識別することは、アプリケーションを System Center 2012 Configuration Manager アプリケーション モデルに移行する上で必須です。InstallShield の [一般情報] ビューには、ソフトウェア ID タグを使ってアプリケーション モデル メタデータの一部を指定するための、いくつかの新しい設定が含まれています。AdminStudio のユーザーが AdminStudio のアプリケーション カタログにパッケージをインポートしたとき、パッケージの要素が解析されて、検出方法、依存関係、要件などの配置データ、およびソフトウェア識別タグ内の情報が確認されます。AdminStudio では、この情報を Microsoft System Center 2012 Configuration Manager に公開する前に、確認およびテスト用として、ユーザーに提供しています。

この機能は、基本の MSI プロジェクトで使用できます。

詳細については、「製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める」を参照してください。

PowerShell カスタム アクションのサポート

Windows PowerShell は、構成タスクのオートメーション化を可能にする .NET Framework ベースのコマンドライン シェルおよびスクリプト言語です。InstallShield では、PowerShell スクリプトを実行するカスタム アクションをサポートします。この種類のカスタム アクションは、インストールの実行時にシステムの構成タスクを実行するプロジェクトに追加することができます。

インストールで、PowerShell カスタム アクションを実行するには、Windows PowerShell がターゲット システムにインストールされている必要がありますので注意してください。InstallShield には、ターゲット システムで PowerShell の有無を確認する新しい定義済み PowerShell システム検索が追加されました。このシステム検索をプロジェクトに含めると、PowerShell カスタム アクションが、PowerShell がインストールされていることが判別されたときのみ実行されるように構成できます。

ターゲット システムで PowerShell スクリプトを実行できるかどうかを判別する PowerShell 実行ポリシーは、デフォルトで、PowerShell スクリプトの実行が許可していない制限付きになっています。インストールで、ターゲット システムの実行ポリシーを、インストールの PowerShell カスタム アクションに適切なポリシーでオーバーライドする場合、Windows Installer プロパティ IS_PS_EXECUTIONPOLICY を使用して、適切な実行ポリシーを指定できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、「PowerShell カスタム アクションの呼び出し」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI インストールのアップデートの自動確認とダウンロードのサポート

スイート/アドバンスト UI およびアドバンスト UI インストールでは、Web サイトでホストする更新済みのスイート/アドバンスト UI またはアドバンスト UI の Setup.exe ファイルを自動確認し、アップデートが存在する場合、それをダウンロードし起動する機能を使用できます。更新されたスイート/アドバンスト UI またはアドバンスト UI の Setup.exe ファイルは、最新のスイート/アドバンスト UI およびアドバンスト UI パッケージ用のアップグレードやパッチを配信するために使用できます。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトの [リリース] ビューの [アップデート] タブに、更新されたスイート/アドバンスト UI またはアドバンスト UI セットアップ ランチャーの絶対パス (ファイル名を含む) を指定できる "アップデート URL" 設定が新しく追加されました。ベースのスイート/アドバンスト UI またはアドバンスト UI セットアップ ランチャーによってメンテナンス以外の操作が実行される場合、アップデート URL をチェックして、ダウンロード可能なアイテムがあるかどうかが確認されます。ダウンロード可能なアイテムが存在する場合、ダウンロードが実行され、デジタル署名が確認されます。アップデート セットアップ ランチャーのデジタル署名がベースのセットアップ ランチャの署名と一致すると、アップデート セットアップ ランチャーが自動的に実行されます。デジタル署名が一致しなかった場合、または、ベース セットアップ ランチャーがデジタル署名されていない場合、エンドユーザーがアップデート セットアップ ランチャーの実行を続けるかどうかを選択できるセキュリティ警告が表示されます。

詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

特定のバージョンのスイート/アドバンスト UI またはアドバンスト UI インストールが既にインストールされているかどうかを判別できる機能

スイート/アドバンスト UI およびアドバンスト UI プロジェクトに、特定のバージョンのスイート/アドバンスト UI またはアドバンスト UI インストールが既にインストールされているかどうかを判別するためのサポートが追加されました。このタイプの条件チェックは "インストール済みスイート" 条件と呼ばれます。

デフォルトで、スイート/アドバンスト UI およびアドバンスト UI インストールには、それぞれ、2 つの "インストール済みスイート" 条件が含まれています:

- 新しいインストール済みスイートの終了条件は、エンドユーザーが、スイート/アドバンスト UI またはアドバンスト UI インストールの現在のバージョンによって、同じスイート/アドバンスト UI またはアドバンスト UI インストールの将来のバージョンが上書きインストールされるのを防ぎます。
- 同じバージョンのスイート/アドバンスト UI またはアドバンスト UI インストールが既にインストールされている場合、"インストール済みスイート" のモード条件によって、インストールが初回インストール モードで実行されるのが禁止されることがあります。

これらの新しいデフォルトの条件は、スイート/アドバンスト UI およびアドバンスト UI プロジェクトで使用できます。InstallShield 2012 スイート プロジェクトを InstallShield 2025 にアップグレードした場合、これらのデフォルト条件は自動的にプロジェクトに追加されます。

デフォルトの "インストール済みスイート" 条件は、必要に応じて編集が可能です。また、必要に応じて、プロジェクトに自作のインストール済みスイート条件を追加することもできます。

詳細については、次を参照してください。

- 特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する
- アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする
- "インストール済み" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

インストールのフォルダーのネットワーク共有を構成できる機能

InstallShield では、ネットワーク上でファイルの共有を有効にするネットワーク共有の作成、変更、および削除が可能になりました。プロジェクトでフォルダーを構成するとき、ネットワーク共有を有効にするかどうかを指定します (デフォルトでは無効になっています)。また、共有の名前、共有にアクセスできる同時ユーザーの最大数など、その他のオプションも構成できます。

共有を有効にするには、[プロパティ] ダイアログ ボックスに追加された新しい [共有] タブ ([ファイルとフォルダー] ビューでフォルダーを右クリックすると表示されます) を使用し、[プロパティ] をクリックします。

この機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、および MSM データベース プロジェクト タイプで使用できます。

詳細については、「[フォルダーのプロパティ] ダイアログ ボックス」を参照してください。

特定のプロセスを終了する新しいビルトイン カスタム アクション

InstallShield に、新しい Kill-Process タイプのカスタム アクションが追加されました。このタイプのカスタム アクションをプロジェクトに追加すると、実行時に強制終了する 1 つまたは複数のプロセスの名前または ID を指定することができます。また、即時または遅延モードのカスタム アクションをスケジュールすることもできます。

このタイプのカスタム アクションを作成する手順には、プロジェクトの [カスタム アクションとシーケンス] ビューにカスタム アクションを追加して構成する作業と、プロパティ マネージャーを使って、適切なプロセスの名前または PID で、プロパティを定義する作業が含まれます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳しい手順については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。

実行時に既定のユーザー アカウントとグループを作成できる機能

InstallShield に、実行時に、複数のユーザー アカウントと対応するグループを、ログイン ダイアログを使わずに作成できるビルトイン サポートが追加されました。アカウントおよびグループを構成するには、プロジェクトのプロパティ ISNetApiLogonUsername、ISNetApiLogonGroup、および ISNetApiLogonPassword の値をそれぞれユーザー名、グループ、パスワードで定義します。複数の名前、グループ、またはパスワードは、角かっこを使ったチルダ [~] で囲んで区切ります。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、「実行時に既定のユーザー アカウントを作成する」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトに InstallShield 前提条件をパッケージとして含める機能

新しい InstallShield で、InstallShield 前提条件を、.msi や .exe パッケージとしてスイート/アドバンスト UI およびアドバンスト UI プロジェクトにインポートすることができるようになりました。インポートできる前提条件は、InstallShield に含まれている InstallShield 前提条件だけでなく、自分で作成したカスタム InstallShield 前提条件もインポートすることができます。[パッケージ] ビューにある [パッケージ] エクスプローラーを右クリックし、新しい 前提条件 (.prq) のインポート] コマンドをクリックすると、プロジェクトに、前提条件に対して実行されるように設定されたファイルの種類に応じて、.msi パッケージまたは .exe パッケージが追加されます。また、.prq ファイルで構成された設定に基づいて、前提パッケージの各設定にデフォルトの値が自動的に設定されます。これらの設定は、スイート/アドバンスト UI またはアドバンスト UI プロジェクトでパッケージの設定を変更するのと同様、必要に応じて変更が可能です。

InstallShield 前提条件に依存関係がある場合 (つまり、1 つまたは複数の .prq ファイルが、スイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加する InstallShield 前提条件で、依存関係として指定されている場合)、依存関係にある前提条件が [パッケージ] エクスプローラーに別のパッケージとして自動的に追加されます。

以前、スイート プロジェクトでは、.msi および .exe インストールをパッケージとして追加し、パッケージごとに、すべての条件と設定を手動で構成する必要がありました。

詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

スイート/アドバンスト UI またはアドバンスト UI プロジェクトに InstallScript インストールをパッケージとして含める機能; 新しいスイート/アドバンスト UI またはアドバンスト UI 固有 InstallScript イベントと関数

新しい InstallShield では、InstallScript インストールを、パッケージとして、スイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加することができます。スイート/アドバンスト UI またはアドバンスト UI インストールで InstallScript パッケージが起動されたとき、それ自身のユーザー インターフェイス (UI) が表示され、InstallScript パッケージの UI は自動的に抑制されます。これにより、インストールで、最新の UI 使用感を提供できます。スイート/アドバンスト UI またはアドバンスト UI インストールでは、InstallScript パッケージに対して、進行状況も表示されます。

これらの変更を可能にするために、スイート/アドバンスト UI およびアドバンスト UI インストールでは、デフォルトで、いくつかのスイート/アドバンスト UI およびアドバンスト UI 固有の InstallScript イベントおよび関数が使用され、一部の標準 InstallScript イベントおよび関数は無視されます。

スイート/アドバンスト UI またはアドバンスト UI プロジェクトにおける新しい InstallScript パッケージのサポート

InstallShield では、InstallScript パッケージが次の要件を満たしたとき、InstallScript パッケージをスイート/アドバンスト UI またはアドバンスト UI プロジェクトに追加することができます。

- InstallScript パッケージは非圧縮です。
- InstallShield 2012 Spring 以降が、InstallScript パッケージまたはスイート/アドバンスト UI またはアドバンスト UI インストールを作成するために使われている。
- InstallScript パッケージで、イベント ベースのスクリプトが使用される。program...endprogram スタイルのスクリプトは使用しない。

詳細については、「InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

新しいスイート/アドバンスト UI およびアドバンスト UI 固有の InstallScript イベントと関数

InstallScript Setup.exe ファイルから起動される (スイート/アドバンスト UI またはアドバンスト UI インストールから起動されない) 標準の InstallScript インストールでは、ほとんどのイベントは OnShowUI イベントから直接呼ばれます。InstallScript パッケージを起動するアドバンスト UI またはスイート/アドバンスト UI またはアドバンスト UI インストールでは、OnShowUI がOnSuiteShowUI によって置き換えられます。インストール状態 (初回インストール、メンテナンス、またはアップデート) に応じて、OnSuiteShowUI では、OnFirstUIBefore や OnFirstUIAfter などの UI イベントが無視され、次のイベントが呼び出されます:

- **初回インストール**—OnSuiteInstallBefore、OnSuiteInstallAfter
- **メンテナンス**—OnSuiteMaintBefore、OnSuiteMaintAfter
- **アップデート**—OnSuiteUpdateBefore、OnSuiteUpdateAfter

InstallScript 言語には、InstallScript パッケージとそれを実行するアドバンスト UI またはスイート/アドバンスト UI インストールの間の対話を可能にする新しいスイート/アドバンスト UI およびアドバンスト UI 固有の関数がいくつか追加されました。たとえば、InstallScript には、InstallScript パッケージの情報をスイート/アドバンスト UI およびアドバンスト UI のデバッグ ログに記録できる関数、スイート/アドバンスト UI およびアドバンスト UI のプロパティを設定および取得する関数、およびスイート/アドバンスト UI およびアドバンスト UI インストールからのデータを InstallScript パッケージに渡す関数が新しく含まれています。

InstallScript インストールが、スイート/アドバンスト UI またはアドバンスト UI インストール内の InstallScript パッケージとして実行されているかどうかを判別するには、InstallScript コードで新しい SUITE_HOSTED 変数を使用します。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける、新しい InstallScript パッケージの条件の種類の確認

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。新しい InstallScript パッケージの条件チェックを使うと、ターゲット システムで特定の InstallScript パッケージによってインストールされた製品の存在をチェックできます。特定の製品コードの条件チェックでは、製品バージョンなどのその他の情報もチェックできます。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- "InstallScript パッケージ" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける、パッケージ ファイルのダイナミック ファイル リンクのサポート

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、.msi、.msp、または .exe パッケージを追加または構成するとき、パッケージに、パッケージ ファイルの近くに保存されている追加ファイルが必要かどうかを示すことができます。たとえば、追加するパッケージが非圧縮の .msi パッケージである場合、その他のファイル (近くのサブフォルダーに保存されている .cab ファイルや非圧縮データ ファイルなど) をパッケージとファイルに含める必要が、場合によって、あります。

InstallShield では、追加のパッケージ ファイルにダイナミック リンクを使うことができます。ダイナミック リンクは、パッケージに必要な追加ファイルのリストがビルドとビルドの間に変更する可能性があるときに便利です。ソース フォルダーは常にビルドの前にスキャンされ、すべての新規または変更されたパッケージ ファイルが自動的にリリースへ組み込まれます。

InstallShield ではまた、ビルド時にダイナミック リンクに含めるまたは除外する追加ファイルを制御するフィルターを定義できます。ダイナミック リンクに対して定義したフィルター基準は、InstallShield で評価されるときの順番を変更することができます。アドバンスト UI またはスイート/アドバンスト UI インストールをビルドすると、ダイナミック リンクのフィルターを基に、常に適切な追加ファイルが含められます。

以前、追加ファイルには、スタティック リンクのみ使用することができました。ビルド間で追加ファイルのリストに変更が発生した時、パッケージ ファイルのリストは手動で更新する必要がありました。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトでダイナミック リンクを作成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する

スイート/アドバンスト UI またはアドバンスト UI のインストール中にエンドユーザー入力を検証したときの拡張フィードバックの生成

スイート/アドバンスト UI およびアドバンスト UI インストールでは、実行時、エンドユーザー入力の検証で、拡張フィードバックを生成することができるようになりました。スイート/アドバンスト UI またはアドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューで提供されている様々なインターフェイス コントロールに、既存の "テキスト スタイル" 設定の下に新しいサブ設定 ("デフォルト"、"有効"、および "無効") が追加されました。これらのサブ設定を構成して、スイート/アドバンスト UI またはアドバンスト UI インストールで、異なる状況に応じて使用するテキスト スタイルを選択することができます。

スイート/アドバンスト UI またはアドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート

スイート/アドバンスト UI またはアドバンスト UI プロジェクトのパッケージ設定を構成するとき、スイート/アドバンスト UI またはアドバンスト UI インストールを /log コマンドライン パラメーターを使ってコマンドラインから起動する場合、新しい "ログの有効化" 設定を使って、ログ ファイルを生成するかどうかを指定できます。パッケージの種類 (.msi パッケージ、.msp パッケージ、またはその他の種類のパッケージ) に応じて、提供されている設定を構成して、その他の情報 (ログ ファイルが作成されるときにパッケージに渡すログ オプションなど) も指定できます。

スイート/アドバンスト UI またはアドバンスト UI の Setup.exe ファイルの新しい /log コマンドライン パラメーターを使用して、パッケージのログ ファイル用のディレクトリを指定することができます。パスが /log パラメーターを使用して指定されなかった場合、パッケージ ログ ファイルは %TEMP% ディレクトリに作成されます。

新しいスイート/アドバンスト UI またはアドバンスト UI インストールのプロパティ ISLogDir は、パッケージ ログ ファイルを含むディレクトリへのパスを格納します。

以前、スイート インストールの Windows Installer ベースのパッケージに対してログを有効化するには、ログ システム ポリシーを使用するか、または、MsiLogging プロパティを使用する方法しかありませんでした。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート
- アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

InstallScript インストールでの 64 ビット コンポーネントのサポート

InstallScript プロジェクトで、InstallScript コードを変更せずに、ファイルを WINSYSDIR64 (64 ビット System32 フォルダーにマップする InstallScript 変数) にインストールするサポートと、レジストリ データを 64 ビットのレジストリの場所にインストールできるサポートが追加されました。これらの 64 ビットの場所にインストールする必要があるファイルまたはレジストリ データがある場合、それらのファイルとレジストリ データをコンポーネントに追加して、コンポーネントの新しい "64 ビット コンポーネント" 設定で [はい] を選択します。実行時に、コンポーネントの System32 ファイルに対して、ファイル システムのリダイレクトが自動的に無効になり、コンポーネントの 64 ビットのレジストリ データのリダイレクトが防止されます。

以前、ファイルを WINSYSDIR64 にインストールするには、その場所にインストールされたコンポーネントを含む機能に対して Installing と Installed のイベントを上書きする必要がありました。Installing イベントでは、WOW64FSREDIRECTION 定数を **Disable** 関数と一緒に使って、ファイル システムのリダイレクトを無効にし、Installed イベントでは、WOW64FSREDIRECTION を **Enable** 関数と一緒に使って、インストールの他の場所でのリダイレクトを再有効にする必要がありました。また、アンインストールでも、これらのファイルが正しく削除されるように、UnInstalling と UnInstalled イベントで、同じ無効と有効の操作が必要でした。

ファイル システムのリダイレクトが、WINSYSDIR64 へのインストールで無効になっていなかった場合、64 ビット Windows では、ファイル転送は自動的に 32 ビットの System32 フォルダー (SysWOW64) にリダイレクトされます。

また、以前のバージョンでは、レジストリ データを 64 ビット領域のレジストリにインストールする場合、InstallScript のレジストリ関数を使って、REGDB_OPTIONS に REGDB_OPTION_WOW64_64KEY が設定されているレジストリ データを作成する必要がありました。そしてこの後、REGDB_OPTION_USE_DEFAULT_OPTIONS を REGDB_OPTIONS と共に使用して、インストールの他の箇所に対して、レジストリのリダイレクトを再有効にする必要がありました。

64 ビットのレジストリの場所 (HKEY_LOCAL_MACHINE¥Software) へのインストールでレジストリのリダイレクトが無効になっていない場合、64 ビットの Windows では、レジストリ変更が、相当する 32 ビットのレジストリの場所 (HKEY_LOCAL_MACHINE¥Software¥Wow6432Node) へ自動的にリダイレクトされます。

今回より、製品のインストール時に、InstallScript インストールによって作成された InstallScript ログ ファイル (.ilg) では、新しい OPTYPE_FILE64 タイプを使って、ファイル システムのリダイレクトが無効のときにインストールされたファイルを識別するために使われています。InstallScript ログ ファイルでは、既存の OPTYPE_REGISTRY64 タイプが使われて、レジストリのリダイレクトが無効のときにインストールされたレジストリ データが識別されます。OPTYPE_FILE64 と OPTYPE_REGISTRY64 タイプを見るには、InstallShield のキャビネット & ログ ファイル ビューアーを使って .ilg ファイルを表示します。

詳細は、次を参照してください:

- 64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする
- コンポーネントの設定
- WINSYSDIR64
- REGDB_OPTIONS
- InstallShield キャビネット & ログ ファイル ビューアー

スイート/アドバンスト UI およびアドバンスト UI プロジェクトの新しいロケール タイプの条件チェック

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な種類から選択できます。新しいロケール タイプの条件チェックを使うと、ターゲット システムで、1 つまたは複数のロケール関連の設定に一致するものがあるかどうかを確認することができます。

詳しくは、次を参照してください:

- アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類
- "ロケール" 条件の設定
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

スイート/アドバンスト UI およびアドバンスト UI におけるレイアウト編集用の新しいウィザード インターフェイス ツールバーと、スイート/アドバンスト UI プロジェクトにおける言語の切り替え機能のサポート

スイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューでウィザード ページまたは二次ウィンドウを選択すると、ウィザード インターフェイスのプレビュー ペインのすぐ上に表示されるツールバーに、選択したページまたはウィンドウを変更できるボタンやコントロールが追加表示されます。また、ウィザード ページまたは 2 番目のウィンドウでコントロールを選択したときも、[ウィザード インターフェイス] ビューにこのツールバーが表示されます。

新しいツールバーには、インストールのユーザー インターフェイスにラベル、テキスト ボックス、チェックボックス、その他のコントロールを追加できるボタンが提供されています。また、ツールバーには、選択したコントロールの整列、サイズ調整、位置調整を簡単に行うことができるボタンも提供されています。スイート/アドバンスト UI プロジェクトの新しいツールバーにある [デフォルト言語] リストでは、このビューのウィザード ページおよび二次ウィンドウで表示する文字列を、プロジェクトで使用する別の言語の文字列に切り替えることができます。

詳細については、次を参照してください。

- ウィザード ページまたは 2 番目のウィンドウにコントロールを追加する
- [ウィザード インターフェイス] ビューのツールバー

スイート/アドバンスト UI プロジェクトに言語を追加するサポート

InstallShield Premier Edition には、35 ヶ国語のデフォルト ランタイム文字列がサポートされています。サポート対象の言語をスイート/アドバンスト UI プロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield はプロジェクトにその言語に翻訳された文字列エントリを追加します。それらは、デフォルト ウィザード ページ、メッセージ、およびその他のエンド ユーザー インターフェイス要素の文字列エントリです。

新しい InstallShield のスイート/アドバンスト UI プロジェクトでは、新しい言語ウィザードを使って 35 ヶ国語以外のサポートされていない言語も追加することができるようになりました。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をスイート/アドバンスト UI プロジェクトに追加すると、その言語は、プロジェクト内の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

スイート/アドバンスト UI プロジェクトで新しい言語ウィザードを起動するには、ツール メニューで、[新しい言語の追加] をクリックします。

詳しくは、次を参照してください:

- 新しい言語ウィザード
- InstallShield 実行時の言語サポート

実行時にターゲット システムでスケジュール タスクを作成および構成できる機能

InstallShield に、ターゲット システムで、実行時に、Windows のタスク スケジューラを使って作成するタスクを構成できる [スケジュール タスク] ビューが追加されました。ビューでは、タスクに対して起動するファイルや、開始日付と時刻などの情報を指定できます。起動するファイルは、インストールに含まれているファイルでも、ターゲット システムに既に存在するファイルでも可能です。

この機能は、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- スケジュール タスク
- [スケジュール タスク] ビュー

新しい FlexNet Connect 13.03 再配布可能ファイル

InstallShield は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 13.03 をサポートします。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 13.03 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。

強化機能

InstallShield には、以下のような強化機能が搭載されています。

オペレーティング システムに関する InstallScript 言語の強化

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- SYSINFO.WINNT.bWin8 – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 8 または Windows Server 8 の場合、この値は TRUE です。
- ISOSL_WIN8 – FeatureFilterOS 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 8 または Windows Server 8 を実行中であることを示します。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

オートメーション インターフェイスの強化機能: Windows 8 と Windows Server 8 用の OSFilter プロパティの値

オートメーション インターフェイスで、ISWiComponent と ISWiRelease オブジェクトの OSFilter メンバーと共に、以下の定数を使用できます:

eosWin8 = &H4000000 (67108864) È これらの定数は Windows 8 および Windows Server 8 用です。

また、eosAll 定数の値が &3D100D0 (64028880) から &7D100D0 (131137744) に変更されました。

OSFilter メンバーは、InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトで ISWiComponent オブジェクトに適用します。OSFilter メンバーは、InstallScript、および InstallScript オブジェクト プロジェクトで ISWiRelease オブジェクトに適用します。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで条件を簡単に移動できる機能

スイート/アドバンスト UI または アドバンスト UI プロジェクトにおいて、終了、検出、対象、または機能条件の 1 つ以上の条件ステートメントがある場合、条件ステートメントを移動して、条件を並べ替えたり、条件ツリーの階層を変更したりできます。たとえば、None 条件グループにプラットフォーム条件ステートメントがある場合で、None 条件グループが All 条件グループの中にあるとき、プラットフォーム条件ステートメントを左に移動させて、All 条件グループの一部となるように変更できます。

条件ステートメントまたはグループを移動するには、移動させるアイテムの設定にある新しい [条件を移動] ボタンをクリックしてから、適切なオプション (上に移動、下に移動、左に移動、または右に移動) をクリックします。

以前は、手作業で新しい条件ステートメントを新しい場所に作成してから、古い場所にある条件ステーメントを削除しなくてはなりませんでした。あるいは、テキスト エディターの .issuite ファイルを編集して、条件ステートメントの順番を変更する方法もあります。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトにおける新しい拡張条件のサポートと機能強化された条件設定

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで、新しい拡張条件機能が使用できるようになりました。このタイプの条件を利用すると、ターゲット システム上で、作成した C/C++ DLL を参照して、自作のカスタム条件があるかどうかを確認できます。

また、スイート/アドバンスト UI およびアドバンスト UI プロジェクトで提供されている条件に関連する設定の多くが機能強化され、条件ステートメントを簡単にビルドできるようになりました。たとえば、いくつかの条件設定の 1 つで、手動で製品コード、アップグレード コード、パッチ コード、またはその他のタイプのデータの入力を試みる代わりに、設定にある新しい省略記号ボタン (...) を使うことができます。ボタンをクリックすると、適切なパッケージを参照し選択できる参照ダイアログ ボックスが表示されます。パッケージを選択すると、スイート/アドバンスト UI またはアドバンスト UI プロジェクトの設定に、そのパッケージの情報が正しく設定されます。

対象となるパッケージ条件を構成するとき、適切なパッケージ GUID を手動で入力する代わりに、スイート/アドバンスト UI またはアドバンスト UI プロジェクトのパッケージ リストから選択できます。

詳細については、次を参照してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールのカスタム条件を作成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する
- "拡張条件" の設定

スイート/アドバンスト UI またはアドバンスト UI プロジェクトのウィザード ページまたはウィンドウのコントロールに対して行う検証とアクションを構成における機能強化

スイート/アドバンスト UI またはアドバンスト UI プロジェクトで、ウィザード インターフェイスで表示できる様々なコントロールの "検証" 設定と "アクション" 設定が強化され、検証の定義と様々のアクションのトリガをより簡単に行えるようになりました。これらの設定に、これらの設定を使って入力可能なサンプル ステートメントのドロップダウン リストが追加されました。また、依然同様、これらの設定をテキスト ボックスとして使用し、ステートメントを手入力することも可能です。たとえば、"検証" 設定では、エンドユーザーがコントロールでシリアル番号を入力したときに合致する必要があるフォーマットを指定することができます。検証ステートメントをすべて手入力する代わりに、"検証" 設定でマスク タイプの検証を選択して、この設定のデフォルト フォーマットをオーバーライドすることもできます。

同様に、"アクション" 設定では、例として、ボタン コントロールの印刷アクションを定義できます。つまり、エンド ユーザーが [印刷] ボタンをクリックすると、[印刷] ダイアログ ボックスが開いて、エンド ユーザーが使用許諾契約を印刷できるようになります。今回より、アクション ステートメントをすべて手入力する代わりに、"アクション" 設定で印刷アクションを選択して、そのファイル名を、印刷されるファイルの名前でオーバーライドすることができます。

"検証" 設定と "アクション" 設定のドロップダウン リストに、自分で作成し、[サポート ファイル] ビューを使ってプロジェクトに追加した C/C++ DLL ファイルが表示されるようになりました。今回より、この機能によって、ウィザード インターフェイスの様々なコントロールに対して、自作の検証やアクションをトリガできます。

詳しくは、次を参照してください:

- ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する
- ウィザード インターフェイス内の要素のアクションを構成する

スイート/アドバンスト UI およびアドバンスト UI リリースのパッケージにカスタム フォルダー名を使用できる機能

スイート/アドバンスト UI またはアドバンスト UI インストールをビルドしたとき、インストールに含まれている各パッケージに対してフォルダーが 1 つ作成されます。これらのフォルダーは、スイート/アドバンスト UI またはアドバンスト UI のセットアップ ランチャーと同じフォルダー内で作成されます。デフォルトで、各フォルダーの名前には、InstallShield で生成された GUID が使用されます。

InstallShield の [パッケージ] ビルドでは、今回から、これらのパッケージ フォルダーに対して、この GUID 名をオーバーライドして、ユーザーに分かりやすい名前を使用できるようになりました。このビューでフォルダーのカスタム名を入力するには、フォルダーの名前をカスタマイズするパッケージの下で、[パッケージ ファイル] フォルダーを見つけます。フォルダーを右クリックし、[名前を変更] をクリックして、新しい名前を入力します。複数のフォルダー名をカスタマイズする場合、各フォルダー名に異なる名前を使用してください。

以前、各フォルダーの名前には GUID のみが使用され、名前のカスタマイズはサポートされていませんでした。

手順については、「アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで他のプロパティを参照する値を持つプロパティにおける新しい "フォーマット済み" サポート

スイート/アドバンスト UI またはアドバンスト UI プロジェクトの [プロパティ マネージャー] ビューで定義された各プロパティに、新しい "フォーマット済み" チェック ボックスが追加されました。この新しいチェック ボックスを使って、実行時に、プロパティの値で参照されるプロパティを解決し、これらのプロパティの値で置換するかどうかを示すことができます。

実行時に、角かっこで囲まれたプロパティ（[PropertyName] など）を置き換える場合、このチェック ボックスを選択します。角かっことそのコンテンツをそのままにしておく場合、このチェック ボックスをクリアします。

詳細については、「[プロパティ マネージャー] ビュー」を参照してください。

起動者マニフェストがあるスイート/アドバンスト UI およびアドバンスト UI インストールに対して権限の昇格が必要なダウンロードされたパッケージの UAC プロンプトのタイミングが改善されました

起動者マニフェストがあるアドバンスト UI またはスイート/アドバンスト UI インストールをビルドして、そのインストールに含まれているターゲット システムでダウンロードし起動する必要があるパッケージの中に、"昇格された権限が必要" 設定で [はい] が選択されているものが存在するとき、UAC プロンプトは、エンドユーザーが [インストール] ボタンをクリックした直後（ダウンロードが開始される前）に表示されます。

以前、UAC プロンプトは、ダウンロードが発生した後に表示されていました。このため、パッケージのステージングに時間がかかっていた場合（パッケージのダウンロードに時間がかかっていた場合など）、エンドユーザーが [インストール] ボタンをクリックした瞬間から、Windows でパッケージの昇格を求める UAC プロンプトが表示される瞬間のまでに大きなギャップが開いていた可能性があります。エンドユーザーが速やかに同意しなかった場合、または認証情報を入力しなかった場合、UAC プロンプトがタイムアウトになり、インストールが失敗していました。

以前の一部のケースでは、UAC プロンプトがエンドユーザーが [インストール] ボタンをクリックした直後に表示されていたため、管理マニフェストを使うことで、この問題を回避することができていましたが、この方法では、インストール全体に昇格された権限が使用されていました。

スイート/アドバンスト UI およびアドバンスト UI インストールのウィザード インターフェイスで使用されているテキストの配置を指定できるサポート

InstallShield には、スイート/アドバンスト UI およびアドバンスト UI プロジェクトのウィザード インターフェイスで使用されているテキスト属性（テキストの色、サイズ、フォント名など）を定義する多くのビルトイン テキスト スタイルが含まれています。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューを使って、これらのビルトイン スタイルの任意の設定を編集、または独自のスタイルを定義することができます。

各ビルトインまたはカスタム テキスト スタイルには、特定のスタイルを使用するコントロール内のテキストに対して配置のタイプを選択できる "テキスト配置" 設定が新しく追加されています。

詳細については、「スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI インストールのウィザード インターフェイスの機能拡張されたコンボ ボックス コントロール

スイート/アドバンスト UI およびアドバンスト UI プロジェクトで、ウィザード ページまたは二次ウィンドウに、コンボボックス コントロールを追加することができます。このタイプのコントロールには、デフォルトで 2 つのコントロールがあります:

- 定義された値を既に持つドロップダウン リストを含むボックス
- エンドユーザーがカスタム値を入力できるテキスト ボックス

以前のバージョンでは、新しいコンボボックス コントロールを追加したとき、コントロールにドロップダウン リストが含まれていましたが、テキストボックスは提供されていなかったため、エンドユーザーはカスタム値を入力することができませんでした。

このコントロールをテキストボックスのないドロップダウン リストに変更するには (つまり、エンドユーザーが定義済みの値を選択できるが、カスタム値を入力できるようにする場合)、このコントロールの CBS_DROPDOWNLIST スタイルを False に設定します。

スイート/アドバンスト UI およびアドバンスト UI ウィザード インターフェイスの Aero フォーマットの拡張と変更

スイート/アドバンスト UI およびアドバンスト UI インストールの Aero フォーマットのウィザード ページが、一部拡張および変更されました。

- ウィザード ページのヘッダーおよびナビゲーション領域が、Aero ガラス効果 (半透明) で表示されるようになりました。以前、ウィザード ページのキャプション バーのみ、Aero ガラス効果で表示されていました。
- Aero フォーマットのウィザード ページでは、今回より、Wizard 97 フォーマットのウィザード ページと同じレイアウトが使用されています。つまり、Aero フォーマットのウィザード ページのキャプション バーの縦幅が低くなり、Wizard 97 フォーマット ウィザード ページのキャプション バーと同じ縦幅が適用されています。また、Aero フォーマットのウィザード ページの [戻る] ボタンが、Wizard 97 フォーマットのウィザード ページ同様、ナビゲーション領域で表示されるようになりました。以前、[戻る] ボタンは、Aero フォーマットのウィザード ページのキャプション バーの左上に表示されていました。

設定から文字列値とその ID を削除できる機能

InstallShield のビュー内にある設定で値を入力するときに、その値がエンド ユーザーに表示されるテキスト文字列である場合、自動的にその設定の文字列 ID が使用されます。InstallShield は、文字列値の前に中括弧で囲まれた文字列 ID を配置します。これらの種類の設定には、**[この文字列参照を削除]** ボタンが追加されました。

設定から文字列 ID とその値を削除する場合、この新しいボタンをクリックします。このボタンを使って、設定のエントリをクリアできます。文字列 ID とその値をプロジェクトから削除するには、[文字列エディター] ビューを使います。

InstallShield 2012 SP1 の新しい機能

強化機能

InstallShield には、以下のような強化機能が搭載されています。

ソフトウェア識別タグのデジタル署名サポート

プロジェクトにソフトウェア ID タグを含めて [リリース] ビューで .pfx ファイルを使ってリリースに署名を行うことを設定すると、InstallShield がビルド時にタグをデジタル署名します。タグ ファイルに署名するためには、.NET Framework 2.0 以降をビルドマシンにインストールする必要があります。

詳細については、「製品のソフトウェア識別タグを含める」を参照してください。

InstallShield 2012 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

複数のパッケージを実行するスイート インストールを作成できる機能、最新のカスタマイズ可能なエンド ユーザー インターフェイス、ハイブリッド 32 ビット/64 ビット インストールをビルドできる機能

今回より InstallShield Premier Edition では、条件付きで複数のインストールを実行し、必要に応じてターゲット システムに Windows Installer パッチ (.msp) を適用することが可能な次世代のセットアップランチャー (Setup.exe) を使ったスイート インストールをビルドすることができます。この機能は、新しいスイート プロジェクト タイプで使用できます。スイート インストールは Windows XP 以降および Windows Server 2003 以降が搭載されたシステムで実行します。

機能の主な特徴は以下の通りです。

複数のインストールを単一のインストールとしてパッケージできる機能

新しいスイート プロジェクト タイプには、以下の種類のパッケージから 1 つ以上を指定できる [パッケージ] ビューが含まれています。

- ターゲット システムで実行する Windows Installer ベースおよび Windows Installer ベース以外のインストールを含む実行可能ファイル (.exe)
- ターゲット システムで実行する Windows Installer パッケージ (.msi)
- ターゲット システムに適用する Windows Installer パッチ (.msp)

また [パッケージ] ビューを使って、トランザクション処理を使って実行する .msi および .msp パッケージを含めることもできます。これは Windows Installer 4.5 以降の機能です。パッケージを連鎖させることによって、単一のトランザクションとして処理します。各スイート インストールには、複数の個別のトランザクションを含めることができます。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は現在のトランザクションに含まれる連鎖パッケージ全てのロールバックを開始して、システムを以前の状態に復元します。

スイート インストールは定義された条件および [パッケージ] ビューのパッケージでリストされている順番に従って、実行時に適切なパッケージを起動します。

最新のカスタマイズ可能なインストールのユーザー インターフェイス、およびスイート Setup.exe ウィザード ページをカスタマイズできる新しいエディター

InstallShield のスイート プロジェクト タイプでは、カスタマイズが可能な全く新しい、最新デザインのビルトイン ウィザード ページを使った、エンド ユーザー インターフェイスをインストールに含めることができます。このプロジェクトの種類で使用できる新しいウィザード ページ エディターでは、必要に応じてページを追加、シーケンス、または削除することができます。また、様々な異なる種類のコントロールを追加または削除することで任意のページのレイアウトを編集することもできます。

64 ビットおよび 32 ビットの Windows Installer パッケージを単一のインストールに組み合わせるサポート

より多くのユーザーが Windows 64 ビット バージョンに移行する中、32 ビット システム上では 32 ビットの場所に、また 64 ビット システムでは 64 ビットの場所にインストールする単一のインストールを、今すぐまたは近い将来に配布する必要が出てくるでしょう。スイート プロジェクト タイプを使うと、32 ビット パッケージと 64 ビット パッケージの両方を単一のスイート インストールに含めて、各ターゲット システム上に適切なパッケージのみを実行することができます。以前は、2 つの個別のインストール (32 ビット システム用および 64 ビット システム用) を配布するか、カスタム起動ツール、ブートストラップ アプリケーション、または InstallShield インストールを作成する必要がありました。

スイート パッケージ全体のステータスを表示する単一の進行状況バーを表示するサポート

スイート インストールの進行状況ウィザードページで表示される進行状況バーには、スイート パッケージ全体のステータスが表示されます。この統合された進行状況バーによって、エンド ユーザーはスイート インストール全体の進行状況をはっきりと視覚的に確認することができます。エンド ユーザーに対して統合された進行状況バーのみが表示されるように、ユーザー インターフェイスを非表示に (つまり、サイレントで実行) するコマンドライン パラメーターを指定した .exe インストール、.msi パッケージ、および .msp パッチのみを含めてください。

スイート インストールにおけるオプションの [プログラムの追加と削除] エントリ

スイート プロジェクトでは、スイート インストールに [プログラムの追加と削除] を含めるかどうかを指定できます。このエントリを使って、エンド ユーザーは必要に応じてスイートの管理、変更、または削除を行うことができます。スイート プロジェクトの [一般情報] ビューには、適切な動作を指定できる "[プログラムの追加と削除] エントリの表示" 設定があります。

スイート全体に対して単一のエントリのみを表示する場合、スイート プロジェクトに含めるパッケージからのエントリを非表示にしてください。

ユーザー インターフェイス無しでスイート インストールを実行するサポート

エンド ユーザーは、ユーザー インターフェイスを使って、またはユーザー インターフェイスなしのサイレントでスイート インストールを実行することができます。サイレント インストールは、エンド ユーザーによる操作が不要で、実行時にランタイム ウィザード ページを使った情報の入力や監視が不要です。

35 すべてのサポート言語に新しく追加された "デフォルト Setuup.exe ユーザー インターフェイス" 設定を使った、スイート ランタイム文字列の編集機能

スイート プロジェクトのビルトイン ウィザード ページに表示されるすべてのデフォルト文字列について、InstallShield でサポートされているすべての 35 ランタイム言語に対してその翻訳が用意されています。これらすべてのスイート文字列は、スイート プロジェクトの [文字列エディター] ビューに表示されます。[文字列エディター] ビューでは、その他のプロジェクトの種類で提供されているのと同じ堅牢な機能を使用することができるため、実行時、スイート インストールの処理中に表示されるテキスト文字列すべてを 1 ヶ所でまとめて制御することができます。

必要なときに必要なパッケージのみをダウンロードできる機能を搭載したコンパクトなベース Setup.exe ファイル

スイート プロジェクトには、スイート インストールに含まれる各パッケージのランタイム ソースの場所を指定できる柔軟なオプションが含まれています。スイート プロジェクトに含めるパッケージを定義するとき、個別のパッケージの場所を指定できます。選択可能なオプションは、以下のとおりです。

- Web 上で、必要に応じて Setup.exe によるダウンロードが可能
- Setup.exe に埋め込み、必要に応じてターゲット システムに展開
- 非圧縮ファイルでスイートのソース メディアに格納

スイート インストールで使用されるベース Setup.exe ファイルは、基本の MSI、InstallScript MSI、および InstallScript インストールで使用されるベース Setup.exe ファイルよりもはるかにサイズが小さいです。このため、エンド ユーザーはコンパクトなスイート Setup.exe ファイルを短時間でダウンロードすることができ、Setup.exe ファイルが必要に応じて 1 つ以上のパッケージをダウンロードおよび起動します。

詳細については、次を参照してください。

- アドバンスト UI およびスイート/アドバンスト UI インストールの作成
- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

- 実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する
- ウィザード インターフェイスを使って作業する
- アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター
- アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化が新しく更新され、サポートが拡張されました

InstallShield に新しいプロジェクト タイプ、DIM (*デベロッパー インストール マニフェスト*) が追加されました。DIM プロジェクトは、機能サイズのプロジェクトで、インストール パッケージの別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。以下は、DIM を利用した時のいくつかの利点です:

- DIM では、基本の MSI プロジェクトで提供されている機能とほぼ同じ機能がサポートされています。このため、DIM の作成者には、インストールの部分開発を行うために必要な柔軟性がすべて提供されています。
- リリース エンジニアは、DIM を複数の基本の MSI プロジェクトで繰り返し使用できるため、非常に効率的です。
- DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数の基本の MSI プロジェクトでそれらを参照することができます。

DIM を作成したあと、2 つある方法のいずれかで基本の MSI プロジェクトに追加することができます:

- **参照** – [セットアップのデザイン] ビューまたは [DIM リファレンス] ビューから、DIM プロジェクトの参照を基本の MSI プロジェクトに追加することができます。この方法を使用すると、DIM の要素は、ビルド時に基本の MSI プロジェクトにマージされます。基本の MSI インストールがビルドされるたびに、DIM プロジェクトの最新バージョンが参照され、生成されたインストールに含められます。この方法は、最も頻繁に使われる方法です。
- **インポート** – 新しい DIM のインポート ウィザードを使用して、DIM プロジェクトを基本の MSI プロジェクトにインポートできます。この方法は、デザイン時に、DIM データを基本の MSI プロジェクトにマージするときに行う目的で一回切り行われ、一度行われると元に戻すことはできません。

この新しくデザインが変わり拡張された DIM のサポートにより、ユーザーは、InstallShield Collaboration という別のツールを使って DIM を作成し、その DIM ファイルを InstallShield の基本の MSI プロジェクトにインポートしなくてはならないという手間がなくなりました。新しい DIM のサポートは、これまでのサポートに比べより堅牢になりました。新しい DIM プロジェクトによって、基本の MSI プロジェクトで使用可能な柔軟性の高いオーサリング機能が事実上すべて利用可能になりました。たとえば、新しい DIM プロジェクトでは、コンポーネントの作成においてフル制御が可能になり、これにより、コンポーネントを新しい DIM プロジェクトに追加したり、コンポーネントのキー ファイルを設定したり、コンポーネントの設定を構成できるようになりました。新しい DIM プロジェクトではまた、IIS Web サイトの構成も可能になりました。InstallShield Collaboration では、コンポーネントのデザインにおいて、同レベルの柔軟性が提供されておらず、また、IIS Web サイトを構成するためのビルトインサポートも提供されていませんでした。

DIM プロジェクトの作成機能は、InstallShield Premier Edition で使用できます。このサポートは、新しいコラボレーション アドオン、InstallShield Developer Installation Manifest Editor でも提供されています。DIM ファイルを基本の MSI プロジェクトに追加する機能は、InstallShield Premier Edition で提供されています。

詳細は、次を参照してください:

- 開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化
- プロジェクトに DIM を追加する
- インストール プロジェクトに DIM を統合するための適切な方法を判別する
- DIM でパス変数を使用する
- [DIM リファレンス] ビュー
- DIM のインポート ウィザード

Internet Explorer 9、SQL Server 2008 R2 Native Client、Windows Identity Foundation、その他の再配布可能ファイル用の新しい InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、いくつかの新しい InstallShield 前提条件が含まれています:

- Internet Explorer 9.0
- Microsoft SQL Server 2008 R2 Native Client 10.50.1600.1
- Windows Identity Foundation
- Microsoft VSTO 2010 Runtime (x64)
- Microsoft Office 2010 PIA (この前提条件は Microsoft Office 2010 Primary Interop Assemblies をインストールします。この前提条件を使用するには、マイクロソフトの Web サイトから PrimaryInteropAssembly.exe ファイルをダウンロードおよび実行して、.msi ファイルを抽出します。)

64 ビット依存関係スキャンのサポート

InstallShield の依存関係スキャナー (スタティック スキャン ウィザードおよびダイナミック スキャン ウィザード) には、プロジェクトに含まれる 64 ビット ファイルの 64 ビット依存関係を識別できるサポートが追加されました。Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用する場合、スキャナーは 64 ビット 依存関係を検出することができます。ウィザードでは、プロジェクト内で検出された依存関係の可能性がある各ファイルについて、それらを含めるかどうかを指定することができます。

また、Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用していて、以下のどちらかのビルトイン処理によって依存関係を検出する場合、InstallShield はプロジェクトに含まれる 64 ビット .NET アセンブリの 64 ビット 依存関係をスキャンすることができます。

- スタティック スキャン ウィザードを使って、64 ビット .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別できます。このウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。
- コンポーネントの "ビルド時に .NET スキャン" 設定を使って、プロジェクトをビルドする度に InstallShield が 64 ビット .NET アセンブリの依存関係を識別するかどうかを指定できます。InstallScript プロジェクトの場合、コンポーネントの ".NET アセンブリ" 設定を [ローカル アセンブリ] に設定しなくてはなりません。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

64 ビット依存関係の 64 ビット ファイルをスキャンするためには、InstallShield が、64 ビット オペレーティング システムにインストールされていなくてはなりません。Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのビルトイン スキャンはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみを検出できます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

詳しくは、次を参照してください:

- 32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い
- 64 ビット依存関係のスキャン
- アプリケーションの依存関係を識別する
- 依存関係の 64 ビット .NET アセンブリをスキャンする
- .NET アセンブリのプロパティおよび依存関係を識別する

64 ビットの場所にあるファイル、フォルダー、およびレジストリ キーのアクセス許可を設定するサポート

InstallShield では 64 ビットの場所にあるファイル、フォルダー、およびレジストリ キーのアクセス許可を設定するためのサポートが提供されています。サポートは、使用するプロジェクトの種類によって異なります。

Windows Installer ベースのプロジェクトにおけるカスタム InstallShield 処理の使用

カスタム InstallShield 処理を使ってファイル、フォルダー、およびレジストリ キーのアクセス許可を設定する場合、今回より、これらのアイテムが 64 ビットの場所にある場合にアクセス許可を設定できます。これには、レジストリの 64 ビットの場所および 64 ビット システム上の 64 ビット System32 フォルダも含まれます。フォルダーまたはレジストリ キーは、 64 ビットとマークされたコンポーネントに含めなくてはなりません (つまり、"64 ビット コンポーネント" 設定で [はい] が選択されている)。以前、コンポーネントが 64 ビットとマークされている場合、アクセス許可を設定することはできず、ランタイム エラーが表示されました。

このカスタム InstallShield 処理サポートは、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。[一般情報] ビューの "ロックダウンの設定方法" 設定で、アクセス許可の設定に使用する方法 (カスタム InstallShield 処理または従来型の Windows Installer 処理のいずれか) を指定できます。

InstallScript ベースのプロジェクトおよび Windows Installer ベースのプロジェクトにおける InstallScript 関数 SetObjectPermissions の使用

REGDB_OPTION_WOW64_64KEY オプションが有効な場合に、InstallScript 関数 **SetObjectPermissions** を使って 64 ビット レジストリ キーのアクセス許可を設定したとき、関数がそのアクセス許可を正しく設定します。REGDB_OPTION_WOW64_64KEY オプションの有効/無効に関わらず、**SetObjectPermissions** で強制的に 64 ビット レジストリ キーのアクセス許可を設定するには、新しい定数 IS_PERMISSIONS_OPTION_64BIT_OBJECT を使用します。定数 IS_PERMISSIONS_OPTION_64BIT_OBJECT は **SetObjectPermissions** の nOptions パラメーターに渡します。この定数を 32 ビット ターゲット システムで渡すことはできません。

SetObjectPermissions を呼び出して 64 ビットの System32 フォルダーにあるファイルまたはフォルダーのアクセス許可を設定するとき、WOW64FSREDIRECTION 定数を使ってシステム リダイレクトを無効とした場合、関数がアクセス許可を正しく設定します。ファイル システム リダイレクトが無効でない場合、アクセス許可は設定できません。

InstallScript および InstallScript MSI プロジェクトでは InstallScript イベントの **SetObjectPermissions** 関数を使用できます。この関数は、InstallScript、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで、InstallScript カスタム アクションでも使用できます。

詳細については、「SetObjectPermissions」を参照してください。

COM 抽出機能の強化

InstallShield では、COM 抽出時に新しい監視方式をサポートします。Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で、InstallShield を使用している場合、この新しい方式がデフォルトとなります。この方法は、カーネル ドライバーを使って、ビルド時のダイナミック COM 抽出中、およびデザイン時のスタティック COM 抽出中に変更されたレジストリ領域を監視します。この新しい方式は、DLL が既存のレジストリ エントリを読み込んでビルド マシンへの変更を妨げる以前の方法の利点を組み合わせたものです。

必要な場合、UseAPIRegistryHooks レジストリ値 (32 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥RegSpy レジストリ キーに含まれる、64 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥InstallShield¥RegSpy レジストリ キーに含まれる) の値データを設定して 3 つの異なる COM 抽出方式を切り替えることができます。使用可能な REG_DWORD 値データ:

- 0—API フックを使って、既存 DLLのレジストリ エントリを読み取ります。
- 1—レジストリのリダイレクトを使って、ビルド マシン上の登録済み DLL への変更を防止します。値を設定しなかった場合、これが Windows XP および Windows Server 2003 システム上でのデフォルト動作となります。
- 2—新しいカーネル モードの監視を使って、2 つのメソッドの両方の利点を組み合わせます。値が設定されていない場合、これが Windows Vista 以降および Windows Server 2008 以降のシステム上でのデフォルト動作となります。

この機能は、基本の MSI、DIM、InstallScript MSI、および マージ モジュール プロジェクト タイプに適用します。

Adobe Reader 10、Internet Explorer 9、および Microsoft Office 用の定義済みシステム検索

InstallShield に新しい定義済みシステム検索が追加されました:

- Adobe Reader 10
- Internet Explorer 9
- Microsoft Office 2010
- Microsoft Office 2007
- Microsoft Office 2003

インストールでこれらの 1 つ以上が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

ソフトウェア識別タグ機能のサポート

ISO/IEC 19770-2 は、ソフトウェア識別タグを作成するための国際規格です。ソフトウェア識別タグは、製品名、製品エディション、パブリッシャーなど、ソフトウェアに関する説明的な情報を含む XML ベースのファイルです。ソフトウェア資産管理ツールは、企業でインストールされているソフトウェアについての正確なアプリケーション ID を提供する目的で、タグ内のデータを収集するツールです。

ソフトウェア識別タグ機能は、業界標準として現在進化しつつあるもので、この機能により、独立系ソフトウェア ベンダーは、ソフトウェア資産管理およびライセンス最適化イニシアチブに役立つ情報を提供する、より洗練されたアプリケーションを作成することができます。製品のインストール パッケージに識別タグを持たせることで、顧客は、インストールした製品の内部的使用状況を監視できるツールが使えるようになります。これにより、顧客がソフトウェア会社などから入手して精神のライセンスの数を管理および最適化することができるようになり、ライセンス契約内容に違反するリスクがなくなります。

InstallShield の [一般情報] ビューには、製品の識別タグを作成するために必要な情報を指定するための新しい設定がいくつかあります。また、ビルド時にタグを自動生成して、それをインストールに含めるかどうかを指定できる新しい "ソフトウェア識別タグの使用" 設定も追加されました。この設定のデフォルト値は [はい] です。

"ソフトウェア識別タグの使用" 設定で [はい] が選択されている時に、4 つの必須識別設定 ([一般情報] ビューの "一意な ID"、"一意な登録 ID"、"タグ作成者"、および "タグ作成者 ID" 設定) のうち最低 1 つの値が設定されていなかった場合、空白の設定ごとに、ビルド警告 -7235 が一回発生します。このビルド警告では、特定の必須タグが空白であるために、ソフトウェア識別タグが作成されず、インストールに含まれなかったことが通知されます。この警告を解決するには、各設定に適切な値を入力するか、または "ソフトウェア識別タグの使用" 設定で [いいえ] を選択します。

オートメーション インターフェイスには、新しいタグの設定が含まれています。ISWiProject オブジェクトに、プロジェクトで、ソフトウェア識別タグの作成を有効または無効にできる新しい EnableSwidtag プロパティが追加されました。また、[一般情報] ビューでも構成可能な様々なタグ関係の設定を構成できる新しい SwidtagProperty プロパティも追加されました。

この機能は、基本の MSI プロジェクトに適用します。

詳細については、次を参照してください。

- 製品のソフトウェア識別タグを含める
- [一般情報] ビューの設定
- ISWiProject オブジェクト

強化機能

InstallShield には、以下のような強化機能が搭載されています。

マージ モジュール プロジェクトに、IIS、テキスト ファイルの変更、および XML ファイルの変更のためのビルトイン サポートが追加されました

いくつかの既存のビューが、InstallShield のマージ モジュール プロジェクトでも使用可能になりました:

- **IIS 構成** – このビューでは、新しい IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を作成および管理することができます。
- **テキスト ファイルの変更** – このビューを使って、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。
- **XML ファイル変更** – このビューでは、実行時に変更する XML ファイルのノードとノード セットの参照を追加します。

マージ モジュール プロジェクトのこれらのビューでは、IIS を構成したり、テキスト ファイルの変更を指定したり、XML ファイルの変更を指定したりできます。インストール プロジェクトでこれらの変更を構成するために使用されていた手順が、マージ モジュール プロジェクトでもサポートされるようになりました。これらのビューで

プロジェクト情報を含むマージ モジュールをビルドし、そのマージ モジュールをインストール プロジェクトに追加して、そのインストール プロジェクトでリリースをビルドした場合、適切なランタイム サポートがインストールに含められます。

以前、これらのビューは、インストール プロジェクトでのみ使用可能でした。

詳しくは、次を参照してください:

- インターネット インフォメーション サービス
- テキスト ファイルの変更
- XML ファイルの変更

IIS 7.x の新しいアプリケーション プール ID オプション

新しい ApplicationPoolIdentity オプションは、[IIS 構成] ビューで構成されたアプリケーション プールの "識別" 設定で使用できます。アプリケーション プールのワーカプロセスを実行するために選択されたアプリケーション プールに一意な仮想 ID を使用する場合は、このオプションを選択します。このオプションのサポートは、IIS 7 以降が搭載されたターゲット システムで使用できます。IIS 6 が搭載されているシステム上で実行されるインストールでこの新しいオプションが選択された場合、アプリケーション プールの ID には NetworkService アカウントが代わりに使用されます。

この機能は、基本の MSI、InstallScript、InstallScript MSI、および マージ モジュール プロジェクト タイプに適用します。

詳細については、「アプリケーション プールの設定」を参照してください。

オートメーション インターフェイスの機能強化: 新しく追加された Setup.exe の必須実行レベルを指定するための RequiredExecutionLevel プロパティ

ISWiRelease オブジェクト に読み取り RequiredExecutionLevel プロパティが追加されました。このプロパティは、[リリース] ビューにあるリリースで開かれている [Setup.exe] タブにある "必要実行レベル" 設定に対応しています。このプロパティは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

InstallShield 2011 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

InstallScript における Unicode サポート

InstallShield を使って、ランタイム文字列、ファイル、パス、レジストリ エントリ、その他のインストール データで Unicode を使用できる InstallScript インストールおよび InstallScript カスタム アクションを作成できます。さらに、InstallScript コンパイラおよび InstallScript エンジンでは、今回より、スクリプト コード外部に実装された関数に Unicode 文字列へのポインターを渡すことができます。また、スクリプト コード外部に渡される構造体に Unicode 文字列を格納することもできます。そのほか、InstallScript にはモダン言語の多言語インストールを完全にサポートするための機能が追加されています。

その結果、すべての言語は、そのサポートがインストールされているシステム上で正しく表示されます。エンド ユーザーは、システム上で Unicode 対応ではないプログラム用に使用される言語とインストールに使用される言語とを一致させる必要がなくなりました。ただし、ターゲット システムには、フォントをインストールする必要

があります。Windows の一部のバージョンでは、そのフォントがデフォルトでインストールされない言語もあります。たとえば、Windows XP 英語版システムでは、日本語のフォントがデフォルトではインストールされません。その場合、インストールが日本語の文字を使用できるように、フォントをインストールする必要があります。

InstallScript および InstallScript MSI プロジェクトにおける Unicode セットアップランチャー

今回より、InstallShield ではすべての Setup.exe と Update.exe ファイルが Unicode でビルドされます。以前、InstallScript および InstallScript MSI プロジェクトでは、すべての Setup.exe と Update.exe ファイルが ANSI でビルドされました。

Unicode セットアップランチャーは、ターゲット システムで適切なコード ページが実行されているいないにかかわらず、セットアップランチャーのユーザー インターフェイスで文字を正しく表示することができます。ANSI セットアップランチャーは、ターゲット システムで適切なコード ページが実行されている場合のみ、セットアップランチャーのダイアログで文字を正しく表示します。しかし、ターゲット システムで適切なコード ページが実行されていない場合、文字化けすることがありました。

エンド ユーザーは、ターゲット システムの言語にかかわらず、Setup.exe および Update.exe ファイルを Unicode パス内部から起動することができます。たとえば、今回より、エンド ユーザーは、英語版システム上で C:¥Users¥*日本語文字*¥Desktop¥Setup.exe からインストールを起動することができます。以前、このインストールは失敗しました。

ファイル、フォルダー、レジストリ エントリ、およびサポート ファイルにおける Unicode サポート

インストール ランタイムの主要な部分に Unicode サポートが追加されました。これによって、ファイル名、フォルダー名、レジストリ エントリ、およびサポート ファイル名に任意の言語の文字を同時に使用できるようになりました。たとえば、ファイル名またはターゲット パスに日本語文字が含まれるファイルを、英語版システムにインストールすることが可能です。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。

ポインター サポート

InstallScript エンジンとコンパイラーは、今回より、WPOINTER と呼ばれる新しいポインターをサポートします。対応する名前、wpointer および LPWSTR も使用可能です。たとえばパラメーターで Unicode 文字列へのポインターを受け付ける DLL 関数がある場合、この新しい種類のポインターを使うことができます。実行時にスクリプトで DLL 関数が呼び出されると、InstallScript エンジンはポインターを ANSI バージョンではなく、文字列の Unicode コピーに渡します。以前、ポインターは文字列の ANSI コピーにのみ渡すことができました。

InstallShield の以前のバージョンで使用される WSTRING データ タイプを使って Unicode 文字列を渡す機能は、引き続きサポートされています。

詳細については、次を参照してください。

- ポインター
- データ型および定義済み構造

構造体のサポート

InstallScript の構造体には、文字列、ポインター、その他の構造体をはじめとする任意の基本データ タイプを含めることができます。構造体に Unicode 文字列を含む必要があり、その構造体が外部 DLL に渡される場合、InstallScript エンジンはその構造内の文字列メンバー タイプを区別して、構造体サイズとメンバー オフセットを正しく計算します。Unicode として保存して渡す必要がある文字列メンバーは、WSTRING タイプを使って宣言できます。

以前、構造体に Unicode 文字列を含む必要があり、その構造が外部 DLL に渡される場合、InstallScript エンジンはその構造内に含まれる文字列がすべて ANSI であると想定しました。その結果、構造体のサイズとメンバーのオフセットに誤りが生じることがあり、DLL がその構造体に関連するデータの読み取りまたは書き込みを正しく行いませんでした。構造体の文字列メンバーに WSTRING を使用しても、効果がありませんでした。

既存の文字列を STRING タイプのままに残すことができます。InstallScript エンジンは、スクリプト コード外部で文字列を渡すとき、引き続きこれらを ANSI 文字列として処理します。

構造体のポインター メンバーも、今回より、WPOINTER として宣言できます。これによって、構造体で Unicode 文字列へのポインターを格納できます。

詳細については、次を参照してください。

- データ構造体
- データ型および定義済み構造

文字列テーブルのサポート

InstallShieldは、今回より、InstallScript プロジェクトの文字列テーブルをビルドする際に Unicode エンコードを使用します。このサポートにより、今回より、InstallScript プロジェクトの文字列テーブルに複数言語を含めることが可能となり、それらの言語がターゲット システムのコードページ設定に影響を受けることがなくなりました。さらに今回より、文字列テーブルに、ヒンディー語のようなコード ページを持たない言語の文字列を含めることができます。

インストールのユーザー インターフェイスに表示されるすべての文字列を、文字列テーブルに格納することが推奨されます。InstallScript コード内に直接文字列を書き込むこともできますが、その場合 Unicode として格納されません。したがって、正しいコード ページが設定されたシステム上でインストールが実行された場合にのみ、言語が正しく表示されます。文字列テーブルに文字列を格納して、InstallScript コードからそれらを参照することで、この問題を回避することができます。

詳細については、次を参照してください。

- InstallShield で文字列エントリを使用する
- 文字列定数演算子 (@)

Unicode を使った InstallScript ダイアログ

InstallScript ダイアログは、Unicode サポートを含みます。これによって、たとえば InstallScript ダイアログで日本語とドイツ語、またはロシア語とポーランド語を同時に使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。

この機能は、InstallScript および InstallScript MSI プロジェクト タイプで使用できます。

InstallScript デバッガーにおける Unicode サポート

InstallScript デバッガーは、今回より Unicode をサポートします。たとえば、szMsg = @ID_MSG という InstallScript コードの行をデバッグするとき、今回より、使用中のオペレーティング システム、または文字列に使用している言語にかかわらず、szMsg の値が表示されます。ID_MSG の値に中国語文字、または複数の言語からの文字が含まれている場合、InstallScript デバッガーのあらゆる領域 (スクリプト ウィンドウ内の変数に対して表示される変数ウィンドウ、ウォッチ ウィンドウ、およびツールヒント) で、疑問符の代わりに適切な文字が表示されます。

InstallScript デバッガについての情報は、*InstallScript デバッガー ユーザー ガイド*の「InstallScript ベース インストールのデバッグ」を参照してください。

InstallShield キャビネット ビューアーおよびログ ファイル ビューアーにおける Unicode サポート

InstallShield キャビネット ビューアーおよび InstallShield ログ ファイル ビューアーが、新しく InstallShield キャビネット & ログ ファイル ビューアー という 1 つのツールにまとまりました。この新しいツールを使って、InstallScript キャビネット ファイル (.cab)、InstallScript ヘッダー ファイル (.hdr) および InstallScript ログ ファイル (.ilg) を開いて参照することができます。このツールには Unicode サポートが含まれているため、使用中のオペレーティング システムの言語や文字列に使用されている言語にかかわらず、各ファイル名、レジストリ キー、ショートカット、および .cab、.hdr、または .ilg ファイルに含まれるその他のデータを正しく表示することができます。以前、一部の状況において一部の文字が疑問符で表示される場合がありました。

キャビネットおよびヘッダー ファイル サポートは、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクト タイプに適用します。

ログ ファイル サポートは、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクト タイプに適用します。 InstallScript および InstallScript MSI プロジェクト

このツールについては、次を参照してください:

- InstallShield キャビネット & ログ ファイル ビューアー
- InstallScript キャビネット ファイル (.cab) およびヘッダー ファイル (.hdr) を表示する
- InstallScript ログ ファイル (.ilg) を参照する

Team Foundation Server (TFS) との統合

InstallShield では、Team Foundation Server (TFS) 2010 との統合サポートが強化されています。

Visual Studio 2010 内から InstallShield を使用する場合、Source Control Explorer にアクセスして、InstallShield プロジェクトを Team Foundation バージョン コントロールと統合し、InstallShield プロジェクトと Visual Studio ソリューションへの変更を管理することができます。

Team Foundation Build を使って、InstallShield プロジェクトと Visual Studio ソリューションを定期的、またはオンデマンドでコンパイル、テスト、およびデプロイすることもできます。インストールはソリューションがビルドされる度に、自動的に最新のソース ファイルで更新されます。

さらに、InstallShield と Visual Stuio が搭載された同じマシン上に Team Explorer をインストールした場合、Visual Studio で開かれている InstallShield プロジェクト内から Team Explorer を使用できます。これで、次のようなタスクを行うことができます:

- InstallShield プロジェクトでの作業中にソース管理エクスプローラーを使用。
- InstallShield プロジェクトと Visual Studio ソリューションのビルドを構成。
- 新しいビルドをキューに配置。
- InstallShield プロジェクトと Visual Studio ソリューションのバグやタスクなどの作業項目を追跡する。

詳細については、次を参照してください。

- Microsoft Visual Studio Team Foundation Server との統合
- InstallShield プロジェクトを Team Explorer に追加する

Microsoft SQL Server 2008 R2 サポート

InstallShield は、今回より、SSQL Server 2008 R2 上で SQL スクリプトを実行するためのサポートを含みます。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2008 R2 が含まれています。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。

SQL Server 2008 R2 Express、SQL Server Native Client、Visual C++ 2010 用の新しい InstallShield 前提条件およびその他の再配布可能ファイル

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる多くの InstallShield 前提条件が含まれています。

- Microsoft SQL Server 2008 R2 Express
- Microsoft SQL Server 2008 Native Client 10.00.2531
- Microsoft SQL Server Native Client 9.00.4035
- Microsoft SQL Server System CLR 型 10.00.2531
- Microsoft SQL Server 2008 Management Objects 10.00.2531
- Microsoft Visual C++ 2010 再配布可能パッケージ
- Microsoft Visual C++ 2008 SP1 Redistributable Package
- Microsoft Visual C++ 2005 SP1 再配布可能パッケージ (x64)
- Windows Installer 3.1 – 日本語
- MSXML 6.0 SP1 – 日本語
- Microsoft .NET Framework 4.0 Client Language Pack – 日本語
- Microsoft .NET Framework 4.0 Full Language Pack – 日本語

InstallScript コード、SQL スクリプト、VBScript カスタム アクション、および JScript カスタム アクションの作成および変更を行うスクリプト エディターの強化

InstallShield のいくつかのビューで、プロジェクトのコードを書き込むためのスクリプト エディターが強化されました。スクリプト エディターは、以下の点で強化されています:

- **オートコンプリート機能の拡張** – スクリプト エディターに文字を入力するとき、InstallShield はその文字で始まる関数、キーワード、および定数をアルファベット順にリストするポップアップを表示します。用語全体を手入力する代りに、そのポップアップ リストから選択すると、InstallShield によってその用語がスクリプトに追加されます。

  [InstallScript] ビューでスクリプト エディターを使用する場合、文字列定数演算子 (@) を入力すると、使用可能な文字列識別子のポップアップ リストが表示されます。リストから適切な文字列識別子を選択すると、InstallShield がそれをスクリプトに追加します。

  ローカル変数のオートコンプリート機能も有効な場合、[InstallScript] ビューのスクリプト エディターで表示されるポップアップ リストには、ローカル変数も含まれています。

  オートコンプリートを使うと、コードを手入力する手間が省けるため、作業効率が上がります。また、コードのスペル ミスを回避することができます。

  InstallShield の以前のバージョンでは、[InstallScript] ビューのスクリプト エディターでのみオートコンプリートがサポートされていました。そのため、オートコンプリート機能は SQL スクリプト、VBScript コード、または JScript コードでは使用できませんでした。さらに、ポップアップ リストも InstallScript 関数に限られており、キーワード、定数、ローカル変数、または文字列 ID はオートコンプリートで使用できませんでした。

- **詳細な InstallScript 関数呼び出しのヒント** – 関数呼び出しのヒントが有効な場合、[InstallScript] ビューでスクリプトに関数の呼び出しを入力しているときに、InstallScript 関数呼び出しのヒント (一種のツールヒント) が表示されます。関数呼び出しのヒントは、ビルトイン関数のパラメーター情報を表示します。その他、ビルトイン関数の説明や、入力しているパラメーターについての説明も表示します。詳細な呼び出しのヒントを使うと、スクリプト エディターからヘルプ ライブラリへ、そしてまたスクリプト エディターへと切り替える必要がなく、関数に関するヘルプ情報を表示できます。

  InstallShield の以前のバージョンの [InstallScript] ビューでも、呼び出しのヒントをサポートしますが、その情報量が限られています。呼び出しのヒントには、関数呼び出しとその関数のすべてのパラメーターが表示されましたが、呼び出しのヒントには、関数の説明、または入力しているパラメーターの説明は含まれませんでした。

- **構文の折りたたみ機能** – InstallShield の様々なビューで、スクリプト エディターの構文の折りたたみ機能をサポートするかどうかを指定できます。構文の折りたたみ機能が有効な場合、スクリプトで展開可能または折りたたみ可能なブロックで始まるコードの各行の横のマージンに、プラス (+) またはマイナス (–) 記号が追加されます。プラス記号をクリックして非表示となっているコードを展開したり、マイナス記号をクリックして表示されているコードを非表示にしたりできます。

  構文の折りたたみ機能を使って、長いスクリプトを縮小することで、現在行っている作業に関連のあるコードに焦点を当てることができます。また、スクリプトの全体的な構造を確認するのに便利です。

強化されたスクリプト エディターを含むビューは、[InstallScript] ビュー、[SQL スクリプト] ビュー、および [カスタム アクションとシーケンス] ビュー (このビューで VBScript または JScript ファイルを参照する場合) です。

デフォルトで、スクリプト エディタにおける構文の折りたたみ機能は無効になっていますが、オートコンプリート、ローカル変数のオートコンプリート、および関数呼び出しのヒントは有効です。これらの機能を有効または無効にするには、[スクリプト エディタのプロパティ] ダイアログ ボックスを使います。このダイアログを使って、フォント、構文の色、行番号などのスクリプト エディターのその他の設定も変更できます。このダイアログ ボックスにアクセスするには、スクリプト エディター内を右クリックしてから、[プロパティ] を選択します。

詳細は、次を参照してください:

- 様々なビューで、[スクリプト エディター] ペインを使って作業する
- スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う
- スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する
- スクリプト エディターで構文ハイライトの色を変更する
- スクリプト エディターで構文の折りたたみ機能を有効または無効にする
- スクリプト エディターのキーボード ショートカット
- [スクリプト エディターのプロパティ] ダイアログ ボックス

64 ビットの Microsoft App-V パッケージを作成できる機能

InstallShield の Microsoft App-V アシスタントには、64 ビットの Windows Installer パッケージを 64 ビットの Microsoft App-V パッケージに変換するためのサポートが搭載されています。64 ビットの Microsoft App-V パッケージは、64 ビットの Microsoft App-V 4.6 Client を搭載した 64 ビット Windows システム上に展開することができます。リパッケージャーに追加された新しい 64 ビット リパッケージ サポートとこの機能を組み合わせて、任意の 64 ビット インストールを 64 ビット App-V パッケージに変換することができます。

64 ビット Windows Installer パッケージを App-V パッケージに変換する場合は、64 ビット Windows システム上で行うことをお勧めします。32 ビット システム上で変換しようとすると、64 ビット バイナリ ファイルの COM 情報抽出エラーが発生する場合があります。また、一部の状況において、Windows Installer パッケージはパッケージそ

のものに含まれていない実行可能ファイルをターゲットとするショートカットを含みます。ショートカットが、64 ビットの場所にある実行可能ファイルをターゲットとする場合、これらのショートカットは、32 ビット システム上で正しく処理されません。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

InstallShield 前提条件の検索パスを指定できる機能 (前提条件ファイルのソースの場所におけるパス変数および相対パスのサポート)

InstallShield では、今回より、InstallShield 前提条件ファイル (.prq ファイル) を検索するフォルダを指定できます。この機能によって、複数の開発者の間で InstallShield 前提条件を共有し、それらをソース コード管理システムで保管することが容易になります。以前は、InstallShield が .prq ファイルを検索するのは、*InstallShield Program Files フォルダー*¥SetupPrerequisites のみでした。

InstallShield でフォルダーを指定するとき、いくつかの方法があります:

- InstallShield 内部から編集またはビルドを行う場合、[ツール] メニューで [オプション] をクリックすると表示される [オプション] ボックスにある新しい [前提条件] タブを使って、マシン共通および現在のユーザーのフォルダをコンマ区切りのリストで指定できます。このタブは、マージ モジュールの検索パスを指定できる、[オプション] ダイアログ ボックスの [マージ モジュール] タブと似ています。
- ISCmdBld.exe を使ってコマンドラインからビルドを行う場合、新しい -prqpath パラメーターを使って、コンマ区切りのフォルダーのリストを指定できます。

  .ini ファイルを使って ISCmdBld.exe パラメーターを指定する場合、.ini ファイルの [Mode] セクションで新しい PrerequisitePath パラメーターを使用して、フォルダのコンマ区切りのリストを指定できます。
- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで新しい PrerequisitePath パラメーターを使います。このパラメーターは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPrerequisitePath として露出されます。複数のパスを指定するには、順序指定されたパスの配列を使用します。

InstallShield 前提条件エディターの [含めるファイル] タブを使って InstallShield 前提条件にファイルを追加すると、エディターは今回より、適切な場合に <WindowsFolder> および <ISProductFolder> などの定義済みパス変数を使用します。また、追加するファイルが InstallShield 前提条件の .prq ファイルと同じフォルダ (または .prq ファイルが含まれているフォルダのサブフォルダ) に格納されている場合、InstallShield 前提条件エディターは .prq ファイル内のファイルに相対パスを使用します。[含めるファイル] タブでファイルのパスを参照したとき、InstallShield 前提条件エディターは相対パスではなく、完全パスをリストします。

InstallShield 前提条件エディターで [名前を付けて保存] コマンドを使用して .prq ファイルの場所を変更すると、InstallShield 前提条件エディターは、適切な場合に前提条件のファイルのパスを更新します。

詳しくは、次を参照してください:

- InstallShield 前提条件を含むディレクトリを指定する
- [前提条件] タブ
- ISCmdBld.exe
- .ini ファイルでコマンドライン ビルド パラメーターを渡す
- Microsoft ビルド エンジン (MSBuild)

Windows Installer ベースのダイアログにおける新しいハイパーリンク コントロール

コントロール ツールバーに、Windows Installer ベースのダイアログで [ダイアログ] ビューに使用できる、新しいハイパーリンク コントロールが追加されました。ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。

Windows Installer 5 は、この新しいハイパーリンク コントロールをサポートします。このコントロールを Windows Installer の以前のバージョンを持つシステムで表示されるダイアログに使用すると、ランタイム エラー 2885 が発生して、インストールが中止されます。このため、ダイアログ上でハイパーリンク コントロールを使用するとき、インストールが Windows Installer 4.5 以前をターゲットとする場合、プロジェクトにハイパーリンク コントロールを含むバージョンと、含まないバージョンの 2 種類のダイアログを含めることをお勧めします。ダイアログに条件を追加して、ターゲット マシンの Windows Installer バージョンによってダイアログを表示または非表示とします。

この機能は、基本の MSI およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、「ハイパーリンク コントロール」を参照してください。

Setup.exe および Update.exe においてカスタム アイコンとカスタム バージョン リソースのプロパティを指定できる機能

InstallShield では、今回より、ビルド時に作成する Setup.exe ファイルでカスタム アイコンおよびカスタムバージョン リソースのプロパティを使用できます。アイコンとバージョン リソースのプロパティは、Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。エンド ユーザーが Windows Explorer で Setup.exe ファイルを参照したときにも、このアイコンが表示されます。このサポートは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

今回より、同じ機能 (カスタム アイコンとカスタム バージョン リソースのプロパティを指定できる機能) が、基本の MSI、InstallScript MSI、および QuickPatch プロジェクトで作成された Update.exe ファイルでも使用できます。

Setup.exe および Update.exe のカスタム アイコン

[リリース] ビューの [Setup.exe] タブには、Setup.exe セットアップランチャーに使用するアイコンを指定できる "Setup.exe アイコン ファイル" 設定があります。アイコンには、.exe、.dll、または .ico ファイルを使用できます。アイコンを指定しなかった場合、InstallShield は Setup.exe ファイルのデフォルト アイコンを使用します。

以前は、ビルド中の Setup.exe ファイルが自己展開型の単一セットアップランチャーの場合に、InstallScript プロジェクトでカスタム Setup.exe アイコンの指定がサポートされていました。これは、基本の MSI または InstallScript MSI プロジェクトではサポートされていませんでした。また、圧縮された InstallScript インストールにおいてもサポートされていませんでした。

新しいアイコン設定を使って、Update.exe ランチャーに使用するアイコンを指定できます。この設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビュー内にあるパッチの構成の [詳細] タブで行います。また、QuickPatch プロジェクトの場合、[一般情報] ビューの [ビルドの設定] 領域にある [詳細] タブで設定できます。アイコンを指定しなかった場合、InstallShield は Update.exe ファイルのデフォルト アイコンを使用します。

以前は、InstallShield は Update.exe ファイルのカスタム アイコンの指定を一切サポートしませんでした。

Setup.exe および Update.exe のカスタム バージョン リソースのプロパティ

InstallShield がビルド時に Setup.exe ランチャーの以下のバージョン リソースを構成するとき、今回より、[一般情報] ビューおよび [リリース] ビューで入力されたカスタム情報を使用します。

- 会社名

- 製品名
- 製品バージョン
- 著作権情報
- ファイル バージョン
- ファイルの説明

カスタム著作権情報およびカスタム ファイルの説明を使用する場合、[リリース] ビューで "カスタム バージョンのプロパティを使用する" 設定で [はい] を選択してください。

以前、InstallShield は多くの場合にカスタム情報を使用しませんでした。たとえば、InstallShield が以前に作成した InstallScript Setup.exe ファイルには、その Setup.exe ファイルをビルドした InstallShield のバージョン固有の詳細が含まれました。そのため、Setup.exe プロパティ ダイアログ ボックスに表示される製品名は、インストールする製品の名前ではなく、InstallShield と表示されました。

"会社名"、"製品名"、"製品バージョン"、"説明"、"著作権情報" といった新しい設定を使って、InstallShield で Update.exe ファイルをビルドするときに使用するカスタム情報を指定できます。これらの設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビュー内にあるパッチの構成の [詳細] タブで行います。また、QuickPatch プロジェクトの場合、[一般情報] ビューの [ビルドの設定] 領域にある [詳細] タブで設定できます。

以前、InstallShield は Update.exe ファイルにカスタム バージョン リソース情報を使用しませんでした。

詳しくは、次を参照してください:

- セットアップランチャーのファイルのアイコンを指定する
- セットアップランチャーのファイルのプロパティをカスタマイズする
- リリースの [Setup.exe] タブ
- アップデート起動ツールのアイコンを指定する
- アップデート起動ツールのファイルのプロパティをカスタマイズする
- [詳細] タブ ([パッチのデザイン] ビュー)
- [詳細] タブ (QuickPatch プロジェクト)

ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する機能

InstallShield Premier Edition には、ビルド プロセスの様々な段階で実行するコマンドを指定することができる、新しいリリースの設定が追加されました。これらの新しい設定は、[リリース] ビューでリリースを選択したときに、新しい [イベント] タブに表示されます。

- **ビルド前のイベント** – この設定を使って、InstallShield がリリースのビルドを開始する前に実行するコマンドを指定します。このイベントは InstallShield がリリース フォルダとログ ファイルを作成した後、リリースのビルドを開始する前に実行します。

  この設定は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで使用できます。

- **圧縮前のイベント** – この設定を使って、製品のデータ ファイルを .cab ファイルに格納する場合、InstallShield が .msi パッケージと .cab ファイルをビルドした後に実行するコマンドを指定します。このイベントは .cab ファイルが .msi パッケージにストリームされた後、.msi パッケージにデジタル署名が行われて Setup.exe ファイルにストリームされる前に発生します。

  この設定は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

- **ビルド後のイベント** – この設定を使って、InstallShield がリリースをビルドして署名を行った後に実行するコマンドを指定します。

  この設定は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで使用できます。

新しい [イベント] タブは、以前の [ポストビルド] タブに取って代わります。以前 [ポストビルド] タブで提供されていた設定は、新しい [イベント] タブに表示されます。また、以前マージ モジュール プロジェクトの [ビルド] タブにあったパブリッシュ関連の設定は [イベント] タブに移動しました。

オートメーション インターフェイスは、この新しいビルド イベントの設定をサポートします。ISWiRelease オブジェクトには、ビルド プロセスの様々な段階におけるコマンドを指定できる 3 つの新しいプロパティが含まれてます。

- PrebuildEvent
- PrecompressionEvent
- PostbuildEvent

詳しくは、次を参照してください:

- ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する
- リリースの [イベント] タブ
- ISWiRelease オブジェクト

Setup.exe に有効期限日を設定する機能

InstallShield では、Setup.exe の有効期限日と有効期限切れメッセージを設定できます。エンド ユーザーが Setup.exe をプロジェクトで指定された日付以降に実行すると、有効期限切れメッセージが表示されて、インストールが終了します。

Setup.exe ファイルの有効期限日とメッセージの設定は、[リリース] ビューで選択されたリリースの [Setup.exe] タブに新しく追加された "有効期限日と有効期限メッセージ" 設定で行います。デフォルトで、Setup.exe に有効期限日は設定されていません。有効期限日を指定すると、デフォルトの有効期限切れメッセージまたは独自のカスタムメッセージを使用できます。

オートメーション インターフェイスには、これらの新しい設定のサポートが含まれています。ISWiRelease オブジェクトには、有効期限日とメッセージを設定できる 2 つの新しいプロパティ (ExpirationDate および ExpirationMessage) が含まれています。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- リリースの [Setup.exe] タブ
- ISWiRelease オブジェクト

Visual Studio セットアップとマージ モジュール プロジェクトを既存の InstallShield プロジェクトにインポートする機能 (プロジェクト コンバーターの強化)

InstallShield では、今回より、Visual Studio セットアップ プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートするか、Visual Studio マージ モジュール プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートすることができます。これらのタスクを行うと、Visual Studio プロジェクトに含まれている同じデータと設定を含む InstallShield イ

ンストール プロジェクトおよびマージ モジュール プロジェクトを開発できます。ウィザードを使って、プロジェクト出力、ファイル、レジストリ キー、ファイル拡張子、カスタム アクション、ターゲット システム検索、および起動条件を Visual Studio プロジェクトから既存の InstallShield プロジェクトにインポートします。

Visual Studio プロジェクトを既存の InstallShield プロジェクトにインポートするには、InstallShield の Visual Studio デプロイメント プロジェクト インポート ウィザードを使います。このウィザードでは、Visual Studio プロジェクト内の特定の設定をインポートするか、無視するかを選択することができます。

Visual Studio プロジェクトを新しい InstallShield プロジェクトに変換するための、これまでのサポートが拡張されました。Visual Studio プロジェクトに定義済み前提条件が含まれている場合、InstallShield は今回より、プロジェクトの変換中にそれらを対応する InstallShield 前提条件に変換します。これと同じ前提条件の変換機能が、Visual Studio プロジェクトを InstallShield プロジェクトにインポートするための新しいウィザードにも搭載されています。

Visual Studio プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合は、変換プロセスの代わりにインポート ウィザードを使用します。InstallShield プロジェクトは、Visual Studio セットアップまたはマージ モジュール プロジェクトおよびそのプロジェクト依存関係が含まれているのと同じ Visual Studio ソリューションに含まれていなくてはなりません。インポート ウィザードを使ってプロジェクト出力を InstallShield プロジェクトにインポートするためには、InstallShield を Visual Studio 内部で開いてください。

詳しくは、次を参照してください:

- Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする
- Visual Studio デプロイメント プロジェクト インポート ウィザード

SQL スクリプトにおける Unicode および UTF-8 サポート

InstallShield は、Unicode BOM エンコードまたは UTF-8 BOM エンコードのいずれかを使った SQL スクリプトをデザイン時および実行時にサポートします。[SQL スクリプト] ビューを使って、どちらかのエンコードを使った SQL スクリプトをプロジェクトに追加できます。これらの SQL スクリプトを、必要に応じて [SQL スクリプト] ビュー内から編集することもできます。実行時に、必要に応じてインストールおよびアンインストール中に SQL スクリプトが実行されます。

以前、InstallShield では Unicode BOM エンコード を使った SQL スクリプトが実行時にはサポートされていましたが、デザイン時にはサポートされていませんでした。したがって、このエンコードを使ったスクリプトをプロジェクトの [SQL スクリプト] ビューに追加すると、InstallShield がスクリプトを ANSI 形式に変換するかどうかを指定するためのプロンプトが表示されました。InstallShield で ANSI への変換を許可すると、それを [SQL スクリプト] ビュー内から編集できます。ただし、一部の状況において、デザイン時と実行時に文字化けが発生することがありました。InstallShield が ANSI に変換することを許可しなかった場合、そのファイルは Unicode BOM エンコードのままとなりました。このエンコードを使うと、インストールは実行時にターゲット システムのコード ページと一致しない言語の文字列を正しく使用しますが、InstallShield 内部からスクリプトを参照または編集することができませんでした。

さらに以前、InstallShield は UTF-8 BOM エンコードを使用する SQL スクリプトを適切にサポートしませんでした。このエンコード タイプを使用する SQL スクリプトをプロジェクトに追加して、スクリプトに開発システムのコード ページと一致しない言語の文字列が含まれていた場合、[SQL スクリプト] ビューは、そのファイルを ANSI エンコードとして処理するために、文字化けが発生することがありました。また、実行時に SQL スクリプトが実行されると、文字化けが発生しました。バイト オーダー マークも文字化けして表示されました。

[SQL スクリプト] ビュー内から新しい SQL スクリプトを作成すると、InstallShield はそのファイルに Unicode BOM エンコードを使用します。[SQL スクリプト] ビューで ANSI または UTF-8 BOM エンコードを使用したい場合、別のツールを使って適切なエンコードを用いて .sql ファイルを作成してから、プロジェクトの [SQL スクリプト] ビューにそのスクリプトをインポートまたは挿入することをお勧めします。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。

SQL Server 2008 Express SP1 および Adobe Reader 9 のための定義済みシステム検索

InstallShield に新しい定義済みシステム検索が追加されました:

- SQL Server 2008 Express SP1
- Adobe Reader 9

インストールでこれらの 1 つまたは両方が必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

64 ビット マネージコード カスタム アクションのサポート

InstallShield は、今回より、64 ビット マネージコード カスタム アクションをサポートします。プロジェクトでマネージコード カスタム アクションを含むリリースをビルドするとき、InstallShield はカスタム アクションと関連付けられたメインの .NET アセンブリのターゲット アーキテクチャ (32 ビットまたは 64 ビット) を判別しようとします。InstallShield は、実行時にマネージコードを実行するときに .NET Framework の適切なバージョン (32 ビットまたは 64 ビット) が使用されるようにリリースを構成します。

新しいデフォルトの動作をオーバーライドするには、[ダイレクト エディター] ビューを使って、次のフィールドを含む新しいレコードを ISClrWrap テーブルに追加します。

- Action_ — 変更するマネージコード カスタム アクションの名前を示します。
- Name — 次を入力します: **TargetPlatform**
- Value — 適切なアーキテクチャを指定します。.NET Framework の 32 ビット バージョンを使用するには、次を入力します:**x86**

  .NET Framework の 64 ビット バージョンを使用するには、次を入力します:**x64**

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、「32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い」を参照してください。

64 ビット .NET Installer クラスと COM Interop のサポート

InstallShield は今回より、64 ビット .NET Installer クラスと COM Interop をサポートします。64 ビット システム上で InstallShield を使用している場合、InstallShield の [ツール] メニューにある [オプション] をクリックすると表示される [オプション] ボックスでは、今回より、.NET Framework に含まれている Regasm.exe および InstallUtilLib.dll ファイルの場所として、32 ビット と 64 ビットの 2 つのパスを指定できます。InstallShield は、.NET Installer クラスと COM Interop を含むリリースで、ビルド時に指定されたパスを使用します。

ISCmdBld.exe を使ってコマンドラインからビルドしている場合に、既存の -t パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ISCmdBld.exe は今回より、適切な場合に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

MSBuild または Team Foundation Server (TFS) を使ってビルドしている場合に、InstallShield タスクで既存の DotNetUtilPath パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ビルドは今回より、適切な場合に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

詳細は、次を参照してください:

- [.NET] タブ ([オプション] ダイアログ ボックス)
- ISCmdBld.exe
- Microsoft ビルド エンジン (MSBuild)

.NET Framework 4 をターゲットにする DLL で InstallScript 関数 DotNetCoCreateObject またはマネージ コード カスタム アクションの使用をサポート

Visual Studio 2010 で作成した DLL が .NET Framework 4 を使用する場合に、InstallScript 関数 **DotNetCoCreateObject** を使って、この DLL で関数を呼び出すことができます。以前、DLL が .NET Framework の以前のバージョンを使用したときにはインストールはクラッシュしませんでしたが、バージョン 4 を使用したときに、インストールがクラッシュしました。この機能は、次のプロジェクト タイプで使用できます: この機能は InstallScript と InstallScript MSI プロジェクトで使用できます。またこの機能は、InstallScript カスタム アクションを含む基本の MSI およびマージ モジュール プロジェクトでも使用できます。

さらに、今回より、マネージコード カスタム アクションで同じような DLL ファイルを使用できます。以前は、インストールに .NET Framework のバージョン 4 を使用する DLL のマネージコード カスタム アクションが含まれている場合、クラッシュが起こりました。この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- DotNetCoCreateObject
- 32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い

リリースのパス変数の値をオーバーライドする機能

InstallShield では今回より、プロジェクト内の各リリースでプロジェクトのカスタムパス変数 (標準パス変数、環境パス変数、およびレジストリ パス変数) の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、この機能を使ってカスタム アクションのバイナリ ファイルを入れ替えることができます。個別の 32 ビットと 64 ビットのターゲット システム用に個別のリリースを作成した場合、カスタム アクションに選択された DLL を参照するパス変数をオーバーライドすることができます。これによって、InstallShield が 32 ビット リリースには 32 ビット DLL を、また 64 ビット リリースには 64 ビット DLL を含みます。インストールがインストール処理を行っているファイルを入れ替えるのにパス変数のオーバーライドを行うことはお勧めできません。これは、ファイルの 32 ビット バージョンと 64 ビット バージョンに個別のコンポーネントを使用しなくてはならないためです。

プロジェクトで 1 つ以上のパス変数をオーバーライドする場合、[リリース] ビュー内のリリースの [ビルド] タブに追加された "パス変数のオーバーライド" 設定を使います。新しい "パス変数のオーバーライド" 設定と、IsCmdBld.exe または MSBuild で -I パラメーターを使ってパス変数をオーバーライドした場合、コマンドラインまたは MSBuild 値がリリース設定で指定された値よりも優先します。

オートメーション インターフェイスには、この新しい設定のサポートが含まれています。ISWiRelease オブジェクトには、リリースに含まれるプロジェクトのユーザー定義のパス変数、環境変数、およびレジストリ変数の値をオーバーライドできる、新しい PathVariableOverrides プロパティが含まれています。

この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプで使用できます。

詳しくは、次を参照してください:

- リリースのカスタム パス変数の値をオーバーライドする
- "パス変数のオーバーライド" 設定
- [パス変数のオーバーライド] ダイアログ ボックス
- ISWiRelease オブジェクト

IIS Web サイト、アプリケーション、および仮想ディレクトリの MIME の種類を構成する機能

[IIS 構成] ビューに新しく追加された "MIME の種類" 設定を使って、プロジェクトに含まれる Web サイト、アプリケーション、または仮想ディレクトリの MIME の種類を構成できます。この設定を使って、ターゲット システム上の Web サーバーからブラウザーまたはメール クライアントに送信できるコンテンツの種類を指定できます。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、次を参照してください。

- Web サイトにおける MIME の種類設定
- アプリケーションまたは仮想ディレクトリにおける MIME の種類設定

既存の IIS アプリケーション プールを上書きする、またはそれが既存しない場合のみ作成する機能

[IIS 構成] ビューに、新しい "既存のアプリケーション プールを上書きする" 設定が追加されました。この設定は、[IIS 構成] ビューでアプリケーション プールを選択したときに、右側のペインに表示されます。この設定を使って、実行時にターゲット システム上に既に選択されたアプリケーション プールが存在する場合に起こる動作を指定できます。インストールによって、ターゲット システム上のアプリケーション プールの設定を上書きするか、アプリケーション プールをそのまま存続させることができます。この設定のデフォルト値は [はい] で、実行時に既存のアプリケーション プールが上書きされます。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、アプリケーション プールの「既存のアプリケーション プールを上書きする」に記述されている説明を参照してください。

InstallScript および InstallScript MSI プロジェクトにおける新しいビルボード スタイル

InstallScript および InstallScript MSI プロジェクトには、進行状況ダイアログの進行状況バーの上に表示されるビルボードの新しいスタイルがサポートされています。このビルボード スタイルは、イメージ ファイル (.bmp, .gif, .jpg, and .jpeg) だけでなく、Adobe Flash アプリケーション ファイル (.swf) もサポートします。

プロジェクトに Flash ファイルとイメージ ファイルの両方が含まれているとき、ターゲット システムに Adobe Flash Player がインストールされていない場合、インストールはこれを検出して Flash ビルボードの代わりにイメージ ビルボードを表示します。

以前、InstallScript および InstallScript MSI プロジェクトで使用できる唯一のビルボード サポートでは、背景ウィンドウを使用する必要がありました。新しいビルボード スタイルで、背景ウィンドウは不要です。

新しいビルボード スタイルをプロジェクトに追加するには、[サポート ファイル/ビルボード] ビューを使って、プロジェクトにビルボード ファイルを追加します。実行時に、この新しいスタイルのビルボードを表示するには、新しい STATUSBBRD 定数を **Enable** 関数と一緒に使います。

新しいスタイルのビルボードは、スキン ダイアログを使用するプロジェクトでは使用できません。

詳しくは、次を参照してください:

- InstallScript および InstallScript MSI インストールでビルボードを表示する
- InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類
- InstallScript または InstallScript MSI プロジェクトでコードを追加または変更してビルボードを表示する
- InstallScript または InstallScript MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する
- InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する
- InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前

基本の MSI プロジェクトにおけるイメージ ビルボードのループ サポート

基本の MSI プロジェクトの [ビルボード] ビューに新しく追加された "ビルボードのループ" 設定を使って、インストールがファイルの転送を完了するまでイメージ ビルボードを継続してループ表示してから適切な SetupComplete ダイアログを表示するかどうかを指定できます。

この設定で [はい] を選択して、ビルボードに割り当てた時間よりもファイルの転送に時間がかかった場合、インストールは最初のビルボードから再び表示します。必要な場合、ループはファイルの転送が終了するまで継続します。この設定のデフォルト値は [いいえ] です。これは、InstallShield の以前のバージョンの動作と同じです。

以前は、ビルボードに割り当てられた時間よりもファイルの転送に時間がかかった場合、インストールはファイル転送が終了するまで最後のビルボードを表示し続け、ビルボードをループしませんでした。

詳細については、次を参照してください。

- ビルボードを含む基本の MSI インストールにおける実行時の動作
- ビルボード設定

Windows Installer 5 における Windows サービスのアクセス許可の構成サポート

InstallShield は今回より、Windows サービスのアクセス許可の構成をサポートします。このサポートは Windows Installer 5 で提供されている MsiLockPermissionsEx テーブルを使用します。Windows Installer 5 は、新しいサービスのアクセス許可の設定をサポートします。以前の Windows Installer は、これらの設定を無視します。

新しいサービスのアクセス許可設定を構成するには、[セットアップのデザイン] ビュー (インストール プロジェクトの場合) または [コンポーネント] ビューで、コンポーネントの [詳細設定] 領域にある [サービス] ノードを使用します。[サービス] ノードで、プロジェクトに新しいサービスを追加します。[サービス] ノードの下にあるサブノードを選択すると、右側のペインでそのサービスの新しいアクセス許可と関連設定を構成できます。

この機能は、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- サービスの設定

InstallScript プロジェクトのダイナミック ファイル リンクの構成におけるオートメーション インターフェイス サポート

InstallScript プロジェクトで、オートメーション インターフェイスの ISWiDynamicFileLinking オブジェクトおよび ISWiDynamicFileLinkings コレクションに、ダイナミック ファイル リンク サポートが追加されました。さらに今回より、InstallScript プロジェクトで ISWiComponent オブジェクトの 2 つのメソッドとプロパティを使用できます: AddDynamicFileLinking メソッドは、コンポーネントに新しいダイナミック ファイル リンクを追加し、RemoveDynamicFileLinking メソッドは、コンポーネントのダイナミック ファイル リンクを削除し、ISWiDynamicFileLinkings プロパティは、ダイナミック ファイル リンクのコンポーネントのコレクションを取得します。ISWiFile オブジェクトの DynamicFile 読み取り専用プロパティを InstallScript プロジェクトでも使用できるようになりました: このプロパティを使って、ファイルのソースがコンポーネントにダイナミックにリンクされているか、スタティックにリンクされているかを判別します。

詳細については、次を参照してください。

- ISWiDynamicFileLinking オブジェクト
- ISWiDynamicFileLinkings コレクション
- ISWiComponent オブジェクト
- AddDynamicFileLinking メソッド
- RemoveDynamicFileLinking メソッド
- ISWiFile オブジェクト

新しい FlexNet Connect 12.01 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 12.01 をサポートします。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 12.01 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。

強化機能

InstallShield には、以下のような強化機能が搭載されています。

InstallShield の Unicode ビュー

InstallShield のいくつかのビューと領域は、すべての言語からの文字を表示および入力できるように強化されています。たとえば、今回より、英語版のコンピュータ上で [システム検索] ビューにおいてシステム検索を構成するときに、ファイル名、パス、レジストリ エントリに中国語の文字を使用できます。以前、InstallShield のこれらの領域で表示される文字は疑問符として表示されました。

InstallShield で Unicode サポートが強化された領域は、[システム検索] ビュー、および [リリース] ビュー内のリリースの [複数インスタンス] タブ、および [リリース] ビュー内のリリースに表示されるタブです。InstallShield 2010 より、その他の多くのビューで Unicode がサポートされています。

[システム検索] ビューの強化は、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトに適用します。

[リリース] ビュー内のリリースのタブにおける強化は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクトに適用します。

[リリース] ビューの [複数インスタンス] タブの強化は、基本の MSI プロジェクトに適用します。

InstallShield MSI Diff を使用中にログファイルを生成するためのコマンドライン サポート

InstallShield MSI Diff には、新しく /out コマンドライン パラメーターが含まれています。InstallShield MSI Diff を実行中に、コマンドラインからこの新しいパラメーターを使って、2 つの .msi、.msm、または .pcp ファイル間の差分を確認したり、トランスフォーム (.mst) またはパッチ ファイル (.msp) によって Windows Installer データベースに適用される変更点を表示するログ ファイルを生成できます。また、このツールをコマンドラインから使用して、バイナリ形式で保存されている 2 つの InstallShield プロジェクト ファイル (.ism または .ise) 間の違いを識別するログ ファイルを生成することもできます。

詳細については、「InstallShield MSI Diff をコマンドラインから実行して、差分のログ ファイルを生成する」を参照してください。

Microsoft SQL Server のローカル インスタンスを 64 ビット システムでリストするサポート

InstallShield の [SQL スクリプト] ビューを使ってプロジェクトに SQL サポートを追加してから、そのインストールを 64 ビット ターゲット システムで実行すると、SQL 関連のビルトイン ランタイム ダイアログは、今回より、32 ビットの SQL Server のローカル インスタンス、32 ビットの SQL Server のネットワーク インスタンス、および 64 ビットの SQL Server のネットワーク インスタンスだけでなく、64 ビットの SQL Server のローカル インスタンスもリストします。以前は、インストールが 64 ビット ターゲット システムで実行された場合、SQL 関連のビルトイン ランタイム ダイアログは 64 ビットの SQL Server のローカル インスタンスをリストせず、32 ビットのローカル インスタンス、32 ビットのネットワーク インスタンス、および 64 ビットのネットワーク インスタンスのみがリストされました。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで利用できます。

64 ビット レジストリの変更についての InstallScript ログ記録とアンインストール

今回より、64 ビット ターゲット システム上で InstallScript または InstallScript MSI インストールが行われるときに、デフォルトで InstallScript エンジンがレジストリの 64 ビット部分に行われる変更をログ記録します。さらに、InstallScript エンジンによってログ記録された 64 ビット レジストリの変更が、今回より、アンインストール中にアンインストールされます。

以前、InstallScript エンジンがインストール中にレジストリの 64 ビット部分に変更を加えた場合、これらの変更は 64 ビット特定の変更としてログ記録されませんでした。そのためアンインストール中に削除も行われませんでした。

Microsoft SQL Server Management Studio のサポート

[SQL スクリプト] ビューでは、今回より、Microsoft SQL Server Management Studio がサポートされています。プロジェクト内の SQL スクリプト ファイルを SQL Server Management Studio で開くには、[SQL スクリプト] ビューで SQL スクリプトを右クリックしてから、[Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。この新しいコマンドは、同じコンテキスト メニューにあった [Microsoft Query Analyzer でスクリプトを開く] の代わりとなります。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで利用できます。

Visual Studio Web セットアップ プロジェクトからプロジェクト出力を追加するサポート

Web セットアップ プロジェクトを含む Visual Studio ソリューションと、InstallShield インストール プロジェクトを作成した場合に、Visual Studio 内部から InstallShield を使用しているとき、今回より、Web セットアップ プロジェクトのプロジェクト出力を InstallShield プロジェクトに追加することができます。

テキスト ファイルの変更を構成するときに Windows Installer プロパティを選択できる機能

[テキスト ファイルの変更] ビュー内の 2 つの設定について、テキスト ファイルのコンテンツにおける検索置換動作をより簡単に構成できるように強化されています。[テキスト ファイルの変更] ビューで置換ノードを選択すると右側のペインに表示される "検索する文字列" 設定と "置換後の文字列" 設定では、今回より、リストから Windows Installer プロパティを選択するか、文字列を入力することができます。このリストには、プロジェクトの [プロパティ マネージャー] ビューで使用可能なすべてのプロパティが含まれています。以前、これらの設定では手動で文字列またはプロパティを入力できましたが、プロパティのリストは表示されませんでした。

この強化は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

IISROOTFOLDER で環境変数パスを解決する InstallScript テキスト置換の強化

InstallScript インストールには、IIS PathWWWRoot レジストリ値に環境変数が含まれている場合に、IIS データを IISROOTFOLDER にインストールするためのサポートが追加されました。以前、IIS サポートを含む InstallScript インストールで IISROOTFOLDER 変数のテキスト置換が行われた場合、Windows Server 208 以降のシステム上では、IISROOTFOLDER の環境変数部分が解決されませんでした。コンポーネントが IISROOTFOLDER にインストールされるように構成されている場合、未解決のパスが無効であるためにインストールが失敗しました。

Update.exe マニフェストに必要実行レベルを指定する機能

"必要実行レベル" 設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビューにあるパッチの構成の [詳細] タブで設定できます。また、QuickPatch プロジェクトの場合、[一般情報] ビューの [ビルドの設定] 領域にある [詳細] タブで設定できます。

これらの新しい設定を使って、Windows Vista 以降のプラットフォーム上で Update.exe ファイルがアップグレードを実行するために必要とする最低実行レベルを指定します。InstallShield が、必要レベルを指定するマニフェストを追加します。デフォルトで、InstallShield は以前のセットアップランチャーのマニフェストで構成されたレベルを使用します。

以前、"必要実行レベル" 設定は Setup.exe セットアップランチャーでのみ使用できました。Update.exe パッチを作成するとき、InstallShield は以前のセットアップ ランチャーのマニフェストで構成された実行レベルを必要としました。

詳しくは、次を参照してください:

- [詳細] タブ ([パッチのデザイン] ビュー)
- [詳細] タブ (QuickPatch プロジェクト)

非圧縮 .cab ファイルの作成機能

[リリース] ビューで選択されたリリースの [ビルド] タブに、新しい "Cab の最適化タイプ" 設定が追加されました。"圧縮" 設定で [圧縮] またはカスタム オプションの 1 つを選択した場合、"Cab 最適化タイプ" 設定を使って、リリースの .cab ファイルをビルドするときに InstallShield が使用する圧縮の種類を指定します。使用可能なオプションには、[LZX 圧縮]、[MSZIP 圧縮]、または [非圧縮] があります。

"Cab の最適化タイプ" 設定は、これまでの "サイズの最適化" 設定に取って代わります。"サイズの最適化" 設定は、LZX 圧縮と MSZIP 圧縮のみをサポートし、非圧縮オプションは提供されていませんでした。

オートメーション インターフェイスには、この新しい設定のサポートが含まれています。ISWiRelease object オブジェクトには、新しく CabCompressionType プロパティが追加されました。このプロパティを使って、オートメーション インターフェイスを通してリリースをビルドするときに、3 つの圧縮オプションの中から 1 つを指定できます。

この強化は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

詳細については、「リリースの [ビルド] タブ」を参照してください。

InstallShield 前提条件エディターにおける定義済みオペレーティング システム条件のリスト拡張

InstallShield 前提条件エディターを使って InstallShield 前提条件の設定を追加または変更するときに表示される [前提条件設定] ダイアログ ボックスに、Windows Server 2008 R2 用の定義済みオペレーティング システム条件が追加されました。一部の Windows 7 オプションは、ワークステーションのバージョン、サーバーのバージョン、またはいずれかのバージョンをチェックするため、どの Windows バージョンが選択されているかをより明確に示すように、これらのオプションの名前が変更されました。[前提条件設定] ダイアログ ボックスには、Windows Server 2008 R2 ではなく Windows 7 のみをチェックする新しいオプションも追加されました。これらの変更によって、任意の InstallShield 前提条件について、よりターゲットを絞ったオペレーティング システム条件を素早く定義することができます。

InstallShield 前提条件は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加できます。

[再配布可能ファイル] ビューと [前提条件] ビューの新しい [更新] ボタン

[再配布可能ファイル] ビュー (基本の MSI、InstallScript MSI、およびトランスフォーム プロジェクト) および [前提条件] ビュー (InstallScript プロジェクト) に表示される再配布可能ファイルのリストを更新するための、新しい [更新] ボタンが追加されました。以前は、InstallShield でいずれかのビューが開かれている状態でコンピュータに再配布可能ファイルを追加したとき、更新済みのリストを表示するためには、プロジェクトを一度閉じてから再び開く必要がありました。

この強化は、基本の MSI、InstallScript、InstallScript MSI、およびトランスフォーム プロジェクトで使用できます。

IUSR ユーザー アカウントのアクセス許可設定におけるサポート

InstallShield は、よく知られるユーザー アカウント IUSR のファイル、フォルダー、およびレジストリ キーの保護をサポートします。

プロジェクトでアクセス許可の構成にカスタム InstallShield 処理を使用する場合、[アクセス許可] ダイアログ ボックスの [ユーザー] リストに新しい [IUSR] オプションが追加されています。カスタム InstallShield 処理を使う場合、[一般情報] ビューの "ロックダウンの設定方法" 設定でこのオプションを選択します。この機能は、基本の MSI、InstallScript MSI、マージ モジュール、およびトランスフォーム プロジェクト タイプで使用できます。

InstallScript 関数 **SetObjectPermissions** を使ってアクセス許可を設定する場合、今回より、szUser パラメーターで IUSR を渡すことができます。この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。基本の MSI、InstallScript MSI、およびマージ モジュールプロジェクトでは、InstallScript カスタム アクションを使って、この関数を使用することができます。

SetObjectPermissions 関数の新しい IS_PERMISSIONS_OPTION_ALLOW_ACCESS 定数

InstallScript 関数 **SetObjectPermissions** と使用できる、IS_PERMISSIONS_OPTION_ALLOW_ACCESS 定数が追加されました。**SetObjectPermissions** の nOptions パラメータで、数値 0 の代わりにこの定数を渡して、設定されている許可が特定のファイル、フォルダー、またはレジストリ キーへのアクセスを可能とすることを示します。

この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。基本の MSI、InstallScript MSI、およびマージ モジュールプロジェクトでは、InstallScript カスタム アクションを使って、この関数を使用することができます。

詳細については、「SetObjectPermissions」を参照してください。

DialogSetInfo 関数の新しい定数

nInfoType パラメーターに DLG_INFO_ALTIMAGE を使用するときに、InstallScript 関数 **DialogSetInfo** の nParameter パラメーターで 2 つの定数を利用できます:

- DLG_INFO_ALTIMAGE_VERIFY_BMP — このていすうは、szInfoString で示されたビットマップを、次に続くダイアログに使用することを指定します。このビットマップを使用する前に、インストールはそのビットマップの存在を確認します。そのビットマップが存在しない場合、関数はビットマップが見つからなかったことを示すエラーを返します。
- DLG_INFO_ALTIMAGE_REVERT_IMAGE — この定数は、そのダイアログが、デフォルトのビットマップを表示することを指定します。この定数は、値 -1 の代替として使用できます。この定数を使用すると、インストールはビットマップの存在を確認しません。

nParameter パラメーターで TRUE を渡すと、インストールはビットマップの存在を確認しません。この状況でビットマップが存在しない場合、関数はエラーを返します。

この関数は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの InstallScript イベントで使用できます。

詳細については、「DialogSetInfo」を参照してください。

.NET Framework 4.0 サポートの強化

新しい FOLDER_DOTNET_40 InstallScript 変数が、追加されました。この変数は、.NET Framework 4.0 ファイルのパスを格納します。

Is 関数と使用するための 2 つの新しい定数が追加されました:

- REGDB_KEYPATH_DOTNET_40_CLIENT
- REGDB_KEYPATH_DOTNET_40_FULL

各パッチ構成の出力場所とキャッシュにおけるパス変数のサポート

[パッチのデザイン] ビューで選択されたパッチの構成の [共通] タブには、適切なフォルダを参照できる 2 つの設定 (″パッチ出力の場所″ および ″パッチ作成キャッシュ″) があります。InstallShield では、フォルダに絶対パスの代わりにパス変数を使用できるようになりました。このサポートによって、異なるディレクトリ構造を使用するビルドおよび開発マシンでリリースをビルドし易くなります。

このサポートは、開発システム上で InstallShield がパス変数を使用するように構成されている場合に活用できます。(つまり、[オプション] ダイアログ ボックスの [パス変数] タブで、パス変数のサポートを許可するように設定する必要があります。)InstallShield で絶対パスを使用するように構成した場合、[パッチのデザイン] ビューにはフォルダへの絶対パスが表示されます。

この強化は、基本の MSI および InstallScript MSI プロジェクトで使用できます。

InstallScript 関数 ServiceExistsService および ServiceGetServiceState は昇格された権限を必要としません

今後、InstallScript 関数 **ServiceExistsService** および **ServiceGetServiceState** は昇格された権限を必要としません。したがって、基本の MSI インストールの [インストール] – UI シーケンスなどで、エンド ユーザーが制限された権限を持つ場合でも、インストールでこれらの関数を呼び出すことができます。

重要な情報

InstallShield の複数エディションをインストールする

InstallShield 2020 のどれか 1 つのエディション (Premier、Professional、または Express) を同じシステムに同時にインストールすることができます。

InstallShield の複数バージョンをインストールする

InstallShield 2020 は、別のバージョンの InstallShield と同じマシン上にインストールすることが可能です。

InstallShield 2020 Standalone Build は、同じマシン上で別のバージョンの Standalone Build と共存することができます。ほとんどの場合、InstallShield がインストールされているマシン上に Standalone Build がインストールされることはありません。両方を同じマシン上にインストールしてオートメーション インターフェイスを使用したい場合は、登録およびアンインストール時の特別考慮について記載されている「Standalone Build と InstallShield を同一のマシン上にインストールする」を参照してください。

InstallShield と Visual Studio との統合

Microsoft Visual Studio の統合は 1 度に InstallShield の 1 バージョンとのみ可能です。システムで最後にインストールまたは修復された InstallShield のバージョンが Visual Studio の統合に使用されます。

InstallShield 2010 の新しい機能

Expansion Pack for Visual Studio 2010

InstallShield 2010 Expansion Pack for Visual Studio 2010 は、Visual Studio 2010 および .NET Framework 4 の最終版をサポートするための変更が含まれています。また、その他の変更も含まれています。

Microsoft .NET Framework 4 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい 4 つの InstallShield 前提条件が含まれています:

- Microsoft .NET Framework 4.0 Full
- Microsoft .NET Framework 4.0 Full (Web ダウンロード)
- Microsoft .NET Framework 4.0 Client
- Microsoft .NET Framework 4.0 Client (Web ダウンロード)

これらの前提条件ついて以下の点にご注意ください:

- 完全版の前提条件は、.NET Framework 4 をターゲットにするアプリケーションを実行および開発するのに必要な .NET Framework ランタイムと関連ファイルをインストールします。
- クライアント前提条件は、ほとんどのクライアント アプリケーションを実行するのに必要な .NET Framework ランタイムと関連ファイルをインストールします。
- 2 つの Web ダウンロード前提条件には、インターネットへの接続が必要です。これらの前提条件のダウンロードには、適切な場合、再配布可能ファイルが必要です。その他 2 つの前提条件は、インターネットへの接続が不要なスタンドアロン インストールです。

.NET Framework 4.0 には、Windows Installer 3.1 以降、および Windows Imaging Component が必要です。したがって、.NET Framework 4.0 Full 前提条件は、以下の InstallShield 前提条件を依存関係とします:

- Windows Installer 3.1 (x86)
- Windows Imaging Component (x86)
- Windows Server 2003 SP1 (x86) 用 Windows Installer 3.1
- Windows Imaging Component (x64)
- Windows Server 2003 SP1 (IA64) 用 Windows Installer 3.1
- Windows Server 2003 SP1 (x64) 用 Windows Installer 3.1
- Windows XP (x64) 用 Microsoft Windows Installer 3.1

また、.NET Framework 4.0 Client 前提条件は、以下の InstallShield 前提条件を依存関係とします:

- Windows Installer 3.1 (x86)
- Windows Imaging Component (x86)
- Windows Server 2003 SP1 (x86) 用 Windows Installer 3.1
- Windows Imaging Component (x64)
- Windows Server 2003 SP1 (x64) 用 Windows Installer 3.1
- Windows XP (x64) 用 Microsoft Windows Installer 3.1

したがって、任意の .NET Framework 4.0 前提条件をプロジェクトに追加すると、InstallShield はデフォルトで Windows Installer と Windows Imaging Component の前提条件ををインストールに追加します。このため、インストールのサイズが大きくなることがあります。一部の状況において、InstallShield 前提条件エディタ (InstallShield Premier および Professional Edition で提供されています) を使って .NET Framework 4 前提条件から一部の依存関係を削除する必要があります。たとえば、製品が Windows Vista以降または Windows Server 2008 以降を必要とする場合は、以前のバージョンの Windows をターゲットとする Windows Installer 3.1 の依存関係を削除する必要があるかもしれません。

Visual Studio 2010 サポートのための追加前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる、以下の新しい InstallShield 前提条件が含まれています:

- Microsoft SQL CE 3.5 SP2
- Microsoft ReportViewer 2010
- Microsoft VSTO 2010 Runtime
- FSharp Redistributable Package 2.0
- Microsoft Office 2007 PIA (この前提条件は Microsoft Office 2007 Primary Interop Assemblies をインストールします。この前提条件を使用するには、マイクロソフトの Web サイトから PrimaryInteropAssembly.exe ファイルをダウンロードおよび実行してo2007pia.msi ファイルを抽出します。)システム上の .msi パッケージの場所によって、.prq ファイル内の o2007pia.msi インストールのパスを変更しなくてはならない場合があります。)

.NET Framework 4 用の定義済みシステム検索

InstallShield に 2 つの新しい定義済みシステム検索が追加されました:

- Microsoft .NET Framework 4.0 Full パッケージ
- Microsoft .NET Framework 4.0 Client パッケージ

インストールでこれらのいずれかが必要な場合、プロジェクト アシスタントの [インストール要件] ページまたは [システム検索] ビューを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

InstallShield 2010 SP1 の新しい機能

InstallShield 2010 Service Pack 1 (SP1) には、Windows 7、Windows Server 2008 R2、および Windows Installer 4.5 の最終版をサポートするための変更が含まれています。また、その他の変更も含まれています。

重要 *• InstallShield 2010 SP1 で InstallShield 2010 を開くとき、プロジェクトを InstallShield 2010 SP1 にアップグレードすることを許可する必要があります。InstallShield 2010 SP1 では、InstallShield 2010 プロジェクトで使用できないテーブルをサポートするため、アップグレード中にこれらのテーブルを追加しなくてはなりません。InstallShield 2010 SP1 プロジェクトを以前のバージョンの InstallShield (SP1 を適用する前の InstallShield 2010 を含む) で開くことはできませんので、ご注意ください。このため、複数のユーザーが InstallShield プロジェクトを開いて編集する場合、すべてのユーザーが同時に SP1 パッチを適用するようにしてください。*

InstallShield 2010 プロジェクトを InstallShield 2010 SP1 で開くと、そのプロジェクトを新しいバージョンに変換するかどうかをたずねるメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前にプロジェクトのバックアップ コピーが作成されます。

Setup.exe マニフェストには、Windows 7 および Windows Server 2008 R2 システム上で Program Compatibility Assistant のトリガーを回避するための、互換性セクションが追加されました。

InstallShield プロジェクトで、インストール用のセットアップランチャーの作成を構成した場合、InstallShield がセットアップランチャーに対して作成するマニフェストには、今回より互換性セクションが含まれます。以前は、この互換性セクションが含まれておらず、Windows 7 および Windows Server 2008 R2 システム上でインストールの終了時に Program Compatibility Assistant (PCA) ダイアログ ボックスが表示されました。この PCA ダイアログ ボックスは、プログラムが正しくインストールされなかった可能性があることを通知しました。このダイアログ ボックスは、インストールがアプリケーションのアンインストール キーを作成しなかった場合に表示されました。この状況は、エンド ユーザーがインストールをキャンセルした場合、またはインストールが正しく完了しなかった場合に発生します。

App-V パッケージのアップグレードの作成と App-V パッケージの圧縮サポート

InstallShield は、今回より App-V アップグレードの作成をサポートします。Microsoft App-V アシスタントの [パッケージ情報] ページに、新しい [アップグレードの設定] リンクが追加されました。このリンクをクリックして、アップグレードを作成するかどうかを指定します。アップグレードを作成すると指定した場合、App-V パッケージのファイル名の後にバージョン番号を付加するかどうかなど、追加の情報を指定することができます。デフォルトでは、App-V パッケージのアップグレードは作成されません。

Microsoft App-V アシスタントの [ビルド オプション] ページには、App-V パッケージ内のデータ ファイルの圧縮を使用するかどうかを指定できる新しい設定が追加されました。この設定に [はい] を選択すると、InstallShield は App-V パッケージに zlib 圧縮を使用します。

App-V のアップグレードおよび圧縮機能は、基本の MSI と MSI データベース プロジェクトタイプで使用できます。

Microsoft App-V アシスタントは、InstallShield Premier Edition で使用できます。

詳細については、次を参照してください。

- アップグレード パッケージ情報を指定する
- App-V パッケージ内のデータ ファイルを圧縮するかどうかを指定する

Windows サービス用の新しいカスタマイズ オプションの構成機能およびその他のサービス関連の強化機能における Windows Installer 5 サポート

InstallShield は今回より、Windows サービスに対する拡張カスタマイズ オプションの構成をサポートします。カスタマイズ オプションには、システム起動時のパフォーマンスを向上させる新しい遅延自動開始機能、強化されたエラー検出機能、およびシステムの信頼性を向上させる回復オプションなどがあります。Windows Installer 5 は、これらの新しいオプションをサポートしますが、以前のバージョンの Windows Installer はこれらを無視します。

新しいサービス関連の設定を構成するには、[セットアップのデザイン] ビュー (インストール プロジェクトの場合) または [コンポーネント] ビューで、コンポーネントの [詳細設定] 領域にある [サービス] ノードを使用します。[サービス] ノードで、コンポーネントのサービスはすべて、サービス名ごとにグループに分けてリストされています。また、以前から利用可能なサービス関連の設定は、今回より新しい設定と共に同じグリッドに統合されています。InstallShield の以前のバージョンでは、これらの以前から利用可能なサービス関連の設定は [NT サービスのインストール] ノードと [NT サービスのコントロール] ノードのサブビューに分けて表示されました。統合された設定グリッドを使って、インストール中、またはアンインストール中に新しい、または既存のサービスをインストール、構成、開始、停止、または削除するコンポーネントを簡単に作成することができます。

この機能は、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- Windows サービスのインストール、制御、および構成
- サービスの設定

Windows 7 ロゴ プログラムの検証

InstallShield では、新しい 2 つの検証スイート (InstallShield 検証スイート – Windows 7 および InstallShield マージ モジュール検証スイート - Windows 7) が提供されています。これらの検証スイートを使って、Windows Installer ベースのインストール、またはマージ モジュールが Windows 7 ロゴ プログラムに対応するインストール要件を満たすかどうかを検証することができます。Windows 7 ロゴを使用する場合、アプリケーションのインストールがプログラムの要件を満たしている必要があります。

InstallShield を使って、リリースが正しくビルドされるたびに、これらの検証スイートを実行するように構成する場合: [ツール] メニューから [オプション] を選択します。[検証] タブで、適切なチェック ボックスを選択します。

ビルド処理とは別に検証を行う場合: [ビルド] メニューから [検証] をポイントして、適切な新しいスイートを選択します。

ショートカットに Windows シェル プロパティを設定する Windows Installer 5 サポート

InstallShield の [ショートカット] ビューでは、実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。たとえば、エンド ユーザーが製品をインストールした後に、そのショートカットの [スタート] メニュー エントリを新しくインストールされた製品として強調表示しないようにする場合、ショートカットのプロパティを設定できます。このプロパティは、インストールの一部であるツールまたは従属的な製品のショートカットで使用すると便利です。

Windows Installer 5 では、シェル ショートカット プロパティの設定がサポートされています。以前のバージョンの Windows Installer は、これらのプロパティを無視します。

詳細については、「ショートカットのシェル プロパティを設定する」を参照してください。

MsiPrint および MsiLaunchApp コントロール イベントに関する Windows Installer 5 サポート

プロジェクトに含まれるダイアログにコントロールを追加するとき、Windows Installer 5 で使用できる新しいイベントの 1 つを選択できます:

- **MsiPrint** – このイベントを、スクロール可能なテキスト コントロールを含むダイアログにあるプッシュ ボタン コントロールに追加します。エンド ユーザーがプッシュ ボタン コントロールをクリックすると、スクロール可能なテキスト コントロールの内容が印刷されます。
- **MsiLaunchApp** – このイベントをダイアログに含まれるチェック ボックス コントロールに追加してから、イベントの "引数" 設定で起動するファイルを選択できます。エンド ユーザーはチェック ボックスを使って、インストールの終わりにファイルを実行するかどうかを選択できます。通常このイベントは、SetupCompleteSuccess ダイアログ上のチェック ボックス コントロールと共に使用します。チェック ボックス コントロールには、アンインストール中にコントロールが表示されるのを防ぐ条件を含みます。

Windows Installer 5 で、これらのコントロール イベントがサポートされています。以前のバージョンの Windows Installer は、これらのイベントを無視します。したがって、Windows Installer 4.5 以前が搭載されているシステム上でインストールを実行するときに、これらの新しいイベントの 1 つまたは両方を使用する場合、ダイアログ コントロールに条件を追加することで、Windows Installer 4.5 以前が搭載されたシステム上でそれらが表示されないようにします。

この機能は、基本の MSI プロジェクトで使用できます。

プロジェクトに印刷または起動サポートを追加するときに、インストールのターゲット システムに Windows Installer 4.5 以前が搭載されている場合、InstallShield が提供するサポートの使用をお勧めします。印刷サポートの追加に関する情報については、「プロジェクト アシスタントでインストールに使用するダイアログを指定する」を参照してください。印刷サポートの追加に関する情報については、「[印刷] ボタンをダイアログに追加する」を参照してください。InstallShield で提供されているサポートに、Windows Installer 5 は不要です。

追加の変更

InstallShield 2010 SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2010 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

カスタマイズされた仮想アプリケーションの作成をサポート

InstallShield を使って、Microsoft App-V フォーマットでカスタマイズされた仮想アプリケーションを作成できます。仮想化技術を使って、アプリケーションを独自の環境に隔離することで、既存アプリケーションとの競合を回避したり、基盤となるオペレーティング システムの変更を避けたりすることが可能です。

仮想アプリケーションは、アプリケーション層とオペレーティング システム層を別々に維持する仮想環境で実行されます。各アプリケーションには、その仮想環境に、それ自身の構成情報を持ちます。これにより、多数のアプリケーションは、競合することなく、同一のコンピュータで他のアプリケーションとサイド バイ サイドで実行が可能になります。

仮想アプリケーションを作成するには、基本の MSI および MSI データベース プロジェクトで使用できる、新しい Microsoft App-V タブを使います。

仮想化サポートは、InstallShield Premier Edition で提供されています。

また、InstallShield には Microsoft App-V 4.5 Desktop Client インストールおよび Microsoft Application Error Reporting インストール用の InstallShield 前提条件が含まれています。Application Error Reporting 前提条件は、App-V 前提条件の依存関係です。これらの InstallShield 前提条件用の再配布可能ファイルは Microsoft から入手しなくてはならないため、InstallShield 内部からダウンロードすることはできません。Microsoft からそれらのファイルを入手した後、 InstallShield 前提条件エディターで前提条件を編集するときに表示される場所に配置してください。

詳細については、次を参照してください。

- カスタム仮想アプリケーションの作成
- 仮想化について
- Microsoft App-V アプリケーションの作成

Windows 7 および Windows Server 2008 R2 システムをターゲットにできる機能

InstallShield では、インストールに Windows 7 または Windows Server 2008 R2 が必要であることを指定できます。また、これらのオペレーティング システムに対する機能条件およびコンポーネント条件をビルドすることができます。

Windows 7 および Windows Server 2008 R2 でタスクバーにインストールの進行状況を表示するサポート

Windows 7 と Windows Server 2008 R2 で実行されるインストールは、今回より、ファイル転送中に Windows タスクバーに進行状況バーを表示します。この情報は、InstallScript および InstallScript MSI インストールに適用します。また、[ビルボード] ビューで構成されたビルボードを表示する基本の MSI インストールにも適用します。進行状況バーは、Windows の以前のバージョンのタスクバーには表示されません。また、セットアップの初期化中、または InstallShield 前提条件のインストール中にも表示されません。

Beta Windows Installer 5 におけるユーザーごとインストールをサポート

[一般情報] ビューには、"[ユーザーごと] オプションの表示" 設定があります。この設定を使って、特定の状況下において ReadyToInstall ダイアログにエンド ユーザーが製品をインストールする方法 (現在のユーザーまたはすべてのユーザー) を指定できるボタンを含めるかどうかを指定できます。[ユーザーごと] ボタンは、Windows Installer プロパティ MSIINSTALLPERUSER を 1 に等しく設定して、パッケージを現在のユーザーにインストールすることを示します。MSIINSTALLPERUSER プロパティは Windows Installer 5 で使用できます。

この機能は、基本の MSI プロジェクトで使用できます。

詳細については、「ユーザーごとのインストールとマシンごとのインストールの違い」を参照してください。

Windows Installer 5 Beta における大きいパッケージのインストールにかかる所要時間の短縮機能をサポート

[一般情報] ビューにある "高速インストール" 設定では、大きい Windows Installer パッケージのインストールにかかる所要時間を短縮するのに役立つ 1 つまたは複数のオプションを選択できます。たとえば、インストールについてシステム復元ポイントを保存しないことを指定できます。インストールがファイル コスティングのみを実行して、その他のコスト チェックをスキップするように指定することも可能です。

この設定は、新しい Windows Installer プロパティ MSIFASTINSTALL を構成します。このプロパティは、コマンドラインで設定できます。Windows Installer 5 で、このプロパティがサポートされています。以前のバージョンの Windows Installer はこれを無視します。

この設定は、基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、"高速インストール" 設定の説明を参照してください。

Windows Installer 5 追加機能のベータ サポート

InstallShield のダイレクト エディターには、Windows Installer 5 テーブル (MsiLockPermissionEx、MsiServiceConfig、MsiServiceConfigFailureActions、および MsiShortcutProperty) のベータ サポートが含まれています。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

仮想マシンの存在を検出できる機能

InstallShield を使って、インストールが以下の種類の仮想マシンで実行されているかどうかを検出できます:

- Microsoft Hyper-V
- VMware Player、VMware Workstation、または VMware Server などの VMware 製品
- Microsoft Virtual PC

MSI DLL カスタム アクションをプロジェクトに追加する場合に、仮想マシンを検出するための 2 つの新しい Windows Installer プロパティIS_VM_DETECTED と IS_VM_TYPE が追加されました。カスタム アクションを、InstallShield と共にインストールされる SetAllUsers.dll ファイルで ISDetectVM 関数を呼び出すように構成します。

また、InstallScript 言語が検出をサポートできるように拡張されました。構造 SYSINFO は、新しい bIsVirtualMachine メンバーを含み、InstallScript 関数 **GetSystemInfo** と使用できる、新しい VIRTUAL_MACHINE_TYPE 定数が追加されました。

この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。

64 ビット COM 抽出サポート

InstallShield は、今回より、64 ビット COM 抽出をサポートします。InstallShield を 64 ビット オペレーティング システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。

64 ビット COM 抽出を実行するためには、InstallShield が、64 ビット オペレーティング システムにインストールされていなくてはなりません。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- 64 ビット オペレーティング システムをターゲットにする
- COM 情報を COM サーバーから抽出する

ファイル、フォルダー、およびレジストリ キーのアクセス許可を設定するための新しいサポート

InstallShield では、ロックダウン環境で製品を実行するエンド ユーザー向けに、ファイル、フォルダー、およびレジストリ キーを保護するための新しい 2 つの方法が提供されています:

- **カスタム InstallShield 処理** – Windows Installer ベースのプロジェクトでは、実行時にアクセス許可を設定するためのカスタム サポートの使用を選択できます。このオプションを使うと、InstallShield は .msi データベースのカスタム ISLockPermissions テーブルに製品のアクセス許可情報を格納します。InstallShield はまた、アクセス許可を設定するためのカスタム アクションをプロジェクトに追加します。このサポートは、基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。
- **InstallScript 関数、SetObjectPermissions** – InstallScript イベントおよび InstallScript カスタム アクションで新しい **SetObjectPermissions** 関数を使って、実行時にアクセス許可を設定できます。この関数は、InstallScript、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

[カスタム InstallShield 処理] オプションを使う場合、アクセス許可を設定するファイル、フォルダー、またはレジストリ キーを、インストールの一部としてインストールしなくてはなりません。**SetObjectPermissions** 関数を使う場合、ファイル、フォルダー、またはレジストリ キーをインストールの一部としてインストールするか、ターゲット システム上に既存するかどうかを問いません。

以前、InstallShield でアクセス許可を設定するための唯一のオプションは、従来型の Windows Installer 処理のみでした。このオプションを使うと、.msi データベースの LockPermissions テーブルに製品のアクセス許可情報が格納されます。新しい [カスタム InstallShield 処理] オプションと **SetObjectPermissions** 関数には、[従来型の Windows Installer 処理] オプションよりも多くの利点があります。

- カスタム オプションおよび **SetObjectPermissions** 関数を使うと、[従来型の Windows Installer 処理] オプションではサポートされていない、多くのセキュリティ識別子 (SID) を使用できます。
- カスタム オプションと **SetObjectPermissions** 関数は、従来型のオプションとは異なり、サポート対象の SID の翻訳されたユーザー名をサポートします。従来型のオプションで、非英語システム上で翻訳された名前を使ってアクセス許可を設定すると、インストールが失敗する可能性があります。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、指定するアクセス許可を特定のユーザーまたはグループが所持することを拒否できます。従来型の処理で、これは不可能です。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、ターゲット システムに既存するファイル、フォルダー、またはレジストリ キーに、アクセス許可を追加できます。このとき、そのオブジェクトに既存するアクセス許可は削除されません。従来型の処理では、既存するアクセス許可が削除されました。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、フォルダー (またはレジストリ キー) のアクセス許可を構成し、そのフォルダーのサブフォルダーおよびファイル (またはレジストリ キーのサブキー) すべてに同じアクセス許可を適用するかどうかを指定できます。従来型の処理では、フォルダー内のサブフォルダーまたはファイル (レジストリ キーの下にあるサブ キー) にアクセス許可を構成する場合、ターゲット システム上で作成された親は、自動的に子のアクセス許可を継承します。
- カスタム オプションおよび **SetObjectPermissions** 関数を使って、インストール中に作成される新しいユーザーのアクセス許可を構成できます。従来型の処理では、実行時にユーザーがターゲット システム上に既存する必要があったため、これは不可能でした。

[一般情報] ビューに追加された "ロックダウンの設定方法" 設定を使って、プロジェクトに含まれるファイル、フォルダー、およびレジストリ キーに設定する新しいアクセス許可について、新しいカスタム InstallShield 処理または従来型の Windows Installer 処理のどちらを使用するかを指定できます。プロジェクトで既にいくつかのアクセス許可を構成済みの場合にこの設定の値を変更する場合、InstallShield では、既存アクセス許可に別の処理方法を使用するかどうかを指定できます。すべての新しいプロジェクトにおける、この設定のデフォルト値は [カスタム InstallShield 処理] オプションです。プロジェクトを InstallShield 2009 以前から InstallShield 2010 にアップグレードした場合、この設定のデフォルト値は [従来型の Windows Installer 処理] オプションです。この新しい設定は、基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで利用できます。

詳細は、次を参照してください:

- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する
- SetObjectPermissions
- SetObjectPermissions の例

InstallScript プロジェクトにおける InstallShield 前提条件のサポート

InstallShield では、InstallScript プロジェクトに InstallShield 前提条件を追加できるようになりました。以前、この種類の再配布可能ファイルは基本の MSI プロジェクトおよび InstallScript MSI プロジェクト プロジェクトでのみサポートされていました。

InstallShield 前提条件 は、通常、製品が必要とする製品またはテクノロジ フレームワークをインストールする再配布可能ファイルです。今回より、InstallShield で提供されている InstallShield 前提条件または独自で作成したカスタム InstallShield 前提条件をどれでも InstallScript プロジェクトに追加できます。異なる種類のプロジェクト タイプを組み合わせている場合、InstallShield ではすべての基本の MSI、InstallScript、および InstallScript MSI プロジェクトに含まれるこの種類の再配布可能ファイルを再利用できるため、簡単にマトリックスのテストを行うことができます。

InstallShield 前提条件を InstallScript プロジェクトに追加するには、新しい [前提条件] ビューを使います。InstallScript プロジェクトでは、InstallShield 前提条件タイプのセットアップ前提条件をサポートします。これは、メイン インストールのユーザー インターフェイスが実行される前に実行します。InstallScript プロジェクトは、機能と関連付けられた InstallShield 前提条件である、機能前提条件をサポートしません。

詳細については、次を参照してください。

- InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する
- InstallShield 前提条件を含むインストールの実行時の動作
- InstallShield 前提条件のインストール順を指定する
- 特定の InstallShield 前提条件の実行時の場所を指定する
- リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する
- InstallShield 前提条件およびその他の再配布可能ファイルをデザインする
- [前提条件] ビュー

Windows Installer、.NET Framework、Crystal Reports、およびその他の再配布可能ファイルのための新しい InstallShield 前提条件

InstallShield には、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに追加することができる多くの InstallShield 前提条件が含まれています。

- Windows Installer 4.5 (Windows Installer 4.5 用の InstallShield 前提条件は、Microsoft KB958655 の修正を含みます。)
- Windows Installer 4.5 Update (Windows Installer 4.5 用の InstallShield 前提条件は、Microsoft KB958655 の修正を含みます。このアップデートには、Windows Installer 4.5 がターゲット システム上にインストール済みでなくてはなりません。)
- Windows Installer 3.1、Windows Installer 3.0、および Windows Installer 2.0 (これらのバージョンの Windows Installer 再配布可能ファイルは、以前、[リリース] ビューを使ってプロジェクトに Windows Installer を追加すると利用することができました。InstallShield 前提条件としては提供されていませんでした。)
- .NET Framework 3.0 SP1
- .NET Framework 2.0 SP2
- Internet Explorer 8
- Microsoft SQL Server 2008 Express SP1
- Microsoft SQL Server 2005 Express SP3
- Microsoft Visual C++ 2005 SP1 Redistributable Package
- Oracle 11g Instant Client 11.1.0.7 (Oracle は、Oracle Instant Client ファイル用のインストーラーを提供していません。したがって、プロジェクトでこの InstallShield 前提条件を使用する前に、.msi パッケージを作成する必要があります。これは、InstallShield と共にインストールされる Oracle Instant Client インストール プロジェクト (*InstallShield Program Files フォルダー*¥Support¥Oracle Instant Client) を使って簡単に作成できます。
- Crystal Reports Basic for Visual Studio 2008 (Visual Studio 2008 と共にインストールされた Crystal Reports Basic と共に使用できる前提条件。システム上の .msi パッケージの場所によって、.prq ファイル内の Crystal Reports Basic のパスを変更しなくてはならない場合があります。)

Microsoft SQL Server 2008 SP1 サポート

InstallShield は、今回より、SQL Server 2008 SP1 上で SQL スクリプトを実行するためのサポートを含みます。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2008 SP1 が含まれています。

Oracle 11g サポート

InstallShield には、今回より、Oracle 11g で SQL スクリプトを実行するためのサポートが含まれています。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、Oracle 11g が含まれています。

IIS Web サイトのスキャン、その設定の記録、およびそれらの設定を InstallShield プロジェクトにインポートする新しいツール

InstallShield には、IIS スキャナー (IISscan.exe) が搭載されています。この新しいコマンドライン ツールを使って、既存の IIS Web サイトをスキャンし、その Web サイトに関する IIS データを記録できます。IIS スキャナーは、Web サイト、その仮想ディレクトリ、アプリケーション、およびアプリケーション プールすべての設定を含む XML

ファイルを作成します。この XML ファイルを使って、InstallShield の [IIS 構成] ビューに IIS データをインポートできます。IIS データをプロジェクトにインポートしてから、必要に応じて、[IIS 構成] ビューを使って IIS の設定を変更できます。

IIS データを InstallShield プロジェクトにインポートできる機能は、InstallShield Premier Edition のみで使用できます。

詳細については、次を参照してください。

- IIS Web サイトのスキャンをして、その設定を InstallShield プロジェクトにインポートする
- Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする
- IISscan.exe

IIS Web アプリケーションを Web サイトに追加できる機能と、その他の IIS 関連機能の強化

InstallShield では、今回より、IIS Web アプリケーションを Web サイトに追加できます。そのためには、[IIS 構成] ビューで Web サイトを右クリックしてから、[新しいアプリケーション] をクリックします。新しいアプリケーションを追加した後、その設定を右側のペインで構成できます。

また、InstallShield ではアプリケーション無しで仮想ディレクトリを作成することもできます。以前は、仮想ディレクトリを作成するたびに、アプリケーションも自動的に作成されました。

また、InstallShield には次の新しい IIS 設定が追加されました:

- **マネージ パイプライン モード** – 適切な要求処理パイプライン モード (統合またはクラシック) を指定できます。
- **32 ビット アプリケーションを有効にする** – 選択したアプリケーション プールの 32 ビット アプリケーションを 64 ビット システム上で実行可能にするかどうかを指定できます。
- **.NET Framework バージョン** – アプリケーション プールがロードする .NET Framework のバージョンを指定できます。
- **ASP.NET プラットフォーム** – .NET Framework がインストールされている Windows の 64 ビット バージョンでインストールを実行できるようにする場合、どの ASP.NET プラットフォームを使って Web サイト、アプリケーション、または仮想ディレクトリを ASP.NET バージョンにマップするのかを指定します。

この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。

詳細は、次を参照してください:

- Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加
- [IIS 構成] ビュー

ランタイム テキスト ファイルの変更を構成できる新しいビュー

InstallShield に新しく追加された [テキスト ファイルの変更] ビューを使って、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。

Windows Installer プロパティを使って、検索に含める、または検索から除外するテキスト ファイルの名前を指定することができます。プロパティを使って、検索文字列と置換文字列を指定することもできます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、製品のテキスト ファ

イルがターゲット システムで変更される時に使用できるようになります。たとえば、エンド ユーザーが IP アドレスを指定しなくてはならないダイアログがプロジェクトに含まれている場合、インストールは一連のファイル内でトークンを検索して、エンド ユーザーが入力した IP アドレスで、それを置換します。

[テキスト ファイルの変更] ビューは、[XML ファイルの変更] ビューで XML ファイルを構成する代わりの方法として使用できます。[テキスト ファイルの変更] ビューには、いくつかの利点があります。たとえば、この新しいビューにはランタイム要件がありません。一方、[XML ファイルの変更] ビューを使って構成された変更の場合、ターゲット システムで MSXML が必要です。また、[テキスト ファイルの変更] ビューで変更を構成する場合、XPath クエリの入力は不要ですが、[XML ファイルの変更] ビューを使った場合は、それが必要です。

[テキスト ファイルの変更] ビューは、基本の MSI プロジェクトおよび InstallScript MSI プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- テキスト ファイルの変更
- テキスト 変更セットを作成する
- テキスト ファイルの変更を指定する
- テキスト ファイルの変更が行われる順番を変更する
- Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する
- ANSI テキスト ファイルを開く時に使用するコード ページを指定する

新しい [文字列エディター] ビュー

InstallShield に [文字列エディター] ビューが追加されました。このビューには、プロジェクト内の言語非依存文字列 ID と、それにに対応する言語固有の値がスプレッドシート形式のテーブルで表示されます。[文字列エディター] ビューでは、インストール処理で実行時に表示されるローカライズ可能なすべてのテキスト文字列を、1 ヶ所でまとめて制御することができます。このビューは、以前 [一般情報] ビュー内のノードで表示された文字列テーブルの代わりとなります。以下は、新しいビューの主な特徴を説明します:

- ビューには文字列エントリを追加、編集、削除、検索、置換、エクスポート、およびインポートするためのボタンを含むツールバーがあります。特定の文字列 ID が使用されているすべてのインスタンスを識別するために、プロジェクト内を検索するためのボタンもあります。
- ビューの上部には新しいグループ ボックス領域があり、列ヘッダーをこの領域にドラッグ アンド ドロップすることで、ビュー内の行を階層形式に整列することができます。この機能を使って、言語または更新日などのカテゴリ別に文字列エントリを簡単に並べ替えることができます。
- このビューではダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まない文字列を隠します。

この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- [文字列エディター] ビュー
- エンドユーザー インターフェイスのローカライズ

Unicode サポート

InstallShield は次の 3 つの側面から、モダン言語の複数言語インストールを完全にサポートします: Windows Istaller データベースを Unicode 形式でビルド可能、InstallShield プロジェクトを Unicode 形式で保存可能、InstallShield インターフェイスで複数文字セットから同時に Unicode 文字を入力および表示可能

Unicode (UTF-8) データベース

[リリース] ビューの [ビルド] タブにある "UTF-8 データベースのビルド" 設定では、Windows Installer データベースを指定して、任意のインスタンスまたは言語トランスフォームと共に UTF-8 エンコードでビルドできます。UTF-8 エンコードは、すべての言語の文字を同時にサポートするため、エンド ユーザーに表示するテキストおよびファイル名とレジストリ キーの両方で、たとえば日本語とドイツ語、またはロシア語とポーランド語のように自由に言語を組み合わせて使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。Unicode サポートはまた、IIS サポートやテキストおよび XML ファイルの変更をはじめとするインストール実行時の主要な部分にも追加されています。

新しい "UTF-8 データベースのビルド" 設定のデフォルト値は [いいえ] です。

オートメーション インターフェイスには、この新しい設定のサポートが含まれています。ISWiRelease オブジェクトには、UTF-8 エンコードを使用するかどうかを指定できる、新しい BuildUTF8Database プロパティが含まれています。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクト タイプで使用できます。

さらに詳しい情報は、"UTF-8 データベースのビルド" 設定の説明を参照してください。

Unicode プロジェクト ファイル (.ism)*

InstallShield は、今回より、バイナリおよび XML プロジェクト ファイルを保存するときに UTF-8 エンコードを使用します。この変更により、プロジェクトで使用されるファイル、レジストリ データ、その他の文字列には、同時にすべての言語の文字を含めることが可能です。このエンコードにより、InstallShield では今後 ISString テーブルに格納される文字列に人間が判読不可能な Base64 エンコードを使用する必要はありません。代りに、プロジェクトに文字列を追加または変更するとき、InstallShield はそれらを判読可能な Unicode 文字列として保存します。これによって、プロジェクトの修正時に変更点を容易に確認することができます。このため、InstallShield 2010 で作成されるすべての新規プロジェクトでは Unicode 文字列のみが使用されます。アップグレード プロジェクトでは、新しい文字列または変更された文字列、およびエクスポート後に再インポートされた文字列には Unicode が使用されます。

ターゲットのビルドで表示できない Unicode 値を使用すると (たとえば、"UTF-8 データベースのビルド" 設定で [いいえ] が選択されている InstallScript インストールまたは基本の MSI インストール)、新しいビルド エラーが変更を必要とする項目をポイントします。一部において、InstallShield の以前のバージョンではサイレントで受け入れられていた無効な文字列エントリを検出する場合もあります。

この機能はすべてのプロジェクト タイプに適用します。

InstallShield の Unicode ビュー

InstallShield の多くのビューは、すべての言語からの文字を表示および入力できるように強化されています。たとえば、[ファイルとフォルダー] ビューでは、英語ローカル システムにある中国語のファイル名が疑問符で表示されることがなくなり、これらのファイルをプロジェクトに追加できるようになりました。同様に、[レジストリ] ビューでタイ語のレジストリ キーまたは値が疑問符に変換されることがなくなり、これらを Windows Installer ベースのプロジェクトにインストールできるようになりました。さらに、新しい独立した [文字列エディター] ビューで、すべての言語からの文字列を表示および変更できるようになりました。以前、文字列エントリは [一般情報] ビューで個別の言語ノードに表示されました。その他、[プロパティ マネージャー]、[パス変数]、[ダイレクト エディター]、[一般情報]、および [セットアップのデザイン] ビューも強化されています。

"UTF-8 データベースのビルド" 設定を [いいえ] に設定する場合は常に、すべてのファイル名、レジストリ キー、およびその他の文字列をターゲット言語のコードページからの文字で構成しなくてはなりません。その場合、ある項目がターゲット言語のコード ページで使用不可能な文字を含む場合、InstallShield はその項目をポイントする新しいビルド エラーを報告します。つまり、"UTF-8 データベースのビルド" 設定に [はい] が選択されていない限り、中国語ファイル名を英語インストールで使用することはできません。

この機能はすべてのプロジェクト タイプに適用します。

基本の MSI プロジェクトで新しく追加された、ビルボード サポート

基本の MSI プロジェクトには、ビルボード サポートが含まれています。ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

基本の MSI プロジェクトにおけるビルボード サポートの主な特徴は次のとおりです:

- プロジェクトに Adobe Flash アプリケーション ファイル (.swf) をビルボードとして追加できます。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。
- InstallShield では .bmp、.gif、.jpg、および .jpeg ファイルをビルボードとして使用できます。
- InstallShield には "ビルボードの種類" 設定が含まれていて、インストールで使用するビルボードのスタイルを指定できます。たとえば、インストールが全画面背景を使用し、ビルボードを前景に、また小さい進行状況ボックスを画面の右下に表示するスタイルがあります。別のスタイルでは、インストールがビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。
- InstallShield では、リリースをビルドおよび実行せずに、実行時にビルボードがどのように表示されるのかをプレビューできます。ビルボードをプレビューすると、そのビルボードに現在構成されている背景色、位置、および関連設定を使ったビルボードの外観を確認できます。

InstallShield の [ビルボード] ビューでは、ビルボード ファイルの追加、その関連設定の構成、ならびにビルボードのプレビューを行います。

詳しくは、次を参照してください:

- 基本の MSI インストールでビルボードを表示する
- 基本の MSI プロジェクトにおけるビルボード ファイルの種類
- 基本の MSI プロジェクトにおけるビルボードの種類
- 基本の MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する
- 基本の MSI プロジェクトにイメージ ビルボードを追加する
- 基本の MSI プロジェクトでビルボードの設定を構成する
- リリースをビルドまたは起動せずにビルボードをプレビューする
- 基本の MSI プロジェクトでビルボードの順序を設定する
- ビルボードを含む基本の MSI インストールにおける実行時の動作

以前は、InstallScript とInstallScript MSI プロジェクトでのみ、ビルボードがサポートされていました。これらのプロジェクト タイプに含まれるビルボード サポートは、基本の MSI プロジェクトに含まれるサポートとは異なりますので、ご注意ください。詳細については、「ビルボードを表示する」を参照してください。

InstallScript および InstallScript MSI プロジェクトで拡張されたビルボード サポート

InstallScript および InstallScript MSI プロジェクトでは、今回より .gif、.jpg、および .jpeg ファイルをビルボードとして使用できます。以前は、.bmp ファイルしかサポートされていませんでした。[サポート ファイル/ビルボード] ビューを使って、プロジェクトにビルボードを追加できます。

InstallScript および InstallScript MSI プロジェクトのビルボード ファイルは、指定された命名規則に従わなくてはなりません。各ファイル名は **bbrd** で始まり、ビルボード番号 (1 から 99) が続きます。最後に、サポート対象のファイル拡張子 (.bmp、.gif、.jpg、または .jpg) で終わります。ファイル名番号が連続番号である必要はなくなりました。つまり、プロジェクトに bbrd1.jpg、bbrd3.jpg、および bbrd5.jpg を追加すると、実行時に各イメージが順番通りに表示されます。以前、シーケンス内にあるビルボード ファイル名の番号は連続している必要がありました。たとえば、bbrd1.bmp、bbrd2.bmp、および bbrd4.bmp をインストール プロジェクトに追加した場合、bbrd1.bmp と bbrd2.bmp は表示されますが、bbrd4.bmp はシーケンスに bbrd3.bmp がないため表示されません。

詳細については、次を参照してください。

- InstallScript および InstallScript MSI インストールでビルボードを表示する
- InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前
- InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する

InstallScript 関数 PlayMMedia を使って Adobe Flash Application File (.swf) を再生できる機能

InstallScript 関数 **PlayMMedia** が、今回より、InstallScript と InstallScript MSI インストール中に背景ウィンドウで Adobe Flash application file (.swf) の再生をサポートします。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。

Flash ファイルを使用する場合、**SizeWindow** と **PlaceWindow** を使って、Flash ファイルを表示する背景ウィンドウのサイズと配置を制御できます。

詳細については、次を参照してください。

- PlayMMedia
- SizeWindow
- PlaceWindow
- InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する

InstallScript と InstallScript MSI インストール中、ダイアログにおける HTML コントロールをサポート

InstallShield では、InstallScript と InstallScript MSI プロジェクトに含まれるダイアログで HTML コントロールをサポートします。HTML コントロール を使って、ダイアログ コントロールで HTML マークアップを使用できます。ダイアログで Web ページ、インストール済みの HTML ファイル、および HTML サポートファイルへのリンクを含めることができます。エンド ユーザーが実行時のダイアログでハイパーリンクをクリックしたときに、インターネット ブラウザーで HTML ページを開くか、InstallScript コードを使って定義した別の動作をトリガーできます。HTML コントロールを使って、外観を制御するためのスタイルを含む、有効な HTML マークアップを使用できます。

HTML コンテンツがインストール済みの HTML ファイルまたは HTML サポート ファイルである場合、HTML コントロールを使って、それを直接ダイアログに表示することも可能です。

CtrlGetUrlForLinkClicked という名前の新しい InstallScript 関数が追加されました。この関数は、エンド ユーザーがクリックしたリンクへの URL を取得します。

詳細については、次を参照してください。

- ダイアログで HTML コントロールを使用する
- CtrlGetUrlForLinkClicked
- CtrlGetUrlForLinkClicked の例

サポートされていない言語を InstallScript プロジェクトに追加できる機能

InstallShield Premier Edition には、InstallScript プロジェクト用に 33 ヶ国語のデフォルト ランタイム文字列が含まれています。InstallShield Premier Edition では、新しい言語ウィザードを使って 33 ヶ国語以外のサポートされていない言語を InstallScript プロジェクトに追加することができます。サポートされていない言語を InstallScript プロジェクトに追加した後、[文字列エディター] ビューを使ってその言語用の翻訳された文字列を提供できます。別の手段として、言語の文字列エントリをファイルにエクスポートして、そのファイルの文字列値を翻訳してから、プロジェクトにそれをインポートすることもできます。以前、新しい言語ウィザードを使ってサポートされていない言語を追加できるのは、基本の MSI プロジェクトと InstallScript MSI プロジェクトのみでした。

詳細については、次を参照してください。

- InstallShield 実行時の言語サポート
- 新しい言語ウィザード
- [文字列エディター] ビュー

InstallScript コンパイラがリンクするライブラリのパスを指定できる機能

[設定] ダイアログ ボックスにある [コンパイル/リンク] タブに新しく追加された [追加ライブラリ パス] ボックスを使って、InstallScript コンパイラがスクリプトにリンクする InstallScript ライブラリ (.obl ファイル) を検索する場所を指定できます。さらに、このタブでカスタム ライブラリを指定するとき、フル パスではなく、ファイル名のみを指定できるようになりました。これらの変更によって、ライブラリを別のディレクトリに移動させた場合でも、[コンパイル/リンク] タブでライブラリのパスを手動で変更せずにスクリプトを正常にコンパイルすることができます。

詳細については、「[コンパイル/リンク] タブ」を参照してください。

新しい FlexNet Connect 11.6 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 11.6 をサポートします。InstallShield の [アップデート通知] ビューを使って、2 つの FlexNet Connect 11.6 マージ モジュールうち、いずれかが含まれています (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール)。これらのマージモジュールは、FlexNet Connect 11 マージ モジュールに取って代わります。

DIFx 2.1.1 サポート

InstallShield では、最新版の Driver Install Frameworks for Applications (DIFx 2.1.1) がサポートされています。Microsoft の最新バイナリ ファイルを含むこの新バージョンは、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで使用できます。 基本の MSI、InstallScript、InstallScript MSI

強化機能

InstallShield には、以下のような強化機能が搭載されています。

ユーザビリティの強化点

InstallShield の多くのビューは、その生産性とユーザビリティが強化されています。たとえば、一部のビューにはオプションを簡単に見つけられるツールバーが含まれています。いくつかのビューでは、グリッド内の行を整列する方法をカスタマイズできます。検索機能があるビューでは、より迅速に検索が行われます。主要な例は以下のとおりです:

- [ダイレクト エディター] ビュー – このビューでテーブルを選択すると、新しいツールバーが表示されます。ツールバーには、テーブル内でデータを追加、切り取り、コピー、貼り付け、検索、および置換できるボタンがあります。また、ビューではダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まないテーブル レコードを隠します。レコードを追加または編集するとき、InstallShield は各フィールドのヘルプ テキストを表示します。
- [プロパティ マネージャー] ビュー – このビューには、プロパティの追加および削除などの処理を行うためのボタンを含む新しいツールバーがあります。また、ビューではダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まないテーブル レコードを隠します。ビューの上部には新しいグループ ボックス領域があり、列ヘッダーをこの領域にドラッグ アンド ドロップすることで、ビュー内の行を階層形式に整列することができます。このビューでは、マウスと SHIFT または CTRL ボタンを使って複数のプロパティを選択してから、すべてを一斉に削除することができます。
- [再配布可能ファイル] ビュー – このビューに含まれる新しいツールバーとグループ ボックス領域は、強力な検索および組織化機能を提供します。列ヘッダーをグループ ボックス領域にドラッグ アンド ドロップして、再配布可能ファイルのリストを階層形式で表示することができます。また、ツールバーの検索ボックスに文字列を入力すると、InstallShield はその文字を含まないすべての再配布可能ファイルを隠します。
- [IIS 構成] ビュー – このビューは IIS 7 に似せて外観デザインが一新され、設定がタブではなくグリッドで表示されるようになりました。グリッドには、カテゴリ別、またはアルファベット順にグリッド設定を並べ替えるためのボタンがあります。このビューのグリッドの中から設定の 1 つを選択すると、右下のペインにその設定のヘルプ情報が表示されます。
- [一般情報] ビュー – このビューのすべての設定は、ビュー内でノードに関連付けられた個別のグリッドの代わりに、1 つのグリッドで表示されます。設定はいくつかのカテゴリにグループ分けされていて、特定の設定を見つけやすくなっています。また、カテゴリ別またはアルファベット順にグリッド設定を並べ替えることができるボタンもあります。以前、このビューに表示された文字列テーブルは [文字列エディター] ビューに移動しました。
- [パス変数] ビュー – このビューには、パス変数の追加および削除などの処理を行うためのボタンが含まれる新しいツールバーがあります。このビューでまた、ダイナミック検索もサポートされていて、検索ボックスに文字列を一文字づつ入力するのと同時に InstallShield がその文字を含まない行を隠します。ビューの上部には新しいグループ ボックス領域があり、列ヘッダーをこの領域にドラッグ アンド ドロップすることで、ビュー内の行を階層形式に整列することができます。このビューでは、マウスと SHIFT または CTRL ボタンを使って複数のパス変数を選択してから、すべてを一斉に削除することができます。

また、リリースをビルド、検証、またはスクリプトをコンパイルするときに表示される [出力] ウィンドウが強化されました。[出力] ウィンドウ、またその個別のタブは、InstallShield のワークスペースの任意の側面に固定するか、独立した場所にドラッグすることができます。[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端にドラッグすると、固定ウィンドウとして表示されます。[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端から離れた場所にドラッグすると、取り外されます。

詳細は、次を参照してください:

- 様々なビューで、[グループ ボックス] 領域を使って作業する
- [出力] ウィンドウを固定する/取り外す

アクションの進行状況メッセージの指定をサポート

エンド ユーザーに進行状況を伝えるために、インストールは一般的に、実行中の処理を説明する進行状況ダイアログで説明テキストを表示します。通常、インストールのステータスを表示する進行状況バーも同時に表示されます。標準アクションおよびカスタム アクションが発生すると、そのアクションについてのメッセージが進行状況ダイアログに表示されます。この機能は、実行するのに時間がかかるアクションで、特に役立ちます。インストールのログ記録が作成される場合、同じアクション テキストが記録されます。

[カスタム アクションとシーケンス] ビューに新しく追加された [アクション テキスト] 領域では、アクションの説明と詳細を指定できます。これは、基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクト タイプで使用できます。

詳細については、次を参照してください。

- アクション テキストを使う
- アクションの説明とアクション データのテンプレートを指定する
- 進行状況ダイアログにアクションの説明を表示する
- 進行状況ダイアログにアクション データを表示する

64 ビット システムが 32 ビット IIS アプリケーションの実行を許可するかどうかを検出できる機能

実行時、IIS 6 を持つシステム上でインストールが Enable32bitAppOnWin64 プロパティのチェックが必要な場合があります。そのとき、製品の要件およびチェック結果に基づいて、たとえば 32 ビット IIS 6 アプリケーションまたは 64 ビット IIS アプリケーションを含む特定のコンポーネントのインストールをスキップして、残りのファイル転送を続行できます。

InstallShield には、ターゲット システム上で Enable32bitAppOnWin64 プロパティがどのように設定されているかを検出できるサンプル Windows Installer DLL ファイルが含まれています。プロジェクトに、この DLL のカスタムアクションを追加できます。32 ビット アプリケーションが許可されている場合、Windows Installer プロパティ ISIIS6APPPOOLSUPPORTS32BIT は値 1 に設定され、許可されていない場合、このプロパティは設定されません。条件でこのプロパティをを使って、特定の動作をトリガーまたは阻止することができます。たとえば、ISIIS6APPPOOLSUPPORTS32BIT が設定されているかいないかによって、インストールが終了する起動条件を作成できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

プロジェクトにカスタム アクションを追加する方法についての詳細は、「64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮」を参照してください。

IIS 6 メタベースの互換性機能がインストールされているかどうかを検出できる機能

実行時に、インストールによって、ターゲット システムに IIS メタベースおよび IIS 6 構成との互換性機能がインストールされているかどうか、または IIS 以前がインストールされているかを検出できます。製品の要件と検出結果に基づいて、インストールを終了およびエラー メッセージを表示できます。

たとえば、Web サービス拡張は IIS 6 を持つシステムにインストールできます。IIS 7 を持つシステムでは、Web サービス拡張は IIS メタベースおよび IIS 6 構成との互換性機能がインストールされている場合のみインストール可能です。このため、IIS 6 が存在するか、IIS 6 との互換性機能がインストールされていることをインストールが検証するように構成することが必要な場合があります。これらのどちらの条件も False の場合、インストールが終了してエラー メッセージが表示されます。

InstallShield には、IIS メタベースおよび IIS 6 構成との互換性機能がインストールされているかどうかを検出するためのサンプル Windows Installer DLL ファイルが含まれています。プロジェクトに、この DLL のカスタムアクションを追加できます。If the IIS Metabase and IIS 6 Configuration Compatibility feature is installed, the Windows Installer property ISIISMETABASECOMPATPRESENT is set to a value of 1; if it is not installed, this property is not set.条件でこのプロパティをを使って、特定の動作をトリガーまたは阻止することができます。たとえば、ISIISMETABASECOMPATPRESENT が設定されているかいないかによって、インストールが終了する起動条件を作成できます。

この機能は、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

プロジェクトにカスタム アクションを追加する方法についての詳細は、「ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する」を参照してください。

名前空間プレフィックスを XML ファイル要素の属性と関連付ける機能

InstallShield では、今回より、名前空間プレフィックスを [XML ファイルの変更] ビューで属性に追加できるようになりました。そのためには、属性名の前にプレフィックスを入力して、コロン (:) を続けます。以前、属性に名前空間プレフィックスが含まれている場合、インストールが失敗しました。

この強化機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、「名前空間プレフィックスを属性に追加する」を参照してください。

ランタイム変更後に XML ファイルをフォーマットするかどうかを指定できる機能

[XML ファイルの変更] ビューで XML ファイルの [詳細] タブに追加された **[変更後に XML をフォーマットする]** チェック ボックスを使って、実行時にファイルが変更された後に XML ファイルをフォーマットするかどうかを指定できます。ファイルをフォーマットするとき、インストールはファイルにインデントを追加して、空白要素タグを開始タグと終了タグに置換します。これは、web.config ファイルでは問題を起こす可能性があるため、プロジェクトでこのチェック ボックスをクリアしなくてはならない場合もあります。

この強化機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプに適用します。

詳細については、「XML ファイルの [詳細] タブ」を参照してください。

.NET Framework および Internet Explorer 8 のための定義済みシステム検索

InstallShield に 2 つの新しい定義済みシステム検索が追加されました:

- Microsoft .NET Framework 3.5 SP1
- Internet Explorer 8

インストールでこれらのいずれかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この強化は、基本の MSI および InstallScript MSI プロジェクトに適用します。

カスタム プロパティを条件ビルダーに一覧表示

[条件ビルダー] ダイアログ ボックスのプロパティ リストには、今回より、[プロパティ マネージャー] ビューで追加されたプロパティが含まれます。また、システム検索ウィザードで作成された検索からのプロパティも含まれています。このため、InstallShield で条件を作成または編集するとき、独自のカスタム プロパティを手入力する代りに、リストから選択することができます。

この強化機能は、基本の MSI、InstallScript MSI、マージ モジュール および トランスフォーム プロジェクト タイプに適用します。

InstallShield 前提条件の設定で最大サービス パック番号と 64 ビットの場所を指定できる機能

InstallShield 前提条件エディターを使って InstallShield 前提条件の設定を追加または変更するときに表示される [前提条件設定] ダイアログ ボックスに、最大サービス パック番号を指定できるフィールドが新しく追加されました。以前は、最小サービス パック番号を指定できるだけで、最大サービス パック 番号を指定することはできませんでした。

また、[前提条件設定] ダイアログ ボックスでは、ファイル関連の条件を指定するときに表示されるボックスに 64 ビットの場所 ([CommonFiles64Folder]、[ProgramFiles64Folder]、および [System64Folder]) が含まれています。ファイル パスに 64 ビットの場所を使用する場合、これらのプロパティから選択できます。実行時に、ターゲット システムが 64 ビットの場合、インストールはこれらの 64 ビットの場所をチェックします。32 ビット システムの場合、インストールは対応する 32 ビットの場所をチェックします。

詳細については、「[前提条件設定] ダイアログ ボックス」を参照してください。

MSBuild サポートの強化点

MSBuild の InstallShield タスクには、次の新しいパラメーターが追加されました:

- **RunMsiValidator** — 検証に使用する .cub ファイルを指定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldMsiValidators として露出されます。
- **PatchConfiguration** — MSBuild を使ってビルドするパッチ構成を指定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、プロパティ InstallShieldPatchConfiguration として露出されます。
- **PathVariables** — パス変数の値をオーバーライドします。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。
- **PreprocessorDefines** — InstallScript をコンパイルするためのプリプロセッサ定義を指定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPreprocessorDefines として露出されます。

詳細については、「Microsoft ビルド エンジン (MSBuild)」を参照してください。

オートメーション インターフェイスの強化

Windows 7、Windows Server 2008 R2、Windows Vista、Windows Server 2008、および Windows Server 2003 の OSFilter 値

オートメーション インターフェイスで、ISWiComponent と ISWiRelease オブジェクトの OSFilter メンバーと共に、以下の定数を使用できます:

- eosWin7 (33554432) — Windows 7 および Windows Server 2008 R2 用
- eosWinVista (16777216) — Windows Vista および Windows Server 2008 用
- eosWinServer2003 (8388608)

また、eosAll の値は 5308624 から 64028880 に変更されました。

OSFilter メンバーは、InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトで ISWiComponent オブジェクトに適用します。OSFilter メンバーは、InstallScript、および InstallScript オブジェクト プロジェクトで ISWiRelease オブジェクトに適用します。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

新しい IsPlatformSelected プロパティ値

ISWiComponent オブジェクトの IsPlatformSelected プロパティで使用できるプロパティ値のリストが拡張されました。このプロパティには、32-bit、64-bit Itanium、AMD64、および Windows Server 2003 R2 の値が含まれます。これは、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクト タイプに適用します。

既存の定数の一部の値が変更されています。ご注意ください。新しい値についての詳細は、「ISWiComponent オブジェクト」を参照してください。

SQL スクリプトを接続に追加する新しい AddSQLScriptEx メソッド

ISWiSQLConnection オブジェクト に AddSQLScriptEx メソッドが追加されました。このメソッドを使って ISSQLScriptFile エントリを追加し、渡された文字列から有効な名前を生成します。このメソッドは、ISSQLScriptFile テーブルのエントリ名が一意であること、またその文字数が 47 文字未満であることを保証します。

ISWiSQLScript オブジェクトの新しい RunOnLogon および Condition プロパティ

ISWiSQLScript オブジェクト に読み取り RunOnLogon プロパティが追加されました。このプロパティは、[SQL スクリプト] ビューにある SQL スクリプトの [ランタイム] タブにある [ログイン中にスクリプトを実行] チェック ボックスに対応します。

ISWiSQLScript オブジェクトにはまた、読み取り/書き込み Condition プロパティが追加されました。このプロパティは、SQL スクリプトをインストールまたはアンインストール中に実行するかどうかを判断するために実行時に評価される条件を指定します。この条件が True と評価された場合に、スクリプトが実行します。このプロパティは、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

ISWiUpgradeTableEntry オブジェクトの新しい DisplayName プロパティ

ISWiUpgradeTableEntry オブジェクト に読み取り DisplayName プロパティが追加されました。このプロパティは、アップグレード エントリの名前を設定または取得します。これは [アップグレード] ビューのアップグレード アイテムに表示される内部名です。このプロパティは、基本の MSI および InstallScript MSI プロジェクト タイプで使用できます。

オペレーティング システムに関する InstallScript 言語の強化

次の構造メンバーと定義済み定数が InstallScript 言語に追加されました:

- **SYSINFO.WINNT.bWin7_Server2008R2** – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows 7 または Windows Server 2008 R2 の場合、この値は TRUE です。
- **SYSINFO.bWinServer2003R2** – 新しい SYSINFO 構造メンバーです。オペレーティング システムが Windows Server 2003 R2 の場合、この値は TRUE です。(このオペレーティング システムでは、SYSINFO.WINNT.bWinServer2003 の値も TRUE です。)

- ISOSL_WIN7_SERVER2008R2 – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しい定義済み定数です。これは、ターゲット システムが Windows 7 または Windows Server 2008 R2 を実行中であることを示します。
- ISOS_ST_SERVER2003_R2 – **FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できる新しいオペレーティング システム スイートです。これは、ターゲット システムが Windows Server 2003 R2 を実行中であることを示します。

また、一部のアイテムの名前が変更されました:

- SYSINFO.WINNT.bWinVista 構造メンバーは Windows Vista または Windows Server 2008 を区別しません。したがって、新しいメンバー SYSINFO.WINNT.bWinVista_Server2008 が追加されました。古いエイリアスも引き続き使用できますが、コードを明確にするため、新しいメンバーが推奨されます。
- ISOSL_WINVISTA 定義済み定数は Windows Vista または Windows Server 2008 を区別しません。したがって、新しい定数 ISOSL_WINVISTA_SERVER2008 が追加されました。古いエイリアスも引き続き使用できますが、コードを明確にするため、新しいメンバーが推奨されます。

詳細については、「FeatureFilterOS 関数と SYSINFO 構造変数」を参照してください。

[InstallScript] ビューで InstallScript ダイアログ ソース コードを容易にオーバーライドできる機能

[InstallScript] ビューのイベント カテゴリ ドロップダウン リストに、新しい Dialog Source オプションが追加されました。このオプションを選択すると、イベントハンドラー ドロップダウン リストには、すべての InstallScript ダイアログがリストされます。このリストから任意のダイアログを選択して、そのコードを変更できます。

この機能は、InstallScript 、InstallScript MSI、および InstallScript オブジェクト プロジェクト タイプに適用します。

Dialog Source オプションを選択すると使用可能となる ビルトイン InstallScript ダイアログ 関数についての詳細は、「ダイアログ関数」を参照してください:

InstallScript インストールのアンインストール キーにおけるメジャーおよびマイナー バージョンのレジストリ エントリの変更

InstallScript インストールは、アンインストール キーに VersionMajor および VersionMinor レジストリ値を作成します。これらの値の名前は、今回より、基本の MSI および InstallScript MSI インストール中に作成されるエントリの名前に一致します。これは、InstallScript 2010 で作成された新しいインストール、および InstallShield 2009 以前からアップグレードされたインストールに適用します。以前、InstallShield 2009 以前では、InstallScript インストールが作成する値の名前は MajorVersion および MinorVersion でしたが、今後は作成しません。

新しいレジストリ値を使用するため、次の InstallScript 定数の値が変更されました:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION は、今回より、MajorVersion ではなく VersionMajor です。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION は、今回より、MinorVersion ではなく VersionMinor です。

MaintenanceStart 関数が呼び出されると、レジストリに更新された値名が作成されます。デフォルトで、古い値名が存在するとき、それが削除されます。ターゲット システムから古い値名を削除したくない場合、REGDB_OPTION_NO_DELETE_OLD_MAJMIN_VERSION という名前の新しい REGDB_OPTIONS オプションを使用します。

RegDBGetItem 関数と共に REGDB_UNINSTALL_MAJOR_VERSION または REGDB_UNINSTALL_MINOR_VERSION を使用すると、**RegDBGetItem** は、まず新しい値をチェックします。新しい値が検出されると、関数は新しい値から値データを返します。新しい値が検出されなかった場合、関数は自動的に古い値をチェックします。古い値が検出されると、関数は古い値から値データを返します。

下位互換性を目的として、次の新しい定数が提供されています:

- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD

これらの定数を **RegDBGetItem**、**RegDBSetItem**、および **RegDBDeleteItem** 関数と共に指定して、古い値を取得、設定、および削除できます。

次の新しい文字列定数も使用できます:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD は、MajorVersion として定義されています。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD は、MinorVersion として定義されています。

詳細は、次を参照してください:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION
- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD
- REGDB_VALUENAME_UNINSTALL_MINORVERSION
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD
- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD
- REGDB_OPTIONS
- RegDBSetItem
- RegDBGetItem
- RegDBDeleteItem

カスタム InstallScript ダイアログ内のコントロールのウィンドウ ハンドルを取得するための、新しい CtrlGetDlgItem 関数

CtrlGetDlgItem という名前の新しい InstallScript 関数が追加されました。**CtrlGetDlgItem** は、カスタム ダイアログ内のコントロールのウィンドウ ハンドルを取得します。**CtrlGetDlgItem** は Windows API **GetDlgItem** と似ていますが、**CtrlGetDlgItem** の場合、ダイアログのウィンドウ ハンドルの代わりに InstallScript ダイアログ名を指定できます。詳細については、「CtrlGetDlgItem」を参照してください。

InstallScript 文字列のポインターを外部 DLL 関数に渡す、新しい InstallScript 定数

IS_NULLSTR_PTR 変数を使って、ヌル ポインターを InstallScript 文字列としてプロトタイプされているパラメーターを通して 外部 DLL 関数または Windows API に渡すことができます。この機能は byval 文字列、byref 文字列、wstring、およびバイナリ データ タイプで使用できます。詳細については、「IS_NULLSTR_PTR」を参照してください。

InstallScript サイズ単位定数を表示文字列に変換する、新しい StrConvertSizeUnit 関数

StrConvertSizeUnit という名前の新しい InstallScript 関数が追加されました。**StrConvertSizeUnit** 関数は、指定された InstallScript サイズ単位定数の適切な表示文字列を返します。詳細は、「StrConvertSizeUnit」を参照してください。

文字列から先頭と行末の空白およびタブを削除する、新しい StrTrim 関数

StrTrim という名前の新しい InstallScript 関数が追加されました。**StrTrim** 関数は、文字列から先頭と行末の空白およびタブを削除します。詳細については、「StrTrim」を参照してください。

既存の SdLicense* ダイアログ関数に優先する新しい SdLicense* ダイアログ関数

新しい 2 つの InstallScript ダイアログ 関数 (**SdLicenseEx** と **SdLicense2Ex**) が追加されました。どちらの関数も、複数行編集フィールドに使用許諾契約書が記載されたダイアログを表示します。使用許諾契約書は、テキスト ファイル (.txt) またはリッチ テキスト ファイル (.rtf) で保存できます。

- SdLicenseEx は、質問をスタティック フィールドに表示するダイアログを表示します。エンド ユーザーは、[はい] または [いいえ] ボタンをクリックして質問に答えます。

  SdLicenseEx は、**SdLicense** と **SdLicenseRtf** よりも優先します。

- SdLicense2Ex は、2 つのラジオ ボタンを持つダイアログを表示します (使用許諾契約書の条件に同意するためのボタンと、同意しないためのボタン)。エンド ユーザーが使用許諾契約書の条件に同意するための適切なボタンをクリックすると、[次へ] ボタンが有効になります。

  SdLicense2Ex は、**SdLicense2** と **SdLicense2Rtf** に優先します。.

キーと値のペアー覧を検索するための、新しい ListFindKeyValueString 関数

ListFindKeyValueString という名前の新しい InstallScript 関数が追加されました。**ListFindKeyValueString** 関数は、指定された値の文字列または数値リストを検索します。最初のリストで見つかった文字列の位置に対応する追加リストから値が返されます。これによって、キーと値のペアー覧で特定のキーを検索して、対応する値を取得できます。

詳細については、「ListFindKeyValueString」を参照してください。

新しい InstallScript コードの例

InstallShield ドキュメントには、次の InstallScript 機能のためのサンプル コードが含まれています:

- AdminAskPath
- CharReplace
- FormatMessage
- LogReadCustomNumber
- LogReadCustomString
- LogWriteCustomNumber
- LogWriteCustomString

このコードを InstallShield ドキュメントからコピーして、InstallScript コードに貼り付け、必要に応じてカスタマイズできます。

InstallShield キャビネット ファイル ビューアーの拡張機能

キャビネット ファイル ビューアーでは、InstallScript プロジェクト用の .cab ファイルについての追加情報が提供されています。

- InstallShield でビルドされた機能について、ビューアーは "パスワード保護"、"前を分割"、"後を分割"、"分割なし"、および "前を分割しない" フィールドとイメージ インデックスを表示します。

- InstallScript インストールに含まれる InstallScript オブジェクトについて、ビューアーは新しい "オブジェクト バージョン" フィールドを表示します。
- コンポーネントについて、ビューアーは新しい "暗号化"、"データ ファイル"、および ".NET アセンブリ" フィールドを表示します。
- メディアについて、ビューアーは新しい "実行可能ファイル" フィールドを表示します。

InstallShield 2009 SP2 の新しい機能

InstallShield 2009 Service Pack 2 (SP2) には次の変更が含まれています:

Microsoft Visual Studio 2008 SP1 サポート

InstallShield は、今回より以下に記述されているとおり、Microsoft .NET Framework 3.5 SP1 と Microsoft SQL Server 2008 をサポートします。

また、InstallShield と Visual Studio との統合では Visual Studio 2008 SP1 がサポートされており、Visual Studio 内でインストールおよび製品の開発を行うことができます。

新しい Microsoft .NET Framework 3.5 SP1 前提条件

InstallShield には、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい 2 つの InstallShield 前提条件が含まれています。

- Microsoft .NET Framework 3.5 SP1 (Web ダウンロード版)
- Microsoft .NET Framework 3.5 SP1 (完全パッケージ)

詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

Microsoft SQL Server 2008 サポート

InstallShield は、今回より、SQL Server 2008 上で SQL スクリプトを実行するためのサポートを含みます。また、InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストには、SQL Server 2008 が含まれています。

新しい Microsoft SQL Server 2008 Express 前提条件

InstallShield に Microsoft SQL Server 2008 Express Edition の InstallShield 前提条件が追加されました。これらの InstallShield 前提条件は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加できます。

これらの新しい前提条件をサポートするため、InstallShield 前提条件エディターに新しいレジストリ条件タイプが追加されました。

InstallShield 前提条件エディターに追加された新しいレジストリ条件タイプ

InstallShield 前提条件エディターに、新しく [**レジストリ エントリが指定のバージョン値を含む**] 条件オプション タイプが追加されました。この条件タイプを使って、ターゲット システム上で特定のレジストリ エントリの値データとして格納されているバージョン番号が、InstallShield 前提条件エディターで指定されたバージョン番号よりも大きいか、小さいか、または等しいかをインストールがチェックするように設定できます。この新しい条件の詳細については、「[前提条件設定] ダイアログ ボックス」を参照してください。

新しい Microsoft SQL Server 2005 Express SP2 (x86 と x64 WOW) 前提条件

InstallShield に Microsoft SQL Server 2005 Express Edition SP2 の InstallShield 前提条件が追加されました。この前提条件は Microsoft SQL Server 2005 Express Edition SP2 を 32 ビットおよび 64 ビット (WOW) システムにインストールします。この InstallShield 前提条件は、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加できます。

新しい FlexNet Connect 11.0.1 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで Connect 11.0.1 のサポートを含みます。InstallShield の [アップデート通知] ビューで、2 つの FlexNet Connect 11.0.1 マージ モジュール (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール) のどちらかを含みます。これらのマージモジュールは、FlexNet Connect 11 マージ モジュールに取って代わります。更新されたマージ モジュールの変更点についての詳細は、FlexNet Connect のリリース ノートを参照してください。

追加の変更

InstallShield 2009 SP2 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2009 SP1 の新しい機能

InstallShield 2009 は、Windows Installer 4.5 ベータ版をサポートします。InstallShield 2009 Service Pack 1 (SP1) には、Windows Installer 4.5 の最終リリース版のサポートを提供する変更が含まれています。

Windows Installer 4.5 用 InstallShield 前提条件

InstallShield は、Windows Installer 4.5 再配布可能ファイルの最終リリース バージョン用の以下の InstallShield 前提条件を含みます。

- Windows Vista と Server 2008 (x86) 用 Windows Installer 4.5
- Windows Vista と Server 2008 (x64) 用 Windows Installer 4.5
- Windows Vista と Server 2008 (IA64) 用 Windows Installer 4.5
- Windows Server 2003 SP1 以降 (x86) 用 Windows Installer 4.5
- Windows Server 2003 と XP (x64) 用 Windows Installer 4.5
- Windows Server 2003 (IA64) 用 Microsoft Windows Installer 4.5
- Windows XP SP2 以降 (x86) 用 Windows Installer 4.5

実行時に Windows Installer 4.5 をインストールする場合、これらの InstallShield 前提条件を 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加できます。

詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

追加の変更

InstallShield 2009 SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 2009 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

InstallShield 前提条件を機能と関連付けて、インストールを連鎖させる機能

InstallShield では、InstallShield 前提条件を 1 つまたは複数の機能と関連付けられるようになりました。この新しい種類の InstallShield 前提条件は、*機能前提条件*と呼ばれます。機能前提条件は、前提条件を含む機能がインストールされたときに、その前提条件がシステム上に既にインストールされていない場合にインストールされます。

プロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖することができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。Setup.exe セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。

[再配布可能ファイル] ビューを使って、InstallShield 前提条件をプロジェクトに追加して、その前提条件をメイン インストールの前に実行するのか、メイン インストールに含まれる 1 つまたは複数の機能と関連付けるのかを指定できます。

以前は、すべての InstallShield 前提条件インストールが、メイン インストールの実行前に実行されたため、InstallShield 前提条件を機能に関連付けることはできませんでした。この種類の前提条件は今後も使用できますが、今回より、*セットアップ前提条件*と呼ばれます。

基本の MSI プロジェクトが、この機能のサポートを含みます。

詳細は、次を参照してください:

- セットアップ前提条件と機能前提条件の違い
- 基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける
- InstallShield 前提条件を含むインストールの実行時の動作
- インストール プロジェクトに含まれている InstallShield 前提条件を利用する

Windows Installer 4.5 Beta におけるトランザクション処理を使った複数パッケージのインストールを可能にする機能のサポート

InstallShield を使って、基本の MSI と InstallScript MSI プロジェクトに Windows Installer パッケージを連鎖 .msi パッケージとして追加できます。基本の MSI または InstallScript MSI インストールに連鎖 .msi パッケージが含まれていて、ターゲットシステムに Windows Installer 4.5 が存在するとき、Windows Installer はトランザクション処理を使って複数のパッケージをインストールします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

[リリース] ビューの [連鎖 .msi パッケージ] 領域で、メイン インストールに連鎖させる 1 つまたは複数の .msi パッケージをプロジェクトに追加できます。この領域ではまた、連鎖 .msi パッケージにリリース フラグを割り当てたり、連鎖されたパッケージに渡す必要があるプロパティなどの設定を構成したり、条件を指定したりできます。

詳細については、次を参照してください。

- トランザクション処理を使って複数パッケージ インストールを構成する
- トランザクション処理を使った複数パッケージ インストールについての概要

- 新しい連鎖 .msi パッケージをプロジェクトに追加する
- 連鎖 .msi パッケージの設定

Windows Installer 4.5 Beta における InstallScript MSI インストールで InstallScript エンジンを埋め込みユーザー インターフェイスとして使用できる機能のサポート

InstallScript MSI プロジェクトでは、InstallScript エンジンを埋め込みカスタム ユーザー インターフェイス (UI) ハンドラーとして使用するオプションが追加されました。従来、InstallScript エンジンは外部カスタム UI ハンドラーとして使用されていました。Windows Installer 4.5 は、この新しい機能をサポートします。この新しい埋め込みオプションを使用すると、エンド ユーザーがインストールを .msi パッケージから直接起動することができると共に、InstallScript で定義された高度にカスタマイズされたユーザー インターフェイスを含んだインストールを作成できます。

この新しい埋め込み InstallScript UI 機能と、従来の外部 InstallScript UI 機能のどちらを使用するかを指定するには、新しい "InstallScript ユーザー インターフェイスの種類" 設定を使います。この設定は [一般情報] ビューにあり、プロジェクト全体に反映します。デフォルトで、InstallShield はすべての InstallScript MSI プロジェクトで従来のスタイルを使用します。この従来型のオプションには、Setup.exe セットアップランチャーが必要です。

新しく INSTALLSCRIPTMSIEEUI 変数が追加されました。新しいスタイルが使用されるとき、この変数は実行時に True に設定されます。それ以外の場合は False に設定されます。

新しいスタイルには、従来のスタイルには適用されなかった制限事項がいくつかありますので、ご注意ください。たとえば、一部の InstallScript 関数およびコマンドライン パラメーターは、新しいスタイルでは、サポートされていないか、または動作が異なるものがあります。2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

追加の情報については、次を参照してください:

- InstallScript MSI インストールの InstallScript ユーザー インターフェイス タイプを指定する
- InstallScript MSI インストール プロジェクト
- INSTALLSCRIPTMSIEEUI

共有コンポーネントのパッチにおける Windows Installer 4.5 ベータ サポート

InstallShield では、Windows Installer 4.5 で提供されている新しい共有コンポーネントのパッチ機能がサポートされています。[コンポーネント] ビューの新しい "複数パッケージの共有コンポーネント" 設定を使って、選択したコンポーネントについて共有コンポーネントのパッチを有効にするかどうかを指定できます。[はい] を選択すると、新しいコンポーネントの属性が設定されます。この属性は、複数のパッケージ間で共有されているコンポーネントを含むパッチがアンインストールされるときに、Windows Installer がファイルをダウングレードすることを阻止します。これによって、最新バージョンを含むパッチがアンインストールされた場合でも、ターゲット システム上に常にコンポーネントの最新バージョンが保持されます。このオプションのデフォルト値は [いいえ] です。

Windows Installer 4.5 はこの新しい機能をサポートします。Windows Installer の以前のバージョンは、この新しい設定を無視します。また、ターゲット システムで DisableSharedComponent ポリシーが 1 に設定されている場合、Windows Installer はすべてのパッケージについて、この設定を無視します。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール、およびトランスフォーム プロジェクトに適用します。

詳細については、「コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する」を参照してください。

Windows Installer 4.5 Beta における優先コンポーネント機能のサポート

[コンポーネント] ビューに追加された新しい "置き換えられたコンポーネントのアンインストール" 設定を使って、特定の条件下で優先するパッチのインストール中に、選択されたコンポーネントを Windows Installer 4.5 が処理する方法を指定できます。現在のパッチに含まれるこのコンポーネントをアンインストール用にフラグすることで、優先するパッチが適用された後にターゲット システム上でこのコンポーネントが孤立しないようにするためには、[はい] を選択します。後続のパッチがインストールされ、またそれが最初のパッチよりも優先されることがフラグされている場合、適切な場合に Windows Installer はこのコンポーネントを登録解除およびアンインストールします。[いいえ] を選択した場合、優先するパッチはターゲット マシン上に孤立したコンポーネントを残す可能性があり、後に残された機能のコンポーネントの管理は一切行われません。この設定のデフォルト値は [いいえ] です。

Windows Installer 4.5 はこの新しい機能をサポートします。Windows Installer の以前のバージョンは、この新しい設定を無視します。

この機能は、基本の MSI、InstallScript MSI、およびマージ モジュール、およびトランスフォーム プロジェクトに適用します。

さらに詳しい情報は、「"置き換えられたコンポーネントのアンインストール" 設定」の説明を参照してください。

パッチ シーケンスに関する詳しい情報は、「パッチ シーケンス」をご覧ください。

Windows Installer 4.5 Beta におけるパッチのアンインストール時のみカスタム アクションを実行できる機能のサポート

カスタム アクションのための新しい "パッチのアンインストールのみ実行" 設定を利用して、Windows Installer が選択されたカスタム アクションをパッチのアンインストール時のみ実行するかどうかを指定できます。この設定は、基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの [カスタム アクションとシーケンス] ビューにあります。マージ モジュール プロジェクトおよび MSM データベース プロジェクトでも提供されています。

また、カスタム アクション ウィザードの [追加のオプション] パネルにある [パッチのアンインストール時に実行する] チェック ボックスを利用しても、カスタム アクションの動作を構成することができます。

この新しい機能については、「パッチのアンインストール時に実行」をご覧ください。

Windows Installer 4.5 再配布可能ファイルのベータ サポート

InstallShield には、Windows Installer 4.5 用のいくつかの InstallShield 前提条件ファイル (.prq) が含まれています。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

Setup.exe および Update.exe ブートストラッパの Unicode バージョンを作成する機能

InstallShield では、基本の MSI プロジェクトで Setup.exe セットアップ ランチャーを Unicode バージョンまたは ANSI バージョンのどちらで作成するかを指定できます。以前、基本の MSI プロジェクトにセットアップランチャーが含まれているとき、常に ANSI バージョンがビルドされていました。Unicode バージョンのビルドはサポートされていませんでした。

Unicode セットアップランチャーは、ターゲット システムで 2 バイト言語のための適切なコード ページが実行されているいないにかかわらず、セットアップランチャーのユーザー インターフェイスで 2 バイト文字を正しく表示することができます。ANSI セットアップランチャーは、ターゲット システムで適切なコード ページが実行されている場合のみ、セットアップランチャー ダイアログで 2 バイト文字を正しく表示します。適切なコード ページが実行されていない場合、これらのダイアログで 2 バイト文字が文字化けして表示されます。

[リリース] ビューでリリースについて表示される Setup.exe タブに新しく追加された "最短初期化時間" 設定を利用して、Unicode を使うか、または ANSI を使うかを指定することができます。すべての基本の MSI プロジェクトでは、Unicode がデフォルトとして使用されます。

パッチまたは QuickPatch パッケージについて Update.exe アップデート ランチャを作成するときも、Unicode バージョンで作成するか、または ANSI バージョンで作成するかを指定することができます。

[パッチのデザイン] ビューでパッチ構成について表示される Setup.exe タブに新しく追加された "アップデート ランチャの種類" 設定を利用して、Unicode を使うか、または ANSI を使うかを指定することができます。QuickPatch プロジェクトでは、"アップデート ランチャの種類" 設定は [一般情報] ビューの [ビルド設定] 領域の [詳細] タブにあります。すべての新しいパッチ構成と QuickPatch パッケージでは、Unicode がデフォルトとして使用されます。

Setup.exe ブートストラッパのログファイルを作成する機能

基本の MSI プロジェクトと InstallScript MSI プロジェクトで、新しい /debuglog コマンドライン パラメーターが Setup.exe セットアップランチャーに追加されました。このコマンドライン パラメーターを使用して、デバッグ用のログ ファイルを生成することができます。詳細については、「/debuglog」を参照してください。

マネージコード カスタム アクションのサポート

InstallShield では、マネージ コード カスタム アクションを 基本の MSI プロジェクト、InstallScript MSI プロジェクト、マージ モジュール プロジェクトに追加することができます。この種類のカスタム アクションは、Visual Basic .NET または C# などのマネージ コードで書かれた .NET アセンブリ内にあるパブリック メソッドを呼び出します。

詳細については、次を参照してください。

- マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- マネージコード カスタム アクションの実行時要件
- アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する
- カスタム アクションの設定を構成する

製品の複数インスタンスをインストールするためのサポート

InstallShield では今回より、.msi パッケージを使って同じマシン上に同じコンテキストで製品の複数インスタンスをインストールすることができるインストールの作成をサポートします。[リリース] ビューの製品構成に追加された新しい [複数インスタンス] タブを使って製品の異なるインスタンスを定義し、各インスタンスの関連プロパティを構成することができます。

ビルド時、InstallShield は各インスタンスに対して製品コードを変更するためのインスタンス トランスフォームを作成し、それを .msi パッケージにストリームします。実行時、セットアップランチャーは [新しいインスタンスの選択] ダイアログを表示します。このダイアログでエンド ユーザーは新しいインスタンスをインストールするか、インストール済みインスタンスのアップデートまたはメンテナンスを行うかを指定することができます。

さらに今回より、[パッチのデザイン] ビューでインストールに複数インスタンス サポートを含む製品用のパッチをビルドすると、エンド ユーザーが特定のインスタンスまたはすべてのインスタンスを更新することができるパッチが作成されます。実行時、Update.exe ファイルはパッチ バージョンの [インスタンスの選択] ダイアログを表示します。

Setup.exe と Update.exe に新しい /instance コマンドライン オプションが追加されました。このオプションを使って、インストール、更新、またはアンインストールするインスタンスを指定することができます。また、[インスタンスの選択] ダイアログを抑制することも可能です。

複数インスタンスは、基本の MSI プロジェクトでサポートされています。

詳細については、次を参照してください。

- 製品の複数のインスタンスをインストールする
- 複数インスタンス サポートの実行時要件
- InstallShield で複数インスタンスを構成する
- 複数インスタンス サポートに関する特別考慮
- 複数インスタンスのサポートを含むリリースの構成とビルド
- 製品の複数インスタンスをインストールするためのサポート
- 製品のインスタンスを複数インストールするときの実行時の動作
- セットアップランチャーの作成
- 製品構成の [複数インスタンス] タブ
- Setup.exe および Update.exe コマンドライン パラメーター

新しい Microsoft .NET 再配布可能ファイル

InstallShield には今回、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる .NET 関連の新しい InstallShield 前提条件が多数含まれています。

- Microsoft .NET Framework 3.5 (Web ダウンロード版)
- Microsoft .NET Framework 3.5 (完全パッケージ)
- Microsoft .NET Framework 3.5 言語 パック (x86、x64、IA64)
- Microsoft .NET Framework 3.0 SP1 (Web ダウンロード版)
- Microsoft .NET Framework 3.0 言語パック
- Microsoft .NET Framework 2.0 SP1 (x86、x64、IA64)

また InstallShield では、InstallScript プロジェクト用のアップデートされた Microsoft .NET Framework オブジェクトも利用できます。このオブジェクトには、32 ビット、64 ビット x64、および 64 ビット Itanium バージョンを含む、バージョン 3.5、3.0 SP1、3.0、2.0 SP1、2.0、1.1 SP1、および 1.0 SP3 の .NET Framework のサポートが含まれています。オブジェクトには、Web ダウンローダ再配布可能ファイルだけでなく、利用可能なサポート対象の言語パックも含まれています。

詳細については、次を参照してください。

- .NET Framework 再配布可能ファイルをプロジェクトへ追加する
- Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める
- Microsoft .NET Framework オブジェクト ウィザード

新しい .msi パッケージ ツール

InstallShield には、いくつかの新しいツールが含まれています:

- InstallShield MSI Diff を使って、2 つの .msi、.msm、または .pcp ファイルを素早く比較できます。これを使って、1 つの .msi ファイルに対して 1 つまたは複数の .msp と .mst ファイルを適用し、その結果となる .msi

データベースの変更を確認することができます。また、このツールを使って、バイナリ形式で保存された 2 つの InstallShield プロジェクト ファイル (.ism または .ise) を比較することができます。このツールでは、追加、変更、削除、スキーマの違いを示すためにカラー コードが使用されます。このツールは、ほとんどのソース コード管理システムに簡単に統合できます。

- InstallShield MSI Query を使って、SQL ステートメントをビルド スクリプトで実行する前に、SQL の Windows Installer バージョンを用いてステートメントをテストすることができます。SQL ステートメントが正しくフォーマットされているかを素早く確認でき、また生成される結果を参照できます。
- InstallShield MSI Sleuth は、ターゲット システムの現在のインストール済み状態を参照できる診断ツールです。InstallShield MSI Sleuth はインストール済みのすべての .msi パッケージの一覧を表示します。リストの中から任意の .msi パッケージをクリックすると、データベース内のテーブルやバイナリ ストリームだけでなく、その機能とコンポーネントの状態および既知のソースの場所を参照できます。このツールは、特定のコンポーネント コードを持つパッケージを含むインストール済みの製品 (複数可) を識別するのにも役立ちます。
- InstallShield MSI Grep は、.msi と .msm パッケージのコレクションの中から特定のテキストを検索します。特定のテーブルまたは列に関する結果のみを表示するための詳細検索も可能です。

これらすべてのツールは、Windows [スタート] メニューの [InstallShield ツール] サブフォルダーから起動できます。

これらの InstallShield MSI ツールは、InstallShield Premier Edition および Professional Edition に含まれています。Premier Edition には、別のマシンに InstallShield 以外のツールのみをインストールできる、個別のインストールおよび追加ライセンスが含まれています。特定の条件については、InstallShield エンド ユーザー使用許諾契約書を参照してください。

詳細については、「InstallShield MSI ツール」を参照してください。

Setup.exe と ISSetup.dll にストリームされるファイルを圧縮し、圧縮レベルを指定できる機能

Setup.exe セットアップランチャーまたは ISSetup.dll ファイルを使用するリリースをビルドしたとき、InstallShield によって、Setup.exe ファイルまたはISSetup.dll ファイルにストリームされるファイルが圧縮されるようになりました。InstallShield が使用するデフォルトの圧縮レベルは、ファイルのサイズと実行時に圧縮ファイルを展開するために必要な時間のバランスをとっての目安です。圧縮レベルを変更する場合、または圧縮をしない場合、マシン全体に適用する設定を利用してデフォルトのレベルをオーバーライドすることができます。

デフォルトで、ビルド時に InstallShield がファイルを Setup.exe ファイルにストリームするとき、.cab ファイル拡張子を持つファイルは圧縮されません。これは、.cab ファイルが既に圧縮されているファイルであるためです。デフォルトの圧縮除外一覧を変更して、他の種類のファイルや特定のファイルを必要に応じて選択することができます。除外一覧は、マシン全体に適用される設定です。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。

詳細については、「Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する」を参照してください。

マルチパート .cab ファイルのサポート

.cab ファイルには、いくつかの制限事項があります。たとえば、単一 .cab ファイルの最大サイズは 2 GB に設定されています。また、サイズの大きい .cab ファイルを署名しようしたとき、およびサイズの大きい署名済み .cab ファイルのデジタル署名を検証しようとしたときにトラブルが生じた経験があるユーザーもいるかもしれません。今回より、これらの制限事項を回避するために、.cab ファイルのデフォルトの制限が 600 MB に設定されました。InstallShield で、リリースの .cab ファイルを作成しているとき、この制限に達すると、データが 2 つ以上の .cab ファイルに分割され、マルチパートの .cab ファイルが作成されます。

最大サイズは必要に応じて変更することができます。InstallShield でマルチパート .cab ファイルを作成しない場合、単一の .cab ファイルを作成するように構成できます。

この機能は、基本の MSI および InstallScript MSI プロジェクトに適用します。また、この機能は、すべてのファイルが単一ファイルの .msi パッケージまたは Setup.exe セットアップランチャーに埋め込まれている圧縮済みネットワーク イメージ リリースをビルドしている場合のみ適用します。この機能は、1 つまたは複数の機能に関連付けられているファイルのみが .cab ファイルに圧縮されるカスタム圧縮には適用しません。

詳細については、「.cab ファイルの最大サイズを構成する」を参照してください。

基本の MSI インストールに [ファイルを開く] ダイアログを追加して、エンド ユーザーによるファイルの参照を可能にするサポート

InstallShield は、基本の MSI インストールのダイアログの 1 つから [ファイルを開く] ダイアログを起動するサポートを含みます。エンド ユーザーがダイアログの 1 つから [参照] ボタンをクリックすると、[ファイルを開く] ダイアログが起動します。[ファイルを開く] ダイアログを使って、エンド ユーザーはファイルを参照できます。エンド ユーザーがファイルを選択して [開く] ボタンをクリックすると、[ファイルを開く] ダイアログが閉じて、インストールがダイアログの編集フィールドにその完全パスとファイル名を書き込みます。インストールはまた、IS_BROWSE_FILEBROWSED プロパティの値を、エンド ユーザーが選択したファイルのパスとファイル名に設定します。

この機能を利用するには、プロジェクトに FileBrowse という名前の Windows Installer DLL カスタム アクションを追加する必要があります。このカスタム アクションは FileBrowse.dll ファイルを呼び出します。さらに、[ファイルを開く] ダイアログを起動する編集フィールド コントロールと [参照] ボタンを追加して、関連ダイアログイベントを設定しなくてはなりません。詳しい手順については、「[ファイルを開く] ダイアログを起動する」を参照してください。

Windows Server 2008 上に IIS 7 Web サイトをインストールするためのサポート

InstallShield では、今回より Windows Server 2008 システム上で IIS 7 Web サイト、仮想ディレクトリ、Web サービス拡張、およびアプリケーション プールの作成ならびに管理ができるようになりました。この機能は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。

追加された Microsoft SQL Server 2005 Express SP2 前提条件

InstallShield に Microsoft SQL Server 2005 Express Edition SP2 の InstallShield 前提条件が追加されました。この InstallShield 前提条件は、基本の MSI および InstallScript MSI プロジェクトに追加することができます。

MySQL 5.0 サポート

InstallShield の [SQL スクリプト] ビューで、製品がサポートするターゲット データベース サーバーを指定するときに選択可能な定義済みデータベース サーバーのリストに 、MySQL の 5.0.x バージョンもリストされるようになりました。これまでは、カスタム バージョン要件を作成しなくてはなりませんでした。

Microsoft Visual Studio 2008 サポート

InstallShield が Visual Studio 2008 と統合され、インストールと製品の開発を同じ Visual Studio インターフェイス内で行えるようになりました。

Visual Studio セットアップとマージ モジュール プロジェクトを InstallShield プロジェクトに変換する機能

今回より、InstallShield で、Visual Studio 2008、Visual Studio 2005、Visual Studio .NET 2003、または Visual Studio .NET セットアップ プロジェクト (.vdproj) を基本の MSI プロジェクト (.ism) に変換することができるようになりました。また、Visual Studio 2008、Visual Studio 2005、Visual Studio .NET 2003、または Visual Studio .NET マージ モジュール プロジェクト (.vdproj) を InstallShield マージ モジュール プロジェクト (.ism) に変換することもできるようになりました。

Visual Studio プロジェクトを InstallShield プロジェクトに変換すると、ダイアログ エディターを使ってダイアログのレイアウトを視覚的に変更したり、機能やコンポーネントを管理したり、InstallShield で提供されている他の機能を利用したりすることができます。

詳細については、「Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする」を参照してください。

アラビア語 (サウジアラビア) とヘブライ語サポート、およびダイアログ エディターにおける右から左方向に書く言語サポート

InstallShield には、右から左に記述する言語であるアラビア語 (サウジアラビア) とヘブライ語のサポートが新しく含まれています。デフォルトのエンド ユーザー ダイアログ文字列のすべてが、これらの言語で利用できます。

これらの言語は右から左方向に読まれるため、InstallShield にはアラビア語とヘブライ語ダイアログのミラー サポートが含まれます。これによって、アラブ語とヘブライ語のダイアログには、右から左方向へのレイアウトが使用されます。たとえば、英語やその他の左から右に読まれる言語のダイアログで右側にあるボタンは、右から左に読まれる言語のダイアログでは左側に移動されます。また InstallShield は、ビルトイン ダイアログ テーマで表示されるダイアログ イメージのミラー画像バージョンを使用します。

右から左方向へのレイアウトと反転イメージは、InstallShield の [ダイアログ] ビュー内の [ダイアログ エディター] ペイン、および実行時に使用されます。

アラビア語とヘブライ語のサポートは、InstallShield Premier Edition で利用できます。Premier Edition には、35 の言語サポートが含まれています。

基本の MSI とマージ モジュール プロジェクト タイプに、この機能のサポートが含まれています。

詳細については、「右から左に記述される言語のダイアログ サポート」を参照してください。

InstallScript ファイル (.rul) の文字列エントリの検証

InstallScript ファイル (.rul) を含むプロジェクトをビルドするとき、InstallScript コードに @ 演算子を使用する文字列エントリへの参照が 1 つ以上含まれている場合、InstallShield はビルド時に文字列エントリを検証します。

プロジェクトに含まれる InstallScript ファイルの文字列 ID がプロジェクトの文字列エントリの 1 つに定義されていない場合、InstallShield はビルド警告 -7174 を表示します。

これは、InstallScript カスタム アクションを含む基本の MSI プロジェクト、InstallScript、InstallScript MSI、InstallScript オブジェクト、および InstallScript カスタム アクションを含む マージ モジュール プロジェクト タイプに適用します。

詳細は、次を参照してください:

- ビルド警告 -7174 の説明
- 文字列定数演算子 (@)

新しい FlexNet Connect 11 再配布可能ファイル

InstallShield はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで FlexNet Connect 11 をサポートします。InstallShield の [アップデート通知] ビューを使って、2 つの FlexNet Connect 11 マージ モジュールうち、いずれかが含まれています (Common Software Manager が含まれているマージ モジュールと、含まれていないマージ モジュール)。

強化機能

InstallShield には、以下のような強化機能が搭載されています。

ベスト プラクティス ダイナミック ファイル リンク

プロジェクトにダイナミック ファイル リンクを追加、または変更するとき、InstallShield がコンポーネントを作成する方法について、新しいベスト プラクティスを使用するか、これまでと同様にディレクトリごとに 1 つのコンポーネント作成するのかを指定できるようになりました。

コンポーネント作成のベスト プラクティスに従うと、ダイナミック リンクを持つフォルダーにある各ポータブル実行可能ファイル (PE) ファイルにコンポーネントが別々に作成されます。ダイナミック リンクがある PE ファイルの 1 つを更新するパッチをあとで作成する場合、「ディレクトリごとに 1 つのコンポーネント」方式の代わりに、この方式を利用したほうが、パッチを簡単に作成できます。

以前、ダイナミック ファイル リンクをプロジェクトに追加すると、ビルド時に、ダイナミック リンクを持つすべてのファイルについてコンポーネントが 1 つ自動的に作成されていました。ただし、ダイナミック ファイル リンクに PE ファイルが含まれているとき、コンポーネントの作成時に Windows Installer ベスト プラクティスが実行されませんでした。

デフォルトで、InstallShield は .exe、.dll、.ocx、.vxd、.chm、.hlp、.tlb、および .ax を PE ファイルとして認識します。[オプション] ダイアログ ボックスにある新しい [ファイルの拡張子] タブで、この一覧を変更することができます。

オートメーション インターフェイスに、新しいベスト プラクティス メソッドのサポートが追加されました。ISWiDynamicFileLinking オブジェクトに、ダイナミック ファイル リンクにベスト プラクティス方式、または以前提供されていた「ディレクトリごとに 1 つのコンポーネント」方式を使用するかを指定できる新しい CreateBestPracticeComponents プロパティが追加されました。AddDynamicFileLinking メソッドを使用して新しいダイナミック ファイル リンクを作成すると、ベスト プラクティス メソッドがデフォルトで使用されます。

ベスト プラクティス ダイナミック ファイル リンクは、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトに適用します。

詳細は、次を参照してください:

- ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する
- [ダイナミック ファイル リンクの設定] ダイアログ ボックス
- [ファイルの拡張子] タブ
- ISWiDynamicFileLinking オブジェクト
- AddDynamicFileLinking メソッド

[インストールされる前提条件] 一覧からセットアップ前提条件を隠す機能

ターゲット システムに 1 つまたは複数のセットアップ前提条件のインストールが必要な場合、実行時、メイン インストールが実行する前にセットアップ前提条件ダイアログが表示されます。セットアップ前提条件エディターの [動作] タブに追加された新しいチェック ボックスを使って、前提条件ダイアログで表示される前提条件の一覧から InstallShield 前提条件を隠すことを指定できます。前提条件を隠した場合、インストールが必要な前提条件の 1 つとしてリストされていなくても、条件がそれを必要したときに前提条件がインストールされます。

この機能が利用可能になる以前に作成された新しい前提条件や既存の前提条件は、デフォルトで表示されます。この動作は、[動作] タブに追加された新しいチェック ボックスを選択して変更できます。

詳細については、「ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する」を参照してください。

実行時に InstallShield 前提条件のインストール進行状況を表示できる機能

IInstallShield 前提条件エディターの [動作] タブに追加された新しいチェック ボックスを使って、実行時に前提条件のインストールで Windows Installer からのインストール進行状況メッセージを表示するかどうかを指定できます。これによって、進行状況バーに進行中の .msi インストールのステータスが反映されます。この機能は、前提条件が .msi ファイルを起動する場合のみ使用が可能で、Setup.exe ファイルを起動する場合は使用できません。詳細については、「実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する」を参照してください。

この機能が利用可能になる以前に作成された新しい前提条件や、既存の前提条件では、デフォルトで進行状況は表示されません。この動作は、[動作] タブに追加された新しいチェック ボックスを選択して変更できます。

進行状況の表示を指定した場合、利用可能なコマンドライン パラメーターのうち一部のみがサポートされている点に、ご注意ください。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

InstallShield 前提条件に、コマンドライン ステートメントで Windows Installer プロパティを使用できる機能

基本の MSI インストールおよび InstallScript MSI インストールのセットアップ前提条件コマンドラインで、ProductLanguage、ISPREREQDIR、SETUPEXEDIR、および SETUPEXENAME プロパティを置換、および渡すことができます。これらのプロパティを使って起動するインストールを識別 (**"[SETUPEXEDIR]¥[SETUPEXENAME]"**)、前提条件に含まれるファイルへの完全パスを識別 (**"[ISPREREQDIR]myconfig.ini"**)、または InstallShield セットアップランチャーなどの複数言語前提条件に選択された言語を渡す (**/l[ProductLanguage]**) ことができます。新しい機能前提条件は、前述のプロパティを含む任意の Windows Installer プロパティのコマンドライン サポートを含みます。

InstallShield 前提条件エディターの [実行するアプリケーション] タブでコマンドラインを指定できます。

詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

InstallShield 前提条件の再起動動作に追加された新しいオプション

InstallShield 前提条件エディターの [動作] タブでは、ターゲット マシンの再起動が必要な可能性がある場合、InstallShield 前提条件のインストールの処理方法を指定できます。**[前提条件が再起動を必要としているように見える場合]** リストで、動作を指定できます。このリストには **[記録する。マシンが再起動された場合、再開を失敗して、インストールの後に再起動する]** と呼ばれる新しいオプションが追加されました。実行時に再起動が必要であるように見える場合で、メインのインストールが終了するまで (または次の前提条件が再起動をトリガーするまで) 延期したいときは、この新しいオプションを選択します。

詳細については、「[動作] タブ」を参照してください。

最高 4 GB までの単一ファイル、圧縮 InstallScript インストールをサポート

単一ファイルの圧縮 InstallScript インストールのサイズは、最高で 4 GB まで可能です。以前、エンド ユーザーが大きな単一ファイルの圧縮 InstallScript インストールを実行した場合、インストールはセットアップの初期化中にクラッシュしました。

Windows は 4 GB を超える実行可能ファイルをロードしないため、Setup.exe ファイルは 4 GB 以下でなくてはなりません。

仮想ディレクトリを含まない IIS Web サイトをインストールできる機能と、Web サイトをコンポーネントと関連付けられる機能

InstallShield は、今回より、仮想ディレクトリを一切含まない IIS Web サイトのインストールをサポートできるようになりました。また、[IIS の構成] ビューで Web サイトを選択すると表示される [全般] タブには "コンポーネント" 設定が追加されました。この設定を使って選択された Web サイトとコンポーネントを関連付けることができます。これらの強化によって、Web サイトに関連付けられた仮想ディレクトリまたはコンポーネントがインストールされると、ターゲット マシン上にその Web サイトが作成されます。

Web サイトがコンポーネントと関連付けられている場合、Web サイトの **[アンインストール時に Web サイトを削除する]** チェック ボックスは、そのコンポーネントの "パーマネント" 設定 (基本の MSI プロジェクトまたは InstallScript MSI プロジェクトの場合) または "アンインストール" 設定 (InstallScript プロジェクトの場合) に対応します。つまり、Web サイトの **[アンインストール時に Web サイトを削除する]** チェック ボックスを選択またはクリアすると、自動的にコンポーネントの "パーマネント" 設定または "アンインストール" 設定が適切に更新されます。

以前、InstallShield では Web サイトとコンポーネントを関連付けるためのサポートが提供されていませんでした。そのため、インストールに含まれる Web サイトが仮想ディレクトリを持たない場合、実行時 Web サイトが作成されませんでした。

InstallShield 2009 の [IIS の構成] ビューを使って新しい Web サイトを追加すると、InstallShield は自動的にその Web サイトをコンポーネントに関連付けます。IIS Web サイトを含む InstallShield 2008 以前のプロジェクトをアップグレードした場合、その Web サイトは自動的にコンポーネントに関連付けられません。

基本の MSI、InstallScript、および InstallScript MSI プロジェクト タイプで、これらの IIS 強化内容をサポートします。

InstallScript プロジェクトで Web サイトがコンポーネントに関連付けられている場合、その Web サイトに追加される仮想ディレクトリも同じコンポーネントに関連付けなくてはなりません。したがって、1 つまたは複数の仮想ディレクトリを含む Web サイトのコンポーネントを変更しようとすると、InstallShield はメッセージ ボックスを表示して、Web サイトの仮想ディレクトリのすべてに対して、同じコンポーネントの変更が行われることを通知します。このメッセージ ボックスは、Web サイトに含まれる任意の仮想ディレクトリのコンポーネントを変更しようとした場合にも表示されます。どちらの場合も、メッセージ ボックスはコンポーネントの変更を続行するか、キャンセルするか選択肢を提供します。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、Web サイトのすべての仮想ディレクトリと同様に、Web サイトを同じコンポーネントに保持することが必須ではありませんが、推奨されます。

詳細は、次を参照してください:

- Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加
- IIS サポートの機能とコンポーネントの関連付け
- Web サイト、アプリケーション、および仮想ディレクトリのアンインストール

QuickPatch パッケージの簡素化

今回より、一般的に以前の InstallShield でビルドされたパッケージに比べて新しいサブ機能とビルトイン InstallShield カスタム アクションの数が少ない、簡素化された QuickPatch パッケージのビルドが可能となりました。QuickPatch プロジェクトの [詳細] タブに追加された "QuickPatch の簡素化" 設定で、この新しいタイプの QuickPatch パッケージを作成するかどうかを指定できます。

詳細については、次を参照してください。

- QuickPatch パッケージを簡素化するかどうかを指定する
- [詳細] タブ

コマンドラインと MSBuild からパッチをビルドできる機能

–patch_config コマンドライン パラメーターが、IsCmdBld.exe を使用したコマンドライン ビルド (Standalone Build コマンドライン ビルドを含む) 用に追加されました。このパラメーターを使用して、コマンドラインからパッチをビルドすることができます。

.ini ファイルを使ってコマンドラインにパラメーターを渡す場合、.ini ファイルの [Project] セクションに追加された PatchConfigName パラメーターを使って、ビルドするパッチ構成を指示することができます。

また、MSBuild の InstallShield タスクに、MSBuild を使ってビルドするパッチ構成を指定することができる PatchConfiguration パラメーターが追加されました。

詳しくは、次を参照してください:

- ISCmdBld.exe
- .ini ファイルでコマンドライン ビルド パラメーターを渡す
- Microsoft ビルド エンジン (MSBuild)

パッチと QuickPatch パッケージをパスワードで保護する機能

今回より、パッチと QuickPatch パッケージをパスワードで保護するためのパスワード設定が追加されました。これらの設定は基本の MSI プロジェクトと InstallScript MSI プロジェクトでは [パッチのデザイン] ビューの [詳細] タブに、また QuickPatch プロジェクトでは [詳細] タブにあります。

パッチまたは QuickPatch パッケージをパスワードで保護すると、すべてのエンド ユーザーはパッケージをインストールする時に、アップデートを起動するためのパスワード (大文字と小文字を区別する) を入力しなくてはなりません。

詳しくは、次を参照してください:

- パッチ パッケージをパスワードで保護する
- QuickPatch パッケージをパスワードで保護する
- [詳細] タブ ([パッチのデザイン] ビューのパッチ構成用)
- [詳細] タブ (QuickPatch プロジェクト)

QuickPatch プロジェクトのパッチとアップグレードの検証サポート

QuickPatch プロジェクトをビルドしたとき、InstallShield はパッチとアップグレードの検証を実行するようになりました。この検証は、QuickPatch パッケージを使って製品のアップグレードを試みるときに発生する可能性のある、一般的な問題を識別するのに役立ちます。

これまでパッチとアップグレードの検証は、基本の MSI プロジェクトと InstallScript MSI プロジェクトで [パッチのデザイン] ビューで作成されたパッチでしか利用できませんでした。

詳細については、「アップグレード、パッチ、および QuickPatch パッケージを検証する」を参照してください。

ビルド時の検証に複数 .cub ファイルを選択できる機能

InstallShield の [オプション] ダイアログ ボックスにある [検証] タブでは、InstallShield 内でビルドされたインストール パッケージまたはマージ モジュールを検証する際に使用される .cub ファイルを複数指定できるようになりました。

また、ISCmdBld.exe を使ったコマンドライン ビルド (Standalone Build コマンドライン ビルドを含む) で -m コマンドライン パラメーターを使って、複数の .cub ファイルを渡すこともできるようになりました。.iniファイルを使ってコマンドラインにパラメーターを渡す場合、今回より .ini ファイルの CubFile パラメーターに 複数の .cub ファイルを指定することができるようになりました。

MSBuild 用の InstallShield タスクの RunMsiValidator パラメーターが強化され、複数の .cub ファイルを使用できるようになりました。

詳細については、次を参照してください。

- Windows ロゴ プログラムの要件
- インストール パッケージまたはマージ モジュールを検証する
- ビルド時に検証を行うかどうかを指定する
- インストール パッケージまたはマージ モジュールを検証する
- [検証] タブ ([オプション] ダイアログ ボックス)
- [ビルド] メニュー
- ISCmdBld.exe
- .ini ファイルでコマンドライン ビルド パラメーターを渡す
- Microsoft ビルド エンジン (MSBuild)

InstallShield ベストプラクティス スイートの新しい検証ツール

ISBP20 は InstallShield ベスト プラクティス スイートで利用可能な新しい検証ツールです。ISBP20 は、Registry テーブルに含まれるレジストリ エントリが、ルート レベルのレジストリ キーや、その削除によってターゲット マシン上でのトラブルの原因となるようなキーの削除が行われないことを検証します。

InstallShield ベスト プラクティス スイートは、InstallShield Premier Edition の基本の MSI、InstallScript MSI、および MSI データベース プロジェクトで利用できます。

/v コマンドライン パラメーターを複数回使用して、Setup.exe から .msi ファイルに複数のパラメーターを渡すことができる機能

Setup.exe から Msiexec.exe へ複数の引数を渡す場合、コマンドラインで /v オプションを各引数ごとに 1 回ずつ、複数回使用することができます。以前、/v オプションの使用は 1 回に限られていたため、すべてのパラメーターはこのインスタンスを通して渡されていました。

基本の MSI と InstallScript MSI プロジェクト タイプに、この強化サポートが含まれています。

詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

改善された Standalone Build と InstallShield の間の互換性

今回より、InstallShield Premier Edition で提供されている Standalone Build は、InstallShield がプログラム ファイルに使用するディレクトリ構造と同じディレクトリ構造を使用します。これにより、InstallShield があるマシンから再配布可能ファイルや他のファイルを Standalone Build があるマシンにコピーするとき、同じ相対パスを使用できるようになりました。以前、異なるディレクトリ構造が別々に使用されていました。

また、InstallShield のコマンドライン ビルドで使われる ISCmdBld.exe も Standalone Build と共にインストールされています。以前、Standalone Build は別のファイル (IsSaBld.exe) をコマンドライン ビルドに使用していました。ISCmdBld.exe は今回より、以前 IsSaBld.exe のみがサポートしていたパラメーターをサポートします。

- –o <マージ モジュール検索パス>–プロジェクトで参照されるマージ モジュール (.msm) ファイルを含むフォルダーをコンマで区切って指定します(複数指定可)。
- –t <Microsoft .NET Framework のパス>– Microsoft .NET Framework へのパスを指定します。パスは、ビルド マシンにインストールされている .NET Framework の場所です。
- –h – ビルドの終わりでアップグレード検証ツールをスキップすることを示します。
- –g <最小ターゲット MSI バージョン> – ターゲット マシン上でインストールに必要な Windows Installer の最小バージョンを指定します。
- –j <最小ターゲット Microsoft .NET Framework バージョン> – ターゲット マシン上でインストールに必要な .NET Framework の最小バージョンを指定します。

この強化の一部として、Standalone Build のインターフェイスで、InstallShield と同じ ISWiAutomation15.dll ファイルが使用されるようになりましたが、インストールされる場所は異なります。InstallShield オートメーション インターフェイスと共に動作する既存のオートメーション スクリプトがある場合、スタンドアロン オートメーション インターフェイスと共に使用するためにスクリプト全体でライブラリの名前を IswiAuto*N* から SAAuto*N* (N はバージョン番号です) に変更する作業は必要なくなりました。この変更により、ISWiProject オブジェクトに、以前 Standalone オートメーション インターフェイスでのみ提供されていたいくつかのプロパティのサポートが追加されました。

- DotNetFrameworkPath
- MergeModuleSearchPath
- MinimumTargetDotNetVersion
- MinimumTargetMSIVersion
- SelfRegistrationMethod
- SkipUpgradeValidators

互換性の改善により、別々の 2 セットのバイナリの代わりに 1 セットのバイナリのみが Standalone Build と InstallShield にビルドされるため、InstallShield ホットフィックスまたはサービス パックがリリースされたとき、それを Standalone Build と InstallShield に使用することができます。以前、別々のホットフィックスとサービス パックが必要でした。

詳しくは、次を参照してください:

- Standalone Build をビルド マシンにインストールする
- Standalone Build 用に再配布可能ファイルをビルド マシンへ追加する
- スタンドアロン コマンドライン ビルド
- スタンドアロン オートメーション インターフェイス

- Standalone Build と InstallShield を同一のマシン上にインストールする
- ISCmdBld.exe
- ISWiProject オブジェクト

SQLBrowse 実行時ダイアログで、ローカル、リモート、エイリアス SQL Server を参照が可能

SQLBrowse ダイアログのフィルター機能が強化されました。

基本の MSI と InstallScript MSI インストールの SQL Server の参照コンボ ボックスとリスト ボックス コントロールでリモート サーバーのみ表示したい場合、新しい Windows Installer プロパティ IS_SQLSERVER_REMOTE_ONLY を設定します。基本の MSI と InstallScript MSI インストールの SQL Server の参照コンボ ボックスとリスト ボックス コントロールでサーバー エイリアスのみ表示したい場合、新しい Windows Installer プロパティ IS_SQLSERVER_ALIAS_ONLY を設定します。以前は、ローカル サーバーのみを表示するフィルターだけしか利用できませんでした。これには、IS_SQLSERVER_LOCAL_ONLY プロパティを設定します。これらのプロパティを任意に組み合わせて、SQL Server 参照コンボ ボックスとリスト ボックス コントロールで、複数の種類のサーバーを表示することができます。

詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。

InstallScript プロジェクトに、次の 2 つの新しい InstallScript 関数が追加されました:

- SQLRTSetBrowseOption—この関数を利用して、SQL Server の参照コンボ ボックスとリスト ボックス コントロールで、ローカル サーバー、リモート サーバー、サーバーのエイリアス、またはこれらのサーバーの組み合わせを表示するかどうかを指定できます。
- SQLRTGetBrowseOption—この関数は、SQL Server の参照コンボ ボックスとリスト ボックス コントロールの参照オプションの現在の値を返します。これらでは、ローカル サーバー、リモート サーバー、サーバー エイリアス、およびこれらのサーバーの組み合わせを表示できます。

SQL データベース サーバーの最小要件が満たされない場合でもインストールを続行する機能

[SQL スクリプト] ビューで SQL 接続を選択すると表示される [要件] タブに、新しい **[最小要件が満たされていない場合でもインストールの続行を許可する]** チェック ボックスが追加されました。

このチェック ボックスを選択すると、ターゲットシステム上でデータベース サーバーの最小要件が満たされていない場合、ランタイムは SQL 接続とその SQL スクリプトのすべてをスキップしてインストールを続行します。

このチェック ボックスをクリアしていて最小要件が満たされない場合、エンド ユーザーはインストールを続行することができません。これがデフォルトの動作です。

この強化内容は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します。

詳細については、「[要件] タブ」を参照してください。

SQL Server エイリアスのサポート

SQL ランタイム サポートが強化され、インストールが SQLBrowse ダイアログで SQL Server のエイリアス名をリストできるようになりました。また SQLLogin ダイアログを使って、エンド ユーザーはエイリアス名で SQL Server に接続することが可能です。

この強化内容は、基本の MSI、InstallScript、および InstallScript MSI プロジェクトに適用します。

詳細については、「新しい SQL 接続の追加」を参照してください。

.NET Framework 用の定義済みシステム検索

InstallShield に、次の新しい定義済みシステム検索が追加されました。

- Microsoft .NET Framework 3.5
- Microsoft .NET Framework 3.0 SP1
- Microsoft .NET Framework 3.0
- Microsoft .NET Framework 2.0 SP1
- Microsoft .NET Framework 2.0
- Microsoft .NET Framework 1.1
- Microsoft .NET Framework 1.0

インストールでこれらのいずれかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

この強化は、基本の MSI および InstallScript MSI プロジェクトに適用します。

エラー カスタム アクションの拡張サポート

このカスタム アクション ウィザードに、指定されたエラー メッセージを表示し、失敗を戻して、インストールを終了するタイプ 19 カスタム アクションのサポートが追加されました。以前、この種類のカスタム アクションを作成するには、[カスタム アクションとシーケンス] ビューで [カスタム アクション] エクスプローラーを右クリックしてから、[エラー] をクリックするか、またはダイレクト エディターを利用して、手動でカスタム アクションのテーブル エントリを入力する必要がありました。

基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプでエラー カスタム アクションがサポートされるようになりました。以前、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでのみサポートされていました。

ソース ファイルにデジタル署名するかどうかを指定できる機能

[リリース] ビューの [署名] タブに、新しい **[元の場所にあるファイルに署名する]** チェック ボックスが追加されました。このチェック ボックスを利用して、InstallShield で、元のソース ファイルにも署名をするか、またはリリースに組み込まれるファイルのみ署名するかを指定できます。このチェック ボックスは、リリース ウィザードの [デジタル署名のオプション] パネルでも提供されています。このチェック ボックスは、デフォルトで、クリアになっています。

基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで、このチェック ボックスを選択すると、もともと署名されていなかったファイルを含むリリースの圧縮バージョンと非圧縮バージョンの両方を更新する単一のパッチを作成する場合に便利です。

以前、このチェック ボックスがなかったため、InstallShield で元の場所にあるファイルを署名することができませんでした。

オートメーション インターフェイスには、このあたらしいデジタル署名機能のサポートが含まれています。ISWiRelease オブジェクトに、InstallShield が元のソース ファイルに署名するか、リリースに組み込まれたファイルにのみ署名するかを指定できる **SignFilesInPlace** プロパティが追加されました。

詳しくは、次を参照してください:

- リリースの [署名] タブ
- [デジタル署名のオプション] パネル
- デジタル署名とセキュリティ
- ISWiRelease オブジェクト

改良されたスタティック COM 抽出

スタティック COM 抽出を使用する場合、Registry テーブルのプライマリ キーを生成するときに InstallShield が MD5 アルゴリズムを使用するようになりました。したがって、COM データが変更されない場合、パッケージの異なるバージョン間や、抽出された COM データが更新されるときにプライマリ キーが変更されることはありません。

以前、InstallShield はスタティック COM 抽出中に作成されたプライマリ キーにはランダム値を使用しました。その結果、COM データが更新されたとき、またはパッチがビルドされたとき、COM データが変更されていないにも関わらず新しいプライマリ キーが作成される可能性がありました。パッチに関しては、プライマリ キーが変更された場合に COM データがパッチに含まれます。パッチが変更されていない COM データを持つコンポーネントを更新した場合、パッチのアンインストール中に COM データが削除される可能性があり、これは製品の以前のバージョンに問題をもたらす原因となりかねません。

この強化は、基本の MSI および InstallScript MSI プロジェクトに適用します。

InstallScript 言語の強化および新機能

InstallScript 言語が一部強化されました。

.NET Framework 3.5 および .NET Framework 3.0 SP1 の強化

新しい FOLDER_DOTNET_35 InstallScript 変数が追加されました。この変数は、.NET Framework 3.0 ファイルのパスを格納します。

Is 関数と使用するための 2 つの新しい定数が追加されました:

- REGDB_KEYPATH_DOTNET_35
- REGDB_KEYPATH_DOTNET_30_SP

.NET Framework 3.0 の SP1 またはそれ以降のサービス パックがインストールされているかどうかをクエリするとき、REGDB_KEYPATH_DOTNET_30_SP 変数を使用できます。.NET Framework 3.0 の RTM バージョンがインストールされているかどうかを検出するには、REGDB_KEYPATH_DOTNET_30 を使用します。

一時ファイルを作成する Windows API GetTempFileName を呼び出す、新しい GetTempFileNameIS 関数

GetTempFileNameIS という名前の新しい InstallScript 関数が追加されました。この関数は Windows API **GetTempFileName** を呼び出して一時ファイルを作成し、その関連アクションを実行します。

詳しくは、「GetTempFileName」を参照してください。

InstallShield 2008 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

基本の MSI プロジェクト用の新しいエンド ユーザー ダイアログ テーマ

ダイアログ テーマは、エンドユーザー ダイアログに統一感のとれた個性的な印象を与えることができる、あらかじめ定義されている 1 セットのイメージです。ボタンをクリックするだけで、プロジェクトに提供されているテーマから 1 つ選んで、Setup.exe 初期化ダイアログを含む、プロジェクトで使用されているすべての内部および外部ダイアログに適用することができます。それぞれのダイアログは [ダイアログ] ビュー内から簡単にプレビューすることができ、選択したテーマがどのように見えるかを実際に確認することができます。

一部のテーマは、InstallShield の Premier Edition と Professional Edition の両方で提供されていますが、Premier Edition のみで提供されているものもあります。詳細については、「提供されているテーマと対応するダイアログのサイズ」を参照してください。

この機能の詳細は、「ダイアログのテーマ」を参照してください。

デジタル署名の強化機能

ビルド時に、製品の実行可能ファイルを含むプロジェクト内のすべてのファイルにデジタル署名が可能になりました。また、今回より、デジタル署名に personal information exchange ファイル (.pfx) が使用できるようになりました。この機能は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクト タイプでサポートされています。

[リリース] ビューで新しい [署名] タブを使って、InstallShield がファイルに署名するときに使用するデジタル署名に関する情報 (証明機関より付与されたデジタル証明書ファイルを含む) を指定します。[署名] タブでまた、デジタル署名をするインストール内のファイルを指定することもできます。リリース ウィザードを使って、すべてのデジタル署名情報を指定することもできます。

署名に .pfx ファイルを指定すると、InstallShield でファイルが署名されるとき SignTool.exe が使用されます。.spc ファイルと .pvk ファイルを指定すると、ファイルの署名に Signcode.exe が使用されます。.pfx ファイルは、より多くの異なる環境 (ロックされたビルド マシンなど) で動作するため、より頻繁に利用されています。InstallShield でデジタル署名パスワードを指定するとき、.pfx ファイルを使用している場合、パスワードのプロンプトは表示されなくなります。.spc ファイルと .pvk ファイルを使用している場合は、パスワードのプロンプトが表示されることがあります。

以前、InstallShield では、.msi ファイル、.hdr ファイル、Setup.exe ファイルのみ署名が可能でした。また、デジタル署名用に .spc ファイルと .pvk ファイルは指定できましたが、.pfx ファイルは指定できませんでした。

詳しくは、次を参照してください:

- デジタル署名とセキュリティ
- ビルド時にリリースとそのファイルにデジタル署名を行う
- リリースの [署名] タブ

新しい InstallShield ベスト プラクティス スイート (Premier Edition)

InstallShield には、InstallShield ベスト プラクティス スイートという名前の 1 セットの検証ツールが含まれています。インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの InstallShield ベスト プラクティス (ISBP) 検証ツールによって警告されます。

この機能は、基本の MSI、InstallScript MSI、MSI データベース プロジェクトで提供されています。

詳細については、次を参照してください。

- InstallShield ベスト プラクティス スイート
- 検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する

インターネット インフォメーション サービス (IIS) 7.0 と SSL のサポート

InstallShield には今回より IIS 7 のサポートも含まれています。

また、インストールに、Web サイトの SSL 証明書を含めることもできます。SSL サーバー証明書を含めることにより、ユーザーは Web サーバーの認証および Web コンテンツの有効性の確認を行うことができると共に、セキュリティで保護された接続を確立することができます。

詳細については、次を参照してください。

- [IIS 構成] ビュー
- Web サイトの SSL 証明書を指定する
- ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する
- InstallShield における IIS サポートのバージョン固有情報

新しく追加された Microsoft .NET 前提条件

InstallShield には今回、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる .NET 関連の新しいセットアップ前提条件が多数含まれています。

- .NET Framework 2.0 (x64)
- .NET Framework 2.0 (x64) 言語パック
- .NET Framework 2.0 (IA64)
- .NET Framework 2.0 (IA64) 言語パック
- .NET Framework 3.0 (x64)

詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

InstallScript プロジェクトに更新された Microsoft .NET オブジェクト

新たに更新された Microsoft .NET オブジェクトが、InstallShield で提供されています。このオブジェクトには、32 ビット、64 ビット x64、および 64 ビット Itanium バージョンを含む、バージョン 1.0 (SP3)、1.1 (SP1)、2.0、および 3.0 の .NET Framework のサポートが含まれています。オブジェクトにはまた、言語パックおよび、1.0 と 1.1 の最新サービス パックが含まれています。

このオブジェクトはまた、.NET オブジェクトを含む機能がインストールされたとき、.NET Framework のインストールを起動して、実行完了します。このオブジェクトを使用して .NET Framework を即座にインストールすることにより、その後にインストールされるファイルのインストールまたは構成で .NET Framework が必要になったときに、確実に使用できるようになります。

詳細については、「Microsoft .NET Framework オブジェクト ウィザード」を参照してください。

Visual C++ 8.0 マージ モジュールの追加

InstallShield に、Visual C++ 8.0 SP1 のマージ モジュール (バージョン 8.0.50727.762) が追加されました。

ダイアログ ボタンの UAC シールド アイコンのサポート (基本の MSI プロジェクト)

基本の MSI プロジェクトのダイアログ エディターで、すべてのボタン コントロールに新しい Show UAC Shield Icon プロパティが追加されました。このプロパティで True を選択すると、エンドユーザーが Windows Vista システムでインストールを実行しているとき、[ユーザー アカウント制御 (UAC)] シールド アイコンがボタンに表示さ

れます。Windows Vista システムで InstallShield を使用している場合、実行時と同じシールド アイコンがダイアログ エディターのボタンに表示されます。シールド アイコンによって、昇格された権限が必要かもしれないことがエンドユーザーに通知されます。

これから作成するすべての新しい基本の MSI プロジェクトでは、Show UAC Shield Icon プロパティは ReadyToInstall ダイアログの [インストール] ボタンについて True に設定されています。InstallShield 12 以前で作成された基本の MSI プロジェクトを InstallShield 2008 にアップグレードしたとき、[インストール] ボタンの "UAC シールド アイコンの表示" プロパティのデフォルト値は False になっています。この値は、このボタンについてのみではなく、他のすべてのボタンについても、必要に応じてオーバーライドすることができます。

LicenseAgreement ダイアログで、エンドユーザーが EULA の最後までスクロールすることを必須にする機能

InstallShield では、エンドユーザーがマウスまたはキーボードを使ってスクロール可能な EULA コントロール内にあるエンドユーザー使用許諾契約 (EULA) テキストの終わりに到達するまで、LicenseAgreement ダイアログ上の [次へ] ボタンを無効にすることができます。

エンドユーザーはまた、[次へ] ボタンが有効にされる前に **[ソフトウェア使用許諾契約に同意します]** オプションを選択する必要があります。この動作は、InstallShield の以前のリリースと同じです。

デフォルトで、スクロール要件は LicenseAgreement ダイアログで提供されていません。この機能を利用するには、プロジェクトに WatchScroll という名前の Windows Installer DLL カスタム アクションを追加する必要があります。このカスタム アクションは EulaScrollWatcher.dll ファイルを呼び出します。また、[次へ] ボタンのコントロール条件を変更し、イベントを Memo コントロールに追加する必要があります。

これは基本の MSI プロジェクトで提供されています。

詳しい手順については、「LicenseAgreement ダイアログでエンドユーザーが EULA を最初から最後までスクロールするのを必須にする」を参照してください。

追加された SQL Server 2005 Express Edition SP1 セットアップ前提条件

InstallShield に Microsoft SQL Server 2005 Express Edition SP1 のセットアップ前提条件が追加されました。このセットアップ前提条件は、基本の MSI および InstallScript MSI プロジェクトに追加することができます。

更新された DirectX 9.0c オブジェクト

InstallShield では 2 つの DirectX 9.0 オブジェクト (基本の MSI プロジェクトと InstallScript MSI プロジェクトで提供されている Windows Installer ベースのオブジェクトと、InstallScript プロジェクトで提供されている InstallScript ベースのオブジェクト) が利用できます。これらの両方のオブジェクトは、32 ビット固有および 64 ビット固有のコンポーネントを含む、最新の DirectX 9.0c コアおよびオプションのコンポーネントをすべてインストールします。

また、基本の MSIプロジェクトおよび InstallScript MSI の DirectX 9 オブジェクト ウィザードにも一部変更が加えられました。このウィザードでは今回より、再配布可能ファイルを Disk1 フォルダーに含めるか、または .msi ファイルにストリームするかを指定することができます。この変更により、圧縮インストールで DirectX 9 オブジェクトが使用できるようになりました。また、サイレント インストールでも今回より DirectX 9 オブジェクトが使用できるようになりました。

基本の MSI プロジェクトと InstallScript MSI プロジェクトで、DirectX インストールを起動するカスタム アクションは Windows Vista システムで昇格された権限を使って実行できるように、[実行] シーケンスにスケジュールされ、遅延システム コンテキストで実行されます。

詳細は、次を参照してください:

- DirectX 9.0 オブジェクトを含める

- DirectX オブジェクト ウィザード

DIFx 2.1 のサポート

InstallShield では、最新版の Driver Install Frameworks for Applications (DIFx) がサポートされています。この新しいバージョンには、マイクロソフトからの最新バイナリ ファイルが含まれているため、InstallShield で作成する基本の MSI プロジェクト、InstallScript プロジェクトまたは InstallScript MSI プロジェクトで使用することができます。この新しいバージョンは Windows Vista システムにインストールすることができます。InstallShield の以前のバージョンには、場合によって Windows Vista にインストールすることができなかった DIFx の以前のバージョンが含まれていました。

COM サーバーの 64 ビット自己登録のサポート (基本の MSI プロジェクト)

InstallShield では今回より、基本の MSI プロジェクトで、COM サーバーの 64 ビット自己登録がサポートされています。コンポーネントを 64 ビットとしてマークしてから、そのコンポーネントにファイルを追加したとき、そのファイルの [自己登録] チェック ボックスを選択して、インストール時の 64 ビット自己登録を有効にすることができます。InstallShield ではまた、動的にリンクされている COM サーバーの 64 ビット自己登録もサポートされています。詳細については、次を参照してください。

- 64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする
- [ファイルのプロパティ] ダイアログ ボックス
- ダイナミック ファイル リンクをコンポーネントに追加する

セットアップ前提条件エディターの拡張されたオペレーティング システム条件の設定

セットアップ前提条件エディターで、前提条件に条件を作成するとき、より詳しいオペレーティング システム要件に関する情報を指定できるようになりました。セットアップ前提条件エディターで前提条件の条件を追加または変更するときに表示される [前提条件設定] ダイアログ ボックスでは、定義済みのオペレーティング システムを選択することもできますし、カスタム オプションを選択することもできます。新しいカスタム オプションを使用して、プラットフォーム、メジャーとマイナー バージョン、プロセッサ アーキテクチャ (64 ビットまたは 32 ビット) およびサービス パックなどのオペレーティング システム要件の設定を構成することができます。ターゲット システムがオペレーティング システムの要件を満たさない場合、前提条件はインストールされません。

詳細は、次を参照してください:

- InstallShield 前提条件のオペレーティング システム条件を追加する
- [前提条件設定] ダイアログ ボックス
- [条件] タブ

Windows Server 2008 システムをターゲットする機能

InstallShield では、Windows Server 2008 がインストールに必須であると指定することができます。また、機能およびコンポーネントに Windows Server 2008 に関連する条件をビルドすることもできます。

Windows Vista と Windows Server 2008 は、同じメジャー バージョン番号とマイナー バージョン番号を持つ点に注意してください。したがって、InstallScript を使って Windows Server 2008 と Windows Vista を区別するには、SYSINFO.nOSProductType = VER_NT_WORKSTATION が、Windows Vista の場合は TRUE、Windows Server 2008 の場合は FALSE であるかどうかを確認してください。詳細については、「SYSINFO」を参照してください。

新しい MSXML 6 SP1 セットアップ前提条件

InstallShield には今回より、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができる新しい MSXML セットアップ前提条件がいくつか含まれています。

- MSXML 6.0 SP1
- MSXML 6.0 SP1 (IA64)
- MSXML 6.0 SP1 (x64)

MSXML の詳しい情報については、「XML ファイルの変更の実行時の要件」を参照してください。

FlexNet Connect サポート

基本の MSI プロジェクト、および InstallScript MSI プロジェクトに FlexNet Connect 6.1 の再配布可能ファイルを追加することができます。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの [アップデート通知] ビューで、プロジェクトに含める FlexNet Connect のバージョンを選択することができます。バージョン 6.1、または [オプション] ダイアログ ボックスの [マージ モジュール] タブにある [マージ モジュールの場所] 領域で指定されている場所にインストールされている任意のバージョンを含めることができます。

[アップデート通知] ビューに、FlexNet Connect 6.1 がサポートする新しい "ベンダー データベース" 設定が追加されました。

強化機能

リリースにおけるユーザビリティの強化点

[リリース] ビューで、リリースの設定がカテゴリ別に複数のタブで再構成されました。

また、[リリース] ビュー内から "圧縮" 設定に、[圧縮] または [非圧縮] を選択できるようになりました。これまで、この設定の変更を行うには、リリース ウィザードを起動しなければなりませんでした。カスタム圧縮の設定を指定する場合 (機能ごとに 1 つの .cab ファイル、またはコンポーネントごとに 1 つの .cab ファイル)、リリース ウィザードをこれまでどおり利用する必要があります。"圧縮" 設定は、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクト使用できます。

基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトでは、[配布] ビューにあった設定は、[リリース] ビューの新しい [ポストビルド] タブに移されました。[ポストビルド] タブには、ビルド時にリリースをフォルダーまたは FTP サイトに自動的に配布できるように構成できる設定があります。

基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで、[リリース] ビューでリリースを右クリックしたときに新しい [配布] コマンドが表示されます。このコマンドを選択すると、リリースに関連するすべてのファイルが [ポストビルド] タブで指定された場所にコピーされます。InstallScript および InstallScript Object プロジェクトの場合、リリースは、ビルドされるたびに、自動的に [ポストビルド] タブで指定した場所 (複数可) にコピーされます。

詳しくは、次を参照してください:

- リリースの [ビルド] タブ
- リリースの [Setup.exe] タブ
- リリースの [署名] タブ
- リリースの [インターネット] タブ

- リリースの [イベント] タブ
- フォルダーまたは FTP サイトにリリースを自動的に配布する

カスタム アクションとシーケンスにおけるユーザビリティの強化

[カスタム アクション] ビューと [シーケンス] ビューが [カスタム アクションとシーケンス] ビューという名前のビューに統合され、より強力な機能を提供できるようになりました。統合されたビューでは、ドラッグアンドドロップ編集およびコピー機能がサポートされています。

- 新しいカスタム アクションをシーケンスするには、[カスタム アクション] エクスプローラーからそれを、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。
- ダイアログ、標準アクション、またはカスタム アクションを、シーケンス内の異なる位置へ(または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置から新しい場所へドラッグします。
- カスタム アクションを別のシーケンスへコピーするには、CTRL を押しながら、カスタム アクションを別のシーケンスへドラッグします。

[カスタム アクションとシーケンス] ビューは、基本の MSI、InstallScript、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトで使用できます。

詳細については、次を参照してください。

- [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー)
- [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する
- アクションをシーケンスに挿入する
- カスタム アクションを別のシーケンスへコピーする
- シーケンスの順序を変更する

[ファイルとフォルダー] ビュー、[レジストリ] ビュー、および [再配布可能ファイル] ビューにおけるユーザビリティの強化

[ファイルとフォルダー] ビューにおける強化内容は次のとおりです。

- **[インストール先コンピューターのファイル]** ペイン内でファイルを右クリックしてから、新しい [1 つ上のフォルダーを開く] コマンドをクリックできます。Windows エクスプローラー ウィンドウが開き、右クリックして選択したファイルを含むフォルダーが表示されます。
- **[インストール先のコンピューターのファイル]** ペインを右クリックすると、新しい [追加] ボタンを使用できます。このコマンドを使うと [開く] ダイアログ ボックスが表示され、プロジェクトに追加するファイルを参照することができます。
- このビューの右上に、新しいリンク ([ソース ペインの表示] または [ソース ペインの非表示]) が追加されました。この新しいリンクを使うと、このビューの上部に表示される **[ソース コンピューターのフォルダー]** ペインおよび **[ソース コンピューターのファイル]** ペインを表示または非表示に切り替えることができます。この 2 つのペインを非表示にして Windows エクスプローラー ウィンドウを開き、InstallShield 内に表示されている残りの 2 つのペインに Windows エクスプローラー ウィンドウから直接ファイルをドラッグ アンド ドロップすることができます。

[レジストリ] ビューの右上にも新しいリンク ([ソース ペインの表示] または [ソース ペインの非表示]) が追加されました。この新しいリンクを使うと、このビューの上部に表示される **[ソース コンピューターのフォルダー]** ペインおよび **[ソース コンピューターのファイル]** ペインを表示または非表示に切り替えることができます。

以下はまた、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの [再配布可能ファイル] ビューにおける 2 の強化点です:

- このビューの右側のペインに、左上のペインで選択されたマージ モジュール、オブジェクト、またはセットアップ前提条件に関する詳細が表示されます。このビューの右上にある [詳細の表示] または [詳細の非表示] リンクをクリックすると、この詳細ペインを表示または非表示に切り替えることができます。
- セットアップ前提条件の [詳細] ペインに、選択されたセットアップ前提条件に関する完全な情報が表示されます。この情報には、前提条件に構成されている条件、コマンドライン パラメーター、およびその他の情報が含まれます。

オートメーション インターフェイスの強化

オートメーション インターフェイスが、大幅に強化されました。

可能になった CreateProject メソッドによるマージ モジュール プロジェクトの作成

ISWiProject オブジェクトの CreateProject メソッドを使用して、マージ モジュール プロジェクトを作成することができるようになりました。以前、このメソッドは、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト プロジェクトでのみサポートされていました。

詳細については、「CreateProject メソッド」を参照してください。

ダイナミック ファイル リンクの構成をサポート

オートメーション インターフェイスに、ダイナミック ファイル リンクのための新しいオブジェクトと新しいコレクションが追加されました。また、ISWiComponent オブジェクトに、 2 つの新しいメソッドと 1 つのプロパティ (コンポーネントのダイナミック ファイル リンクを追加する AddDynamicFileLinking と、削除する RemoveDynamicFileLinking、およびダイナミック ファイル リンクのコレクションを取得する ISWiDynamicFileLinkings) が追加されました。

詳細については、次を参照してください。

- ISWiDynamicFileLinking オブジェクト
- ISWiDynamicFileLinkings コレクション
- ISWiComponent オブジェクト
- AddDynamicFileLinking メソッド
- RemoveDynamicFileLinking メソッド

パス変数の構成をサポート

オートメーション インターフェイスに、プロジェクトでパス変数を構成するための新しいオブジェクトとコレクションが追加されました。また、ISWiProject オブジェクトに、パス変数のコレクションを取得 (ISWiPathVariables) でき、かつパス変数をプロジェクトに追加 (AddPathVariable) し、削除 (DeletePathVariable) することができる 2 つの新しいメソッドとプロパティが追加されました。

詳しくは、次を参照してください:

- ISWiPathVariable オブジェクト
- ISWiPathVariables コレクション
- ISWiProject オブジェクト
- AddPathVariable メソッド

- DeletePathVariable メソッド

文字列テーブル エントリの変更をサポート

オートメーション インターフェイスに、プロジェクトで言語と文字列エントリを構成するための新しいオブジェクトとコレクションが追加されました。また、ISWiLanguage オブジェクトに、文字列エントリのコレクションを取得 (ISWiStringEntries) でき、かつ文字列エントリをプロジェクトに追加 (AddStringEntry) し、削除 (DeleteStringEntry) することができる 2 つのメソッドとプロパティが追加されました。また、ISWiProject オブジェクトに、現在のプロジェクトに含まれている言語のコレクションを取得する新しいプロパティ (ISWiLanguages) が追加されました。

詳細については、次を参照してください。

- ISWiLanguage Object
- ISWiLanguages コレクション
- ISWiStringEntry オブジェクト
- ISWiStringEntries コレクション
- AddStringEntry メソッド
- DeleteStringEntry メソッド
- ISWiProject オブジェクト

環境変数の構成をサポート

オートメーション インターフェイスに、環境変数のための新しいオブジェクトと新しいコレクションが追加されました。また、ISWiComponent オブジェクトに、2 つの新しいメソッドとプロパティ (コンポーネントの環境変数を追加する AddEnvironmentVar と 削除する RemoveEnvironmentVar、および環境変数のコレクションを取得する ISWiEnvironmentVars) が追加されました。

- ISWiEnvironmentVar オブジェクト
- ISWiEnvironmentVars コレクション
- ISWiComponent オブジェクト
- AddEnvironmentVar メソッド
- RemoveEnvironmentVar メソッド

会社名、会社 URL、および会社の電話番号を設定するための ISWiProject オブジェクトのプロパティ

ISWiProject オブジェクトに新しくプロパティが追加され、[一般情報] ビューの設定の値を指定することができます。

- CompanyName
- CompanyURL
- CompanyPhone

ファイルにデジタル署名するための ISWiRelease オブジェクトのプロパティ

ISWiRelease オブジェクトに追加された新しいプロパティを使用して、ビルド時にリリース用のファイルに行うデジタル署名用の設定を構成することができます。新しいプロパティは、以下のとおりです。

- CertificatePassword
- SignFiles
- SignFilesExclude
- SignFilesInclude
- SignSignedFiles

Directory テーブルに未使用のディレクトリを保持するかどうかを指定するための ISWiRelease オブジェクトのプロパティ

ISWiRelease オブジェクトに新しく追加された KeepUnusedDirectories プロパティを使用して、このリリースをビルドしたとき InstallShield が .msi ファイルの Directory テーブルから未使用のディレクトリを削除するかどうかを指定することができます。

.NET Framework のインストール/更新のための再起動を遅延するかどうかを構成するための ISWiRelease オブジェクト プロパティ

ISWiRelease オブジェクト に追加された新しい DotNetDelayReboot プロパティ を使用して、インストールが完了するまで、ターゲット システム上での .NET Framework のインストールまたは更新に関連付けられた再起動を遅延するかどうかを指定できます。

エンドユーザーに .NET Framework をインストールするかどうかをたずねるメッセージ ボックスを表示するかどうかを指定するための ISWiRelease オブジェクト プロパティ

ISWiRelease オブジェクトに、新しい DisplayDotNetOptionDialog プロパティが追加されました。このプロパティを使用して、エンドユーザーが .NET Framework のインストールが必要がどうかを指定できるメッセージ ボックスを実行時に表示するかどうかを指定できます。

ビルド時の配布オプションを構成するための ISWiRelease オブジェクトのプロパティ

ISWiRelease オブジェクトに追加された新しいプロパティを使用して、ビルド時にリリースをフォルダーまたは FTP サイトに自動的に配布するための設定を構成することができます。新しいプロパティは、以下のとおりです。

- DistributeLoc
- DistributeToURLLoc
- DistributeToURLUserName
- DistributeToURLPassword
- DistributeAfterBuild

ISWiSequence コレクションの新しい RemoveSequenceRecord メソッド

ISWiSequence コレクションに、シーケンスからアイテムを削除することができる RemoveSequenceRecord メソッドが新しく追加されました。詳細については、「RemoveSequenceRecord メソッド」を参照してください。

基本の MSI プロジェクトにおけるサポート ファイルの追加と削除

オートメーション インターフェイスは今回より、基本の MSI プロジェクトで ISWiSetupFile オブジェクトと ISWiAdvancedFile オブジェクトをサポートします。これらのオブジェクトには、今回より基本の MSI プロジェクトで利用できるメソッド (AddSetupFile、DeleteSetupFile、AddAdvancedFile、および DeleteAdvancedFile) が含まれています。以前、これらのオブジェクトとメソッドは、InstallScript、InstallScript MSI、InstallScript オブジェクト プロジェクトでのみ提供されていました。

詳細は、次を参照してください:

- ISWiAdvancedFile オブジェクト
- ISWiAdvancedFiles コレクション
- AddAdvancedFile メソッド
- DeleteAdvancedFile メソッド
- ISWiSetupFile オブジェクト
- ISWiSetupFiles コレクション
- AddSetupFile メソッド
- DeleteSetupFile メソッド
- ISWiProject オブジェクト

[XML ファイルの変更] ビューの強化

InstallShield で、XML ファイルの変更のサポートが拡張されました。

- 今回より、インストール全部をビルドして実行する手間をかけずに、[XML ファイルの変更] ビューでプロジェクトで構成された XML ファイルの変更のみをテストすることができます。
- [XML ファイルの変更] ビューは今回より、XML ファイルで名前空間をサポートします。
- InstallShield では、XML ファイルの XML エンコードを指定することができます。

詳しくは、次を参照してください。

- XML ファイルに加えられるインストール時の変更をテストする
- XML ファイルに加えられるアンインストール時の変更をテストする
- XML ファイルで名前空間を使用する
- XML ファイルに名前空間のマッピングを宣言する
- 名前空間プレフィックスを要素に追加する
- [XML ファイルの変更] ビュー
- XML ファイルの [名前空間] タブ

XML ファイルの変更に関する詳しい情報は、InstallShield ヘルプ ライブラリの「XML ファイルの変更」セクションを参照してください。

より速くなったダイレクト エディター、文字列テーブル エディター および [ファイル] サブビュー

InstallShield のダイレクト エディターと文字列テーブル エディターで、レコードのロード時間が短縮されました。また、プロジェクトに多数のファイルが含まれているとき、InstallShield の [コンポーネント] ビュー内にある [ファイル] サブビューで、ファイルが以前のリリースに比べより一層速く表示されるようになりました。

強化された InstallScript プロジェクトにおけるユーザー アカウント制御のサポート

InstallScript プロジェクトで、[リリース] ビューの Setup.exe タブに "必要実行レベル" 設定が追加されました。この設定を使って、インストールの Setup.exe ファイルが Windows Vista プラットフォーム上でインストール (セットアップランチャー) を実行するために必要な最低権限レベルを指定できます。InstallShield が、必要レベルを指定するマニフェストを追加します。

以前は、InstallScript プロジェクトには常に最高権限マニフェストが含まれ、"必要実行レベル" 設定は基本の MSI プロジェクトまたは InstallScript MSI プロジェクトでしか利用できませんでした。

詳細については、「Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する」を参照してください。

[ショートカット] ビューの強化

基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで、[ショートカット] ビューが一部強化されました。

- ショートカットに使用されるアイコンを変更するには、そのショートカットを右クリックして、新しい **[ショートカット アイコンの変更]** コマンドをクリックします。[アイコンの変更] ダイアログ ボックスが開き、ショートカットが実行時にターゲット システムで作成されるときに使用されるアイコン ファイルと関連付けられたアイコン インデックスを選択することができます。
- [ショートカット] エクスプローラーに一覧表示されるショートカットは、ターゲット システムで使用されるアイコン イメージと共に表示されます。以前、[ショートカット] エクスプローラーでは、アイコンがショートカットに指定されていても、すべての種類のショートカットに異なるイメージが使用されていました。

強化内容についての詳細は、次を参照してください:

- ショートカットのアイコンを指定する
- [ショートカット] ビュー

One-Click Install インストールにおける Windows Vista と Internet Explorer 7 のサポート

InstallScript プロジェクトで、One-Click Install インストールを Windows Vista システムで Internet Explorer 7 と共に使用できます。One-Click Install セットアップ プレーヤーは Setup.exe ファイルに埋め込まれた形式から、外部 .ocx ファイル形式へ変更されました。セットアップ プレーヤーは適切なコマンドラインを使って、Setup.exe ファイルのダウンロードと起動を行います。これにより、限定された権限で Windows Vista 以降を使用しているエンドユーザーもインストールを実行できるようになります。インストールのマニフェストで指定された必要実行レベルにより昇格された権限が必要な場合、適切なユーザー アカウント制御 (UAC) プロンプトが Setup.exe ファイルが起動されたときに表示されます。

この新しい動作をサポートするため、新しい "One-Click Install の生成" 設定が [リリース] ビューの [インターネット] タブとリリース ウィザードの [インターネット オプション] パネルに追加されました。この設定で [はい] を選択すると、インストールに .ocx ファイルが含められます。

One-Click Install インストールについての詳細は、「標準 Web インストールと One-Click Install の違い」を参照してください。また、次も参照してください:

- リリースの [インターネット] タブ
- [インターネット オプション] パネル

SecureCustomProperties プロパティの強化されたサポート

[実行] シーケンスで昇格された権限が必要なインストールの [ユーザー インターフェイス] シーケンスにパブリック プロパティが設定されている場合、そのプロパティの値を [実行] シーケンスに渡すためには、プロパティが SecureCustomProperties プロパティの値としてリストされているか、または制限付きパブリック プロパティである必要があります。

InstallShield は今回より、場合によって [ユーザー インターフェイス] シーケンスから [実行] シーケンスに渡す必要があるプロパティを SecureCustomProperties プロパティに自動的に追加します。詳細については、「パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する」を参照してください。

このサポートは、基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトに適用します。

基本の MSI プロジェクトと InstallScript MSI プロジェクトにおける自動ダウングレード防止エントリ

エンドユーザーが製品の現在のバージョンで同製品の将来のメジャー バージョンを上書きインストールできないようにするためには、、[アップグレード] ビューにメジャー アップグレード アイテムがあること、製品の現在のバージョンによって将来のバージョンが上書きインストールされないようにメジャー アップグレード アイテムが適切に構成されていること、および製品に適切に構成、スケジュールされたタイプ 19 のカスタム アクションが含まれている必要があります。

新しい基本の MSI または InstallScript MSI プロジェクトを作成したとき、現在のインストールが将来のメジャー バージョンを上書きするのを防ぐためのサポートが自動的に追加されます。詳しくは、次を参照してください:

- 現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ
- ISICE19

ALLUSERS と CustomerInformation ダイアログの変更

InstallShield 2008 より、新しい基本の MSI プロジェクトでは、デフォルトで ALLUSERS プロパティに 1 が設定されます。ほとんどのインストールは、マシンごとに管理者権限を使用して実行される必要があるため、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、ALLUSERS プロパティの値は自動的に変更されません。また、このプロパティが以前のプロジェクトで定義されていない場合も、自動的に追加されません。

また installShield 2008 から、デフォルトで、すべての新しい基本の MSI プロジェクトの CustomerInformation ダイアログは、エンドユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーのみにインストールするかを指定できるラジオ ボタン グループを表示しないようになっています。このダイアログについては、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、CustomerInformation ダイアログは自動的に変更されません。

詳しくは、次を参照してください:

- ユーザーごとのインストールとマシンごとのインストールの違い
- ALLUSERS

コマンドラインまたは MSBuild タスク パラメーターで製品バージョンを変更する機能

-y コマンドライン パラメーターが、ISCmdBld.exe と IsSaBld.exe を使用したコマンドライン ビルド用に追加されました。このパラメーターを使用して、コマンドラインから製品のバージョンを指定することができます。

また、MSBuild の InstallShield タスクに、MSBuild で製品バージョンを指定することができる ProductVersion パラメーターが追加されました。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、プロパティ InstallShieldProductVersion として露出されます。

–y コマンドライン パラメーターまたは InstallShield タスク ProductVersion パラメーターは、製品バージョンのビルド バージョン (3 番目のフィールド) を増加するとき、特に便利です。

詳しくは、次を参照してください:

- ISCmdBld.exe
- スタンドアロン コマンドライン ビルド
- Microsoft ビルド エンジン (MSBuild)

コマンドラインまたは MSBuild タスク パラメーターで Windows Installer のプロパティ値をオーバーライドする機能

–z コマンドライン パラメーターが、ISCmdBld.exe と IsSaBld.exe を使用したコマンドライン ビルド用に追加されました。このパラメーターを使用して、Windows Installer プロパティの値をオーバーライドしたり、プロパティが存在しないとき、それを新規作成したりできます。

また、MSBuild の InstallShield タスクに新しく追加された PropertyOverrides パラメーターを利用して、Windows Installer プロパティの値をオーバーライドしたり、または、そのプロパティが存在しないとき、新しく作成したりできます。このプロパティは、InstallShield タスクの PropertyOverrides プロパティに InstallShieldPropertyOverrides ItemGroup パススルーとして露出されます。

詳しくは、次を参照してください:

- ISCmdBld.exe
- スタンドアロン コマンドライン ビルド
- Microsoft ビルド エンジン (MSBuild)

デフォルトで、IIS データに使用されている Windows Installer プロパティがレジストリに格納されます

実行時に、IIS Web サイトをインストールし、Windows Installer プロパティを使用して動的に IIS データを設定するインストールでは、プロパティとその値が次のレジストリ キーに書き込まれます:
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥InstallShield Uninstall Information¥{ProductCode}

この変更は、アンインストールと修復の実行時に、この値を使用可能にするために行われました。したがって、Web サイトがエンドユーザーによって指定されたサイト番号にインストールされた場合、Web サイトと仮想ディレクトリはこのサイト番号から正常にアンインストールされるようになります。IIS データをレジストリに格納しないほうが良い場合、プロジェクトで IS_IIS_DO_NOT_USE_REG プロパティを設定します。

この変更は、基本の MSI および InstallScript MSI プロジェクトに適用します。

IIS Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定するための新しい設定

IIS Web サーバーを構成して、#exec ディレクティブの CMD コマンドがシェル コマンドの実行に使用されるのを防いだり、CMD コマンドがこのタイプのコマンドの実行に使用されることを許可することができます。HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters レジストリ キーの SSIEnableCmdDirective レジストリ値によって、CMD コマンドが許可されているかどうかが判別されます。

InstallShield の [IIS 構成] ビューに、新しい "SSIEnableCmdDirective レジストリ値" 設定が追加されました。この設定を使って、インストールがターゲット システム上で SSIEnableCmdDirective レジストリ値をどのように構成するのかを指定することができます。また、SSIEnableCmdDirective レジストリ値を実行時に変更しないように指定することもできます (デフォルト動作)。

詳細については、「Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する」を参照してください。

IIS Web サイト用の新しい "ホスト ヘッダー名" 設定

[IIS 構成] ビュー内の Web サイトについて、[Web サイト] タブに追加された新しい "ホスト ヘッダー名" 設定を使って、インストール中に追加された IIS Web サイトを識別するホスト ヘッダー名を指定することができます。以前は、ホスト ヘッダー名を指定するには、同じビュー内の [詳細] タブにある ServerBindings プロパティで構成しなくてはなりませんでした。

詳細については、「Web サイトの IIS ホスト ヘッダー名を指定する」を参照してください。

IIS Web サイトとその仮想ディレクトリを次に利用できる新しいサイト番号にインストールできる機能

InstallShield で、IIS Web サイトとその仮想ディレクトリを次に利用できる新しいサイト番号にインストールすることができます。実装は、作業をしているプロジェクトの種類によって若干異なります。詳細については、「TCP ポート番号とサイト番号の構成」を参照してください。

新しい SQL 接続で同じ Windows Installer プロパティを共有するかどうかを指定できる機能

[オプション] ダイアログ ボックスに新しく追加された [SQL スクリプト] タブで、新規のデータベース接続がデフォルトでどのように作成されるかを指定することができます: プロジェクトの最初の接続にデフォルトで使用された Windows Installer プロパティを使用する方法と、新しい一意の Windows Installer プロパティ セットを使用する方法があります。

これは、基本の MSI および InstallScript MSI プロジェクト タイプに適用されます。

詳細は、次を参照してください:

- 新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する
- [SQL スクリプト] タブ

SQLLogin ダイアログの強化

SQLLogin ダイアログに、エンドユーザーがターゲット データベース カタログの名前を指定できるようにする新しいコントロールが追加されました。新しいコントロールのとなりには [参照] ボタンもあり、エンドユーザーはこれをボタンを利用して、ターゲット データベース サーバーで提供されているデータベース カタログの一覧から選択することができます。

SQL サポートを含む基本の MSI プロジェクトを InstallShield 12 以前から InstallShield 2025 にアップグレードした場合、新しいバージョンの SQLLogin ダイアログを使用するには、SQLLogin.isd と SQLBrowse.isd ダイアログを手動でプロジェクトにインポートする必要があります。.isd ファイルは、*InstallShield Program Files フォルダー*¥Support にインストールされます。この更新された SQLLogin ダイアログを InstallScript プロジェクトまたは InstallScript MSI プロジェクトで使用する場合、InstallScript コード内の **SQLServerSelectLogin** 関数の呼び出しを新しい **SQLServerSelectLogin2** の呼び出しで置き換えてください。

データベース インポート ウィザードの強化

データベース インポート ウィザードに、データベース スクリプト、データベース ユーザーとデータベース ロール スクリプト、SQL Server ログイン スクリプト、オブジェクト レベル権限スクリプトなどのセキュリティ スクリプト作成オプションを指定することができる新しい [スクリプト作成詳細オプション] パネルが追加されました。

詳細については、「[スクリプト作成詳細オプション] パネル」を参照してください。

インストールまたはアンインストールしない SQL 接続を指定するための新しい Windows Installer プロパティ

InstallShield に、新しい Windows Installer プロパティ IS_SQLSERVER_CXNS_ABSENT_FROM_INSTALL のサポートが追加されました。このプロパティを使用して、インストールまたはアンインストール中にスキップする 1 つまたは複数の SQL 接続を指定することができます。1 つ以上の SQL 接続を指定するには、各接続をセミコロン (;) で区切ります。すべての SQL 接続をスキップするには、このプロパティの値を **ALL** に設定します。このプロパティは、SQL スクリプト作成エラーが原因で製品をアンインストールできないとき便利です。

SQL 関連のプロパティの詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。

.msi ファイルから参照されていないディレクトリを削除する機能

[リリース] ビューの [ビルド] タブに、新しい "未使用のディレクトリを保持する" 設定が追加されました。この設定を使って、選択されたリリースをビルドするときに、InstallShield が .msi ファイルの Directory テーブルから未使用のディレクトリを削除するかどうかを指定することができます。デフォルト値は [いいえ] です。

この設定は、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで使用できます。

詳細については、「リリースの [ビルド] タブ」を参照してください。

[コンポーネント サービス] ビュー内から COM+ コンポーネント ファイルのインストール先を指定する機能

[コンポーネント サービス] ビューの [インストール] タブに次の 2 つの新しい "インストール先" フィールド (サーバー タイプのインストール用と、プロキシ タイプのインストール用) が追加されました。これらのフィールドを使用して、選択した COM+ コンポーネント ファイルのターゲット先を指定することができます。以前、COM+ アプリケーションに関連付けられたコンポーネントのインストール先は、[コンポーネント] ビューまたは [セットアップ デザイン] ビューでしか指定できませんでした。

InstallFinalize アクションの後 COM+ アプリケーションがインストールされるように指定するための新しい Windows Installer プロパティ

InstallShield に、新しい Windows Installer プロパティ IS_COMPLUS_INSTALL_AT_FINALIZE のサポートが追加されました。このプロパティを使用して、ISComponentServiceFinalize アクションによってインストールされる COM+ アプリケーションを指定できます(複数可)。ISComponentServiceFinalize アクションは InstallFinalize アクションの後に呼び出される。1 つ以上の COM+ アプリケーションを指定するには、それぞれをセミコロン (;) で区切ります。すべての COM+ アプリケーションが ISComponentServiceFinalize アクションによってインストールされるように指定するには、このプロパティの値を **ALL** に設定します。

Windows Installer は、InstallFinalize アクションが実行されるまで、スクリプト内のセッションで加えられた変更を GAC にコミットしないため、このプロパティは、GAC にインストールされる .NET アセンブリを含む COM+ アプリケーションをインストールするときに便利です。

基本の MSI プロジェクトと InstallScript MSI プロジェクトに追加された定義済みシステム検索

InstallShield に、次の新しい定義済みシステム検索が追加されました。

- Adobe Reader 7
- Adobe Reader 6
- Internet Explorer 7.0

インストールでこれらの製品のいずれかが必要な場合、[システム検索] ビューまたはプロジェクト アシスタントの [インストール要件] ページを使って、これらのシステム検索をプロジェクトに追加することができます。エンド ユーザーがインストールを起動すると、Windows Installer はターゲット システムの要件が満たされているかどうかを確認します。要件が満たされていない場合、インストールでシステム検索用に定義されているエラー メッセージが表示されます。

新規および強化された Setup.exe コマンドライン パラメーターと Update.exe コマンドライン パラメーター (InstallScript プロジェクト)

Setup.exe と Update.exe に、次の 2 つの新しいコマンドライン パラメーターが追加されました:

- /installfromweb
- /media_path

また、16 進と 10 進の値が両方 /L パラメーターでサポートされるようになりました。

詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

ダウンローダー タイプの Web リリースで増加された .cab ファイルにサポートされている場所のオプション

ダウンローダー タイプの Web リリースで、インストール時に必要に応じてダウンロードされる外部 .cab ファイルを InstallShield で作成するかどうかを指定することができるようになりました。以前、ダウンローダー タイプの Web リリースでは、.cab ファイルは常に .msi パッケージにストリームされていました。

新しい外部 .cab ファイルのオプションがリリース ウィザードの [ダウンローダーのオプション] パネルに追加されました。このパネルは、メディアの種類が Web で、Web タイプが [ダウンローダー] の場合、リリース ウィザードで表示されます。[ダウンローダーのオプション] パネルに今回より、新しい [外部 .cab ファイルを作成する] チェック ボックスがあります。このチェック ボックスをクリアした場合の動作は以前と同じで、.CAB ファイルが .msi パッケージにストリームされます。このチェック ボックスを選択すると、.cab ファイルの作成方法 (機能ごとに 1 つの .cab ファイル、コンポーネントごとに 1 つの .cab ファイル、または指定した特定のサイズに基づいて複数の .cab ファイルを作成) を指定することができます。

すべてのトランスフォームが埋め込まれるようになったダウンローダーと Web タイプの Web リリースからのインストール

ダウンローダー タイプの Web リリースと Web タイプの Web リリースからのインストールでは今回より、すべての .mst ファイルと .ini ファイルを Setup.exe ファイルに埋め込みます。

パッチ表示情報における強化

[識別] タブ (以前は [アンインストール] タブと呼ばれていました) では、Windows Installer 3.0 以降を実行しているシステムの [プログラムの追加と削除] でパッチについて表示される情報を指定することができます。基本の MSI または InstallScript MSI プロジェクトの [パッチのデザイン] ビューおよび QuickPatch プロジェクトの [一般情報] ビューにあるこのタブには、表示名、製造元名、サポート URL などのアイテムのための設定があります。[

パッチのデザイン] ビューまたは QuickPatch プロジェクトでパッチ構成の最新セットアップを変更するたびに、その最新セットアップからの [プログラムの追加と削除] 情報が [識別] タブにある設定用の値として使用されます。必要に応じて、[識別] タブの値をオーバーライドすることもできます。

また、**[パッチのアンインストールを許可する** (Windows Installer 3.0 が必要)] チェック ボックスも今回 [共通] タブに追加されました。この設定は以前、[アンインストール] タブで提供されていました。

詳細については、次を参照してください。

- [共通] タブ ([パッチのデザイン] ビュー)
- [識別] タブ ([パッチのデザイン] ビュー)
- [共通] タブ (QuickPatch プロジェクト)
- [識別] タブ (QuickPatch プロジェクト)

最短初期化時間の指定機能

[リリース] ビューにあるリリースについての Setup.exe タブに "最短初期化時間" 設定が新しく追加されました。この設定を利用して、エンドユーザーがこのリリースを実行した時に、インストールが初期化ダイアログ (およびスプラッシュ画面) を表示する最短時間 (秒) を指定できます。

InstallScript 言語の強化および新機能

InstallScript 言語が一部強化されました。

アプリケーションを昇格された権限で起動するとき LaunchAppAndWait に優先する新しい LaunchApplication 関数と WaitForApplication 関数

LaunchApplication 関数は Windows API 関数 **CreateProcess** または Windows API 関数 **ShellExecuteEx** を使用して、指定されたアプリケーションを起動します。アプリケーションが起動されたあと、インストールで新しい **WaitForApplication** 関数を呼び出して (オプション)、アプリケーションが終了するのを待機することができます。

WaitForApplication 関数は、結果を返す前に、実行されているアプリケーション、および実行されているアプリケーションによって起動されたすべての子アプリケーション (オプション) が終了するのを待機します。

今回より、**LaunchAppAndWait** を呼び出すと、次が呼び出されますので注意してください:

LaunchApplication(szProgram, szCmdLine, "", LAAW_STARTUPINFO.wShowWindow, LAAW_PARAMETERS.nTimeOut, nOptions | LAAW_OPTION_CHANGEDIRECTORY | LAAW_OPTION_FIXUP_PROGRAM);

新しい **LaunchApplicationInit** 関数が追加されました。**LaunchApplication** 関数のネーミング規則に合致するために追加されたこの関数は、**LaunchAppAndWaitInitStartupInfo** 関数と同じ動作をします。

詳細については、次を参照してください。

- LaunchApplication
- WaitForApplication
- LaunchAppAndWait
- LaunchApplicationInit

2 つ新しい定義済み定数が追加されました:

- LAAW_OPTION_CHANGEDIRECTORY
- LAAW_OPTION_FIXUP_PROGRAM

- LAAW_OPTION_USE_SHELLEXECUTE
- LAAW_OPTION_WAIT_INCL_CHILD

また、LAAW_OPTION_NO_CHANGEDIRECTORY は現在使用されていません。このパラメーターを渡しても、何も効果はありません。

次のスクリプト変数もまた、追加になりました:

- LAAW_SHELLEXECUTEINFO
- LAAW_SHELLEXECUTEVERB

InstallScript MSI プロジェクトで再起動マネージャー インフラストラクチャで再起動回数を最小化するための新しい SdRMFilesInUse ダイアログと OnRMFilesInUse イベント ハンドラー

新しい **SdRMFilesInUse** 関数が追加されました。この関数は、開いた状態でファイルをロックしているアプリケーションの一覧を表示するリスト ボックスを含むダイアログを表示します。ダイアログにはまた、インストールが、
1) 再起動マネージャーを使用して、ファイルをロックしているアプリケーションを閉じる試みをするか、または
2) ロックされているファイルを上書きする試みをするか (結果として、インストールの完了に再起動が必要になる可能性が高くなります) エンドユーザーが指定できる 2 つのラジオ ボタンがあります。

InstallScript MSI プロジェクトで、新しい OnRMFilesInUse イベント ハンドラーで、新しい **SdRMFilesInUse** ダイアログが表示されます。このイベント ハンドラーは、再起動マネージャーが有効になっていて、Windows Installer 4.0 が INSTALLMESSAGE_RMFILESINUSE メッセージをインストールに送ったときに呼び出されます。

詳細については、次を参照してください。

- SdRMFilesInUse
- OnRMFilesInUse

プロジェクトごとに、追加のコンパイラ インクルード パスを指定できる機能

InstallShield の [ビルド] メニューで使用できる [設定] ダイアログ ボックスの [コンパイル/リンク] タブに、[インクルード パス] ボックスが追加されました。この [インクルード パス] ボックスを使用して、#include ステートメントによってメイン インストールの InstallScript コードに含められたソース ファイルを探すときに InstallShield が検索するディレクトリを指定します。このボックスは、コマンドライン コンパイラ Compile.exe の -i オプションに対応しています。

詳細については、「[コンパイル/リンク] タブ」を参照してください。

.NET Framework 3.0 サポートの強化

新しい FOLDER_DOTNET_30 InstallScript 変数が追加されました。この変数は、.NET Framework 3.0 のパスを格納します。この変数に対応するテキスト置換は、<FOLDER_DOTNET_30>です。

.NET Framework 3.0 は、レジストリがインストールされていることを示すとき、レジストリに *InstallSuccess* という値を 1 という値データと共に書き込みます。今回より、この InstallSuccess 値を確認するために、**Is** 関数を使って、DOTNETFRAMEWORKINSTALLED 定数を渡すことができます。この定数を **Is** 関数を使って渡すことにより、.NET Framework の以前のバージョンで使用されている *Install* の値も従来どおり確認することができます。詳細については、「Is」を参照してください。

Is 関数と使用するための 2 つの新しい定数が追加されました。

- REGDB_KEYPATH_DOTNET_30
- REGDB_VALUENAME_INSTALLSUCCESS

.NET Framework の最小サービス パック番号の存在を確認するための新しい Is 定数

新しい DOTNETSERVICEPACKINSTALLED 定数を **Is** 関数と使用して、.NET Framework の特定のサービス パック（または、より新しいバージョンのサービス パック）がインストールされているかどうかを判断します。詳細については、「Is」を参照してください。

SQL サポート用の新規追加および更新された関数

SQL サポート用に、いくつかの InstallScript 関数が新しく追加および更新されました。

InstallScript プロジェクトと InstallScript MSI プロジェクト用に、次の新しい InstallScript 関数が追加されました:

- SQLRTInitialize2— InstallScript プロジェクトでは SQLRT.dll ファイルをロードし、InstallScript MSI プロジェクトでは ISSQLSRV.dllファイルをロードします。また SQLRTInitialize2 は、設定ファイルを使用して .dll ファイルを初期化します。この関数は SQLRTInitialize 関数に優先します。
- SQLServerSelectLogin2—ターゲットされたエンド ユーザーが、現在の接続に使用する SQL Server および使用するログインの資格情報を指定できるログイン ダイアログを作成します。オプションで、接続情報に関連付けられた接続名をこのダイアログに表示することもできます。またオプションで、エンドユーザーは現在の接続にどのデータベース カタログを使用するのかを指定することができます。この関数は SQLServerSelectLogin 関数に優先します。
- SQLDatabaseBrowse—エンド ユーザーが、指定されたデータベース サーバー上で使用できるすべてのデータベース カタログを一覧表示することができるダイアログを作成します。
- SQLBrowse2—エンド ユーザーが、接続に指定されたデータベース テクノロジ用にネットワーク上で提供されているすべてのデータベース サーバーのリストを表示できるようにするダイアログを作成します。この関数は SQLBrowse 関数に優先します。
- SQLRTPutConnectionInfo2—接続情報（デフォルト サーバー、デフォルト データベース カタログ、デフォルト ユーザー名、デフォルト パスワード）を設定します。この関数は SQLRTPutConnectionInfo 関数よりも優先されます。

InstallScript プロジェクト用に、次の新しい InstallScript 関数が追加されました。

- SQLRTGetLastError2—SQL ランタイムで最後に発生したエラーについて詳細な情報を返し、適切な SQL エラー メッセージをロードします。この関数は SQLRTGetLastError 関数に優先します。
- SQLRTGetErrorMessage—接続を開いているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。
- SQLRTGetScriptErrorMessage—SQL スクリプトが実行しているときに、SQL ランタイムで最後に発生したエラーについての説明を返します。

InstallScript MSI プロジェクト用に、次の新しい InstallScript 関数が追加されました。

- SQLRTTestConnection2—接続を確立します。この関数は SQLRTTestConnection 関数に優先します。

次の InstallScript 関数は、以前 InstallScript プロジェクトでのみ提供されていましたが、今回より InstallScript MSI プロジェクトでも使えるようになりました:

- SQLRTGetConnections
- SQLRTGetConnectionInfo
- SQLRTPutConnectionInfo
- SQLRTGetConnectionAuthentication

- SQLRTPutConnectionAuthentication

InstallShield では、新しいバージョンに優先される関数も、従来どおりサポートされています。ただし、InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしても、既存の InstallScript コードは新しい関数を使えるように自動的に更新されません。

詳細については、次を参照してください。

- InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する
- SQL 関数

ユーザーが管理者グループに属するかどうかを確認するための新しい Is 定数

新しい USER_INADMINGROUP 定数を **Is** 関数で使用すると、ユーザーがインストールを標準アクセス トークンを使って実行しているかどうかにかかわらず、ユーザーが管理者グループかどうかを判別することができます。詳細については、「Is」を参照してください。

Is 関数を使って USER_ADMINISTRATOR 定数を渡したとき、今回より、ユーザーが管理者グループに属していても、グループの SE_GROUP_USE_FOR_DENY_ONLY セキュリティ ID (SID) 属性が設定されている場合 (つまり、ユーザーが標準アクセス トークンを使ってインストールを実行している場合)、FALSE を返します。

InstallScript を使ってロードされた .NET ライブラリをインストールが完了する前にアンロードできるようにする新しい関数

新しい 2 つの InstallScript 関数 (DotNetCoCreateObject と DotNetUnloadAppDomain) を利用できます。

- **DotNetCoCreateObject** 関数は、アセンブリが COM 相互運用性のために登録されることなく .NET アセンブリの関数を呼び出します。この関数では、.NET アセンブリがロードされ実行される .NET アプリケーションのドメインを指定することができます。

  DotNetCoCreateObject 関数は、CoCreateObjectDotNet 関数に類似しています。唯一異なる点は、**DotNetCoCreateObject** と使用すると、ロードする .NET アプリケーション ドメインを指定できるという点です。このドメインで、アセンブリが実行されます。

  CoCreateObjectDotNet の場合、.NET アセンブリは、インストールが完了したあと、デフォルトのアプリケーションにロードされます。このため、.NET アセンブリ ファイルは、インストールが完了するまでロックされます。

- **DotNetUnloadAppDomain** 関数は、指定された .NET アプリケーション ドメインをアンロードし、現在ロードされているアセンブリをすべて指定されたアプリケーション ドメインにリリースします。

新しい RegDBDeleteItem 関数

RegDBDeleteItem という名前の新しい InstallScript 関数が追加されました。**RegDBDeleteItem** 関数は、nItem の値に従ってアプリケーションごとのパスキーまたはアプリケーション アンインストール キーの下にある値を削除します。詳細については、RegDBDeleteItem を参照してください。

オブジェクトのための新しい GetStatus 関数

GetStatus という名前の新しい InstallScript 関数が、InstallScript オブジェクト プロジェクトで使用できるようになりました。**GetStatus** 関数は、オブジェクトの現在のステータス (Status.Number の現在の値) を取得します。詳細については、「GetStatus」を参照してください。

ListWriteToFileEx 関数とそれに関連する定数への変更

LWTF_OPTION_APPEND_TO_FILE 定数が、InstallScript 言語に追加されました。この定数を **ListWriteToFileEx** 関数の nOptions パラメーターとして渡すことにより、既存のファイルにリストを追加することができます。

また、2 つの既存の nOptions 定数の名前が変更されました。

- LWFT_OPTION_WRITE_AS_ANSI は LWTF_OPTION_WRITE_AS_ANSI という名前に変わりました。
- LWFT_OPTION_WRITE_AS_UNICODE は LWTF_OPTION_WRITE_AS_UNICODE という名前に変わりました。

後方互換性を維持するために、上記の 2 つの LWFT_* 定数は継続して使用することができます。これらの定数の定義は、対応する新しい LWTF_* 定数と同じです。

詳細は、次を参照してください:

- ListWriteToFileEx
- LWTF_OPTION_APPEND_TO_FILE
- LWTF_OPTION_WRITE_AS_ANSI
- LWTF_OPTION_WRITE_AS_UNICODE

InstallScript テキスト置換機能の拡張

今回より、InstallScript テキスト置換の関連付けを、別のテキスト置換の関連付けに埋め込むことができるようになりました。たとえば、"<MYTEXTSUB1>" を "My Text Sub 1 Value" に、"<MYTEXTSUB2>" を "Text Sub <MYTEXTSUB1> Embedded" にというように関連付けることができます。以前、ローカルのテキスト置換の関連付けがグローバルのテキスト置換の関連付けに埋め込まれた場合、ローカルのテキスト置換の関連付けは実行されませんでした。

InstallShield 12 SP1 の新しい機能

InstallShield 12 は、Microsoft Windows Vista のベータ版をサポートしています。InstallShield 12 SP1 には、Microsoft Windows Vista の RTM 版のサポートを提供する変更が含まれています。

新しい Microsoft .NET 3.0 前提条件条件

InstallShield には今回、基本の MSI プロジェクトと InstallScript MSI プロジェクトに追加することができる .NET Framework 3.0 前提条件が含まれています。

セットアップ前提条件に関する詳しい情報は、「InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する」をご覧ください。

アドバタイズを使用して、前提条件インストールで発生する UAC プロンプトの回数を最小限にする機能

[リリース] ビューの新しい "前提条件が昇格必要時のアドバタイズ" 設定では、Windows Vista マシンにおけるインストールでセットアップ前提条件が昇格された権限で正常にインストールされたあとに、.msi ファイルをアドバタイズして、実行するかどうか指定することができます。アドバタイズすることにより、エンドユーザーにユーザー アカウント制御 (UAC) のプロンプトを避けることを許可できる場合があります。アドバタイズを行わなかった場合、このプロンプトが表示され、昇格された権限を必要とする .msi パッケージを要求します。詳細については、「InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する」を参照してください。

"前提条件が昇格必要時のアドバタイズ" 設定は、インストールで、昇格された権限を要求するための UAC の同意プロンプトまたは資格情報プロンプトを起動するかどうかを判断するときに関わってくる InstallShield の設定の 1 つです。Windows Vista システムでエンドユーザーがインストールを実行する際の UAC の動作を適切に定義するためには、これらの異なる設定をよく理解することが必要です。また、これらの設定を使って、場合により、インストール中に表示される UAC プロンプトの数を最小限に抑えることもできます。これらの各設定についての情報は、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

Certified for Windows Vista ロゴ プログラム用に強化された検証

InstallShield に新しく追加された InstallShield 内部整合性評価プログラム (ICE) を利用して、作成したインストールが Certified for Windows Vista ロゴ プログラムで定義されているベスト プラクティスに十分に準拠していることを確認することができます。新しい評価ツール (ISICE11 ～ ISICE20) は、すべての .exe ファイルに埋め込みマニフェストがあること、保護されているレジストリ キーが変更されていないこと、.dll および .exe ファイルで古い形式の API が使われていないことなどを確認します。また、ISICE02 は、今回より、.ocx、.sys、.cpl、.drv、および .scr ファイルが署名されていることを検証します。これまで、ISICE02 は、exe ファイルの署名だけ検証していました。ISICE06 は今回より、保護された Windows ファイルと同じ名前を持つインストール内のファイルのパスを確認します。インストール内のファイルのインストール先パスが、保護された Windows ファイルの場所と一致しないとき、ISICE06 は今回より、検証エラーの代わりに検証警告を表示します。

パッケージに検証を実行すると、検証メッセージが出力ウィンドウの [タスク] タブに一覧表示されます。[タスク] タブの検証メッセージをダブルクリックすると、今回より、そのメッセージに対応するダイレクト エディターの領域にすぐ移動できます。この機能は、標準 ICE 同様、ISICE でも利用できます。以前は、標準 ICE にのみ提供されていました。

詳細については、次を参照してください。

- ISICE
- 検証結果を表示する

追加の変更

InstallShield 12 SP1 で解決されている問題の一覧は、リリース ノートをご覧ください。リリース ノートは、InstallShield の [ヘルプ] メニューからご覧になることができます。

InstallShield 12 の新しい機能

新しい機能

InstallShield には、以下のような新しい機能が搭載されています。

Windows Vista システムをターゲットする機能

InstallShield では、Windows Vista がインストールに必須であると指定することができます。また、機能およびコンポーネントに Windows Vista に関連する条件をビルドすることもできます。

Certified for Windows Vista ロゴ プログラムの検証

エンドユーザーに最高のインストール エクスペリエンスを提供するため、作成したインストールが Certified for Windows Vista ロゴ プログラムで定義されているベスト プラクティスに準拠していることが大切です。InstallShield には今回より、基本の MSI プロジェクトと InstallScript MSI プロジェクトで作成したインストールが Certified for

Windows Vista ロゴ プログラムのインストール要件を満たしているかどうかを検証する検証スイートが搭載されています。Windows Vista ロゴ を使用する場合、アプリケーションのインストールがプログラムの要件を満たしている必要があります。

また、インストール パッケージの検証およびマージ モジュールの検証に使用する内部整合性評価プログラム (ICE) の評価項目をカスタマイズすることも可能になりました。このため、ロゴ資格認定を取得する必要はないけれども、検証で作成中のインストールまたはマージ モジュールが特定のベスト プラクティスに適合していることを確認したいときに便利です。検証の設定を構成するには、[ツール] メニューで、[オプション] をクリックし、[検証] タブを選択します。

詳しくは、次を参照してください:

- ビルド時に検証を行うかどうかを指定する
- 検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する
- ISICE

ユーザー アカウント制御のサポート

InstallShield では、Microsoft が Windows Vista に追加したユーザー アカウント制御機能がサポートされています。[一般情報] ビューに新しく追加された "管理者権限" 設定を使って、インストールが管理者権限を必要とするかどうかをプロジェクト全体に対して指定することができます。また、[リリース] ビューの [必要実行レベル] 設定では、Setup.exe ファイルが Windows Vista プラットフォーム上でインストール (セットアップランチャー、すべてのセットアップ前提条件、および .msi ファイル) を実行するときに必要な最小特権レベルを指定できます。

詳細については、次を参照してください:

- 概要情報ストリーム データを入力する
- Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する

再起動マネージャー インフラストラクチャによる再起動回数の最小化をサポート

インストール終了後のシステム再起動は、エンドユーザーにとって不都合なものです。Certified for Windows Vista ロゴ プログラムの要件の 1 つに、エンドユーザーがインストール完了後自動的にアプリケーションを閉じて再起動を行うことができるオプションを含まなくてはならないという項目があります。

この品質ガイドラインをサポートするため、すべての基本の MSI プロジェクトで MsiRMFilesInUse ダイアログが提供されています。インストール中に更新が必要なファイルが他のアプリケーションによって使用中の場合、インストールでこのダイアログが表示されます。詳細については、「Windows Vista 以降のシステムの再起動を最小限にする」を参照してください。

デジタル署名の強化

インストールにデジタル署名情報を指定した場合、InstallShield は自動的に MsiDigitalCertificate および MsiPatchCertificate テーブルへ必要な情報を追加します。**MsiPatchCertificate** テーブルには、ユーザー アカウント制御 (UAC) のパッチを有効化するために必要な情報が含まれています。これによって、非管理者がインストールすることができるパッチの作成が可能になります。

また、プロジェクト アシスタントの [インストールのビルド] ページで、インストールにデジタル署名情報を指定できるようになりました。

詳細については、次を参照してください。

- 非管理者パッチのインストールを準備する

- デジタル署名とセキュリティ

カスタム アクション動作の文書化をサポート

各カスタム アクションの意図された動作は、Certified for Windows Vista ロゴ プログラムに基づいて文書化する必要があります。これは、システム管理者が製品をエンタープライズ環境に配布する場合など、カスタム アクションの動作を把握する必要がある場面において特に有益です。この要件を満たすための支援として、InstallShield の [カスタム アクション] ビューに新しい "ヘルプ ファイル パス" 設定が追加されました。この設定を使って、プロジェクトに作成したカスタム アクションの動作について説明するファイルへのパスを指定します。詳細については、次を参照してください:

- カスタム アクションの動作をドキュメント化する
- Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン

この文書は、様々な機能をサポートするために InstallShield プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションそれぞれについて説明します。「InstallShield カスタム アクション リファレンス」を参照してください。

セットアップ前提条件の強化

セットアップ前提条件エディターが大幅に強化されました。

- ユーザー インターフェイスが改善され、簡単にアクセスできるコマンドを一覧表示するメニューが追加されました。
- 新しい [動作] タブでは、エンドユーザーが前提条件のインストールをスキップできるかどうかを指定することができます。また、ターゲット マシンの再起動の要否に応じて、前提条件のインストールをどのように進めるかを指定することもできます。
- DWORD レジストリを比較するときの前提インストール条件を作成できるようになりました。64 ビット マシン用の条件の作成も可能になりました。

詳細については、次を参照してください。

- [動作] タブ
- InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする
- InstallShield 前提条件のインストールに伴う可能性のある問題の対処方法について
- 再起動を必要とする InstallShield 前提条件の動作を指定する

セットアップ前提条件の再配布は、これまで以上に柔軟になりました。インストールに含まれる個々のセットアップ前提条件について異なる提供方法を指定することができるようになりました。これによって、セットアップ前提条件ファイルを部分的に、ソース メディアに格納したり、Setup.exe に圧縮して実行時に抽出したり、ダウンロードしたりすることが可能になりました。さらに、セットアップ前提条件にリリース フラグを割り当てることができるようになりました。このため、異なるリリースのビルド時にセットアップ前提条件を含めたり除外するなど自由に組み合わせることが可能です。

詳しくは、次を参照してください:

- 特定の InstallShield 前提条件の実行時の場所を指定する
- リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する
- リリース フラグを InstallShield 前提条件に割り当てる

プロジェクト全体における Windows Installer 4.0 ログ ファイルのサポート

InstallShield では Windows Installer 4.0 を使って実行されるインストールについて、コマンドラインの使用やレジストリを使ったログ パラメーターの設定などの必要なしに、プロジェクト全体で簡単にログ記録を行うオプションが提供されています。ログ記録を有効にするには、[一般情報] ビューの新しい "MSI ログの作成" 設定で [はい] を選択します。デフォルトのログ記録パラメーターをオーバーライドするには、[プロパティ マネージャー] ビューの MsiLogging プロパティを編集します。インストールが Windows Installer 4.0 以降が搭載されているターゲット システム上で実行される場合、インストーラーはログ ファイルを作成して MsiLogFileLocation プロパティにそのパスを挿入します。さらに、SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。

詳細については、「Windows Installer インストールをログ記録するかどうかを指定する」を参照してください。

64 ビット システムにおけるレジストリ リフレクションのサポート

InstallShield では、Component テーブルにある msidbComponentAttributesDisableRegistryReflection という名前の新しい Windows Installer 4.0 属性がサポートされています。この属性を素早く構成するには、[コンポーネント] ビューで選択したコンポーネントについて "レジストリ リフレクションを無効にする" 設定を使用します。レジストリ リフレクションによって、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期が保たれます。この設定は、Windows Installer 4.0 がある 64 ビット システムでのみサポートされています。他のシステムでは、この設定は無視されます。詳細については、「レジストリ リフレクションの有効化と無効化」を参照してください。

複数言語ユーザー インターフェイス (MUI) のサポート

Windows Installer 4.0 によって実行される複数言語アプリケーション用のインストールを作成するとき、InstallShield を使って Windows 複数言語ユーザー インターフェイス (MUI) のサポートを含むショートカットを作成することができます。[ショートカット] ビューでは、選択されたショートカットに対して、次の 4 つの新しい設定が使用できるようになりました。

- 表示リソース DLL
- 表示リソース ID
- 説明リソース DLL
- 説明リソース ID

これらの新しい設定は、Windows Installer 4.0 の Shortcut テーブルにある新しい 4 つの列に対応しています。詳細については、「[ショートカット] ビュー」を参照してください。

Premier Edition 用の追加 InstallShield Collaboration ライセンス

InstallShield Premier Edition には、InstallShield Collaboration for Visual Studio のライセンスが 5 本含まれています。これらのライセンスを使って InstallShield はインストールされていないが、Visual Studio .NET 2002、Visual Studio .NET 2003 または Visual Studio 2005 がインストールされている開発システムに InstallShield Collaboration をインストールすることができます。

InstallScript アーキテクチャ再構築による InstallScript MSI プロジェクトおよび InstallScript カスタム アクションがある基本の MSI プロジェクトの機能強化

InstallScript MSI アーキテクチャには、いくつかのセキュリティ (COM/DCOM) とその他の箇所に関する問題があったため、様々な原因でインストールが失敗することがありました。これらの問題を解決して InstallScript MSI プロジェクトの安定性を高めるため、InstallShield 12 ではアーキテクチャが大幅に改良されました。この改良によって、InstallScript カスタム アクションを含む基本の MSI プロジェクトの安定性も高まりました。

詳しくは、次を参照してください:

- InstallScript カスタム アクションを使った、基本の MSI インストールのランタイム動作
- InstallScript MSI インストールのランタイム動作
- InstallShield 11.5 以前のプロジェクトをアップグレードする

DIFx 2.01 のサポート

InstallShield では、最新版の Driver Install Frameworks for Applications (DIFx) がサポートされています。この新しいバージョンには、マイクロソフトからの最新バイナリ ファイルが含まれているため、InstallShield で作成する基本の MSI プロジェクト、InstallScript プロジェクトまたは InstallScript MSI プロジェクトで使用することができます。

Setup.exe コマンドライン パラメーターと Update.exe コマンドライン パラメーターの強化

Setup.exe および Update.exe のコマンドライン パラメーターの一部は追加または変更されています。

- /Clone_wait (InstallScript のみ)
- /runfromtemp (基本の MSI 、InstallScript、および InstallScript MSI)
- /v"ISSCRIPTCMDLINE=¥"<Option1> <Option2>¥"" (基本の MSI のみ)
- /hide_progress (基本の MSI プロジェクト、InstallScript プロジェクト、および InstallScript MSI プロジェクト)
- /hide_splash (InstallScript および InstallScript MSI プロジェクト)

詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

COM 抽出と依存関係スキャナーにおけるレジストリおよびファイルのフィルター機能の強化

InstallShield と共にインストールされる新しい Filters.xml を編集して、COM サーバーから必要のない COM データが抽出されるのを防ぐことができます。この Filters.xml ファイルを編集することにより、COM 抽出から除外されるレジストリ キーの一覧をカスタマイズすることができます。

Filters.xml ファイルに、スタティック依存関係スキャン、ダイナミック依存関係スキャン、および Visual Basic 依存関係スキャンで含めるまたは除外するファイルがリストされます。以前は、2 つの異なるファイル (Userscan.ini と Iswiscan.ini) が除外および選択の対象になるファイルをリストするために使用されていました。

詳細については、次を参照してください。

- COM 抽出のレジストリ変更をフィルターする
- 依存関係スキャナーでファイルをフィルターする

Internet Explorer 7.0 との互換性

Internet Explorer 7.0 と互換性を持たせるために、InstallShield の一部が今回改訂されました。主な例として、トランスフォーム ウィザード、カスタム アクション ウィザード、文字列テーブル エディター、[配布] ビュー、および [一般情報] ビューが更新されました。

強化されたスタート ページ

スタート ページで表示される最近開いたプロジェクトの一覧に、プロジェクトの種類を表示する列が挿入されました。また、一覧に表示されるプロジェクトの最大数が、4 つから 8 つに増えました。

InstallScript の強化機能および新機能

InstallScript 言語が一部強化されました。新規および改訂された変数、関数および定数が今回のリリースで提供されています。

システム変数の変更およびテキスト置換の追加

DISK1SETUPEXENAME と呼ばれるシステム変数が追加されました。

2 つのシステム変数には新しいデフォルト値が割り当てられました。

- UNINST
- UNINSTALL_STRING

次の 2 つの InstallScript テキスト置換が新しく提供されました。

- DISK1SETUPEXENAME
- SELECTED_LANGUAGE

さらに詳しい情報は、「システム 変数」を参照してください。

スクリプト変数の追加

2 つのスクリプト変数が新しく提供されました。

- INSTALLSCRIPTMSI
- BASICMSI

関数および定義済み定数の変更

新しい定義済み定数 LANGUAGE_SUPPORTED は、**Is** 関数と共に使用できます。詳しくは、「Is」を参照してください。

新しい ISOSL_WINVISTA 定義済み定数は、**FeatureFilterOS** 関数と SYSINFO 構造変数と共に使用できます。詳細については、「FeatureFilterOS 関数と SYSINFO」を参照してください。

DoInstall 関数は、**LaunchAppAndWait** 関数と類似しています。詳細については、「DoInstall」を参照してください。

アップグレードの詳細

InstallShield 12 では、セキュリティ関連の問題 (COM/DCOM)、およびその他の様々な理由で起こったインストールの失敗を解決するため、InstallScript MSI アーキテクチャが大幅に改良されました。InstallShield 12 でアーキテクチャが大幅に改良された理由は、これらの問題を解決して InstallScript MSI プロジェクトの安定性を高めるためです。この改良によって、InstallScript カスタム アクションを含む基本の MSI プロジェクト、および InstallScript の安定性も高まりました。

さらにマージ モジュール プロジェクト タイプが強化され、基本の MSI プロジェクトと同じ InstallScript ビューを使用できるようになりました。この新しく強化されたマージ モジュール プロジェクト タイプが、InstallScript MSI オブジェクト プロジェクト タイプ の代わりに使用されます。 InstallScript MSI オブジェクト プロジェクト タイプは InstallShield では使用できなくなりました。

詳細については、「InstallShield 11.5 以前のプロジェクトをアップグレードする」を参照してください。

ターゲット システムの要件

InstallShield を使って、Windows ベースのシステムをターゲットにするインストールを素早くビルド、テストおよび配布することができます。

デスクトップ コンピューターの要件

オペレーティング システム

ターゲット システムは、次のオペレーティング システムの最小要件を満たさなくてはなりません:

- Windows 7
- Windows Server 2008 R2
- Windows 8
- Windows Server 2012
- Windows Server 8.1
- Windows Server 2012 R2
- Windows 10
- Windows Server 2016
- Windows Server 2019
- Windows 11
- Windows Server 2022
- Windows Server 2025

ターゲット システムで、SSE2 インストラクション セットがサポートされていることが必須です。

インストーラー エンジンの要件

各インストーラー エンジンの最小ターゲット システム要件は、以下のとおりです。

テーブル 1-3 ・ デスクトップ コンピューターのターゲット システム要件

インストーラー エンジン	オペレーティング システム、その他の要件
Windows Installer 5.0 **(これは再配布可能ファイルとしては利用できません。)**	Windows 7 以降および Windows Server 2008 R2 以降
Windows Installer 4.5	Windows サーバー2003 SP1 以降
Windows Installer 4.0 **(これは再配布可能ファイルとしては利用できません。)**	Windows Server 2008

テーブル 1-3 • デスクトップ コンピューターのターゲット システム要件

インストーラー エンジン	オペレーティング システム、その他の要件
スイート/アドバンスト UI	Windows 7 以降および Windows Server 2008 R2 以降
アドバンスト UI	Windows 7 以降および Windows Server 2008 R2 以降
InstallScript	Windows 7 以降および Windows Server 2008 R2 以降

Windows Installer 再配布可能ファイルについての詳細は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

64 ビット オペレーティング システムをターゲットにする

Microsoft では、既存の 32 ビットのアプリケーションがシームレスに動作する 64 バージョンの Windows がデザインされています。また、同一コードが再コンパイルされたバージョンが 64 ビット アプリケーションとしてシームレスに動作する 64 ビット バージョンの Windows もデザインされています。このサポートを提供するために、64 ビット バージョンの Windows では、主に 2 つの方法で 32 ビットと 64 ビットをアプリケーションの分離ウィザードしています。それぞれのファイルは、別々の場所 (Program Files と. Program Files (x86); System32 と. SysWow64) に格納され、レジストリ キーは、別々に分かれています (HKLM¥Software と HKLM¥Software¥Wow6432Node)。

一部のケースでは、通常有益な分離となるものも、問題になることがあります (例、インストール)。通常、インストールは、(32 ビット マシンで実行できるように) それ自身が 32 ビットのアプリケーションであるため、64 ビットの場所にアクセスして、64 ビットのアプリケーションをインストールことは、標準的なファイル コピーやレジストリの書き込みにくらべ、かなり複雑です。InstallShield で提供されている各プロジェクト タイプでは、この複雑さを異なる方法で対応しています。

Windows Installer ベースのインストールでは、ターゲットが 64 ビット システムのみのときを除き、64 ビットの場所へのアクセスを許可していません。ただし、64 ビット システムでの実行時は常に 64 ビット DLL コードの実行がサポートされています。一部のターゲット システム (たとえば、 Windows Server Core システム) は、WOW64 (32-bit Windows-on-Windows) をサポートしません。InstallShield では、これらのシステム上で動作する64 ビット専用のインストールを作成できます。詳細については、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。

InstallScript のインストールは、**[リリース]** エクスプローラーの **[ビルド]** タブにある **"アーキテクチャ"** 設定に応じて、32 ビットまたは 64 ビットにできます。したがって、32 ビットまたは 64 ビット DLL を実行するように、または 32 ビットまたは 64 ビット .exe ファイルを起動するように構成できます。一部のターゲットシステムは、WOW 64 (32 – bit Windows – on – Windows) 機能をサポートしない場合があります。InstallShield では、すべてのコンポーネントを 64 ビットとして設定することで、そのようなシステムと互換性のある純粋な 64 ビット InstallScript インストールを作成できます。詳しくは、「64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする」を参照してください。

スイート/アドバンスト UI およびアドバンスト UI インストールでは、32 ビットおよび 64 ビットのいずれの場所も参照することができ、ターゲット システムに応じて、32 ビットおよび 64 ビットのいずれのインストールも起動することができます。ただし、スイート/アドバンスト UI またはアドバンスト UI プロジェクトのすべての拡張 DLL コードは、32 ビットである必要があります。詳細については、「64 ビット オペレーティング システムをスイート/アドバンスト UI およびアドバンスト UI インストールでターゲットにする」を参照してください。

64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallScript MSI プロジェクトを作成してから、"テンプレートの概要" プロパティを Intel から x64 に変更すると、ビルド時に 64 ビット MSI インストールと 32 ビット InstallScript MSI インストールの組み合わせが作成されます。

InstallScript MSI プロジェクトでは、アーキテクチャの検証がサポートされていないので注意してください。

64 ビット Windows ベースのシステム用のアプリケーションを構築する場合、64 ビット ファイルおよびその他のデータをインストールする手段が必要です。32 ビットの Windows Installer ベースのインストールを 64 ビットのマシンで実行することは可能ですが、64 ビットの場所にインストールすることはできません。Windows Installer サービスは、64 ビットのインストールをサポートしています。これらのセットアップは 32 ビットマシン上で設計およびビルドできますが、64 ビットのマシン上でしか実行できません。リリース フラグを使用して、単一のプロジェクトから 2 つのインストール (32 ビットと 64 ビット) をビルドすることができます。

"テンプレート概要" プロパティの構成

"テンプレート概要" プロパティでは、.msi パッケージが 32 ビット パッケージであるか 64 ビット パッケージであるかを指定します。

インストールが 64 ビット システムをターゲットにする場合、適切な 64 ビット値 (x64 または Intel64) を使用して "テンプレート概要" プロパティを構成します。この構成により、エンドユーザーが 64 ビット システムの 64 ビットの場所に貴社の製品をインストールできるようになります。また、エンドユーザーが貴社の製品を 32 ビット システムにインストールすることを防ぐこともできます。

[一般情報] ビューにある "テンプレートの概要" 設定を使って、"テンプレートの概要" プロパティを構成できます。[リリース] ビューで、製品構成について "テンプレートの概要" 設定を構成することもできます。[リリース] ビューに入力された値は、[一般情報] ビューに設定された値を上書きします。

詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。

64 ビット コンポーネントの作成

コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。1 以上の 64 ビット コンポーネントを含み、"テンプレートの概要" プロパティが 64 ビット値に設定されたリリースをビルドすると、その結果、64 ビットのパッケージが作成されます。

"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

COM データを 64 ビット COM サーバーから抽出する

InstallShield を 64 ビット オペレーティング システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。

InstallShield を 32 ビット システム上で使用している場合は、64 ビット COM 抽出機能は使用できません。

COM データの抽出についての詳細は、「COM 情報を COM サーバーから抽出する」を参照してください。

64 ビット COM サーバーの自己登録

InstallShield では、COM サーバーの 64 ビット自己登録がサポートされています。コンポーネントを 64 ビットとしてマークしてから、そのコンポーネントにファイルを追加したとき、そのファイルの [自己登録] チェック ボックスを選択して、インストール時の 64 ビット自己登録を有効にすることができます。[自己登録] チェック ボックスは、ファイルの [プロパティ] ダイアログ ボックスにあります。

InstallShield では、動的にリンクされている COM サーバーの 64 ビット自己登録もサポートされています。これを有効にするには、ファイルを動的に 64 ビット コンポーネントに追加する際、[ダイナミック ファイル リンクの設定] ダイアログ ボックスにある **[すべてのファイルを自己登録する]** チェック ボックスを選択します。詳細については、「ダイナミック ファイル リンクをコンポーネントに追加する」を参照してください。

レジストリ リフレクションの有効化と無効化

InstallShield では、Component テーブルに Windows Installer 4.0 属性 msidbComponentAttributesDisableRegistryReflection が含まれています。この属性を使用するには、[コンポーネント] ビューで選択したコンポーネントの "レジストリ リフレクションを無効にする" 設定を [はい] に設定します。レジストリ リフレクションによって、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期が保たれます。この設定は、Windows Installer 4.0 以降がある 64 ビット システムでのみサポートされています。他のシステムでは、この設定は無視されます。詳細については、「レジストリ リフレクションの有効化と無効化」を参照してください。

他の 64 ビット要素を追加する

InstallShield では、追加の 64 ビット要素もサポートされています。たとえば、InstallShield には、プロジェクトに追加することができる 64 ビットの .NET Framework 再配布可能ファイルが含まれています。カスタム アクションには 64 ビットの Jscript または VBScript コードを含めることができます。64 ビット フォルダー プロパティからは、64 ビット Program Files フォルダーなどの重要なオペレーティング システム フォルダーにアクセスすることができます。さらにセットアップ前提条件エディターを使って、ターゲット システム上の 64 ビット要件をチェックする条件を持つセットアップ前提条件を構成することができます。

***メモ・** 64 ビット サポートには、Windows Installer バージョン 2.0 以上が必要です。*

適切なアーキテクチャの検証を指定する

基本の MSI プロジェクトには、ビルド時にアーキテクチャの検証を使用するかどうかを指定できる製品構成設定が含まれています。アーキテクチャの検証を使って、インストールがターゲット システムのアーキテクチャと一致しない製品ファイルをインストールしようとしたり、ランタイム バイナリを使用したりする問題を引き起こす可能性のある状況を検出することができます。

たとえば、エンド ユーザーが WOW64 (32-bit Windows-on-Windows) サポートが搭載されていない 64 ビット Windows Server Core システムでインストールを実行する場合、アーキテクチャの検証を使って、インストールに含まれるすべての 32 ビット の製品ファイルおよび 32 ビットのカスタム アクション ファイルを識別できます。32 ビット バイナリを WOW64 サポートを搭載しない 64 ビット ターゲット システムにロードすることはできません。

アーキテクチャの検証を使って、ビルトイン InstallShield カスタム アクション DLL の 32 ビット バージョンのみを含む 32 ビット専用 .msi パッケージ、またはビルトイン InstallShield カスタム アクション DLL の 64 ビット バージョンのみを含む 64 ビット専用 .msi パッケージを生成することができます。

ビルド時に使用するアーキテクチャの検証の種類を指定し、32 ビット専用パッケージ、または 64 ビット パッケージのどちらをビルドするのかを識別するには、[リリース] ビューの製品構成で "アーキテクチャの検証" 設定を使用します。

詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

64 ビット セットアップ ランチャーで InstallScript MSI をサポート

InstallScript MSI プロジェクトは、32 ビットと 64 ビットの両方のWindows オペレーティング システム用にビルドできるだけでなく、64 ビットの Windows オペレーティング システム専用にビルドすることができるようになりました。**[アーキテクチャ]** 設定を構成すると、**[メディア]** ビューの **[リリース]** エクスプローラーにあるリリースの **[ビルド]** タブで、アプリケーションを実行するために必要なターゲット システムに基づいて、setup.exe ファイル (セットアップランチャー) を作成できます。詳細については、「リリースの [ビルド] タブ」を参照してください。

単一プロジェクトから 2 つのインストール (32 ビットと 64ビット) をビルドするためのヒント

単一プロジェクトから 2 つのインストール (32 ビットおよび 64 ビット) をビルドできるようにするためには、リリース フラグの使用を考慮してください。必要に応じてリリース フラグを機能、InstallShield 前提条件、および連鎖 .msi パッケージに割り当てて、32 ビットのアイテムと 64 ビットのアイテムとを区別します。次に、[リリース] ビューの各リリースまたは製品構成において、ビルド時に特定のリリース フラグを持つアイテムを含めるか除外するかを構成します。リリース フラグに基づいて特定のカスタム アクションを条件付きで起動することによって、32 ビット インストール中に 32 ビット カスタム アクションを起動し、64 ビット インストール中に 64 ビット カスタム アクションを起動することもできます。

詳しくは、次を参照してください:

- リリース フラグ
- リリース フラグに基づくフィルター
- カスタム アクションをリリース フラグに基づいて条件付きで起動する

InstallShield では、プロジェクト内の各リリースでプロジェクトのパス変数の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、この機能を使ってカスタム アクションのバイナリ ファイルを入れ替えることができます。個別の 32 ビットと 64 ビットのターゲット システム用に個別のリリースを作成した場合、カスタム アクションに選択された DLL を参照するパス変数をオーバーライドすることができます。これによって、InstallShield が 32 ビット リリースには 32 ビット DLL を、また 64 ビット リリースには 64 ビット DLL を含みます。インストールがインストール処理を行っているファイルを入れ替えるのにパス変数のオーバーライドを行うことはお勧めできません。これは、ファイルの 32 ビット バージョンと 64 ビット バージョンに個別のコンポーネントを使用しなくてはならないためです。

プロジェクトで 1 つ以上のパス変数をオーバーライドする場合、[リリース] ビュー内のリリースの [ビルド] タブに追加された "パス変数のオーバーライド" 設定を使います。詳細については、「リリースの [ビルド] タブ」を参照してください。

Using 64-Bit Windows Installer Packages (Windows Installer ヘルプ ライブラリ)

64 ビット セットアップ ランチャーのビルド

64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

1 つの InstallScript プロジェクトから、32 ビット システムの 32 ビットの場所、および、必要に応じて、64 ビット システムの 64 ビットと32 ビットの場所にインストールされるインストールを 1 つ作成することができます。

InstallScript インストールが 32 ビットの場合、32 ビットの DLL コードのみ実行 (または、32 ビットまたは 32 ビット .exe ファイルを起動) します。ただし、すべての InstallScript インストールでは、64 ビット コンポーネントの設定およびその他の方法を通じて、64 ビットの場所へのアクセスがサポートされています。

64 ビット System32 フォルダーの読み取りと書き込み

InstallScript には、ターゲット システム上の 32 ビットの System32 フォルダーと 64 ビットの System32 フォルダーのいずれかを選択できる次のシステム変数が使われています:

- WINSYSDIR
- WINSYSDIR64

WINSYSDIR64 変数は 64 ビット Windows フォルダーに設定されていますが、64 ビット Windows には、自動的に 32 ビットアプリケーション (InstallScript エンジンなど) を 32 ビット System32 フォルダー (SysWOW64) にリダイレクトする機能が含まれています。従って、WINSYSDIR64 にファイルとフォルダーをインストールする場合、64 ビットとマークされているコンポーネント内にフォルダーまたはファイルを配置して、ファイル システムのリダイレクトを無効にできます。コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

代わりに、定数 WOW64FSREDIRECTION を Disable と Enable いう 2 つの関数を使って、InstallScript コードから WINSYSDIR64 フォルダーに書き込みまたは読み取りを実行している間に 64 ビット Windows のファイル システムのリダイレクトを無効にして、書き込み/読み取りが完了したら、再度有効にすることができます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。

その他の 64 ビットのインストール先フォルダーの読み取りと書き込み

InstallScript には、ターゲット システム上の 64 ビットの場所をターゲットにするための、いくつかの追加システム変数が提供されています:

- COMMONFILES64
- FOLDER_APPLICATIONS64
- PROGRAMFILES64

64 ビットのレジストリの場所の読み取りと書き込み

64 ビット バージョンの Windows には、一部の 64 ビットのレジストリの場所を対応する 32 ビットの場所にマップする機能が含まれています。たとえば、64 ビット バージョンの Windows では、HKEY_LOCAL_MACHINE¥Software¥Wow6432Node は、32 ビットの HKEY_LOCAL_MACHINE¥Software に相当します。

レジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット領域のレジストリにインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

代わりに、REGDB_OPTION_WOW64_64KEY オプションを REGDB_OPTIONS 変数と一緒に使って、InstallScript コードでレジストリの編集または読み取りを実行中に、レジストリのリダイレクトを回避し、REGDB_OPTION_USE_DEFAULT_OPTIONS オプションを REGDB_OPTIONS と一緒に使って、32 ビット領域のレジストリへの変更を有効にすることもできます。インストールが利用する可能性のある Windows 機能にはファイル システム リダイレクトを有効にしておく必要があるため、Windows ドキュメンテーションではリダイレクトを無効にするのはそれが必要な場合のみにとどめることが推奨されています。

自己登録型の 64 ビット COM サーバーのインストール

InstallShield では、COM サーバーの 64 ビット自己登録がサポートされています。自己登録型の DLL、.exe、.tlb、または .olb ファイルを含むコンポーネントの場合、コンポーネントの "自己登録" 設定で [はい] を選択します。

自己登録に対してファイル システムのリダイレクトは無効にできないため、32 ビットの自己登録型ファイルは 64 ビットの System32 フォルダーにはインストールできません。このため、32 ビットの System32 フォルダーにインストールする必要がある 32 ビットの自己登録型 COM サーバーがある場合、COM サーバーを含むコンポーネントの "64 ビットのコンポーネント" 設定で [いいえ] が設定されていることを確認してください。

[プログラムの追加と削除] 情報について

InstallScript インストールは、レジストリの 64 ビット部分にある [プログラムの追加と削除] 情報の読み書きをサポートしません。したがって、InstallScript インストールは、レジストリの 32 ビット部分に [プログラムの追加と削除] 情報をインストールします。また、REGDB_OPTION_WOW64_64KEY オプションを使って、**CreateInstallationInfo**、**MaintenanceStart**、**RegDBGetItem**、**RegDBSetItem**、**RegDBGetAppInfo**、**RegDBSetAppInfo**、および **RegGetUninstCmdLine** などのレジストリ関数に対して、レジストリ リフレクションが無効にすることはできません。

64 ビット オペレーティング システムをスイート/アドバンスト UI およびアドバンスト UI インストールでターゲットにする

***プロジェクト・**この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・**アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

メモ・デフォルトで、スイート/アドバンスト UI およびアドバンスト UI インストールは 32 ビットです。必要な場合、64 ビット インストールのリリース構成の下にある ""64 ビット セットアップ ランチャー" 設定を変更することができます。64 ビット セットアップ ランチャーについては、「64 ビット セットアップ EXE のビルド」を参照して下さい。

スイート/アドバンスト UI およびアドバンスト UI インストールでは、32 ビットおよび 64 ビットのいずれの場所も参照することができ、ターゲット システムに応じて、32 ビットおよび 64 ビットのいずれのインストールも起動することができます。

たとえば、スイート/アドバンスト UI プロジェクトに、32 ビット システムをターゲットするインストールを 1 つと、64 ビット システムをターゲットするインストールを 1 つ含める場合、これら両方のパッケージに対して、[パッケージ] ビューで対象条件を作成できます。32 ビット パッケージの場合、対象条件には、x86 アーキテクチャの有無を確認するプロファイル条件が含まれ、64 ビット パッケージには、x64 または IA64 アーキテクチャの有無を確認するプラットフォーム条件が含まれます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

32 ビットのオペレーティング システムを 64 ビット InstallScript セットアップ ランチャーを使ってターゲットする

プロジェクト・この情報は、InstallScript プロジェクトに適用します。

InstallShield では、64 ビット セットアップ ランチャーと 64 ビットInstallScript エンジン ファイルを使用して、64 ビット Windows オペレーティング システムを対象とした InstallScript プロジェクトを構築することができます。64 ビット システムでインストールを起動すると、64 ビットファイル システムとレジストリの場所をターゲットにして、完全に 64 ビット環境内で実行されます。

InstallScript ベースのプロジェクトの 64 ビット アーキテクチャ オプションでは、純粋な 64 ビット パッケージをビルドできますが、64 ビット プラットフォーム上の 32 ビットファイルシステムの場所と 32 ビット レジストリ ビューへのアクセスもサポートしています。必要に応じて、適切な REGDB_OPTIONS オプションとシステム変数を使用してこれを達成できます。

その他の 32 ビットのインストール先フォルダーの読み取りと書き込み

InstallScript には、64 ビット ターゲット システム上の 32 ビットの場所をターゲットにするための、いくつかのシステム変数が提供されています:

- COMMONFILES
- FOLDER_APPLICATIONS
- FOLDER_APPLICATIONS32
- PROGRAMFILES

64 ビット プラットフォーム上の 32 ビットのレジストリの場所の読み取りと書き込み

Windows の 64 ビット バージョンでは、64 ビット アプリケーション (64 ビット InstallScript エンジンなど) は、デフォルトで適切な 64 ビットレジストリビューにアクセスします。レジストリ データをレジストリの 64 ビット領域ではなく 32 ビット領域に明示的にインストールしたい場合は、32 ビットとマークされたコンポーネントにレジストリ データを配置することができます。コンポーネントが 32 ビットであると指定するには、**[コンポーネント]** ビューでコンポーネントの **"64 ビット コンポーネント"** 設定で **[いいえ]** を選択します。"64 ビット/32 ビットコンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください:

別の方法として、REGDB_OPTION_WOW 64_32 KEY オプションをREGDB_OPTIONS 変数とともに使用すると、システム アーキテクチャに関係なく、すべてのレジストリ操作をレジストリの 32 ビット部分に転送できます。

64 ビット プラットフォームに自己登録型の 32 ビット COM サーバーをインストール

InstallShieldは、64 ビット Windows オペレーティング システムに 32 ビット COM サーバーをインストールして自己登録するためのサポートを提供します。自己登録 COM サーバーは、通常 **DllRegisterServer** を呼び出すことによって、インストール中に Windows レジストリに自己登録を行うロジックを含む DLL または EXE です。32 ビット COM サーバーを 64 ビット プラットフォームにインストールする場合、環境の 32 ビットの場所に正しくインストールされ、登録されていることを確認することが重要です。

64 ビットマシンの SysWOW 64 フォルダーに 32 ビットの自己登録型 COM サーバーをインストールするには、COM サーバーを含む **[コンポーネント]** ビューでコンポーネントの **"64 ビット コンポーネント"** 設定に **[いいえ]** が選択されていることを確認します。

InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い

管理者権限を持たずに InstallShield を起動した場合、以下の機能は使用できなせん:

- **COM 抽出**—COM サーバーからの COM 情報の抽出には、管理者権限が必要です。

  InstallShield を管理者権限を持たずに実行している場合に、プロジェクトで COM サーバーから COM 情報の抽出を行うように指定してからリリースをビルドしようとすると、ビルド エラー -6017 が発生します。

- **再配布可能ファイルのダウンロード**—[再配布可能ファイル] ビュー内から再配布可能ファイルをダウンロードするには、管理者権限が必要です。これは、InstallShield がファイルをマシンごとの場所にダウンロードする際に管理者権限が必要なためです。

  [再配布可能ファイル] ビュー内から再配布可能ファイルをダウンロードしようとしたとき、管理者権限が無い場合には次のメッセージが表示されます:

  ダウンロードが失敗しました。管理者として実行していること、および使用中のマシンがインターネットに接続されていることを確認してください。再試行しますか?

- **InstallShield 前提条件に [すべてのユーザー] の場所を指定できる機能** — [オプション] ダイアログ ボックスの [前提条件] タブを使って、[再配布可能ファイル] ビューおよび [前提条件] ビューに表示する InstallShield 前提条件を含むフォルダーを指定できます。このタブで [すべてのユーザー] の場所を変更するためには管理者権限が必要です。これは、InstallShield がレジストリ内でマシンごとの場所に情報を書き込むためです。したがって、InstallShield を管理者権限を持たずに実行している場合、このタブの [すべてのユーザー] の場所は無効となっています。

- **マージ モジュールに [すべてのユーザー] の場所を指定できる機能** – [オプション] ダイアログ ボックスの [マージ モジュール] タブを使って、[再配布可能ファイル] ビューに表示するマージ モジュールを含むフォルダーを指定できます。このタブで [すべてのユーザー] の場所を変更するためには管理者権限が必要です。これは、InstallShield がレジストリ内でマシンごとの場所に情報を書き込むためです。したがって、InstallShield を管理者権限を持たずに実行している場合、このタブの [すべてのユーザー] の場所は無効となっています。
- **Regasm.exe および InstallUtilLib.dll の場所を編集できる機能** – [オプション] ダイアログ ボックスの [.NET] タブで、.NET Framework に含まれているユーティリティである Regasm.exe および InstallUtilLib.dll ファイルの場所を指定できます。これらのユーティリティは COM interop と .NET カスタム アクションで利用されます。[.NET] タブでこれらの場所を変更するためには管理者権限が必要です。これは、InstallShield がレジストリ内でマシンごとの場所に情報を書き込むためです。したがって、InstallShield を管理者権限を持たずに実行している場合、このタブの場所設定は無効となっています。

管理者コンテキストと非管理者コンテキストの切り替えを行い、プロジェクトでマップされたドライブの場所を使用している場合、問題が発生することがあります。たとえば、管理者権限を持たずに Windows Explorer を使ってドライブ名を共有ネットワーク フォルダーにマップした場合、InstallShield の非管理者インスタンスではこのドライブにアクセスできますが、管理者インスタンスではアクセスできません。同様に、管理者権限を持っている場合に Windows Explorer を使ってドライブ名を共有ネットワーク フォルダーにマップすると、InstallShield の管理者インスタンスではこの場所にアクセスできますが、非管理者インスタンスではアクセスできません。このため、プロジェクトでネットワークの場所を参照する場合は、UNC パス (たとえば、¥¥server¥share) を使用するか、管理者と非管理者の両方でドライブ名をマッピングすることをお勧めします。

Visual Studio 内から InstallShield を使用している場合、管理者権限を持たない可能性があります。デフォルトでは、Windows Vista 以降のシステム上で Visual Studio のショートカットをダブルクリックして起動した場合、管理者権限はありません。

タスク ***Windows Vista 以降のシステム上で管理者権限を使って Visual Studio 内部から InstallShield を実行するには、以下の手順に従います:***

1. [スタート] メニューで Visual Studio のショートカットを右クリックしてから、**[管理者として実行]** をクリックします。
2. 新しい InstallShield プロジェクトを作成するか、または既存のプロジェクトを開きます。詳細は、以下のどちらかを参照してください:
   - Microsoft Visual Studio で InstallShield プロジェクトを作成する
   - Microsoft Visual Studio で InstallShield プロジェクトを開く

32 ビットと 64 ビット システムにおけるインストールの開発およびビルドの違い

InstallShield は、32 ビットと 64 ビット システムの両方で実行できる 32 ビット アプリケーションです。場合によって、InstallShield は、32 ビット システムで使用されているか、または 64 ビット システムで使用されているかによって動作が異なります。また、使用しているオペレーティング システムによっても、違いが見られることがあります。次のセクションは、これらの違いを説明します。

InstallShield 同様、コマンドライン ビルド (ISCmdBld.exe)、Standalone Build、およびオートメーション インターフェイスも、32 ビット アプリケーションです。したがって、同じプラットフォーム特有の違いは、これらのツールでも発生します。

オートメーション インターフェイスのロード

オートメーション インターフェイスは 32 ビット インターフェイスを持つため、32 ビット プロセスからロードしなくてはなりません。オートメーション インターフェイスを 64 ビット マシンで使用する場合、これを 32 ビット実行可能ファイルを使ってロードする必要があります。そうしないと、エラーが発生する可能性があります。

詳細については、「64 ビット システム上でオートメーション インターフェイスを使用する」を参照してください。

プロジェクトへシステム ファイルを追加する

64 ビット システム上で InstallShield を使用している場合に、ソースの場所が開発マシン上の 64 ビット システム フォルダー (System32) であるシステム ファイルをプロジェクトに追加するとき、[ファイルとフォルダー] ビューの上部にあるソース コンピューターのペインからインストール先コンピューターのペインの適切な場所にドラッグすることはできません。

開発マシン上の 64 ビット System32 ファイルを InstallShield プロジェクトに追加するには、マシン上の Sysnative フォルダーを参照してから、プロジェクトに適切なファイルを選択します。詳細については、「64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する」を参照してください。

64 ビット開発システム上で、ソースマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示する

InstallShield を 64 ビット開発システム上で使用する場合、InstallShield が表示する [レジストリ] ビューは、使用中のマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示します:

- HKEY_LOCAL_MACHINE¥Software
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node

このサポートによって、プロジェクトのレジストリ データの変更を構成する際、これらのソース領域からこのビューのインストール先ペインの適切な領域にエントリーをドラッグ アンド ドロップすることが可能となります。

インストールがレジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット ターゲット システム上のレジストリの 64 ビット領域にインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。[レジストリ] ビューのソースペインから 64 ビット データを、ビュー内のインストール先ペインにドラッグするだけでは、そのコンポーネントが 64 ビットであるとマークされません。詳細は、次を参照してください:

- 64 ビット オペレーティング システムをターゲットにする
- レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する

COM データを 64 ビット COM サーバーから抽出する

デザイン タイムまたはビルド時に、64 ビット COM サーバーから COM データを抽出する場合、64 ビット オペレーティング システム上で InstallShield を使用する必要があります。InstallShield を 32 ビット システム上で使用している場合は、64 ビット COM 抽出機能は使用できません。

COM データの抽出についての詳細は、「ビルド時に COM 登録データを抽出する」を参照してください。

64 ビット ファイルをスキャンして依存関係を確認する

InstallShield で依存関係スキャナーを使って、プロジェクトに追加する必要がある可能性がある依存関係を判別する場合、64 ビット ファイルの依存関係の確認は、InstallShield を 64 ビット オペレーティング システムでを使用している場合のみ可能です。

32 ビット システムで InstallShield を使用している場合、InstallShield の依存関係スキャナーを使って、プロジェクトの 64 ビット ファイルの依存関係を確認することはできません。32 ビット システムで 64 ビットの依存関係をプロジェクトに追加する場合、必須ファイルおよびマージ モジュールをプロジェクトに手動で追加する方法があります。

ファイルの依存関係のスキャンに関する詳細は、「アプリケーションの依存関係を識別する」を参照してください。

依存関係の 64 ビット .NET アセンブリとプロパティをスキャンする

Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用していて、いずれかのビルトイン処理によって依存関係を検出する場合、InstallShield はプロジェクトに含まれる 64 ビット .NET アセンブリの 64 ビット依存関係をスキャンすることができます。

これらの処理は、プロジェクトに含まれる 32 ビット .NET アセンブリの 32 ビット依存関係のスキャンも行います。

Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのビルトイン スキャンはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみを検出できます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

詳しくは、次を参照してください:

- .NET アセンブリのプロパティおよび依存関係を識別する
- 依存関係の 64 ビット .NET アセンブリをスキャンする

64 ビット .NET Installer クラスと COM Interop を使用する

Windows の 64 ビット バージョンで InstallShield を使用している場合、InstallShield の [ツール] メニューにある [オプション] をクリックすると表示される [オプション] ボックスでは、今回より、.NET Framework に含まれている Regasm.exe および InstallUtilLib.dll ファイルの場所として、32 ビット と 64 ビットの 2 つのパスを指定できます。InstallShield は、.NET Installer クラスと COM Interop を含むリリースで、ビルド時に指定されたパスを使用します。

64 ビット バージョンの Windows 上で ISCmdBld.exe を使ってコマンドラインからビルドしている場合に、既存の -t パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ISCmdBld.exe は、64 ビット .NET Installer クラスと COM Interop 用に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

MSBuild または Team Foundation Server (TFS) を使ってビルドしている場合に、InstallShield タスクで既存の DotNetUtilPath パラメーターを使って .NET Framework の 32 ビット バージョンのパスを指定すると、ビルドは、64 ビット .NET Installer クラスと COM Interop 用に Regasm.exe および InstallUtilLib.dll の 64 ビットの場所を使用します。

32 ビット オペレーティング システムで InstallShield を使用している場合、64 ビットの場所の設定は無効になり、64 ビット .NET Installer クラスと COM Interop の 32 ビット サポートのみ有効になります。このサポートは、ISCmdBld.exe を使ってコマンドラインから実行するビルド、および、MSBuild または TFS を介して実行するビルドにも適用します。

詳しくは、次を参照してください:

- [.NET] タブ ([オプション] ダイアログ ボックス)
- ISCmdBld.exe
- Microsoft ビルド エンジン (MSBuild)

ヘルプの使い方

Revenera は、役に立つ情報やヘルプ リソースがいつでもすぐ取り出せることがいかに重要かを理解しています。InstallShield インターフェイスの様々なビュー内部に埋め込まれているインライン ヘルプとは別に、InstallShield には製品と共にインストールされるヘルプ ライブラリ (Web で使用できるオンライン ヘルプ システム、および PDF フォーマットで提供されるドキュメント) が含まれています。

InstallShield ヘルプ ライブラリ

製品に関してご不明な点がありましたら、まず InstallShield ヘルプ ライブラリを参照してください。これは総合的なユーザー ガイドです。

InstallShield の **[ヘルプ]** メニューから InstallShield ヘルプ ライブラリにアクセスするには、F1 を押すか、インターフェイスの [ヘルプ] ボタンをクリックします。

InstallShield ヘルプ ライブラリの利用に、インターネットの接続は必要ありません。オンライン ヘルプ ビューアーは、基本的に、個人のニーズに基づいてテクニカル情報を表示、検索、フィルターするためのツールです。

コンテキスト ヘルプを使用する

プロジェクトの作業中に、ソフトウェア オブジェクトをクリックすると、[ヘルプ] ウィンドウにヘルプ情報が表示されます。これは、「コンテキスト」ヘルプとも呼ばれます。

ヘルプ情報は、エクスプローラー ウィンドウで表示される選択されたソフトウェア オブジェクトの各プロパティでも参照することができ、プロパティの設定方法を見ることができます。

Web ベースのオンライン ヘルプ

Web ベースのオンライン ヘルプは、24 時間いつでも、http://docs.revenera.com から利用することができます。このヘルプ リソース センターより、最新の情報がほぼリアル タイムに入手可能です。

PDF 版のドキュメント

InstallShield ドキュメントの PDF 版は https://community.revenera.com/s/ からご利用いただけます。

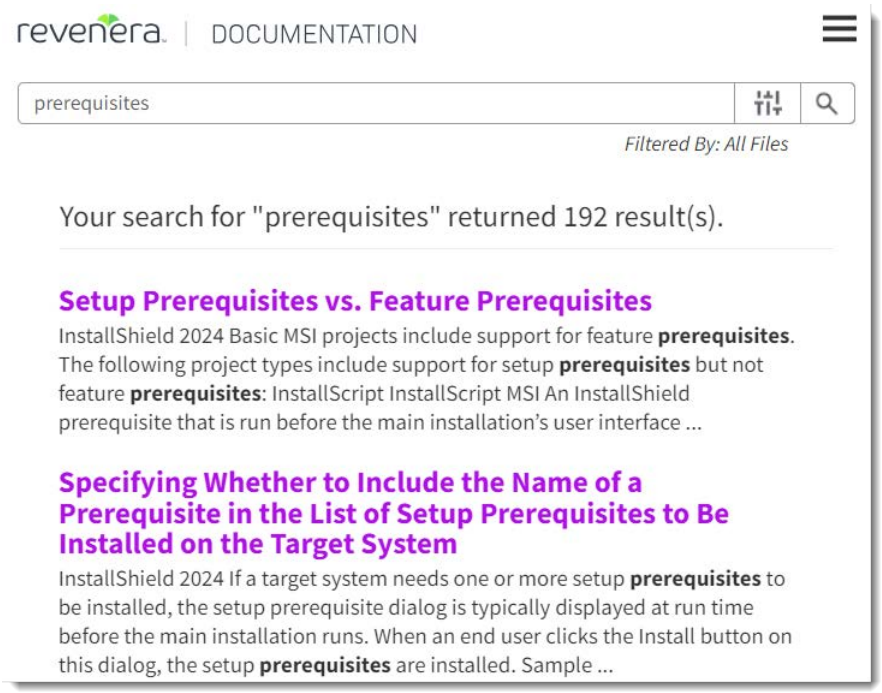
docs.revenera.com でプロジェクトの種類ごとに検索結果を絞り込むことができる機能

InstallShield 2024 から、docs.revenera.com の InstallShield Help Library (英語版) を参照する時、プロジェクトの種類ごとに検索結果を絞り込むことができます。

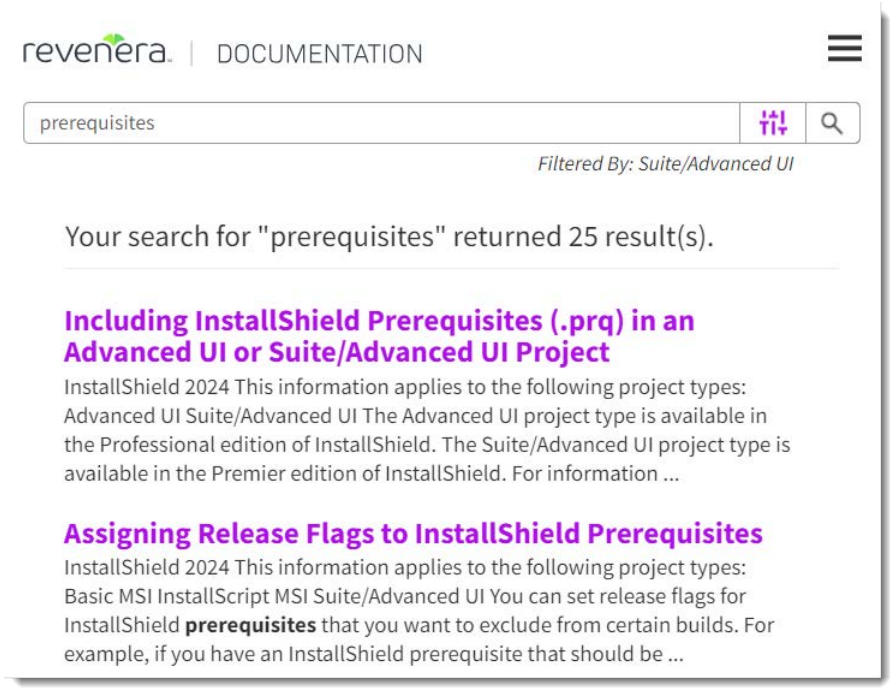
ブラウザー ウィンドウの右上にある**検索**バーに**フィルター** アイコンが表示され、選択したフィルターがアイコンの下に表示されます。

タスク ***検索結果を絞り込むには、次の手順に従います:***

1. **検索**ボックスに検索語を入力し、虫メガネをクリックするか、エンター キーを押します。検索結果が表示され、結果の総数が一覧表示されます(この例では**192**件):



2. **フィルター**アイコンをクリックし、**[スイート/アドバンスト UI]** など、検索結果を絞り込みたい**プロジェクトの種類**を選択します。絞り込まれた検索結果が表示され、選択した**プロジェクト タイプ** に適用するように指定されたトピックのみが一覧表示されます (この例では **25** 件):



***メモ・** InstallShield ヘルプ ライブラリのプロジェクトの種類で絞り込めるトピックには、そのトピックがどのプロジェクトの種類に適用されるかを示す**プロジェクト** メモが上部に表示されます。*

お問い合わせ先

Revenera は本社をイリノイ州イタスカに置き、世界各地に拠点を持ちます。製品についてのお問い合わせなど、連絡先は当社の Web サイトをご覧ください:

http://www.revenera.com

2

スタート ガイド

InstallShield では、安定した Windows Installer および InstallScript インストールを簡単に作成することができる強力な機能と、作業時間を大幅に節約することができるツールが多数提供されています。InstallShield ヘルプ ライブラリは、InstallShield で提供されている機能をより効果的に活用するためのリソースです。まず最初にどのトピックから読み始めるかは、InstallShield インストール作成ソフトウェアの利用経験によって異なります。下のテーブルは、それぞれの経験レベルに応じて様々なトピックにリンクしています。

テーブル 1 ・ スタート ガイド ロード マップ

経験レベル	トップ ヘルプ トピック
インストールを以前に作成した経験なし。	インストールに関する一般的な情報については、次のトピックを参照してください。 ・ インストールの基本 ・ アプリケーション ライフサイクル
InstallShield を利用するのは初めて。	インストール作成経験はあるが、InstallShield を利用するのは初めての場合、次のトピックを参照してください。 ・ InstallShield インターフェイスを使って作業する ・ プロジェクトについて ・ 基本の MSI プロジェクト チュートリアル ・ グローバリゼーション チュートリアル ・ InstallScript プロジェクト チュートリアル ・ InstallShield 実行時の言語サポート ・ サポートされているアプリケーション プログラミング言語

テーブル 1 • スタート ガイド ロード マップ (続き.)

経験レベル	トップ ヘルプ トピック
他の Revenera 製品を利用した経験がある。	Revenera の他の製品を利用したことがある場合、次のトピックを参照してください。 • 以前のバージョンの InstallShield からアップグレードする • InstallShield の他のエディションからアップグレードする • プロジェクト アシスタント • インストール プロジェクトの概要
InstallShield の中級、または上級レベルユーザー。	InstallShield を使い慣れている場合、次のトピックの参照をお勧めします: • コマンドライン ツール • エラーと警告 • InstallScript 言語リファレンス • サイレント モードでインストールを実行する • InstallShield の詳細設定を構成する

インストールの基本

インストールとは、一言で説明すると、ファイルおよびプログラムをユーザーのマシンにインストールするために使われる「パッケージ」を意味します。インストールは、インストーラー エンジンと対話するロジック、並びにアプリケーション ファイルで構成される集合体です。インストールの最も基本的な役割は、アプリケーション ファイルをソース媒体からエンド ユーザーのコンピューターへ転送することです。Windows オペレーティング システムは大変複雑なため、InstallShield のようなユーティリティのヘルプなしに、効果的で首尾一貫したインストールを簡単に作成することはできません。

インストールは、「製品」、「機能」、「コンポーネント」の 3 つの階層に分けられます。次の図はこの階層を説明します。

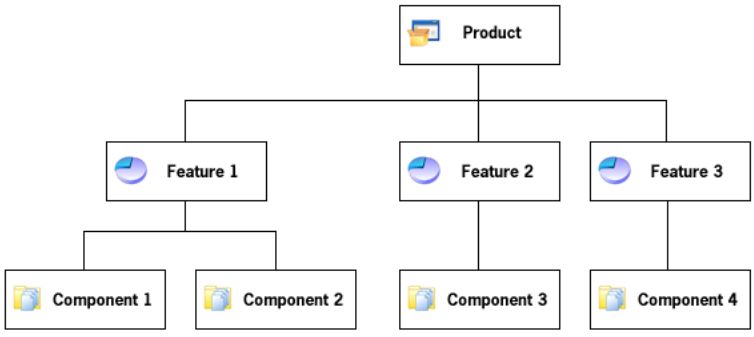


図 1: インストールの階層構造

*製品*は、インストール プロジェクトの中で最も上位にある構成区分です。製品は、通常 1 つのメイン アプリケーション (たとえば、ワードプロセッサ) と、そのアプリケーションが必要とするすべてのファイルとデータで構成されます。一連のアプリケーションが 1 つの製品となることも可能です。

*機能*は、エンドユーザーから見て、個別にインストール可能な最小の製品構成単位です。インストール プログラムの設計者は一般的に、どの機能をインストールし、どの機能をソース メディアに残すのかをユーザーが選択できるようにします。ワードプロセッサ製品では、メインの実行可能ファイルが 1 つの機能で、オプションの辞書が別の機能となっている場合があります。機能には必要な要素のすべてが含まれており、兄弟機能を必要としないのが原則です。たとえば、類語辞典機能はユーザーがインストールしないことを選択できる辞書機能を必要とはしません。ただし、機能にサブ機能を含めることもできます。サブ機能があると、エンドユーザーはインストールするファイルやデータをさらに細かく選択できるようになります。

プロジェクト内の各機能は 1 つ以上の*コンポーネント*で構成されます。コンポーネントは、インストール開発者から見て、個別にインストール可能な最小の製品構成単位です。コンポーネントはエンド ユーザーには見えないようになっています。各コンポーネントは類似のプロパティを持つファイル (およびその他のリソース) を含んでいます。たとえば、コンポーネント内のファイルはすべてエンド ユーザーのマシン上の同じディレクトリにインストールされます。また、コンポーネント内のファイルはすべて同じオペレーティング システムまたは言語に適用される必要があります。辞書機能には複数の言語別辞書コンポーネントが含まれている場合が考えられます。コンポーネントはファイルだけでなく、レジストリデータ、ショートカット、ファイル拡張子情報、およびユーザーのマシンに書き込まれるその他のシステムデータを通常含んでいます。

インストールを設計する際の総合的なタスクは、ファイルのインストール先、オペレーティング システム、言語、およびその他のプロパティに基づいて製品のファイルをコンポーネントに分割し、各コンポーネントを 1 つまたは複数の機能に関連付けることです。

アプリケーション ライフサイクル

アプリケーションのライフ サイクルは、顧客がアプリケーションをインストールした段階ですべて終了ではありません。ソフトウェア ベンダーとして、初回インストールが顧客のデスクトップで完了しただけでは、アプリケーションが成功したとは言えません。顧客は、製品のアップデートや強化内容、および重要な情報に簡単にアクセスできることを当然と考えています。この意味で顧客とのコミュニケーションやアプリケーションの状態を監視できることは、将来的な利益と発展には非常に重要です。

ソフトウェア ベンダーの中には、顧客主導のコミュニケーションを要求しがちですが、積極的に顧客との対話の機会を持たないベンダーは、多くのチャンスを逃すことになります。顧客が Web サイトまたはユーザー コミュニティを頻繁に訪れない限り、彼らはアップデート、アップグレード、パッチ、および一般的な技術記事などの情報を見逃すことになり、ベンダーは収益とサービスの機会を逃す結果となります。

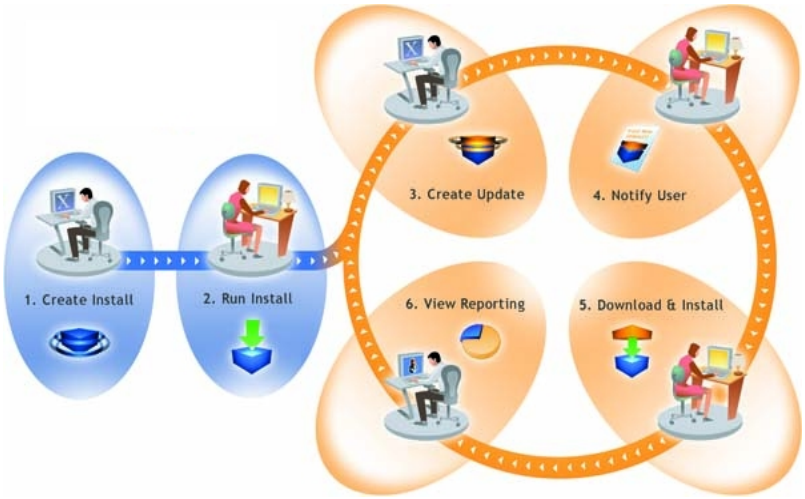


図 2: FlexNet Connect のアプリケーション ライフサイクルの管理方法

上記のダイアグラムは、アプリケーション ライフサイクルの管理における FlexNet Connect の役割を図式したものです。

1. **インストールの作成**—を利用して、ソフトウェア開発者はすべてのプラットフォーム上で実行可能なインストールを容易に作成することができます。
2. **インストールの実行**—InstallShield 技術を使って作成されたインストールは、世界中 4 億以上のマシン上にインストールされています。
3. **アップデートの作成**—InstallShieldを使用すると、ソフトウェア開発者は素早くパッチおよびアップデートをビルドすることができます。
4. **ユーザーへの通知**—FlexNet Connect は、新しいアップデートのインストールが入手可能であることを各ユーザーに通知します。
5. **ダウンロードとインストール** — FlexNet Connect はアップデートのダウンロードおよびインストールをシームレスに統合し、一括処理します。
6. **レポートの表示** — FlexNet Connect はアップデートの利用率について即座にフィードバックを提供します。

InstallShield の起動

InstallShield を開始すると（製品インストール語の初回起動を除く）、すぐに InstallShield スタート ページが表示されます。スタート ページを使うと、製品情報、前回開いたプロジェクト、InstallShield リソースに簡単にアクセスすることができます。

InstallShield スタート ページ

InstallShield スタートページから、製品情報、最近開いたプロジェクト、InstallShield のリソースに簡単にアクセスすることができます。[スタート ページ] は次のセクションに分かれています。

テーブル 2 • スタート ページのセクション

セクション	説明
プロジェクト タスク	プロジェクト タスクをクリックして、新規プロジェクトを簡単に作成したり、既存プロジェクトを開くことができる他、InstallShieldのインストール付属のサンプル プロジェクトを参照することもできます。
ヘルプ トピック	アクセスの多いヘルプ トピックをこのセクションに示します。[スタート] ページから InstallShield ヘルプ ライブラリ全体にアクセスするには、F1 を押すか [リソース] セクションのヘルプ ライブラリ リンクをクリックします。
(最近開いたプロジェクト)	スタート ページの中央には、前回アクセスしたプロジェクト、プロジェクト タイプ、および前回変更が加えられた日時が表示されます。
スタート ガイド	InstallShield およびインストール作成ツールに関する知識レベルに基づいて、InstallShield ヘルプ ライブラリのどのトピックを読むべきかを案内するスタート ガイドをクリックしてください。
リソース	[リソース] セクションには、役に立つ InstallShield 情報へのリンクが表示されています。 • **ヘルプ ライブラリ** — InstallShield のマニュアルを表示します。 • **InstallShield コミュニティ**—他の InstallShield ユーザーとの情報交換、質問の投稿、または回答の検索ができる Web ベースのフォーラムです。 • **オンライン セミナー** — InstallShield の評価、また Revenera 製品を最大限に利用するのに役立つ Web ベースの無料セミナーへのリンクです。 • **ダウンロード**—最新版の InstallShield 前提条件、InstallShield マージ モジュール、オブジェクト、ならびにサービス パック、パッチなど、ご使用中の InstallShield バージョンで利用できるダウンロードへのリンクです。 • **リリース ノート** — Revenera Web サイトのナレッジ ベースに掲載されているリリース ノートを表示します。 • **既知の問題**—ご使用中の InstallShield バージョンに関する既知の問題を取り扱うナレッジベース記事を表示し、ワークアラウンドおよび解決策を提供します。 • **アップグレード アラート** —ナレッジベース記事が表示され、InstallShield の古いバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードしたときに発生する可能性がある問題についての情報が提供されます。また、新しい InstallShield 2025 プロジェクトと以前のバージョンからアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。 • **RSS フィード** — InstallShield ナレッジ ベース記事をサブスクライブできる Web ページが表示されます。

テーブル 2・スタート ページのセクション (続き.)

セクション	説明
お問い合わせ	Revenera Web サイトの [サポート] にアクセスするには、このセクションにあるリンクの 1 つをクリックしてください。

プロジェクトについて

InstallShield プロジェクトは、プロジェクトの出力を構成するファイル、フォルダーおよびオペレーションを指定します。プロジェクトの出力は、以下のいずれか 1 つです (プロジェクトの種類による):

- インストール: アプリケーションのコピーをターゲット システムで作成または更新します。
- 再配布可能ファイル (マージ モジュールまたは InstallShield オブジェクト): ンストールに含められるとき、個別の機能をインストールするために必要なロジックとファイルがすべて含まれています。
- トランスフォーム: 管理者は、インストールを配布する際、変更した設定を Windows Installer インストールに適用することができます。

プロジェクトは、必要度に応じてシンプルにも複雑にもなります。シンプルなプロジェクトは、ファイル、コンポーネント、機能、レジストリ エントリのみということもあります。より複雑なプロジェクトはこれらの他さらに、再配布可能ファイル、初期化ファイルの変更を含み、外部の DLL 関数を呼び出すこともあります。

このセクションには、さらなる詳細を含む次のサブセクションが含まれています:

- インストール プロジェクトの概要
- ニーズに合ったインストール プロジェクトを選択する
- プロジェクトの種類
- プロジェクトの使用
- サンプル プロジェクト

インストール プロジェクトの概要

InstallShield では、異なるインストール プロジェクトの種類を選択できます (InstallShield の堅牢な InstallScript プログラム言語 (InstallScript) を使ったプロジェクト、Windows Installer データベース (基本の MSI) 、またはその両方 (InstallScript MSI) を使ったプロジェクト)。

インストール プロジェクトの種類の選択は、ソフトウェアのインストールのニーズによって異なります。InstallShield は、どのプロジェクトの種類にも同じ直感的なユーザー インターフェイスを提供します。したがって、アプリケーションのニーズに応じて簡単にプロジェクトの種類が選択することができ、オーサリング エクスペリエンスはまったく変わりません。さらに、1 つの種類のプロジェクト作成で身に付けた知識は、他の種類のプロジェクトにも適用できます。

プロジェクトの種類の類似事項

すべての InstallShield プロジェクト タイプでは、ほとんどのエンドユーザーによく知られている InstallShield 独特の業界標準の外観と使用感を持ったインストールを作成することができます。すべてのプロジェクトで、エンドユーザーのニーズや期待に沿ったカスタム インストールを作成することができます。

プロジェクトタイプの機能セットは似ていますが、異なるプロジェクト間には（特に InstallScript と基本の MSI プロジェクト間）では、いくつかの重要な相違点があります。ソフトウェア インストールのニーズに最も適したプロジェクトタイプを決める方法については、「ニーズに合ったインストール プロジェクトを選択する」を参照してください。

ニーズに合ったインストール プロジェクトを選択する

InstallShield を使って様々な種類のプロジェクトを作成できますが、初心者の場合、Windows Installer テクノロジまたは InstallScript テクノロジのどちらを使ってインストールをビルドするのかという決定的な選択が 1 つだけ必要です。最も適したプロジェクトの種類の選び方は、InstallShield ソフトウエアおよびインストール開発における経験度、そしてインストールとデプロイメントにおけるニーズによって異なります。以下は、3 つの主なプロパティの種類の説明です。

基本の MSI プロジェクト

基本の MSI プロジェクトは Windows Installer サービスを使ってカスタム アクション（InstallScript、VBScript、JScript、.exe ファイル、.dll ファイル、マネージ コード）の呼び出しを含んだインストール全体を制御します。基本の MSI プロジェクト タイプは、以下のいずれかを行う場合にお勧めします:

- Windows ロゴ プログラムの要件を満たしたいとき。
- Microsoft System Center Configuration Manager などの管理ツールを使って互換性を最大限にあげたいとき、または作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合。基本の MSI プロジェクト タイプでは、インストールのリパッケージする手間を省いて、インストール パッケージとその関連プロパティのトランスフォーム作成することができます。
- スクリプト コードを作成せずに、プロパティを設定してテーブル エントリを作成したいとき。
- 既存の基本の MSI プロジェクトをアップグレードしたいとき。

InstallScript プロジェクト

InstallScript プロジェクトは InstallScript を使用してインストールを操作します.このプロジェクトの種類は、以下のシナリオに推奨されます:

- エンドユーザー エクスペリエンス（エンドユーザー ダイアログ）で高度な要件がある（例、マルチメディア要素など）。
- インストールの実行時に、全画面ビルボードを使用する。
- 構造の中心で、データベース テーブルを使わずに、実行言語を使ってプロジェクトを作成したい。
- メインのインストールが実行される前後にアクションを実行したい。
- 既存の InstallScript プロジェクトをアップグレードしたい。

InstallScript MSI プロジェクト

InstallScript MSI プロジェクトは Windows Installer エンジンと InstallScript エンジンを使って、インストールの実行を進めます。インストールに、これらのエンジンが両方使用されるため、このプロジェクト タイプは最も複雑であるため、初心者ユーザーには推奨されません。このプロジェクト タイプは、次のようなシナリオで、使用できます:

- Windows ロゴ プログラムの要件を満たしたいとき。

- エンドユーザー エクスペリエンス (エンドユーザー ダイアログ) で高度な要件がある (例、マルチメディア要素など)。
- 構造の中心で、データベース テーブルを使わずに、実行言語を使ってプロジェクトを作成したい。
- メインのインストールが実行される前後にアクションを実行したい。
- 既存の InstallScript MSI プロジェクトをアップグレードしたい。

ヒント: *Repackager は AdminStudio で提供されているプロジェクト変換ツールです。このツールを使って、InstallScript プロジェクトや InstallScript MSI プロジェクトを基本の MSI プロジェクトに変換できます。*

プロジェクトの種類

InstallShield は様々なプロジェクトの種類を提供し、エンドユーザー向けの最適なプロジェクトの作成を支援します。

テーブル 3 ・ プロジェクトの種類の説明

プロジェクトの種類	アイコン	Description
アドバンスト UI		 ***エディション:*** *このプロジェクト タイプは、InstallShield の Professional Edition で使用できます。InstallShield Premier で、このタイプのプロジェクトを開くと、スイート/アドバンスト UI プロジェクトとして開きます。* アドバンスト UI プロジェクトでは、単一の .msi パッケージ、.msp パッケージ、および InstallScript パッケージだけでなく InstallShield 前提条件にも対して、最新のカスタマイズ可能なユーザー インターフェイスを持つブートストラップ アプリケーションを作成することができます。アドバンスト UI では、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。
基本の MSI		基本の MSI プロジェクトは Windows Installer を使用してインストールのユーザー インターフェイスを提供します。このプロジェクト タイプを選択した場合、機能またはコンポーネントを手動で作成し、アプリケーション ファイルおよび他の配布可能なデータをすべて指定する必要があります。

テーブル 3 • プロジェクトの種類の説明 (続き.)

プロジェクトの種類	アイコン	Description
DIM		DIM プロジェクトは、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。デベロッパー インストール マニフェスト (DIM) は、製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、およびインストール パッケージの論理的に分かれている部分を別個に構成している要素など、関連するアイテムを集めたものです。サイズは、機能とほぼ同じです。基本の MSI に DIM を複数統合することができます。
InstallScript		InstallScript インストール プロジェクト は InstallScript 言語が持つ柔軟性を提供すると共に、機能が証明されている InstallScript エンジンを利用します。
InstallScript MSI		InstallScript MSI プロジェクト は InstallScript と Windows Installer テーブルの両方を利用します。このプロジェクトの種類では、機能またはコンポーネントを手動で作成し、アプリケーション ファイルおよび他の配布可能なデータをすべて指定する必要があります。
InstallScript オブジェクト		InstallScript オブジェクト プロジェクトは、InstallScript を利用した InstallShield マージ モジュールを作成することができます。ただし、このタイプのマージ モジュールは InstallShield でしか利用できません。
マージ モジュール		マージ モジュールを作成するには、このオプションを選択します。コンポーネントやファイルの関連付けを手動で作成する必要があります。
MSI データベース		MSI データベースを選択すると、ダイレクト編集モードでインストール プロジェクトを編集することができます。つまり、InstallShield 内で Windows Installer テーブルを直接編集してインストールを構成することができるということです。
MSM データベース		MSM データベースを選択すると、ダイレクト編集モードでマージ モジュールを編集することができます。つまり、InstallShield 内で Windows Installer テーブルを直接編集してマージ モジュールを構成することができるということです。
QuickPatch		このプロジェクトの種類は、サイズが小さい単一アップグレードをエンドユーザーに配布するインストール作成者にお勧めします。QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わない別のパッチ構成方法として利用できます。QuickPatch の作成は、新規 QuickPatch 作成ウィザードを使って開始してください。

テーブル 3 • プロジェクトの種類の説明 (続き.)

プロジェクトの種類	アイコン	Description
スイート/アドバンスト UI		 ***エディション•*** *このプロジェクト タイプは、InstallShield の Premier Edition で使用できます。* スイート/アドバンスト UI プロジェクトでは、複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、Web 配布パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installer トランザクションだけでなく、複数の InstallShield 前提条件にも対して、最新でカスタマイズ可能なユーザー インターフェイスを持つブートスラップ アプリケーションを作成することができます。スイート/アドバンスト UI インストールでは、複数の個別のインストールを、統合されたユーザー インターフェイスを使用する単一のインストールにパッケージ化し、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。
MSIX		 ***エディション•*** *このプロジェクトの種類は、InstallShield Premier Edition および Professional Edition で使用できます。* ***エディション•*** *MSIX は Windows アプリ パッケージ形式で、すべての Windows アプリにモダンなパッケージ エクスペリエンスを提供します。MSIX パッケージ形式は、既存アプリ パッケージおよび/またはファイルのインストール機能をはそのままに、Win32、WPF、および WinForm アプリに新しくモダンなパッケージ作成およびデプロイメント機能を追加します。*
MSI からMSIX への変換ウィザード		MSI からMSIX への変換ウィザードを使って、基本の MSI プロジェクトを通して既存の Windows Installer ファイル (.msi ファイル)を MSIX パッケージに変換することができます。

テーブル 3 • プロジェクトの種類の説明 (続き.)

プロジェクトの種類	アイコン	Description
MSIX 変更パッケージプロジェクト		 ***エディション•*** *このプロジェクトの種類は、InstallShield Premier Edition および Professional Edition で使用できます。* MSIX 変更パッケージは、カスタマイズを格納する新しい種類のパッケージです (Windows 10、バージョン 1809 から使用可能)。MSI 変更パッケージは、インストール時ではなくプリケーションの実行時に MSIX パッケージのオーバーレイを使って適用されます。変更パッケージは、ライセンス認証ポイントがないプラグイン/アドオンの場合もあります。IT プロフェッショナルはこの機能を使用して、企業のカスタマイズによってアプリケーションがオーバーレイされるように、MSIX コンテナを柔軟に変更できます。
トランスフォーム		トランスフォーム (.mst ファイル) は、2 つの Windows Installer データベースの差分が含まれる Windows Installer の簡易データベースです。新しいトランスフォーム プロジェクトを作成すると、トランスフォーム オープン ウィザードが開始します。
Visual Basic .NET ウィザード		このオプションを選択して、Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードを起動します。
C# .NET ウィザード		このオプションを選択して、Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードを起動します。
Visual C++ .NET ウィザード		このオプションを選択して、Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードを起動します。

アドバンスト UI およびスイート/アドバンスト UI プロジェクト

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールは、インストールと InstallShield 前提条件を合わせてパッケージ化し、カスタマイズ可能な統合ユーザー インターフェイスを持つ単一インストールを作成するブートストラップ アプリケーションです。これらのインストールでは、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。

アドバンスト UI およびスイート/アドバンスト UI インストールには、Windows Vista 以降または Windows Server 2008 以降が必要です。

サイドローディング Windows アプリ パッケージ (.appx) を含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。

MSIX プロジェクト

エディション: *MSIX プロジェクト タイプは、InstallShield Premier Edition および Professional Edition で使用できます。*

MSIX は Windows アプリ パッケージ形式で、すべての Windows アプリにモダンなパッケージ エクスペリエンスを提供します。MSIX パッケージ形式は、既存アプリ パッケージおよび/またはファイルのインストール機能をはそののままに、Win32、WPF、および WinForm アプリに新しくモダンなパッケージ作成およびデプロイメント機能を追加します。

InstallShield 2019 以降では、MSIX プロジェクト タイプを使って MAIX パッケージをビルドすることができます。サイドロードまたは Windows ストア経由で配布を行う Windows アプリケーションが含まれます。

- パッケージ情報
- パッケージ ペイロード
- メディア

パッケージ情報

今回より、次を使ってプロジェクトを説明および識別することができます:

- **[一般情報]** ビューでは、プロジェクト ファイルの名前やフォーマットなどの詳細を指定します。
- **[MSIX ビジュアル資産]** エクスプローラーでは、統合的に視覚的な方法で、MSIX アプリのビジュアル的な側面を説明することができます。
- **[アプリ機能]** ビューは、特定の API または画像、音楽、またはカメラやマイクなどのデバイスといったリソースにアクセスできるよう、アプリのパッケージ マニフェスト内で有効化さている必要があります。
- **[宣言]** ビューは、アプリケーション、パッケージ レベルの宣言を作成および管理できるビジュアル ツールを提供し、それらのプロパティを構成することができます。
- **[コンテンツ URI]** ビューでは、window.external.notify を使って Script-Notify イベントをアプリに送る URI を指定します。

パッケージ ペイロード

今回より、MSIX アプリケーションをデザインするファイルを指定することができます:

- **[ファイルとフォルダー]** ビューで、InstallShield プロジェクトにファイルを追加することができます。ターゲット上にあるフォルダーへこれらのファイルを整理することができます。

重要: *MSIX プロジェクトを使って UWP アプリケーションをパッケージしている場合、同じフォルダー内にすべての資産を追加する必要があります。*

- **[アプリケーション]** ビューを使って、パッケージで提供される機能の一部またはすべてを構成する Windows アプリのプロパティを視覚的および統合的にデザインすることができます。
- **[レジストリ]** ビューでは、インストールが作成するレジストリキーおよびレジストリ値を定義することができます。

メディア

今回より、インストールを配布するのに利用するファイルとフォルダーをカスタマイズできます:

- **[パス変数]** ビューでは、変数を使って開発システム間で簡単にインストールを移動させることができます。
- **[リリース]** ビューには、プロジェクトに含まれる各製品のビジュアルツールがあります。

基本の MSI インストール プロジェクト

基本の MSI は、InstallShield における Windows Installer ベースのプロジェクト タイプです。基本の MSI プロジェクトは、Windows Installer サービスを使ってインストール全体を作成する場合にお勧めです。これを使うと、Windows Installer 機能セットだけを使ってインストールをオーサリングすることができます。MSI パッケージでは、セットアップのユーザー インターフェイス (UI) のフローだけでなく、エンドユーザー ダイアログの寸法も、直接オーサリングすることができます。 Windows Installer サービスはユーザー インターフェイスのレンダリング機能を使用して、エンドユーザーに UI を表示します。このプロジェクトの種類は、インストールをオープン仕様で作成する必要がある場合に、最も適します。

メモ: *作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合、基本の MSI プロジェクトが推奨されます。これはインストールのリパッケージを行なわずにインストール パッケージやその関連プロパティのトランスフォーム作成を可能にする柔軟性を提供します。*

基本の MSI プロジェクトでは、InstallScript コードをカスタム アクションという形で実行することができます。InstallScript プロジェクトと同様、基本の MSI プロジェクトは、ダイアログのオーサリング、標準 Windows .dll ファイルでのカスタム アクションの呼び出し機能、サポート ファイルの指定機能などの InstallShield により提供される他の堅牢な機能を使用します。

InstallScript カスタム アクションを使った、基本の MSI インストールのランタイム動作

InstallShield 12 以降のプロジェクト

次のステップは、InstallScript カスタム アクションを使った基本の MSI インストールのランタイム動作の概要を説明します。

1. Windows Installer サービスによってパッケージが起動されます。
2. シーケンスされた InstallScript カスタム アクションが次のように実行する。
    a. Windows Installer が Binary テーブルからロードされた ISSetup.dll の関連エントリ ポイントを呼び出す。

b. ISSetup.dll が Setup.inx のリソースを一時保管場所へ抽出する。

c. ISSetup.dll が関連スクリプト コードを実行する。

d. Windows Installer が ISSetup.dll をアンロードする。

InstallScript カスタム アクションがある InstallShield 12 以降の基本の MSI のランタイム動作は、InstallShield 11.5 以前での動作とは大きく異なります。これは強化されたアーキテクチャによるものです。強化内容についての詳細は、「InstallShield 11.5 以前のプロジェクトをアップグレードする」を参照してください。

InstallShield 11.5 および以前のプロジェクト

InstallShield 11.5 および以前のバージョンで作成されたインストールのランタイム動作は次の通りです。

1. Setup.exe が ISScript.msi をインストールする。
2. Setup.exe が MsiExec.exe を起動して、プライマリ .msi ファイルをインストールする。Setup.exe がインストールに含まれているか、またはターゲットシステムに既存する必要があります。)
3. ISMsiServerStartup (始めの カスタム アクション) が開始し、IDriver.exe が起動する。
4. シーケンスされた InstallScript カスタム アクションが次のように実行する。
   a. Windows Installer が ISScriptBridge.dll で関連エントリ ポイントを探す。
   b. ISScriptBridge.dll が ROT (実行中オブジェクト テーブル) を使って、実行中の IDriver.exe インスタンスを検出する。
   c. IDriver.exe が関連スクリプト コードを実行する。
5. ISCleanUpSuccess が IDriver.exe をシャット ダウンする。

InstallShield 11.5 以前のプロジェクトを InstallShield 2025 へアップグレードした場合、ランタイム動作は InstallShield 12 以降のプロジェクトで説明されている動作にアップデートされます。

DIM プロジェクト

InstallShield の DIM プロジェクトは、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。デベロッパー インストール マニフェスト (DIM) は、製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、およびインストール パッケージの論理的に分かれている部分を別個に構成している要素など、関連するアイテムを集めたものです。サイズは、機能とほぼ同じです。以下は、DIM を利用した時のいくつかの利点です:

- DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。
- リリース エンジニアは、DIM を複数のインストール プロジェクトで繰り返し使用できるため、非常に効率的です。
- DIM では、基本の MSI プロジェクトで提供されている機能とほぼ同じ機能がサポートされています。このため、DIM の作成者には、インストールの部分開発を行うために必要な柔軟性がすべて提供されています。

InstallScript インストール プロジェクト

InstallScript インストール プロジェクトの種類は、広く親しまれている InstallShield Professional プロジェクトと共に、InstallScript プログラム言語のパワーと柔軟性を提供します。Windows Installer は一切使用されません。インストールは、完全にスクリプトによって制御され、非常に柔軟性があります。InstallShield Professional で使用されている信頼性の高い InstallScript エンジンが使用されています。

メモ: *InstallScript インストールは一般的にシステム管理者からは好まれません。このタイプは .msi ファイルの形式ですべてがまとまっていないので Setup.exe を利用しなくてはならず、また配布に先立って完全にカスタマイズすることが不可能な場合もあるからです。作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合、基本の MSI インストール プロジェクトが推奨されます。*

InstallScript エンジン

InstallScript プロジェクトでは、InstallScript によってランタイムユーザー インターフェイスの表示やフロー制御が行われ、ターゲットオペレーティング システムへの変更は Windows Installer に全く頼ることなく、信頼性の高い InstallShield の InstallScript エンジンによって行われます。

InstallScript をインストールのドライバーとして使用することには多くの利点があります。最初の利点は、InstallScript のイベントモデルによりコードを 1 行も書かずにスクリプト型のインストールを作成できる点です。カスタムの機能性を追加したい場合は、機能性を変更したいイベントを実装するだけで済みます。

基本の MSI プロジェクトのユーザー インターフェイス機能には、使用できるコントロールの種類やダイアログイベントに行うコントロールの種類に関していくらかの制限があります。InstallScript MSI プロジェクトにはそのような制限はなく、カスタム ダイアログコントロールを追加する機能とともに幅広い標準コントロールを提供します。スクリプト内でダイアログメッセージを使用することで、エンドユーザーダイアログの動作を完全にコントロールすることができます。

InstallShield Professional を使用して作成したプロジェクトのアップグレード

InstallShield Professional (バージョン 5.5 以降) を使用して作成されたインストール プロジェクトは、InstallScript プロジェクトの種類にアップグレードされます。

InstallScript プロジェクトの追加機能

InstallScript プロジェクトは、基本の MSI プロジェクトでは利用できない追加機能を提供します。

テーブル 4 · InstallScript プロジェクトの追加機能

機能	説明
[セットアップの種類] ビュー	このビューでアプリケーションの異なるインストールの構成を簡単に作成できます。また、このビューでは、[カスタム] セットアップ タイプのデフォルトを選択することもできます。
ビルボード サポート	ランタイム ビルボード サポートは、InstallScript プロジェクトでのみ利用可能です。ビルボードにより、マシンへのファイル転送中にエンドユーザーにビットマップを表示することができます。これらのビットマップを使用して、エンターテイメントや情報を提供することができます。

InstallScript MSI インストール プロジェクト

InstallScript MSI インストールは 2 つの異なるインストーラー エンジンを使用します:

- Windows Instaler エンジンは、.msi パッケージの標準 [実行] シーケンスを実行します。これは、通常ターゲット システムを変更するシーケンスです。
- InstallScript エンジンは、インストールのカスタム ユーザー インターフェイス (UI) ハンドラーとしての役割を果します。また、InstallScript コードを実行します。UI に InstallScript エンジンを使用する際の利点は、高度にカスタマイズされたランタイム ダイアログのサポートが提供されていることです。

さらに、InstallShield では、InstallScript MSI インストールのUI に、次の 2 つの異なるスタイルが追加されました:

- **従来型スタイル** – InstallScript MSI インストールで、InstallScript エンジンを外部 UI ハンドラーとして使用することができます。このスタイルを使用する場合は、インストールに Setup.exe セットアップランチャーを必ず含めてください。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。
- **新しいスタイル** – InstallScript MSI インストールで、InstallScript エンジンを埋め込み UI ハンドラーとして使用することができます。このスタイルでは、InstallScript エンジンが .msi パッケージ内に埋め込まれます。Windows Installer は InstallScript エンジンを呼び出して UI を表示します。Windows Instaler はまた、.msi パッケージの [実行] シーケンスを実行します。

  このオプションには、ターゲット マシン上に Windows Installer 4.5 が必要です。また、いくつかの制限事項があるため、このスタイルを使用する場合は綿密な計画が必要です。

2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

メモ: このプロジェクトの種類は異なるエンジンを使用するため、単なる InstallScript や 基本の MSI インストール プロジェクトより複雑になっています.これは高度な知識をもつユーザーのみに推奨されます。

システム管理者には、従来型スタイルの InstallScript MSI インストールよりも、基本の MSI インストールまたは新しいスタイルの InstallScript MSI インストールが好まれます。これは、従来型スタイルの InstallScript MSI インストールが、.msi パッケージのように自己完結形式ではないため、Setup.exe を使用する必要があることと、展開の前に完全なカスタマイズが不可能な場合があるためです。作成中のソフトウェアが、配布される前に企業のシステム管理者によってカスタマイズされる場合、基本の MSI プロジェクトが推奨されます。代わりの方法として、新しいスタイルの InstallScript MSI インストールを使用することもできますが、必要なすべてのシステム変更は、[インストール]–[実行] シーケンスでカスタム アクションを使用して行う必要があることに注意してください。権限が不足しないように、カスタム アクションは、システム コンテキストで "スクリプト内実行" 設定が遅延実行になっている必要があります。

InstallScript MSI プロジェクトのその他の機能

InstallScript MSI プロジェクトでは、基本の MSI プロジェクトで利用できない追加機能がいくつか提供されます:

テーブル 5 ・ InstallScript MSI プロジェクトの追加機能

機能	説明
[セットアップの種類] ビュー	このビューでは、アプリケーションに、[標準]、[最小] など、異なる定義済みのインストール構成を簡単に作成することができます。また、このビューでは、[カスタム] セットアップ タイプのデフォルトを選択することもできます。
ビルボード サポート	実行時のビルボードは、InstallScript MSI インストールでのみサポートされています。基本の MSI インストールではサポートされていません。ビルボードを利用すると、エンド ユーザーのマシンへファイルを転送中に、ビットマップを表示することができます。これらのビットマップを使用して、エンドユーザーにエンターテイメントや役に立つ情報を提供することができます。

InstallScript MSI インストールのランタイム動作

InstallShield 12 以降のプロジェクト

次のステップは、InstallScript MSI インストールのランタイム動作についての概要を説明します。

1. Setup.exe が ISSetup.dll を起動する。
2. ISSetup.dll が行う処理は次の通りです。
    a. Setup.inx をロードする。
    b. プリ インストール InstallScript イベント (例、OnBegin) を実行する。
    c. Windows Installer (外部ユーザー インターフェイスの場合、InstallScript エンジン) を起動する。
    d. 各 InstallShield カスタム アクションが次のように実行する (例、OnMoved)。(これは、InstallScript カスタム アクションを使った基本の MSI で使われるメカニズムと基本的に同じです。)
        i. Windows Installer が ISSetup.dll で適切なエントリ ポイント呼び出す。(これは、ISSetup.dll の異なるインスタンスで、Binary テーブルからロードされます。)
        ii. ISSetup.dll が Setup.inx のリソースを一時保管場所へ抽出する。
        iii. ISSetup.dll が関連スクリプト コードを実行する。
        iv. ポスト インストール InstallScript イベント (例、OnEnd) を実行する。

InstallShield 12 以降の InstallScript MSI プロジェクトのランタイム動作は、InstallShield 11.5 以前での動作とは大きく異なります。これは強化されたアーキテクチャによるものです。強化内容についての詳細は、「InstallShield 11.5 以前のプロジェクトをアップグレードする」を参照してください。

InstallShield 11.5 および以前のプロジェクト

InstallShield 11.5 および以前のバージョンで作成された InstallScript MSI インストールのランタイム動作は次の通りです。

1. Setup.exe が ISScript.msi をインストールする。
2. Setup.exe が IDriver.exe を起動する。
3. IDriver.exe が行う処理は次の通りです。
    a. Setup.inx をロードする。
    b. プリ インストール InstallScript イベント (例、OnBegin) を実行する。
    c. Windows Installer (外部ユーザー インターフェイスの場合、InstallScript エンジン) を起動する。

       各 InstallScript カスタム アクションが次のように実行する (例えば、OnMoved)。(これは、カスタム アクションを使う基本の MSI インストールで使われるメカニズムと基本的に同じです。)

        i. Windows Installer が ISScriptBridge.dll で適切なエントリ ポイントを呼び出す。
        ii. ISScriptBridge.dll が ROT (実行中オブジェクト テーブル) を使って、実行中の IDriver.exe インスタンスを検出する。
        iii. IDriver.exe が関連スクリプト コードを実行する。
    d. ポスト インストール InstallScript イベント (例、OnEnd) を実行する。

マージ モジュール プロジェクト

マージ モジュールにより、既存の各機能をインストールに追加できます。たとえば、アプリケーションで VB Runtime DLL が必要な場合、Microsoft's Visual Basic Virtual Machine モジュールをセットアップ アプリケーションに追加することができます。従来は、レジストリ エントリを作成し、ターゲットフォルダー、カスタム アクション、および任意の他の必要なステップを設定しなければならなりませんでした。マージ モジュールでは、既存のマージ モジュールをインストールに関連付けて、次の作業に移動するだけでこれらのステップを完了できます。

InstallShield を利用して、専用のマージ モジュールを作成し、インストールの配布または他の開発者への供与ができるようになります。これは、すべてのインストール プロジェクトに機能を分散する場合に役立ちます。その部分のインストールを毎回作り直すのではなく、マージ モジュールとして 1 度作成し、それをすべてのインストール プロジェクトに関連付けます。

マージ モジュール プロジェクトに含まれる InstallScript カスタム アクション

マージ モジュールに InstallScript カスタム アクションを含めることができます。ただし、カスタム アクションが適切にマージされるよう、Module**Sequence テーブルエントリを手作業で作成しなくてはなりません。

マージ モジュールの動作のしくみ

マージ モジュールをインストール プロジェクトに関連付けると、ビルド時に .msi データベースにマージされます。その結果、2 つのプロジェクト (新規インストールと既存のマージ モジュール) が MSI データベースにマージされることになります。マージ モジュールはコンポーネントのように動作します。つまり、関連付けされている機能がインストールされなければ、マージ モジュールもインストールされません。したがって、VB Runtime DLL を必要とする機能があるのに VB Runtime DLL がインストール対象になっていない場合、VB マージ モジュールはインストールされません。

MSI データベースおよび MSM データベース プロジェクト

MSI データベースおよび MSM データベース プロジェクトを使用すると、中間プロジェクト フォーマット (.ism ファイル) で作業せずに、Windows Installer インストール パッケージおよびマージ モジュール データベースを編集することができます。これらのプロジェクトタイプは、ダイレクト エディター機能を拡張して、標準インストール プロジェクトとマージ モジュール プロジェクトでサポートされている異なるビューを含めます。これらのプロジェクトの種類でサポートされているビューは、 .ism プロジェクトで機能するのと同じように機能するように設計されています。

プロジェクト: *MSI データベースおよび MSM データベース プロジェクトでは、文字列エントリもパス変数もサポートされていません。*

既存の .msi ファイルまたは .msm ファイルを直接編集したり、新規の MSI データベースまたは MSM データベースを開いて新規のデータベースを直接作成することもできます。このファイルの中のデータは、ブランクの基本の MSI、またはマージ モジュール プロジェクトを作成してそのプロジェクトからデータベースをビルドしたときに取得するものと合致します。

InstallScript オブジェクト プロジェクト

InstallShield オブジェクトを使用して、 Windows テクノロジを簡単にインストールできます.テクノロジをインストールするには、インストールに対応する InstallShield オブジェクトを含めるだけです。特定のコンポーネントが必要な場合は、機能に対応する InstallShield オブジェクトを関連付けます。

QuickPatch プロジェクト

QuickPatch プロジェクトは、規模の小さいシングル アップグレードをユーザーへ配布したいインストール作成者へお勧めするプロジェクトの種類です。カスタム アクションの追加、.ini データの変更などのより広範囲におよび変更には通常、標準パッチが必要です。

QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ構成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

QuickPatch では、次のすべてを実行することができます。

- 元のインストールまたは以前の QuickPatch へ新しいファイルを追加する。
- 元のインストールのファイルを削除する。
- 以前の QuickPatch と共に追加されたファイルを削除する。
- 上記と同じ操作をレジストリ エントリで実行する。
- 元のインストールに含まれていたが、現在の QuickPatch プロジェクトには適用しないカスタム アクションを削除する。

QuickPatch プロジェクトの作成は、新規 QuickPatch 作成ウィザードで始めます。ウィザードを完了した後、InstallShield でプロジェクトが開いてから QuickPatch プロジェクトの設定を構成できます。

トランスフォーム プロジェクト

トランスフォーム (.mst ファイル) は、2 つの .msi データベース間の差分を含むシンプルな Windows Installer データベースです。トランスフォームを使うと、管理者は、インストール パッケージを配布する際、変更した設定をデータベースに適用することができます。

InstallShield は、.msi パッケージを、InstallShield (.ism) プロジェクトに変換することなく編集することができるトランスフォーム プロジェクト タイプを備えています。変更点はトランスフォーム (.mst) ファイルとして保存できます。

トランスフォームプロジェクトを作成、または InstallShield でトランスフォームを開くと、トランスフォーム オープン ウィザードが起動し、ベース .msi ファイルやベース .msi ファイルに適用するその他のトランスフォームファイルについての情報を収集することができます。

プロジェクトの使用

InstallShield では、インストール プロジェクトから InstallShield 内で機能性を再使用することができる InstallShield オブジェクト プロジェクトまで、様々な種類のプロジェクトを作成、編集、アップグレードおよび保存することができます。

このセクション内のページでは、特定のプロジェクトの種類およびプロジェクトテンプレートの作成の仕方、また、異なる InstallShield 製品からアップグレードするとき考慮に入れる点等、いろいろなトピックがカバーされています。

新しいプロジェクトの作成

InstallShield プロジェクトを新規に作成するとき、その方法はいくつかあります。

タスク

新しいプロジェクトを作成するには、以下のいずれかを行います。

- ツールバーまたは InstallShield スタート ページ 上で **[新規プロジェクト]** ボタンをクリックします。
- Ctrl+N を押します。
- **[ファイル]** メニューで、**[新規]** をクリックします。

これらの手順を行うと、[新規プロジェクト] ダイアログ ボックスが起動されるので、そこで作成するプロジェクトを選択できます。

テンプレートを使って新規プロジェクトを作成する

プロジェクト テンプレートは、該当するタブの [新規プロジェクト] ダイアログ ボックスに表示されます。詳細については、「テンプレートを使用した新規プロジェクトの作成」を参照してください。

Microsoft Visual Studio 内からプロジェクトを作成する

Microsoft Visual Studio .NET ワークスペース内から InstallShield プロジェクトを作成することができます。詳細については、「Microsoft Visual Studio で InstallShield プロジェクトを作成する」を参照してください。

プロジェクトを開く

タスク ***既存の InstallShield プロジェクトを開くには、次のいずれかの操作を実行します。***

- ツールバーにある **[プロジェクトを開く]** ボタンをクリックする。
- **[ファイル]** メニューで、**[開く]** をクリックします。[**開く**] ダイアログ ボックスで、プロジェクトファイルに移動します。
- Ctrl+O を押します。
- **[既存のプロジェクトを開く]** リンクをクリックするか、InstallShield スタート ページで前回開いたプロジェクトのリンクをクリックします。
- デスクトップ上または Windows エクスプローラー内でプロジェクト ファイル (.ism) をダブルクリックする。

特定のファイルまたはファイル リンクをクリックした場合を除き、上のすべてのオプションで [開く] ダイアログ ボックスが開くので、ここでプロジェクト ファイルを参照する。

ソース コード管理からプロジェクトを開く

[開く] ダイアログのオプションを使うと、ソース コントロール アプリケーションから最新バージョンのプロジェクト フォルダーを使用できるようになります。

タスク ***ソース管理からプロジェクトを開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. **[ソース管理]** ボタンをクリックします。ソース管理プログラムのログイン ダイアログが表示されます。
3. ソース管理プログラムにログインし、プロジェクト フォルダーまたはデータベースを参照します。

InstallShield では、プロジェクトのすべてのファイルについて最新のバージョンが作業ディレクトリに取得されます。

メモ: *InstallShield があるシステムにソース管理アプリケーションがない場合、[開く] ダイアログ ボックスは、[ソース管理] ボタンを表示しません。*

ダイレクト編集モードでパッチ作成プロパティ ファイルを開く

[開く] ダイアログ ボックスから、以前に作成したパッチ作成プロパティ ファイルを開くことができます。

タスク ***パッチ作成プロパティ (.pcp ファイル) を開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. **[ファイルの種類]** 一覧で、**[パッチ作成プロパティ ファイル (*.pcp)]** をクリックします。
3. 編集する .pcp ファイルを選択して、**[開く]** をクリックします。

パッチプロジェクトファイルがダイレクトエディターで開きます。

Windows Installer パッケージを開く

InstallShield には、IDE から直接、ダイレクト編集モードを使って MSI パッケージを編集するか、InstallShield プロジェクト (.ism) に変換してから IDE で基本の MSI プロジェクトとして編集するオプションがあります。

メモ: *MSI パッケージをダイレクト編集モードで編集する場合、制限される機能もあります。*

タスク ***MSI パッケージを IDE の直接編集モードで開くには、以下の手順を実行します。***

1. ツールバーにある **[プロジェクトを開く]** ボタンをクリックする。参照ダイアログが表示されます。
2. **[ファイルの種類]** リストから **[Windows Installer パッケージ (*.msi)]** を選択します。
3. 開く MSI パッケージを選択します。
4. 必ず **"形式を選択して開く"** フィールドを [自動] に設定してから **[開く]** をクリックしてください。

メモ: *デフォルトでは、IDE は直接編集モードで MSI パッケージを開きます。*

MSI/MSM オープン ウィザードを使用すると、既存の Windows Installer (.msi) パッケージを InstallShield プロジェクト (.ism) ファイルに変換し、新規プロジェクトを開いて IDE で変更できます。

メモ: *InstallShield で、パッチ作成プロジェクト ファイル (.pcp ファイル) を開いて、ダイレクトエディターで編集することもできます。*

マージ モジュールを開く

MSI/MSM オープン ウィザード を使用すると、既存の Windows Installer のマージ モジュール (.msm ファイル) を InstallShield プロジェクト (.ism) ファイルに変換し、InstallShield IDE で新規プロジェクトを開くことができます。

タスク ***MSI/MSM オープンウィザードを起動するには、次の操作を実行します。***

1. ツールバーにある **[プロジェクトを開く]** ボタンをクリックする。参照ダイアログが表示されます。
2. **[ファイルの種類]** リストから **[Windows Installer モジュール (*.msm)]** を選択します。
3. マージ モジュールまで移動して [**開く**] をクリックします。

作成するプロジェクトの詳細情報を指定するよう、MSI/MSM オープンウィザードにメッセージが表示され、新規マージ モジュールプロジェクトが IDE で開かれます。

オブジェクト プロジェクトを開く

InstallShield Professional を使って作成されたオブジェクトプロジェクト (.ipo または .ipr) を開くことができます。InstallShield でこのプロジェクトを開くと、プロジェクトはマージ モジュール (.msm) プロジェクトに移行されます。

[再配布可能ファイル] ビューにマージ モジュールを配置する

マージ モジュールを InstallShield インストール プロジェクトを関連付けるとき、セットアッププロジェクトの [マージ モジュール] ビュー内で使用することができるように、マージ モジュールプロジェクトを Modules フォルダーに自動的に配置することができます。

タスク ***マージ モジュール プロジェクトを Modules フォルダーに自動的に配置するには、以下の手順に従います:***

リリース ウィザードの [マージ モジュールのオプション] パネルで "ローカル マージ モジュール カタログに追加" オプションを選択します。

プロジェクトの保存

新しいプロジェクトを作成すると、プロジェクトは自動的に [新規プロジェクト] ダイアログ ボックス で指定した名前と場所を使って保存されます。

InstallShield では、開いたプロジェクトのコピーをテンプレートとして、または新しいプロジェクトとして別の名前をつけて別の場所に保存することができます。

新しい名前と場所でプロジェクトを保存する

新しい名前をつけて新しい場所にプロジェクト保存したとき、名前が変更されたプロジェクトファイルおよびその関連ファイル並びにフォルダーのコピーが新しい場所に保存されます。次回プロジェクトを保存する際、変更はすべてこの新しいフォルダーに保存されます。

タスク ***新しい名前を付けて、新しい場所にプロジェクトを保存するには、次の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. **[名前を付て保存]** ボックスで、適切な場所を選択します。
3. 必要に応じて **[*"プロジェクト名" サブフォルダーを作成し、作成したフォルダーにプロジェクトを保存する*]** チェック ボックスを選択またはクリアします。どちらの場合も、*<プロジェクト名>* サブフォルダーは、**[プロジェクトの場所]** ボックス (**[オプション]** ダイアログ ボックスの **[ファイルの場所]** 上) で作成されます。

   必要に応じて **[*'プロジェクト名' サブフォルダーを作成してプロジェクトを保存する*]** チェック ボックスを選択すると、プロジェクト ファイル (.ism) は、*<プロジェクト名>* サブフォルダーに保存されます。チェック ボックスをクリアすると、プロジェクト ファイルは **[プロジェクトの場所]**ボックスで指定した場所に保存されます。

4. 新しいプロジェクトの GUID を、オリジナルプロジェクトとは異なる GUID にするには、[**保存したプロジェクトに新しいプロジェクト GUID を作成して割り当てる**] を選択します。新しいプロジェクトの GUID をオリジナルプロジェクトと同じ GUID にするには、このチェック ボックスをクリアします。
5. ([**一般情報**] ビューにある) **"製品名"** 設定の値として、**[ファイル名]** ボックスで指定した名前を使用するには、**[新しいプロジェクト名に基づいて適切にプロジェクト設定をアップデートする]** チェック ボックスを選択します。新しいプロジェクトの **"名前"** プロパティ を元のプロジェクトと同じにするには、このチェック ボックスをクリアします。

メモ: *InstallScript ファイルなど、外部依存関係と共にプロジェクトを保存すると、新しいプロジェクトはこれらのファイルの元のコピーをポイントします。コピーは複製されません。したがって、オリジナルのプロジェクトを削除する場合は、新しいプロジェクトで使用する可能性のあるファイルを削除しないように注意してください。*

プロジェクトを別の種類のプロジェクトに変換する

InstallShield では、一部のプロジェクトを別の種類のプロジェクトに変換することができます。

テーブル 6 ・ プロジェクト コンバーター

プロジェクト コンバーター	手順
基本 MSI プロジェクトを InstallScript MSI プロジェクトに変換	「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。
InstallScript MSI プロジェクトを InstallScript プロジェクトに変換	「InstallScript MSI プロジェクトから InstallScript プロジェクトへ変換する」を参照してください。
InstallScript プロジェクトを基本の MSI プロジェクトに変換	Repackager を使って、InstallScript プロジェクトを基本の MSI プロジェクトに変換します。 ***エディション:*** *Repackager は AdminStudio に含まれています。*
InstallScript MSI プロジェクトを基本の MSI プロジェクトに変換	Repackager を使って、InstallScript MSI プロジェクトを基本の MSI プロジェクトに変換します。 ***エディション:*** *Repackager は AdminStudio に含まれています。*

テーブル 6 • プロジェクト コンバーター

プロジェクト コンバーター	手順
InstallScript プロジェクトを InstallScript MSI プロジェクトに変換	変換は、次の 2 つのステップを実行します: 1. Repackager を使って、InstallScript プロジェクトを基本の MSI プロジェクトに変換します。 2. InstallShield を使って、基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換します。「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。 ***エディション:*** *Repackager は AdminStudio に含まれています。*
Visual Studio セットアップ プロジェクトを InstallShield の基本の MSI プロジェクトに変換	「Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする」を参照してください。
Visual Studio マージ モジュール プロジェクトを InstallShield のマージ モジュール プロジェクトに変換	「Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする」を参照してください。

基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する

基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換すると、InstallShield のスクリプト作成機能を最大限に活用できます。

注意 *• 一度変換したプロジェクトを元に戻すことはできません。InstallScript MSI プロジェクトに変換された基本の MSI プロジェクトは、一度保存されると、容易に元の基本の MSI プロジェクトに戻すことはできなくなります。基本の MSI プロジェクトを変換する前に、プロジェクトのバックアップ コピーを作成することが一般的に推奨されます。*

プロジェクトの変換

タスク

基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換するには、以下の手順を実行します。

1. 基本の MSI プロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[プロジェクト コンバーター]** をポイントして、**[InstallScript MSI プロジェクトへ変換]** をクリックします。

ダイアログの変換

InstallScript MSI プロジェクトは、Windows Installer ダイアログ関連のテーブルではなく、リソースベースのダイアログ を使用するため、元のプロジェクトで使っていた Windows Installer ベースのダイアログは、プロジェクトの移行後は利用不可能になります。このような場合、カスタム ダイアログは手作業で追加してください。

変換後のインストール プロジェクトには、デフォルトの InstallScript MSI ダイアログが用意されています。このダイアログ ボックスを確かめるには、InstallScript ビューに移動して、OnFirstUIBefore か OnFirstUIAfter イベントで表示されるダイアログを確認してください。

InstallScript MSI プロジェクトから InstallScript プロジェクトへ変換する

InstallShield、InstallShield DevStudio、または InstallShield Developer を使用して作成された InstallScript MSI プロジェクトも、簡単に InstallScript プロジェクトに変換することができます。

タスク ***既存の InstallScript MSI プロジェクトを InstallScriptプロジェクトに変換するには、以下の手順を実行します。***

1. InstallScript MSI プロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[プロジェクト コンバーター]** をポイントして、**[InstallScript プロジェクトへ変換]** をクリックします。

GUID

GUID は、グローバル固有 ID (*Globally Unique Identifier*) の略です。GUID は 128 ビット長で、各 GUID は GUID 生成に使用されるアルゴリズムによって一意になるように生成されます。GUID は固有でなければならないため、COM クラスや製品コード、およびその他のさまざまなコードを識別するのに使用することができます。

たとえば製品をインストールした後で、HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall の下にキーを作成し、インストールの製品コードに従って命名することができます。このキーは製品名に従って命名されていたことがありました。しかしこのために競合が起きる可能性がありました。2 つのインストールが同じマシンにインストールされて、両方が同じ製品名を共有する場合、同じレジストリキーが共有されることになったからです。GUID が使用されるようになったため、このような競合が起きることがなくなりました。

GUID は {5D607F6A-AF48-4003-AFA8-69E019A4496F} のような形式です。GUID の文字はすべて大文字でなければなりません。

プロジェクトの GUID

インストール プロジェクトを作成すると、プロジェクトに関連した多くの異なる GUID ができます。

テーブル 7 ・ GUID

GUID 名	説明
製品コード	製品 GUID は、アプリケーションを一意に識別します。
パッケージ コード	パッケージ コードはインストール パッケージを一意に識別します。 ***プロジェクト:*** *この GUID は、基本の MSI、InstallScript MSI、および QuickPatch プロジェクト タイプで使用できます。*

テーブル 7 • GUID (続き.)

GUID 名	説明
パッケージ GUID	パッケージ GUID は、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるインストール パッケージを一意に識別します。 ***プロジェクト:*** *この GUID は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます。*
パッチ GUID	パッチ GUID は、パッチ パッケージを一意に識別します。 ***プロジェクト:*** *この GUID は、基本の MSI、InstallScript MSI、および QuickPatch プロジェクト タイプで使用できます。*
スイート GUID	スイート GUID は、アドバンスト UI またはスイート/アドバンスト UI インストールを一意に識別します。 ***プロジェクト:*** *この GUID は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます。*
アップグレード コードとアップグレード GUID	アップグレード GUID はアップグレードの目的で製品のファミリーを識別します。これはメジャーアップグレードに重要です。 ***プロジェクト:*** *この GUID は、基本の MSI、InstallScript MSI、および QuickPatch プロジェクト タイプで使用できます。*

プロジェクト: *基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで GUID を変更する必要がある場合についての情報は、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。*

デフォルトのプロジェクトの場所を変更する

すべての新しいプロジェクトは次のデフォルトの保存場所に保存されます:

C:¥InstallShield 2022 Projects

インストール プロジェクトに対して新規デフォルトの場所を指定するには、**[オプション]** ダイアログ ボックスの **[ファイルの場所]** タブを利用します。

メモ: このフォルダーは、すべての新規インストール プロジェクトに対して使用されますが、既存のプロジェクトは以前のフォルダーにそのまま保存されます。

プロジェクト要素を再利用する

プロジェクトの要素をいくつか保存して別のプロジェクトで再利用することができます。データによっては中間ファイルに保存しなければならないものもあれば、別の .ism ファイルに直接エクスポートできるものもあります。各要素の使用方法を以下に説明します。

プロジェクト全体

プロジェクトを InstallShield プロジェクトのテンプレートとして保存すると、元のプロジェクトとほぼ同じプロジェクトを、そのテンプレートから作成することができます。

レジストリ エントリ

REG ファイルにレジストリ エントリをエクスポートすると、同じプロジェクトまたは全く別のプロジェクトの別のコンポーネントのレジストリデータに、このエクスポートしたエントリをインポートすることができます。

文字列のエントリ

プロジェクトの特定言語の文字列エントリを、タブ区切りのテキスト ファイルにエクスポートすることができます。一般的には、テキスト ファイルを翻訳に出して、その文字列エントリを別の言語用にプロジェクトにインポートしますが、テキスト ファイルを別のプロジェクトにインポートする方法で、エクスポートされた文字列エントリを別のプロジェクトに追加することもできます。

エンドユーザー ダイアログ

すべてのダイアログを .rc ファイルに保存し、それぞれを個別に InstallShield ダイアログ テンプレート ファイルにエクスポートするか、または別のプロジェクトに直接エクスポートすることができます。

コンポーネント

コンポーネントを別のプロジェクトにエクスポートするには、コンポーネントを右クリックして [コンポーネントのエクスポート ウィザード] を選択します。ウィザードの指示に従うと、データのすべてが指定のプロジェクトに追加されます。

カスタム アクション

[カスタム アクション] エクスプローラーでカスタム アクションを右クリックして、[エクスポート] をクリックすることにより、カスタム アクションを別のプロジェクトにエクスポートすることができます。[カスタム アクション] エクスプローラーは、[カスタム アクションとシーケンス] ビュー (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) および [カスタム アクション] ビュー (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) にあります。

リポジトリを使用してプロジェクトの要素を共有する

リポジトリの使用は、InstallShield でサポートされています。リポジトリは、異なるインストール プロジェクトで共有および再利用することができる共通要素のコレクションです。以下は、リポジトリに格納できる要素です。

- エンドユーザー ダイアログ
- InstallScript ファイル
- マージ モジュール
- プロジェクト テンプレート
- SQL スクリプト
- システム検索

リポジトリには、複数のプロジェクトでプロジェクト要素を再利用することができる機能が用意されており、整合性を保つのに役立ちます。また、前に行った作業を複製する手間も省くことができます。たとえば、ある組織で、多数の製品インストールが特定のカスタム ダイアログを含むとき、そのカスタム ダイアログを一度だけ作成し、それをリポジトリにパブリッシュするだけですみます。他のインストールでそのダイアログを使用したいとき、そのダイアログをリポジトリからプロジェクトに追加するだけです。

InstallShield で利用できるリポジトリには 2 つのタイプがあります。

- ネットワーク リポジトリ – ローカル リポジトリは、複数のプロジェクトで再利用することができる、自分専用のインストール要素のコレクションです。ローカル リポジトリは、ローカルマシンに格納され、他のインストール作成者は利用することが *できません*。
- ネットワーク リポジトリ – ネットワーク リポジトリは、複数のインストール作成者が必要に応じてアクセスでき、プロジェクトで再利用することができるインストール要素のコレクションです。ネットワーク リポジトリは、インストール作成者同士のコラボレーション作業を促進し、ネットワークに格納されます。ネットワーク リポジトリの構成するための手順と注意事項は、「ネットワーク リポジトリを設定する」を参照してください。

エディション: *ネットワーク リポジトリは、InstallShield の Premier Edition でのみ利用することができます。ローカル レポジトリは InstallShield Premier Edition および Professional Edition の両方で使用できます。*

ネットワーク リポジトリを設定する

エディション: *ネットワーク リポジトリは、InstallShield の Premier Edition でのみ利用することができます。*

ネットワークリポジトリを構成する、またはネットワークリポジトリの設定を変更するには、[オプション] ダイアログ ボックスにある [リポジトリ] タブを利用します。

プロジェクト テンプレート

プロジェクト テンプレートは、インストール プロジェクトやマージ モジュールの作成を開始するときに使用する、すべてのデフォルトの設定とデザイン要素を含みます。

InstallShield でテンプレートを開き、通常のプロジェクトを編集するのと同様に編集してください。テンプレートを作成するには、任意のプロジェクトをテンプレートとして保存します。詳細については、「プロジェクト テンプレートの作成」を参照してください。

テンプレートを基に新しいプロジェクトを作成する方法については、「テンプレートを使用した新規プロジェクトの作成」をご覧ください。

InstallShield プロジェクトテンプレートは、新規プロジェクトの開始点としてのみ利用することができます。テンプレートを使ってプロジェクトを作成しても、現在のプロジェクトと既存のテンプレートの間に何ら関連はありません。つまり、テンプレートの要素を変更しても、このテンプレートを使って作成したプロジェクトは影響を受けません。ただし、テンプレートを変更し、テンプレートの更新バージョンを使って別のプロジェクトを作成することはできます。

プロジェクト テンプレートの作成

テンプレート ファイルの作成方法は、.ism プロジェクト ファイル (基本の MSI、InstallScript、InstallScript MSI、マージ モジュール) を使用するプロジェクト タイプと、.issuite プロジェクト ファイルを使用するプロジェクト タイプのどちらのテンプレートを作成するのかによって異なります。

.ism プロジェクト ファイルを使用するプロジェクトのプロジェクト テンプレート ファイルを作成する

任意の .ism プロジェクト ファイル (インストール プロジェクトとマージ モジュール プロジェクトを含む) からプロジェクト テンプレート を作成できます。.ism プロジェクト ファイルからテンプレートを作成すると、テンプレート ファイルに拡張子 .ist が付きます。

タスク ***.ism プロジェクトからテンプレート (.ist) を作成するには、以下の手順に従います:***

1. 希望のデフォルトの設定、ダイアログ、機能、その他を含めるプロジェクトを編集します。
2. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
3. **[名前を付けて保存]** ボックスで **[InstallShield テンプレート (*.ist)]** を選択します。
4. ファイル名 (.ist 拡張子を付ける) を入力して、新しいテンプレートの場所を選択します。テンプレートの推奨場所は、プロジェクトの場所フォルダーの下にある [テンプレート] フォルダーです。
5. **[OK]** をクリックします。

InstallShield でテンプレートを開き、変更を追加することができます。最終バージョンのテンプレートを使用して、新規プロジェクトを作成 することができます。

.issuite プロジェクト ファイルを使用するプロジェクトのプロジェクト テンプレート ファイルを作成する

任意のアドバンスト UI またはスイート アドバンスト UI プロジェクト (.issuite) をテンプレートに指定できます。これらのプロジェクト タイプのプロジェクト ファイルとテンプレート ファイルは、同じ拡張子 .issuite を持ちます。

タスク ***.issuite プロジェクト ファイルをテンプレート ファイルとし、[新規プロジェクト] ダイアログ ボックスでそのテンプレートを選択できるようにするには、以下の手順に従います:***

[新規プロジェクト] ダイアログ ボックスの **[すべてのタイプ]** タブを右クリックしてから、**[新規テンプレートの追加]** をクリックします。テンプレートとして使用する .issuite ファイルを参照して、**[すべてのタイプ]** タブに新しく追加したテンプレートのアイコンが表示されます。

テンプレートを使用した新規プロジェクトの作成

時間を節約するため、デフォルトの設定と設計要素を持つ単一のプロジェクトテンプレート上に複数のプロジェクトを作成できます。テンプレートは、システム上の任意の場所に格納することが可能です。レポジトリに格納することもできます。

タスク ***テンプレートを出発点としてプロジェクトを作成するには、次の手順に従います。***

1. **[ファイル]** メニューで、**[新規]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[すべてのタイプ]** タブをクリックします。
3. 使用するテンプレートを選択します。

   ***メモ・レポジトリに格納されているテンプレートを表示する**には、プロジェクトタイプのボックスを右クリックして、レポジトリのテンプレートを表示するが有効となっていることを確認します。タブへテンプレートを追加するには、プロジェクトタイプのボックスを右クリックして、**[新規テンプレートの追加]** をクリックします。そして必要なテンプレートファイル (.ist ファイル) を参照します。*
4. **[プロジェクト名]** ボックスで、プロジェクトの名前を入力します。
5. **[場所]** ボックスに、テンプレートを格納するディレクトリへのパスを入力するか、**[参照]** ボタンをクリックして、その場所を指定します。
6. **[OK]** をクリックします。InstallShield が選択したテンプレートを基にして、新しいプロジェクトを作成し、新しいプロジェクトは InstallShield で開かれます。

新しいプロジェクトは、事実上、プロジェクト テンプレートと同じですが、InstallShield が新しい製品コード (インストール プロジェクトの場合)、パッケージ コード、およびアップグレード コード (Windows Installer ベースのインストール プロジェクトの場合)、それにマージ モジュール プロジェクトの モジュール ID を生成する点が異なります。

プロジェクト テンプレートをリポジトリにパブリッシュする

別のプロジェクトで再利用、または他のユーザーと共有したい既存のテンプレートがある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***テンプレートをリポジトリにパブリッシュするには、次の手順に従います。***

1. **[ファイル]** メニューで、**[新規]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[すべてのタイプ]** タブをクリックします。
3. パブリッシュするテンプレートを右クリックしてから、**[テンプレートをリポジトリへパブリッシュする]** をクリックします。**パブリッシュ ウィザード**が開きます。
4. パブリッシュ ウィザードのパネルを完成します。

リポジトリにあるテンプレートに基づいてプロジェクトを作成しても、現在のプロジェクトと既存するリポジトリ テンプレートとの間にリンクはありません。テンプレートを変更してリポジトリへ再パブリッシュしても、テンプレートに基づいて作成したプロジェクトには影響しません。ただし、リポジトリにある更新済みのテンプレートに基づいて新しいプロジェクトを作成することはできます。

サンプル プロジェクト

InstallShield インストールには、いくつかのプロジェクト (.ism) ファイルの例が付属しています。これらのプロジェクトは、Samples フォルダーに格納されています。デフォルト保存先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples

これらのプロジェクトを見ると、インストール プロジェクトのある側面 (リリースフラグ等) をどのように導入するかが分かります。

プロジェクト アシスタント

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield で用意されている プロジェクト アシスタントを使って、基本のインストール プロジェクトを素早く簡単に作成することができます。プロジェクト アシスタントはインストール プロジェクト作業のフレームワークを提供し、プロジェクト作成プロセスの手順を案内し、その関連情報を提供します。

プロジェクト アシスタントのしくみ

基本の MSI プロジェクトまたはトランスフォームを新しく作成すると、[プロジェクト アシスタント] ビューが自動的に開きます。

プロジェクト アシスタントで入力した情報は、基本となるプロジェクト ファイルに直接保存されます。したがって、インストールデザイナ(下記参照)に切替えて、InstallShield インストール開発環境(IDE) の機能をフル活用して情報を表示したり変更することができます.この場合、プロジェクト アシスタントを使用すると、必要に応じてインストール デザイナーを使用する高度なインストールのために基礎を作成します。

インストールデザイナとの統合

[インストール デザイナー] タブには、作成中のプロジェクトの種類に利用可能な InstallShield インターフェイスのビューがすべて表示されます。ここから、より複雑でパワフルな要素をインストール プロジェクトに追加することができます。プロジェクト アシスタントでインストール プロジェクトを作成してから、インストールデザイナー を使用してプロジェクト要素を微調整することができます。

インストール デザイナーとプロジェクト アシスタントは同時に実行されます。一方に変更を加えると、すぐにもう一方にその変更が反映されます。たとえば、[インストール デザイナー] タブを使用時にある機能を削除すると、その機能はインストール プロジェクトからなくなり、プロジェクト アシスタントでも表示されなくなります。

このセクションには、さらなる詳細を含む次のサブセクションが含まれています:

- プロジェクト アシスタントを使用する
- [アプリケーション情報] ページ
- [インストール要件] ページ
- [インストール アーキテクチャ] ページ
- [アプリケーション ファイル] ページ
- [アプリケーションのショートカット] ページ
- [アプリケーション レジストリ] ページ
- [インストール インタビュー] ページ
- [インストール ローカリゼーション] ページ
- [インストールのビルド] ページ

プロジェクト アシスタントを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストール プロジェクトを新しく作成すると、プロジェクト アシスタントが自動的に開きます。ホーム ページには、インストール デザインのダイアグラムがあり、インストール作成の手順が一目で分かるようになっています。プロジェクト アシスタントを使ってプロジェクトを作成するか、[インストールデザイナ] タブをクリックして基本のインストール プロジェクトをさらに定義します。

プロジェクト アシスタントの [他のオプション]、[他の場所]、および [ヘルプ リンク] を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの各ページにある左のコラムのリンクは、インストールの作成および情報の検索に使用します。

- **その他のオプション**—プロジェクト アシスタント ページにある特定部分に関する追加構成オプションを提供します。これらは、プロジェクト アシスタントの機能のための他より一般的ではないオプションです。
- **他の場所**—現在のプロジェクト アシスタントのページに対応するインストール デザイナー内のビュー。リンクをクリックすると、インストール デザイナが起動され、そのビューがアクティブになります。
- **ヘルプ リンク**—このリストは現在のプロジェクト アシスタント ページに関連するヘルプ トピックへのリンクを提供します。

プロジェクト アシスタント内を移動する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***別のプロジェクト アシスタントに移動するには、次のどれかを実行します。***

- 特定のページに直接移動するには、ページの下部にあるナビゲーション バーで表示されている該当するアイコンをクリックします。
- プロジェクト アシスタントの手順に従うには、以下の通りに実行します。
  - [次へ] または [戻る] 矢印ボタンをクリックして、前/後ろに移動します。
  - CTRL+TAB を押すと、次のページに移動し、CTRL+SHIFT+TAB を押すと、前のページに移動します。
- [ホーム] に戻ってインストール デザイン ダイアグラムを表示するには、ナビゲーション バーの [ホーム] ボタンをクリックします。

インストール デザイナーを開く

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール デザイナー] タブには、InstallShield インストール開発環境 (IDE) のビューが表示されます。このタブでは、プロジェクト アシスタントでは実現不可能な、より複雑でパワフルな要素をインストール プロジェクトに追加することができます。インストール デザイナーでビューを開くには、[インストール デザイナー] タブをクリックします。

メモ: *インストール デザイナーとプロジェクト アシスタントは同時に実行されます。一方に変更を加えると、すぐにもう一方にその変更が反映されます。*

プロジェクト アシスタントの表示または非表示

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントでは、簡素化されたインストール プロジェクトのタスクを見ることができます。インストール デザイナーでは、これらすべてのタスクに加え、プロジェクトをさらに高度にカスタマイズすることができます。

InstallShield プロジェクトの作成をインストール デザイナーのみを利用して行う場合、プロジェクト アシスタントを非表示にして、タブが InstallShield インターフェイスで表示されないようにすることができます。同様に、プロジェクト アシスタントが非表示になっている場合、それを選択で表示することもできます。

タスク

プロジェクト アシスタントの表示または非表示の選択は、以下の手順に従います:

[ビュー] メニューで、**[プロジェクト アシスタント]** をクリックします。

プロジェクト アシスタント コマンドの隣にチェック マークがあるとき、プロジェクト アシスタントのタブが InstallShield インターフェイスに表示されます。チェック マークが表示されていないとき、プロジェクト アシスタントは非表示になっています。

プロジェクト アシスタント コマンドの隣にチェック マークがないとき、インストール デザイナーが、プロジェクトを新規作成するときのデフォルト タブになります。

[アプリケーション情報] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーション情報] ページで、プロジェクトがインストールするアプリケーションに関する情報を指定することができます。これには、アプリケーション名とバージョン、会社名、Web サイト アドレス、およびアプリケーション アイコンがあります。

コントロール パネルの [プログラムの追加と削除]

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

コントロール パネルの [プログラムの追加と削除] には、コンピューター システムにインストールされているアプリケーションの一覧が表示されます。[プログラムの追加と削除] から特定のプログラムについての情報を表示したり、プログラムを追加、変更または削除することができます。

プロジェクト アシスタントの [アプリケーション情報] ページに提供する情報は、アプリケーションがインストールされた時にその [プログラムの追加と削除]の情報を完成する際に利用されます。

インストールの会社名と製品名

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

会社名と製品名は、インストール プロジェクトの数箇所で使用されます。

インストール プロジェクト内で会社名はどのように使われますか？

会社名は、アプリケーションのデフォルト インストール ディレクトリを設定するのに使用されます。また、エンド ユーザーのシステム上のアプリケーションのコントロール パネルにある [プログラムの追加と削除] でも使用されます。

インストール プロジェクト内で製品名はどのように使われますか？

製品名はアプリケーションの [プログラムの追加と削除] (サポート情報リンク) で使用されてます。これはデフォルト インストール ディレクトリの設定にも使用されます。

[インストール要件] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール要件] ページでは、ターゲット システムのインストールの要件を簡単に設定することができます。たとえば、アプリケーションに特定の OS が必要な場合、このページの最初の部分で示すことができます。

プロジェクト アシスタントでオペレーティング システムを指定する

プロジェクト: *プロジェクト アシスタントでオペレーティング の設定が可能なプロジェクトの種類は以下の通りです:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [インストール要件] ページでオペレーティング を指定すると、InstallShield が起動条件を作成します。これらの条件は .msi ファイルの LaunchCondition テーブルに追加されます。このテーブルはダイレクトエディター のテーブルで表示および編集することが可能です。

InstallShield がオペレーティング システム起動条件を作成する方法

[インストール要件] ページでオペレーティングを指定すると、アプリケーションをサポートしないオペレーティング システムを除外することになります。

たとえば、最新 Windows オペレーティング システムのチェック ボックスのみを選択した場合、InstallShield は [インストール要件] ページで選択しなかったオペレーティング システムを除外する起動条件を作成します。この起動条件タイプでは、将来的にリリースされる Windows オペレーティング システムのバージョンが自動的にサポートされます。これは起動条件で除外されていないためです。

インストールによる要件確認のタイミング

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

必要なソフトウェアまたはオペレーティング システムがターゲット システムに確実に存在するよう、インストールはファイル転送を開始する前にインストールについてこれらの必要条件を確認します。

ソフトウェア要件のランタイム メッセージを変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストールにソフトウェア要件があり、ターゲット システムに選択されたソフトウェアがない場合、ランタイムのメッセージがインストール中に表示されます。表示されるメッセージは編集できます。

タスク ***ランタイム メッセージを編集するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[インストール要件]** ページを開きます。
2. ソフトウエア要件に関する質問には、**[はい]** を選びます。
3. アプリケーションに必要なソフトウェアを選択します。デフォルトのランタイム メッセージが右側に表示されます。
4. ランタイム メッセージを編集します。

カスタム インストール要件の作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール デザイナー] で製品の "インストール条件" 設定を使って、カスタマイズされたインストール要件を作成できます。この設定に、インストールを起動する前に Windows Installer が評価しなくてはならない条件を作成することができます。詳細については、「製品の条件を設定する」を参照してください。

また、システム検索ウィザードを利用してターゲット システム上にある特定のファイル、フォルダー、レジストリキー、または .ini 値を検索することもできます。システム検索ウィザードを利用して、検索値を使った条件を作成することができます。

[インストール アーキテクチャ] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[インストール アーキテクチャ] ページで、インストールでエンド ユーザーに表示する機能を指定することができます。機能とは、ユーザーから見て個別にインストール可能な最小の製品構成単位のことです。インストール中に [カスタム] セットアップ タイプを選択すると、個々の機能がエンドユーザーに対して表示されます。

メモ: *機能には、サブ機能やサブサブ機能などを含めることも可能で、セットアップ プログラムの要求に応じて必要なレベルの多層構造化が可能です。*

プロジェクト アシスタントで機能を追加する

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***機能を追加するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[インストール要件]** ページを開きます。
2. **インストール アーキテクチャをカスタマイズしますか?** の質問で、**[はい]** を選択します。
3. メイン機能を追加するには、**[インストール アーキテクチャ]** エクスプローラーをクリックします。サブ機能を追加するには、親機能となる機能をクリックしてから **[新規作成]** をクリックします。プロジェクト アシスタントは新しい機能を作成します。
4. 機能に名前を付けるか、後で **[名前の変更]** をクリックして名前を入力します。

複数機能インストールを作成するかどうかを判断する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

機能は、エンド ユーザーから見たインストールの構成ブロックです。このため、機能はインストール内部での機能上はっきりと区別できる要素を象徴しなくてはなりません。

アプリケーションに機能的に異なるブロックが複数存在する場合、複数機能インストールを作成しなくてはなりません。たとえば、インストールにアプリケーション (.exe ファイル) とヘルプ ライブラリ (.hlp ファイル) を含む場合、インストール プロジェクトは各要素につき 1 つの機能とした最低 2 つの機能を含まなくてはなりません。

プロジェクト アシスタント内での複数機能インストールの作成に関する詳しい情報は、「複数の機能を持つインストールを作成する.」をご覧下さい。

複数の機能を持つインストールを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストール デザイナー (IDE) のプロジェクト アシスタントを利用して、複数機能を持つインストールを作成することができます。

タスク ***プロジェクト アシスタントで複数機能インストールを作成するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[インストール要件]** ページを開きます。
2. インストール アーキテクチャをカスタマイズするには **[はい]** を選択します。
3. **[インストール アーキテクチャ]** エクスプローラーをクリックしてから、**[新規作成]** をクリックします。プロジェクト アシスタントは新規機能を作成します。
4. F2 を押すか、機能を右クリックして、**[名前の変更]** を選択して新しい機能の名前を付けます。
5. 同じレベルに別の機能を追加するには、**[インストール アーキテクチャ]** エクスプローラーをクリックしてから **[新規作成]** をクリックします。サブ機能を作成するには、親機能となる機能をクリックしてから **[新規作成]** をクリックします。プロジェクト アシスタントは新しい機能を作成します。
6. 必要に応じて機能およびサブ機能を追加します。

インストールデザイナの機能を使った作業については、「機能の作成」を参照してください。

デフォルトの機能

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

*デフォルトの機能*の概念はプロジェクト アシスタントでのみ存在します。インストール プロジェクトに追加されたすべてのリソース (たとえば、ファイルまたはレジストリデータ) は機能に割り当てる必要があります。リソースが機能に割り当てられていない場合、それらは実行時にターゲット システムへインストールされません。

デフォルトの機能を利用すると、プロジェクト アシスタントでのオーサリング作業が簡素化されます。プロジェクトリソースが確実にインストールされるよう機能への割り当てに気を配る必要はありません。レジストリデータを追加するとき、新規ショートカットを作成するとき、またはすべてのアプリケーションデータが選択された時にファイルを追加するとき、これらすべてのリソースはデフォルト機能へ追加されます。

デフォルトの機能の設定

デフォルトの機能は、プロジェクト アシスタントの [インストール アーキテクチャ]ページで設定することができます。

機能が存在しない、またはデフォルトの機能が選択されていない場合はどうなりますか?

[インストールアーキテクチャ]ページへ移動したとき、および [アプリケーションファイル]、[アプリケーションショートカット]、または [アプリケーションレジストリ] ページにデータを追加したとき、InstallShield は 最初のルート機能をデフォルトの機能として選択します。機能が存在しない場合、InstallShield がサイレントで作成します。

機能の階層を定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

最上位の機能とは、機能階層の最も上にある機能です。最上位の機能にはインストールするアプリケーション、ヘルプ ライブラリ機能、そしてサンプル プロジェクト機能などを含みます。

最上位の機能の下には、サブ機能または 子機能があります。これはインストールの都合上、別の機能に依存している機能です。*親*機能 (最上位の機能) がターゲット システムにインストールされないと、子機能もインストールされません。

[アプリケーション ファイル] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーションファイル] ページでは、機能に関連付けるファイルを指定することができます。

プロジェクト アシスタントで機能にファイルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***ファイルを機能に追加するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. ページ上部にある機能リストで、ファイルが含まれる機能を選択します。
3. **[インストール先コンピューター]** エクスプローラーで、ファイルを追加するフォルダーを選択します。
4. **[ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 追加するファイルを参照します。
6. **[開く]** をクリックして、選択した機能にファイルを追加します。「追加したファイルには依存関係がある可能性があります」というメッセージが表示されます。
7. インストール プロジェクトに依存関係を自動的に追加するようにする場合は、**[はい]** をクリックします。

プロジェクト アシスタントで機能からファイルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***ファイルを機能から削除するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. 削除するファイルをクリックして、**Delete** キーを押します。

固定のフォルダーの場所へファイルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ターゲット システム上のどの場所にプロジェクトファイルをインストールするのかが明白な場合、固定フォルダーの場所へのパスをハードコードで入力することができます。

タスク ***プロジェクト アシスタントでファイルを固定フォルダーの場所へ追加するには、以下の手順を実行します。***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. **[インストール先コンピューター]** を右クリックして、**[新しいフォルダー]** を選択します。
3. 新しいフォルダーの名前には、たとえば C: の様にインストール先を含むドライブ名を入力します。
4. ドライブ名フォルダーの下にサブフォルダーを追加して、詳しいインストール先のパスを定義します。

追加の定義済みフォルダーを表示する

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [アプリケーション ファイル] ページには、一般的によく利用される定義済みフォルダーが表示されます。このページで定義済みフォルダーを表示または非表示にすることができます。

タスク ***追加定義済みフォルダーを表示するには、以下の手順に従います:***

1. プロジェクト アシスタントで、**[アプリケーション ファイル]** ページを開きます。
2. **[インストール先コンピューター]** を右クリックして、**[定義済みフォルダーを表示]** を選択します。
3. 定義済みフォルダーのリストで表示するフォルダーを選択します。

ヒント: *定義済みフォルダーを非表示にするには、定義済みフォルダーのリストで該当するものについて選択を解除します。*

[アプリケーションのショートカット] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーション ショートカット] ページでは、ターゲット システムのデスクトップ、または [スタート] メニュー上にあるアプリケーション ファイルのショートカットを指定できます。デフォルトで、このページは [プロジェクト アシスタント] を使ってプロジェクトに追加された各実行可能ファイルのショートカットを表示します。これらは削除することができ、インストール プロジェクトに入れたショートカットを他のファイルに追加することもできます。

ファイル拡張子

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ファイル名拡張子の関連付け、つまりファイルの関連付けは、特定の種類のファイルを開くのにどのアプリケーションを使用すべきかを Windows に指示するためのレジストリ設定です。たとえば、Windows は通常 Windows のメモ帳を使用してテキストファイル (.txt の拡張子を持つファイル) を開き、Microsoft ペイントを使用してビットマップファイル (.bmp 拡張子を持つファイル) を開きます。

ファイル拡張子を使って、ファイルにアクセスせずにファイルの種類を識別することができます。ファイル名の末尾に接尾辞 (.abc) が追加されます。ファイル拡張子はまた、別のアプリケーションが拡張子に基づいてそのファイルと互換性があるかどうか (たとえば、そのファイルを開くのか変更するのか) を識別するのにも便利です。

InstallShield では、[コンポーネント] ビューまたは [セットアップのデザイン] ビューでコンンポーネントの [詳細設定] 領域にある設定を構成して、独自のファイル拡張子を登録できます。ファイル拡張子を登録すると、エンド ユーザーがファイルをクリックしたとき、ターゲット マシンのオペレーティング システムが特定のファイル拡張子を持つファイルを特定のアプリケーションで開くように指示することができます。

インストールに含まれていないファイルへのショートカットを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

既にターゲット システム上に存在するファイルを対象としたショートカットを作成するようにインストールを設定できます。このファイルはインストール プロジェクトに含まれている必要はありません。

タスク ***インストールに含まれていないファイルへのショートカットを作成するには、以下の手順を実行します。***

1. インストール デザイナーを開きます。
2. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
3. **[ショートカット]** エクスプローラで、ショートカットのインストール先を右クリックしてから、**[既存ファイルへの新しいショートカット]** をクリックします。**[ショートカット ターゲットの参照]** ダイアログが開きます。
4. ターゲット ファイルの場所を参照して **"ファイル名"** 設定にファイルの名前を入力します。
5. **[OK]** をクリックします。
6. ショートカットの設定を構成します。

プロジェクト アシスタントでデフォルトのショートカットを変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***デフォルトのショートカットを変更するには、以下の手順に従います:***

1. **プロジェクト アシスタント**で、**[アプリケーション ショートカット]** ページを開きます。
2. 変更するショートカットを選択します。
3. 必要に応じて変更を行います。

プロジェクト アシスタントでショートカットのターゲットにファイル拡張子を関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ファイル拡張子とショートカットのターゲットを関連付けることができます。これを行うとき、Windowsはターゲット ファイルを使用し、指定された拡張子のファイルを開きます。たとえば、.txt と入力すると、エンドユーザーが .txtファイルを開く際、このショートカットのターゲットファイルが起動して開きます。

タスク ***ショートカットのターゲットをファイル拡張子に関連付けるには、以下の手順に従います:***

1. **プロジェクト アシスタント**で、**[アプリケーション ショートカット]** ページを開きます。
2. ショートカットをクリックして、ショートカット オプションを有効にします。
3. **[ショートカットとファイル拡張子を関連付けるオプション]** を選択します。
4. このショートカットのターゲットと関連付けるファイル拡張子を入力します (例、txt)。カンマで区切って複数の拡張子を追加することもできます。

[アプリケーション レジストリ] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アプリケーション レジストリ] ページでは、アプリケーションに必要なレジストリ データを指定できます。

レジストリの更新

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

レジストリはコンピューターの構成情報が入ったデータベースです。コンピューターのレジストリに情報にはユーザープロフィール、コンピューターにインストールされたハードウエアおよびソフトウェア、プロパティ設定が含まれます。

アプリケーションに必要なレジストリデータの調べ方

アプリケーション開発者にレジストリ情報をたずねてください。特に、インストールするアプリケーションにユーザー固有 (HKEY_CURRENT_USER) またはマシン固有 (HKEY_LOCAL_MACHINE) の設定が必要かどうかを知る必要があります。

開発者からインストールに追加する .reg ファイルを受け取ります。InstallShield では、.reg ファイルをインストール プロジェクトにインポートすることができます。

プロジェクト アシスタントでレジストリ データを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***レジストリ データを構成するには、以下の手順に従います。***

1. **プロジェクト アシスタント**で、**[アプリケーションのレジストリ]** ページを開きます。
2. レジストリ データの構成についての質問に対して、**[はい]** を選選択します。
3. データを追加するレジストリ アイテムを右クリックして、**[新規]** を選択してから **[キー]** をポイントします。
4. キーに名前を入力します。
5. キーを右クリックし、[**新規**]を選択してから適切なコマンドをポイントします。登録するデータのタイプによって、[デフォルト値]、[文字列値]、[バイナリ値]、[DWORD 値]、[拡張可能文字列値] または [複数行値] から選択します。

プロジェクト アシスタントでレジストリ データの値を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***レジストリ データを変更するには、以下の手順に従います。***

1. **プロジェクト アシスタント**で、**[アプリケーションのレジストリ]** ページを開きます。
2. データをダブルクリックする。

複数行文字列をダブルクリックすると、**[複数行文字列値]** ダイアログ ボックスが開きます。別の種類のレジストリ データをダブルクリックすると、**[データの編集]** ダイアログ ボックスが開きます。

3. データを編集して、**[OK]** をクリックします。

機能にレジストリ データを関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [アプリケーション レジストリ] ページに追加したすべてのレジストリ データがプロジェクトのデフォルトの機能へ追加されます。インストール デザイナーでレジストリ データを別の機能に関連付けることができます。

タスク

インストール デザイナーを使って、レジストリ データをデフォルトの機能以外の機能に関連付けるには、以下の手順を実行します:

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[ビュー フィルター]** リストで、レジストリ データを関連付ける機能を選択します。
3. レジストリ データを適切なレジストリの場所で作成するか、またはドラッグアンドドロップします。

レジストリ データで変数データ型を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield ではインストール プロジェクト用レジストリデータを作成する際に、変数データ型またはプロパティを利用することができます。

タスク

レジストリで INSTALLDIR を変数として利用するには、以下の手順を実行します(Windows Installer ベースのプロジェクトのみ)。

1. **プロジェクト アシスタント**で、**[アプリケーションのレジストリ]** ページを開きます。
2. **[はい]** を選択して、アプリケーションがインストールするレジストリ データを構成することを示します。

3. HKEY_CLASSES_ROOT を右クリックし、**[新規]** をポイントしてから、**[キー]** をクリックします。
4. キーに [インストール先] という名前を付けます。
5. [インストール先] キーを右クリックし、**[新規作成]** をポイントしてから、**[文字列値]** をクリックします。
6. 文字列値に My Installation Location と名前を付けます。
7. My Installation Location キーをダブルクリックします。**[データの編集]** ダイアログ ボックスが開きます。
8. **"値データ"** フィールドに **[INSTALLDIR]** と入力します。

実行時に、[INSTALLDIR] の値はインストール ディレクトリと置き換えられます。

アプリケーション パス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

アプリケーション パスのレジストリ キーには、指定アプリケーションの .dll ファイル用のプライベート検索パスとして Windows が使用するデータが含まれます。アプリケーションの .dll ファイルを PATH 環境変数で指定されていないディレクトリ (および、アプリケーションのディレクトリ以外) にインストールする場合は 、インストール中に .dll ファイル ディレクトリを含む適切なアプリケーション パスを設定するようにしてください。アプリケーション パス情報は、レジストリの HKLM¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥App Paths¥AppName.exe の下に格納されます。

[インストール インタビュー] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *MSI データベース*

[インストール インタビュー] ページでは、インストール プログラムの実行時にエンド ユーザーに表示するダイアログ ボックスを選択します。このページでの回答によって、プロジェクト アシスタントは対応するダイアログをインストール プロジェクトに追加します。

プロジェクト アシスタントでインストールに使用するダイアログを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *MSI データベース*

InstallScript プロジェクトでは、インストール完了時にアプリケーションを起動するオプションは使用できません。

プロジェクト アシスタントの [インストール インタビュー] ページに表示される質問に答えると、作成したインストール プログラムが実行されるときにエンド ユーザーに対して表示されるダイアログを指定することができます。

タスク ***インストールのダイアログを指定するには、次の手順に従います。***

1. **使用許諾契約ダイアログを表示しますか?**—**[はい]** を選択して、使用許諾契約書ファイルを参照します。
2. **ユーザーに対して会社名とユーザー名の入力をプロンプトしますか?** — **[はい]** を選択してこの情報を要求するダイアログを表示します。
3. **ユーザーがアプリケーションのインストール場所を変更することができるようにしますか?** — **[はい]** を選択して、エンド ユーザーがインストールの場所を変更できるようにします。詳細については、「エンドユーザーによるインストール先の変更を許可する」を参照してください。
4. **ユーザーがアプリケーションの特定部分のみ選択してインストールできるようにしますか?** — **[はい]** を選択して、エンド ユーザーがアプリケーションの特定部分のみインストールできるようにします。詳細については、「インストールが選択できるインストールを作成する」を参照してください。
5. **インストールの完了時にアプリケーションを起動するオプションをユーザーに提供しますか?**—**[はい]** を選択して、アプリケーション ファイルを参照します。このオプションが **[はい]** に設定されると、最後のダイアログにはチェック ボックスがあり、エンドユーザーが **[完了]** ボタンを押すとアプリケーションが直ちに起動します。

ヒント: *カスタム グラフィックをインストール ダイアログに追加するには、[他のオプション] セクションでリンクをクリックし、[ダイアログイメージ] ダイアログ ボックスを起動します。*

エンドユーザーによるインストール先の変更を許可する

エンドユーザーがシステム上でソフトウェアをインストール場所を選べるように、インストール場所の変更を許可することができます。

Windows Installer ベースのインストールでは、Windows Installer プロパティ INSTALLDIR がデフォルト インストール ディレクトリとなります。ユーザーがインストール場所を変更できるようにすると、インストール中に CustomSetup ダイアログが表示されます。

InstallScript プロジェクトでは、TARGETDIR がデフォルトインストールディレクトリとなります。ユーザーがインストール場所を変更できるようにすると、インストール中に SdSetupType2 ダイアログが表示されます。

使用許諾契約書

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *MSI データベース*

製品をインストールするために、エンド ユーザーが特定の法的要件に同意しなくてはならない場合があります。たとえば、ほとんどのソフトウェア ベンダはエンド ユーザーによるソフトウェアのコピーや第三者への配布を禁止しています。

インストールは実行時に、[使用許諾契約] ダイアログでエンド ユーザー使用許諾契約書 (EULA) を表示できます。EULA は製品の使用について、エンドユーザーとの間に交わされる法的契約です。

[使用許諾契約書] ダイアログは、使用許諾契約書テキストを表示するとともに、ラジオボタングループ ([はい] または [いいえ]) を含みます。エンドユーザーが EULA に同意しない場合、ソフトウェアはインストールされることなくインストール処理が終了します。

タスク ***プロジェクト アシスタントで [使用許諾契約書] ダイアログをプロジェクトへ追加するには、以下の手順を実行します。***

1. **プロジェクト アシスタント**で、**[インストール インタビュー]** ページを開きます。
2. **[はい]** を選択して **[使用許諾契約]** ダイアログの追加を指定します。
3. 使用許諾契約書ファイルへのパスを入力する、またはファイルを参照します。

メモ: *次の情報を参考にして下さい:*

- *このファイルは、基本の MSI と MSI データベース プロジェクトではリッチ テキスト ファイル (.rtf) でなくてはなりません。*
- *InstallScript プロジェクトでは、どのライセンス ダイアログを使用するのか、またどのパラメーターに渡すのかによって、ファイルの種類が異なります。SdLicenseEx および SdLicense2Ex 関数は、リッチテキスト(.rtf) およびテキスト (.txt) ファイルをサポートします。*
- *InstallScript プロジェクトでは、**[使用許諾契約]**ダイアログで使用許諾契約テキスト内のハイパーリンクもサポートされています。リッチテキスト (.rtf) ファイルにハイパーリンクを含めることで、外部リソースへの直接アクセスを提供できます。実行時に、エンドユーザーはダイアログ内のハイパーリンクをクリックして、定義されたリンクターゲット (Webサイト、ドキュメント、拡張ライセンス条件など) を開くことができます。*

インストールが選択できるインストールを作成する

エンドユーザーがインストールする部分を選択してシステムにインストールするようなインストールを作成することができます。これは、インストール内で使用できる機能の一覧を表示するカスタムインストールです。エンドユーザーは実行時に表示されるダイアログでインストールする機能を選択することができます。

たとえば、インストールの中にアプリケーションの実行可能ファイル(.exe)ファイル、ドキュメント(.chm)ファイル、およびサンプルファイルが入っているとします。これらのファイルはすべて異なる機能の中に入っていて、オプション一覧からエンドユーザーに提供されます。エンドユーザーがアプリケーションのみを必要とする場合は、実行可能ファイルだけをインストールし、ドキュメント ファイルとサンプル ファイルはインストールしないように選択することができます。

[インストール ローカリゼーション] ページ

エディション: *複数言語サポートは、InstallShield の Premier Edition で使用することができます。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[インストールのローカライズ] ページで、インストールが対応する言語を指定することができます。文字列値と関連識別子を指定して、エンド ユーザーのインターフェイスで簡単に他の言語にローカライズすることができます。

メモ: *ドロップダウンメニューで編集する文字列データセットを指定できます(ページ上部)。*

プロジェクト アシスタントから文字列データをローカライズする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストール プロジェクトのテキストの多くは文字列エントリとして保存されます。インストールのローカライズを簡単にするため、InstallShield では文字列をエクスポートして翻訳し、翻訳した文字列をプロジェクトにインポートする機能があります。

タスク ***プロジェクト内のスクリプト ファイルすべてを翻訳用にエクスポートするには、以下の手順に従います:***

1. **プロジェクト アシスタント**で、**[インストール のローカリゼーション]** ページを開きます。
2. ページ上部にあるデータ リストで、**[すべての文字列データ]** を選択します。
3. 言語一覧の中から、言語を選択します。
4. **[他のオプション]** 領域で、**[ローカライズ文字列のエクスポート]** をクリックします。**[ファイル名]** ダイアログ ボックスが開きます。
5. 任意の場所にテキスト (.txt) ファイルを保存します。
6. テキスト ファイルを翻訳会社または担当部署に渡します。

タスク ***テキストファイルの文字列の翻訳終了後、その文字列をプロジェクトに再インポートすることができます。***

1. **プロジェクト アシスタント**で、**[インストール のローカリゼーション]** ページを開きます。
2. ページ上部にあるデータ リストで、**[すべての文字列データ]** を選択します。
3. 言語一覧で、インポートする文字列と一致する言語をクリックします。
4. **[他のオプション]** 領域で、**[ローカライズ文字列のインポート]** をクリックします。**[ファイル名]** ダイアログ ボックスが開きます。
5. .txt ファイルを参照して、**[開く]** をクリックします。翻訳済み文字列がプロジェクトに再インポートされます。

インストールでローカライズされた文字列データの使用のされ方

インストール プロジェクトには、インストール専用の文字列データとアプリケーション専用の文字列データの 2 種類の文字列データがあります。

InstallShield プロジェクトでの大半の文字列データは、エンドユーザーがインストールを実行するときにらいタイムで使用されます。但し、文字列データの一部はターゲット システムインストールされます。たとえば、ローカライズされたショートカットは、エンド ユーザーのシステムにインストールされるローカライズされた文字列データです。

[インストールのビルド] ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ヒント: *以下の情報は、(Visual Studio との統合を行わず) InstallShield 内部でビルドされたリリースに適用します。Visual Studio 内部から InstallShield リリースをビルドする方法については、「Microsoft Visual Studio でリリースをビルドする」を参照してください。*

[インストールのビルド] ページでは、ビルドするタイプを指定し、再配布可能ファイルをコピーする場所も指定することができます。パッケージにデジタル署名を行うこともできます。

プロジェクト アシスタントからインストールをビルドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript*
- *InstallScript MSI*

***ヒント・** 以下の情報は、(Visual Studio との統合を行わず) InstallShield 内部でビルドされたリリースに適用します。Visual Studio 内部から InstallShield リリースをビルドする方法については、「Microsoft Visual Studio でリリースをビルドする」を参照してください。*

タスク ***インストールをビルドするには次の手順を実行します。***

1. **プロジェクト アシスタント**で、**[インストールのビルド]** ページを開きます。
2. インストールのイメージの種類を選択します。

   ***メモ・** Windows Installer が既にインストールされているマシンには、Windows Installer の存在をチェックし、またそれが無かった場合にエンジンをインストールすることがない **"単一の MSI パッケージ"** オプションを選択します。*
3. InstallShieldで、ビルド後に自動的にインストールを別の場所にコピーする場合、各ビルド オプションの**オプション配布の設定** リンクをクリックして、場所を指定します。

   ビルド後に InstallShield がインストールを配布するように設定する場合、各ビルドオプションの **オプション配布**の設定 リンクをクリックして、**ビルド後に配布** をチェック ボックスを選択します。
4. Setup.exe ファイルにデジタル署名を行い、アプリケーションに含まれるコードが変更または破損していないことをエンド ユーザーに対して保証するには、**[セットアップにデジタル署名する] ハイパーリンク** をクリックします。[**セットアップにデジタル署名する**] ダイアログ ボックスが開きます。必要に応じて設定を指定します。
5. **[インストールのビルド]** をクリックします。

[出力] ウィンドウが開き、[ビルド] タブにビルドの進行状況についての情報が表示されます。[ビルド] タブがログファイル情報を表示すると同時にビルドが完了します。

***ヒント・** インストール デザイナーの [署名] タブでは、ビルド時に、インストールのどの部分に対してデジタル署名を行うかを指定できます。InstallShield は、作業中のプロジェクトの種類に応じて、リリースに含まれる次の任意およびすべてのファイルに署名することができます。*

- *基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトの Windows Installer パッケージ (.msi ファイルと .msm ファイル)*
- *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの Setup.exe ファイル*
- *InstallScript プロジェクトのメディア ヘッダー ファイル*
- *InstallScript プロジェクトのパッケージ (自己展開型実行可能ファイル)*
- *リリースの任意のファイル (アプリケーション ファイルを含む)*

詳細については、「ビルド時にリリースとそのファイルにデジタル署名を行う」を参照してください。

Windows ロゴ *・インストールのすべての実行可能ファイル（.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル）は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

プロジェクト アシスタントの完了後: 次のステップ

プロジェクト *・この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの一連のページでフィールドへの入力を完了すると、インストール プロジェクトの枠組みが完成します。これは機能するインストールとして利用することができ、また必要に応じてさらにカスタマイズすることも可能です。

プロジェクトをさらにカスタマイズする

インストールデザイナから、InstallShield で利用可能なすべてのインストール作成ビューへ簡単にアクセスすることができます。[プロジェクト アシスタント] ワークスペースの上部にある [インストールデザイナ] タブをクリックしてビューを表示します。

インストール デザイナー内には、作成中のプロジェクトの種類で利用可能なビューが一覧となった [ビュー リスト] があります。ワークスペースの左側に [ビュー リスト] を表示するには、F4 を押します。

InstallShield ビューについて詳しい情報

InstallShield の各ビューに関する詳細は、InstallShield ヘルプライブラリの [ビューリスト] セクションにあるトピックを参照してください。

InstallShield インターフェイスを使って作業する

InstallShield インターフェイスは、メニュー バー、ツールバー、ダイアログ ボックスなど通常の Windows ベースの要素をもったグラフィックなユーザー インターフェイスです。このセクションでは、これらの要素を使用した基本的なタスクの実行方法、および、インターフェイスのカスタマイズの方法について説明します。さらなる詳細は、次のセクションに分けて説明されています:

- ビュー リストを表示する
- InstallShield ユーザー インターフェイスでビューを開く
- 様々なビューで、[グループ ボックス] 領域を使って作業する
- ツールバーの表示または非表示
- ツールバーにボタンおよびメニューを追加する

- ツールバーからボタンおよびメニューを削除する
- カスタム ツールバーの作成
- [出力] ウィンドウを固定する/取り外す
- 様々なビューで、[スクリプト エディター] ペインを使って作業する

ビュー リストを表示する

タスク ***InstallShield インターフェイスでビュー リストを表示するには、以下の手順に従います:***

1. InstallShield インターフェイス上部の **[インストール デザイナー]** タブをクリックします。
2. 以下のいずれかの方法でビュー リストを表示します。
   - **[ビュー]** メニューで、**[ビュー リスト]** をクリックします。InstallShield インターフェイスの左側に、**ビュー リスト**が表示されます。
   - ツールバーの **[ビュー リスト]** ボタンをクリックします。
   - F4 を押すと、**[ビュー リスト]** が表示または非表示になります。

InstallShield ユーザー インターフェイスでビューを開く

InstallShield での多くの手順では、まず最初にインストール開発環境 (IDE)で特定のビューを開きます。

タスク ***ビューを開くには、以下の手順を実行します。***

1. **[インストール デザイナー]** タブをクリックします。IDE の左側に**ビュー リスト**が表示されます。**ビュー リスト** が表示されない場合、「ビュー リストを表示する」を参照してください。
2. **ビュー リスト**で、開くビューを選択します。使用できるすべてのビューを表示するには、[ビュー リスト] フォルダーを展開します。

様々なビューで、[グループ ボックス] 領域を使って作業する

InstallShield の多くのビューには、ビュー内の行をグループ分けすることができるグループ ボックス領域があります。このグループ ボックスを含むビューでは、行ヘッダーをグループ ボックス ([フィールドごとにグループ分けするときは、ここにフィールド名をドラッグします。] と表示される領域) にドラッグするするだけで、複数階層のグループに分けることができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。グループ ボックスを含むビューには、[文字列エディター]、[プロパティ マネージャー]、[再配布可能ファイル]、[前提条件]、および [パス変数] ビューなどがあります。

グループ ボックスを使用する際、以下の点を参考にしてください:

- 列ヘッダーをグループ ボックス領域に移動するとき、列ヘッダーをドラッグしてグループ ボックスにドロップします。

- 列ヘッダーをグループ ボックス領域にコピーするとき、CTRL キーを押しながら列ヘッダーをドラッグしてグループ ボックスにドロップします。この場合、列ヘッダーはそのままの位置に残り、またグループ ボックスにも表示されます。
- グループ ボックスのヘッダーをグループ ボックス領域にドロップするとき、別の列ヘッダー上にドロップすることができます。これによって、行が階層構造で表示されます。
- 列ヘッダーをグループ ボックスから削除するには、グループ ボックス領域からそれをドラッグして列ヘッダーの行にドロップします。列ヘッダーの行をドラッグすると、それをドロップしたときに列ヘッダー行のどの位置に表示されるのかが、矢印で示されます。
- グリッド内のアイテムを特定の列ヘッダーごとに並べ替えるには、グループ ボックス内または列ヘッダー行内の列ヘッダーをクリックします。

次の例を使って、ビュー内のコンテンツをグループ ボックスを使ってグループ化する様々な方法をデモンストレーションします。

デフォルトの動作: グループ ボックス領域を空白にする

デフォルトで、グループ ボックスに列ヘッダーは表示されません。次のスクリーン ショットは、[再配布可能ファイル] ビューの一部です。項目が [名前] 列で整列しています。

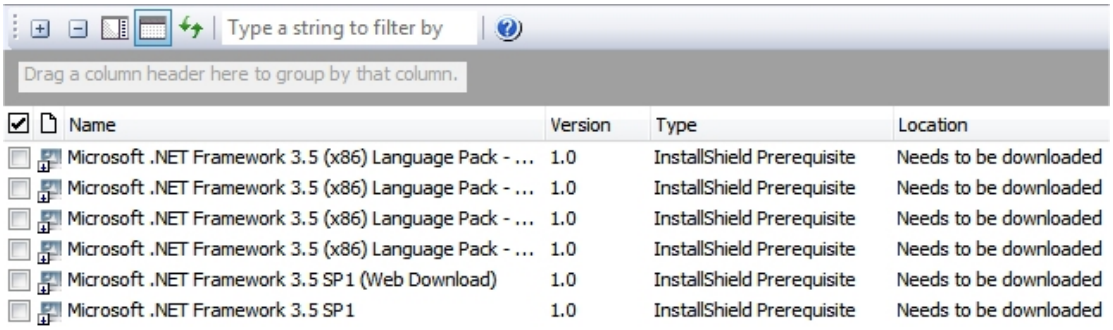


図 3: [グループ ボックス] 領域を空白にする

1 つの列ヘッダーでグループ化する

CTRL を押しながらグループ ボックスに列ヘッダーをドラッグ アンド ドロップするとき、グリッドの行が項目ごとにグループ化されます。次のスクリーン ショットは、[文字列エディター] ビューの一部です。翻訳が必要な最近追加および変更された行が識別しやすくグループ化されます。

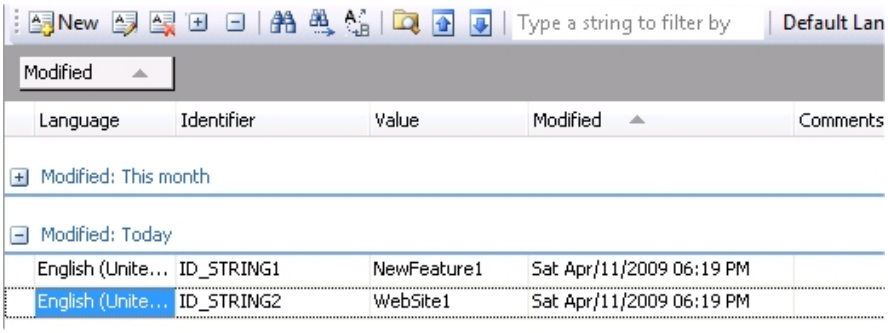


図 4: 1 つの列ヘッダーで行をグループ化する

2 つの列ヘッダーでグループ化する

CTRL を押しながらグループ ボックスに列ヘッダーをドラッグ アンド ドロップするとき、グリッドの行が複数の項目ごとにグループ化されます。次のスクリーン ショットは、[再配布可能ファイル] ビューの一部です。プロジェクトに追加された InstallShield 前提条件とマージ モジュールが、識別しやすくグループ化されます。

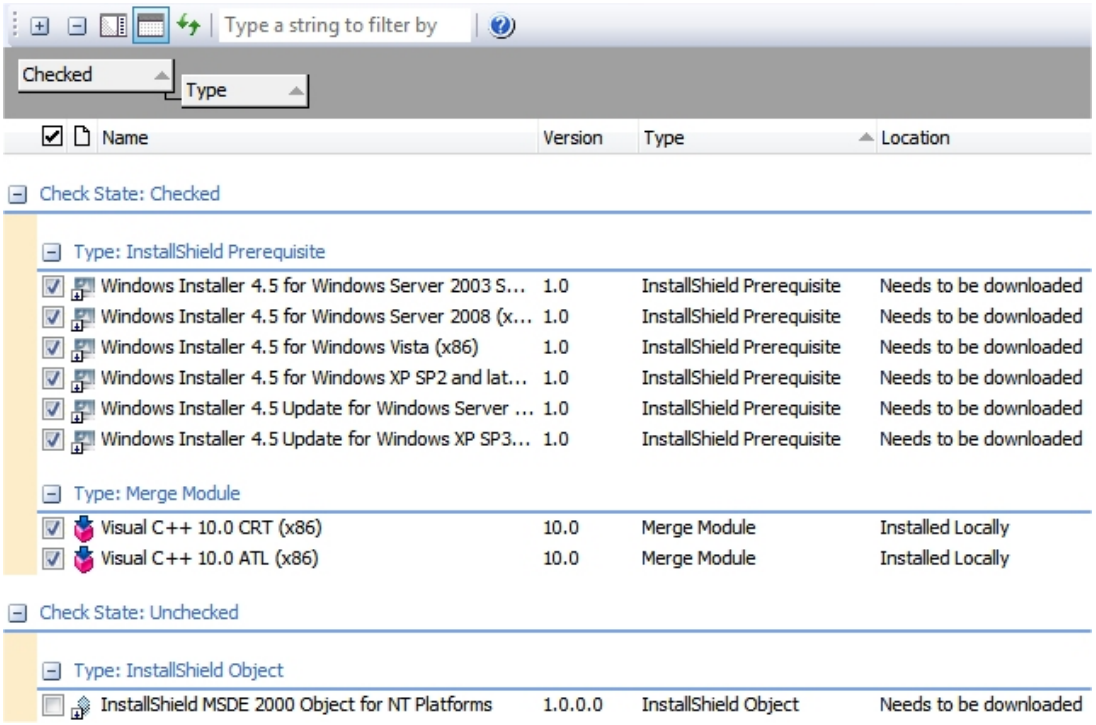


図 5: チェック ボックス列および種類列ごとに行を並べ替える

ツールバーの表示または非表示

タスク ***ツールバーを表示または非表示にするには、以下の手順を実行します。***

- ツールバーを右クリックして表示する、または非表示にするツールバーを選択します。
- **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。表示する各ツールバーのチェック ボックスを選択します。非表示にする各ツールバーのチェック ボックスを選択解除します。

ツールバーにボタンおよびメニューを追加する

タスク ***ツールバーにボタンまたはメニューを追加するには、以下の手順を実行します。***

1. 変更するツールバーが表示されていることを確認します。
2. **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。
3. **[コマンド]** タブをクリックします。
4. **[カテゴリ]** ボックスで、追加するボタンまたはメニューのカテゴリをクリックします。
5. **[コマンド]** ボックスから、ボタンまたはメニューを適切なツールバーへドラッグします。

ヒント: *独自のカスタム ツールバーを作成するには、ボタンまたはメニューをツールバーの近くにある灰色の空白部分にドラッグします。*

ツールバーからボタンおよびメニューを削除する

タスク ***ツールバーからボタンまたはメニューを削除するには、以下の手順を実行します。***

1. 変更するツールバーが表示されていることを確認します。
2. **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。
3. 削除するボタンまたはメニューを右クリックして、**[削除]** をクリックします。

カスタム ツールバーの作成

タスク ***カスタム ツールバーを作成するには、以下の手順を実行します。***

1. **[ツール]** メニューで、**[カスタマイズ]** をクリックします。**[カスタマイズ]** ダイアログ ボックスが開きます。
2. **[ツールバー]** タブをクリックします。
3. **[新規]** ボタンをクリックします。**[新規ツールバー]** ダイアログ ボックスが開きます。
4. **[ツールバーの名前]** テキスト ボックスにツールバーの説明的な名前を入力し、**[OK]** をクリックします。
5. メニューまたはボタンを追加して、新規作成したツールバーをカスタマイズします。

[出力] ウィンドウを固定する/取り外す

[出力] ウィンドウ、またその個別のタブは、InstallShield のワークスペースの任意の側面に固定するか、独立した場所にドラッグすることができます。

[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端にドラッグすると、固定ウィンドウとして表示されます。[出力] ウィンドウまたはそのタブの 1 つを InstallShield インターフェイスの端から離れた場所にドラッグすると、取り外されます。

タスク ***[出力] ウィンドウを取り外すには、以下の手順に従います:***

[出力] ウィンドウのタイトル バーを新しい場所にドラッグします。必要に応じて、[出力] ウィンドウのサイズを変更します。

タスク ***[出力] ウィンドウを固定するには、以下の手順に従います:***

[出力] ウィンドウのタイトル バーを InstallShield インターフェイスの右、左、上、下の端にドラッグします。

タスク ***[出力] ウィンドウのタブを取り外すには、以下の手順に従います:***

タブを新しい場所にドラッグします。必要に応じて、[出力] ウィンドウのサイズを変更します。

タスク ***[出力] ウィンドウのタブの 1 つを固定するには、以下の手順に従います:***

タブを InstallShield インターフェイスの右、左、上、下の端にドラッグします。

様々なビューで、[スクリプト エディター] ペインを使って作業する

InstallShield のいくつかのビューにはスクリプト エディター ペインがあり、そこにプロジェクトのコードを書き込むことができます。以下は、スクリプトエディター ペインを持つビューの例です:

- **[InstallScript] ビュー** – このビューでスクリプト ファイルを選択すると、InstallScript コードを書き込める [スクリプト エディター] ペインが表示されます。
- **[SQL スクリプト] ビュー** – このビューで SQL スクリプトを選択すると、[スクリプト] タブに SQL ファイルを編集できる [スクリプト エディター] ペインが表示されます。
- **[カスタム アクションとシーケンス] ビュー** – このビューで VBScript または JScript カスタム アクションを選択すると、[スクリプト] タブに VBScript または JScript ファイルを編集できる [スクリプト エディター] ペインが表示されます。

このセクションでは、InstallShield のスクリプト エディターの使い方について説明します。

スクリプト エディターでブックマークを使用する

ブックマークによって、スクリプト内の特定の行へ最小のキーストロークを使ってジャンプできます。ブックマークは、スクリプト エディターの左マージンに表示されます。InstallShield では、プロジェクトを閉じた時、ブックマークは保存されませんので注意してください。

タスク ***新しいブックマークを追加したり、既存のブックマークをクリアするには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - ブックマークを追加するスクリプト内の行に挿入ポイントを置きます。
   - クリアするブックマークを含むスクリプト内の行に挿入ポイントを置きます。
2. ALT+K を押します。

タスク ***挿入ポイントをブックマークが含まれる次の行に移動するには、以下を実行します:***

CTRL+K を押します。

タスク ***挿入ポイントをブックマークが含まれる前の行に移動するには、以下を実行します:***

CTRL+SHIFT+K を押します。

スクリプト エディターでオートコンプリート機能を有効または無効にする

InstallShield 様々なビュー (たとえば、[InstallScript] ビューや [SQL スクリプト] ビュー) のスクリプト エディターに入力するときに、オートコンプリート機能を使用するかどうかを指定できます。オートコンプリート機能を有効にすると、スクリプト エディター内に入力している文字で始まる関数、キーワード、定数、文字列識別子の一覧がアルファベット順でポップアップ リストに表示されます。用語全体を手入力する代りに、そのポップアップ リストから選択すると、InstallShield によってその用語がスクリプトに追加されます。

オートコンプリートを使うと、コードを手入力する手間が省けるため、作業効率が上がります。また、コードのスペル ミスを回避することができます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じオートコンプリートの設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***オートコンプリート機能をスクリプト エディターで有効にするかどうかを指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"オートコンプリート"** 設定で、適切なオプションを選択します:
   - **はい** – オートコンプリート機能を有効にすると、スクリプト エディター内に入力している文字で始まる関数、キーワード、定数、文字列識別子の一覧がアルファベット順でポップアップ リストに表示されます。デフォルトでは、これが設定されています。
   - **いいえ** – スクリプト エディター内にコードを入力しているとき、ポップアップ リストを表示しません。
3. **"ローカル変数を含める"** 設定で、適切なオプションを選択します:
   - **はい** – InstallScript エディター内に入力しているコードの文字で始まる使用可能な関数およびその他のスクリプト用語のポップアップ リストに InstallScript で定義されたローカル変数が追加されます。デフォルトでは、これが設定されています。
   - **いいえ** – InstallScript エディター内にコードを入力しているとき表示されるポップアップ リストに、ローカル変数は表示されません。

     [いいえ] を選択した場合、[InstallScript] ビューの中央ペインにある関数、プロパティ、およびメソッド フォルダーには、スクリプトファイルからの関数、プロパティ、またはメソッドが一覧表示されません。

     この設定で [はい] を選択して、[InstallScript] ビューでコードを入力しているときにパフォーマンスに影響が出た場合は、この設定を [いいえ] に設定します。

   メモ・"ローカル変数を含める" *設定は、[InstallScript] ビュー内のスクリプト エディターに適用します。たとえば [SQL スクリプト] ビューなど、その他のスクリプト エディターには反映しません。* ***"オートコンプリート"*** *設定に* ***[いいえ]*** *を選択すると、この設定は無視されます。*
4. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じて InstallScript エディターでオートコンプリート機能を有効または無効にします。

オートコンプリート機能の仕組みについての詳細は、「スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う」を参照してください。

スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う

InstallShield のスクリプトエディタでオートコンプリート機能を有効にすると、スクリプト エディター内に入力している文字で始まる関数、キーワード、定数、文字列識別子の一覧がアルファベット順でポップアップ リストに表示されます。ポップアップ リストには、使用中のビューによって言語に適したオプションが表示されます。たとえば、[InstallScript] ビューのスクリプト エディターでは、InstallScript コードのオートコンプリート機能がサポートされています。[カスタム アクションとシーケンス] ビューの VBScript スクリプト エディターでは、VBScript のオートコンプリート機能がサポートされています。

また、ローカル変数のオートコンプリート機能が有効な場合、[InstallScript] ビューのスクリプト エディターで表示されるポップアップ リストには、ローカル変数も含まれています。

オートコンプリート機能を有効にする方法については、「スクリプト エディターでオートコンプリート機能を有効または無効にする」をご参照ください。

タスク ***関数、キーワード、定数、または文字列識別子のオートコンプリート機能を使用するには、以下の手順に従います:***

1. 適切なスクリプト エディターを含むビューを開いてから、編集するスクリプトを選択します。
2. スクリプトでコードを入力する挿入ポイントを置きます。
3. 以下のいずれかを実行します。
   - 関数、キーワード、その他のスクリプト用語の最初の文字を入力します。

     使用しているビューによって、またそのビュー内のスクリプト エディターに入力した言語によって、スクリプト エディターに入力している文字で始まる関数、キーワード、その他のスクリプト用語のアルファベット順のリストを含むポップアップ リストが表示されます。入力した文字と一致する最初の用語がリストで選択されています。入力した文字と一致する用語がない場合、リスト内の用語は選択されていません。
   - [InstallScript] ビューでスクリプト エディタ使うとき、使用可能な文字列識別子のリストにオートコンプリート機能を使用する場合、文字列定数演算子 (@) を入力すると、InstallShield は文字列識別子がアルファベット順に並んだポップアップ リストを表示します。
4. 選択された用語が希望の用語と異なるか、用語が選択されなかった場合、次のいずれかの方法で別の用語を選択します。
   - ポップアップ リストのスクロールバーを使ってリスト内を移動してから、希望の用語をクリックして選択します。
   - 上矢印と下矢印キーを使って前または次の用語を選んで変更します。
5. 選択した用語をスクリプトに貼り付けるには、その用語をダブルクリックするか、ENTER または TAB キーを押します。

用語を貼り付けずにリストを閉じる場合は、ESC を押すかポップアップ リストの外側をクリックします。

[スクリプト エディター] 内で InstallScript 関数の呼び出しについてのヒントを有効または無効にする

InstallShield では、[InstallScript] ビューを使ってスクリプトに関数呼び出しを入力するときに、InstallScript 関数の呼び出しについてのヒント (一種のツールヒント) を表示するかどうかを指定できます。関数呼び出しのヒントは、ビルトイン関数のパラメーター情報を表示します。その他、ビルトイン関数の説明や、入力しているパラメーターについての説明も表示します。

関数呼び出しのヒントを使うと、別のウィンドウに切り替えなくてもスクリプト エディター内で言語情報を参照できます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じオートコンプリートの設定が、すべてのスクリプト エディター*

で使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。

タスク ***[InstallScript] ビューのスクリプト エディターで、関数呼び出しのヒントを表示するかどうかを指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"関数呼び出しのヒントを表示"** 設定で、適切なオプションを選択します:
   - **はい** – スクリプトに関数呼び出しを入力するときに、関数呼び出しのヒントが表示されます。デフォルトでは、これが設定されています。
   - **いいえ** – スクリプトに関数呼び出しを入力するときに、関数呼び出しのヒントは表示されません。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じて InstallScript エディター内で関数呼び出しのヒントを有効または無効にします。

関数呼び出しのヒントについての詳細は、「スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する」を参照してください。

スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する

関数呼び出しのヒントが有効な場合、[InstallScript] ビューでスクリプトに関数の呼び出しを入力しているときに、InstallScript 関数呼び出しのヒント (一種のツールヒント) が表示されます。関数呼び出しのヒントは、ビルトイン関数のパラメーター情報を表示します。その他、ビルトイン関数の説明や、入力しているパラメーターについての説明も表示します。

関数呼び出しのヒントを有効化する方法については、「[スクリプト エディター] 内で InstallScript 関数の呼び出しについてのヒントを有効または無効にする」を参照してください。

タスク ***関数の説明、およびその関数のパラメーターに関する詳細を表示するには、以下の手順に従います:***

1. 上記の説明のように、直接入力するまたは関数名の自動完了を利用してスクリプトに関数名を入力します。
2. 左かっこ "(" を関数名の後に入力します。すべてのパラメーターを含む、完全な関数の呼び出しを表示する関数呼び出しのヒントが表示されます。このヒントには、関数の説明と、次に入力する必要があるパラメーターの説明が含まれています。最初のパラメーターは、デフォルトで青色で表示されます。
3. 呼び出しのヒントに表示された通りにパラメーターを入力します。カンマを入力する度に、デフォルトで関数構文の次のパラメーターが青色で表示されます。
4. 呼び出しヒントを閉じるには、右かっこ ")" を入力します。または、ESC を押すか、呼び出しヒントの外にあるスクリプト エディター ペインをクリックします。

ヒント: *特定のビルトイン関数、定数、その他の InstallScript 用語に関するさらなる詳細を表示するには、挿入ポイントをその用語内に配置してから、F1 を押します。InstallShield ヘルプ ライブラリが開いて、その特定の用語に関する説明が表示されます。*

スクリプト エディターで構文ハイライト機能を有効または無効にする

InstallShield の様々なビューに含まれているスクリプト エディターでスクリプト ファイルを表示するときに、色属性を使ってキーワード、コメント、文字列、その他のスクリプト要素を識別するうかどうかを指定できます。各カテゴリは、識別しやすくするために異なる色で表示されます。

この機能は「*構文ハイライト*」と呼ばれ、コードを読み取りやすくして、コンテキストを改良します。さらに、たとえば予約語をユーザー定義識別子として使用したときなどに発生するエラーを避けるのに役立ちます。また、構文の色付けは、キーワードのスペルミス、文字列末尾の引用符の付け忘れ、コメント文字の付け忘れなど、スクリプト内の他のエラーの特定にも役立ちます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ構文ハイライトと色設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***構文ハイライト機能をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"構文ハイライト"** 設定で、適切なオプションを選択します:
   - **はい** – InstallShield は、**[スクリプト エディターのプロパティ]** ダイアログ ボックスにある **[色]** 領域で定義された構文ハイライトをすべてのスクリプト エディターで使用します。デフォルトでは、これが設定されています。
   - **いいえ** – InstallShield は、いずれのスクリプト エディターでも構文ハイライトを使用しません。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで構文ハイライト機能を有効または無効にします。

InstallShield では、スクリプト用語のカテゴリごとに使用される前景と背景の色を定義できます。詳しくは、「スクリプト エディターで構文ハイライトの色を変更する」をご覧ください。

スクリプト エディターで構文ハイライトの色を変更する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューといった、様々なスクリプト ビューで、異なるスクリプト要素を区別できるように色を変更することができます。コメント、関数、および文字列などのスクリプト要素について、前景の色 (テキストに使用される色) または背景色を変更できます。

この機能は「*構文ハイライト*」と呼ばれ、コードを読み取りやすくして、コンテキストを改良します。さらに、たとえば予約語をユーザー定義識別子として使用したときなどに発生するエラーを避けるのに役立ちます。また、構文の色付けは、キーワードのスペルミス、文字列末尾の引用符の付け忘れ、コメント文字の付け忘れなど、スクリプト内の他のエラーの特定にも役立ちます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ構文色の設定が、すべてのスクリプト エディターで使用されま*

す。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。

タスク ***スクリプト エディターで使用される構文ハイライトを変更するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **[色]** 領域で、構成するスクリプト要素の設定をクリックします。その設定で、[前景] と [背景] ボタンが表示されます。
3. 以下のいずれかを実行します。
   - [前景] ボタンまたは [背景] ボタンをクリックします。[**色**] ダイアログ ボックスが開いて、使用する色を選択できます。
   - 前景または背景に使用する色の RGB 値を入力します。たとえば、0; 0; 0 / 255; 255; 255 は、前景に黒 (RGB 値、0; 0; 0) を使って、背景には 白 (255; 255; 255) を使うことを示します。特定の領域に色を設定したくない場合は、ダッシュ (-;-;-) を使います。
4. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

スクリプトのハイライト機能が有効な場合、必要に応じて、スクリプト エディターでテキストを表示するのに使用される色が変更されます。詳細については、「スクリプト エディターで構文ハイライト機能を有効または無効にする」を参照してください。

この機能を最大限に活用するために、以下の点にご注意ください:

- 予約語をスペルミスして入力した場合、その予約語はエディターで認識されないため色付けされません。スクリプト内に色属性なしで表示された ″予約語″ があれば、それはおそらくスペルミスです。
- 文字列リテラルの色付けには、前後の引用符も含まれます。後ろの引用符を忘れた場合、文字列の色は行の終わりまで拡張されます。この場合、引用符の後に続くはずだったテキストは、文字列リテラルの一部であるかのように表示されます。
- 色による構文のハイライトにより、 /* で始まって */ で正しく閉じられなかったコメントを、容易に識別することができます。この場合、コメントに続くテキストは、すべてコメントの色属性で表示されます。
- 2 つのスラッシュ (//) で始まるコメントを含む行では、コメント文字以降のすべてのテキストが行末までコメントとして認識されます。
- [InstallScript] ビューで表示されるスクリプト エディターでは、ログ ファイルやレポート ファイルなどの .rul または .h 以外の拡張子を持つファイルに、構文色は使用されません。

スクリプト エディターでテキストを表示するためのフォントを変更する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューのスクリプト エディターで使用する色を変更することができます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで同じフォントが使用されます。*

[SQL スクリプト] ビューでフォントを変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューのフォントも変更されています。

タスク ***スクリプト エディターでテキストを表示するためのフォント、フォント スタイル、またはフォント サイズを変更するには、以下に従います。***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **[全般]** 領域で、**"フォント"** 設定をクリックしてから、この設定の省略記号ボタン (...) をクリックします。**[フォント]** ダイアログ ボックスが開きます。
3. 必要に応じてフォントの変更を指定します。
4. **[OK]** をクリックします。[**フォント**] ダイアログ ボックスが閉じます。
5. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターでテキストを表示するためのフォントを変更します。

スクリプト エディターで行番号を表示または非表示にする

InstallShield のスクリプト エディターの左マージンに、行番号を表示するかどうかを指定できます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ行番号を付ける設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***行番号をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"行番号を付ける"** 設定で、適切なオプションを選択します:
   - **はい** — InstallShield が、すべてのスクリプト エディターの左マージンに行番号を追加します。
   - **いいえ** — InstallShield は、いずれのスクリプト エディターの左マージンにも行番号を表示しません。デフォルトでは、これが設定されています。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで行番号を付ける設定を表示または非表示にします。

スクリプト エディター内で表示されているスクリプトの行番号へジャンプする

タスク ***スクリプト中の特定の行番号へ挿入ポイントを移動するには、以下の手順を実行します:***

1. **[編集]** メニューで **[行へジャンプ]** をクリックします。[**行へジャンプ**] ダイアログ ボックスが開きます。
2. **[行]** ボックスで、挿入ポイントを動かす行の番号を入力します。
3. **[OK]** をクリックします。

スクリプト エディターで自動インデント機能を有効または無効にする

InstallShield では、コードを書き込み中にスクリプト エディター内で自動インデント機能を使用するかどうかを指定できます。自動インデントを使うと、読みやすくなります。

自動インデント機能が有効な場合、InstallShield は前の行で使われたインデントに基づいて新しい行をインデントさせます。[スクリプト エディター] ペインで ENTER を押すと、挿入ポイントは新しい行で、前の行の最初の文字の真下に置かれます。新しい行のインデントを無効にするには、BACKSPACE を押して、インデントを削除します。

自動インデントが無効な場合、改行後の行はインデントされません。つまり挿入ポイントは新しい行の第 1 列に置かれます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、同じ自動インデント設定が、すべてのスクリプト エディターで使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***自動インデント機能をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"自動インデント"** 設定で、適切なオプションを選択します:
   - **はい** – InstallShield は、すべてのスクリプト エディターで自動インデントをサポートします。デフォルトでは、これが設定されています。
   - **いいえ** – InstallShield は、いずれのスクリプト エディターでも自動インデントを行いません。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで自動インデント機能を有効または無効にします。

スクリプト エディターでタブ幅を設定する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューのスクリプト エディターで使用するタブ幅を指定することができます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで同じタブ設定が使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***スクリプト エディターで入力されるタブの幅を指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"タブ幅"** 設定、スクリプトで使用する幅 (文字のスペース サイズの数の倍数) を入力します。デフォルト値は **4**です。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプトのタブ幅を調整します。

スクリプト エディターで構文の折りたたみ機能を有効または無効にする

InstallShield の様々なビューで、スクリプト エディターの構文の折りたたみ機能をサポートするかどうかを指定できます。構文の折りたたみ機能が有効な場合、スクリプトで展開可能または折りたたみ可能なブロックで始まるコードの各行の横のマージンに、プラス (+) またはマイナス (–) 記号が追加されます。プラス記号をクリックして非表示となっているコードを展開したり、マイナス記号をクリックしてコードを非表示にしたりできます。

構文の折りたたみ機能を使って、長いスクリプトを縮小することで、現在行っている作業に関連のあるコードに焦点を当てることができます。また、スクリプトの全体的な構造を確認するのに便利です。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで構文の折りたたみ設定が使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***構文の折りたたみ機能をスクリプト エディターで有効にするには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"構文の折りたたみ"** 設定で、適切なオプションを選択します:
   - **はい** – InstallShield は、すべてのスクリプト エディターで構文の折りたたみを行います。
   - **いいえ** – InstallShield は、いずれのスクリプト エディターでも構文の折りたたみを行いません。デフォルトでは、これが設定されています。
3. **[OK]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディターで構文の折りたたみ機能を有効または無効にします。

スクリプト エディターで空白スペースを表示または非表示にする

InstallShield では、様々なビューのスクリプト エディターで、スクリプト内の空白スペースが用いられている場所に、記号その他の印を表示するかどうかを指定することができます。たとえば、[はい] を選択すると、スクリプト内の各スペースは中黒 (·)、各タブは矢印記号で表示されます。また、空白スペースが有効な場合は、改行も表示されます。また、空白スペースが有効な場合は、改行も表示されます。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで空白スペース設定が使用されます。[SQL スクリプト] ビューで設定を変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューの設定も変更されています。*

タスク ***スクリプト エディターで空白スペースを表示するか非表示にするかを指定するには、以下の手順に従います:***

1. スクリプト エディター ペイン内を右クリックし、**[プロパティ]** をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが開きます。
2. **"空白スペースを表示"** 設定で、適切なオプションを選択します:
   - **はい** – InstallShield は、すべてのスクリプト エディターで空白スペース文字と記号を表示します。
   - **いいえ** – InstallShield は、いずれのスクリプト エディターでも空白スペース文字と記号を表示しません。デフォルトでは、これが設定されています。
3. [OK] をクリックします。**[スクリプト エディターのプロパティ]** ダイアログ ボックスが閉じます。

InstallShield は、必要に応じてスクリプト エディター内の空白スペース文字と記号を表示または非表示にします。

スクリプト エディター内でテキストをドラッグ アンド ドロップする

InstallShield のスクリプト エディターでは、テキスト編集時にドラッグ アンド ドロップをサポートするため、選択したテキストをコード内のある場所から別の場所に移動またはコピーすることができます。

テキストの移動

タスク ***テキストを移動するには、以下の手順を実行します。***

1. 移動するテキストを選択します。
2. 選択したテキスト上にマウスポインターを置きます。
3. 左マウスボタンを押したままにします。
4. マウスポインターの下に小さな四角が表示されたら、スクリプト内での選択テキストの移動先にポインターを動かします。ポインターを動かしている間は、左マウスボタンを押したままにしてください。
5. ポインターをテキストの移動先まで動かしたら、マウスボタンを離します。

テキストのコピー

タスク ***ドラッグアンド ドロップを使って選択したテキストをコピーするには、次の手順を実行します。***

1. 移動するテキストを選択します。
2. 選択したテキスト上にマウスポインターを置きます。
3. CTRL を押して、左マウスボタンを押したままにします。
4. マウスポインターの下に小さな四角とプラス記号が表示されたら、スクリプト内の選択テキストのコピー先にポインターを動かします。ポインターを動かしている間は、CTRL と左マウスボタンを押したままにしてください。
5. ポインターをテキストの移動先まで動かしたら、マウスボタンを離します。

[スクリプト エディタ] 内からスクリプト ファイルを印刷する

InstallShield では、たとえば [InstallScript] ビューや [SQL スクリプト] ビューなどのスクリプト エディターで表示されるスクリプトを印刷することができます。

タスク ***アクティブな スクリプト エディター ペインに表示されているスクリプトを印刷するには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[印刷]** をクリックします。このコマンドは、挿入ポイントがスクリプト エディター ペインにある場合にみ有効です。[**印刷**] ダイアログ ボックスが開きます。
2. どちらかひとつのオプションを設定するには、以下の手順を実行します。
3. **[OK]** をクリックします。

スクリプト エディターのキーボード ショートカット

InstallShield のスクリプト エディターは、以下のキーボード ショートカットをサポートします。

テーブル 8 · スクリプト エディターのキーボード ショートカット

キーボード ショートカット	説明
CTRL+C CTRL+INSERT	選択されたテキストをクリップボードにコピーします。
CTRL+X SHIFT+DELETE	選択した文字列を削除して、クリップボードに置きます。
Ctrl+F	指定した文字列を検索します。

テーブル 8 • スクリプト エディターのキーボード ショートカット (続き.)

キーボード ショートカット	説明
CTRL+H	テキストまたはその他の文字を検索/置換できる [置換] ダイアログ ボックスを表示します。
CTRL+G	指定した行に移動します。
TAB	選択された文字列をタブ 1 つ分右インデントします。
CTRL+U	選択した文字列をすべて小文字に変更します。
CTRL+V SHIFT+INSERT	クリップボードの内容をカーソル位置に挿入します。
CTRL+Y	取り消した操作をやり直します。
CTRL+A	文書の全文字列を選択します。
CTRL+Z ALT+BACKSPACE	直前の操作を元に戻します。
SHIFT+TAB	選択した文字列のインデントを戻します。
CTRL+SHIFT+U	選択した文字列をすべて大文字に変更します。
CTRL+M	スクリプト エディターの幅を最大化、または以前の幅に戻します。
CTRL+I	[InstallScript] ビューに表示されているスクリプトの挿入ポイントで、関数の呼び出しを追加できる関数ウィザードを開きます。
ALT+K	現在挿入ポイントが置かれているスクリプト行で、ブックマークの追加と削除操作が切り替わります。
CTRL+K	次のブックマークを含むスクリプト行に移動します。
CTRL+SHIFT+K	前のブックマークを含むスクリプト行に移動します。

InstallShield の詳細設定を構成する

InstallShield プログラム ファイルの中に Settings.xml というファイルがあります。このファイルで、InstallShield のためのマシン全体に関する詳細設定が一部含まれています。InstallShield をインストールしたとき、使用している InstallShield の言語に応じて Settings.xml が次の場所の 1 つにインストールされます。

- **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
- **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411

Standalone Build の Settings.xml ファイルは、次の場所のいずれかにインストールされています:

- **英語** – *Standalone Build Program Files フォルダー*¥Support¥0409
- **日本語** – *Standalone Build Program Files フォルダー*¥Support¥0411

通常、Settings.xml ファイルの変更は推奨されません。ただし、あるケースでは、このファイルでの変更が必要になる場合があります。このセクションでは、Settings.xml の変更が必要にある場合のシナリオがいくつか説明されています:

- デジタル署名のタイムスタンプ サーバーを変更する
- Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する
- .cab ファイルの最大サイズを構成する
- 仮想メディア メモリの最大サイズを構成する
- Standalone Build のポータブル実行可能ファイルの一覧を変更する
- XML エンコード オプションのサポートを追加する
- すべての仮想パッケージをビルドする場所を指定する
- セットアップのデフォルト Clone_wait パラメーターを変更する
- デジタル署名のプラットフォーム アーキテクチャを構成する
- ランタイム中の MSI の検証を構成する
- イベント設定コマンドの実行失敗に対するエラー表示の管理

注意 - *Settings.xml ファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

デジタル署名のタイムスタンプ サーバーを変更する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*

リリースのデジタル署名情報を指定すると、InstallShield はビルド中に、DigiCert のサーバー (**http://timestamp.digicert.com**) をデフォルトのタイムスタンプ サーバーとして使用します。InstallShield には、そのデフォルト サーバーを異なるタイム スタンプ サーバーに変更できるマシン全体の設定があります。この設定を使って、タイムスタンプを無効にすることもできます。

InstallShield 2019 R2 以降では、連続するデジタル署名の間に遅延を設定することができます。この機能は、タイムスタンプ サーバーが連続する署名要求の処理に失敗した場合のみ必要になります。

Settings.xml の <DevStudio/Build> ノードの下にある <DelayBetweenSigning default="1500" /> ノードで、ミリ秒単位で指定する必要があります。

***注意・** 次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***デジタル署名のタイムスタンプ サーバーを構成するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <DigitalSignature> 要素を検索します。要素は次のように表示されています:

   <DigitalSignature Timestamp="http://timestamp.verisign.com/scripts/timstamp.dll" />
6. 別のタイムスタンプ サーバーでオーバーライドするには、Timestamp 属性の値を適切な URL に設定します。

   タイムスタンプを無効にするには、Timestamp 属性の値を空白にします:

   <DigitalSignature Timestamp="" />

   ***メモ・** タイムスタンプを無効にすると、デジタル署名の有効期間に影響します。*
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

InstallShield 2019 以降では、タイムスタンプ サーバーが SHA-2 サーバーに設定されています:

- <DigitalSignature Timestamp="http://sha256timestamp.ws.symantec.com/sha256/timestamp" />

デジタル署名情報を含むリリースをビルドするたびに、構成された設定に基づいて InstallShield によってタイムスタンプが設定されます。

***ヒント・** Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている Settings.xml ファイルを更新します。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

この情報は、1 つまたは複数の機能に関連付けられているファイルのみが .cab ファイルに圧縮されるカスタム圧縮には適用しません。

InstallShield には、ビルド時に Setup.exe ファイルと ISSetup.dll ファイルにストリームされるファイルに使用する圧縮レベルを指定することができる設定が含まれています (マシン全体に適用されます)。以下は、Setup.exe ファイルと ISSetup.dll ファイルにストリームすることができるファイルの例です:

- すべての製品ファイル (すべてのファイルが Setup.exe セットアップランチャーに圧縮されるリリースの場合)
- "Setup.exe から抽出" の場所がある InstallShield 前提条件インストール
- "Setup.exe から抽出" の場所がある .NET Framework インストール
- "Setup.exe から抽出" の場所がある Windows Installer インストール

InstallShield では、ファイルが Setup.exe ファイルまたは ISSetup.dll ファイルにストリームされるとき、圧縮から除外するファイルを指定することができます (ワイルドカード文字の使用可)。

注意・ *次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***ストリームされたファイルの圧縮設定を構成するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <StreamCompression> 要素を検索します。要素は次のように表示されています:

<StreamCompression exclude="*.CAB" compressionlevel="-1"/>

6. ファイルが Setup.exe ファイルまたは ISSetup.dll ファイルにストリームされるとき、特定のファイルまたはファイル種類を圧縮から除外する場合、除外属性の値をこれらのファイルの名前に設定します。次の事項に注意してください。

   - 複数のファイルを指定する場合、各ファイル名をカンマで区切ります。
   - ワイルドカード文字の指定には、アスタリスク (*) を使用します。

   たとえば、.cab ファイル、.exe ファイル、および test.txt という名前のファイルを圧縮から除外するように指定する場合、除外属性の値を次のように設定します:

   <StreamCompression exclude="*.CAB,*EXE,test.txt" compressionlevel="-1"/>

   .cab ファイルは圧縮ファイルであるため、デフォルトの値は ***.CAB** に設定されています。

7. 以下のいずれかを実行します。

   - 圧縮ファイルのサイズと実行時に圧縮ファイルを展開するために必要な時間のバランスを考慮に入れた圧縮レベルを使用する場合、compressionlevel 属性の値を **-1** に設定します。これがデフォルトの値です。
   - 特定の圧縮レベルを指定する場合、compressionlevel 属性の値に 0 から 9 の間の数値を指定します。0 は圧縮なしを意味し、9 が最大圧縮を意味します。

     一般的に、0 から 9 の値を指定した場合、指定した値が大きいほど、圧縮されたファイルのサイズは小さくなり、実行時にファイルを展開するときの時間が長くなります。

8. Settings.xml ファイルを保存します。

9. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

アプリケーション プロジェクト タイプの 1 つに圧縮されたリリースをビルドしたとき、構成した設定に従って、ファイルが Setup.exe ファイルと ISSetup.dll ファイルにストリームされます。

ヒント: *Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている* Settings.xml *ファイルを更新します。*Settings.xml *は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

.cab ファイルの最大サイズを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

また、この機能は、すべてのファイルが単一ファイルの .msi パッケージまたは Setup.exe *セットアップランチャーに埋め込まれている圧縮済みネットワーク イメージ リリースをビルドしている場合のみ適用します。*

この情報は、1 つまたは複数の機能に関連付けられているファイルのみが .cab ファイルに圧縮されるカスタム圧縮には適用しません。

.cab ファイルには、いくつかの制限事項があります。たとえば、単一 .cab ファイルの最大サイズは 2 GB に設定されています。また、サイズの大きい .cab ファイルを署名しようしたとき、およびサイズの大きい署名済み .cab ファイルのデジタル署名を検証しようとしたときにトラブルが生じた経験があるユーザーもいるかもしれません。

これらの制限事項を回避するため、InstallShield では、圧縮ネットワーク イメージ リリースに対してビルドされる各 .cab ファイルの最大サイズを指定することができます (設定はマシン全体に適用されます)。InstallShield で、リリースの .cab ファイルを作成しているとき、構成した .cab ファイルのしきい値に達すると、データが 2 つ以上の.cab ファイルに分割され、マルチパートの .cab ファイルが作成されます。InstallShield でマルチパート .cab ファイルを作成しない場合、単一の .cab ファイルにデータを格納するように構成できます。

注意・*次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***InstallShield でマルチパート .cab ファイルを作成するかどうか、また .cab ファイルの最大サイズを指定する場合、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <CompressedNetworkCABSize> 要素を検索します。要素は次のように表示されています:

   <CompressedNetworkCABSize default="600" />
6. 以下のいずれかを実行します。
   - .cab ファイルの最大サイズを設定する場合、デフォルト属性の値としてサイズを MB で入力します。上記の例では、最大サイズは 600 に設定されています。.cab ファイルの最大サイズは 2 GB (2048 MB) に設定されているため、値は 2048 以下でなければなりません。

     デフォルト値は **600**です。
   - InstallShield でマルチパート .cab ファイルを作成しない場合、デフォルト属性の値を **-1** に設定します。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、

不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

アプリケーション プロジェクト タイプの 1 つに圧縮されたネットワーク イメージ リリースをビルドしたとき、Settings.xml ファイルで構成した要件に従って、.cab ファイルが作成されます。Settings.xml ファイルで指定した値に応じて、.cab ファイルが .msi パッケージのMedia テーブルに表示されます。

***メモ・** .cab ファイル サイズを設定する前にメディア サイズをチェックする。異なる配布メディアの種類についての情報は、「[メディアの種類] パネル」を参照してください。*

***ヒント・** Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている Settings.xml ファイルを更新します。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

仮想メディア メモリの最大サイズを構成する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

仮想メディア メモリ タイプには制限事項があります。たとえば、仮想メディア メモリの最大サイズは 512MB に設定されています。

***注意・** 次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***InstallShield が仮想メディア メモリの最大サイズを指定するかどうかを指定するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。

4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <CompressedNetworkCABSize> 要素を検索します。要素は次のように表示されています:

   <CompressedNetworkCABSize default="600"/>
6. 以下のいずれかを実行します。
   - .cab ファイルの最大サイズを設定する場合、デフォルト属性の値としてサイズを MB で入力します。上記の例では、最大サイズは 600 に設定されています。.cab ファイルの最大サイズは 2 GB (2048 MB) に設定されているため、値は 2048 以下でなければなりません。

     デフォルト値は **600**です。
   - InstallShield でマルチパート .cab ファイルを作成しない場合、デフォルト属性の値を **-1** に設定します。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

アプリケーション プロジェクト タイプの 1 つに圧縮されたネットワーク イメージ リリースをビルドしたとき、Settings.xml ファイルで構成した要件に従って、.cab ファイルが作成されます。Settings.xml ファイルで指定した値に応じて、.cab ファイルが .msi パッケージのMedia テーブルに表示されます。

ヒント: *Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている* Settings.xml *ファイルを更新します。*Settings.xml *は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:*

- **英語** — *Standalone Build Program Files フォルダー¥Support¥0409*
- **日本語** — *Standalone Build Program Files フォルダー¥Support¥0411*

Standalone Build のポータブル実行可能ファイルの一覧を変更する

InstallShield にある [オプション] ダイアログ ボックスの [ファイルの拡張子] タブで、InstallShield でポータブル実行可能 (PE) ファイルと見なすファイル拡張子を変更することができます。InstallShield は、この一覧を使って、ダイナミック リンクがあるファイルのコンポーネントを、コンポーネント作成時のベスト プラクティスに沿ってどう作成するかを判別します。

Standalone Build を使ってリリースをビルドする場合、[オプション] ダイアログ ボックスは使用できません。ただし、Standalone Build が PE ファイルと見なすファイル拡張子の一覧を変更する場合、InstallShield Standalone Build Program Files フォルダーのサブフォルダーにインストールされているファイルの 1 つを変更することはできます。以下は、その手順の説明です。

注意: *次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***Standalone Build が PE ファイルと見なすファイル拡張子の一覧を変更するには、以下の手順に従います:***

1. Standalone Build を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語** – *Standalone Build Program Files フォルダー*¥Support¥0409
   - **日本語** – *Standalone Build Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. PEFileExtensions 要素を検索します。要素は次のように表示されています:

   <PEFileExtensions default="EXE|DLL|OCX|VXD|CHM|HLB|TLB|AX">
6. 必要に応じて、デフォルト属性のファイル拡張子の一覧に変更を加えます。各ファイル拡張子は垂直バー (|) で区切られています。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。ファイル内にある主な要素は、縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

次回 Standalone Build を使用するとき、ファイル拡張子の更新された一覧が使用されます。

XML エンコード オプションのサポートを追加する

InstallShield では、XML ファイルに使用するエンコードの種類を指定することができます。この設定は、[XML ファイルの変更] ビューで XML ファイルを選択したとき [詳細] タブで表示されます。使用するエンコードの種類が提供されているエンコード オプションの一覧にない場合、IInstallShield Program Files フォルダーのサブフォルダーにインストールされているファイルの 1 つを変更して、追加のオプションに追加することができます。MSXML でサポートされているエンコードはすべて追加することができます。以下は、その手順の説明です。

注意 – 次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。

タスク ***[XML ファイルの変更] ビューの [詳細] タブで追加のエンコード オプションを [エンコード] 一覧に追加するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:

- **英語** — *InstallShield Program Files フォルダー*¥System¥0409
- **日本語**—*InstallShield Program Files フォルダー*¥System¥0411

3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. ISXML 要素とその子要素を検索します。要素は次のように表示されています:

```
<ISXML>
 <Encodings>
  <Encoding>WINDOWS-1250</Encoding>
  <Encoding>WINDOWS-1251</Encoding>
  <Encoding>WINDOWS-1252</Encoding>
  <Encoding>WINDOWS-1253</Encoding>
  <Encoding>WINDOWS-1254</Encoding>
  <Encoding>WINDOWS-1255</Encoding>
  <Encoding>WINDOWS-1256</Encoding>
  <Encoding>WINDOWS-1257</Encoding>
  <Encoding>WINDOWS-1258</Encoding>
  <Encoding>ISO-8859-1</Encoding>
  <Encoding>ISO-8859-2</Encoding>
  <Encoding>ISO-8859-3</Encoding>
  <Encoding>ISO-8859-4</Encoding>
  <Encoding>ISO-8859-5</Encoding>
  <Encoding>ISO-8859-6</Encoding>
  <Encoding>ISO-8859-7</Encoding>
  <Encoding>ISO-8859-8</Encoding>
  <Encoding>ISO-8859-9</Encoding>
  <Encoding>US-ASCII</Encoding>
  <Encoding>UNICODE-1-1-UTF-8</Encoding>
  <Encoding>UNICODE-2-0-UTF-16</Encoding>
  <Encoding>UNICODE-2-0-UTF-8</Encoding>
  <Encoding>ISO-10646-UCS-2</Encoding>
  <Encoding>UCS-4</Encoding>
  <Encoding>UCS-2</Encoding>
  <Encoding>UTF-16</Encoding>
  <Encoding Default="true">UTF-8</Encoding>
 </Encodings>
</ISXML>
```

6. 始めと終わりの Encodings タグの間に、次のような新しい行を追加します:

   <Encoding>Type_of_Encoding</Encoding>

   Type_of_Encoding がある箇所は、使用可能にするエンコードを示します。ここには、InstallShield で XML 文書のエンコード属性に使用する値が入ります。
7. Settings.xml ファイルを保存します。
8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。<ISXML> 要素、および <Encodings> 要素は縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

次回 InstallShield で [XML ファイルの変更] ビューを開いたとき、使用可能なオプションの 1 つとして追加したエンコードの種類が [XML ファイルの変更] ビューにある XML ファイルの [詳細] タブで **[新しいファイルに使用するエンコード]** 一覧に表示されます。

すべての仮想パッケージをビルドする場所を指定する

エディション: *InstallShield Premier Edition には、仮想化サポートが含まれています。*

InstallShield が .msi パッケージを変換するとき、デフォルトで、その .msi ファイルが含まれるフォルダーのサブフォルダーに仮想パッケージを保存します。場合によって、すべての仮想パッケージを生成する別の場所を指定したい場合が想定されます。たとえば、変換する .msi パッケージが読み取り専用の場所にある場合、デフォルトの仮想パッケージの場所を既存の書き込み可能な場所でオーバーライドする必要があります。

InstallShield には、すべての仮想パッケージをビルドする既存の書き込み可能な場所を指定できる、マシン全体に適用される設定が含まれています。このグローバル設定を構成する方法は、以下の通りです。

注意: *次の手順では、InstallShield と共にインストールされている* Settings.xml *ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

タスク ***すべての仮想パッケージをビルドする場所を指定するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語** — *InstallShield Program Files フォルダー*¥System¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥System¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <Virtualization> 要素とその子要素を検索します。要素は次のように表示されています:

   ```
   <Virtualization>
     <!— Instructions on how to specify a global path —>
     <GlobalBuildRedirectFolder></GlobalBuildRedirectFolder>
   </Virtualization>
   ```

6. <GlobalBuildRedirectFolder>要素のテキスト コンテンツとしてグローバル パスを入力します。たとえば、¥¥NetworkFolder¥VirtualPackages のサブフォルダーに仮想パッケージを作成するには、次を使用します:

   <GlobalBuildRedirectFolder>¥¥NetworkFolder¥VirtualPackages</GlobalBuildRedirectFolder>

   パスは既存のパスでなくてはなりません。パスを引用符で囲まないでください。
7. Settings.xml ファイルを保存します。

8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Virtualization> 要素は縮小、展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。

InstallShield で .msi パッケージを仮想パッケージに変換するときは常に、指定されたパスが使用されます。

カスタム仮想アプリケーションの作成

セットアップのデフォルト Clone_wait パラメーターを変更する

***注意・** 次の手順では、InstallShield と共にインストールされている Settings.xml ファイルを変更する必要があります。このファイルには重要なデータが含まれているため、ファイルが間違って編集された場合、InstallShield が正しく動作しなくなることがあります。このファイルを編集する場合、十分な注意が必要です。*

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、settings.xml に Clone_wait パラメーターの新しい設定が追加されています。デフォルトでは、元のセットアップがインストール処理の完了を待機するように設定されています。埋め込み Clone_wait オプションなしでセットアップ ランチャーを生成したい場合、この設定を変更できます。

エンド ユーザーは、コマンドライン パラメーターを使ってこの設定をオーバーライドすることができます。

タスク ***セットアップのデフォルト Clone_wait パラメーターを変更するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを見つけます。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <Clone_wait > 要素とその子要素を検索します。デフォルトでは次のように表示されています:

   <Clone_wait default="Yes"/>

6. デフォルト Clone_wait パラメーターを [いいえ] に変更します。

   <Clone_wait default="No"/>

7. Settings.xml ファイルを保存します。

8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。コードのエラーを確認します。

デジタル署名のプラットフォーム アーキテクチャを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*

InstallShield 2025 では、ファイルのデジタル署名に使用するプラットフォーム アーキテクチャを選択できます。

タスク ***デジタル署名のプラットフォーム アーキテクチャを構成するには、以下の手順に従います:***

1. **InstallShield** を閉じます。

2. InstallShield と共にインストールされている **Settings.xml** ファイルを検出します。**Settings.xml** は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:

   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411

3. 後で元のバージョンに戻す必要があるときのために、**Settings.xml** ファイルのバックアップ コピーを作成しておきます。

4. テキスト エディターまたは XML ファイル エディターを使って、**Settings.xml** ファイルを開きます。

   メモ: *ファイルが編集のための管理者モードで開きます。*

5. **<DigitalSignature>**. タグを検索します。実際のタグは **<DigitalSignature Platform="X86"/>** です。

6. 異なるプラットフォーム アーキテクチャを使用する場合、"プラットフォーム" を X86 または X64 のどちらかに設定します。
たとえば、
   - 32ビット署名の場合: <DigitalSignature Platform="X86" />
   - 64ビット署名の場合: <DigitalSignature Platform="X64" />

   ***メモ・** 無効なエントリの場合、自動的に 32 ビット 署名が使用されます。*

7. **Settings.xml** ファイルを保存します。

***メモ・** 32 ビットのオペレーティング システム上で 64 ビット署名を使用することはできません。*

ランタイム中の MSI の検証を構成する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield 2025 では、ランタイム中に行なわれるデジタル署名の検証を有効にして、インストーラーの整合性を保つことができます。

タスク ***デジタル署名の MSI の検証を有効化するには、次の手順に従います:***

1. **InstallShield** を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. 後で元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成しておきます。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。

   ***メモ・** ファイルが編集のための管理者モードで開きます。*

5. **<DigitalSignature ValidateSetupAndMSISignatures>** タグを検索します。実際のタグは **<DigitalSignature ValidateSetupAndMSISignatures="no" />です。**
6. MSI の検証を有効化するには、この値を [はい] に設定します。

7. Settings.xml ファイルを保存します。

イベント設定コマンドの実行失敗に対するエラー表示の管理

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield を使用して、**[イベント]** タブの **[ビルド前のイベント]**、**[ファイルの準備イベント]**、**[圧縮前のイベント]**、または **[ビルド後のイベント]** 設定で指定されたコマンドがビルド時間中に実行されなかった場合、インストーラーが正しくビルドされた後にエラーメッセージを表示するかどうかを管理または制御できます。

デフォルトでは、これらの失敗は、ビルドが成功した後にエラーメッセージをトリガーしません。Settings.xml ファイルを変更することで、このような場合にエラーメッセージを表示するように InstallShield を構成できます。このような失敗に対するエラーメッセージ表示を有効にするには、以下の手順に従ってください。

注意: *Settings.xml ファイルには、重要な構成データが含まれています。変更が正しくないと、InstallShield が動作しなくなる可能性があります。編集する前に、必ずこのファイルをバックアップしてください。*

タスク

コマンド ([イベント] タブの必要な設定で指定します) の実行エラー メッセージの表示を有効化にするには、次の手順に従います:

1. InstallShield を閉じます。
2. InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語バージョンに応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
3. あとで元のバージョンに戻す必要があるときのために、Settings.xml ファイルのバックアップ コピーを作成します。
4. テキスト エディターまたは XML ファイル エディターを使って、Settings.xml ファイルを開きます。
5. <TreatBuildEventFailuresAsErrors> 要素とその子要素を検索します。デフォルトでは次のように表示されています:

   <TreatBuildEventFailuresAsErrors default="no"/>
6. デフォルト TreatBuildEventFailuresAsErrors パラメーターを [はい] に変更します。

   <TreatBuildEventFailuresAsErrors default="yes"/>
7. Settings.xml ファイルを保存します。

8. XML コードが適切に構成されていることを確認してください。不適切なコードは、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで Settings.xml ファイルを開いて、不適切に書かれた XML コードを確認することができます。コードのエラーを確認します。

リリースの配布用の仮想マシン設定を共有する

エディション: *仮想マシンへのリリース配布機能は、InstallShield Premier Edition でご利用いただけます。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態となるようにプロジェクト内のリリースを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。

[リリース] ビューで選択されているリリースの [イベント] タブにある "構成" 設定を使用して VM の詳細を構成すると、InstallShield によって 指定されたデータが VMConfigurations.xml という名前のファイルに書き込まれます。この VMConfigurations.xml ファイルは、マシン全体に適用するファイルで、一度構成を行うと他のInstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルを Standalone Build で使用することも可能です。

このグローバル ファイルを共有可能にする方法は、以下の通りです:

タスク

カスタマイズ済みの VM 構成設定ファイルを、InstallShield または スタントアロン ビルドがインストールされている別のマシンと共有するには、以下の手順に従います:

1. 開発マシン上で、プロジェクト内の 1 つのリリースの VM 設定を構成します。詳しくは、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」をご覧ください。
2. InstallShield を閉じます。
3. InstallShield と共にインストールされている VMConfigurations.xml ファイルを見つけて、別のマシンにこれをコピーします。VMConfigurations.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:
   - **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
   - **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411
4. 元のバージョンに復元する必要がある場合を想定して、別のマシン上の VMConfigurations.xml のバックアップ コピーを作成しておき、使用中の開発マシンからのバージョンを使って、インストール済みのバージョンを上書きします。

Standalone Build を使ってリリースをビルドする場合、Standalone Build と共にインストールされている VMConfigurations.xml ファイルを更新します。VMConfigurations.xml は、使用している InstallShield の言語に応じて、次のいずれかの場所にインストールされています:

- **英語** – *Standalone Build Program Files フォルダー*¥Support¥0409
- **日本語** – *Standalone Build Program Files フォルダー*¥Support¥0411

ヒント: *テキスト エディターを使って XML ファイルを手動で変更する場合、XML コードが正しく作成されていることを確認してください。そうでない場合、InstallShield で問題が発生する可能性があります。ほとんどの場合、インターネット エクスプローラーで* VMConfigurations.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。〈VMConfigurations〉 要素は折りたたみ/展開が可能です。それが不可能な場合、コードにエラーが無いか確認してください。*

以前のバージョンの InstallShield からアップグレードする

InstallShield 2025 を使い始めるとき、以前のバージョンまたは別の InstallShield 製品を使って作成したインストール プロジェクトをアップグレードすることができます。このセクションでは、これらのアップグレードについて説明します。

- InstallShield 2014 以前のプロジェクトをアップグレードする
- InstallShield 2013 以前のプロジェクトをアップグレードする
- InstallShield 2012 Spring 以前のプロジェクトをアップグレードする
- InstallShield 2012 以前のプロジェクトをアップグレードする
- InstallShield 2011 以前のプロジェクトをアップグレードする
- InstallShield 2010 以前のプロジェクトをアップグレードする
- InstallShield 2009 以前のプロジェクトをアップグレードする
- InstallShield 2008 以前のプロジェクトをアップグレードする
- プロジェクトを InstallShield 12 以前からアップグレードする
- InstallShield 11.5 以前のプロジェクトをアップグレードする
- InstallShield Express で作成したプロジェクトをアップグレードする
- InstallShield Professional で作成したプロジェクトをアップグレードする
- InstallShield—Windows Installer Edition で作成したプロジェクトをアップグレードする

InstallShield 2014 以前のプロジェクトをアップグレードする

以下は、InstallShield 2014 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2014 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.775 (.ism プロジェクトの場合) または .2014 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.775 または .2014 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2014 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

ターゲット システムとしてサポートされている Windows のバージョン リストに関する変更

今回より、Windows XP SP3 および Windows Server 2003 SP2 が、InstallShield で作成されたインストールを実行するターゲット システムで必要な Windows の最低バージョンです。これはすべてのプロジェクト タイプに適用します。

.spc および .pvk ファイルを使ったデジタル署名サポートの削除

InstallShield で、ビルド時に .spc および .pvk ファイルにデジタル署名を行うサポートは、今後使用できません。

InstallShield 2014 以前で、リリースまたはパッチに .spc および .pvk ファイルを使って実行時にデジタル署名を行うように構成済みで、そのプロジェクトを InstallShield 2025 で開こうとすると、アップグレード警告 -6048 (リリース)、-6049 (パッチ)、または -6050 (QuickPatch プロジェクト) が表示されます。この警告は、アップグレード中に InstallShield によって .pvk ファイルおよび関連パスワードがプロジェクトから削除されることを説明します。

InstallShield 2025 でリリースまたはパッチを正しくビルドするためには、そのリリースまたはパッチ構成から .spc の参照を削除する必要があります。これを、.pfx 証明書または証明書ストアにある証明書への参照と入れ換えることができます。詳しくは、次を参照してください:

- デジタル署名とセキュリティ
- ビルド時にリリースとそのファイルにデジタル署名を行う
- パッチ パッケージに署名する
- QuickPatch パッケージに署名する

.spc の参照を削除せずにリリースまたはパッチのビルドを試みると、.spc ファイルの削除が必要であることを通知するビルド エラー -7347 が表示されます。

.spc ファイルおよび .pvk ファイルを .pfx ファイルに変換する方法については、「デジタル署名とセキュリティ」を参照してください。

デジタル署名サポートにおけるオートメーション インターフェイスの変更

ISWiRelease オブジェクトは、今回より、次の読み書きプロパティをサポートしません:

- CorrespondingPrivateKey
- SoftwarePublishingCredentials

これらのプロパティを呼び出すと、エラーが発生します。これらのプロパティは、読み書き文字列プロパティ DigitalCertificateInfo に置き換えられています。

詳細については、「ISWiRelease オブジェクト」を参照してください。

InstallShield インストールからの SignTool.exe および Signcode.exe の削除

InstallShield をインストールしたとき、今回より SignTool.exe および Signcode.exe は開発マシンにインストールされません。手動でファイルにデジタル署名を行う場合、SignTool.exe の使用を考慮してください。SignTool.exe は、Microsoft Windows Software Development Kit (SDK) に含まれていて、Visual Studio と共にインストールされます。

基本の MSI インストールおよび InstallScript MSI インストールにおける PowerShell サポートの変更

基本の MSI インストールおよび InstallScript MSI インストールの PowerShell カスタム アクション サポートが改訂されました。サポートは Windows Installer プロパティ IS_PS_EXECUTIONPOLICY を使って、ターゲット システム上で PowerShell カスタム アクションを実行するのに使用する PowerShell 実行ポリシーの名前を示すことがなくなりました。実行時にこのプロパティを設定しても、インストールに何ら影響はありません。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける .msi パッケージのデフォルトの対象条件の変更点

アドバンスト UI およびスイート/アドバンスト UI プロジェクト内の共有パッケージをサポートできるように、InstallShield では以前必要だった .msi パッケージに対する MSI パッケージの対象条件が不要となりました。今回より、プラットフォームなどの実行時環境の要件を確認する .msi パッケージの対象条件の使用を制限できます。

InstallShield 2025 でアドバンスト UI またはスイート/アドバンスト UI プロジェクトに .msi パッケージを追加したとき、デフォルトでそのパッケージに MSI パッケージの対象条件が作成されることがなくなりました。この動作は新しい InstallShield 2025 プロジェクト、および InstallShield 2014 以前で作成してから InstallShield 2025 にアップグレードしたプロジェクトに適用します。

さらに、.msi パッケージに古いデフォルトの MSI パッケージの対象条件を含む InstallShield 2014 以前のプロジェクトを InstallShield 2025 にアップグレードした場合、InstallShield はパッケージの "対象条件" 設定から条件の不要な部分を削除します。

一部の状況では、InstallShield 2014 以前のデフォルト MSI パッケージの対象条件をカスタマイズした場合、InstallShield 2025 はアップグレード中にそれを削除することがありますが、そのまま残す場合もあります。実際のアップグレード動作はカスタマイズによって異なります。たとえば、単純にプラットフォーム条件を追加した場合、InstallShield は条件の元の MSI パッケージ部分を削除し、プラットフォーム条件のみを残します。ただし、カスタマイズが複雑な場合、InstallShield は条件をそのまま残します。

InstallShield 2025 にアップグレードした後、.msi パッケージの対象条件が予定通りに構成されていることを確認することが推奨されます。

カスタマイズされた、または編集済みのデフォルト MSI パッケージの対象条件がある場合、InstallShield はアップグレード時にカスタム条件をそのまま残します。.msi パッケージの対象条件が予定通りに構成されていることを確認することが推奨されます。

以前、.msi パッケージの "対象条件" 設定のデフォルトの MSI パッケージ条件は、パッケージの独自の製品コードと製品バージョンのプレースホルダーとして、条件の "製品コード" および "製品バージョン" 設定にアステリスク (*) を使用しました。

アドバンスト UI およびスイート/アドバンスト UI インストールのウィザード インターフェイスで使用されているアンパサンドの解釈に関する変更

アドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード インターフェイスの特定の領域で、アンパサンドの解釈が更新されました。

新しい InstallShield 2025 プロジェクトの場合、ならびに InstallShield 2014 以前で作成したプロジェクトを InstallShield 2025 にアップグレードした場合、インストールでウィザード インターフェイスの次の任意の領域でアンパサンド (&) を使用すると、今回より、インストールはアンパサンドをリテラル文字として表示します。以前はアンパサンドがキーボード ショートカットの先頭の文字として解釈されました。この変更はまた、これらの文字列にアンパサンドを含む値に解決するプロパティが含まれている場合にも適用します。

- ウィザード ページまたは 2 番目のウィンドウのヘッダー領域にある文字列 – これは [ウィザード インターフェイス] ビューでウィザード ページまたはウィンドウの "タイトル" 設定で構成されます。
- ウィザード ページのキャプション バー – これは [ウィザード インターフェイス] ビューで [ウィザード ページ] ノードの "ウィザード キャプション" 設定で構成されます。
- コマンド リンク コントロールの [補足説明] 領域 – これはウィザード ページまたはウィンドウの "メモ" 設定で構成されます。
- イメージ コントロールの代替テキスト – これはウィザード ページまたはウィンドウにあるイメージ コントロールの "代替テキスト" 設定で構成されます。

さらに、ほとんどのラベル コントロールでは、今回より、"スタイル" 設定の下にある SS_NOPREFIX サブ設定のデフォルトの選択が True となりました。以前はデフォルトで False が選択されていました。このサブ設定が True 値に設定されていることで、これらのコントロールの文字列エントリに含まれるアンパサンドが誤ってキーボード ショートカットとして解釈されないようにし、コントロールの文字列に含まれるアンパサンドをウィザード インターフェイスのアンパサンドとして正確に表示します。

これらの SS_NOPREFIX の変更は、新しい InstallShield 2025 プロジェクトで使用可能なビルトイン デフォルトおよび定義済みのウィザード ページとウィンドウに適用します。InstallShield 2014 以前のプロジェクトを InstallShield 2025 にアップグレードした場合、SS_NOPREFIX 設定は標準のビルトイン デフォルト ウィザード ページとウィンドウへ自動的に変更されます。ただし、これらのプロジェクトを InstallShield 2025 にアップグレードするとき、定義済みウィザード ページを含むカスタムウィザード ページとウィンドウは自動的に変更されません。これらのアンパサンドの解釈を変更する場合、手動で変更を行なってください。

以前は、単一のアンパサンドを表示する代わりに、ウィザード インターフェイスの前述の領域でアンパサンドが誤ってキーボード ショートカットを指定した問題を回避するために、単一のアンパサンドの場所にエスケープ アンパサンド (&&) を使用することができました。しかし、これらのワークアラウンドを 1 つ以上含むプロジェクトを InstallShield 2025 にアップグレードした場合、エスケープ アンパサンドが 2 つのアンパサンドを表示することになります。このため、プロジェクト内のウィザード インターフェイスにおけるアンパサンドの使用を確認して、必要に応じて変更を行なう必要があります。

詳細については、「ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用」を参照してください。

Trialware サポート

InstallShield は今後、Try and Buy/プロダクト アクティベーション タイプの Trialware を作成するためのサポートを含みません。今後、Trialware ビューは InstallShield に含まれていません。

InstallShield 2013 以前のプロジェクトをアップグレードする

以下は、InstallShield 2013 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2013 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.774 (.ism プロジェクトの場合) または .2013 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.774 または .2013 を取り除いてください。
InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2013 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。
InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

InstallShield 2012 Spring 以前のプロジェクトをアップグレードする

以下は、InstallShield 2012 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2012 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.773 (.ism プロジェクトの場合) または .2012.4 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.773 または .2012.4 を取り除いてください。
InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2012 Spring 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

ターゲット システムの要件における変更

InstallShield は、今後 Windows 2000 システム用のインストール作成をサポートしません。Windows 2000 が搭載されているマシン上で、エンド ユーザーが InstallShield 2013 でビルドされたインストールを実行しようとすると、インストールが正常に実行する場合がありますが、プロジェクトにエンド ユーザーがレガシー オペレーティング システム上でインストールを実行しないように防ぐ起動条件を含まない限り、予期しない結果が起こる可能性があります。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、エンド ユーザーが Windows 2000 ステム上でインストールを実行するとメッセージを表示する起動条件を含めることができます。InstallScript プロジェクトでは、Windows 2000 システムを確認して、適切な場合にメッセージを表示する SYSINFO 構造変数を使用する InstallScript コードを追加することができます。

アドバンスト UI およびスイート/アドバンスト UI インストールには、今回より Windows Vista および Windows Server 2008 が必要です。

InstallShield は、今後 モバイル デバイス用のインストール作成をサポートしません。したがって、[モバイル デバイス] ビュー、スマート デバイス プロジェクト タイプ、Palm OS オブジェクト、および Windows Mobile オブジェクトは今後 InstallShield に含まれません。スマート デバイス プロジェクトを InstallShield 2012 Spring Express Edition 以前から InstallShield 2025 にアップグレードしようとすると、がエラー メッセージを表示して、プロジェクトを開くことができません。InstallShield 2012 Spring Express Edition 以前から InstallShield 2025 にプロジェクトをアップグレードして、プロジェクトがデスクトップ プラットフォームをターゲットとし、モバイル デバイス サポートを含む場合、InstallShield はアップグレード中にモバイル デバイス サポートを削除して警告をログ記録します。

InstallShield を実行するシステムの要件に関する変更

InstallShield、Standalone Build、および InstallShield Developer Installation Manifest (DIM) Editor を実行するオペレーティング システムの最小要件は、今回より、Windows XP SP3 または Windows Server 2003 SP2 です。以前、オペレーティング システムの最小要件は、これらのオペレーティング システムのどちらかの RTM バージョンでした。

仮想化パックの変更

仮想化パックは InstallShield Premier のスタンドアロン バージョンの一部として提供されています。InstallShield Professional Edition では、今後 アドオンとして提供されません。

Repackager プロジェクト変換ツールが含まれなくなりました

Repackager は今後、InstallShield Premier Edition には含まれません。Repackager は、今回より AdminStudio でのみ使用することができます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクト ファイル (.issuite) における SML スキーマの変更

アドバンスト UI およびスイート/アドバンスト UI プロジェクト ファイル (.issuite) は XML ベースのファイルです。InstallShield 2025 では、ファイルのユーザー インターフェイス部分の基になる XML スキーマが大幅に変更されています。多くの属性が子要素に移動しています。条件、アクション、および検証が根本的に異なります。属性文字列の代わりに要素コレクションが使用されています。新しいスキーマは、インストールのユーザー インターフェイスに構成されている内容が、より明示的です。

InstallShield 2025 で新しいアドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成すると、InstallShield は自動的に .issuite ファイルに新しい XML スキーマを使用します。InstallShield 2012 Spring 以前から InstallShield 2025 にプロジェクトをアップグレードすると、InstallShield が自動的に XML スキーマを更新します。

ビルトイン DLL カスタム アクションの新しい定義済みパス変数

InstallShield は、プロジェクトに 2 つの新しい定義済みパス変数 (ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder) を含みます。これらのフォルダーのデフォルト値は、ビルトイン InstallShield カスタム アクション DLL の32 ビット バージョンが格納されている、*InstallShield Program Files フォルダー*¥Redist フォルダー内のサブフォルダーを参照します。新しい InstallShield 2025 プロジェクトを作成するか、InstallShield 2012 Spring 以降のプロジェクトを にアップグレードすると、InstallShield は自動的にこれらの 2 つの定義済みパス変数をプロジェクトに含みます。また、InstallShield はビルトイン InstallShield DLL カスタム アクションのソース場所といったパスにこれらのパス変数を使います。これはアップグレードされたプロジェクトだけでなく、新しいプロジェクトにも適用します。以前、InstallShield は DLL ファイルの保存場所として、<ISProductFolder>¥Redist¥Language Independent¥i386 または <ISProductFolder>¥Redist¥Language Independent¥i386 Express のどちらかのフォルダーを使用しました。この変更は、情報提供を目的として報告されています。

オートメーション インターフェイスの変更

InstallShield または Standalone Build のオートメーション インターフェイスまたは Standalone Build を使用する場合、既存のコードを更新して、新しい ProgID (IswiAutoAutomation Interface Version.ISWiProject) を反映させてください。スタンドアロン オートメーション インターフェイスで、InstallShield と同じ ISWiAutomationAutomation Interface Version.dll ファイルが使用されるようになりましたが、インストールされる場所は異なります。

Standalone Build を InstallShield と同じマシンにインストールする場合、最後に登録された ISWiAutomationAutomation Interface Version.dll ファイルが使用されますので注意してください。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

スイート/アドバンスト UI およびアドバンスト UI ウィザード インターフェイスの デフォルト ウィザード フォーマットの変更

InstallShield 2025 で新しいスイート/アドバンスト UI またはアドバンスト UI プロジェクトを作成した場合、"ウィザード フォーマット" 設定のデフォルト値は新しいオプションの [Glass] です。スイート/アドバンスト UI またはアドバンスト UI プロジェクトを以前のバージョンの InstallShield から InstallShield 2025 にアップグレードした場合、"ウィザード フォーマット" 設定の値は変更されません。その場合、以前のバージョンの InstallShield で選択された値が保持されます。

プロジェクトで、適切な場合に異なるウィザード フォーマットを使用するように構成することができます。その場合、[ウィザード ページ] ノードを選択したときに [ウィザード インターフェイス] ビューに表示される "ウィザード フォーマット" 設定の値を変更します。

詳細については、「ウィザード インターフェイスのフォーマットを選択する」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールのナビゲーション ボタン テキストのフォント色の変更

一部の状況下で、アドバンスト UI またはスイート/アドバンスト UI プロジェクトを InstallShield 2012 Spring 以前から InstallShield 2025 にアップグレードしたとき、テキストとボタンの色のコントラストが不十分なために、ナビゲーション ボタンのテキストが判読不可能な場合があります。これは、ナビゲーション ボタン テキストに使用するテキスト スタイルに薄いフォント色が指定されていて、ターゲット システムではウィンドウのボタンに薄い色の使用が構成されている場合に発生します。

InstallShield 2025 より、アドバンスト UI またはスイート/アドバンスト UI インストールでは、ナビゲーション ボタンに使用されているテキストのテキスト スタイルに指定されているフォント色が考慮されます。したがって、アドバンスト UI またはスイート/アドバンスト UI プロジェクトを InstallShield 2025 にアップグレードしたとき、ナビゲーション ボタン テキストに使用するテキスト スタイルの色を調整する必要があるかもしれません。その場合、[ウィザード インターフェイス] ビューで [ウィザード ページ] ノードをクリックしてから、"ナビゲーション テキスト スタイル" 設定に選択されているテキスト スタイルを確認します。次に、同じビューの [スタイル] ノードの下にあるテキスト スタイルで、必要に応じてその設定を調整します。

InstallShield 2012 Spring 以前の場合、アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、ナビゲーション ボタン テキストのテキスト スタイルの色が無視されました。実行時に、インストールは Windows がナビゲーション ボタンに使用した色 (通常は黒) をフォント色に使用しました。

新しいショートカットのシェル プロパティのサポート

InstallShield の [ショートカット] ビューでは、ショートカットの Windows シェル プロパティを設定するためのビルトイン サポートが提供されています:

- "Windows 8 スタート画面にピン留めする" 設定は、次の GUID およびプロパティ ID の組み合わせを使って、System.AppUserModel.StartPinOption プロパティを設定します。

  9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3, 12

- "シェル プロパティ" 設定の [ピン留めをしない] オプションは、次の GUID およびプロパティ ID の組み合わせを使って、System.AppUserModel.PreventPinning プロパティを設定します。

  9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3, 9

- "シェル プロパティ" 設定の [新規として強調表示しない] オプションは、次の GUID およびプロパティ ID の組み合わせを使って、System.AppUserModel.ExcludeFromShowInNewInstall プロパティを設定します。

  9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3, 8

これらのプロパティをサポートしないバージョンの Windows 上で実行されるパッケージに、(GUID とプロパティ ID の代わりに) これらのプロパティ名が 1 つ以上あったとき、Windows Insataller はエラーを生成することがあります。

InstallShield 2012 Spring 以前で "シェル プロパティ" 設定を使ってショートカットのシェル プロパティを構成し、プロジェクトをInstallShield 2025 にアップグレードした場合、プロパティ名は適切な GUID とプロパティ ID の組み合わせに自動的に置換されません。GUID とプロパティ IDの組み合わせに素早く切り替えるには、プロジェクトをアップグレードした後に [シェル プロパティ] ビューで古い構成を削除してから、新しいサポートを使ってこれらのプロパティの 1 つ以上を構成することを考慮してください。その他、適切な GUID とプロパティ ID を使ってエントリを手動でオーバーライドすることもできます。InstallShield のビルトイン サポートについては、「ショートカットのシェル プロパティを設定する」を参照してください。

このサポートは、基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで使用できます。

InstallShield 2012 以前のプロジェクトをアップグレードする

以下は、InstallShield 2012 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2012 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば 0.772 (.ism プロジェクトの場合) または .2012 (.issuite プロジェクトの場合) というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から 0.772 または .2012 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので、ご注意ください。

InstallShield 2012 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

ターゲット システムで以前とバージョンまたは同一バージョンのスイート/アドバンスト UI インストールを起動したときの新しいデフォルト動作

デフォルトで、アドバンスト UI またはスイート/アドバンスト UI インストールには、それぞれ、2 つの "インストール済みスイート" 条件が含まれています。

- 新しいインストール済みスイートの終了条件は、エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールの現在のバージョンによって、同じアドバンスト UI またはスイート/アドバンスト UI インストールの将来のバージョンが上書きインストールされるのを防ぎます。
- 新しい "インストール済みスイート" の終了条件によって、エンドユーザーが、新しいバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールを、以前のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールの上から上書きインストールしたとき、アドバンスト UI またはスイート/アドバンスト UI インストールは初回インストール モードで実行されます。

これらの新しいデフォルト条件は、スイート/アドバンスト UI プロジェクトで使用できます。InstallShield 2012 スイート プロジェクトを InstallShield 2025 にアップグレードした場合、これらのデフォルト条件は自動的にプロジェクトに追加されます。これらの条件に関する詳しい情報は、「特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」をご覧ください。

以前、エンドユーザーが特定のバージョンのスイート インストールを、新しいバージョンのスイートが既にインストールされているターゲット システムでインストールしたとき、古いバージョンのスイートによって新しいバージョンが上書きインストールされていました。また、エンドユーザーが特定のバージョンのスイート インストールを、同一バージョンのスイートが既にインストールされているターゲット システムでインストールしたときも、古いバージョンのスイートが初回インストール モードで実行されていました。さらに、エンドユーザーが新しいバージョンのスイート インストールを、古いバージョンのスイートが既にインストールされているターゲット システムでインストールしたとき、インストールはメンテナンス モードで実行されていました。

オートメーション インターフェイスの変更

InstallShield または Standalone Build のオートメーション インターフェイスまたは Standalone Build を使用する場合、既存のコードを更新して、新しい ProgID (IswiAutoAutomation Interface Version.ISWiProject) を反映させてください。スタンドアロン オートメーション インターフェイスで、InstallShield と同じ ISWiAutomationAutomation Interface Version.dll ファイルが使用されるようになりましたが、インストールされる場所は異なります。

Standalone Build を InstallShield と同じマシンにインストールする場合、最後に登録された ISWiAutomationAutomation Interface Version.dll ファイルが使用されますので注意してください。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

スイート/アドバンスト UI プロジェクトのウィザード インターフェイスに追加された新しいコンボ ボックスのデフォルト動作への変更事項

InstallShield 2025 のスイート/アドバンスト UI プロジェクトのウィザード インターフェイスまたは 2 番目のウィンドウで新しいコンボ ボックスを作成する場合、そのコントロールは、予め定義された値を持つドロップダウン リストを含むボックスになっています。ボックスには、エンドユーザーがカスタム値が入力できるテキスト ボックスです。以前 InstallShield 2012 で、新しいコンボボックス コントロールを追加したとき、コントロールにドロップダウン リストが含まれていましたが、テキストボックスは提供されていなかったため、エンドユーザーはカスタム値を入力することができませんでした。

プロジェクトを InstallShield 2012 から InstallShield 2025 にアップグレードして、スイート/アドバンスト UI ウィザード インターフェイスにコンボ ボックスが使用されている場合、そのコンボ ボックスは、予め定義された値を持つドロップダウン リストとして残りますが、テキスト ボックスではありません。このコントロールをテキスト ボックスを持つドロップダウン リストに変更するには、コントロールの CBS_DROPDOWNLIST スタイルを False に設定します。

InstallShield 2011 以前のプロジェクトをアップグレードする

以下は、InstallShield 2011 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2011 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .771 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .771 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2011 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

InstallScript カスタム アクションで呼び出される一部の MSI API の動作を変更する

InstallScript エンジンが **MsiGetProperty** などの Windows Installer API を呼び出す方法が変更されました。エンジンは今回より、Windows Installer API を直接呼び出します。以前は順番に Windows Installer API を呼び出すラッパー API を使用しました。この変更は、バッファー処理とバッファー サイズに関する問題を解決するために行われました。これは、InstallShield 2025 で作成された新しいプロジェクト、および InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクトに適用します。

この変更によって、インストール ハンドルを必要とするすべての Windows Installer API を使用するカスタム アクション関数に渡される MSI インストール ハンドルを使用することができます。以前、ハンドルは InstallScript エンジンによって管理されていたため、Windows Installer API に直接渡すことはできませんでした。

また、この変更によって、バッファー サイズの指定が必要な任意の Windows Installer API は、正しくそのバッファー サイズが指定されない限り失敗します。0 は無効なサイズです。コード内でバッファーのサイズに 0 を渡さないようにしてください。

Windows Installer API を呼び出す既存の InstallScript カスタム アクションがある場合、十分なサイズのバッファーが指定されていることを確認してください。そうでない場合、予期しないエラーが発生する可能性があります。

次のサンプル コードでは、InstallScript で Windows Installer プロパティ値文字列の値を取得する方法、および必要に応じてバッファーのサイズを増やす方法をデモンストレーションします。

```
prototype STRING MyGetProperty (HWND, STRING);
////////////////////////////////////////////////////////////////////////////
//   MyGetProperty
//
//   MSI プロパティ文字列値を戻す
//   パラメーターを入力:
//        hMSIHandle:      現在実行中の MSI データベースへのハンドル
//        szPropertyName:  :取得するプロパティの名前
//    出力:
//   プロパティ値を含む文字列
////////////////////////////////////////////////////////////////////////////
```

```
function STRING MyGetProperty (hMSIHandle, szPropertyName)
    NUMBER  nvBuf, nResult;
    STRING  szReturn;
begin

//プロパティ値のサイズを取得
    szReturn = "";

    nResult = MsiGetProperty (hMSIHandle, szPropertyName, szReturn, nvBuf);

    if (ERROR_MORE_DATA = nResult) then
        nvBuf = nvBuf + 1;   //最後のヌル文字のバッファー サイズを増加

        //プロパティ値を取得
        nResult = MsiGetProperty (hMSIHandle, szPropertyName, szReturn, nvBuf);

        if (nResult != ERROR_SUCCESS) then
            szReturn = "";
        endif;
    endif;
    return szReturn;
end;
```

スクリプトでif (ERROR_MORE_DATA == nResult) then の代わりに if (ERROR_SUCCESS == nResult) then を使用した場合、予期しないエラーが発生する可能性があります。

基本の MSI プロジェクトにおけるビルド警告 -7235

デフォルトで、ソフトウェア識別タグ機能は、すべての基本の MSI プロジェクトで有効になっています。これは、InstallShield 2025 で作成された新しいプロジェクト、および InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクトに適用します。

基本の MSI プロジェクトで、必須識別タグ設定 ([一般情報] ビューの "一意な ID"、"タグ作成者"、"タグ作成者 ID" 設定) にデータを入力せずにリリースをビルドすると、そのプロジェクトでタグ機能を有効のままにしておいた場合、ビルド警告 -7235 が発生します。このビルド警告では、特定の必須タグが空白であるために、ソフトウェア識別タグが作成されず、インストールに含まれなかったことが通知されます。この警告を解決するには、[一般情報] ビューで、各設定に適切な値を入力するか、または "ソフトウェア識別タグの使用" 設定で [いいえ] を選択します。

COM 抽出の変更

InstallShield では、COM 抽出時に新しい監視方式をサポートします。Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で、InstallShield を使用している場合、この新しい方式がデフォルトとなります。この方法は、カーネル ドライバーを使って、ビルド時のダイナミック COM 抽出中、およびデザイン時のスタティック COM 抽出中に変更されたレジストリ領域を監視します。この新しい方式は、DLL が既存のレジストリ エントリを読み込んでビルド マシンへの変更を妨げる以前の方法の利点を組み合わせたものです。

必要な場合、UseAPIRegistryHooks レジストリ値 (32 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥RegSpy レジストリ キーに含まれる、64 ビット マシンの場合は HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥InstallShield¥RegSpy レジストリ キーに含まれる) の値データを設定して 3 つの異なる COM 抽出方式を切り替えることができます。使用可能な REG_DWORD 値データ:

- 0—API フックを使って、既存 DLLのレジストリ エントリを読み取ります。
- 1—レジストリのリダイレクトを使って、ビルド マシン上の登録済み DLL への変更を防止します。値を設定しなかった場合、これが Windows XP および Windows Server 2003 システム上でのデフォルト動作となります。

- 2—新しいカーネル モードの監視を使って、2 つのメソッドの両方の利点を組み合わせます。値が設定されていない場合、これが Windows Vista 以降および Windows Server 2008 以降のシステム上でのデフォルト動作となります。

この機能は、基本の MSI、DIM、InstallScript MSI、および マージ モジュール プロジェクト タイプに適用します。

マージ モジュール プロジェクトの文字列エントリ

マージ プロジェクトで作成される各文字列 ID には、今回よりマージ モジュールのモジュール ID GUID が含まれます。これは、すべての新しいマージ モジュール プロジェクトで作成される新しい文字列 ID、および InstallShield 2011 以前から InstallShield 2025 にアップグレードされた既存のマージ モジュール プロジェクトで作成された新しい文字列 ID に適用します。マージ モジュール プロジェクトを InstallShield 2011 以前から InstallShield 2025 にアップグレードしたとき、既存の文字列エントリにはモジュール ID GUID が追加されません。

マージ モジュール プロジェクトの文字列エントリの文字列 ID に GUID を使用することで、マージ モジュール プロジェクトとマージ モジュールを含むインストール プロジェクトで同じ文字列 ID が使用されたとき、これらの 2 つの文字列 ID に異なる値が割り当てられた場合に起こる競合を避けることができます。

InstallShield 2010 以前のプロジェクトをアップグレードする

以下は、InstallShield 2010 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2010 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .770 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .770 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2010 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

Visual Studio 2003 以前の統合サポートの終了

Visual Studio 内部から直接 InstallShield プロジェクトを作成、編集、およびビルドするには、Visual Studio 2005 以降が必要です。InstallShield を Visual Studio 2003 以前と統合することはできなくなりました。

InstallScript オブジェクトが非推奨となりました

InstallScript オブジェクトに代わって、InstallShield 前提条件が推奨されます。将来的なリリースで、InstallShield では InstallScript オブジェクトを作成または使用することはできません。また、定義済み InstallScript オブジェクトも提供されません。さらに、マージ モジュール ホルダー オブジェクトも使用できなくなります。InstallScript オブジェクトの代わりに、InstallShield 前提条件が推奨されます。InstallShield 前提条件エディターを使って独自の InstallShield 前提条件を作成して、InstallScript オブジェクト テクノロジが使用できなくなるときに備えてください。これらの InstallShield 前提条件は、InstallScript、InstallScript MSI、および基本の MSI プロジェクトで共有することができます。

InstallShield は Setup.exe および Update.exe の Unicode バージョンのみをビルド (今回より、ANSI バージョンは作成できません)

今回より、ではすべての Setup.exe と Update.exe ファイルが Unicode でビルドされます。これは、InstallShield 2025 で作成されるすべての基本の MSI、InstallScript、InstallScript MSI、および QuickPatch プロジェクトに適用します。また、InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされたすべてのプロジェクトにも適用します。このため、セットアップランチャーを Unicode バージョンまたは ANSI バージョンのどちらでビルドするかを指定するためのこれまでの設定は削除されました:

- 基本の MSI プロジェクトの [リリース] ビューにあるリリースの [Setup.exe] タブにあった "セットアップ ランチャの種類 " 設定が削除されました。
- 基本の MSI、InstallScript MSI、および QuickPatch プロジェクトから "アップデート ランチャの種類" 設定が削除されました。

  この設定は、基本の MSI および InstallScript MSI プロジェクトの [パッチのデザイン] ビュー内にあるパッチの構成の [詳細] タブで使用できました。

  QuickPatch プロジェクトの場合、この設定は [一般情報] ビューの [ビルド設定] 領域にある [詳細] タブで使用できました。

Win32 API の定義変更

InstallScript エンジンに Unicode サポートが追加されたため、InstallScript ヘッダー ファイル ISRTWindows.h でプロトタイプ化される Win32 API 関数が更新されました。適切な場合、既存の ANSI (A) 定義のほかに、API プロトタイプのワイド (W) バージョンが追加されています。一部のプロトタイプでは、A または W バージョンが指定されていません。その場合、エンジンは W バージョンを使用しようとします。以前は A バージョンが使用されました。

InstallShield 2010 以前のプロジェクトを InstallShield 2025 にアップグレードする場合、これらの新しいプロトタイプが、同じ API のユーザー定義のプロトタイプと競合する可能性があります。可能な限り、InstallScript で提供されているプロトタイプを使用することが推奨されます。ただし、これらの Windows API に InstallScript 用に新しく追加されたプロトタイプではなく、独自のプロトタイプを使用する場合、プリプロセッサ定義のリストに ISINCLUDE_NO_WINAPI_H を追加します。その場合、[ビルド] メニューから [設定] を選択します。[コンパイル/リンク] タブにある [プリプロセッサ定義] ボックスに ISINCLUDE_NO_WINAPI_H と入力します。そうしないと、コンパイル エラーが発生する可能性があります。

InstallScript コードにおける Unicode サポートを確認

InstallScript コードでユーザー定義の Win32 API その他の外部 DLL プロトタイプを使用する場合で、API が Unicode 文字列入力をサポートするバージョンを持つとき、プロトタイプが BYVAL/BYREF WSTRING または WPOINTER を使用するように更新します。また、構造体を入力として受け取るコード内の API 関数を確認して、文字列または文字列ポインター メンバーを含む API 関数が必要な場合に Unicode として適切に宣言されていることを確認します。

スクリプトで、現在 Win32 の A バージョンを呼び出す場合、W バージョンに更新します。

InstallScript エンジンが Unicode をサポートするようにアップデートされたため、InstallScript 関数 StrLength の戻り値で、この変更が確認できます。その結果、この関数への InstallScript コード呼び出しに変更を加える必要があるかもしれません。

StrLength は、**StrLengthChars** と同様に動作します。これらの関数はどちらも、指定の文字列変数 (つまり、UTF-16 エンコード文字列のコード ユニット数) で最初のヌル文字までの文字数を返します。これは、InstallShield 2025 で作成された InstallScript プロジェクト、および InstallShield の以前のバージョンから InstallShield 2025 にアップグレードされた InstallScript プロジェクトすべてに適用します。以前、**StrLength** は指定の文字列変数で ANSI コード ページのヌル文字までの文字数を返しました。**StrLengthChars** の動作に変更はありません。

InstallScript プロジェクトにおける差分リリースのサポート

InstallShield 2011 以降の InstallScript プロジェクトで作成された差分リリースは、その InstallScript インストールも InstallShield 2011 以降で作成されている場合にのみ製品をアップデートできます。InstallShield 2011 以降を使って、以前の InstallScript インストールが InstallShield 2010 以前を使って作成されている製品のアップデートを作成する場合、差分リリースではなく、完全リリースを作成してください。

これは、Unicode サポートが InstallShield 2011 の InstallScript エンジンに追加された新しい機能であるためです。差分リリースを使って、ターゲット システム上で製品の以前のバージョンをアップデートした後、以前のバージョンのインストールおよび以前のランタイム エンジンを使って、メンテナンス処理を行います。InstallScript エンジンの InstallShield 2010 以前のバージョンは、Unicode 格納形式を読み取ることができないため、インストールは失敗します。

InstallScript エンジンによってインストールされた 64 ビット レジストリ エントリのアンインストール

今回より、64 ビット ターゲット システム上で InstallScript または InstallScript MSI インストールが行われるときに、デフォルトで InstallScript エンジンがレジストリの 64 ビット部分に行われる変更をログ記録します。さらに、InstallScript エンジンによってログ記録された 64 ビット レジストリの変更が、今回より、アンインストール中にアンインストールされます。

InstallShield 2010 以前を使って、64 ビット レジストリ データを作成するインストールを作成した場合で、InstallShield 2025 を使って製品のアップグレードを作成するとき、ベース インストールによってログ記録された既存の 64 ビット レジストリ エントリは、製品がターゲット システムからアンインストールされるときに削除されません。この制限を回避する唯一の方法は、アンインストール中にレジストリ データを手動で削除する方法のみです。

スクリプト エディターの変更

[InstallScript スクリプト] ビューのスクリプト エディター ペイン内で、InstallShield 2010 以前のバージョンでは CTRL+SPACEBAR キーを押すとオートコンプリート機能としてビルトイン InstallScript 関数のリストが表示されましたが、今後は表示されません。今回より、オートコンプリート機能によるビルトイン InstallScript 関数を参照するには、その関数の最初のいくつかの文字を入力するだけです。ポップアップ リストには、その他の InstallScript キーワードも含まれています。詳細については、「スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う」を参照してください。

スクリプト エディター ペイン内で右クリックすると表示されるコンテキスト メニューは、InstallShield 2010 以前のバージョンとは異なります。

- コンテキスト メニューから [空白スペースを表示] コマンドが削除されました。スクリプト エディターで空白スペース文字や記号を表示または非表示にするには、[スクリプト エディターのプロパティ] ダイアログ ボックスを使用します。このダイアログを使って、フォント、構文の色、行番号などのスクリプト エディターのその他の設定も変更できます。このダイアログ ボックスにアクセスするには、スクリプト エディター内を右クリックしてから、[プロパティ] を選択します。

- コンテキスト メニューから、[大文字に変換する] コマンド、または [小文字に変換する] コマンドが削除されました。スクリプト エディター内のテキストを大文字に変換するには、そのテキストを選択して CTRL+SHIFT+U を押します。小文字に変換するには CTRL+U を押します。
- コンテキスト メニューから [検索] コマンドまたは [置換] コマンドが削除されました。スクリプト内で検索を行うには、CTRL+F を押してから、検索したいテキストを指定します。文字列を検索してから、それを別の文字に置換するには、CTRL+H を押してから、適切な情報を指定します。別の方法として、[編集] メニューにある [検索/置換] コマンドを使用することもできます。

キーボード ショートカットの詳細については、「スクリプト エディターのキーボード ショートカット」を参照してください。

改訂されたスクリプト エディターを含むビューは、[InstallScript] ビュー、[SQL スクリプト] ビュー、および [カスタム アクションとシーケンス] ビュー (このビューで VBScript または JScript ファイルを参照する場合) です。

InstallScript デバッガーの変更

InstallScript コードをデバッグするために InstallScript デバッガー (ISDbg.exe) をインストール開発マシンからデバッグを行うマシンにコピーする場合、必ずコピー先マシン上の同じフォルダに SciLexer.dll と呼ばれるファイルもコピーしてください。SciLexer.dll は、インストール開発マシン上の ISDbg.exe ファイルと同じフォルダにあります (*InstallShield Program Files フォルダー*¥System)。

詳細については、「任意のコンピューターでのインストールのデバッグ」を参照してください。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

DPI の変更と、その InstallScript ダイアログにおける影響

1 つ以上の編集済み InstallScript ダイアログを含む、InstallShield 2010 以前で作成した InstallScript または InstallScript MSI プロジェクトを InstallShield 2025 以降にアップグレードする場合、使用中のコンピュータが InstallScript ダイアログを編集したときに選択されていたのと同じ DPI 値を使用することを確認してください。そうでない場合、実行時にダイアログのサイズに問題が生じることがあります。

プロジェクトをアップグレードした後、必要に応じて使用中のコンピュータの DPI 値を変更することが出来ます。そうすると、実行時にダイアログのサイズが正しく表示されます。1 つ以上の編集済み InstallScript ダイアログを含む InstallShield 2025 以降のプロジェクトを InstallShield の将来のバージョンにアップグレードするとき、アップグレード中に同じ DPI 値を使用する必要はありません。

既存のプロジェクトにおける既存の InstallShield 前提条件のデザイン時およびビルド時の場所変更

今回より、InstallShield が InstallShield 前提条件ファイル (.prq ファイル)、その関連データ ファイル、および依存関係を検索するフォルダーを指定できるようになりました。以前は、InstallShield が .prq ファイルを検索するフォルダーは、*InstallShield Program Files フォルダー*¥SetupPrerequisites のみでした。

InstallShield 前提条件を *InstallShield Program Files フォルダー*¥SetupPrerequisites フォルダの場所から、[オプション] ダイアログ ボックスの [プロパティ] タブで定義した新しいカスタム場所に移動させた場合、InstallShield 2010 以前のプロジェクトを InstallShield 2025 にアップグレードするときに以下の手順が必要な場合があります:

1. [再配布可能ファイル] ビューまたは [前提条件] ビューで、プロジェクトに含まれているが、カスタム場所に存在する各 InstallShield 前提条件のチェック ボックスをクリアします。そのデータ ファイルまたは依存関

係がデフォルトの場所からカスタム場所に移動されている各 InstallShield 前提条件のチェック ボックスもクリアします。

2. 新しい [更新] ボタンをクリックします。

3. 手順 1 で、プロジェクトから削除した各 InstallShield 前提条件のチェック ボックスを選択します。

InstallShield がプロジェクトの ISSetupPrerequisites テーブルから前提条件のパスを削除します。InstallShield 2010 以前のプロジェクトでは、このテーブルに完全パスが格納されました。前提条件のチェック ボックスをクリアしてから、[更新] ボタンをクリックしないで再び選択した場合、InstallShield は ISSetupPrerequisites テーブルにファイル名のみではなく、引き続き完全パスを使用します。

InstallShield 2010 以前のプロジェクトを InstallShield 2025 にアップグレードした場合で、InstallShield 前提条件の場所を変更してから、プロジェクトにその前提条件を追加した場合、この [更新] 手順を行う必要はありません。また、InstallShield 2025 を使って新しいプロジェクトを作成した場合も、[更新] 手順は不要です。両方の状況において、InstallShield はプロジェクトの ISSetupPrerequisites テーブルにパスを含みません。このため、デフォルト パスではなく、カスタム検索パスを利用することができます。

InstallScript MSI インストールによる同製品の将来のメジャー バージョンの上書きを防ぐ

新しい InstallScript MSI プロジェクトの [アップグレード] ビューには、ISPreventDowngrade というメジャー アップグレード項目が含まれています。この項目は、エンド ユーザーが製品の現在のバージョンで同じ製品の将来のメジャー バージョンを上書きできないように防ぎます。InstallScript MSI プロジェクトを InstallShield 2010 以前から InstallShield 2025 にアップグレードした場合、ISPreventDowngrade 項目は自動的に追加されません。必要に応じて、手動で ISPreventDowngrade 項目を追加することができます。詳しくは、「現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ」をご覧ください。

InstallShield 2009 以前のプロジェクトをアップグレードする

以下は、InstallShield 2009 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2009 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .768 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .768 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2009 以前、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

InstallShield の複数エディションをインストールする

InstallShield 2025 のどれか 1 つのエディション (Premier、Professional、または Express) を同じシステムに同時にインストールすることができます。以前は、InstallShield バージョンの Premier または Professional Edition がインストールされた同じシステム上に Express Edition をインストールすることが可能でした。

InstallShield を実行できるサポート対象オペレーティング システムのリストにおける変更

今回より、InstallShield (オーサリング環境) を実行するシステムの最小オペレーティング システム要件は、Windows XP または Windows Server 2003 です。以前、最小オペレーティング システム要件は Windows 2000 XP3 でした。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

Setup.exe が Windows 9x、Windows NT4、または Windows システムで実行不可能となりました

InstallShield で作成された Setup.exe インストールは、Windows 9x、Windows NT4、または Windows Me 上で実行不可能となります。エンド ユーザーが Setup.exe を Windows 9x または Windows Me システム上で実行しようとすると、「*FullSetup.exePathAndFileName* ファイルには、Windows の新しいバージョンが必要です。 Windows のバージョンをアップグレードしてください。」というエラー メッセージが表示されます。Windows のバージョンをアップグレードしてください。」というエラー メッセージが表示されます。Windows NT4 システム上では、Windows はメッセージボックスを表示して、「*FullSetup.exePathAndFileName* は有効な Windows NT アプリケーションではありません。」とエラーを通知します。

InstallShield では、ターゲット オペレーティング システムを選択できる領域で、これらのレガシー オペレーティング システムをリストしません。たとえば、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトのプロジェクト アシスタントにある [インストール要件] タブで、これらのオペレーティング システムはリストされません。InstallScript プロジェクトでは、[プロジェクトの設定] ダイアログ ボックスの [プラットフォーム] タブでこれらのオペレーティング システムはリストされません。

InstallShield 2009 以前で作成された InstallScript プロジェクトを InstallShield 2025 にアップグレードしたときに、以前のプロジェクトのオペレーティング システムの設定にこれらのレガシー オペレーティング システムのみへの参照が含まれている場合、InstallShield によってレガシー オペレーティング システムが、すべてのサポート対象プラットフォームをターゲットにするオプションで置換されます。

Windows Installer 1.x 再配布可能ファイルが使用できなくなりました

Windows Installer 1.x 再配布可能ファイルは、サポートされていない Windows のレガシー バージョンのみをターゲットとするため、今後 InstallShield では使用できなくなりました。以前は、[リリース] ビューまたはリリース ウィザードを使って、プロジェクトに Windows Installer 1.x 再配布可能ファイルを追加することが可能でした。

VBScript ランタイム ファイルの再配布可能ファイルが使用できなくなりました

InstallShield では、今回より、VBScript ランタイム ファイル用の InstallShield オブジェクトが提供されていません。この再配布可能ファイルは、今ではサポートされていない Windows の古いバージョンをターゲットとします。

InstallScript ダイアログ ソース コードの変更

ビルトイン InstallScript ダイアログ用の InstallScript コードは、個別の InstallScript スクリプト ファイル (.rul) から単一の ISRTScriptDialogs.rul ファイルに移動しました。また、InstallScript ビューのイベント クラス ドロップダウン リストに、新しい Dialog Source オプションが追加されました。このオプションを選択すると、イベントハンドラー ドロップダウン リストには、すべての InstallScript ダイアログがリストされます。このリストから任意のダイアログを選択して、そのコードをカスタマイズできます。

InstallShield は、下位互換性をサポートします。ダイアログ ソース コードを InstallShield 2009 以前のプロジェクトにインポートしてから、プロジェクトを InstallShield 2025 にアップグレードしたとき、そのダイアログ コードを引き続き使用できます。ただし、イベント ハンドラー ドロップダウン リストでダイアログを選択すると使用可能となるダイアログ ソースを使う場合、先にアップグレード済みプロジェクトにいくつかの変更を行わなくてはなりません。そうしないと、コンパイル エラーが発生する可能性があります。ダイアログ コードを使用するには、[ビルド] メニューから [設定] を選択します。[コンパイル/リンク] タブにある [プリプロセッサ定義] ボックスに次を入力します:

_ISSCRIPT_NEW_STYLE_DLG_DEFS

この定義を追加すると、それ以前に InstallShield の以前のバージョンでプロジェクトにダイアログがインポートされている場合、これらのインポートされた .rul ファイル内のコードがコンパイル エラーを引き起こす場合があります。これらのエラーは解決する必要があります。

_ISSCRIPT_NEW_STYLE_DLG_DEFS が定義されていない場合、InstallScript ビューのイベント クラス ドロップダウン リストで Dialog Source オプションを選択すると、警告メッセージが表示されます。

_ISSCRIPT_NEW_STYLE_DLG_DEFS 定義は、InstallShield 2025 で作成されたすべての新規プロジェクトに自動的に追加されます。

この機能は、InstallScript 、InstallScript MSI、および InstallScript オブジェクト プロジェクト タイプに適用します。

InstallScript ヘッダー ファイルの変更

InstallScript ヘッダー ファイル (.h) が再編成されました。その結果、いくつかの .h ファイルが使用できなくなりました。プロジェクト内の InstallScript ファイル (.rul) がこれらの古い .h ファイルを 1 つ以上参照する場合、スクリプトをコンパイルするとき、またはリリースをビルドするときにコンパイル警告が表示されます。この警告を解決するには、古い .h ファイルを参照する #include ステートメントをすべて削除してから、リリースをビルドします。また、Setup.rul ファイルまたは Setup.rul が参照するその他のスクリプト ファイルにおいて、ifx.h が #include ステートメントで参照されていることを確認します。

Windows API プロトタイプに関する InstallScript の変更

サービス関連の Windows API プロトタイプ (CreateServiceA、StartServiceA、および ControlService など) が、すべての InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトにおいて ISRTWindows.h でプロトタイプ化されました。また、これらは InstallScript カスタム アクションを含む基本の MSI プロジェクトとマージ モジュール プロジェクトでも、ISRTWindows.h でプロトタイプ化されました。

これらの Windows API に InstallScript 用に今回よりプロトタイプ化された定義ではなく、独自の定義を使用する場合、プリプロセッサ定義のリストに **ISINCLUDE_NO_SERVICEAPI** を追加します。これには、[ビルド] メニューから [設定] を選択します。[コンパイル/リンク] タブにある [プリプロセッサ定義] ボックスに **ISINCLUDE_NO_SERVICEAPI** と入力します。そうしないと、コンパイル エラーが発生する可能性があります。

定数 ISINCLUDE_NO_WINAPI_H は、今回より Windows 定数または Windows 構造定義ではなく、Windows API プロトタイプのみを抑制します。

パッチの作成 (基本の MSI、InstallScript MSI)

InstallShield は今回より、パッチの作成時に Windows Installer 4.5 テクノロジを使用します。この変更は、情報提供を目的として報告されています。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、新しく追加された IIS サポート用のカスタム アクションとデータベース テーブル

InstallShield の基本の MSI または InstallScript MSI プロジェクトで [IIS 構成] ビューを使って IIS サポートを追加した場合、IIS 機能をサポートするためのいくつかの DLL カスタム アクションがプロジェクトに自動的に追加されます。InstallShield 2025 では、これらのカスタム アクションが強化され、アプリケーションを Web サイトに追加できるようになりました。また、これらのアクションの名前は、標準の命名規則に従って変更されています:

- ISIISCosting — caExtractIISSuppFiles アクションに代わるカスタム アクション
- ISIISRollback — caRlbackVRoots アクションに代わるカスタム アクション
- ISIISRollback — caRemoveVRootsアクションに代わるカスタム アクション
- ISIISInstall — caCreateVRoots アクションに代わるカスタム アクション
- ISIISCleanup — caIISCleanup アクションに代わるカスタム アクション

各 DLL カスタム アクションのエントリ ポイントは、対応するカスタム アクションの名前と一致するように名前が変更されています。

IIS 機能には、管理者権限が必要です。そのため、これらの各 DLL カスタム アクションは Privileged プロパティの値をチェックします。値は 1 でなくてはならず、それ以外の場合は、実行時にエラーが表示されます。InstallShield 2009 以前では、各 IIS カスタム アクションは Privileged = 1 条件を含んでいました。InstallShield 2025 では、カスタム アクションが Privileged プロパティ値をチェックするため、IIS Web サイトまたはその他の IIS データをプロジェクトに追加するときに、この条件は設定されません。

IIS サポートを含む基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを InstallShield 2009 以前のバージョンから InstallShield 2025 にアップグレードすると、自動的にカスタム アクションが正しく更新され、名前が変更されます。さらに、古いカスタム アクションの条件がデフォルト条件から変更されていない場合、Privileged = 1 条件も削除されます。条件が変更されている場合、InstallShield は既存の条件をそのまま残します。必要に応じて、任意の条件を手動で変更することができます。

InstallShield 2025 では、すべての IIS データは ISIISItem および ISIISProperty テーブルに格納されます。InstallShield 2009 以前では、IIS データは ISIISAppPool、ISIISCommon、ISIISMetaData、ISIISWebServiceExtension、ISVRoot、ISVRootAppMaps および ISWebSite テーブルに格納されました。IIS サポートを含む基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを InstallShield 2009 以前のバージョンから InstallShield 2025 にアップグレードすると、IIS データが自動的に新しいテーブルに移動し、また古いテーブルはプロジェクトから削除されます。

詳細については、次を参照してください。

- インターネット インフォメーション サービス
- InstallShield カスタム アクション リファレンス

[再配布可能ファイル] ビューの変更

[再配布可能ファイル] ビューには、強力な検索機能と組織化機能を提供する新しいツールバーとグループ ボックス領域が追加されました。このビューの新しい [詳細の表示] ボタンを使って、選択した再配布可能ファイルについての詳細ペインの表示/非表示を切り替えることができます。詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。[詳細の表示] ボタンは、以前、このビューの右上にあった [詳細の表示] および [詳細の非表示] リンクの代わりとなります。

新しいグループ ボックス領域は、[再配布可能ファイル] ビューの新しいツールバーの下にあります。列ヘッダーを、このグループ ボックス領域にドラッグ アンド ドロップして、再配布可能ファイルのリストを階層形式で表示することができます。すべての再配布可能ファイルをチェック ボックスが選択されているアイテムと選択されていないアイテムごとに 2 つのグループに分けて表示するには、チェック ボックス列をグループ ボックス領域にドラッグします。これで、プロジェクトに含まれているすべての再配布可能ファイルを認識しやすくなります。これは、以前、任意の再配布可能ファイルを右クリックしてから [選択したアイテムのみを表示] をクリックしたときの動作と似ています。[再配布可能ファイル] ビューで [選択したアイテムのみを表示] コマンドは、今後使用できません。

詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

連鎖 .msi パッケージの UI レベルをメインの .msi パッケージの UI レベルに限定する機能

連鎖 .msi パッケージの UI レベルは、今回より、親パッケージの現在の UI レベル以内に制限されました。たとえば、[リリース] ビューを使ってプロジェクトに連鎖 .msi パッケージを追加して、その "UI レベル" 設定に 完全 UI (/qf) を選択するが、メイン インストールがサイレント (/qn) で起動される場合、連鎖 .msi パッケージはサイレントで起動します。以前、連鎖パッケージはメイン インストールの [リリース] ビューでオーサされた UI レベルと同じレベルを表示しました。この動作を復元するには、ISChainExceedUILevel プロパティを 1 と等しく設定します。

XML ファイルの変更におけるエンコードと関連する相違点

[XML ファイルの変更] ビューを使ってターゲット マシンに既存するファイルまたはインストールの一部としてインストールされるファイルの変更を構成する場合、インストールは、その XML ファイルで指定されているエンコードを使用するようになりました。以前は、[XML ファイルの変更] ビューで指定されたエンコードが使用されました。これは、InstallShield 2025 で作成された新しいプロジェクト、および InstallShield 2009 以前からアップグレードされたプロジェクトに適用します。

また、ターゲット ファイルに存在しない要素の [この要素が存在しない場合、常に作成する] チェック ボックスがクリアされている場合、今回より、その子要素は作成されません。従って、たとえば //A/B/C という XML ファイルで、システム B が存在しない、または作成されるように設定されていない場合、C がターゲット システムで作成されることはありません。

InstallScript インストールのアンインストール キーにおけるメジャーおよびマイナー バージョンのレジストリ エントリの変更

InstallScript インストールは、アンインストール キーに VersionMajor および VersionMinor レジストリ値を常に作成します。これらの値の名前は、基本の MSI および InstallScript MSI インストール中に作成されるエントリの名前に一致します。これは、InstallShield 2025 で作成された新しいインストール、および InstallShield 2009 以前からアップグレードされたインストールに適用します。以前、InstallShield 2009 以前では、InstallScript インストールが作成する値の名前は MajorVersion および MinorVersion でしたが、今後は作成しません。

新しいレジストリ値を使用するため、次の InstallScript 定数の値が変更されました:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION は、今回より、MajorVersion ではなく VersionMajor です。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION は、今回より、MinorVersion ではなく VersionMinor です。

MaintenanceStart 関数が呼び出されると、レジストリに更新された値名が作成されます。デフォルトで、古い値名が存在するとき、それが削除されます。ターゲット システムから古い値名を削除したくない場合、REGDB_OPTION_NO_DELETE_OLD_MAJMIN_VERSION という名前の新しい REGDB_OPTIONS オプションを使用します。古い値名のみを使用し続けたい場合は、**MaintenanceStart** が戻った後に新しいバージョンを削除しなくてはなりません。

RegDBGetItem 関数と共に REGDB_UNINSTALL_MAJOR_VERSION または REGDB_UNINSTALL_MINOR_VERSION を使用すると、**RegDBGetItem** は、まず新しい値をチェックします。新しい値が検出されると、関数は新しい値から値データを返します。新しい値が検出されなかった場合、関数は自動的に古い値をチェックします。古い値が検出されると、関数は古い値から値データを返します。

下位互換性を目的として、次の新しい定数が提供されています:

- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD

これらの定数を **RegDBGetItem**、**RegDBSetItem**、および **RegDBDeleteItem** 関数と共に指定して、古い値を取得、設定、および削除できます。

次の新しい文字列定数も使用できます:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD は、MajorVersion として定義されています。
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD は、MinorVersion として定義されています。

詳細については、InstallScript 言語リファレンスの次のセクションを参照してください:

- REGDB_VALUENAME_UNINSTALL_MAJORVERSION
- REGDB_VALUENAME_UNINSTALL_MAJORVERSION_OLD
- REGDB_VALUENAME_UNINSTALL_MINORVERSION
- REGDB_VALUENAME_UNINSTALL_MINORVERSION_OLD
- REGDB_UNINSTALL_MAJOR_VERSION_OLD
- REGDB_UNINSTALL_MINOR_VERSION_OLD
- REGDB_OPTIONS
- RegDBSetItem
- RegDBGetItem
- RegDBDeleteItem

編集可能なダイアログの一覧からの SdShowMsg ダイアログの削除

ダイアログ エディターは現在、**SdShowMsg** などのタイトル バーを含まないダイアログをサポートしません。**SdShowMsg** ダイアログをカスタマイズすると、ダイアログが破損する可能性があります。このダイアログは今後、編集可能なダイアログとして [ダイアログ] ビューには表示されません。このダイアログをカスタマイズするには、ダイアログ エディターではなく、**SdShowMsg** 呼び出しを使います。

オートメーション インターフェイスの変更

OSFilter の eosAll 定数の値 (オートメーション インターフェイスの ISWiComponent および ISWiRelease のメンバー) が変更されました。新しい値は 64028880 です。以前は 5308624 でした。この定数の値を使用して、オートメーション インターフェイスを通してコンポーネントまたはリリースのオペレーティング システム一覧を構成する場合、スクリプトを新しい値にアップデートしなくてはなりません。

詳細は、次を参照してください:

- ISWiComponent オブジェクト
- ISWiRelease オブジェクト

InstallScript ランタイム スクリプト、ライブラリ、および ヘッダー ファイルの場所変更

InstallShield と共にインストールされる InstallScript ランタイム ライブラリ ファイルは、複数の個別のサブディレクトリではなく、1 ヶ所にまとめられました。スクリプト、ライブラリ、およびヘッダー ファイルは、以下のディレクトリ内の Src、Lib、および Include フォルダーにインストールされます:

InstallShield Program Files フォルダー¥Script¥Isrt

次のフォルダーは、そこに含まれていたファイルが前述の場所に統合されたため、今後はインストールされません。

InstallShield Program Files フォルダー¥Script¥IISRuntime

InstallShield Program Files フォルダー¥Script¥SQLRuntime

InstallShield Program Files フォルダー¥Script¥XMLRuntime

名前の競合を避けるために、次の場所にある Assert.h ファイルは、ISAssert.h に名前が変更されました:

InstallShield Program Files フォルダー¥Script¥Isrt¥Include

InstallShield 2025 で新しいプロジェクトを作成すると、新しいファイルの場所が使用されます。InstallShield 2009 以前から InstallShield 2025 にプロジェクトをアップグレードすると、新しい場所を使ってプロジェクトが更新されます。

リンク ライブラリとその場所の指定方法に関する変更点

[設定] ダイアログ ボックスにある [コンパイル/リンク] タブに新しく追加された [追加ライブラリ パス] ボックスを使って、InstallScript コンパイラが標準の InstallShield の場所には存在しない InstallScript ライブラリ (.obl ファイル) を検索する場所を指定できます。InstallScript と InstallScript オブジェクト プロジェクトでの標準の場所は次のとおりです:

- <ISProductFolder>¥Script¥Ifx¥Lib
- <ISProductFolder>¥Script¥Isrt¥Lib

基本の MSI および InstallScript MSI プロジェクトでの標準の場所は次のとおりです:

- <ISProductFolder>¥Script¥Iswi¥Lib
- <ISProductFolder>¥Script¥Isrt¥Lib

InstallShield 2025 で新しいプロジェクトを作成すると、[コンパイル/リンク] タブの [ライブラリ (.obl)] ボックスに ISRT.obl などの標準 InstallShield スクリプト ライブラリが自動的に一覧表示されます。ただし、このボックスでフル パスは表示されなくなりました。独自のカスタム ライブラリを追加するには、[ライブラリ (.obl)] ボックスでライブラリ ファイル名を、また [追加ライブラリ パス] ボックスにパスを指定します。[ライブラリ (.obl)] ボックスでフル パスとファイル名を指定する必要はありません。

InstallShield 2009 以前で作成したプロジェクトを InstallShield 2025 にアップグレードすると、[ライブラリ (.obl)] ボックスにリストされている標準スクリプト ライブラリのパスが自動的に削除されます。詳細については、「[コンパイル/リンク] タブ」を参照してください。

ISOSVERSIONINFO の InstallScript 構造定義の削除

ISOSVERSIONINFO 構造の定義と、それに対応する使用されていない、この構造のグローバル インスタンスが削除されました。代りに、OSVERSIONINFO 構造が使用できます。この定義の削除による機能の変更はありませんが、ISOSVERSIONINFO 構造定義またはグローバル構造インスタンスを使用すると、コンパイラ エラーが発生します。

コンパイラ エラーを回避するには、以下のどちらかを行います:

- スクリプトを対応する OSVERSIONINFO 構造を使って更新し、必要な場合は、この構造のローカル インスタンスを宣言します。スクリプトを適切な構造メンバー名で更新します。(OSVERSIONINFO メンバー名は、ISOSVERSIONINFO メンバー名とは異なりますので、ご注意ください。)OSVERSIONINFO の定義は以下のとおりです:

```
typedef OSVERSIONINFO
    begin
        NUMBER nOSVersionInfoSize;
        NUMBER nMajorVersion;
        NUMBER nMinorVersion;
        NUMBER nBuildNumber;
        NUMBER nPlatformId;
        STRING szCSDVersion[128];
    end;
```

- 以下のように、構造と構造インスタンスをローカルで宣言します:

```
// オペレーティング システムのバージョン情報を含むデータ構造。
// ISCompareServicePack が使用します。
typedef ISOSVERSIONINFO // 構造の定義
begin
    LONG ISIOSVersionInfoSize;    // このデータ構造のバイト サイズ
    data structureLONG ISIMajorVersion;        // OS のメジャー バージョン番号
    LONG ISIMinorVersion;         // OS のマイナー バージョン 番号
    LONG ISIBuildNumber;          // OS のビルド番号
    LONG ISIPlatformId;           // OS プラットフォーム
    STRING szISCSDVersion [128];  // OS に関する追加情報
end;

// OS バージョン 情報データ構造の変数
// ISCompareServicePack が使用します。
ISOSVERSIONINFO ISVersion;

// OS バージョン情報変数をポイントするポインター
// ISCompareServicePack が使用します。
ISOSVERSIONINFO POINTER pISVersion;
```

InstallScript プロジェクトにおける Setup.ini の旧形式キーの書き込みについて

InstallScript プロジェクトで、旧形式のキー (Resource、EngineVersion、および EngineBinding) が Setup.ini ファイルに書き込まれることがなくなりました。

ISCab.exe が使用できなくなりました

今回より ISCab.exe はサポートされていません。そのため、ISCab.exe は InstallShield に含まれていません。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

ファイル、フォルダー、およびレジストリ キーのアクセス許可を保護するためのサポート変更点

[一般情報] ビューに追加された "ロックダウンの設定方法" 設定を使って、プロジェクトに含まれるファイル、フォルダー、およびレジストリ キーに設定する新しいアクセス許可について、新しいカスタム InstallShield 処理または従来型の Windows Installer 処理のどちらを使用するかを指定できます。新しい [カスタム InstallShield 処理] オプションには、[従来型の Windows Installer 処理] オプションよりも多くの利点があります。

すべての新しいプロジェクトにおける、この設定のデフォルト値は [カスタム InstallShield 処理] オプションです。プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、この設定のデフォルト値は [従来型の Windows Installer 処理] オプションです。

この新しい設定は、基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで利用できます。

詳細は、次を参照してください:

- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する

Beta Windows Installer 5 のユーザーごとインストール サポートにおける ReadyToInstall ダイアログの変更

[一般情報] ビューには、"[ユーザーごと] オプションの表示" 設定があります。この設定を使って、特定の状況下において ReadyToInstall ダイアログにエンド ユーザーが製品をインストールする方法 (現在のユーザーまたはすべてのユーザー) を指定できるボタンを含めるかどうかを指定できます。[ユーザーごと] ボタンは、Windows Installer プロパティ MSIINSTALLPERUSER を 1 に等しく設定して、パッケージを現在のユーザーにインストールすることを示します。MSIINSTALLPERUSER プロパティは Windows Installer 5 で使用できます。

InstallShield 2025 で新しい基本の MSI プロジェクトを作成すると、ReadyToInstall ダイアログはユーザーごとおよびマシンごとボタンのサポートを含みます。これらのボタンは、適切な場合、実行時に表示または非表示されます。基本の MSI プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、このサポートは ReadyToInstall ダイアログに自動的には追加されません。必要な場合、これらのボタンと関連条件を ReadyToInstall ダイアログに手動で追加することができます。新規 InstallShield 2025 プロジェクトで ReadyToInstall ダイアログを使用することがガイドラインとされています。

SecureCustomProperties に追加された機能のインストール先のパブリック ディレクトリ プロパティ

機能の "インストール先" 設定に場所を指定するとき、その場所にパブリック ディレクトリ プロパティが含まれる場合、今回より、InstallShield はそのプロパティを SecureCustomProperties プロパティに追加して、エンド ユーザーが製品のアドバタイズ後にインストール先を変更できるようにします。これは、InstallShield 2025 で作成された新しいプロジェクトで発生します。プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、すべての機能のインストール先にも同様の変更が行われます。

この変更は、基本の MSI および InstallScript MSI プロジェクト タイプに適用します。

InstallWelcome ダイアログと ResolveSource アクションにおける条件の変更

InstallWelcome ダイアログと ResolveSource アクションの条件は、InstallShield 2025 で作成されたすべての新しい基本の MSI プロジェクトにおいて、**Not Installed** に変更されました。この条件は、InstallWelcome ダイアログと ResolveSource アクションを、パッチを伴う初回インストールで使用できるように変更されました。基本の MSI プロジェクトを InstallShield 2009 以前から InstallShield 2025 にアップグレードした場合、この条件は自動的には変更されません。パッチを含む初回インストールでダイアログとアクションを使用する場合、この条件をアップグレードされたプロジェクトで **"Not Installed"** に変更できます。

SdLicenseRtf と SdLicense2Rtf.rtf 関数の .rtf ファイルサイズの制限についての強化

InstallScript ダイアログ関数 **SdLicenseRtf** および **SdLicense2Rtf** で使用される .rtf ファイルのサイズ制限は、64 KB に代って 16 MB となりました。以前、ファイル サイズが 64 KB を超えると、実行時に EULA テキストの一部がライセンス ダイアログに表示されませんでした。

InstallShield 2009 以前でスクリプトの **SdLicenseRtf** または **SdLicense2Rtf** 関数をオーバーライドしたあと、そのプロジェクトを InstallShield 2025 にアップグレードした場合、**SendMessageTimeout** 呼び出しを DLG_INIT の EM_EXLIMITTEXT メッセージで更新して、サイズ制限を手動で変更する必要があります。**SendMessageTimeout** 呼び出しの iParam パラメーター (4 番目のパラメーター) を変更します。**SendMessageTimeout** 呼び出しは、次のように変更します:

SendMessageTimeout(hEdit, EM_EXLIMITTEXT, 0, 0xffffff);

以前は、コードに以下が含まれていました:

SendMessageTimeout(hEdit, EM_EXLIMITTEXT, 0, 0);

InstallScript MSI インストールからの OnResolveSource イベント ハンドラーの削除

InstallScript イベント ハンドラー OnResolveSource が、InstallScript MSI プロジェクトから削除されました。今回より、Windows Installer がソース解決をすべて処理します。InstallShield 2009 以前で InstallScript MSI プロジェクトに OnResolveSource イベントを追加して、そのプロジェクトを InstallShield 2025 にアップグレードした場合、そのイベントは呼び出されません。

基本の MSI インストールで SetupCompleteSuccess ダイアログからのログ ファイルの表示方法に関する変更

ShowMsiLog カスタム アクションは、今回より、WindowsFolder ディレクトリではなく、SystemFolder ディレクトリから Notepad.exe を起動します。このため、インストールが Windows Vista 以降で実行され、エンド ユーザーが SetupCompleteSuccess ダイアログでログ ファイルの表示を選択した場合、インストールは SystemFolder ディレクトリから Notepad.exe を起動します。これは、Windows Server 2008 Standard Edition では、Notepad.exe が Windows ディレクトリではなく、System32 ディレクトリに配置されているために変更されました。

この動作は、InstallShield で作成されるすべての新しい基本の MSI プロジェクトではデフォルトで使用できます。InstallShield 2009 以前のプロジェクトを InstallShield 2025 にアップグレードした場合、InstallShield が自動的にこの動作を変更することはありません。必要に応じて動作を手動で変更することができます。その場合、[カスタム アクションとシーケンス] ビューで ShowMsiLog アクションをクリックします。(このアクションが表示されない場合、[カスタム アクション] ノードを右クリックしてから、[すべてのカスタム アクションを表示] をクリックします。)"ファイル名とコマンドライン" 設定を次のように構成します:

[SystemFolder]notepad.exe "[MsiLogFileLocation]"

つまり、値には [WindowsFolder] の代わりに [SystemFolder] を使用します。

InstallScript MSI インストールにおける ALLUSERS プロパティの変更点

InstallShield 2010 より、新しい InstallScript MSI プロジェクトでは、デフォルトで ALLUSERS プロパティに 1 が設定されます。ほとんどのインストールは、マシンごとに管理者権限を使用して実行される必要があるため、これが推奨される実装です。この値は、InstallScript MSI がサイレントで実行される際の ALLUSERS 関連の問題を回避するためにも推奨されます。

InstallShield 2009 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、ALLUSERS プロパティの値は自動的に変更されません。また、このプロパティが以前のプロジェクトで定義されていない場合も、自動的に追加されません。

InstallScript インストールは今後、_Setup.dll を含みません

InstallScript インストールは、今回より _Setup.dll を含みません。以前の一部のバージョン (DevStudio 9 および InstallShield X) は、アンインストールで _Setup.dll をログ記録しませんでした。その結果、アンインストール後にこのファイルが Disk1 フォルダーの場所 (DISK1TARGET) に残りました。InstallShield のより新しいバージョン (InstallShield 10.5 から InstallShield 2009) でアップデートを作成して、そのアップデートが DevStudio 9 または InstallShield X で作成された元のインストール向けの場合、_Setup.dll がアップデートによってログ記録されたために、それがアンインストール中に削除されました。_Setup.dll は InstallShield 2025 で作成された InstallScript インストールに含まれていないため、_Setup.dll ファイルが残されたままになる可能性があります。このため、DevStudio 9 または InstallShield X で作成されたインストールからのアップデートを行う場合、アンインストールを確実に完了するために、アンインストール中に _Setup.dll ファイル (DISK1TARGET ^ "_Setup.dll") を手動で削除しなくてはならない場合があります。

プロジェクトを以前のバージョンで保存する

InstallShield は、今後プロジェクトのダウングレードをサポートしません。つまり、InstallShield 2025 プロジェクトを InstallShield 2009 以前のプロジェクトとして保存することはできません。

Trialware サポート

Trialware ビューを含むのは InstallShield Premier Edition のみです。このエディションでは、Try and Die タイプの Trialware を作成できます。InstallShield は今後、Try and Buy/プロダクト アクティベーション タイプの Trialware を作成するためのサポートを含みません。

Web プロジェクト

Web プロジェクト タイプは、今後、InstallShield で作成可能な新しいプロジェクトの種類の 1 つとしてリストされません。InstallShield 2009 以前の Web プロジェクトで提供されていた同じ機能が必要な場合は、基本の MSI プロジェクトを作成してから、[IIS 構成] ビューで Web サイトを追加します。

Web プロジェクトと基本の MSI プロジェクトとの唯一の違いは、新しい Web プロジェクトには [ファイルとフォルダー] ビューの IISROOTFOLDER ディレクトリに定義済みのフォルダーが自動的に含まれていた点です。IISROOTFOLDER ディレクトリに追加されたファイルはすべて、ターゲット システムの Web サーバーのルート ディレクトリにインストールされます。プロジェクトに Web サイトを追加すると、InstallShield が基本の MSI プロジェクトに IISROOTFOLDER ディレクトリの定義済みフォルダーを追加します。このため、[IIS 構成] ビューで 1 つ以上の Web サイトが構成されている基本の MSI プロジェクトは、InstallShield 2009 以前で作成された Web プロジェクトに相応します。

Compact プロジェクト

InstallShield は、今後 Compact プロジェクトをサポートしません。

Visual Studio の統合

Microsoft Visual Studio の統合は 1 回につき、InstallShield Premier Edition または InstallShield Professional Edition のいずれかの 1 バージョンとのみ可能です。システムで最後にインストールまたは修復された InstallShield のバージョンが Visual Studio の統合に使用されます。

InstallShield 2008 以前のプロジェクトをアップグレードする

以下は、InstallShield 2008 および以前のバージョンで作成されたプロジェクトを InstallShield 2025 にアップグレードする際に発生する可能性のある問題についての情報です。また、新しい InstallShield 2025 プロジェクトと InstallShield 2008 および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクト間の潜在的な動作の違いについてもアラートします。

InstallShield の以前のバージョンで作成されたプロジェクトのアップグレードに関する一般情報

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .766 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .766 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 2008、InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2025 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2025 にアップグレードすることはできませんので、ご注意ください。

すべてのプロジェクトに影響する変更 (新規プロジェクトおよびアップグレードされたプロジェクト)

このセクションでは、新規プロジェクトおよび InstallShield の以前のバージョンからアップグレードされたプロジェクトに影響する変更について説明されています。

追加された新しいリリース用のデフォルト セットアップランチャー値: Windows Installer Is Not Included

基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで新しいリリースを作成するとき、Windows Installer エンジンの再配布可能ファイルがデフォルトで含まれなくなりました:

- リリース ウィザードを使って新しいリリースを作成する場合、[セットアップランチャー] パネルで Windows Installer を含めるかどうかを指定します。"**これらのオペレーティング システムをサポートする**" 設定のデフォルト値は **[Windows Installer をインストールしない]** です。以前のデフォルト値は **[Windows 9X と NT の両方]** でした。
- 今回より、[リリース] ビューの [リリース] エクスプローラーで新しいリリースを右クリックすると、[Setup.exe] タブにある [セットアップランチャー] 設定のデフォルト値は **[はい (MSI エンジンを含まない)]** です。この設定の以前のデフォルト値は **[はい (Windows NT および Windows 9x MSI エンジンを含める)]** でした。

この変更は、新しい InstallShield 2025 プロジェクトで作成されたすべての新しいリリースに適用します。

プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードする場合、この変更はすべての新しいリリースに適用します。InstallShield の以前のバージョンで作成されたリリースの値が自動的に変更されることはありません。

アップグレードとパッチの検証

QuickPatch プロジェクトをビルドしたとき、InstallShield はパッチとアップグレードの検証を実行するようになりました。したがって、InstallShield 2025 で QuickPatch パッケージをビルドしたとき、検証エラーおよび警告が表示される場合があります。InstallShield 2008 以前で QuickPatch プロジジェクトをビルドしたとき、InstallShield はアップグレートとパッチの検証を実行しないため、ビルド時にパッチまたはアップグレードの検証エラーまたは警告が表示されることはありませんでした。

さらに、QuickPatch プロジェクトおよび基本の MSI と InstallShield MSI プロジェクトの [パッチのデザイン] ビューを使って作成されたパッチで、Val0015 警告が ISSelfReg テーブルをチェックするようになりました。QuickPatch またはパッチ プロジェクトで ISSelfReg テーブルが追加されたれた場合、Val0015 はパッチがアンインストール可能であることを警告します。以前、Val0015 はこのテーブルのエントリをチェックしませんでした。

ビルド時に Setup.exe と ISSetup.dll にストリームされるファイルの圧縮

Setup.exe セットアップランチャーまたは ISSetup.dll ファイルを使用するリリースをビルドしたとき、InstallShield によって、ビルド時に Setup.exe ファイルまたはISSetup.dll ファイルにストリームされるファイルが圧縮されるようになりました。InstallShield が使用するデフォルトの圧縮レベルは、ファイルのサイズと実行時に圧縮ファイルを展開するために必要な時間のバランスをとっての目安です。これは、InstallShield 2008 以前から InstallShield 2025 にアップグレードされた既存の基本の MSI プロジェクトおよび InstallScript MSI プロジェクト、および新規の基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに適用します。

圧縮レベルを変更する場合、または圧縮をしない場合、マシン全体に適用する設定を利用してデフォルトのレベルをオーバーライドすることができます。詳細については、「Setup.exe と ISSetup.dll にストリームされるファイルの圧縮レベルを構成する」を参照してください。

以前、InstallShield には、ビルド時に Setup.exe ファイルまたは ISSetup.dll ファイルにストリームされたファイルを圧縮することができませんでした。このため、InstallShield 2008 以前でビルドされたリリースと InstallShield 2025 のデフォルト圧縮レベルでビルドされた同じリリースを比較したとき、Setup.exe または ISSetup.dll のファイル サイズに若干違いがある場合があります。また、ファイルの展開にかかる時間にも差が生じる場合があります。

マルチパート .cab ファイル

InstallShield には、今回より、ビルド時にネットワーク イメージ リリースについて作成した各 .cab ファイル (標準圧縮タイプ(カスタム圧縮ではない圧縮) で、かつすべてのファイルが単一ファイルの .msi パッケージまたは Setup.exe セットアップランチャーに埋め込まれている .cab ファイル) に対して 600 MB がデフォルト制限として設定されています。InstallShield で、この種類のリリースについて .cab ファイルを作成しているとき、この制限に達すると、データが 2 つ以上の.cab ファイルに分割され、マルチパートの .cab ファイルが作成されます。これは、InstallShield 2008 以前から InstallShield 2025 にアップグレードされた既存の基本の MSI プロジェクトおよび InstallScript MSI プロジェクト、および新規の基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに適用します。

.cab サイズの制限は必要に応じて変更することができます。InstallShield でマルチパート .cab ファイルを作成しない場合、単一の .cab ファイルを作成するように構成できます。詳細については、「.cab ファイルの最大サイズを構成する」を参照してください。

以前、InstallShield では、マルチパート .cab ファイルを作成できなかったため、.cab ファイルのサイズのビルトイン制限もありませんでした。

オートメーション インターフェイスと Standalone Build

InstallShield オートメーション インターフェイスと共に動作する既存のオートメーション スクリプトがある場合、スタンドアロン オートメーション インターフェイスと共に使用するためにスクリプト全体でライブラリの名前を IswiAuto*N* から SAAuto*N* (*N* はバージョン番号です) に変更する作業は必要なくなりました。

InstallShield Premier Edition で提供されている Standalone Build は、InstallShield と同じディレクトリ構造を使用するようになりました。

InstallShield のコマンドライン ビルドで使われる ISCmdBld.exe も Standalone Build と共にインストールされています。以前、Standalone Build は別のファイル (IsSaBld.exe) をコマンドライン ビルドに使用していました。

InstallScript プロジェクトにおける FlexNet Connect のサポート

FlexNet Connect のサポートをインストール プロジェクトに追加するときに使用する [アップデート通知] ビューが InstallScript プロジェクトで使用できなくなりました。

InstallScript プロジェクトにおける FlexNet Connect のサポートに関する詳細は次の参照してください:

- InstallShield テーブルで ISEnableUpdateService が 1 に変更されたとき、ISUS.obl はライブラリー覧に追加されません (つまり、他の FlexNet Connect 機能が有効にされません)。ISUS.obl ライブラリは現在サポートされていません。
- InstallScript プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードすると、FlexNet Connect のサポートが自動的に無効化され、ISUS.obl がリンクされたスクリプト ライブラリの一覧から削除されます。
- FlexNet Connect の定数と関数はこれまで同様コンパイルされます。ただし、すべての関数は今回より ISERR_NOT_IMPLEMENTED を返します (**UpdateServiceGetAgentTarget** は除きます)。**UpdateServiceGetAgentTarget** はヌル (“”) を返します。
- FlexNet Connect 用の InstallScript 関数の呼び出しは、すべてデフォルトの InstallScript コードから削除されました。OnUpdateUIBefore イベント ハンドラーでは、**UpdateServiceOnEnabledStateChange** のコードが削除されました。OnMoveData イベント ハンドラーでは、**UpdateServiceRegisterProduct** と **UpdateServiceCreateShortcut** のコード ブロックが削除されました。

  InstallScript MSI プロジェクトを InstallShield 2008 以前から InstallShield 2025 へアップグレードしたとき、初期のバージョンのプロジェクトでこれらのイベントがオーバーライドされている場合、上記の関数が呼び出されます。これにより、問題が起きることはありません。関数は、現在、ISERR_NOT_IMPLEMENTED を返します。

  また、OnFirstUIAfter イベント ハンドラーが更新され、**SdFinishUpdate** の呼び出すオプションが除かれました。このオプションは以前デフォルトで無効にされています。
- すべての FlexNet Connect に関する再起動の後に実行される機能が削除されました。このため、製品は、再起動の後、FlexNet Connect に登録されていません。
- 今回より、InstallScript プロジェクトをビルドしたとき、FlexNet Connect に関するファイルはメディアに追加されません。
- ISUS.obl は、サポートされなくなるため、製品で提供されなくなります。

InstallScript MSI プロジェクトにおける FlexNet Connect のサポート

次の FlexNet Connect 関連の関数は、今後 InstallScript MSI プロジェクトでは推奨されません:

- **GetUpdateStatus** – この関数は常に FALSE を返します。
- **SetUpdateStatus** –この関数は ISERR_NOT_IMPL を返します。
- **GetUpdateStatusReboot** – この関数は常に FALSE を返します。
- **SetUpdateStatusReboot** – この関数は ISERR_NOT_IMPL を返します。

SetUpdateStatus と **SetUpdateStatusReboot** 関数へのすべての呼び出しは、デフォルトの InstallScript コードから削除されました。

基本の MSI プロジェクトと InstallScript MSI プロジェクトにおける複数言語サポート

インストールが複数言語をサポートし、(製品が既にインストールされている状態で) エンド ユーザーが Setup.exe を再び起動してインストールをメンテナンス モードで実行した場合、言語選択ダイアログが再度表示されることはありません。メンテナンス ダイアログは、製品をインストールしたときに使用された言語で表示されます。

同様に、インストール済みの製品の以前のバージョンをアップデートする複数言語のマイナー アップグレードまたはスモール アップデートが実行されたとき、言語選択ダイアログは表示されません。アップグレード ダイアログは、製品をインストールしたときに使用された言語で表示されます。

これにより、初回でインストールされず、メンテナンス モード、またはアップグレードでインストールされるその他の機能を追加するときだけでなく、元のインストールが実行され、その機能がインストールされるときに使用された言語が、元のインストールの修復でも使用されるようになります。

InstallScript における複数インスタンスのサポート

言語の選択ダイアログを表示する複数言語、複数インスタンス InstallScript インストールで、インスタンスの選択ダイアログが言語の選択ダイアログの前に表示されるようになりました。ダイアログの順番が変更されたことにより、エンドユーザーが製品の新しいインスタンスに異なる言語を選択することができるようになりました。インスタンスの選択ダイアログで、エンドユーザーが既存のインスタンスを保持することを指定した場合、言語の選択ダイアログは表示されません。インスタンスの選択ダイアログが表示されるとき、言語の選択ダイアログが表示されるときと同じ言語で表示されます。

非複数インスタンス インストールの複数のインスタンスが、Setup.exe /ig コマンドライン パラメーターを通してインストールされた場合、言語の選択ダイアログは初回インストールでのみ表示され、メンテナンス モードでは表示されません。これにより、元のインストールが実行され、その機能がインストールされたときに使用された言語が、元のインストールの修復でも使用されるようになります。また、初回でインストールされず、メンテナンス モードでインストールされた機能を追加するときにも、この言語が使用されます。

サイレント モードまたは Setup.exe -hide_usd コマンドライン パラメーターと共に実行されるインストールなど、インスタンスの選択ダイアログを表示しない InstallScript の完全リリース インストールが実行されるとき、新しいインスタンスが自動的にインストールされます。この動作は、InstallShield 2025 におけるすべてのプロジェクト、および以前のバージョンから InstallShield 2025 にアップグレードされたプロジェクトで発生します。InstallShield 2008 では、/hide_usd が使用されたとき、初回でインストールされたインスタンスが保持され、サイレント モードが実行されたとき、新しいインスタンスがインストールされていました。InstallShield 12 以前では、初回でインストールされたインスタンスが、/hide_usd が使用されたときも、また、サイレント モードが実行されたときも保持されていました。

差分メディアでは、読み込まれた初回のインスタンスが更新されます。

Setup.exe /ig コマンドライン パラメーターを使用してインストールされたインスタンスが 2 つ以上あるシステムで非複数インスタンスが実行されたとき、インスタンスの選択ダイアログが表示されます。非複数インスタンスのインストールでは、新しいインスタンスに対して新しい GUID がランダムに生成されないため、インスタンスの選択ダイアログでは新しいインスタンスをインストールできません。インスタンスの選択ダイアログには、GUID が製品コードに一致するインスタンスがインストールされていて、かつ、それがメンテナンス可能なインスタンスの 1 つである場合、それが「デフォルト」のインスタンスとして表示されます。この動作は、InstallShield 2025 におけるすべての新規プロジェクト、および以前のバージョンからアップグレードされたプロジェクトに適用します。

インスタンスが 1 つのみインストールされている場合、インストール済みインスタンスが自動的にメンテナンスされる試みが行われ、インスタンスの選択ダイアログは表示されません。これは、インストールされているインスタンスがデフォルトのインスタンスではない場合も当てはまります。この動作の結果、デフォルトのインスタンスがインストールされていなく、非デフォルトのインスタンスが (/ig パラメーターが使用されて) インストールされている場合、デフォルトのインスタンスは、(a) 最初に非デフォルトのインスタンスをアンインストールするか、または、(b) /ig パラメーターを使ってデフォルト インスタンスの GUID を指定しないかぎり、インストールまたはメンテナンスできません。したがって、/ig パラメーターを使った複数インスタンスのインストールを可能にする場合、常にこのパラメーターを使用することをお勧めします。この動作は、InstallShield 2025 におけるすべての新規プロジェクト、および以前のバージョンからアップグレードされたプロジェクトに適用します。

InstallShield 2008 では、非複数インスタンスのインストールが、複数のインスタンスがインストールされているターゲット システムで実行された場合、インスタンスの選択ダイアログは表示されませんでした。このようなケースでは、デフォルトのインスタンスが常に使用されていました。InstallShield 12 以前では、非複数インスタンスのインストールが、複数のインスタンスがインストールされているターゲット システムで実行された場合、インスタンスの選択ダイアログは表示されましたが、非複数インスタンスのインストールでは新しいインスタンスに対して新しい GUID がランダムに生成されないため、インスタンスの選択ダイアログでは新しいインスタンスを表示できませんでした。しかしながら、ダイアログのタイトル バーで、エンドユーザーが更新するインスタンスを選択する必要があるというメッセージが表示されていました。エンドユーザーが更新またはメンテナンスを行うインスタンスを選択する必要があるというメッセージが本来表示されるべきでした。

InstallScript MSI インストールにおける SdRegisterUser、SdRegisterUserEx、SdCustomerInformation、およびSdCustomerInformationEx の変更

InstallScript MSI インストールで、次の InstallScript テキスト置換が Windows Installer プロパティに直接マップされるようになりました。

- IFX_PRODUCT_REGISTEREDOWNER は Windows Installer プロパティ USERNAME にマップされます。
- IFX_PRODUCT_REGISTEREDCOMPANY は Windows Installer プロパティ COMPANYNAME にマップされます。

このため、これらの InstallScript 変数が設定および取得されるとき、対応する Windows Installer プロパティが設定および取得されます。これらの変数のデフォルト値はレジストリから読み込まれず、Windows Installer プロパティがデフォルトとして使用されます。以前の機能を使用したい場合は、次のサンプルコードを使って変数を手動で設定します。

```
// 登録された所有者 (ログ ファイルからロード済みの値が "" の場合のみ、レジストリからロードする)
if( !StrLengthChars( IFX_PRODUCT_REGISTEREDOWNER ) ) then
    szValue = "";
    RegDBGetItem( REGDB_WINCURRVER_REGOWNER, szValue );
    if( StrLengthChars( szValue ) ) then
        IFX_PRODUCT_REGISTEREDOWNER = szValue;
    endif;
endif;

// 登録された会社 (ログ ファイルからロード済みの値が "" の場合のみ、レジストリからロードする)
if( !StrLengthChars( IFX_PRODUCT_REGISTEREDCOMPANY ) ) then
        szValue = "";
        RegDBGetItem( REGDB_WINCURRVER_REGORGANIZATION, szValue );
        if( StrLengthChars( szValue ) ) then
            IFX_PRODUCT_REGISTEREDCOMPANY = szValue;
        endif;
endif;
```

InstallScript MSI インストールで、様々な登録関連ダイアログ (**SdRegisterUser**、**SdRegisterUserEx**、**SdCustomerInformation**、および **SdCustomerInformationEx**) は、エンド ユーザーの選択に従って IFX_PRODUCT_REGISTEREDOWNER、IFX_PRODUCT_REGISTEREDCOMPANY、および IFX_PRODUCT_REGISTEREDCOMPANY を適切に設定するようになりました。これによって、対応する Windows Installer プロパティが自動的に更新されます。

登録関連ダイアログ関数 (**SdRegisterUser**、**SdRegisterUserEx**、**SdCustomerInformation**、または **SdCustomerInformationEx**) のいずれかで svName または svCompany のどちらかにヌル文字列 ("") が指定された場合、関数は (Windows Installer プロパティから判別した) 適切な変数をデフォルトとして使用します。以前、変数は svName と svCompany の両方がヌルの場合のみ使用され、**SdCustomerInformation** と **SdCustomerInformationEx** では、スクリプト変数がバイパスされて、Windows Installer プロパティから直接値が読み込まれました。

svSerial パラメーターにヌル文字列 ("") が指定されると、"シリアル番号" フィールドのデフォルト値が IFX_PRODUCT_REGISTEREDSERIALNUM に設定されるようになりました。以前、この場合はダイアログに値が表示されませんでした。

次の InstallScript 変数が新しく追加または変更されました:

- DISABLE_PERUSERBTN – この既存の変数は、通常は有効であるユーザーごとのラジオ ボタンが無効 (または非表示) であることを示します。この変数は、常に FALSE に初期化されます。以前、この変数は Windows 9.x システム上では TRUE に、その他のシステム上では FALSE に初期化されました。
- DISABLE_ALLUSERBTN – この既存の変数は、通常は有効であるすべてのユーザーのラジオ ボタンが無効 (または非表示) であることを示します。この変数は、常に FALSE に初期化されます。以前、オペレーティング システムが Windows 9x 以外で、エンド ユーザーが管理者またはパワー ユーザーではない場合、この値は TRUE に初期化されました。
- HIDE_DISABLED_BTNS – この新しい変数は、すべてのユーザーおよびユーザーごとのラジオ ボタンを無効ではなく非表示にすることを示します。デフォルト値は TRUE です。この変数が TRUE に設定されると、ラジオ ボタンのどちらかが無効である場合、両方のラジオ ボタンが非表示となります。

登録ダイアログは、今回より DISABLE_ALLUSERBTN の値に関わらず Windows 9x 上でユーザーごとのラジオ ボタンを自動的に無効 (または非表示) とします。以前、ユーザーごとのラジオ ボタンは DISABLE_PERUSERBTN が FALSE の場合のみ無効となりました。

登録ダイアログはまた、DISABLE_ALLUSERBTN の値に関わらず、エンド ユーザーが管理者またはパワー ユーザーでない場合に、すべてのユーザー ラジオ ボタンを自動的に無効 (または非表示) にします。この動作は、以前と変更はありません。

これらの変更は、InstallShield 2025 で作成されるすべての新しい InstallScript MSI プロジェクトだけでなく、以前のバージョンの InstallShield で作成され、InstallShield 2025 にアップグレードされた InstallScript MSI プロジェクトに適用します。

プロキシ サーバーのサポート

特定のファイルがターゲット システム上で必要な場合のみ、インストールがそのファイルをダウンロードするように構成できます。たとえば、Windows Installer エンジン、.NET Framework、および一部の InstallShield 前提条件が、一部またはほとんどのターゲット システム上に既存する可能性があります。これらのファイルをインストールに埋め込む代わりに、必要なファイルだけを実行時にダウンロードするようにプロジェクトを構成することができます。こうすることで、インストール全体のサイズを抑えることができます。

エンド ユーザーがプロキシ サーバーを使ってインターネットにアクセスする場合で、インストールがファイルをダウンロードするように構成されていると、今回より、インストールはダウンロード中に、Internet Explorer で手動で構成されたシステム プロキシ設定を使用するようになりました。これは、ターゲット システム上で別のブラウザーがデフォルトとして設定されている場合でも同じです。

InstallShield は、Internet Explorer の "設定を自動的に検出する" 設定をサポートしませんので、ご注意ください。(エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されているときに、インストールでファイルのダウンロードが必要な場合、ファイルのダウンロードができないため、インストールは失敗します。エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されている可能性があるとき、ダウンロードされるように構成する代わりに、すべてのファイルをインストールに埋めこんだ方が良い場合があります。ファイルが埋めこまれている場合、失敗は避けられます。)ただし、InstallShield は Internet Explorer の LAN 接続用にセットアップされた自動構成スクリプト機能をサポートします。

この動作は、InstallShield 2025 におけるすべての新規プロジェクト、および以前のバージョンで作成され、InstallShield 2025 にアップグレードされたプロジェクトでも同様です。

InstallShield 2008 以前では、デフォルト ブラウザーとして指定されているブラウザーで構成されているプロキシ サーバーの設定が使用されていましたが、場合によって失敗し、次のような問題が発生していました:

- Netscape 6 または 7 がデフォルト ブラウザーの場合、Netscape 4 の設定が使用される。Netscape 8 または 9 がデフォルト ブラウザーの場合、システム (Internet Explorer) の設定が使用される。
- Netscape 4 の設定が使用された場合、プロキシ サーバーの一覧のみが正しく読み込まれ、インポートされる。プロキシ バイパス一覧は読み込まれるが、正しくインポートされない。
- Internet Explorer 4 と互換性がない設定 (例、オートプロキシ スクリプト設定) はインポートされない。
- インストールでデフォルト ブラウザーの判別に使用する方法が Windows Vista と互換性がない。このため、Windows Vista で、デフォルト ブラウザーが正しく判別されない可能性があった。

1.x .msi パッケージを .cab ファイルにラップするための Web ダウンローダ オプションの削除

InstallShield では、.msi パッケージを .cab ファイルにラップできるオプションが使用できなくなりました。以前このオプションは、Windows Installer 1.1 または 1.2 を含む基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの Web ダウンローダ タイプのメディアで使用できました。Windows Installer 2.0 以降を使って、パッケージにデジタル署名を施すことが推奨されます。

InstallScript MSI プロジェクトにおける状態テキスト

InstallScript MSI プロジェクトの OnFirstUIBefore、OnMaintUIBefore、OnAdminInstallUIBefore、OnAdminPatchUIBefore、OnPatchUIBefore、OnUninstall、および OnResumeUIBefore イベント ハンドラーには、デフォルトで、すべての新しい InstallScript MSI プロジェクトに **SetStatusExStaticText** の呼び出しを含めます。この新しいデフォルトのコードは STATUSEX ダイアログの状態テキストを設定します。今回より、表示される状態テキストは、実行中の操作の種類 (初回インストール、メンテナンス、アンインストール、修復など) に適しています。以前、**SetStatusExStaticText** の呼び出しを手動で追加する必要があり、追加しなかったとき、表示された状態テキストは、実行中の操作に対応していませんでした。

InstallScript MSI プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードしたとき、初期のバージョンのプロジェクトで 1 つ以上の更新されたイベントがオーバーライドされている場合、**SetStatusExStaticText** コードを手動でこれらのイベントに追加する必要があります。

InstallScript 変数 SHELL_OBJECT_FOLDER の変更点

[ショートカット] ビューにあるフォルダーの "表示名" 設定で **SHELL_OBJECT_FOLDER** (InstallScript または InstallScript MSI プロジェクトの場合) または **<SHELL_OBJECT_FOLDER>** (InstallScript プロジェクトの場合) を指定できます。その後、ショートカットが作成される前にスクリプトで SHELL_OBJECT_FOLDER 変数を設定することにより、このフォルダーの表示名を実行時に定義することができます。通常、ショートカットはファイルの転送時に作成されます。

InstallScript MSI インストールでこの機能を使用する場合、[ショートカット] ビューに表示されるフォルダーの "キー名" 設定で指定される文字は、すべて大文字でなくてはなりません (例、NEWFOLDER1)。

SHELL_OBJECT_FOLDER は、初期化中に IFX_PRODUCT_NAME と同じ値に初期化されるようになりました。これらの変数は、一旦初期化が完了すると同期されません。したがって、片方の変数を変更してから、もう片方の変数も変更する必要がある場合、両方とも手動で変更しなければなりません。

また、この変更の一部として、InstallScript MSI プロジェクトのデフォルトの OnFirstUIBefore イベント ハンドラー コードから次のコード行が削除されました:

```
SHELL_OBJECT_FOLDER = @PRODUCT_NAME;
```

InstallShield 2008 以前で作成された InstallScript MSI プロジェクトで、デフォルトの OnFirstUIBefore コードが変更されている場合、このプロジェクトを InstallShield 2025 にアップグレードしたとき、この行は既存のコードから削除されません。

詳しくは、「SHELL_OBJECT_FOLDER」を参照してください。

InstallScript 関数 FeatureFileEnum における動作の変更

InstallScript 関数 FeatureFileEnum の szQuery パラメーターで疑問符 (?) をワイルドカード文字として使用すると、その関数で疑問符が正確に 1 文字分の代替として使用されるようになりました。以前のバージョンの InstallShield で作成したプロジェクトで、**FeatureFileEnum** で疑問符をワイルドカード文字として使用している場合、InstallShield 2025 にアップグレードする際、InstallScript コードを修正する必要がある場合があります。以前のバージョンの InstallShield では、関数は疑問符を 1 文字または 0 文字の代替として使用していました。

たとえば、新しい InstallScript プロジェクトをデフォルトの設定で作成して、次の InstallScript コードをプロジェクトに追加すると、実行時に表示されるダイアログでは、何も表示されません。

```
#include "ifx.h"

  LIST listFiles;

program

  listFiles = ListCreate( STRINGLIST );
  FeatureFileEnum( MEDIA, "DefaultFeature", "Default?Component", listFiles, NO_SUBDIR );
  SdShowInfoList( "", "", listFiles );

endprogram
```

以前のバージョンの InstallShield で作成されたインストールでは、DefaultComponent がダイアログで表示されていました。

すべての InstallScript インストールで更新 UI の表示が可能

すべての InstallScript プロジェクトで、更新ユーザー インターフェイス (UI) の表示が可能になりました。以前、更新 UI を無効にすることができる紛らわしいオプションが [一般情報] ビューと [プロジェクトの設定] ダイアログ ボックスで提供されていました。更新 UI が無効にされた場合、MEDIA_FLAG_UPDATEMODE_SUPPORTED フラグが MEDIA_FIELD_MEDIA_FLAGS で設定されず、OnSetUpdateMode イベント ハンドラーによって UPDATEMODE 変数が適切に設定されませんでした。この結果、更新 UI が表示されませんでした。

今回より、InstallShield テーブルで ISShowUpdateUI の値 (以前のオプションによって更新されていました) を変更しても、影響は出ません。

以前と同じ機能は、以下の方法で複製することができます:

- UPDATEMODE 変数を設定する前に、OnSetUpdateMode イベントをオーバーライドし、結果を返す関数を更新する。インストールで、更新 UI は表示されません。
- OnMaintUIBefore イベントをオーバーライドし、FeatureRemoveAllInMediaAndLog の呼び出しを FeatureRemoveAllInMedia に変更する。

Certified for Windows Vista 検証スイートの変更

インストールまたはマージ モジュールが Windows Vista ロゴ プログラムの要件を満たしているかどうかを判別するときに使用する 2 つの Certified for Windows Vista 検証スイートが改訂されました。これらのスイートは今回より InstallShield ICE (ISICE) のみで構成されています。マイクロソフトが作成した ICE (ICE01 ～ ICE99) は、インストール パッケージの完全 MSI 検証ツールおよびマージ モジュールのマージ モジュール検証スイートで提供されているため、取り除かれました。

また、両スイートの名前も変更されました。

- Certified for Windows Vista 検証スイート (および InstallShield ICE) は *InstallShield Certified for Windows Vista 検証スイート*に変わりました。
- Certified for Windows Vista マージモジュール検証スイート (および InstallShield ICE) は *InstallShield Certified for Windows Vista マージ モジュール検証スイート*に変わりました。

検証と Certified for Windows Vista ロゴ プログラムの詳しい情報については、次を参照してください:

- Windows ロゴ プログラムの要件
- プロジェクトの検証

アップデートされた InstallScript オブジェクトとオブジェクト テンプレート

InstallShield 2025 InstallScript プロジェクトで使用できる InstallScript オブジェクトと InstallScript オブジェクト テンプレートが大幅に改良されました。

インターネット接続が利用できる場合、InstallShield の [アップデートの確認] 機能を使って、使用中の InstallShield バージョン用の InstallScript オブジェクトとオブジェクト テンプレートのアップデートを取得できます。アップデートを確認するには、[ツール] メニューで [アップデートの確認] をクリックします。InstallShield がウィンドウまたはダイアログを起動して、アップデートを確認します。

新規プロジェクトに影響し、アップグレードされたプロジェクトに影響しない変更

このセクションでは、潜在的に新規プロジェクトに影響し、以前のバージョンからアップグレードされたプロジェクトには影響しない InstallShield の変更について説明されています。アップグレードされたプロジェクトの場合、手動による変更が必要になる場合があります。

Setup.exe および Update.exe ブートストラッパの Unicode バージョンと ANSI バージョン

InstallShield では、基本の MSI プロジェクトで Setup.exe セットアップ ランチャーを Unicode バージョンまたは ANSI バージョンのどちらで作成するかを指定できます。以前、基本の MSI プロジェクトにセットアップランチャーが含まれているとき、常に ANSI バージョンがビルドされていました。Unicode バージョンのビルドはサポートされていませんでした。

Unicode セットアップランチャーは、ターゲット システムで 2 バイト言語のための適切なコード ページが実行されているいないにかかわらず、セットアップランチャーのユーザー インターフェイスで 2 バイト文字を正しく表示することができます。ANSI セットアップランチャーは、ターゲット システムで適切なコード ページが実行されている場合のみ、セットアップランチャー ダイアログで 2 バイト文字を正しく表示します。適切なコード ページが実行されていない場合、これらのダイアログで 2 バイト文字が文字化けして表示されます。

InstallShield 2025 で、新しい基本の MSI プロジェクトのリリースを作成したとき、デフォルトのセットアップランチャーの種類は Unicode になります。また、InstallShield 2025 で新しいパッチ構成または新しい QuickPatch プロジェクトを作成したときも、デフォルトのアップデート ランチャの種類は Unicode になります。

基本の MSI プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、既存のリリースのセットアップランチャーの種類はすべて ANSI です。種類は、必要に応じてオーバーライドが可能です。

同様に、基本の MSI プロジェクトまたは QuickPatch プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、既存のパッチのアップデート ランチャの種類はすべて ANSI です。種類は、必要に応じてオーバーライドが可能です。

ダイナミック ファイル リンク

プロジェクトにダイナミック ファイル リンクを追加、または変更するとき、InstallShield がコンポーネントを作成する方法について、新しいベスト プラクティスを使用するか、これまでと同様にディレクトリごとに 1 つのコンポーネント作成するのかを指定できます。これらの方式は、基本の MSI プロジェクト、InstallScript MSI プロジェクトおよびマージ モジュール プロジェクトに適用されます。

で新しいダイナミック ファイル リンクを作成すると、InstallShield はデフォルトでベスト プラクティス メソッドを使用します。

InstallShield 2008 以前で作成されたすべてのダイナミック ファイル リンクは、[1 つのディレクトリごとに 1 つのコンポーネント] メソッドを使用します。ダイナミック ファイル リンクがあるプロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、InstallShield は、既存するダイナミック ファイル リンクのコンポーネントを作成するとき、継続して [ディレクトリごとに 1 つのコンポーネント] メソッドを使用します。アップグレードされたプロジェクトで作成した新しいダイナミック ファイル リンクには、デフォルトでベスト プラクティス メソッドが使われます。2 つのコンポーネント作成メソッドおよび適切なメソッドの選択方法については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。

IIS Web サイトと関連コンポーネント

InstallShield は、今回より、仮想ディレクトリを一切含まない IIS Web サイトのインストールをサポートできるようになりました。また、Web サイトをコンポーネントに関連付けることも可能です。これらの強化によって、Web サイトに関連付けられた仮想ディレクトリまたはコンポーネントがインストールされると、ターゲット マシン上にその Web サイトが作成されます。

InstallShield 2025 の [IIS の構成] ビューを使って新しい Web サイトを追加すると、InstallShield は自動的にその Web サイトをコンポーネントに関連付けます。IIS Web サイトを含む InstallShield 2008 以前のプロジェクトをアップグレードした場合、その Web サイトは自動的にコンポーネントに関連付けられません。Web サイトとコンポーネントを関連付けるには、[IIS の構成] ビューでその Web サイトを選択したときに表示される [全般] タブを使います。

今回より、Web サイトがコンポーネントと関連付けられている場合、Web サイトの **[アンインストール時に Web サイトを削除する]** チェックボックスは、そのコンポーネントの "パーマネント" 設定 (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合) または "アンインストール" 設定 (InstallScript プロジェクトの場合) に対応します。つまり、Web サイトの **[アンインストール時に Web サイトを削除する]** チェック ボックスを選択またはクリアすると、自動的にコンポーネントの "パーマネント" 設定または "アンインストール" 設定が適切に更新されます。

QuickPatch パッケージの簡素化

QuickPatch プロジェクトの [詳細] タブにある新しい "QuickPatch の簡素化" 設定は、InstallShield で QuickPatch パッケージのビルド方法を決定します。簡素化された QuickPatch には、簡素化されていない QuickPatch パッケージに比べ、より少ない数の新しいサブ機能とカスタム アクションが含まれています。

場合によって、QuickPatch パッケージの簡素化ができない場合があります。たとえば、インストール済みのファイルを削除するように QuickPatch パッケージを構成した場合、簡素化は行えません。

新しい QuickPatch プロジェクトを作成したとき、"QuickPatch の簡素化" 設定のデフォルト値は [はい] に設定されています。ただし、QuickPatch プロジェクトを InstallShield 2008 以前から InstallShield 2025 にアップグレードした場合、この設定は [いいえ] となっています。この値は、必要に応じて変更が可能です。詳細については、「QuickPatch パッケージを簡素化するかどうかを指定する」を参照してください。

SetARPINSTALLLOCATION カスタム アクションのデフォルト条件

デフォルトで、すべての新しい基本の MSI プロジェクトおよび InstallScript MSI プロジェクトには、ビルトイン InstallShield カスタム アクション SetARPINSTALLLOCATION が含まれています。このカスタム アクション (ARPINSTALLLOCATION プロパティの値を製品のプライマリ フォルダーの完全修飾パスに設定します) は、[インストール]–[実行] シーケンスにシーケンスされ、条件を持ちません。InstallShield 2008 以前では、このカスタム アクションのデフォルト条件は Not Installed でした。デフォルトの [Not Installed] 条件を使用すると、カスタム アクションはメンテナンス モード中に実行されず、その結果 ARPINSTALLLOCATION プロパティの値が空白になります。InstallShield 2008 以前から InstallShield 2025 にアップグレードしたプロジェクトでこの動作を回避するには、[カスタム アクションとシーケンス] ビューを開いて、このカスタム アクションの "インストール実行条件" 設定から Not Installed 値を削除します。

MigrateFeatureStates アクションの新しいシーケンスの場所

デフォルトで、すべての新しい基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、MigrateFeatureStates アクションは CostFinalize アクションの直後ににシーケンスされるようになりました。[SQL スクリプト] ビューを使って、SQL サポートを新しいプロジェクトに追加すると、InstallShield ビルトイン カスタム アクション ISSQLServerFilteredList は、今回より、MigrateFeatureStates アクションのすぐあとにスケジュールされます。InstallScript 2008 以前では、ISSQLServerFilteredList は MigrateFeatureStates アクションの前にシーケンスされていたため、ランタイム エラー 2601 が発生していました。InstallShield 2008 以前から InstallShield 2025 にアップグレードされたプロジェクトでこのエラーを回避するには、[カスタム アクションとシーケンス] ビューを開いて、[インストール] – [ユーザー インターフェイス] シーケンスで ISSQLServerFilteredList アクションを MigrateFeatureStates の後に移動します。

"テンプレートの概要" プロパティの変更

[一般情報] ビュー、または [リリース] ビューの製品構成で "テンプレートの概要" プロパティの 値を指定できます。たとえば、複数のプラットフォームまたは複数のセミコロンを使用するなど、無効な値を入力しようとすると、InstallShield は警告メッセージ ボックスを表示して、プロパティ値の変更を阻止します。

InstallShield の以前のバージョンでは、無効な値を入力することが可能でした。InstallShield 2008 以前のプロジェクトを InstallShield 2025 にアップグレードした場合に、無効な値が自動的に変更されることはありませんが、必要に応じて、値を手動で更新することができます。詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。

削除されたウェルカム アシスタント

InstallShield を初めて開いたときに表示されていたウェルカム アシスタントは今回より含まれていません。

InstallShield MSIPackageDiff から InstallShield MSI Diff への変更

InstallShield MSIPackageDiff と呼ばれるツールは、今回より InstallShield には含まれていません。InstallShield MSI Diff と呼ばれる、よりパワフルなツールに変更されています。このツールは、[ツール] メニューで [差分] をポイントして InstallShield MSI Diff をクリックして、InstallShield 内からも起動できます。

詳細については、「InstallShield MSI Diff」を参照してください。

プロジェクトを InstallShield 12 以前からアップグレードする

以下の情報は、InstallShield 12 以前から InstallShield 2025 へアップグレードされたプロジェクトに影響が出る可能性がある変更についての説明です。

以前のバージョンの InstallShield で作成されたプロジェクトをアップグレードする

InstallShield 2025 を使って以前のバージョンで作成されたプロジェクトを開くと、プロジェクトを新しいバージョンに変換するかどうかを質問するメッセージ ボックスが表示されます。[変換する] を選択すると、変換が行われる前に、例えば .765 というファイル拡張子が付加されたプロジェクトのバックアップ コピーが作成されます。以前のバージョンの InstallShield でこのプロジェクトを再度開く場合、元のプロジェクトのファイル名から .765 を取り除いてください。InstallShield 2025 プロジェクトを以前のバージョンの InstallShield で開くことはできませんので注意してください。

InstallShield 12 以前、InstallShield DevStudio、InstallShield Professional 7 以前、および InstallShield Developer 8 以前のバージョンの InstallShield で作成された既存プロジェクトを InstallShield 2008 にアップグレードできます。InstallShield MultiPlatform または InstallShield Universal で作成されたプロジェクトは InstallShield 2008 にアップグレードすることはできませんので、ご注意ください。

ターゲット システムにおける Windows 9x、Windows NT 4、および Windows Me のサポートの終了

InstallShield では、今回のリリースより、Windows 9x、Windows NT4、および Windows Me システムのためのインストールを作成することができなくなりました。これらのオペレーティング システムを使用しているエンドユーザーが InstallShield 2025 でビルドされたインストールを実行しようとしたときに、エンドユーザーがレガシー オペレーティング システム上でインストールを実行するのを防ぐ起動条件がプロジェクトに含まれていない場合、予期しない結果が発生する可能性があります。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、エンド ユーザーがレガシー システム上でインストールを実行するとメッセージを表示する起動条件を含めることができます。InstallScript プロジェクトでは、これらのレガシー システムを確認して、それが存在する場合にメッセージを表示する SYSINFO 構造変数を使用する InstallScript コードを追加することができます。

詳しくは、次を参照してください:

- プロジェクト アシスタントでオペレーティング システムを指定する
- [プラットフォーム] タブ
- [プラットフォーム] ダイアログ ボックス
- [プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル

COM 抽出

今回より、InstallShield を使用して COM サーバーから COM 情報を抽出すると、データは TypeLib テーブルではなく Registry テーブルに書き込まれます。マイクロソフト社は TypeLib テーブルを使用しないことを強く推奨しています。

デフォルトで、ビルド時に未使用のディレクトリを .msi ファイルから自動的に削除する

InstallShield 12 以前のバージョンを使って作成した基本の MSI プロジェクト、InstallScript MSI プロジェクト、またはマージ モジュール プロジェクトを InstallShield 2025 にアップグレードすると、[リリース] ビューの [ビルド] タブに新しく追加された "未使用のディレクトリを保持する" 設定は、デフォルトで [いいえ] に設定されています。したがって、Directory テーブルの Directory 列に一覧表示されているディレクトリが .msi ファイル内の既知の場所で参照されない場合、ビルド時に InstallShield が作成する .msi ファイルの Directory テーブルからそのディレクトリは削除されます。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、これはマージ モジュールがマージされてから削除されますが、.msi ファイルに存在するディレクトリのみが削除の対象となります。したがって、マージ モジュールの Directory テーブルに新しい未使用のディレクトリが含まれている場合、そのディレクトリはインストールに追加されます。

ALLUSERS と CustomerInformation ダイアログの変更

InstallShield 2008 より、新しい基本の MSI プロジェクトでは、デフォルトで ALLUSERS プロパティに 1 が設定されます。ほとんどのインストールは、マシンごとに管理者権限を使用して実行される必要があるため、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、ALLUSERS プロパティの値は自動的に変更されません。また、このプロパティが以前のプロジェクトで定義されていない場合も、自動的に追加されません。

また installShield 2008 から、デフォルトで、すべての新しい基本の MSI プロジェクトの CustomerInformation ダイアログは、エンドユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーのみにインストールするかを指定できるラジオ ボタン グループを表示しないようになっています。このダイアログについては、これが推奨される実装です。

InstallShield 12 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードしたとき、CustomerInformation ダイアログは自動的に変更されません。

詳しくは、次を参照してください:

- ユーザーごとのインストールとマシンごとのインストールの違い
- ALLUSERS

セットアップ前提条件の Windows Server 2003 条件および 64 ビット Windows XP 条件

Windows Server 2003 および 64 ビット Windows XP のオペレーティング システムのバージョン番号は、どちらも 5.2 です。そのため、InstallShield 12 で作成され、かつ Windows Server 2003 を必要とする前提条件を 64 ビット Windows XP システムにインストールすることはできますが、InstallShield 12 で作成され、かつ Windows XP を必要とする前提条件は、64 ビット Windows XP システムにインストールすることはできません。

この問題を解決するため、強化された InstallShield 2008 セットアップ前提条件エディターでは、ターゲット システムがワークステーション、サーバー、またはドメイン コントローラーのいずれかである必要があるかどうかを指定できるようになりました。

Windows Server 2003 要件または 64 ビット Windows XP 要件を含む既存の前提条件については、これを InstallShield 2008 のセットアップ前提条件エディターで開き、[条件] タブで必要に応じて条件を更新することで、問題を回避することができます。[**前提条件を実行するプラットフォームを選択します**] ボックスで、適切なオペレーティング システム要件を選択します。この処理を適切に行なうことで、新しい "製品 (OS) タイプ" 設定がワークステーション、サーバー、またはドメイン コントローラー値に正しく設定されます。

オートメーション インターフェイス

"[保存オプション] ダイアログの表示" 設定が [リリース] ビューから削除されました。したがって、この設定に対応する WebSaveOptionsDlg プロパティが、オートメーション インターフェイスの ISWiRelease オブジェクトで使用できなくなりました。

リリースの "キャッシュ パス"設定における新しいデフォルト値

今回より、[リリース] ビューの "キャッシュ パス" 設定における圧縮リリースのためのデフォルト値が [LocalAppDataFolder]Downloaded Installations に設定されています。以前のデフォルト値 [WindowsFolder]Downloaded Installations は、ロックされたシステムで使用できないことがあります。プロジェクトを InstallShield 12 以前から InstallShield 2025 に移行したとき、"キャッシュ パス" 設定は自動的に変更されないため、必要な場合、この値を手動で変更します。

InstallScript One-Click Install インストール

Setup.exe が InstallScript One-Click Install インストールのセットアップ プレーヤーとして使用されなくなりました。代わりに Setup.ocx が今回より使用されます。Setup.ocx ファイルをインストールに含めるには、[リリース] ビューに新しく追加された "One-Click Install の生成" 設定が [はい] に設定されている必要があります。InstallScript プロジェクトを InstallShield 12 以前から InstallShield 2025 にアップグレードするとき、[リリース] ビューの "デフォルト Web ページの作成" 設定が [はい] に設定されていると、アップグレード中 "ClickOnce Install の生成" 設定は自動的に [はい] に設定されます。反対に、"デフォルト Web ページの作成" 設定が [いいえ] に設定されていて、インストールをインターネット上で配布する予定の場合は、プロジェクトをアップグレードしたあと、"ClickOnce Install の生成" 設定を手動で [はい] に設定する必要があります。

[セットアップを保存/実行] ダイアログ ボックス関連の機能はすべて、InstallShield から削除されました。エンド ユーザーがインストールをダウンロードおよび保存 (またはデスクトップにアイコンを作成) できるオプションを提供するには、これをインストールの Web ページ内に実装して、Web ページを通してダウンロードおよび保存操作を処理する必要があります。

起動された One-Click Install インストールにデータを渡すには、is::CmdLine parameter と CMDLINE スクリプト変数を使用します。以前、Web ページからインストールにデータを渡すことが可能でしたが、このサポートは削除されました。SetProperty の最初のパラメーターに有効な唯一の値は、is::CmdLine です。

詳しくは、「InstallScript プロジェクトの One-Click Install インストール」を参照してください。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト用のパッチの作成

今回のリリースより、InstallShield はパッチの作成に Patchwiz.dll のバージョン 3.1 を使用します。

DemoShield のサポート

DemoShield は現在販売されていません。サポートも終了しました。このため、今後 InstallShield に DemoShield は統合されません。

Windows Vista 検証スイートにおける変更

インストールが Windows Vista ロゴ プログラムのインストール要件を満たしているかどうかを判別するときに使用する検証スイートの名前が変更されました。

- Windows Vista Quality 品質検証スイート (および InstallShield ICE) の新名称は *Certified for Windows Vista 検証スイート (および InstallShield ICE)* です。
- マージ モジュール品質検証スイート (および InstallShield ICE) の新名称は *Certified for Windows Vista マージ モジュール検証スイート (および InstallShield ICE)* です。

また、3 つの InstallShield ICE が InstallShield ベスト プラクティス スイートに移されました。

検証と Certified for Windows Vista ロゴ プログラムの詳しい情報については、次を参照してください:

- Windows ロゴ プログラムの要件
- プロジェクトの検証

InstallShield ベスト プラクティスとして提供されている検証ツールについての詳細は、次を参照してください:

- ISBP17 (以前は ISICE13 という名前でした)
- ISBP18 (以前は ISICE14 という名前でした)
- ISBP19 (以前は ISICE15 という名前でした)

InstallShield 11.5 以前のプロジェクトをアップグレードする

InstallShield 12 では、セキュリティ関連の問題 (COM/DCOM)、およびその他の様々な理由で起こったインストールの失敗を解決するため、InstallScript MSI アーキテクチャが大幅に改良されました。InstallShield 12 でアーキテクチャが大幅に改良された理由は、これらの問題を解決して InstallScript MSI プロジェクトの安定性を高めるためです。この改良によって、InstallScript カスタム アクションを含む基本の MSI プロジェクト、および InstallScript の安定性も高まりました。

次の表には、新旧のデザインの違いが比較されています。

テーブル 9 ・ アーキテクチャの変更

InstallShield 11.5 以前	InstallShield 12 以降
各 InstallScript カスタム アクションは ISScriptBridge.dll (C++ MSI DLL) によって処理されます。この処理により、実際のスクリプトの実行が実行中の IDriver.exe プロセスに渡されます。	各 InstallScript カスタム アクションは、ISSetup.dll (単一の C++ MSI DLL) によって処理されます。これには完全な InstallScript エンジンが含まれます。
エンジン ファイルのセットが必要。これらは、プライマリ インストールが開始する前に ISScript.msi によってインストールされなくてはなりません。	エンジン ファイルまたは ISScript.msi ファイルは不要。ISSetup.dll には、必要な情報がすべてが含まれています。

テーブル 9 • アーキテクチャの変更 (続き.)

InstallShield 11.5 以前	InstallShield 12 以降
すべての InstallScript カスタム アクションは、共有 IDriver.exe プロセスで実行します。IDriver.exe プロセスは最終のシャットダウン カスタム アクションが必要になるまで存在し続けます。	各 InstallScript カスタム アクションは非依存型で、独立したライフサイクルを持ちます。カスタム アクション エントリ ポイントから、ISSetup.dll がスクリプト エンジンを開始し、関連スクリプトを実行してから、エンジンをシャットダウンします。

InstallShield 12 で導入されたアーキテクチャの変更における利点

InstallShield 12 以降のアーキテクチャには、以前のアーキテクチャと比較して次のような利点があります。

- 実行中の IDriver.exe の検出に関連するエラー 1607、1608、またはその他の COM/DCOM 実行時エラーをエンドユーザーが経験することはありません。以前のモデルでこれらのエラーを解決するためには、場合によって DCOM 設定の変更が必要なため、容易ではありませんでした。また実行中のオブジェクト テーブルに依存していたことで、″ユーザーの簡易切り替え″ および Windows Terminal Services などのすべての利用シナリオにおいて、モデルの脆弱性をもたらしていました。
- エンジン バインディングはスタティックです。Program Files¥Common Files¥InstallShield に、共有エンジンファイルはインストールされません。11.5 以前のバージョンの InstallShield では共有エンジン ファイルに依存するため、モデルに脆弱性をもたらしました。これは個別のインストールが隔離されていないためです。1 つのエンジン バージョンを変更すると、既存のインストールを破損する可能性があります。特にインストーラーの隔離は安定性の重要な鍵となります。
- ISScript.msi (InstallScript エンジン インストーラー) ファイルが必要なくなりました。以前のモデルでは、プライマリ .msi ファイルの前に ISScript.msi を実行する必要がありました。完全アプリケーションが単一の .msi ファイルに含まれていないため、InstallScript カスタム アクションがない基本の MSI インストールでは、インストール時にブートストラップが必要でした。Active Directory を使って環境管理を行っている企業にとって、これは柔軟性に欠けていました。また、メインの .msi ファイルに必要な情報がすべてが含まれていることを期待している上級ユーザーを困惑させる原因にもなりました。InstallScript MSI インストールでは、要件が以前のアーキテクチャと変わらないようにするため、これまで同様ブートストラップが必要です。ただし、InstallScript カスタム アクションがある基本の MSI インストールでは、ブートストラップは必要ありません。
- インストールの実行に必要な InstallShield カスタム アクションの数が少なくなりました。以前のモデルは、エラーの起きやすいスタートアップおよびシャットダウン カスタム アクションの密接な結合に依存していました。必要なカスタム アクションのうちの 1 つが削除または移動された場合、インストールが適切に動作しませんでした。

InstallShield 12 で導入されたアーキテクチャの変更におけるマイナス点

InstallShield 12 以降のアーキテクチャには、次のようなマイナス点があります。

- InstallScript イベントは基本の MSI プロジェクトおよびマージ モジュール プロジェクトでは利用できないため、これらのプロジェクトの InstallScript コードは、InstallScript カスタム アクションを使って実行しなくてはなりません。グローバル変数は、これらのカスタム アクションが起動されたときの状態を共有しません。したがって、InstallScript カスタム アクションでグローバル変数を宣言してから、スクリプトでグローバル変数を使用します。ただし、基本の MSI インストールに 2 番目の InstallScript カスタム アクションが含まれている場合、最初のスクリプトで宣言されたグローバル変数を 2 番目のスクリプトで使用することはできません。

以前のモデルでは、InstallScript 開発者は基本の MSI プロジェクトですべての InstallScript イベントのスクリプト コードおよび InstallScript グローバル変数の共有状態を使用することができました。このため、セットアップ作成者は、インストール全体にわたって総体的な視点および自然に流れる感覚でカスタム アクションを作成することができました。しかしながら、プログラミングでは一般的にグローバル変数を使用しないほうがいいため、この新しい制限によって、実際にはより良い InstallScript コードが書かれることになります。

- メモリ内のオブジェクトはカスタム アクションの呼び出し間で共有できるよう、シリアル化しなくてはなりません。以前のモデルでは、InstallScript 開発者はグローバル変数に複雑なオブジェクト ベースのデータを格納することができました。前述の利点の通り、これによってより良い InstallScript コードとなります。

InstallShield 11.5 以前のプロジェクトのアップグレード

InstallShield 12 で導入された広範囲にわたる再アーキテクチャは、InstallShield 11.5 以前で作成されたプロジェクトを InstallShield 2025 にアップグレードするとき、場合によって、手動で変更を行う必要があります。

タスク ***プロジェクトをアップグレードするには、以下の手順に従います:***

1. プロジェクトを変換する前に、プロジェクト ファイルと InstallScript ファイルのバックアップ バージョンを作成してください。
2. InstallShield 2025を開きます。
3. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. [参照] をクリックして、アップグレードする InstallShield 11.5 以前のプロジェクトのプロジェクト ファイル (.ism) を選択してください。プロジェクトをアップグレードするかどうかを指定するように求めるダイアログ ボックスが開きます。
5. [はい] をクリックします。

この後、プロジェクトがアップグレードされ、プロジェクト ファイル (.ism ファイル) のバックアップ コピーが保存されます。

場合によって、アップグレードされたプロジェクトは、手動による変更が必要です。詳しくは、次を参照してください。

- InstallScript カスタム アクションを含む InstallShield 11.5 または以前の基本の MSI プロジェクトをアップグレードする
- InstallShield 11.5 以前の InstallScript MSI プロジェクトをアップグレードする
- InstallShield 11.5 以前の InstallScript プロジェクトをアップグレードする
- InstallScript カスタム アクションを含む InstallShield 11.5 または以前の QuickPatch プロジェクトをアップグレードする
- InstallShield 11.5 以前の InstallScript MSI プロジェクトに標準のパッチを作成する
- InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトまたはこの種類のオブジェクトを含むプロジェクトをアップグレードする

InstallScript カスタム アクションを含む InstallShield 11.5 または以前の基本の MSI プロジェクトをアップグレードする

InstallShield 11.5 以前で作成された基本の MSI プロジェクトを InstallShield 2025 にアップグレードするときの詳しい説明については、以下のセクションを参照してください。

グローバル変数、グローバル ポインター、および SUPPORTDIR

InstallShield 12 以降では、基本の MSI インストールで InstallScript カスタム アクションが実行されたとき、コンパイルされた InstallScript が、アクションが呼び出される前にロードされ、アクションが完了された後、アンロードされます。このため、各 InstallScript カスタム アクションは、それぞれのセッションで実行され、実行ごとに全 InstallScript エンジンがロードおよびアンロードされます。この動作は、InstallShield 11.5 以前での動作とは異なります。InstallShield 11.5 以前では、コンパイルされた InstallScript は、InstallScript で使用された最初の InstallScript カスタム アクションが実行される前に一度だけロードされ、すべての InstallScript カスタム アクションが完了したあと、インストールの最後でアンロードされます。

個々の InstallScript カスタム アクションの呼び出し間で、グローバル変数とポインターが保持されなくなったことが、この動作の変更による最も大きな影響です。

- 複数のカスタム アクションの呼び出し間で値を保存する必要がある場合、レジストリ、Windows Installer プロパティ、または外部データ ファイルなどの外部メカニズムを使用して、呼び出し間の情報を保存しなければなりません。遅延/コミット/ロールバック InstallScript カスタム アクションで Windows Installer プロパティを使用する選択をした場合、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」にあるガイドラインを参照してください。
- カスタム アクションの呼び出し間で COM オブジェクトまたは他のグローバル オブジェクトを使用する必要がある場合、個々のカスタム アクションの呼び出しのオブジェクトを初期化して、それを有効にする必要があります。

また、この変更により、個々のカスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化および使用されます。したがって、各カスタム アクションの起動には、それぞれ一意の SUPPORTDIR が用意されるため、SUPPORTDIR のファイルを使用して、個々の呼び出し間で情報を共有することはできません。情報は、FOLDER_TEMP または他のファイルの場所を使用して共有することができます。

FOLDER_TEMP のパスは、すべての InstallScript カスタム アクションに対して同一とは限りませんので注意してください。システム コンテキストで実行される InstallScript カスタム アクションと実行されない InstallScript カスタム アクションがある場合、パッケージが昇格された状態で実行された場合、それぞれ異なる一時パスが与えられます。InstallScript カスタム アクションは、異なるユーザーのコンテキストで実行されるため、一時ディレクトリにファイルを保存して、それをあとで取り出す操作は、特定のシナリオで適切に動作しないことがあります。

Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション

基本の MSI インストールの遅延/コミット/ロールバックの InstallScript カスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、InstallScript カスタム アクションで他のプロパティ (例、SUPPORTDIR) にアクセスする場合は、それらを CustomActionData に渡す必要があります。これを行うには、すぐにプロパティを設定するタイプのカスタム アクション (例、**MsiSetProperty** 関数をもつ即時 InstallScript カスタム アクション) をスケジュールして、カスタム アクションの名前に一致するプロパティを設定します。このプロパティの値が、遅延のカスタム アクション内の CustomActionData プロパティから取得できるようになります。

たとえば、SUPPORTDIR のようなプロパティにアクセスする場合、MyCustomActionName という名前で、MyCustomActionName プロパティを [SUPPORTDIR] に設定する即時カスタム アクションを作成して、MsiGetProperty 呼び出しで "SUPPORTDIR" を "CutomActionData" で置き換えることもできます。

MsiGetProperty(hMSI, "CustomActionData", szSupportDir, nLen)

詳細については、次を参照してください。

- 遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う
- 遅延実行カスタム アクションのコンテキスト情報を取得する

提供されていない Windows Installer プロパティを補うことを忘れると、インストール時に予期しない結果が起きることがありますので注意してください。たとえば、遅延アクションでは、MsiGetProperty(hMSI, "INSTALLDIR", szInstallDir, nLen) が szInstallDir を空の文字列に設定するため、szInstallDir ^ szFile は、[INSTALLDIR] のファイルの代わりに、現在のディレクトリにあるファイルを参照する場合があります。遅延カスタム アクションの現在のディレクトリは、通常 [SystemFolder] です。

遅延/コミット/ロールバック InstallScript カスタム アクションは、ProductLanguage プロパティへのアクセスがありません。インストールに複数言語のサポートが含まれていて、かつエンドユーザーがインストールを実行している言語へのアクセスが必要な遅延/コミット/ロールバック InstallScript カスタム アクションもあるとき、インストールは、この言語をインストールのデフォルト言語であると仮定します。これは、エンドユーザーがインストールをデフォルト言語以外の言語で実行するとき、問題となります。ただし、遅延/コミット/ロールバック カスタム アクションは通常、ユーザー インターフェイスを表示しないため、これが問題になることは、ごく稀にしかありません。

定義済み InstallScript イベント ハンドラーの関数

定義済みの InstallScript イベント ハンドラーの関数は今回より、InstallScript カスタム アクションを含む基本の MSI プロジェクトで使用できなくなりました。InstallShield 11.5 以前では、次の InstallScript イベント ハンドラーの関数が基本の MSI プロジェクトで提供されていました。

- OnBegin
- OnMoving
- OnMoved
- OnEnd

これらのイベント ハンドラーの関数は、InstallScript カスタム アクションがある基本の MSI プロジェクトではサポートされていません。一旦プロジェクトが InstallShield 12 以降で再ビルドされると、それ以後呼び出されなくなります。これらのイベント ハンドラー関数は、InstallShield 12 以降でも続けてコンパイル可能ですが、呼び出されることはありません。

これらのイベント ハンドラー関数をプロジェクトで使用する場合、これらの関数を呼び出すカスタム アクションを手動でスケジュールする必要があります。詳しくは、「基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」をご覧ください。

InstallShield InstallScript スクリプト作成エンジン マージ モジュール

InstallShield 12 より、InstallShield スクリプト作成エンジン マージ モジュールを、基本の MSI プロジェクトに含めることができなくなりました。InstallShield 11.5 以前では、このマージ モジュールを利用することにより、Setup.exe を含まないインストールで InstallScript エンジンを配布することができました。InstallShield 12 以降では、

Setup.exe が使用されるされないにかかわらず、InstallScript エンジンが、InstallScript カスタム アクションをもつ基本の MSI インストールに ISSetup.dll の一部として自動的に含められます。したがって、InstallShield 2025 プロジェクトで、このマージ モジュールを使う必要はありません。

InstallShield 2025 を使用して、このマージ モジュールがあるプロジェクトのリリースをビルドしようとすると、次のようなビルド エラーが発生することがあります。

ISDEV : エラー -4075: ファイルが見つかりません。機能 'NewFeature1' にモジュール 'InstallShieldScriptingEngine.4F635B62_07BF_4779_B74E_D80C29D508E3:0' をマージ中に、エラーが発生しました。

このビルド エラーを解決するには、このマージ モジュールをプロジェクトから削除します。マージ モジュールは、[再配布可能ファイル] ビューにあるチェック ボックスをクリアすると削除されます。

InstallScript エンジン ファイルへの変更

InstallShield 12 以降では、InstallScript カスタム アクションを含む基本の MSI プロジェクトは、InstallScript エンジン ファイルを次のディレクトリにインストールしません。

<COMMONFILES>¥InstallShield¥Driver¥<バージョン>¥Intel 32

InstallShield 11.5 以前の InstallScript カスタム アクションがある基本の MSI プロジェクトでは、いくつかのファイルは Binary テーブルに格納されていました。

- Setup.inx
- Isconfig.ini
- Isrt.dll
- ISScriptBridge.dll
- _isresXXXX.dll (ここで、XXXX は言語です。.dll は、インストールに含まれている各言語に対して 1 つ含まれました。)
- StringxXXXX.txt (ここで、XXXX は言語です。.txt は、インストールの各言語に対して 1 つ含まれました。)

InstallShield 12 以降では、これらのファイル (使われなくなった ISScriptBridge.dll を除く) はすべて、ISSetup.dll ファイル内に格納されます。ISSetup.dll は、Binary テーブルに格納される唯一のファイルです。

InstallScript エンジン ファイルの変更が原因によるアップグレードの問題は、現在、報告されていません。これらの変更は、情報提供のみを目的として報告されています。

基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール

InstallShield 12 以降で、定義済みの InstallScript イベント ハンドラーの関数は今回より、InstallScript カスタム アクションを含む基本の MSI プロジェクトで使用できません。InstallShield 11.5 以前では、次の InstallScript イベント ハンドラーの関数が、基本の MSI プロジェクトと InstallScript オブジェクト プロジェクトで提供されていました。

- OnBegin
- OnMoving
- OnMoved
- OnEnd

これらのイベント ハンドラーの関数は、InstallScript カスタム アクションがある基本の MSI プロジェクトではサポートされていません。一旦プロジェクトが InstallShield 2025 で再ビルドされると、それ以後呼び出されなくなります。これらのイベント ハンドラー関数は、InstallShield 2025 でも続けてコンパイル可能ですが、呼び出されることはありません。

同様に、これらの定義済み InstallScript イベント ハンドラー関数は、InstallShield 2025 で開かれたときに自動的にマージ モジュール プロジェクトに変換される InstallScript MSI オブジェクト プロジェクトではサポートされていません。これらの関数は、一旦変換されたマージ モジュール プロジェクトが InstallShield 2025 でビルドされると、呼び出されません。

基本の MSI または InstallScript MSI オブジェクト プロジェクトにこれらのイベントがあるときに、プロジェクトを InstallShield 2025 にアップグレードする場合、イベント ハンドラー関数を呼び出すカスタム アクションを手動でスケジュールする必要があります。以下は、その手順です。

タスク

定義済み InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを手動でスケジュールするには、以下の手順に従います:

1. InstallShield 2025 で、アップグレード済みのプロジェクトを開きます。
2. InstallScript ファイルへ適切な変更を行います。
   a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
   b. スクリプトで OnBegin、OnMoving、OnMoved、および OnEnd イベント ハンドラー関数を検索します。ビューの中央にあるペインで関数名をクリックすると、簡単に関数を検索することができます。
   c. 関数の名前を代替名に変更して、ifx.h に自動的に含まれる既存の関数プロトタイプと競合するのを避けるようにします。例:
      - MyOnBegin
      - MyOnMoving
      - MyOnMoved
      - MyOnEnd
   d. 既存の関数を、単一 HWND パラメーターを取得するようにアップグレードします。例:

      ```
      function MyOnBegin(hMSI) begin end;
      ```

   e. これらの新しい関数の適切なプロトタイプを追加します。

      ```
      export prototype MyOnBegin(HWND);
      export prototype MyOnEnd(HWND);
      export prototype MyOnMoved(HWND);
      export prototype MyOnMoving(HWND);
      ```

3. 名前が変更された InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを追加します:
   a. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]**（基本の MSI プロジェクトの場合）または **[カスタム アクション]**（マージ モジュール プロジェクトの場合）をクリックします。
   b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい InstallScript]** をクリックします。InstallScript カスタム アクションが追加されます。

c. カスタム アクションの名前を入力します。この名前には、InstallScript 関数の名前を変更したときに使用した名前を使用します。例:

- MyOnBegin
- MyOnMoving
- MyOnMoved
- MyOnEnd

d. 作成した新しい InstallScript カスタム アクションを選択します。

e. **"関数名"** 設定を、InstallScript 関数の名前に設定します。

f. **"スクリプト内実行"** 設定で、下のテーブルで示されている値を指定します。

4. InstallScript カスタム アクションをインストールの適切な箇所にスケジュールします。

a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

b. InstallScript カスタム アクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。**[アクションの挿入] ダイアログ ボックス**が開き、プロジェクトに現在関連付けられているアクションおよびダイアログの一覧が表示されます。

c. 作成した InstallScript カスタム アクションを選択します。必要に応じて、InstallScript イベントの起動についての条件を **[条件]** ボックスに入力します。

d. **[OK]** をクリックします。

以前の機能にできるだけ近づけるため、ステップ 3f のスクリプト内実行の設定と、ステップ 4b の InstallScript カスタム アクションのスケジュールを次のように行います。

テーブル 10 ・ カスタム アクションのスケジュール

カスタム アクション	スクリプト内実行	シーケンス中の場所
MyOnBegin	即時 (デフォルト)	[インストール]-[ユーザー インターフェイス] シーケンスの最初のアクションの 1 つです。
MyOnMoving	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallInitialize アクションと AllocateRegistrySpace アクションの間。
MyOnMoved	システム コンテキストの遅延	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間
MyOnEnd	即時 (デフォルト)	[インストール] の [実行] シーケンスの最後のイベント、InstallFinalize アクションの後(以前のリリースで、OnEnd は、インストールが完了した結果として他のすべてのイベントの後で呼び出されていました。)

次の事項に注意してください。

- グローバル変数とポインターは、個々の InstallScript カスタム アクションの呼び出し間で保持されなくなりました。また、個々の InstallScript カスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化され、使用されます。したがって、SUPPORTDIR のファイルでは、個々の呼び出し間で情報を共有することはできませ

ん。詳細については、「グローバル変数、グローバル ポインター、および SUPPORTDIR」を参照してください。

- ほとんどの Windows Installer プロパティは、遅延/コミット/ロールバック InstallScript カスタム アクションで使用することはできません。詳細については、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」を参照してください。

InstallShield 11.5 以前の InstallScript MSI プロジェクトをアップグレードする

InstallShield 11.5 以前で作成された InstallScript MSI プロジェクトを InstallShield 2025 にアップグレードするときの詳しい説明については、以下のセクションを参照してください。

グローバル変数、グローバル ポインター、および SUPPORTDIR

InstallShield 2025 およびそれ以前のバージョンでは、InstallScript MSI インストールが初期化されたとき、コンパイルされた InstallScript ファイルがロードされ、必要に応じてインストール中に、この "メイン" のロードされた InstallScript ファイルの InstallScript イベントが呼び出されます。

ただし、InstallShield 12 以降では、基本の MSI インストールで InstallScript 内の関数が Windows Installer エンジンによって InstallScript カスタム アクションとして呼び出されると、カスタム アクションは、InstallScript カスタム アクションと同じ動作をします。つまり、関数は、完全に別々にロードされたスクリプト インスタンスで実行されます。スクリプトは、アクションが呼び出される前にロードされ、アクションが完了した後で、アンロードされます。このため、Windows Installer エンジンによって、InstallScript カスタム アクションとして呼び出された各 InstallScript 関数は、それ自身のセッションで実行され、実行ごとに全 InstallScript エンジンがロードおよびアンロードされます。この動作は、InstallShield 11.5 以前での動作とは異なります。InstallShield 11.5 以前では、カスタム アクションは、初期化中にロードされ、インストールの最後でアンロードされたロードされた単一のメイン スクリプト ファイル インストールで呼び出されていました。

この動作の変更による主な影響は、InstallScript カスタム アクションとして呼び出された InstallScript 関数は、メインの実行スクリプト ファイルで保持されるグローバル変数およびポインターへアクセスしなくなったいうことです。

- メインの実行 InstallScript ファイルとインタラクトして、グローバル変数の使用や変更、またはメインの実行スクリプト ファイル によって保持されているオブジェクトの変更などのタスクを行う必要がある場合 (アンインストールのステータス バージョンの更新や、スクリプト操作のログ記録を含みます)、関数は、InstallScript カスタム アクションとしてではなく、InstallScript イベントから呼び出されなければなりません。InstallScript イベントから呼び出された関数は、メインの InstallScript インスタンスで実行されます。(例外: いくつかの OnFilesInUse などの InstallScript エラー イベントは、Windows Installer エンジンからのエラー メッセージの結果として呼び出されます。Windows Installer エンジンは、これらの InstallScript エラー イベントを InstallScript カスタム アクションとして起動します。このため、InstallScript カスタム アクションの制限は、これらの InstallScript イベントにも適用します。)

- 複数の InstallScript カスタム アクションの呼び出し間で値を保存する必要がある場合、レジストリ、Windows Installer プロパティ、または外部データ ファイルなどの外部メカニズムを使用して、呼び出し間の情報を保存しなければなりません。遅延/コミット/ロールバック InstallScript カスタム アクションで Windows Installer プロパティを使用する選択をした場合、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」にあるガイドラインを参照してください。

- カスタム アクションの呼び出し間で COM オブジェクトまたは他のグローバル オブジェクトを使用する必要がある場合、個々のカスタム アクションの呼び出しのオブジェクトを初期化して、それを有効にする必要があります。

また、この変更により、個々のカスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化および使用されます。したがって、各カスタム アクションの起動には、それぞれ一意の SUPPORTDIR が用意されるため、SUPPORTDIR のファイルを使用して、個々の呼び出し間で情報を共有することはできません。情報は、FOLDER_TEMP または他のファイルの場所を使用して共有することができます。

FOLDER_TEMP のパスは、すべての InstallScript カスタム アクションに対して同一とは限りませんので注意してください。システム コンテキストで実行される InstallScript カスタム アクションと実行されない InstallScript カスタム アクションがある場合、パッケージが昇格された状態で実行された場合、それぞれ異なる一時パスが与えられます。InstallScript カスタム アクションは、異なるユーザーのコンテキストで実行されるため、一時ディレクトリにファイルを保存して、それをあとで取り出す操作は、特定のシナリオで適切に動作しないことがあります。

Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション

InstallScript MSI インストールの遅延/コミット/ロールバックの InstallScript カスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、InstallScript カスタム アクションで他のプロパティ (例、SUPPORTDIR) にアクセスする場合は、それらを CustomActionData に渡す必要があります。これを行うには、すぐにプロパティを設定するタイプのカスタム アクション (例、MsiSetProperty 関数をもつ即時 InstallScript カスタム アクション) をスケジュールして、カスタム アクションの名前に一致するプロパティを設定します。このプロパティの値が、遅延のカスタム アクション内の CustomActionData プロパティから取得できるようになります。

たとえば、SUPPORTDIR のようなプロパティにアクセスする場合、MyCustomActionName という名前で、MyCustomActionName プロパティを [SUPPORTDIR] に設定する即時カスタム アクションを作成して、MsiGetProperty 呼び出しで "SUPPORTDIR" を "CutomActionData" で置き換えることもできます。

MsiGetProperty(hMSI, "CustomActionData", szSupportDir, nLen)

詳細については、次を参照してください。

- 遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う
- 遅延実行カスタム アクションのコンテキスト情報を取得する

提供されていない Windows Installer プロパティを補うことを忘れると、インストール時に予期しない結果が起きることがありますので注意してください。たとえば、遅延アクションでは、MsiGetProperty(hMSI, "INSTALLDIR", szInstallDir, nLen) が szInstallDir を空の文字列に設定するため、szInstallDir ^ szFile は、[INSTALLDIR] のファイルの代わりに、現在のディレクトリにあるファイルを参照する場合があります。遅延カスタム アクションの現在のディレクトリは、通常 [SystemFolder] です。

遅延/コミット/ロールバック InstallScript カスタム アクションは、ProductLanguage プロパティへのアクセスがありません。インストールに複数言語のサポートが含まれていて、かつエンドユーザーがインストールを実行している言語へのアクセスが必要な遅延/コミット/ロールバック InstallScript カスタム アクションもあるとき、インストールは、この言語をインストールのデフォルト言語であると仮定します。これは、エンドユーザーがインストールをデフォルト言語以外の言語で実行するとき、問題となります。ただし、遅延/コミット/ロールバック カスタム アクションは通常、ユーザー インターフェイスを表示しないため、これが問題になることは、ごく稀にしかありません。

DoInstall 関数への変更

DoInstall 関数は今回より、既に実行されているインストール エンジンを使用する代わりに、子インストールのセットアップ実行可能ファイルを起動して、子インストールを実行します。この変更は自動的に行われますので、既存の InstallScript コードへの変更は必要ありません。インストール実行可能ファイルは初期化が必要なため、子インストールの初期化は、以前のバージョンよりも若干時間がかかる場合があります。詳細については、「DoInstall」を参照してください。

今回より、**DoInstall** の呼び出しは、**LaunchAppAndWait** の呼び出しに似ています。インストールが、CD-ROM、フロッピー ディスクなどのリムーバブル メディアから呼び出されたとき、Disk1 の Setup.exe ファイルは、インストール中、常に使用できる状態にあるとはかぎりません。(実行中に、Setup.exe が使用できなくなった場合、オペレーティング システムでプロンプトが表示され、エンドユーザーに正しいディスクを挿入するように求めてくることがあります。これにより、インストールが失敗することがあります。)したがって、この問題を回避するために、Setup.exe ファイルが Temp フォルダーにコピーされ、インストールがそこから再起動されます。元の Setup.exe は、ここで終了します。ただし、この処理が発生すると、**DoInstall** (または、呼び出された場合、**LaunchAppAndWait**) は、インストールが完了したと仮定して動作し、待機しません。

この問題のワークアラウンドは、いくつかあります。その 1 つはまず、子インストールを起動するときに、/Clone_wait パラメーターを使用することです。このワークアラウンドの実行すると、起動されたインストールで、起動された元のプロセスが実行中のまま保たれ、親インストールは待機します。ただし、このワークアラウンドでは、Setup.exe を含む元の CD がインストールの途中で使用できない状態になる場合、問題が発生する場合があります。この例として、インストールの途中で一枚目の CD が使用できなくなる複数 CD のインストールがあげられます。

この問題を回避する唯一の方法は、起動されたプロセスの子プロセスの ID を判別するコードを追加して、子プロセスが完了するのを待機することです。

Disk1 ファイルへの変更

InstallShield は以前、次のファイルをビルドされたインストールの Disk1 イメージに配置していましたが、この処理は必要なくなったため、今後実行されません。

ISScript<バージョン>.msi

InstallShield 12 以降では、次のファイルが、ビルドされたインストールの Disk1 イメージに配置されます。

ISSetup.dll

Disk1 ファイルの変更が原因によるアップグレードの問題は、現在報告されていません。これらの変更は、情報提供のみを目的として報告されています。

InstallScript エンジン ファイルへの変更

InstallShield 12 以降では、InstallScript MSI インストールは、InstallScript エンジン ファイルを次のディレクトリにインストールしません。

<COMMONFILES>¥InstallShield¥Professional¥Runtime¥<MajorVersion>¥<MinorVersion>¥Intel32

InstallShield 11.5 以前の InstallScript MSI プロジェクトでは、いくつかのファイルは Binary テーブルに格納されていました。

- Setup.inx
- Isconfig.ini
- Isrt.dll

- ISScriptBridge.dll
- _isresXXXX.dll (ここで、XXXX は言語です。.dll は、インストールに含まれている各言語に対して 1 つ含まれました。)
- StringxXXXX.txt (ここで、XXXX は言語です。.txt は、インストールの各言語に対して 1 つ含まれました。)

InstallShield 12 以降では、これらのファイル (使われなくなった ISScriptBridge.dll を除く) はすべて、ISSetup.dll ファイル内に格納されます。ISSetup.dll は、Binary テーブルに格納される唯一のファイルです。

InstallScript エンジン ファイルの変更が原因によるアップグレードの問題は、現在、報告されていません。これらの変更は、情報提供のみを目的として報告されています。

InstallScript 関連のカスタム アクション

InstallShield 12 では、いくつかの InstallScript 関連のビルトイン カスタム アクションが、InstallScript MSI プロジェクトで提供されなくなりました。InstallScript MSI プロジェクトを InstallShield 11.5 以前から InstallShield 2025 へアップグレードした場合、InstallShield によって、これらのビルトイン カスタム アクションはプロジェクトから削除されます。この削除のため、以前これらのカスタム アクションによって呼び出されていた既存の InstallScript イベント ハンドラー関数は、今回より、InstallScript から直接呼び出されます。詳細については、「InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」を参照してください。

Setup.exe ファイルのない InstallScript MSI インストールを配布するための変更

パッケージを Active Directory などのテクノロジを通して配布できるようにするために、InstallScript MSI インストールが Setup.exe ファイルなしに実行されるのを許可するには、「ナレッジベース記事 Q108166 – HOWTO: Deploying an MSI Wrapped with an InstallShield Script-Based Setup.exe (MSI を InstallShield スクリプト ベースの Setup.exe でラップして配布する)」で説明されている手順を使用できる場合があります。ただし、この記事で説明されている手順を適用して、ユーザーがインストール時に .msi ファイルを直接起動すると、一部の InstallScript イベント (OnMoved など) が起動されなくなります。これらのイベントが呼び出されない理由は、InstallScript 関連のカスタム アクションが削除されたため、以前 InstallScript 関連のビルトイン カスタム アクションのよって呼び出されていた一部の InstallScript イベント ハンドラー関数が呼び出されないためです。

詳細については、「InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」を参照してください。

InstallScript MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール

InstallShield 12 では、いくつかの InstallScript 関連のビルトイン カスタム アクションが、InstallScript MSI プロジェクトで提供されなくなりました。InstallScript MSI プロジェクトを InstallShield 11.5 以前から InstallShield 2025 へアップグレードした場合、 によって、これらのビルトイン カスタム アクションはプロジェクトから削除されます。この削除のため、以前これらのカスタム アクションによって呼び出されていた既存の InstallScript イベントは、今回より、InstallScript から直接呼び出されます。この記事では、削除されたカスタム アクションが確認でき、スケジュールが InstallShield 11.5 以前で使用されたスケジュールとほぼ同一になるように、これらの InstallScript イベントを呼び出すカスタム アクションをどのように追加するかが説明されています。

パッケージを Active Directory などのテクノロジを通して配布できるようにするために、InstallScript MSI インストールがSetup.exe ファイルなしに実行できるようにした場合、削除されたカスタム アクションが原因で InstallScript MSI インストールでも影響が出ることがあります。これを実装するための手順は、ナレッジベース記事「Q108166 – HOWTO: Deploying an MSI Wrapped with an InstallShield Script-Based Setup.exe (MSI を InstallShield ス

クリプト ベースの Setup.exe でラップして配布する)」を参照してください。この記事で説明されている手順を適用して、ユーザーがインストール時に .msi ファイルを直接起動すると、一部の InstallScript イベント (OnMoved など) が起動されなくなりますますので注意してください。これらのイベントが呼び出されない理由は、InstallScript 関連のカスタム アクションが削除されたため、以前 InstallScript 関連のビルトイン カスタム アクションのよって呼び出されていた一部の InstallScript イベント ハンドラー関数が呼び出されないためです。また、インストールを Q108166 記事にしたがって構成した場合、以下の手順にしたがって、ビルトイン InstallScript イベントを呼び出すカスタム アクションを追加する必要があります。

次の InstallScript 関連のビルトイン カスタム アクションが、InstallShield 12 で提供されなくなりました。これらのカスタム アクションは、InstallShield 11.5 以前から InstallShield 2025 にアップグレードされたとき、InstallScript MSI プロジェクトから削除されます:

- ISCleanupSuccess
- ISCleanUpSuspend
- ISCleanUpUserTerminate
- ISCleanupFatalExit
- ISMsiServerStartup
- ISRebootPatchHandler
- ISRollbackCleanup
- ISStartup
- OnCheckSilentInstall (このカスタム アクションの変更に関する詳細は、「OnCheckSilentInstall」を参照してください。)
- OnFeaturesInstalled
- OnFeaturesInstalling
- OnInstallFilesActionAfter
- OnInstallFilesActionBefore
- OnMoved
- OnMoving

この削除のため、以前これらのカスタム アクションによって呼び出されていた既存の InstallScript イベント ハンドラーは、今回より、InstallScript から直接呼び出されます。これには、ファイル転送の直前で呼び出される InstallScript イベント ハンドラーが含まれます。

- Installing および Uninstalling の機能関数
- OnGeneratedMSIScript
- OnGeneratingMSIScript
- OnInstallFilesActionBefore
- OnMoving

これにはまた、ファイル転送の直後に呼び出される InstallScript イベント ハンドラーも含まれます。

- Installed および Uninstalled の機能関数

- OnMoved
- OnInstallFilesActionAfter

これらの変更は、グローバル変数およびグローバル ポインターが、これらの InstallScript イベント ハンドラー関数で使用できるように加えられました。ただし、イベントは以前、カスタム アクションから呼び出されていたため、これらのイベントのシーケンスは、Windows Installer によって直接呼び出される他のカスタム アクションに関連して変更されています。したがって、場合によって、これらの変更が適切に反映されるように、InstallScript コードを変更する必要があります。

ただし、機能のイベント ハンドラーの場合のみ、スケジュールが InstallShield 11.5 以前で使用されたスケジュールとほぼ同一になるように、これらの InstallScript イベントを呼び出すカスタム アクションを追加することも可能です。ただし、グローバル変数とポインターは、「グローバル変数、グローバル ポインター、および SUPPORTDIR」で議論されているように、個々の InstallScript カスタム アクションの呼び出し間で保持されません。

ファイル転送中にカスタム アクションから OnFeatureInstalling、OnFeatureUninstalling、OnFeatureInstalled、または OnFeatureUninstalled を呼び出しても、インストールへの影響はありません。これは、これらの関数がファイル転送の直前に InstallScript から呼び出されるためで、これらが複数回呼び出されても、影響はありません。

タスク ***定義済み InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを手動でスケジュールするには、以下の手順に従います:***

1. InstallShield 2025 で、アップグレード済みのプロジェクトを開きます。
2. InstallScript ファイルへ適切な変更を行います。
   a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
   b. スクリプトでイベント ハンドラー関数を検索します。ビューの中央にあるペインで関数名をクリックすると、簡単に関数を検索することができます。
   c. 関数の名前を代替名に変更して、ifx.h に自動的に含まれる既存の関数プロトタイプと競合するのを避けるようにします。例:
      - MyOnGeneratingMSIScript
      - MyOnMoving
      - MyOnMoved
   d. 既存の関数を、単一 HWND パラメーターを取得するようにアップグレードします。例:

      ```
      function MyOnGeneratingMSIScript(hMSI) begin end;
      ```

   e. この新しい関数の適切なプロトタイプを追加します。

      ```
      export prototype MyOnGeneratingMSIScript(HWND);
      ```

3. 名前が変更された InstallScript イベント ハンドラー関数を呼び出す InstallScript カスタム アクションを追加します:
   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
   b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい InstallScript]** をクリックします。InstallScript カスタム アクションが追加されます。
   c. カスタム アクションの名前を入力します。この名前には、InstallScript 関数の名前を変更したときに使用した名前を使用します。例:

- MyOnGeneratingMSIScript
- MyOnMoving
- MyOnMoved

d. 作成した新しい InstallScript カスタム アクションを選択します。

e. **"関数名"** 設定を、InstallScript 関数の名前に設定します。

f. **"スクリプト内実行"** 設定で、下のテーブルで示されている値を指定します。

4. InstallScript カスタム アクションをインストールの適切な箇所にスケジュールします。

a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

b. InstallScript カスタム アクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。**[アクションの挿入] ダイアログ ボックス**が開き、プロジェクトに現在関連付けられているアクションおよびダイアログの一覧が表示されます。

c. 作成した InstallScript カスタム アクションを選択します。必要に応じて、InstallScript イベントの起動についての条件を **[条件]** ボックスに入力します。

d. **[OK]** をクリックします。

以前の機能にできるだけ近づけるため、ステップ 3f のスクリプト内実行の設定と、ステップ 4b の InstallScript カスタム アクションのスケジュールを次のように行います。

テーブル 11 ・ カスタム アクションのスケジュール

カスタム アクション	スクリプト内実行	シーケンス中の場所
OnGeneratingMSIScript	即時 (デフォルト)	[インストール] の [実行] シーケンス、RemoveExistingProducts アクションと InstallInitialize アクションの間
OnMoving	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallInitialize アクションと AllocateRegistrySpace アクションの間
OnFeaturesInstalling。定義されたすべての機能の Installing イベントと Uninstalling イベントを呼び出します。メモ: 上記で説明されているように、これによる影響はありません。	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallInitialize アクションと AllocateRegistrySpace アクションの間 (OnMoving の後)
OnInstallFilesActionBefore	システム コンテキストの遅延	[インストール] の [実行] シーケンス、MoveFiles アクションと InstallFiles アクションの間

テーブル 11 • カスタム アクションのスケジュール (続き.)

カスタム アクション	スクリプト内実行	シーケンス中の場所
OnFeaturesInstalled。定義されたすべての機能の Installed イベントと Uninstalled イベントを呼び出します。メモ: 上記で説明されているように、これによる影響はありません。	システム コンテキストの遅延	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間 (OnMoved の前)
OnMoved	システム コンテキストの遅延	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間 (OnGeneratedMSIScript の前)
OnInstallFilesActionAfter	システム コンテキストの遅延	[インストール] の [実行] シーケンス、InstallFiles アクションと PatchFiles アクションの間
OnGeneratedMSIScript	即時 (デフォルト)	[インストール] の [実行] シーケンス、ScheduleReboot アクションと InstallFinalize アクションの間 (OnMoved の後)

次の事項に注意してください。

- グローバル変数とポインターは、個々の InstallScript カスタム アクションの呼び出し間で保持されなくなりました。また、個々の InstallScript カスタム アクションでは、それぞれ固有の SUPPORTDIR が初期化され、使用されます。したがって、SUPPORTDIR のファイルでは、個々の呼び出し間で情報を共有することはできません。詳細については、「グローバル変数、グローバル ポインター、および SUPPORTDIR」を参照してください。
- ほとんどの Windows Installer プロパティは、遅延/コミット/ロールバック InstallScript カスタム アクションで使用することはできません。詳細については、「Windows Installer プロパティと遅延/コミット/ロールバック InstallScript カスタム アクション」を参照してください。

OnCheckSilentInstall

InstallShield 11.5 以前では、InstallScript MSI インストールが Setup.exe を使わずにサイレントでインストールされているとき、OnCheckSilentInstall カスタム アクションが自動的に OnMsiSilentInstall InstallScript イベントを呼び出していました (たとえば、次もコマンドラインが使用された場合:Msi.exe /i<Package> /qn)。InstallShield 12 以降では、

このイベントは InstallScript MSI インストールで呼び出されません。ただし、OnMsiSilentInstall イベント ハンドラー関数を呼び出すカスタム アクションを追加することができます。これを行うには、次のステップ 3f と 4b で説明されているスケジュール時における要件に留意して、上述の手順を実行します。

テーブル 12 - OnCheckSilentInstall カスタム アクションのスケジュール

カスタム アクション	スクリプト内実行	シーケンス中の場所
OnCheckSilentInstall	即時 (デフォルト)	[インストール] の [実行] シーケンスの最初のアクションとして

また、InstallScript イベントが予期されているシナリオでのみ実行されるようにするには、次のコードをイベントに追加します。

```
cchValueBuf = MAX_PATH;
// Setup.exe が使用されているか確認し、使用されている場合、戻ります。
MsiGetProperty(nHandle, "ISSETUPDRIVEN", szValueBuf, cchValueBuf);
if(StrLengthChars(szValueBuf)) then
   return ISERR_SUCCESS;
endif;
```

InstallShield 11.5 以前の InstallScript プロジェクトをアップグレードする

InstallShield 11.5 以前で作成された InstallScript プロジェクトを InstallShield 2025 にアップグレードするときの詳しい説明については、以下のセクションを参照してください。

InstallScript オブジェクトを InstallScript プロジェクトに含める

InstallShield 2025 で作成された InstallScript インストールは、以前のバージョンの InstallShield 11.5 で作成されたオブジェクトを使用できません。

InstallScript プロジェクトを InstallShield 2025 へアップグレードすると、すべての InstallShield オブジェクトへの参照は、InstallShield 2025 バージョンのオブジェクトをポイントするように更新されます。新しいバージョンのオブジェクトがシステムに存在しない場合、「オブジェクトが見つかりませんでした」というエラー メッセージがビルド時に表示されます。

InstallShield 2025 オブジェクトを取得する方法については、「InstallShield のアップデートを取得する」を参照してください。

DoInstall 関数への変更

DoInstall 関数は今回より、既に実行されているインストール エンジンを使用する代わりに、子インストールのセットアップ実行可能ファイルを起動して、子インストールを実行します。この変更は自動的に行われますので、既存の InstallScript コードへの変更は必要ありません。インストール実行可能ファイルは初期化が必要なため、子インストールの初期化は、以前のバージョンよりも若干時間がかかる場合があります。詳細については、「DoInstall」を参照してください。

今回より、**DoInstall** の呼び出しは、**LaunchAppAndWait** の呼び出しに似ています。インストールが、CD-ROM または DVD などのリムーバブル メディアから実行されたとき、Disk1 の Setup.exe ファイルは、インストール中、常に使用できる状態にあるとはかぎりません。(実行中に、Setup.exe が使用できなくなった場合、オペレーティング システムでプロンプトが表示され、エンドユーザーに正しいディスクを挿入するように求めてくることがあります。

これにより、インストールが失敗することがあります。)したがって、この問題を回避するために、Setup.exe ファイルが Temp フォルダーにコピーされ、インストールがそこから再起動されます。元の Setup.exe は、ここで終了します。ただし、この処理が発生すると、**DoInstall** (または、呼び出された場合、**LaunchAppAndWait**) は、インストールが完了したと仮定して動作し、待機しません。

この問題のワークアラウンドは、いくつかあります。その 1 つはまず、子インストールを起動するときに、/Clone_wait パラメーターを使用することです。このワークアラウンドの実行すると、起動されたインストールで、起動された元のプロセスが実行中のまま保たれ、親インストールは待機します。ただし、このワークアラウンドでは、Setup.exe を含む元の CD がインストールの途中で使用できない状態になる場合、問題が発生する場合があります。この例として、インストールの途中で一枚目の CD が使用できなくなる複数 CD のインストールがあげられます。

この問題を回避する唯一の方法は、起動されたプロセスの子プロセスの ID を判別するコードを追加して、子プロセスが完了するのを待機することです。

Setup.exe の変更

Setup.exe の /deleter コマンドライン パラメーターが今回より、必要なくなりました。このパラメーターを指定すると、インストールは実行されません。InstallScript インストールは、インストールの起動後すぐにインストールのクローンを作成しなくなったため、/deleter が必要なくなりました (ただし、/runfromtemp パラメーターが指定されたために、インストールが一時フォルダーから実行されている場合などは除きます)。

Setup.exe の /clone_nowait コマンドライン パラメーターが今回より、必要なくなりました。このパラメーターを指定すると、Setup.exe はそれを無視します。InstallScript インストールは今回より、デフォルトでインストールを完了するためにクローンされたインストールを待機しません (/Clone_wait パラメーターが指定された場合を除きます)。/Clone_wait についての詳しい情報は、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

InstallScript MSI と基本の MSI インストール同様、InstallScript の Setup.exe は、有効なリターン コードを返します。したがって、Setup.exe の戻り値を確認するとき、InstallShield 11.5 ではいつも 0 でしたが、InstallShield 12 以降では、Setup.exe が適切な戻り値を返します。返される戻り値についての詳細については、「Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト)」を参照してください。

Disk1 ファイルへの変更

InstallShield は以前、次のファイルをビルドされたインストールの Disk1 イメージに配置していましたが、この処理は必要なくなったため、今後実行されません。

- Engine32.cab
- Setup.ibt

InstallShield 12 以降で、次のファイルが、ビルドされたインストールの Disk1 イメージに配置されます。

- ISSetup.dll
- _Setup.dll

Disk1 ファイルの変更が原因によるアップグレードの問題は、現在報告されていません。これらの変更は、情報提供のみを目的として報告されています。

InstallScript エンジン ファイルへの変更

InstallShield 12 以降では、InstallScript インストールは、InstallScript エンジン ファイルを次のディレクトリにインストールしません。

<COMMONFILES>¥InstallShield¥Professional¥Runtime¥<MajorVersion>¥<MinorVersion>¥Intel32

この変更が原因によるアップグレードの問題は、現在報告されていません。この変更は、情報提供を目的として報告されています。

InstallScript カスタム アクションを含む InstallShield 11.5 または以前の QuickPatch プロジェクトをアップグレードする

InstallShield 12 以降で InstallScript MSI プロジェクトに作成された QuickPatch プロジェクトは、InstallShield 12 以降で作成されたパッチ メディアのみパッチすることができます。InstallShield 2025 を使って 以前の InstallScript MSI インストールが InstallShield 11.5 以前で作成されたアプリケーションのアップデートを作成する場合、QuickPatch の代わりにマイナーアップグレード パッケージを作成して、[パッチのデザイン] ビューを使って、標準パッチを作成します。このバージョン以後のパッチは、QuickPatch プロジェクトとして作成できます。

(InstallScript カスタム アクションがあるないにかかわらず) 基本の MSI プロジェクトに作成された QuickPatch に関する 問題は、現在報告されていません。

InstallShield 11.5 以前の InstallScript MSI プロジェクトに標準のパッチを作成する

InstallShield 12 以降は、最新および以前のセットアップが InstallShield 11.5 以前で作成された InstallScript MSI パッチの作成 ([パッチのデザイン] ビューを使用して) をサポートしていません。最新のセットアップが InstallShield 12 以降で作成されたパッチ、または以前のセットアップが InstallShield 11.5 以前で作成されたパッチの作成に関する問題は、現在報告されていません。

InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトまたはこの種類のオブジェクトを含むプロジェクトをアップグレードする

InstallShield 11.5 以前には、プロジェクトの InstallScript MSI オブジェクト タイプが含まれました。このプロジェクトの種類は InstallShield では利用できません。

InstallScript MSI オブジェクト プロジェクトをアップグレードする

InstallScript MSI オブジェクト プロジェクトを InstallShield 2025 で開くと、InstallShield はこれをマージ モジュール プロジェクトに変換します。

定義済み InstallScript イベント ハンドラーは現在、InstallScript カスタム アクションを含むマージ モジュール プロジェクトで使用できません。したがって、InstallScript イベント ハンドラー関数を使用する InstallShield 11.5 以前の InstallScript MSI オブジェクト プロジェクトがある場合、InstallShield 2025 でプロジェクトがマージ モジュールの変換されるときに、これらの関数を呼び出すカスタム アクションを手動でスケジュールする必要があります。詳しくは、「基本の MSI プロジェクトで InstallScript イベント ハンドラーを呼び出す InstallScript カスタム アクションの作成とスケジュール」をご覧ください。

InstallShield 2025 は、InstallShield の以前および最新のバージョン同様、InstallScript カスタム アクションがある InstallShield 2025 マージ モジュールを使用することができます。

InstallScript MSI オブジェクトを含むプロジェクトのアップグレード

ビルド済みの InstallScript MSI オブジェクト (.imm フィル) を含むアップグレード プロジェクトは、ビルドをしても失敗します。これは、これらのオブジェクトが InstallShield 2025 では使用できないためです。これを試みると、次のようなビルド エラーが発生します。

ISDEV : エラー -4075: ファイルが見つかりません。機能 'NewFeature1' にモジュール 'InstallShieldMSIObjectName.4F635B62_07BF_4779_B74E_D80C29D508E3:0' をマージ中に、エラーが発生しました。

"InstallShieldMSIObjectName.4F635B62_07BF_4779_B74E_D80C29D508E3:0" は、見つからない InstallScript MSI オブジェクトに固有の情報です。このビルド エラーを解決するには、この InstallScript MSI オブジェクトをプロジェクトから削除します。マージ モジュールは、[再配布可能ファイル] ビューにあるチェック ボックスをクリアすると削除されます。

InstallShield Express で作成したプロジェクトをアップグレードする

InstallShield Express のバージョン 2.1 から 5.x で作成した InstallShield プロジェクトがある場合、そのプロジェクトは InstallShield 2025 で開いたとき自動的に基本の MSI プロジェクトにアップグレードされます。InstallShield 2025 は InstallShield Express プロジェクトを、それと互換性のあるプロジェクトの種類である基本の MSI プロジェクトへ変換します。

アップグレードしたプロジェクトを InstallScript MSI プロジェクトへ変換する方法については、基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する を参照してください。

タスク ***Express プロジェクトをアップグレードするには、以下の手順の従います。***

InstallShield でプロジェクトを開きます。プロジェクトをアップグレードするかどうかたずねるダイアログ ボックスが開きます。

- **[はい]** をクリックしてプロジェクトをアップグレードします。プロジェクトが InstallShield で開きます。
- プロジェクトを開かないでおく場合は **[いいえ]** をクリックします。

元のプロジェクトのバックアップコピーは、このダイアログ ボックスで指定した場所に作成および保管されます。

InstallShield 2025 は、InstallShield 2025 で Express 2.x プロジェクト (.iwz) を基本の MSI プロジェクトに移行したときに、[セットアップのタイプ] ビューのロジックとダイアログを保存します。Express の以前のバージョンは、セットアップのタイプはコンポーネントに基づいていました。Express 2.x プロジェクト (.iwz) では、コンポーネントは機能に置換され、ファイルグループはコンポーネントではなくインストール先フォルダーにマップされています。

InstallShield 2025 で、Express 2.x プロジェクト (.iwz) を 基本の MSI プロジェクトにアップグレードすると、機能の "インストールレベル" プロパティとダイアログ コントロール イベントを変更することによって、セットアップの種類を編集できます。

メモ: *Express プロジェクトの [ビルボード] プロパティは、基本の MSI プロジェクトでサポートされていないため、移行されません。ビルボード プロパティが以前の Express プロジェクトで指定されている場合、警告 701 が出されます。*

現在のバージョンの InstallShield にインストールされていない言語を持つ Express プロジェクトを移行すると、その言語のサポートが失われます。言語サポートは、InstallShield の Premier Edition でのみ使用することができます。

InstallShield Professional で作成したプロジェクトをアップグレードする

InstallShield Professional を使って作成されたプロジェクトをアップグレードすることができます。InstallShield 2025 を使用して InstallShield Professional で作成されたインストール プロジェクトを開くと、プロジェクトは自動的にアップグレードされます。これにより、InstallShield 2025 ですぐ作業を開始することができます。

InstallShield Professional を使用して作成されたプロジェクトは、.ipr ファイルにより参照される、いくつかの .ini ファイルから構成されていました。InstallShield 2025 では、インストール プロジェクトは .ism ファイル 1 つだけです。

最初のステップ: インストール プロジェクトを開く

タスク ***InstallShield Professional で作成されたインストール プロジェクトを開くには、以下の手順を実行します。***

1. InstallShieldを開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. 開きたい InstallShield Professional プロジェクト ファイル(.ipr) を選択します。
4. **[開く]** をクリックします。

プロジェクトが InstallShield で開き、拡張子が .ipr から .ism に変わります。元の .ipr プロジェクトのバックアップは、同じフォルダーに *ProjectName*.ipr.bak として保存されます。

次のステップ: インストール プロジェクトの変更

インストール プロジェクトをアップグレードした後、InstallShield ユーザー インターフェイス のビュー、メニュー、ウィザードその他を使用して、プロジェクト設定を変更できます。

アップグレードしたプロジェクトに変更を加えなくてならない場合もあります。詳しくは、「InstallShield Professional 6.x からの移行」および「InstallShield Professional 5.x からの移行」を参照してください。

メモ: *InstallShield 2025 では、サポート ファイルがプロジェクト フォルダーのサブフォルダーに リンクされています。InstallShield Proefessional の場合、サポート ファイルはプロジェクト フォルダーのサブフォルダーにコピーされました。*

InstallShield Professional 6.x からの移行

InstallShield Professional 6.x インストールを InstallShield 2025 にアップグレードする際は、以下の事項に注意してください:

- program…endprogram ブロックを使用するスクリプトを、ユーザー インターフェイスおよびファイル転送関数を除くすべての適切なイベント ハンドラー関数を呼び出すイベントベースのスクリプトに簡単に変換することができます。詳細については、「OnShowUI」を参照してください。

- インターネット インストール用の 新しい簡易化モデル は、数多くの Ether オブジェクトのメソッドとそのプロパティとサブオブジェクトをすべて省略します（この時同時に、インターネット固有の必要条件に対応できるように InstallScript の強化点も提供します）。インターネット インストールにこれらのプロパティやメソッドを使用している場合、「古いプロパティとメソッドの置換」をご覧ください。

- version 6.x で作成されたインストールは自動作成された文字エントリを数多く含み、これらの文字列エントリを情報取得のために使用していました。これらの文字列エントリは、同じ情報が [プロジェクトの設定] プロパティ シートにも存在していたため、不要でした。InstallShield 2025 は、[プロジェクト設定] で指定された情報を、実行時にも使用できるようにして、作業を単純化しました。

  この情報を使用することは必須ではありません。文字列エントリは、プロジェクト内の既存のものをそのままに残すことで、これまで同様に使い続けることができます。

  移行したインストールがプロジェクトの [プロジェクトの設定] プロパティシートから情報を自動的に使用できるようにしたい場合、以下の文字列エントリをプロジェクトから削除します。

  COMPANY_NAME

  PRODUCT_NAME

  PRODUCT_KEY

  PRODUCT_VERSION

  TITLE_CAPTIONBAR

  FOLDER_NAME

  TITLE_MAIN

  また、次の表で示されたような、これらの文字列エントリのひとつを使って、新規の対応するシステム変数を代わりに使用している可能性があるスクリプト コードを更新します。

テーブル 13・文字列エントリと対応する新しいシステム変数

文字列のエントリ	システム変数
COMPANY_NAME	IFX_COMPANY _NAME
PRODUCT_NAME	IFX_PRODUCT_NAME
PRODUCT_KEY	IFX_PRODUCT_KEY
PRODUCT_VERSION	IFX_PRODUCT_VERSION
TITLE_CAPTIONBAR	IFX_SETUP_TITLE
FOLDER_NAME	IFX_PRODUCT_NAME
TITLE_MAIN	IFX_SETUP_TITLE

ある値を文字列テーブルから読み込み、別の値をプロジェクトの設定から読み込む場合は、これらの文字列値のサブセットのみを削除してください。保持する文字列値がある場合、未使用のプロジェクトの設定を変更せずに、文字列テーブルの値を必要に応じて修正するようにしてください。特に、アップデート インストールを作成する場合は、PRODUCT_VERSION 文字列エントリの値が存在する場合、それを必ず更新してください。

- インストールでイベントベースのスクリプトを使用して、デフォルト OnSetUpdateMode イベント ハンドラー コードがオーバーライドされている場合、実行時に「インストールされているアプリケーションのバージョンを判別できません。セットアップを終了します。」といったメッセージが表示される場合があります。このエラーが発生する場合、OnSetUpdateMode イベント中に手動で IFX_INSTALLED_VERSION を設定するコードを追加します。サンプル コードは、新しい InstallShield 2025 InstallScript プロジェクトの OnSetUpdateMode イベントに含まれているデフォルト コードを参照してください。

- インストールがイベントベースのスクリプトを使用し、デフォルトの OnFirstUIBefore イベント ハンドラー コードがオーバーライドされている場合、ファイルを適切なデフォルトの保存場所にインストールするために、(OnFirstUIBefore の 6.x バージョンからの) 次のコードを置換する必要があります。

  TARGETDIR = PROGRAMFILES ^ @COMPANY_NAME ^ @PRODUCT_NAME;

  (新規のデフォルトの OnFirstUIBefore コードからの)次のコードを TARGETDIR に設定します。

  ```
  /* 管理者権限があるエンドユーザー、またはそのような権限を持たないパワーユーザーおよび
  エンドユーザーにも対処します。*/
  if ( ALLUSERS ) then
     TARGETDIR = PROGRAMFILES ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
  else
     TARGETDIR = FOLDER_APPDATA ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
  endif;

  /* 標準と 複数インスタンス インストールの両方を対処します*/
  if( MAINT_OPTION = MAINT_OPTION_MULTI_INSTANCE && MULTI_INSTANCE_COUNT > 0) then
     nLoop = 1;
     svDir = TARGETDIR;
     while(ExistsDir(TARGETDIR) = EXISTS)
        NumToStr(szTargetDirAppendix,nLoop);
        TARGETDIR = svDir + szTargetDirAppendix;
        nLoop = nLoop + 1;
     endwhile;
  endif;
  ```

  手続き型のスクリプト (programÉendprogram ブロックがあるスクリプト) を使用している場合、管理者権限がないユーザー用に、デフォルトのターゲット ディレクトリを適切に設定する TARGETDIR を設定するコードを更新しなければなりません。

- インストールがイベントベースのスクリプトを使用し、デフォルトの OnMaintUIBefore イベント ハンドラー コードがオーバーライドされていて、アンインストール実行時に、メディアにはリストされていなくログ ファイル(FeatureRemoveAllInMediaAndLog を参照) にのみリストされているものをはじめとするすべての機能を削除する場合、(OnMaintUIBefore の 6.x バージョンからの) 次にコードを置換する必要があります。

  case REMOVEALL: ComponentRemoveAll();

  (新規のデフォルトの OnMaintUIBefore コードからの)次のコードを使用します。

  MediaGetData(MEDIA, MEDIA_FIELD_MEDIA_FLAGS, nMediaFlags, szIgnore);

  ```
  case REMOVEALL:
     /* プロパティは 更新を行います。*/
  ```

```
if( nMediaFlags & MEDIA_FLAG_UPDATEMODE_SUPPORTED ) then
    FeatureRemoveAllInMediaAndLog();
else
    FeatureRemoveAllInMedia();
endif;
```

スクリプトベースのアンインストールをサポートする手続き型のスクリプトを使用している場合、ComponentRemoveAll の呼び出しを適切に更新する必要があります。

- スクリプトが RegDBSetItem を呼んで MaintenanceStart (または、DeinstallStart) によって作成されたレジストリ エントリを変更し、RegDBSetItem への呼び出しを OnMoved ハンドラー (例, OnMoving または <機能名>_OnInstalled) の前で呼び出されたイベント ハンドラーで行う場合、RegDBSetItem への呼び出しを移動する必要があります。MaintenanceStart は現在 OnMoveData イベント ハンドラーのデフォルトのコードで呼び出されています。MaintenanceStart は、以前、メンテナンス/アンインストール機能 がインストールされたすぐ後、他の機能がすべてインストールされる前に自動的に呼び出されていました。

- InstallShield Professional 6.x のデフォルトのイベント ハンドラーのコードは、以下のラインを OnFirstUIBefore と OnMaintUIBefore の両方に含んでいました。

  ```
  SetStatusWindow (0, "");
  Enable(STATUSEX);
  StatusUpdate (ON, 100);
  ```

  InstallShield 2025 では、このコードはデフォルトの OnMoveData イベント ハンドラーコードに移動したので、以下のオプションができました。

  - このコードをカスタマイズしていない場合、余分な呼び出しは問題の原因なるので、特に何もする必要はありません。コードの重複を避けたい場合、このコードを OnFirstUIBefore と OnMaintUIBefore から支障なく削除することができます。
  - このコードが既にカスタマイズされていて、このカスタマイズを、呼び出す UI (ユーザー インターフェイス) イベント関わらず適用したい場合、OnFirstUIBefore と OnMaintUIBefore からコードを削除して、OnMoveData イベントをオーバーライドし、デフォルトのコードの位置にカスタマイズしたコードを起きます。
  - コードが既にカスタマイズされていて、呼び出される UI イベントによって特定のコードを指定したい場合、OnMoveData をオーバーライドして、デフォルトのコードをコメントとして除き、既存のコードを継続して使います。このとき、インストールがアップデートをサポートしていない場合、OnUpdateUIBefore のカスタマイズが必要になる場合もあります。

- バージョン 6.x のデフォルトのイベント ハンドラーコードは、OnFirstUIBefore および OnMaintUIBefore イベントに以下のコメントアウトされたコードを含んでいました。

  ```
  //処理内容: 背景、ウィンドウタイトル、およびキャプションバーのタイトルを有効にする場合
  // SetTitle( @TITLE_MAIN, 24, WHITE );
  // SetTitle( @TITLE_CAPTIONBAR, 0, BACKGROUNDCAPTION );
  // Enable( FULLWINDOWMODE );
  // Enable( BACKGROUND );
  // SetColor( BACKGROUND, RGB( 0, 128, 128 ) );
  ```

  このコードをアクティブにして、呼び出された UI イベントに関わらず安定した UI エクスペリエンスを提供したい場合は、OnFirstUIBefore と OnMaintUIBefore からこのコードを削除し、OnShowUI をオーバーライドしてコードを適切にカスタマイズします。

- InstallShield 2025 では、多くの Windows API 関数はインクルードされたヘッダー ファイル (.h ファイル) 内で宣言されます。これらの関数が明示的にスクリプト コードで宣言されている場合、以下を実行します。

- 関数宣言を削除し、InstallShield Professional が提供する宣言を使用します。
- InstallShield 2025 と以前のバージョン両方でコンパイル可能なスクリプトを作成するとき、宣言を以下のようなコードで囲みます。

```
#if _ISCRIPT_VER < 0x700
prototype ...
#endif
```

InstallShield Professional が定義した指定が異なる時、Windows API 関数を呼び出すコードを更新する必要がある場合があります。

- エンドユーザーのインターフェイスがスムーズに行われるようにするため、インストール初期化ダイアログ ボックスは、スクリプトによってダイアログ ボックスが表示されるまでデフォルトで表示されたままになります。インストール初期化ダイアログ ボックスを閉じるには、Disable(DIALOGCACHE) またはダイアログ関数を呼び出します。

InstallShield Professional 5.x からの移行

プロジェクト: *この情報は、InstallScript Object プロジェクトに適用します。*

スクリプトの変更

InstallShield Professional 5.x スクリプトの基本構造は、InstallShield 2025 の InstallScript プロジェクト タイプでサポートされています。InstallShield Professional 5.x インストールを InstallShield でコンパイルする場合、まず次のような変更を行う必要があります。

プリプロセッサ命令

- スクリプトの始めに、次のステートメントを含みます。

```
#include "ifx.h"
```

- SD_SINGLE_DIALOGS の #define ステートメント、およびそれに関連した SD_<ダイアログ名> 定数を、すべて削除します。
- Windows の定数を定義する #define ステートメントは、コンパイラのエラーの原因になっている場合、削除が必要です。多くの Windows 定数は Ifx.h (*InstallShield Program Files フォルダー*¥Script¥Isrt¥Include サブフォルダー内)、またはそこに含まれるファイルで定義されています。

ビルトイン関数

- 次の関数を削除します。これらの関数は、Professional 5.x ではサポートされていましたが、InstallShield 2025 ではサポートされていません。これらの関数のうちいくつかは、コンパイル可能ですが、期待される結果は得られません。可能な場合、サポートされている関数を代わりに使用します。

テーブル 14 · 現在不使用の Professional 5.x 関数

Professional 5.x 関数	代替となる関数
AppCommand	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)

テーブル 14 • 現在不使用の Professional 5.x 関数 (続き.)

Professional 5.x 関数	代替となる関数
AddProgItemEx	AddFolderIcon
CmdGetMsg	Windows API 関数の GetMessage (CmdGetHwndDlg を呼び出してダイアログ ハンドルを呼び出す)
CmdGetParam1	Windows API 関数の GetMessage (CmdGetHwndDlg を呼び出してダイアログ ハンドルを呼び出す)
CmdGetParam2	Windows API 関数の GetMessage (CmdGetHwndDlg を呼び出してダイアログ ハンドルを呼び出す)
CommitSharedFiles	この関数を使用して Professional 5.x で実行したアクションは、InstallShield 2025 では自動的に実行されます。
CreateProgGroupEx	AddFolderIcon
DeleteGroup	DeleteProgramFolder
DeleteProgItem	DeleteFolderIcon
ExitProgMan	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
GetGroupNameList	GetFolderNameList
GetItemNameList	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
QueryProgGroup	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
RegDBCreateKey	RegDBCreateKeyEx
RegDBCreateKeyValue	RegDBCreateKeyValueEx
RegDBGetKeyValue	RegDBGetKeyValueEx
RegDBSetKeyValue	RegDBSetKeyValueEx
ReloadProgGroup	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
ReplaceProgItem	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)
ShowGroup	なし。(現在はサポートされていないプログラム マネージャー シェル使用のプラットフォームでのみ適用。)

- アンインストールのためにログされてほしくないターゲット システムを変更するコードの前に Disable(LOGGING) を呼び出し、そのコードの後で Enable(LOGGING) を呼び出します。これは、スクリプト コードの実行前にアンインストールの変更のログが自動的に有効になるので、必要です。

  これは、DeInstallStart が呼び出されるまでログが開始しなかった、Professional 5.x のインストールとは異なります。DeInstallStart の前に起こったために以前はログされなかったインストールアクションがある場合、これらのインストールアクションの前にさらに Disable(LOGGING) 呼び出しを追加して、ログを手動で無効にする必要があります。

  インストールで背景を表示する場合はEnable(BACKGROUND)を呼び出します。InstallShield 2025 では、背景はデフォルトで無効になっていますが、Professional 5.x ではデフォルトで有効になっています。(イベント ベースのスクリプトでは、背景のカスタマイズおよび表示用コードは、OnFirstUIBefore に対するデフォルト コードの始めのコメント行部分にあります。このコードをアクティブにするには、希望の行の始めのスラッシュ文字を削除します。)

  PlaceBitmap などの一部のユーザー インターフェイス関数は、インストール背景が有効になっていないと正しく動作しません (また失敗を示す負の値を返すこともありません)。

- アンインストールキーの [DisplayName] の値を設定するために、引数である REGDB_UNINSTALL_NAME を使用して RegDBSetItem を呼び出す必要はありません。このバージョンでは、MaintenanceStart または DeinstallStart を呼び出した際に行います。

- 引数の STATUSBAR を使用して SetColor を呼び出しても、InstallShield 2025 では何も起こりません。ステータス バーは、システム カラーを使用して作成されます。

- StrGetTokens を呼び出す場合、szString の最初 (または最後) の文字が szDelimiterSet の文字と一致すると、ヌル文字列 ("") が最初 (または最後) の要素としてリストに挿入されません。代わりに、最初と 2 番目(または最後と最後から2番目) の区切り文字の間の文字が最初 (または最後) の要素としてリストに挿入されます。

- StrGetTokens を呼び出すと、szSrcFile で指定されたファイル名にワイルドカード文字が含まれる場合、szTargetFile のファイル名部分は以前のバージョンの InstallShield Professional のように無視されません。代わりに szTargetFile はすべて各ソースファイルを既存の名前を使ってコピーするターゲットパスとして処理されます。たとえば、CopyFile("C:¥¥*.*","D:¥¥File.txt") はファイルを D ドライブの File.txt フォルダーにコピーします。

- WaitOnDialog では、エンドユーザーがシステムメニューの [閉じる] コマンドを選択するか、タイトルバーの [閉じる] ボタンをカスタム ダイアログからクリックすると、IDCANCEL を返します。以前のバージョンでは、このような場合に DLG_CLOSE が返されていました。

- GetSystemInfo を DRIVE の最初の引数と UNC パスの 3 番目の引数に渡すと、関数は正しく負の値を返します。これまでは成功を示すゼロが間違って返されていました。オペレーティング システム制限のため、UNC パスはこの場合サポートされていませんでした。

- 3 番目の引数を PATH に設定し、2 番目の引数を行末の円記号(例 "¥¥¥TheServer") がないサーバー名だけを含む UNC パスを設定して、ParsePath を呼び出すと、最初の引数が二重の円記号 ("¥¥¥") に正しく設定されます。Professional 5.x ではこの関数は、最初の引数を単一の円記号("¥") にしていました。

- Professional 5.x ヘルプライブラリには、成功を表す 0 (ゼロ)以外の特定の数値の戻り値がリストされています。これらの数値の中には、InstallShield 2025 で変更されたものもあります。InstallShield 2025 で使用されているスクリプトは、その関数のヘルプに現在記載されている定数とだけその戻り値を比較します。

- エンドユーザーのインターフェイスがスムーズに行われるようにするため、インストール初期化ダイアログ ボックスは、スクリプトによってダイアログ ボックスが表示されるまでデフォルトで表示されたままになり

ます。インストール初期化ダイアログを閉じるには、Disable(DIALOGCACHE) またはダイアログ関数を呼び出します。

- ComponentMoveData の 2 番目の引数は有用なデータを返すことはなくなりました。
- SendMessageTimeout の4番目の引数としてポインターを文字列に渡しても、期待される結果は今後得られません。文字列データを SendMessageTimeout を渡すには、SendMessageTimeout トピックの追加情報セクションを参照してください。

定義済み定数

- 次の定数を削除します。これらの関数は、Professional 5.x ではサポートされていましたが、InstallShield 2025 ではサポートされていません。

  COMPONENT_VALUE_ALWAYSOVERWRITE

  COMPONENT_VALUE_NEVEROVERWRITE

  COMPONENT_VALUE_NEWERDATE

  COMPONENT_VALUE_NEWERVERSION

  COMPONENT_VALUE_OLDERDATE

  COMPONENT_VALUE_OLDERVERSION

  COMPONENT_VALUE_SAMEORNEWDATE

  COMPONENT_VALUE_SAMEORNEWERVERSION

  COMPONENT_FIELD_DESTINATION

  COMPONENT_FIELD_OVERWRITE

- あらかじめ定義されている以下の変数 (TARGETDISK="C:¥¥"; など) に値を割り当てるようなステートメントは削除します。このようなステートメントは、InstallShield 2025 ではコンパイルされません。これらの定数は読み取り専用です。

  CMDLINE

  COMMONFILES

  ERRORFILENAME

  FOLDER_DESKTOP

  FOLDER_PROGRAMS

  FOLDER_STARTMENU

  FOLDER_STARTUP

  ISRES

  ISUSER

  ISVERSION

  MODE

  PROGRAMFILES

SUPPORTDIR

TARGETDISK (TARGETDIR が変更されると、この変数の値も自動的に更新されます。)

UNINST

WINDIR

WINDISK

WINSYSDIR

WINSYSDISK

DLL

- Professional 5.x で DLL 関数を呼び出すと、すべての文字列引数は参照として渡されます。InstallShield 2025 では文字列引数を値で渡すことができます。引数を渡すメソッドを指定するには、関数プロトタイプで、MyDLL.MyFunc(BYREF NUMBER, BYVAL STRING) のように、引数のデータ型の前に BYREF または BYVAL キーワードを付けます。InstallShield 2025 コンパイラには適切なキーワードの使用を推奨するための次のようなコンパイラ警告が含まれています。
  - BYREF も BAVAL も指定されていない場所でスクリプトが DLL 関数呼び出しを含む場合、コンパイラ警告 W7 507 が発生します。この警告を削除するには、関数プロトタイプに適切なキーワードを追加します。
  - 参照として渡された文字列引数のリテラル文字列を指定する場合、コンパイラ警告 W7 511 が発生します。この警告を削除するには、関数プロトタイプに BYVAL キーワードを追加します。

  外部 DLL で関数を呼び出す場合、スクリプトと DLL で同じ呼び出し規則が使われていることを確認してください。Professional 5.x ではインストールエンジンで常に stdcall 規則を使用していましたが、一貫性のない DLL 規則を無視することがありました。ただし InstallShield 2025 エンジンでは、間違った呼び出し規則が使用されるとエラー (例外 0x80040704) が発生します。

  DLL 関数を宣言するとき、スクリプトで呼び出し規則の cdecl または stdcall を宣言できます。例:

  prototype cdecl MyDLL.MyFunction (INT, INT);

  呼び出し規則を明示的に宣言しなかった場合、InstallShield は stdcall を使用します。
- InstallShield 2025 では、InstallScript 配列は、内部的には C または C++ 配列ではなく OLE オートメーション セーフ配列としてフォーマットされます。InstallScript 配列を C/C++ 配列が必要な DLL 関数に渡すためには、GetCArrayFromISArray を呼び出すことによって配列データを要求フォーマットに配置しなければなりません。
- InstallShield 2025 では、アドレスがポインター変数に保存されている文字列変数に関して、ポインターを DLL 関数に渡して変更することはできません。DLL 関数で文字列変数値を変更できるようにするには、BYREF 演算子を指定して引数のデータタイプを宣言してから、変数そのものを引数として関数に渡します。
- Windows の DLL 検索パスにある DLL を呼び出す前に、UseDLL 関数を呼び出す必要はありません。DLL 検索パスは、次のとおりです。
  1. ロードされたアプリケーションからのフォルダー
  2. 現在のフォルダー
  3. 32 ビットの Windows システム フォルダー
  4. 16 ビットの Windows システム フォルダー
  5. Windows フォルダー

6. PATH 環境変数に一覧表示されているフォルダー

現在のフォルダーとエンド ユーザーのマシンにある PATH 環境変数の値は予期できないため、この機能の使用には注意してください。

その他

- 明記されていない配列構文 ArrayName[nArrayIndex] はサポートされていません。InstallShield 2025 では、すべての配列の構文は ArrayName(nArrayIndex) で表されます。
- スクリプトで定義された関数の戻り値の種類を明白に指定する場合、関数定義でキーワードの "function" の後に同じ戻り値の種類を指定する必要があります。これを行わないと、スクリプトを完了できません。(戻り値のタイプを明白に指定しない場合は、戻り値タイプは NUMBER であると想定されます。)以前のバージョンの InstallShield Professional では、戻り値の型は実際に指定された値に関係なく NUMBER と想定されたため、これは問題になりませんでした。
- HKEY_CURRENT_USER の下にアンインストール情報を配置するには、ALLUSERS システム変数を FALSE に設定します。
- リテラル文字を引数としてビルトイン関数またはユーザー定義関数に渡す場合、ASCII 数値を渡す必要があります。
- システム変数である CMDLINE を処理する際、コマンドラインスイッチ (-SMS、-z、-c、-e、-q、-t、および -x) が Setup.exe では現在使用されていないため、CMDLINE に含まれていることに注意してください。-SWS スイッチは、インストールが完了するまで Setup.exe がメモリに残るようになったため、使用されていません。また、256 メガバイト以上のメモリを搭載したシステムでも Setup.exe は正しく初期化できるため、-z スイッチは、現在は使用されていません。
- InstallShield 2025 ではコンパイル不可能な Professional 5.x の ODBC テンプレート コードを削除します。これは、次のような関数呼び出しがコード内でネストされるからです。

```
ComponentGetData ( szMedia, szCompName,
   COMPONENT_FIELD_DESTINATION, nvData, svDest );
```

InstallShield 2025 では、Destination プロパティがコンポーネント プロパティではなく、ファイル グループ プロパティであるため、この関数の呼び出しはコンパイルされません。

他の変更

- インストール終了に F3 を使うことはできません。
- Setup.exe で -f スイッチを使用して名前が変わった Setup.inx を使うことはできません。
- InstallShield 2025 インストールは低解像度システムの特別ビルボードなどをサポートしていません。
- ソース ファイルも既存のターゲット ファイルにもバージョン番号がなく、ソース ファイルのファイル グループの "上書き" プロパティが [新しいバージョン]、[同じか新しいバージョン]、[古いバージョン] のいずれかだった場合、ターゲット システムのファイルは上書きされません。Professional 5.x インストールではターゲットファイルが上書きされます。
- InstallShield 2025 は、Windows 95 形式のダイアログの表示はサポートしていません。
- サイレントインストールの応答ファイル (.iss) の [Application] セクションにある Lang キーは何も効果を持たなくなりました。
- In Professional 5.x インストールでは、インストーラーによってログされたレジストリキーは、アンインストール中、すべて無条件で削除されます。InstallShield 2025 では、RegDBCreateKeyEx で作成されたキーおよ

びレジストリセットの中の非共有キーは、インストーラーがそのキーを作成した場合のみ、つまり、インストールが実行されたときキーが存在しなかったときのみ削除されます。

スクリプトの変更: 語彙変換

用語の変更

InstallShield Professional では、エンド ユーザーの視点から見たインストールの機能的な構成要素をコンポーネントと呼んでいました。InstallShield 2025 では、これらのビルドブロックを機能と呼びます。

このように用語が変更されたため、多くの InstallScript 関数名も変更されました。InstallShield Professional を使用して作成したプロジェクトをアップグレードすると、スクリプトの項目の一部は用語変更され、一部はエイリアスが作成されます。

- 関数名
- ComponentFileInfo (FeatureFileInfo) フラグ
- ComponentSetData (FeatureSetData) および ComponentGetData (FeatureGetData) フラグ

関数名

コンポーネントを参照していたすべての InstallScript 関数名は、機能を参照するようになります。たとえば、ComponentDialog は FeatureDialog になります。これらの関数のパラメーターには変更はありません。機能関数をクリックすると、対応するヘルプトピックを表示できます。

テーブル 15 ・ 関数名の変更

コンポーネント関数	機能関数
ComponentAddItem	FeatureAddItem
ComponentCompareSizeRequired	FeatureCompareSizeRequired
ComponentDialog	FeatureDialog
ComponentError	FeatureError
ComponentErrorInfo	FeatureErrorInfo
ComponentFileEnum	FeatureFileEnum
ComponentFileInfo	FeatureFileInfo
ComponentFilterLanguage	FeatureFilterLanguage
ComponentFilterOS	FeatureFilterOS
ComponentGetData	FeatureGetData
ComponentGetItemSize	FeatureGetItemSize

テーブル 15 • 関数名の変更 (続き.)

コンポーネント関数	機能関数
ComponentGetTotalCost	FeatureGetTotalCost
ComponentInitialize	FeatureInitialize
ComponentIsItemSelected	FeatureIsItemSelected
ComponentListItems	FeatureListItems
ComponentLoadTarget	FeatureLoadTarget
ComponentMoveData	FeatureMoveData
ComponentPatch	FeaturePatch
ComponentReinstall	FeatureReinstall
ComponentRemoveAll	FeatureRemoveAll
ComponentRemoveAllInLogOnly	FeatureRemoveAllInLogOnly
ComponentRemoveAllInMedia	FeatureRemoveAllInMedia
ComponentRemoveAllInMediaAndLog	FeatureRemoveAllInMediaAndLog
ComponentSaveTarget	FeatureSaveTarget
ComponentSelectItem	FeatureSelectItem
ComponentSelectNew	FeatureSelectNew
ComponentSetData	FeatureSetData
ComponentSetTarget	FeatureSetTarget
ComponentSetupTypeEnum	FeatureSetupTypeEnum
ComponentSetupTypeGetData	FeatureSetupTypeGetData
ComponentSetupTypeSet	FeatureSetupTypeSet
ComponentTotalSize	FeatureTotalSize
ComponentTransferData	FeatureTransferData
ComponentUpdate	FeatureUpdate
ComponentValidate	FeatureValidate

テーブル 15 • 関数名の変更 (続き.)

コンポーネント関数	機能関数
SdComponentDialog	SdFeatureDialog
SdComponentDialogAdv	SdFeatureDialogAdv
SdComponentMult	SdFeatureMult
SdComponentTree	SdFeatureTree

ComponentFileInfo (FeatureFileInfo) フラグ

InstallShield Professional の ComponentFileInfo (現在は FeatureFileInfo) 関数で使用していたフラグは、InstallShield 2025 の機能で使用できるように変換されます。

テーブル 16 • ComponentFileInfo フラグ

使用不可のフラグ	新しいフラグ
COMPONENT_INFO_ATTRIBUTE	FEATURE_INFO_ATTRIBUTE
COMPONENT_INFO_LANGUAGE	FEATURE_INFO_LANGUAGE
COMPONENT_INFO_OS	FEATURE_INFO_OS
COMPONENT_INFO_ORIGSIZE	FEATURE_INFO_ORIGSIZE
COMPONENT_INFO_COMPSIZE **COMPONENT_INFO_DATE** **COMPONENT_INFO_DATE_EX** **COMPONENT_INFO_TIME**	 ***メモ•*** *利用不可能なフラグは、オプションが機能していないことを示す -137 を戻します。FeatureError 関数を利用して、戻り値についての詳しい情報を提供することができます。*
COMPONENT_INFO_VERSIONLS	FEATURE_INFO_VERSIONLS
COMPONENT_INFO_VERSIONMS	FEATURE_INFO_VERSIONMS
COMPONENT_INFO_VERSIONSTR	FEATURE_INFO_VERSIONSTR

ComponentSetData (FeatureSetData) および ComponentGetData (FeatureGetData) フラグ

InstallShield Professional の ComponentSetData および ComponentGetData（現在 FeatureSetData および FeatureGetData）関数で使用していたフラグは、InstallShield 2025 の機能で使用できるように変換されます。

テーブル 17 • 使用できない ComponentSetData と ComponentGetData フラグ

使用不可のフラグ	新しいフラグ
COMPONENT_FIELD_CDROM_FOLDER	FEATURE_FIELD_CDROM_FOLDER
COMPONENT_FIELD_DESCRIPTION	FEATURE_FIELD_DESCRIPTION
COMPONENT_FIELD_DISPLAYNAME	FEATURE_FIELD_DISPLAYNAME
COMPONENT_FIELD_FILENEED	FEATURE_FIELD_FILENEED
COMPONENT_FIELD_FTPLOCATION	FEATURE_FIELD_FTPLOCATION
COMPONENT_FIELD_HTTPLOCATION	FEATURE_FIELD_HTTPLOCATION
COMPONENT_FIELD_IMAGE	FEATURE_FIELD_IMAGE
COMPONENT_FIELD_MISC	FEATURE_FIELD_MISC
COMPONENT_FIELD_PASSWORD	FEATURE_FIELD_PASSWORD
COMPONENT_FIELD_SELECTED	FEATURE_FIELD_SELECTED
COMPONENT_FIELD_SIZE	FEATURE_FIELD_SIZE
COMPONENT_FIELD_STATUS	FEATURE_FIELD_STATUS
COMPONENT_FIELD_VISIBLE	FEATURE_FIELD_VISIBLE
COMPONENT_VALUE_CRITICAL	FEATURE_VALUE_CRITICAL
COMPONENT_VALUE_HIGHLYRECOMMENDED	FEATURE_VALUE_HIGHLYRECOMMENDED
COMPONENT_VALUE_STANDARD	FEATURE_VALUE_STANDARD

プロジェクト アシスタント ダイアログ サポートを InstallShield Professional プロジェクトからアップグレードされたプロジェクトに追加する

インストール プロジェクトを InstallShield Professional から InstallShield にアップグレードした場合で、そのプロジェクトに InstallShield Professional で作成した Setup.rul ファイルが含まれるとき、プロジェクト アシスタントの [インストール インタビュー] ページの質問で利用できないものもあります。これは、Setup.rul ファイルから InstallShield がスクリプトタグでキーとして利用する OnFirstUIBefore イベントコードの一部が欠落しているためです。

新規 InstallScript プロジェクトを作成して OnFirstUIBefore イベントを確認すると、ダイアログ関数をサポートするために [プロジェクト アシスタント] が利用する次のコードが見つかります。

```
//{{IS_SCRIPT_TAG(FunctionName)
   InstallScript コード...
//}}IS_SCRIPT_TAG(FunctionName)
```

タスク ***[プロジェクト アシスタント] ダイアログ サポートをアップグレードしたプロジェクトへ追加するには、以下の手順を実行します。***

1. **[InstallScript]** ビューで、Setup.rul ファイルを開きます。
2. OnFirstUIBefore イベントコードをコメントアウトします。
3. InstallScript ペインの上部にあるイベント カテゴリ リストで OnFirstUIBefore イベントを選択します。InstallShield がスクリプトへこのイベントを再び追加します。
4. 適切なダイアログコードを OnFirstUIBefore イベントのコメントアウトコードにコピーします。
5. ステップ 3 で追加した OnFirstUIBefore イベントコードを削除します。
6. 残りの OnFirstUIBefore イベントコードからコメントを削除します。

InstallShield—Windows Installer Edition で作成したプロジェクトをアップグレードする

InstallShield–Windows Installer Edition で作成した InstallShield プロジェクト (.ism ファイル) がある場合、そのプロジェクトは InstallShield で開くと自動的に基本の MSI プロジェクトにアップグレードされます。アップグレードしたプロジェクトを InstallScript MSI プロジェクトに変換する方法については、「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。

タスク ***プロジェクトを開くと、プロジェクトをアップグレードするかどうかたずねるダイアログ ボックスが表示されます。***

- *[はい] をクリックしてプロジェクトをアップグレードします。プロジェクトが IDE で開きます。*
- *プロジェクトを開かないでおく場合は [いいえ] をクリックします。*

元のプロジェクトのバックアップ コピーは、このダイアログで指定した場所に作成および保管されます。

コマンドラインからのセットアップのアップグレード

セットアップ プロジェクトをコマンドラインからビルドしたときもプロジェクトをアップグレードできます。セットアップ プロジェクトをビルドせずにコマンドラインからアップグレードする場合は、-u パラメーターを使用します。

アップグレードエラーのトラブルシューティング

アップグレードが原因で発生したエラーがあれば、IDE 下部の出力パネルに表示されます。これらのエラーについての詳細は、「アップグレード エラーと警告 (InstallShield—Windows Installer Edition からのアップグレード)」を参照してください。

プロパティ マネージャーでのアップグレード後の変更

プロパティマネージャーで空の値 (値なし) のあるプロパティは MSI には無効です。そのため、それらのプロパティの空の値に代わって ***DO_NOT_BUILD*** が表示されます。このようにすることにより、アップグレードの後、空の値のあるプロパティはプレースホルダーとして保存されますが、msi パッケージには組み込まれません。

InstallShield の他のエディションからアップグレードする

InstallShield には、Premier、Professional、および Express の 3 つのエディションが用意されています。

Premier Edition のみで提供されている機能

次の一覧は、Professional Edition または Epress Edition ではなく、Premier Edition のみで使用可能な機能です:

- **スイート/アドバンスト UI インストールの作成およびビルド機能**—複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx) および Windows Installer トランザクションだけでなく、複数の InstallShield 前提条件にも対して、最新のカスタマイズ可能なユーザー インターフェイスを持つブートスラップ アプリケーションを作成することができます。スイート/アドバンスト UI インストールでは、複数の個別のインストールを、統合されたユーザー インターフェイスを使用する単一のインストールにパッケージ化し、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。
- **スイート/アドバンスト UI インストールで InstallScript アクション、およびその他の種類のアクションをサポート** — スイート/アドバンスト UI インストールでは、今回より、インストールに含めるパッケージの範囲を超えた様々な実行時タスク行うための InstallScript アクションを起動できるビルトイン サポートが提供されています。アクションを使って、実行可能ファイルを実行、DLL 関数を呼び出し、PowerShell スクリプトを実行、スイート/アドバンスト UI プロパティを設定、InstallScript コードを実行、またはマネージ アセンブリでパブリック メソッドを呼び出すことができます。
- **仮想化サポート** — Microsoft App-V アシスタントは、InstallShield Premier Edition に含まれています。このアシスタントを使って、Microsoft App-V 形式で、カスタム仮想アプリケーションを作成することができます。仮想化技術を使って、アプリケーションを独自の環境に隔離することで、既存アプリケーションとの競合を回避したり、基盤となるオペレーティング システムの変更を避けたりすることが可能です。
- **アプリケーション仮想化適合性スイート**—InstallShield では、製品を仮想化するための準備が整っているかどうかを判断するのに役立つ、新しい検証スイートが提供されています。これらのスイートに含まれている InstallShield 仮想化内部整合性検証ツール (ISVICE) を使って、Microsoft App-V 4.x、Microsoft App-V 5、

Microsoft Server App-V、VMware ThinApp、および Citrix XenApp との適合性をチェックできます。顧客に仮想バージョンの提供を考慮する場合に、検証スイートを使用して、製品をどのようにビルドするか豊富な情報に基づいた意思決定を行うことができます。

- **DIM ファイルのサポート** – DIM プロジェクトの作成機能はInstallShield Premier Edition で使用できます。このサポートは、コラボレーション アドオン、InstallShield Developer Installation Manifest Editor でも提供されています。DIM ファイルを基本の MSI プロジェクトに追加する機能は、InstallShield Premier Edition で提供されています。

  DIM プロジェクトは、機能サイズのプロジェクトで、インストール パッケージの別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数の基本の MSI プロジェクトでそれらを参照することができます。

- **複数言語インストール** – 複数の言語でエンドユーザー テキストを表示する単一のインストール プロジェクトを作成し、言語固有のファイルの条件付きインストールを処理できます。翻訳済みの文字列を利用して、ダイアログおよびメッセージを 34 の追加言語の 1 つに変更することができます。

プロジェクト・*アラビア語 (サウジアラビア) とヘブライ語の 2 つの言語サポートは、基本の MSI とマージ モジュール プロジェクトでのみ利用できます。*

- **アラビア語 (サウジアラビア) とヘブライ語サポート** – InstallShield Premier Edition は、右から左方向へ読み書きするアラビア語 (サウジアラビア) とヘブライ語のサポートを含みます。デフォルトのエンド ユーザー ダイアログ文字列のすべてが、これらの言語で利用できます。

  これらの言語は右から左方向に読まれるため、Premier Edition では、アラビア語とヘブライ語ダイアログのミラーリングが サポートされています。このサポートにより、アラブ語とヘブライ語のダイアログでは、右から左方向へのレイアウトが使用されます。たとえば、英語やその他の左から右に読まれる言語のダイアログで右側にあるボタンは、右から左に読まれる言語のダイアログでは左側に移動されます。

- **ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する機能**– InstallShield Premier Edition には、ビルド プロセスの様々な段階で実行するコマンドを指定できるリリース設定が含まれています。次のビルド イベントで実行するコマンドをスケジュールすることができます: (a) InstallShield がリリースのビルドを開始する前、(b) InstallShield が .msi パッケージおよび (製品のデータ ファイルが .cab ファイルに格納される場合) .cab ファイルをビルドした後、.msi パッケージに署名が行われて、Setup.exe ファイルにストリームされる前、(c) InstallShield がリリースをビルドならびに署名を行った後。

- **ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにインストールを配布できる機能** – インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態になるようプロジェクトを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。このテスト準備機能により、手作業で行う処理を減らし、テストにかかる時間を短縮できます。VM は Microsoft Hyper-V Server、VMware ESX または ESXi Server、または VMware Workstation を指定できます。

- **InstallShield MSI ツール用の追加のライセンス** – InstallShield には、InstallShield MSI Diff、InstallShield MSI Query、InstallShield MSI Sleuth、および InstallShield MSI Grep 用のライセンスが追加で含まれています。これらのツールを利用して、Windows Installer パッケージに関連する問題を解決することができます。InstallShield Premier Edition には、別のマシンに InstallShield 以外のツールのみをインストールできる、個別のインストールおよび追加ライセンスが含まれています。詳しい使用条件については、InstallShield MSI ツールの使用許諾契約書を参照してください。

- **既存の IIS Web サイトからプロジェクトへの IIS データのインポート機能** – InstallShield では、IIS スキャナー (IISscan.exe) が提供されています。このコマンドライン ツールを使って、既存の IIS Web サイトをスキャンして、Web サイトに関する IIS データを記録できます。IIS スキャナーは、Web サイト、その仮想ディレクトリ、アプリケーション、およびアプリケーション プールすべての設定を含む XML ファイルを作成します。この XML ファイルを使って、InstallShield Premier Edition の [IIS 構成] ビューに IIS データをインポートできます。IIS データをプロジェクトにインポートしてから、必要に応じて、[IIS 構成] ビューを使って IIS の設定を変更できます。
- **Web 配置パッケージの配置をサポート** – InstallShield Premier Edition で作成可能なスイート/アドバンスト UI インストールでは、Web 配置パッケージを IIS Web サーバーまたはクラウドに配置できるビルトイン サポートを利用できます。
- **InstallShield Collaboration** – InstallShield Premier Edition には、InstallShield Collaboration for Visual Studio のライセンスが含まれています。
- **ネットワーク リポジトリ** – ネットワーク リポジトリは、複数のインストール作成者が必要に応じアクセスでき、プロジェクトで再利用することができるインストール要素のコレクションです。ネットワーク リポジトリは、インストール作成者同士のコラボレーション作業を促進し、ネットワークに格納されます。
- **InstallShield ベスト プラクティス スイート**– InstallShield には、InstallShield ベスト プラクティス スイートという名前の 1 セットの検証ツールが含まれています。インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの InstallShield ベスト プラクティス (ISBP) 検証ツールによって警告されます。
- **追加のダイアログ テーマ** – 一部のダイアログ テーマは InstallShield Premier Edition のみで提供されています。

Premier Edition および Professional Edition のみで使用できる機能

以下は、Premier Edition および Professional Edition で提供されていて、Express Edition では提供されていない機能の一部です:

- **アドバンスト UI インストールの作成およびビルド機能**–単一の .msi パッケージ、.msp パッケージ、および InstallScript パッケージだけでなく InstallShield 前提条件にも対して、最新のカスタマイズ可能なユーザー インターフェイスを持つブートスラップ アプリケーションを作成することができます。アドバンスト UI では、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。

  (InstallShield Premier Edition の スイート/アドバンスト UI 機能には、この機能に対して、さらに拡張サポートが追加されています。スイート/アドバンスト UI 機能を利用すると、複数の .msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installer トランザクションだけでなく、複数の InstallShield 前提条件も、モダンでカスタマイズ可能なユーザー インターフェイスを持つ単一のアプリケーションにパッケージできます。)
- **Standalone Build**– InstallShield Premier Edition および Professional Edition で使用可能なこのツールは、InstallShield のインストールをビルドする機能のみ、および選択した任意の再配布可能ファイルをビルドマシン上にインストールすることができます。Standalone Build 用の追加ライセンスを購入することもできます。
- **ダイアログ エディター** – ダイアログ エディターを利用して、既存のエンド ユーザー ダイアログのレイアウトを変更したり、新規カスタム ダイアログを作成することができます。プロジェクト全体でダイアログをインポートおよびエクスポートして、共有することができます。プロジェクトでサポートしている各言語用に、別のダイアログを作成します。
- **InstallShield MSI ツール**– InstallShield Premier Edition および Professional Edition には、InstallShield MSI Diff、InstallShield MSI Query、InstallShield MSI Sleuth、および InstallShield MSI Grep が含まれています。これらのツールを利用して、Windows Installer パッケージに関連する問題を解決することができます。

- **オートメーション インターフェイス** – スクリプトを指定して、新しいファイルの追加、機能の追加または削除、製品名とアップグレード コードの変更、リリースの設定の変更、概要情報ストリーム項目の変更、リリース フラグの変更、プロパティの変更、ビルド処理の開始などを行うことができます。
- **リリースのカスタマイズ** – プロジェクトのどの部分を圧縮し、どの機能をどのディスクに配置して、どの言語を含めるか定義します。ローカライズ作業をサポートするために、言語に基づいてアプリケーションデータをフィルタリングしてください。
- **ソース コード管理の統合** – プロジェクトをソース コード管理システムにチェックインおよびチェックアウトするプロセスを単純化し、プロジェクトの差分を検出するときに容量を節約します。InstallShield Premier Edition および Professional Edition で SCC の統合機能は、様々なソース コード コントロール システムとの統合をサポートします。
- **柔軟なローカリゼーション サポート**– InstallShield Premier Edition および Professional Edition には、[文字列エディター] ビューが搭載されていて、インストール中、実行時に表示される翻訳可能なテキスト文字列すべてを一箇所にまとめて管理できます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。このビューを使って、文字列エントリを翻訳可能な形式にエクスポートしてから、翻訳済み文字列をプロジェクトにインポートすることもできます。
- **プロジェクト検証** – 標準 .cub ファイルを使って、インストールとマージ モジュールを検証します。アップグレードとパッチの検証を利用して、潜在的に存在するアップグレードに関する問題を検出し、リリースされる前にそれらを解決します。
- **パッチの作成** – QuickPatch プロジェクトの作成のほか、Premier Edition と Professional Edition では、製品の以前のバージョンに対するアップデートを含む標準パッチを作成することができます。
- **複数製品バージョンの管理** – 1 つのプロジェクトから、評価版、デバッグ版、標準版、拡張版などのバージョンをビルドします。ユーザー定義フラグを使って、リリースに特定の機能、InstallShield 前提条件、およびその他の要素を選択 (または除外) することができます。
- **InstallScript のサポート**– InstallShield Premier Edition と Professional Edition では、シンプルかつパワフルなプログラム言語 InstallScript がサポートされています。InstallScript カスタム アクションを Windows Installer ベースのインストールに追加したり、Windows Installer エンジンの代わりに InstallScript エンジンを使用してインストール全体を制御する InstallScript プロジェクトを作成したりできます。
- **柔軟なカスタム アクション サポート**– InstallShield Premier Edition および Professional Edition は、Express Edition では提供されていない種類のいくつかのカスタム アクションをサポートします。これらの追加カスタム アクションを利用して、プロパティの設定、ディレクトリの設定、マネージ アセンブリ内でのパブリック メソッドの呼び出し、または特定の条件下でエラーメッセージを表示してインストールを中止することが可能です。
- **柔軟なショートカット サポート**– InstallShield Premier Edition および Professional Edition は、Express Edition では構成することができない、ショートカットの様々な詳細設定を構成するためのサポートが含まれています。たとえば、Premier Edition と Professional Edition では、プロジェクト内のショートカットをタスクバーまたは [スタート] メニューにピン留めしないように防ぐことができます。また、エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐことができます。これらのショートカット オプションは、しばしばインストールの一部となるツールまたは付属的な製品に使用されます。
- **マージ モジュールの作成と編集** – 他のアプリケーションのインストール パッケージで再利用できるように、プロジェクトの小さい単位をパッケージすることができます。独自に作成したもの、または、製品に含まれるすべてのモジュールを再利用できます。さらに詳細なカスタマイズを行うために、モジュールを開いて編集することができます。

- **プロジェクトのテンプレート** – インストール プロジェクトやマージ モジュールプロジェクトの作成するとき、その出発点として利用できるすべてのデフォルト設定とデザイン要素を含むプロジェクト テンプレートを作成できます。
- **複数の IIS Web サイト** – InstallShield Express および Limited Edition では、1 つのインストールによって 1 つの Web サイトのみインストールできます。Premier Edition および Professional Edition では、1 つのインストールによって複数の Web サイトをインストールできます。
- **IIS アプリケーション プールと Web サービス拡張のサポート** – IIS アプリケーション プールと Web サービス拡張を管理します。
- **Windows サービスの詳細サポート**– InstallShield Express Edition では、サービスに関するいくつかの作業がサポートされていますが、 Premier Edition および Professional Edition では、サービスに関する追加の柔軟性が提供されています。たとえば、Premier Edition および Professional Edition ではインストールおよびアンインストール中にサービスを開始、停止、または削除することができます。サービスはインストールの一部である場合と、ターゲット システムに既存する場合があります。これらのエディションではまた、Windows Installer 5 で利用可能な拡張サービス カスタマイズ オプションを構成することもできます。
- **SQL サポート** – SQL サーバーへの接続、データベース スキーマおよびデータのインポート、SQL スクリプトと機能の関連付けなど多数サポートされています。
- **テキスト ファイルまたは XML ファイルを変更できる機能** – [テキスト ファイルの変更] ビューまたは [XML ファイルの変更] ビューを使って、実行時にターゲット システム上で変更するファイルを構成できます。
- **64 ビット専用サポート** – Windows Server Core は、WOW64 (32-bit Windows-on-Windows) サポートの無効化をサポートします。この構成がより一般化するのに伴い、64 ビットのアプリケーションが 32 ビットの機能に全く依存しないでインストールできるようにすることが必要となります。これを実現するため、InstallShield Premier Edition および InstallShield Professional Edition では 64 ビット専用 .msi パッケージのビルドが可能です。これらは、WOW 64 機能を搭載していない 64 ビット版の Windows ベースのシステム上で実行します。
- **InstallShield 前提条件エディター** – このツールを使用して、新しい InstallShield 前提条件を作成したり、既存の前提条件を変更したりできます。
- **Setup.exe および Update.exe のカスタム アイコン–ビルド時に作成する** – Setup.exe および Update.exe ファイルに使用するカスタム アイコン (.exe、.dll、または .ico ファイル) を指定します。アイコンは Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。エンド ユーザーが Windows Explorer で Setup.exe ファイルを参照したときにも、このアイコンが表示されます。
- **Setup.exe の有効期限日** – Setup.exe の有効期限日と有効期限メッセージを設定します。エンド ユーザーが Setup.exe をプロジェクトで指定された日付以降に実行すると、有効期限切れメッセージが表示されて、インストールが終了します。
- **トランザクション処理を使った複数パッケージのインストールをサポート** – Windows Installer 4.5 以降では、トランザクション処理を使った複数パッケージのインストールをサポートします。InstallShield Premier Edition および Professional Edition では、インストール プロジェクトに連鎖 .msi パッケージを追加できます。パッケージ並びに追加 .msi パッケージは連鎖されて、単一トランザクションとして処理されます。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。
- **様々なプロジェクト要素をエクスポートおよび再利用できる機能** – 既存のプロジェクトの一部 (ダイアログ、カスタム アクション、または機能) をマージ モジュールまたは別のインストール プロジェクトに移動して、効率を高めます。

- **複数インスタンスのサポート** – エンド ユーザーが製品の複数のインスタンスを同じマシン上に同じユーザー コンテキストでインストールできるインストールを作成します。
- **デバイス ドライバー サポート** – Premier Edition および Professional Edition に搭載されているデバイス ドライバー サポートは、Microsoft の DIFxApp (Driver Installation Frameworks for Applications) を使って、インストールからデバイス ドライバーをインストールする処理を簡素化します。
- **追加のダイアログ テーマ**– いくつかのダイアログ テーマは InstallShield Premier Edition および Professional Edition のみで提供されています。Limited および Express Edition で提供されてるテーマは 2 種類だけです。
- **Visual Studio マージ モジュール プロジェクトの変換**– InstallShield Premier および Professional Edition では、Visual Studio マージ モジュール プロジェクトを InstallShield マージ モジュール プロジェクトに変換することができます。この処理は、他のプロジェクトで消費するマージ モジュールをビルドするのに必要です。
- **COM+ アプリケーション プロキシ サポート** – インストール中に COM+ アプリケーション プロキシを管理します。COM+ アプリケーション プロキシはサーバー アプリケーション属性のサブセットで構成され、これはクライアン ト コンピューターからアプリケーションが存在するマシンへのリモート アクセスを可能にします。

各エディションで提供されている機能についての詳細情報は、InstallShield セールスに問い合わせるか、http://www.installshield.com にアクセスしてください。

エディションからエディションへのアップグレードについて詳しく知りたいときは、該当するトピックを参照してください。

- Express Edition から Professional Edition または Premier Edition にアップグレードする
- Professional Edition から Premier Edition にアップグレードする

Express Edition から Professional Edition または Premier Edition にアップグレードする

InstallShield Express Edition を使って Express プロジェクトが作成された場合、同じプロジェクトを InstallShield Premier Edition または Professional Edition で開くと、基本の MSI プロジェクトにアップグレードされます。このアップグレードは、基本の MSI が Express プロジェクトの種類に対して同様のプロジェクトの種類なため、実行されます。Express Edition で作成された QuickPatch プロジェクトは、InstallShield Premier Edition まてゃ Professional Edition で QuickPatch エディションのままです。

InstallShield Premier または InstallShield の Express Edition で作成されたプロジェクトを開くと、そのファイルは .ise ファイルから .ism ファイルへ変換されます。.ise ファイルのバックアップは、変換が行われる前に保存されます。

Express Edition プロジェクトからのビルボードのプロパティは、基本の MSI プロジェクトでサポートされていないので、移行されません。ビルボード プロパティは Express Edition プロジェクトで指定されている場合、警告 701が出されます。

タスク ***Express Edition プロジェクトをアップグレードするには、以下の手順の従います。***

InstallShield Premier Edition または Professional Edition でプロジェクトを開いて下さい。プロジェクトをアップグレードするかどうかたずねるダイアログ ボックスが開きます。

- **[はい]** をクリックしてプロジェクトをアップグレードします。プロジェクトが InstallShield で開きます。

- プロジェクトを開かないでおく場合は **[いいえ]** をクリックします。

元のプロジェクトのバックアップコピーは、このダイアログ ボックスで指定した場所に作成および保管されます。

Professional Edition から Premier Edition にアップグレードする

Premier Edition で Professional Edition プロジェクトを開くとき、プロジェクトの変換は必要ありません。プロジェクトは、Premier Edition でシームレスに開き、Premier Edition 限定の機能を利用することができます。特定の機能に関する詳しい情報は、「InstallShield の他のエディションからアップグレードする」を参照してください。

各エディションで提供されている機能についての詳細情報は、http://www.installshield.com にアクセスしてください。

Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする

Visual Studio は、制限付きでセットアップおよびマージ モジュール プロジェクトの作成をサポートします。InstallShield では、これらの種類の Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートすることで、InstallShield に搭載されている拡張機能と使ってインストールとマージ モジュールを作成することができます。

InstallShield では、次を行うことができます:

- Visual Studio セットアップ プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートするか、Visual Studio マージ モジュール プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポートする。これらのタスクを行うと、Visual Studio プロジェクトに含まれている同じデータと設定を含む InstallShield インストール プロジェクトおよびマージ モジュール プロジェクトを作成できます。インポート処理中に、Visual Studio プロジェクト内の特定の設定について、それらをインポートするか無視するかを選択できます。
- Visual Studio セットアップ プロジェクトを InstallShield 基本の MSI プロジェクトに変換するか、Visual Studio マージ モジュール プロジェクトを InstallShield マージ モジュール プロジェクトに変換する。その他のプロジェクトで使用するマージ モジュールをビルドするためには、Visual Studio マージ モジュール プロジェクトを InstallShield マージ モジュール プロジェクトに変換する必要があります。

メモ: *Visual Studio プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合、InstallShield を使ってその Visual Studio プロジェクトを InstallShield プロジェクトにインポートできますが、InstallShield がその Visual Studio プロジェクトを InstallShield プロジェクトに変換することはできません。*

サポート対象の Visual Studio バージョン:

- Visual Studio 2010
- Visual Studio 2008
- Visual Studio 2005
- Visual Studio .NET 2003

- Visual Studio .NET

Visual Studio プロジェクトの代わりに InstallShield プロジェクトを使用することの利点

Visual Studio セットアップ プロジェクトを InstallShield プロジェクトに変換またはインポートすると、InstallShield の機能を使ってプロジェクトをカスタマイズすることができます。

以下は、InstallShield プロジェクトで実行可能で Visual Studio プロジェクトでは実行不可能ないくつかのタスクの一覧です:

- 視覚的なダイアログ エディターを使用して、ダイアログのレイアウトを修正する。
- コンポーネントと機能を管理する。
- カスタム アクションを .msi または .msm データベースの Binary テーブルに格納する。(Visual Studio では、すべてのカスタム アクションは製品と共にインストールされる必要があります。)
- SQL サーバーの接続や SQL スクリプトの実行など、SQL 関係のタスクを管理する。
- IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を管理する。
- ファイル転送中に表示するビルボードの追加。
- 実行時にターゲット システムで XML ファイルまたはその他のテキスト ファイルを修正する。
- ターゲット システムに既に存在するファイルのショートカットを作成する。
- COM+ アプリケーションを管理する。

インポート処理

InstallShield を使って、Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield 基本の MSI またはマージ モジュール プロジェクトにインポートできます。

メモ: *InstallShield プロジェクトにインポートする Visual Studio セットアップ またはマージ モジュール プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合、その InstallShield は、Visual Studio セットアップまたはマージ モジュール プロジェクトおよびそのプロジェクトのすべての依存関係を含む、同じ Visual Studio ソリューションに含まれていなくてはなりません。*

プロジェクト出力を含む Visual Studio プロジェクトをインポートするには、Visual Studio 内部から InstallShield を使用しなくてはなりません。InstallShield プロジェクトが InstallShield で開かれていて、Visual Studio 内部から開かれていない場合に、プロジェクト出力を含む Visual Studio プロジェクトを InstallShield プロジェクトにインポートしようとすると、エラーが発生します。

タスク ***Visual Studio プロジェクト (.vdproj) を InstallShield プロジェクト (.ism) にインポートするには、以下の手順に従います:***

1. InstallShield で、基本の MSI またはマージ モジュール プロジェクト開く、または作成します。
2. **[プロジェクト]** メニューで、**[Visual Studio デプロイメント プロジェクト ウィザード]** ボタンをクリックします。

3. Visual Studio デプロイメント プロジェクト ウィザードのパネルを完成させる。

InstallShield は、ウィザードで構成された設定に基づいて Visual Studio プロジェクトを現在開かれている InstallShieldプロジェクトにインポートします。InstallShield がプロジェクトをインポートするとき、[出力] ウィンドウにプロジェクトのインポート ステータスが表示されます。出力ウィンドウでは、変換プロセスの各ステップが表示され、すべての変換エラーと警告が一覧で表示されます。

変換プロセス

InstallShield を使って、Visual Studio セットアップ プロジェクトを変換すると、InstallShield では InstallShield 基本の MSI プロジェクト (.ism) が作成されます。

InstallShield を使って、Visual Studio マージ モジュール プロジェクトを変換すると、InstallShield では InstallShield マージ モジュール プロジェクト (.ism) を作成します。

タスク ***Visual Studio プロジェクト (.vdproj) を InstallShield プロジェクト (.ism) に変換するには、以下の手順に従います:***

1. InstallShieldを開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. **[ファイルの種類]** ボックスで、**Visual Basic セットアップ プロジェクト(*.vdproj)** を選択します。
4. 開く Visual Studio プロジェクトの場所を参照し、プロジェクト ファイルを選択します。
5. **[開く]** ボタンをクリックします。

InstallShield は、Visual Studio プロジェクトの設定に基づいて InstallShield プロジェクトを作成します。InstallShield は、.ism ファイルを .vdproj ファイルと同じフォルダーに格納します。.ism ファイルが作成されると、出力ウィンドウにプロジェクト変換のステータスが表示されます。出力ウィンドウでは、変換プロセスの各ステップが表示され、すべての変換エラーと警告が一覧で表示されます。

変換処理が成功すると、その新しい InstallShield プロジェクトが InstallShield で表示されます。

インポート後および変換後のタスク

前提条件タスク

Visual Studio では、1 つ以上の定義済み前提条件を Visual Studio セットアップ プロジェクト内の 1 つ以上の構成に追加することができます。InstallShield のインポートおよび変換プロセスは、すべての構成に含まれるすべての前提条件を、対応する InstallShield 前提条件に変換しようとします。InstallShield に対応する InstallShield 前提条件が含まれていない場合、警告 -9071 が発生して、その前提条件が変換されなかったことを通知します。この警告を解決するためには、必要な再配布可能ファイルをインストールする InstallShield 前提条件を作成して、それをプロジェクトに追加する方法を考慮してください。詳細については、「InstallShield 前提条件を定義する」および「インストール プロジェクトに含まれている InstallShield 前提条件を利用する」を参照してください。

ユーザー インターフェイス タスク

インポートおよび変換処理では、Visual Studio プロジェクトからのダイアログを InstallShield プロジェクトに組み込みません。プロジェクトをインポートまたは変換した後、InstallShield の [ダイアログ] ビューでプロジェクトに含まれるダイアログの設定を構成することができます。

言語のタスク

Visual Studio プロジェクトを InstallShield プロジェクトにインポートしたときに、以下の条件がある場合、InstallShield はプロジェクトに含まれる既存の文字列エントリ値を Visual Studio プロジェクトの言語のデフォルト文字列エントリ値で置換します:

- Visual Studio デプロイメント プロジェクト ウィザードで、Visual Studio プロジェクトの言語をインポートすることを指定した。
- Visual Studio プロジェクトの言語が InstallShield プロジェクトのデフォルト言語と一致しない。(Visual Studio で、Localization プロパティがプロジェクトの言語を示します。InstallShield では、[文字列エディター] ビューの "デフォルト言語" 設定が、プロジェクトのデフォルト言語を示します。)
- InstallShield は、Visual Studio プロジェクトで使用されている言語をサポートします。(InstallShield Profesioanl Edition を使用している場合、Visual Studio プロジェクトで使用した言語がサポートされていない場合があります。)

たとえば、Visual Studio デプロイメント プロジェクト ウィザードで Visual Studio プロジェクトの言語をインポートする場合に、InstallShield プロジェクトの言語がスペイン語で、Visual Studio プロジェクトの言語がドイツ語のとき、InstallShield Premier Edition はスペイン語のランタイム文字列をデフォルトのドイツ語翻訳と置換します。このため、[一般情報] ビュー内の "発行者" 設定を更新している場合など、文字列エントリの値を編集した後にウィザードで Visual Studio プロジェクトの言語をインポートすると、InstallShield は "発行者" 設定の値およびその他の文字列の値を、デフォルトのドイツ語の文字列エントリの値に置換します。

したがって、Visual Studio プロジェクトのインポート中にプロジェクト言語を変更する場合は、[一般情報] ビューと [文字列エディター] ビューで設定を確認の上、適切な場合は文字列エントリを変更してください。

.NET インストーラー クラスのタスク

Visual Studio プロジェクトに .NET インストーラー クラス カスタム アクションが含まれる場合、InstallShield が変換処理中に、.NET インストーラー クラス カスタム アクションを作成する代わりに、.NET アセンブリのコンポーネントの .NET インストーラー クラス情報を構成します。InstallShield は、Visutal Studio における .NET インストーラー クラス カスタム アクションの Condition プロパティをサポートしません。このため、Visual Studio プロジェクトに条件がある .NET インストーラー クラス カスタム アクションが含まれている場合、プロジェクトを変換したあと、InstallShield の [コンポーネント] ビューを使って、.NET アセンブリを含むコンポーネントの条件を作成したほうが良い場合があります。

追加のタスク

InstallShield の他のビューを利用すると、プロジェクトに追加の変更も行うことができます。

メモ: *Visual Studio では、アプリケーション フォルダーに複数のフォーマットされたプロパティが含まれたディレクトリ パス (例、[ProgramFilesFolder][Manufacturer]¥[ProductName]) を指定できます。Visual Studio プロジェクトは、実行時にディレクトリ カスタム アクションを使用して、パスを解決します。ただし、InstallShield では、この種類のディレクトリ パスはサポートされていません。したがって、InstallShield はパスを変換処理中に解決し、パスの* INSTALLDIR *プロパティを使用します。*

InstallShield のアップデートを取得する

インターネットに接続されている場合、ご使用中の InstallShield バージョンで利用可能な最新の InstallShield 前提条件、マージ モジュール、オブジェクト、およびサービス パック、パッチ、またはその他のアップデートを確認および取得することができます。

タスク ***アップデートを確認するには、以下の手順に従います:***

アップデートが使用可能な時に、InstallShield でアップデート ダイアログが表示されます。アップデートを取得するには、次のステップを実行します:

- システムに対するアップデートを確認する。
- アップデートに関する説明およびリリース ノートを確認する。
- アップデートをダウンロードおよびインストールします。

InstallShield 実行時の言語サポート

重要: *言語サポートは、ご使用中の InstallShield のエディション、プロジェクト タイプ、および InstallShield のインターフェイス言語 (英語、日本語) によって異なります。*

アラビア語 (サウジアラビア) とヘブライ語は、Premier Edition の基本の MSI、マージ モジュール、およびスイート/アドバンスト プロジェクトでのみ使用できます。

InstallShield Premier Edition には、35 ヶ国語のデフォルト ランタイム文字列がサポートされています。サポート対象の言語をプロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield はプロジェクトにその言語に翻訳された文字列エントリを追加します。それらは、デフォルト ダイアログ、メッセージ、およびその他のエンド ユーザー インターフェイス要素の文字列エントリです。サポート対象言語の一覧は、「言語識別子」を参照してください。

InstallShield Premier Edition ではまた、基本の MSI、InstallScript、InstallScript MSI、および スイート/アドバンスト UI プロジェクトで 新しい言語ウィザード を使って、サポート対象の 35 ヶ国語以外の言語を追加することができます。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をプロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

InstallShield Professional Edition の英語版では、33 ヶ国語 (アラビア語とヘブライ語を除くすべてのサポート対象言語) のうち 1 つの言語を、作成するすべてのプロジェクトのランタイム言語として使用できます。InstallShield Professional Edition の英語は、選択した 1 つの言語用のデフォルトのラインタイム文字列を含みます。

InstallShield Professional Edition の日本語版では、英語に加えてもう 1 つ追加サポート対象言語を選択できます。つまり、InstallShield Professional Edition の日本語版を使用している場合、英語のランタイム文字列を使ったプロジェクト、およびもう 1 つのサポート対象言語のランタイム文字列を使ったプロジェクトを作成できます。
InstallShield Professional Edition の日本語版には、英語のデフォルト ランタイム 文字列と、選択可能な追加のサポート対象言語が 1 つ含まれています。

デフォルトのランタイム文字列を変更したい場合、またはカスタム ダイアログ、メッセージ、および他のエンド ユーザー インターフェイス要素をプロジェクトに追加する場合、カスタム ランタイム文字列を InstallShield プロジェクトに直接入力することができます。別の手段として、言語の文字列エントリをファイルにエクスポートして、そのファイルの文字列値を翻訳してから、プロジェクトにそれをインポートすることもできます。この機能は、すべての InstallShield エディションでサポートされています。

InstallShield の言語サポートについての詳細は、ドキュメントの次のセクションを参照してください:

- エンドユーザー インターフェイスのローカライズ
- 複数言語インストールの作成
- 言語 サポートのコード ページ要件
- 言語の設定

言語 サポートのコード ページ要件

2 バイト文字を含む言語 (日本語、ギリシャ語、韓国語など) を使ったランタイム文字列を含むリリースの作成を予定している場合は、ビルド マシンに特定のコード ページをインストールする必要があるかどうかを判断してください。

InstallShield ではランタイム文字列の表示に Unicode エンコーディングが使用されます。Unicode エンコーディングを使うと、プロジェクトのターゲット言語がインストールされていないシステム上で InstallShield を使用している場合でも、InstallShield 内では 2 バイト文字が正しく表示されます。さらに、リリースのターゲット言語がインストールされていないシステム上でリリースをビルドすることもできます。

ただし、ANSI エンコードをランタイム文字列に使用する場合 (たとえば、プロジェクトから文字列エントリを ANSI テキスト ファイルとしてエクスポートしてから、翻訳済みの ANSI テキスト ファイルをプロジェクトにインポートする場合) は、ビルド マシンにこれらの言語用のコード ページをインストールする必要があります。コードページによって、システムがそれらの言語の文字を正確に表すことができるようになります。ビルド マシンにコード ページがインストールされていない場合、リリースをビルドすると InstallShield がビルド エラーを報告し、インストールは実行時に 2 バイト文字を正しく表示しません。

ヒント: *翻訳のために文字列エントリをエクスポートするとき、ANSI エンコードの代わりに Unicode エンコードを使用することが推奨されます。*

ビルド マシン上に補足言語サポートをインストールする

以下は、Windows XP が搭載されたビルド マシン上にコード ページをインストールする方法を説明します。これは、中国語、日本語、および韓国語を含むリリースをビルドする場合に必要になります。新しいオペレーティング システムでは、一般的にコード ページがデフォルトでインストールされます。

タスク ***Windows XP が搭載されたビルド マシン上にコード ページをインストールするには、以下の手順に従います:***

1. **[コントロール パネル]** で **[地域と言語のオプション]** を選択します。
2. **[言語]** タブをクリックします。
3. **[詳細]** ボタンをクリックします。

4. **[追加]** ボタンをクリックします。
5. コードページを追加する言語を選択します。
6. **[OK]** をクリックします。

次の手順は、アラビア語やヘブライ語など、右から左に読み書きする言語を Windows XP が搭載されたビルド マシン上にインストールする方法を説明します。新しいオペレーティング システムでは、一般的にこれらのファイルがデフォルトでインストールされます。これらのファイルがインストールされていない場合、[文字列エディター] ビューで文字列が右から左ではなく、左から右に表示されます。

タスク ***右から左に読み書きする言語のサポートを Windows XP が搭載されたビルド マシンにインストールするには、以下の手順に従います:***

1. **[コントロール パネル]** で **[地域と言語のオプション]** を選択します。
2. **[言語]** タブをクリックします。
3. **[補足言語サポート]** 領域で **[複雑なスクリプトおよび右から左に読み書きする言語用のファイルをインストールする]** チェック ボックスを選択します。
4. **[追加]** ボタンをクリックします。
5. コードページを追加する言語を選択します。
6. **[OK]** をクリックします。
7.

サポートされているアプリケーション プログラミング言語

InstallShield は、Java, Pascal, C++, Visual Basic, Delphi, C# .NET, Visual Basic .NET, ASP .NET, および Cobol など、*任意*のプログラム言語でアプリケーションのインストールを作成できます。

上記の言語以外でアプリケーションを作成した場合でも、InstallShield 使ってそのアプリケーションのためのインストールを作成することができます。また、アプリケーションを配布はしないけれども、ファイルやデータベースをパッケージしたい場合、InstallShield を作業のために使用できます。

3

チュートリアル

このセクションでは、InstallShield を使ったインストールの作成手順をステップ バイ ステップで説明します。

テーブル 1 ・ チュートリアルの種類

チュートリアル	説明
InstallScript プロジェクト チュートリアル	InstallScript インストール プロジェクトの作成手順を、ステップ バイ ステップでご案内します。また、このチュートリアルでは、インストール プロジェクトに関するさまざまなタスクの情報と、これらのタスクを行うために使用する InstallShield インターフェイス内の各ビューについて紹介します。
基本の MSI プロジェクト チュートリアル	基本の MSI インストール プロジェクトについて説明します。また、このチュートリアルでは、インストール プロジェクトに関するさまざまなタスクの情報と、これらのタスクを行うために使用する InstallShield インターフェイス内の各ビューについて紹介します。
グローバリゼーション チュートリアル	言語および言語固有のコンポーネントをプロジェクトに追加する方法を説明します。さらに、ターゲット システムの言語に基づいてコンポーネントを条件付きでインストールする方法も紹介します。 ***メモ:*** *このチュートリアルを完了するには、開発システム上で InstallShield Premier Edition のインストールが必要です。*

InstallScript プロジェクト チュートリアル

このチュートリアルでは、InstallShield を使用した InstallScript インストール プログラムの作成、ビルド、実行、およびそ強化方法について順を追って説明します。

チュートリアルの構成は複数のステップに分かれています。最初のステップ（ステップ 1: プロジェクトの作成、ビルドおよびテスト）を除き、残りのステップはすべて独立した内容になっているので、これらは特に順番にこだわることなく現在の作業に必要な情報のみを調べることができます。

本チュートリアルでは、インストールが取り扱うタスクの多くの処理方法について解説します。

- ファイルのインストール
- ショートカットおよびレジストリ データの設定
- データの条件付きインストール
- インストールのユーザー インターフェイスの変更
- リリース イメージのビルド
- インストールのテスト

チュートリアルを通して、ヘルプライブラリの該当箇所へのリンクが設けてあります。

ステップ 1: プロジェクトの作成、ビルドおよびテスト

このステップでは、インストール プロジェクトの作成、リリース イメージのビルド、およびインストールのテストについて説明します。本ステップを一通り完了すると、以下の項目がマスターできます。

- プロジェクト アシスタントを使った新規プロジェクトの作成法。
- インストール プロジェクトのグローバル プロパティの指定方法。
- セットアップタイプ、機能、コンポーネント、およびファイルリンクの指定法。
- 複製および配布用のリリース イメージのビルド方法。
- InstallShield ユーザー インターフェイスからのインストールの実行方法。

インストールは 3 つのレベルで構成されます。

テーブル 1 ・ インストール内のレベル

レベル	説明
コンポーネント	コンポーネントとは、開発者から見て、個別にインストール可能な最小の製品構成単位のことです。コンポーネントは、ファイル、ショートカット、レジストリデータなど、ターゲット システムにインストールされるデータを指定します。ユーザーがコンポーネントを直接処理することは決してありません。 1 つのコンポーネントを複数の機能に配置することもでき、ユーザーがこのコンポーネントと関連付けられた機能の 1 つを選ぶと、このコンポーネントのデータがインストールされます。

テーブル 1・インストール内のレベル (続き.)

レベル	説明
機能	機能とは、ユーザーから見て、個別にインストール可能な最小の製品構成単位のことです。ユーザーが [カスタム] セットアップ タイプを選択すると、どの機能をインストールするかをユーザーが選択することができるダイアログが表示されます。 各機能はコンポーネントを含みます。
セットアップの種類	セットアップの種類とは、事前に定義された機能のコレクションです。デフォルトで、インストールは [完全] および [カスタム] のセットアップの種類を提示し、ユーザーはどのセットアップの種類でインストールするかを SetupType ダイアログで選択します。

本チュートリアルで作成するインストールは、*Tutorial App* という名前のアプリケーションについて、そのインストールと設定を行うものです。Tutorial App のソース ファイルは InstallShield Program Files フォルダーの Samples サブフォルダーにあります。デフォルト インストール先は次の場所です:

C:¥Program Files¥InstallShield¥2020¥Samples¥WindowsInstaller¥Tutorial Project

プロジェクト アシスタントを使ったプロジェクトの作成

InstallShield を起動した後、次の手順に従って新規 InstallScript プロジェクトを作成します。

1. 次のひとつを行って [**新規プロジェクト**] ダイアログ ボックスを開きます:
   - **[スタートページ] の [プロジェクト タスク]** セクション (ページの左側) にある **[新規プロジェクトの作成]** をクリックします。
   - **[ファイル]** メニューで、**[新規]** をクリックします。
   - ツールバーの **[新規プロジェクト]** ボタンをクリックします。
2. **[新規プロジェクト]** ダイアログ ボックスで、**InstallScript** タブをクリックします。
3. **InstallScript タブ**のリスト ビューで、**[InstallScript プロジェクト]** アイコンを選択します。
4. ここでは [**プロジェクト名**] 編集ボックスに、プロジェクト名として *Tutorial* と入力します。
5. **[OK]** をクリックします。

プロジェクトの作成については InstallShield Professional で作成したプロジェクトを更新するなど、他にも多くの方法があります。詳細については、「新しいプロジェクトの作成」を参照してください。

InstallShield は、*ProjectName*.ism という名前のプロジェクト ファイルを作成します。この場合、Tutorial.ism となります。InstallShield ユーザー インターフェイスで行ったすべての設定は、プロジェクトファイルに保管されます。プロジェクトを他のマシンに移動する場合は、移動先のシステムに .ism ファイル (およびインストールソースファイル) をコピーします。

ヒント: *新規プロジェクトファイルが作成されるディレクトリを変更するには、[ツール] メニューから [オプション] を選択し、[ファイルの場所] タブをクリックして表示される "プロジェクトの場所" フィールドに、新しいディレクトリを指定します。*

新規プロジェクトは、その [プロジェクト アシスタント] タブに開きます。プロジェクト アシスタントを利用するには、右下角にある [次へ] ボタンをクリックします。

ヒント: *プロジェクト アシスタントのステップの順序は問いません。また（適切なタブをクリックすることで）プロジェクト アシスタントとインストール デザイナーとを自由に切り替えることができ、インストール プロジェクトへさらに複雑でパワフルな要素を追加することが可能です。*

アプリケーション情報を入力する

[アプリケーション情報] ページでは、インストールするアプリケーションについての一般情報を設定することができます。

タスク ***このチュートリアルでは次を行います:***

1. **[会社名を入力してください]** 編集ボックスに名前「*Tutorial Co*」を入力します。
2. [**アプリケーション名を入力してください**] 編集ボックスに名前 *Tutorial App* を入力します。
3. その他のフィールドはそのままにします。

"アプリケーション名" フィールドへの入力値は、エンドユーザーに対して表示されるダイアログで、コントロール パネルの [プログラムの追加と削除] でのアプリケーションの表示名として表示されます。入力するアプリケーション名と会社名は Windows スタートメニュー上のアプリケーションのショートカットのデフォルトの場所、並びにコンポーネントファイルのデフォルトのインストール先を指定する *TARGETDIR* システム変数のデフォルト値を決定します。

インストール アーキテクチャのカスタマイズ

[インストール アーキテクチャ] ページでは、インストールに表示させる機能を指定することができます。機能とは、ユーザーから見て個別にインストール可能な最小の製品構成単位のことです。カスタム セットアップの種類を選択すると、個々の機能がエンドユーザーに対して表示されます。

メモ: *機能には、サブ機能やサブサブ機能などを含めることも可能で、インストールの要求に応じて必要なレベルの多層構造化が可能です。*

タスク ***このチュートリアルでは次を行います:***

1. **[インストールアーキテクチャをカスタマイズしますか？]** という質問には **[はい]** を選択します。
2. 既存 **DefaultFeature** 機能を選択して、その名前を **ProgramFiles** に変更します。
3. **HelpFiles** という新しい機能を作成します。そのためには **[インストール アーキテクチャ]** をクリックしてから **[新規]** ボタンをクリックします。

プロジェクトへファイルを追加する

[アプリケーション ファイル] ページでは、各機能にリンクさせるファイルを指定することができます。

まず最初に、機能の一覧の中から、ファイルを挿入する機能を選択します。ファイルリンクを追加するには、[ファイルの追加] ボタンをクリックし、選択されている機能に含めるファイルを選びます (複数指定可能)。

タスク ***このチュートリアルでは、次の手順に従って、ProgramFiles 機能に Tutorial.exe というファイルを追加します:***

1. 機能の一覧表で、**ProgramFiles** をクリックします。
2. ツリー コントロール (トップノードは[インストール先コンピューター]) 内にある **[アプリケーション ターゲット フォルダー]** を選択します。
3. **[ファイルの追加]** をクリックします。
4. ソース ディレクトリに置かれた Tutorial.exe を参照します。
5. 「追加したファイル...は依存関係を持つ可能性があります」メッセージが表示されたとき、**[いいえ]** をクリックします。Tutorial.exe には依存関係は含まれません。

ショートカットの作成

[アプリケーション ショートカット] ページでは、ターゲット システムのデスクトップ又は [スタート] メニュー上にあるアプリケーションファイルのショートカットを指定することができます。デフォルトで、このページはインストール プロジェクトに含んだ各実行可能ファイルのショートカットを表示します。これらを削除して、インストール プロジェクトに含めた別のファイルへのショートカットを追加することができます。

このチュートリアルでは、このページのデフォルトの仕様 ([スタート] メニューの Tutorial.exe へのショートカット) をそのまま利用します。

レジストリ データを構成する

[アプリケーション レジストリ] ページでは、アプリケーションに必要なレジストリ エントリを指定することができます。

メモ: *InstallScript プロジェクトは、デフォルトでアプリケーション アンインストールキー (必要に応じて HKEY_LOCAL_MACHINE の下の Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥<GUID> または HKEY_CURRENT_USER ルートキー) を作成するスクリプト コード、その値、およびデータを含みます。これらのレジストリ エントリを指定する必要はありません。*

このチュートリアルでは、このページでレジストリ エントリの指定を行いません。レジストリのエントリについては、

チュートリアルの ステップ 2 ″ショートカットとレジストリ データ″ で説明されています。

インストール インタビューを使ってダイアログを選択する

[インストール インタビュー] ページでは、インストールの実行時にエンドユーザーに対して表示するダイアログ ボックスを選択します。このページの質問に対する回答に基づいて、プロジェクト アシスタントは対応するダイアログ関数をインストールスクリプトへ追加します。ダイアログに関連するスクリプトの変更については、チュートリアルのステップ 6で解説します。

タスク ***このチュートリアルでは次を行います:***

1. "使用許諾契約書ダイアログを表示しますか?" というテキストの下にあるオプションボタンで **[いいえ]** を選択します。
2. その他のオプションボタンを **[はい]** のままにします。

インストール用の言語を選択する

[インストール ローカリゼーション] ページでは、インストールがサポートする言語を指定することができます。別の言語に簡単にローカライズすることができるようにエンドユーザー インターフェイスで利用可能な文字列値および関連する識別子を指定することもできます。

エディション: *複数言語サポートは、InstallShield の Premier Edition で使用することができます。*

タスク ***このチュートリアルでは、次の要領で HelpFiles 機能の表示名を変更します:***

1. リストボックスで、**"機能文字列データ"** を選択します。右の表は、すべての機能文字列エントリを表示します。
2. **[値]** 列で、**HelpFiles** (識別子 IDS_FEATURE_DISPLAY_NAME2 に関連付けられている値) をクリックして、**Help Files** (空白スペースを挿入) に変更します。

インストールをビルドする

[インストールのビルド] ページでは、どの配布タイプをビルドするのかを指定できます (オプションで、配布ファイルをコピーする場所も指定できます)。

タスク ***このチュートリアルでは次を行います:***

1. **CD-ROM** オプションを選択します。
2. **[インストールのビルド]** をクリックします。

上部に [ビルド] タブを持つ出力ウィンドウが開き、ビルドの進行状況についての情報を表示します。[ビルド] タブに「ビルドが終了しました。 *日付と時刻*」が表示されたとき、ビルドは終了します。

インストールを実行する

インストールをInstallShield 内から実行するには、[実行] ツールバーボタンをクリックするか、 CTRL+F5 を押します。

インストールは、プロジェクト アシスタントの [インストール インタビュー] ページで選択したダイアログを表示します。プロジェクト アシスタントで入力した値は、適切なダイアログでエンドユーザーに対して表示されます。たとえば、プロジェクト アシスタントで指定した INSTALLDIR のデフォルト値は、実行時に [インストール先の選択] ダイアログで表示されます。このときエンドユーザーが異なるインストール先ディレクトリを選ぶと、TARGETDIR にはこの新しい値が格納されます。

インストールの終了後、インストール先のディレクトリを開いて、セットアッププログラムによってインストールされたファイルを確認してください。インストールが成功していれば、チュートリアルファイルがインストールされているはずです。

メンテナンス モード

既に製品がインストールされているシステム上で、ユーザーが 2 回目 (およびそれ以降) にインストールを実行をすると、インストールは*メンテナンス モード*で行われます。このメンテナンスモードを使って、ユーザーは初回インストールからの機能選択の変更、インストール済みの機能の修復、およびプログラム全体の削除が行えます。

プログラムのアンインストール

プログラムをアンインストールするには、[実行] ボタンをクリックして (または CTRL+F5 を押し)、[セットアップ メンテナンス] ダイアログから [削除] を選択します。これで、[プログラムの追加と削除] でアプリケーションを選択した場合と同じ結果になります。

基本のインストール プロジェクトを作成したあと、[インストール デザイナー] タブ (ウィンドウの上部) をクリックして、チュートリアルの 次のステップ で説明されているようにインストールを拡張、または微調整します。

InstallShield インターフェイスを使って作業する

基本のインストール プロジェクトを作成したあと、[インストール デザイナー] タブ (ウィンドウの上部) をクリックして、InstallShield のユーザー インターフェイスでプロジェクトを拡張、または微調整します。InstallShield のユーザー インターフェイスは機能カテゴリー別に並べられているので、プロジェクトに情報を簡単に追加または編集できます。ここから先のチュートリアルでは、異なる InstallShield ビューのいくつかを紹介します。

本ステップを一通り完了すると、以下の項目がマスターできます。

- プログラムの機能の表示プロパティの設定法。
- プログラムのセットアップの種類の定義。
- コンポーネントおよびファイル リンクの作成法。

機能のプロパティを設定する

まず最初に行うのは、プロジェクト アシスタントで作成した機能に対する追加プロパティの設定で、これには機能の表示名や説明などがあります。機能プロパティを編集するには、IDE の [機能] ビューへ移動します。

タスク ***機能のプロパティを設定するには、以下の手順に従います:***

1. **[移動]** メニューで、**[編成]** をポイントして **[機能]** をクリックします。
2. **[機能]** エクスプローラーで、DefaultFeature 機能を選択して **"説明"** プロパティを **[この機能は Tutorial App のプログラムファイルを含んでいます]** に設定します。
3. [新規機能] 機能を選択して **"説明"** プロパティを **[この機能は Tutorial App のヘルプ ファイルを含んでいます]** に設定します。各説明を入力すると、InstallShield はそれぞれの値を表すための文字列テーブルエントリ（{ID_STRING*n*} と表示）を作成します。
4. [**機能**] ビューにある機能の名前を、それぞれの表示名と同じ名前に変更します。機能の名前を変更するには、機能を 2 回クリックして名前をハイライト表示し、新しい名前を入力します。

実行時、エンド ユーザーが [カスタム] セットアップ タイプを選択した場合、インストールはユーザーがどの機能をインストールするかを選択できるダイアログを表示します。このダイアログには、ここで設定した機能の表示名および説明が表示されます。

"セットアップの種類" プロパティの設定

セットアップの種類はインストールされる機能のコレクションです。一般的なインストールは [完全] と [カスタム] セットアップ タイプを提供します。[完全] セットアップ タイプはすべての機能をインストールするのに対して、[カスタム] セットアップ タイプはエンド ユーザーがインストールする機能を選択することができるダイアログを表示します。

IDE の [セットアップの種類] ビュー（[ビュー リスト] の [編成] ノードの下）内にあるセットアップの種類プロパティを変更します。

タスク ***各セットアップの種類について、機能名の前にあるボックスを選択することでインストールする機能を選択します:***

1. [完全] セットアップの場合、両方の機能を選択します。
2. [カスタム] セットアップの場合、両方の機能を選択します。

コンポーネントおよびファイルリンクを作成する

[ファイルとフォルダー] ビューの追加ファイルにリンクを作成することができます。このステップでは、HelpFiles 機能にファイルを 1 つ追加します。[ファイルとフォルダー] ビューでファイルを追加すると、セットアップベストプラクティス規則に従ってIDE がコンポーネントを作成します。

タスク ***ヘルプファイル機能にある新規コンポーネントにソースファイル Tutorial.html を追加するには、以下の手順を実行します。***

1. **[ファイルとフォルダー]** ビュー（[ビュー リスト] の **[アプリケーション データ]** ノードの下）を開きます。
2. ビューの最上部にある機能のリストから、**ヘルプファイル** を選択します。

3. [インストール先コンピューターのフォルダー] 領域で、[**インストール先コンピューター**] アイコンを右クリックし、[コンポーネントの表示] が選ばれていることを確認します。
4. **[アプリケーション ターゲット フォルダー]** アイコンを右クリックし、**[新規コンポーネント]** を選択します。
5. 新しいコンポーネントの名前を HelpComponent に変更します。
6. [ソース コンピューターのフォルダー] 領域で、TutorialHelp.html が含まれるソースフォルダーを選びます。
7. [ソース コンピューターのファイル] 領域にある TutorialHelp.html アイコンを、**HelpComponent** アイコンまでドラッグします。

コンポーネントにリンクされるファイルのリストに変更が無い、この種のファイルリンクを「*スタティック ファイル リンク*」と呼びます。ビルド毎に内容に違いがあるディレクトリ(及びそのサブディレクトリ)にリンクするには、「ダイナミック ファイル リンク」を参照してください。

ヒント: *InstallShield の依存関係スキャナーを使うと、アプリケーションについて現段階でプロジェクトに含められていない必要な追加ファイルを確認できます。たとえば、Tutorial App は MFC を使用するので、MFC のランタイムライブラリがインストールされていないシステムをターゲットにする場合は、[再配布可能ファイル] ビューでプロジェクトに MFC ランタイムオブジェクトを加える必要があります。*

チュートリアルの次のステップでは、インストール プロジェクトのリリースイメージのビルド方法について説明します。

リリースのビルド

インストールをテストする前に、リリースをビルドする必要があります。リリースイメージには、CD-ROM、フロッピーディスク、またはネットワーク経由でエンドユーザーに配布するすべてのファイルが含まれます。

リリース ウィザードを利用すると、最も簡単に新しいリリースを作成することができます。リリース ウィザードでは、使用するメディアの種類 (たとえば CD-ROM) や、メディア上のファイルを圧縮するかなど、リリースのプロパティを指定します。リリース ウィザードの起動は、[リリース ウィザード] ツールバーボタンをクリックするか、[ビルド] メニューから [リリース ウィザード] を選択して行います。

[ようこそ] パネルで [次へ] をクリックして、必要なリリース設定を行います。いずれのパネルでも [ヘルプ] をクリックすると、現在のステップに関する説明が表示されます。

リリースに名前を付ける

[リリースの指定] パネルでは、リリース名を指定します。リリース名は、ビルドされたリリースを配置するフォルダーの名前として使用されます。このサンプルでは、*cdrom* という名前で新規にリリースを作成してください。

メディアタイプと一般オプションの選択

[メディアの種類] パネル

[メディア タイプ] パネルでは、リリースをビルドするメディアの種類を指定します。ここで指定するメディアの種類は、リリース ウィザードが作成するディスク イメージ フォルダーのサイズを示します。たとえば CD-ROM 用にリリースをビルドすると、リリース ウィザードがディスクイメージを複数のフォルダーに分割し、それぞれのフォルダーサイズを 650MB以下とします。

ここでのチュートリアルでは、CD-ROM を選択してください。

[次へ] をクリックして、作成するリリースの一般オプションを指定してください。

[一般オプション] パネル

[一般オプション] パネルでは次の処理が可能です。

- インストールを配布するための自己展開型実行可能ファイルを作成する。
- コマンドライン オプションを Setup.exe へ渡す
- プリプロセッサ変数定義をコンパイラへ渡す。
- コンパイルしたスクリプト ファイル (.inx ファイル) をキャビネット ファイルに配置するかどうかを選択する。

このチュートリアルでは、パネルのデフォルト設定をそのまま利用します。

パスワードとサポートするプラットフォームを指定する

[パスワード] パネル

[パスワード] パネルではインストールのパスワードを指定することができます。またその場合 OnCheckMediaPassword イベント ハンドラー関数のデフォルトコード内でパスワード確認コードを実行するかどうかも指定することができます。

このチュートリアルでは、パスワードの指定は行いません。

[次へ] をクリックして、現在のリリースでサポートするオペレーティング システムを指定します。

[プラットフォーム] パネル

[プラットフォーム]パネルでは、現在のリリースでサポートするオペレーティング システムを指定できます。

このチュートリアルでは、デフォルトの選択 (**"プラットフォーム" プロジェクト プロパティで指定されたプラットフォームを使用"**) をそのまま利用します。

セットアップ言語と含める機能を指定する

[セットアップ言語] パネル

[セットアップ言語] パネルでは、インストールを実行することが可能な言語の指定や、エンドユーザーによるセットアップが表示する言語の選択を可能にするダイアログを表示するかどうかを指定することができます。

ウィザードはインストールに、このパネルで選択した言語固有の要素（文字列エントリやダイアログなど）だけを組み込みます。製品プロパティやビルトインアクションなど、すべての言語非依存のリソースは当然含まれます。

このチュートリアルでは、デフォルトの選択のままにしてください。

[次へ] をクリックして、現在のリリースに含む機能を指定します。

[機能] パネル

[機能] パネルでは、ビルドしたリリースに含める機能を指定することができます。

このチュートリアルでは、デフォルトの選択（**選択するかどうかの判別に "ビルドに含める" 機能プロパティを使用**）をそのまま利用します。

メディア レイアウトとダイアログの外観を定義する

[メディア レイアウト] パネル

[メディア レイアウト] パネルでは、各機能またはすべての機能に対して、機能のファイルをキャビネット フォルダーに保存するか、ディスク イメージ内に圧縮せずに配置するかを指定します。

このチュートリアルでは、デフォルトの選択（キャビネット ファイル）をそのまま利用します。

[ユーザー インターフェイス] パネル

[ユーザー インターフェイス] パネルでは、インストールのエンド ユーザーダイアログの外観を指定します。

このチュートリアルでは、デフォルトの選択のままにしてください。

インターネット オプションの指定およびアプリケーションのデジタル署名

[インターネット オプション] パネル

[インターネットオプション] パネルでは、様々なインターネット関連オプションを指定できます。すべてのリリースは、メディアの種類を問わずインターネット上で実行できます。

このチュートリアルでは、[セットアップのデフォルトの Web ページを作成する] ボックスをチェックして、その他のデフォルトの選択はそのままにします。

[デジタル署名] パネル

[デジタル署名] パネルでは、アプリケーションにデジタル署名を行うことができます。アプリケーションにデジタル署名をすることで、アプリケーション内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証することができます。

このチュートリアルでは、デフォルトの選択のままにしてください。

アップデートとポスト ビルド情報を指定する

[アップデート] パネル

[アップデート] パネルでは、リリースの形式と現在のリリースをアップデートとして実行できる既存のリリースを指定できます。

このチュートリアルでは、デフォルトの選択のままにしてください。

[ランチャ オプション] パネル

[ポストビルド] オプション パネルでは、リリースのビルドが完了した後に、ディスク イメージ フォルダーをフォルダーまたは FTP サイトにコピーするか、またはバッチ ファイルを実行します。

このチュートリアルでは、デフォルトの選択のままにしてください。

設定を確認する

[概要] パネルは、現在のリリースに関する リリース ウィザードの設定を表示します。設定が正しい場合、[リリースの ビルド] チェック ボックスを選択して [完了] をクリックし、リリースをビルドします。そうでない場合は、[戻る] をクリックして設定を修正してください。

進行中のビルドに関するステータスメッセージは、アウトプットウインドウに表示されます。

ビルドが完了すると、CD にコピーするファイルは、次のディレクトリに置かれます:

<ProjectFolder>¥Tutorial¥cdrom¥DiskImages¥DISK1.

チュートリアル後半のステップで行うプロジェクトに対する変更が終了したら、最新リリースの再ビルドを行います。この処理は、[ビルド] ツールバーボタンをクリックし、[ビルド] メニューから [ビルド] を選択するか、あるいは F7 を押して実行します。

前にこのチュートリアルで行った要領でインストールを実行することができます。インターネット インストールとしてインストールを実行するには、ツールバーの [リリース フォルダーを開く] ボタンをクリックして Setup.htm を起動します。

チュートリアルの次のステップでは、インストールのショートカットとレジストリデータの作成方法について説明します。

インストールのトラブルシューティング

インストールを実行してもファイルがインストールされなかった場合は、プロジェクトについて下記の点をチェックしてください。

- TARGETDIR が適切な値に設定されていることを確認します。これは、[一般情報] ビューで設定されます。

  次は、本チュートリアルの推奨値です。

  [ProgramFilesFolder]TutorialCo¥TutorialApp

- セットアップの種類に、関連付けられた機能が指定されていることを確認します。
- 機能にコンポーネントとファイルが関連付けられていることを確認します。
- インストールに対して何らかの変更を行った場合は、[ビルド] ボタンをクリックするか F7 キーを押して、プロジェクトの再ビルドが必要です。

ステップ 2: ショートカットとレジストリ データ

このステップでは、次の処理を行う場合の IDE 利用法を説明します。

- プログラムショートカットを作成する
- レジストリ情報の作成

ショートカットの作成

[ショートカット] ビューでは、ショートカットの作成と変更を行うことができます。ショートカットのプロパティには、表示名、ターゲット実行可能ファイルおよび引数、表示用アイコンなどがあります。プロジェクト アシスタントを利用すると、エンド ユーザーの [スタート] メニューにある Programs フォルダーに、Tutorial App へのショートカットが自動的に作成されます。

タスク ***このセットアップでは、エンド ユーザーのデスクトップ上にショートカットを作成します。***

1. [**ショートカット**] ビューに移動します。
2. **[デスクトップ]** アイコンを右クリックし、**[新規ショートカット]** を選択します。**[ショートカット ターゲットを参照する]** ダイアログ ボックスが開きます。
3. ダイアログ ボックスでは、**探す場所** ドロップダウンメニューから **[アプリケーション ターゲットフォルダー]** を選択し、ファイルリストから Tutorial.exe を選びます。
4. **[開く]** をクリックして **[ショートカットターゲットを参照する]** ダイアログ ボックスを閉じます。
5. ショートカットアイコンの内部名を、*tutorial* などに変更します。

次に、下記のショートカット用プロパティを設定します。

テーブル 2 ・ ショートカットのプロパティ

プロパティ	値	コメント
表示名	Tutorial App	名前を別の言語へ容易に翻訳できるように、InstallShield はプロジェクトの文字列に表示名を追加します (また、中括弧にかこまれた文字列識別子が、[表示名] フィールドに表示されます)。
ターゲット	<TARGETDIR>¥Tutorial.exe	
アイコン ファイル	<TARGETDIR>¥Tutorial.exe	
アイコン インデックス	0	
作業ディレクトリ	<TARGETDIR>	

ヒント: ユーザーのマシン上に既に存在するファイルへのショートカットを作成するには、可能なときにファイルへのパスを示すシステム変数を利用してパスを入力します。たとえば、ユーザーマシンの Windows または WinNT フォルダーに置かれた Windows Notepad のコピーを起動するには、ショートカットターゲットに <WINDIR>¥Notepad.exe と入力します。

レジストリ データを作成する

インストールに求められる機能の 1 つとして、ターゲット システムのレジストリへの情報の書き込みがあります。コンポーネントにレジストリを追加するには、[レジストリ] ビューを利用することができます。

たとえば、HKEY_LOCAL_MACHINE¥Software¥Tutorial Co¥Tutorial¥1.00.0000 の下に *TutorialData* というレジストリ値を作成するには、以下の手順を実行します。

タスク ***レジストリデータを作成するには、以下の手順を実行します。***

1. [**レジストリ**] ビューを開きます。
2. **[インストール先コンピューターのレジストリ]** ビュー 領域で **[インストール先コンピューター]** を右クリックして、**[新しいレジストリセット]** を選択します。
3. レジストリセットの名前を *tutorial* に変更します。
4. *tutorial* レジストリセットの下にある HKEY_LOCAL_MACHINE を右クリックして **[新規]** サブメニューから **[キー]** を選択します。
5. キーの名前を *Software* に変更します。
6. サブキー *Tutorial Co*、*Tutorial*、と *1.00.0000* に関しても同じ操作を繰り返します。
7. **[インストール先コンピューターのレジストリデータ]** 領域で、右クリックして、**[新しい文字列値]** を選択します。
8. 値を *TutorialData* に変更します。
9. TutorialData 値をダブルクリックして、**[値]** データ フィールドに *<TARGETDIR>* と入力します。
10. *tutorial* レジストリセットをクリックして、[**レジストリセットインストール条件**] パネルで [DefaultComponent] をチェックします。

インストール実行時に Tutorial.exe コンポーネントを含むセットアップの種類や機能のコレクションをユーザーが選択すると、ターゲット システム上にレジストリデータが作成されます。

ショートカットが作成されたことを確認する

タスク ***インストールがショートカットを作成したことを検証するには、以下の手順を実行します。***

1. **[ビルド]** ツールバーボタンをクリックするか、**F7** を押してプロジェクトを再ビルドします。
2. **[実行]** ボタンをクリックするか、CTRL+F5 を押してプロジェクトを実行します (最初にシステムからプログラムの既存バージョンをすべて削除します)。これで Tutorial App のショートカットが、**スタート** メニューの **Programs** フォルダーに作成されているはずです。

レジストリデータが作成されたことを確認する

タスク ***インストールがレジストリデータを作成したことを検証するには、以下の手順を実行します。***

1. ショートカットから Tutorial App を起動します。
2. **Tutorial** メニューから **Verify Registry Data** を選択します。レジストリデータが作成されていれば、〈*TARGETDIR*〉テキストを含むメッセージボックスが表示されます。

チュートリアルの次のステップでは、COM サーバー（自己登録ファイル）の登録方法について説明します。

ステップ 3: COM サーバーの登録

ほとんどのファイルに関して、インストールが果たすべき唯一の役割は、ソース メディアからターゲット システムにファイルをコピーすることです。これに対し一部のファイルについては、ターゲット システムへのファイル登録作業をインストーラーが実行する必要があります。別途処理が必要なファイルのカテゴリは *自己登録型ファイル* です。

本ステップでは、Tutorial.ocx 本体およびこれが使用する HTML ファイルのインストールと登録を行うコンポーネントを作成します。

COM サーバーは通常 DLL や .ocx の形を取りますが、こうした自己登録ファイルは、これらを使用するアプリケーションや Web ページが認識できるようあらかじめターゲット システムのレジストリへ書き込んでおくため、追加情報を必要とします。

タスク ***自己登録ファイルをインストールするには、以下の手順に従います:***

1. **[ファイルとフォルダー] ビュー**に移動します。
2. **[ファイルとフォルダー]** ビューの上部にある **[ビュー フィルター]** リストから **ProgramFiles** 機能を選択します。
3. **[ソース コンピューターのフォルダー] ペイン**で、ソース ディレクトリ 内の Tutorial.ocx を参照します。
4. **[ソースコンピューターのフォルダー]** ペインから **[インストール先コンピューターのフォルダー] ペインにある [アプリケーションのターゲットフォルダー]** に Tutorial.ocx をドラッグ アンド ドロップします。コンポーネント SelfRegFiles が［アプリケーションターゲット フォルダー］の下に作成されます。このコンポーネントには Tutorial.ocx が含まれ、その自己登録プロパティは **[はい]** に設定さています。これはコンポーネントを右クリックして **[プロパティ]** を開いて確認することができます。

次に、ProgramFiles 機能と関連付けられている新しいコンポーネントに HTML ファイルを追加します:

タスク ***ProgramFiles 機能と関連付けられている新しいコンポーネントに HTML ファイルを追加します:***

1. **[ファイルとフォルダー]** ビューの **[インストール先コンピューターのフォルダー] ペイン**で、**[アプリケーション ターゲットフォルダー]** を右クリックして **[新規コンポーネント]** を選択します。
2. コンポーネントの名前を *OcxHTML* に変更します。
3. TutorialCtrl.html を **[ソースコンピューターのファイル] ビュー**から *OcxHTML* コンポーネントにドラッグします。

タスク ***リリースを再ビルドし (F7 キーを押す) インストールを実行 (CTRL+F5 を押す) をした後、ファイルが正常に登録されたことを次の手順で確認できます。***

1. [**プログラム**] メニューのショートカットか、アイコンをダブルクリックして、Tutorial App を起動します。
2. **Tutorial** メニューから **COM Server Test** を選択します。
3. COM サーバーが正常に登録されていれば、「登録成功」のメッセージを示す HTML ページが表示されます。

チュートリアルの次のステップでは、ファイルの条件付きインストールについて説明します。

ステップ 4: 条件とプロパティ

このステップでは、ターゲット システムにデータを条件付きでインストールする方法について解説します。

インストール通常求められる機能は、特定の条件が満たされた場合にのみ必要なファイルをシステムにインストールすることです。たとえばある種のファイルは、特定のオペレーティング システムや言語にのみ必要であったり、ユーザー必要な権限を持つ場合に限りインストールできる、ということがあります。

コンポーネント (およびそのファイルとその他のデータ) を特定のオペレーティング システムにのみインストールさせるには、コンポーネントの "オペレーティング システム" プロパティを使います。コンポーネントのプロパティの変更は、[セットアップのデザイン] ビューを開いて該当する機能のアイコンを展開し、必要なコンポーネントを選択して行います。

タスク ***Windows 8 が搭載されたシステムだけにインストールされるコンポーネントを作成するには、次の手順を実行します。***

1. [**セットアップのデザイン**] ビューを開きます。
2. **HelpFiles** 機能を右クリックして、**[新規コンポーネント]** を選択します。
3. 作成されたコンポーネントの名前を *windows_8_files* に変更します。
4. コンポーネントの**ファイル**アイコンをクリックし、**ファイル**ペイン内で右クリックしてファイルを参照してソースフォルダーから ReadmeNT.txt ファイルを追加します。
5. windows_8_files コンポーネントをクリックして、コンポーネントのプロパティ グリッドを表示します。
6. コンポーネントの "オペレーティング システム" 設定を選択してから、[参照] ボタンをクリックします。**[プラットフォーム]** ダイアログが開きます。
7. [Windows 8] チェック ボックスを選択します。
8. **[OK]** をクリックします。

(F7 を押して) 再ビルドの後、(CTRL+F5 を押して) インストールを実行すると、ターゲット システムが Windows 8 を実行している場合に限りコンポーネントに含まれるファイルおよびその他のデータがインストールされるようになります。

チュートリアルの次のステップでは、プロジェクトのスクリプトの修正方法について説明します。

ステップ 5: スクリプトでの作業

InstallShield は、InstallScript 言語を使って、インストールを制御します。InstallShield ビューでプロジェクトのスクリプトを修正することができます。

ヒント: *CTRL+M を押して [スクリプト エディター] を最大化し、また CTRL+M をもう一度押して元に戻します。*

InstallScript プロジェクトのスクリプトはイベントドリブン型モデルと呼ばれるもので、定義済みの一連の関数を特定の順序で呼び出しながら処理を進めてゆきます。ビルトイン イベント ハンドラー、およびイベント ハンドラー関数の呼び出し順序については、「イベント ハンドラー」を参照してください。

スクリプト中の定義済み関数は [関数] ツリーに表示されます。既存の関数の表示および編集をするには、[関数] ツリーの該当する関数名をクリックします。

メモ: *イベント ハンドラー関数はスクリプト内にはっきりと記述されていなくても呼び出されます。イベント ハンドラー関数がスクリプト中に記述されていない場合は、そのデフォルトのコードが使用されます。*

イベント ハンドラー関数の追加および編集をするには、スクリプト エディターの左側にあるドロップダウン リストから適切なイベント カテゴリ (Before Move Data など) を選択してから、右側のドロップダウン リストから必要なイベント名 (Begin など) を選択します。スクリプト中で明示的に定義されたイベント ハンドラー関数は、太字で表示されます。

たとえば OnBegin イベント ハンドラーは、初回インストール時およびメンテナンス モード時の両方で、インストール スクリプトの最初に呼び出される関数です。

タスク ***この OnBegin イベント ハンドラーに独自のコードを追加するには、最初にこのための関数を作成します。***

1. 左側のイベントカテゴリリストから **Before Move Data** を選択します。
2. 右側のイベント ハンドラーリストから **Begin** を選択します。

これで、次のコードがスクリプトに追加されます。

```
////////////////////////////////////////////////////////////////////////////
//
//  関数: OnBegin
//
// イベント: Begin イベントはインストールの過程で常に最初のイベントとして
// 送られます。
//
////////////////////////////////////////////////////////////////////////////
function OnBegin()
begin
   // 作業: たとえば、デフォルトの非 UI 設定を変更します。
   //
   // またカスタムの初期化処理、要件のチェックなどを実行することもできます。
end;
```

この **OnBegin** 関数には、インストールの最初に実行させるコードを置きます。

関数ウィザードの使用

ここでは、MessageBox 関数の呼び出しを追加する関数ウィザードの利用法について説明します。これはユーザーがインストールを始めた時にメッセージを表示します。

タスク ***OnBegin 関数への MessageBox 関数の追加は、次の手順で行います。***

1. begin と end の間にあるコメント行 (// で始まる行) を削除し、テキストの挿入ポイントをこれらの行の間に置きます。
2. CTRL+I を押して、**関数ウィザード**を起動します。
3. **[関数カテゴリ]** リストから、**[ビルトイン ダイアログ]** を選択します。
4. **[関数名]** リストから、**MessageBox** を選択します。
5. **[次へ]** をクリックします。
6. 表示するメッセージを含む **szMsg** フィールドには、*"チュートリアル インストールへようこそ"* と入力します。(引用符が必要です)。
7. 表示するメッセージボックスのタイプを指定する **nType** ドロップダウンリストでは、**INFORMATION** を選択します。
8. [**完了**] をクリックすると、ここで指定した関数呼び出しがスクリプトに挿入されます。

この結果 **OnBegin** 関数は、次のように表示されます。

```
function OnBegin()
begin
MessageBox ("チュートリアル インストールにようこそ", INFORMATION );
end;
```

スクリプトのコンパイル

スクリプトを修正した場合、加えた変更を有効にするには、スクリプトをコンパイルする必要があります。スクリプトをコンパイルするには、[コンパイル] ツールバー ボタンをクリックし [ビルド] メニューから [コンパイル] を選択するか、CTRL+F7 を押します。コンパイル中のステータスメッセージは、IDE の出力ウインドウに表示されます。

ヒント: *スクリプトのコンパイル中にエラーや警告が発せられた場合は、エラーメッセージをダブルクリックすると、エラーの発生したスクリプト行がハイライト表示されます。*

([実行] ツールバー ボタンをクリックするか、CTRL+F5 を押して) インストールを実行すると、[ようこそ] ダイアログの前に、メッセージ ボックスが表示されます。

初回インストールの時だけメッセージを表示する

OnBegin 関数は、初回インストール時およびメンテナンスモードインストール時の両方で呼び出されるので、メッセージボックスはユーザーがプログラムを再インストール、又は削除する時にも表示されます。初回インストール時のみにメッセージボックスを表示するには、ユーザーがインストールを実行するのは初めてかどうかを判断する if ステートメント の中に MessageBox 呼び出しを配置します。

InstallScript は *MAINTENANCE* というシステム変数を定義しており、メンテナンス モードでインストールが実行している場合は True、そうでない場合は False を取ります。メッセージボックスを初回インストール時のみに表示するには、**OnBegin** 関数を次のよう変更してください。

```
function OnBegin()
begin
if (!MAINTENANCE) then
    MessageBox("チュートリアル インストールにようこそ!", INFORMATION);
endif;
end;
```

このインストールをコンパイルして実行すると、メッセージボックスは初回インストール時にのみ表示され、メンテナンス モードインストール時には現れなくなります。

InstallScript

InstallScript 言語には、レジストリおよび INI ファイルの処理、ターゲット オペレーティング システムの特性テスト、またはダイアログの表示など、インストール関連タスクを実行する数百以上の関数が用意されています。これらの詳細と使用例については、「InstallScript 言語リファレンス」を参照してください。

チュートリアルの次のステップでは、インストールが表示するダイアログを変更する方法について説明します。

ステップ 6: ユーザー インターフェイスの変更

このステップでは、インストールのユーザー インターフェイスを変更する 3 種類の方法について説明します。

1. ユーザーの入力に応じて、異なるアクションを行う。
2. ユーザー インターフェイス関連のスクリプト関数を修正して、表示するダイアログを変更する。
3. **ダイアログ エディター**を使用して、ダイアログのレイアウトとプロパティを変更する。

ユーザー入力の処理

InstallScript コード中で、実行時に表示されるダイアログ関数を含むのは次のイベント ハンドラー関数です:

- **OnFirstUIBefore**、初回インストール時のデータ転送を行う前に表示されるダイアログを含む。
- **OnFirstUIAfter**、初回インストール時のデータ転送を行った後に表示されるダイアログを含む。
- **OnMaintUIBefore**、メンテナンスモードインストール時のデータ転送を行う前に表示されるダイアログを含む。
- **OnMaintUIAfter**、メンテナンス モード インストール時のデータ転送を行った後に表示されるダイアログを含む。

メモ: *プロジェクトの "メンテナンスの有効化" プロパティが [アンインストールまたはメンテナンスをしない] に設定されている場合は、OnMaintUIBefore および OnMaintUIAfter イベント ハンドラー関数は呼び出されません。*

プロジェクト アシスタントを使って作成されたデフォルト InstallScript プロジェクトは、初回インストールのユーザー インターフェイスを定義する **OnFirstUIBefore** イベント ハンドラー関数を定義します。**OnFirstUIBefore** は、ダイアログ関数を呼び出して、プロジェクト アシスタントの [インストール インタビュー] ページで指定したダイアログを表示します。たとえば、次のコードはエンド ユーザーに対してユーザー名と会社名の入力を要求するダイアログを表示します:

```
szMsg = "";
szTitle = "";
nResult = SdRegisterUser( szTitle, szMsg, szName, szCompany );
```

エンド ユーザーのユーザー名と会社名は最後の 2 つの変数で戻されます。これらを自由に利用してレジストリ キーの作成や、ファイルに格納した情報との照合が可能です。

表示ダイアログを変更する

InstallScript 言語には、インストールのユーザー インターフェイス用に、あらかじめ定義された多数のダイアログが用意されています。

タスク ***デフォルトのユーザー情報ダイアログを、ユーザーにシリアル番号の入力も要求するダイアログに置き換えるには、SdRegisterUser 関数を SdRegisterUserEx の呼び出しと置き換えます:***

1. **InstallScript** ビューの **[関数]** ノードで **OnFirstUIBefore** を選択します。
2. 文字列スクリプト変数 szSerial を宣言するには、**OnFirstUIBefore()** 関数の begin の前に次のスクリプトを追加します。

   ```
   string szSerial;
   ```

3. SdCustomerInformation の呼び出しを含む行を削除します:

   ```
   Dlg_SdRegisterUser:
       szMsg = "";
       szTitle = "";
       nResult = SdRegisterUser( szTitle, szMsg, szName, szCompany );
       if (nResult = BACK) goto Dlg_SdLicense2;
   ```

4. 削除した行の代わりに、次の行を入力します:

   ```
   Dlg_SdRegisterUserEx:
       szMsg = "";
       szTitle = "";
       szSerial = "";
       nResult = SdRegisterUserEx(szTitle, szMsg, szName, szCompany, szSerial );
       if (nResult = BACK) goto Dlg_SdLicense2;
   ```

5. Dlg_SdRegisterUser にポイントしていた goto ステートメントを Dlg_SdRegisterUserEx へポイントするように変更します。
6. CTRL+F7 を押して、スクリプトをコンパイルします。

同様の手順で、インストールのユーザー インターフェイスに新しいダイアログを追加したり、既存のダイアログを別のものと置き換えることができます。

ダイアログ エディターを使用する

ダイアログ エディターを使って、インストールで表示されるダイアログの外観を修正することができます。

以前のステップで説明したように、エンドユーザーが入力するシリアル番号は SdRegisterUserEx ダイアログ ボックス上にプレーン テキストで表示されます。

タスク ***ダイアログを修正して、エンド ユーザーが入力したパスワードが隠れるように設定するには、次の手順に従います。***

1. **[ダイアログ]** ビュー（[リスト] ビューの [ユーザー インターフェイス] ノードの下）で、**SdRegisterUserEx** アイコンを右クリックして **[編集]** を選択します。
2. **英語 (U.S.)** アイコンを選択します。
3. **ダイアログ エディター**で、**シリアル番号** ラベルの編集フィールドを選択します。
4. **編集**コントロールの**″ パスワード″** プロパティを False から True に変更します。

ヒント: *CTRL+M を押すと、[ダイアログ エディター] ビューが最大化します。*

プロジェクトを再ビルド（F7 を押す）してから実行する（CTRL+F5 を押す）と、ユーザーが入力するシリアル番号は隠されて表示されます。

ヒント: *ダイアログをデフォルトの状態に戻すには、ダイアログのアイコンを右クリックして、[ダイアログをデフォルトに戻す] を選択します。*

基本の MSI プロジェクト チュートリアル

このチュートリアルでは、InstallShield を使用した基本の MSI インストール プロジェクトの作成、ビルド、実行、およびその他各種の機能について必要な操作方法を順を追って説明します。

チュートリアルの構成は複数のステップに分かれています。最初のステップ（ステップ 1: プロジェクトの作成、ビルドおよびテスト）を除き、残りのステップはすべて独立した内容になっているので、これらは特に順番にこだわることなく現在の作業に必要な情報のみを調べることができます。

本チュートリアルでは、インストールが取り扱うタスクの多くの処理方法について解説します。

- ファイルのインストール
- ショートカットおよびレジストリ データの設定
- データの条件付きインストール
- COM サーバーの登録
- インストールのユーザー インターフェイスの変更
- リリースのビルド
- インストールのテスト

チュートリアルを通して、InstallShield ヘルプ ライブラリの該当箇所へのリンクが設けてあります。

ステップ 1: プロジェクトの作成、ビルドおよびテスト

このステップでは、インストール プロジェクトの作成、リリース イメージのビルド、およびインストールのテストについて説明します。本ステップを一通り完了すると、以下の項目がマスターできます。

- プロジェクト アシスタントを使った新規プロジェクトの作成法。
- インストール プロジェクト内の多くの設定を構成。
- 機能、コンポーネント、およびファイル リンクの指定方法。
- 複製および配布用リリースのビルド方法。
- インストールの実行。

基本の MSI インストール構成ブロック

基本の MSI インストールは 2 段階で構成されます:

テーブル 1・基本の MSI インストール内のレベル

レベル	説明
コンポーネント	コンポーネントとは、開発者から見て、個別にインストール可能な最小の製品構成単位のことです。コンポーネントには、ファイル、ショートカット、レジストリ データなど、ターゲット システムにインストールされるデータが含まれます。ユーザーがコンポーネントを直接処理することは決してありません。 1 つのコンポーネントを複数の機能に配置することもでき、ユーザーがこのコンポーネントと関連付けられた機能の 1 つを選ぶと、このコンポーネントのデータがインストールされます。
機能	機能とは、エンドユーザーから見て、個別にインストール可能な最小の製品構成単位のことです。ユーザーがインストール中に [カスタム] セットアップ タイプを選択すると、どの機能をインストールするかをユーザーが選択することができるダイアログが表示されます。 各機能はコンポーネントを含みます。

チュートリアルファイル

本チュートリアルで作成するインストールは、*Tutorial App* という名前のアプリケーションについて、そのインストールと設定を行うものです。Tutorial App のソース ファイルは InstallShield Program Files フォルダーの Samples サブフォルダーにあります。デフォルト インストール先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Tutorial Project

新規基本の MSI プロジェクトの作成

チュートリアルの最初のステップは、新しい基本の MSI プロジェクトの作成です。

タスク

新規で基本の MSI プロジェクトを作成するには、以下の手順を実行します。

1. **[ファイル]** メニューから **[新規作成]** を選択するか、ツールバーの **[新規プロジェクト]** ボタンをクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **Windows Installer** タブをクリックして、**基本の MSI** プロジェクトを選択します。
3. **"プロジェクト名"** フィールドに **Tutorial** と入力します。
4. **[場所]** ボックスでは、デフォルト値をそのまま残します。
5. **[[プロジェクト名] サブフォルダーにプロジェクト ファイルを作成]** を選択します。
6. **[OK]** をクリックします。

InstallShield がプロジェクトを作成して開きます。

InstallShield は、*ProjectName*.ism という名前のプロジェクト ファイルを作成します。この場合、Tutorial.ism となります。プロジェクト ファイルには、InstallShield で構成されたすべての設定が格納されます。プロジェクトを他のマシンに移動する場合は、移動先のシステムに .ism ファイル (およびインストールソースファイル) をコピーします。

ヒント: *新しいプロジェクト ファイルが作成されるデフォルト ディレクトリを変更するには、[オプション] ダイアログ ボックスを開きます ([ツール] メニューから [オプション] をクリック)。[ファイルの場所] タブにある "場所" 設定に新しいパスを入力します。*

アプリケーション情報を入力する

新規プロジェクトを作成すると、プロジェクト アシスタントが起動し、プロジェクトやアプリケーション情報の指定手順を案内します。プロジェクト アシスタントの最初のページでは、インストール作成プロセスの概要が図式化されます。プロジェクト アシスタントを利用するには、ビューの下にある [アプリケーション情報] アイコンをクリックします。

[アプリケーション情報] ページでは、プロジェクトがインストールするアプリケーションについての一般情報を指定します。

タスク ***チュートリアル用のアプリケーション情報を指定するには、以下の手順を実行します。***

1. **"会社名を指定"** 設定に **TutorialCo** を入力します。これにより、**"会社の Web アドレスを指定"** 設定の情報が自動的に更新されます。
2. **"アプリケーション名を指定"** 設定に **TutorialApp** を入力します。入力された値は、エンド ユーザーに表示されるダイアログに使用されます。また、[プログラムの追加と削除] に表示されるアプリケーションの名前として使用されます。
3. **"アプリケーション バージョンの指定"** および **"会社 Web アドレスの指定"** 設定は、デフォルト値のままにします。
4. **[プログラムの追加と削除] でアプリケーションと共に表示するアイコンを選択します]** の下にある **[参照]** ボタン をクリックして Tutorial.exe の場所を参照します。デフォルトの場所は、C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Tutorial Project です。.exe ファイルを開いて **アイコン インデックス 0** を選択します。

完了すると、[アプリケーション情報] パネルは次のように表示されます。

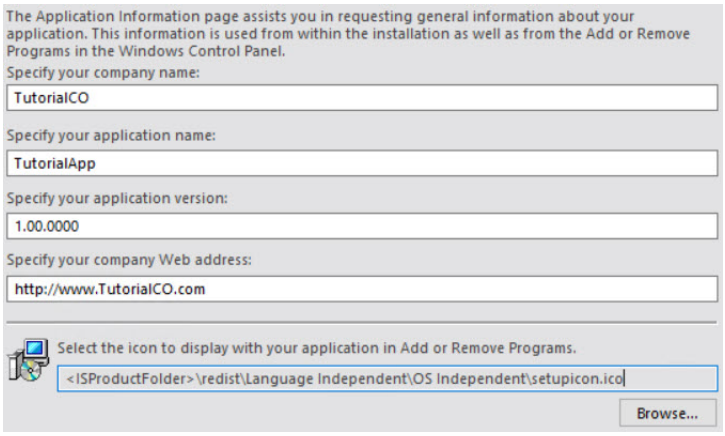


図 1: [アプリケーション情報] パネル

入力するアプリケーション名と会社名は *Windows スタート* メニュー上のアプリケーションのショートカットのデフォルトの場所、並びにプログラム ファイルのデフォルトのインストール先を指定する *INSTALLDIR* プロパティのデフォルト値を決定します。

メモ: *[INSTALLDIR] のデフォルト値は [ProgramFilesFolder] 会社名¥製品名 です。このうち [ProgramFilesFolder] の部分は、実行時にユーザーの Program Files フォルダーの位置に置き換えられます。Windows Installer が定義する他のディレクトリ関連のプロパティについては、Windows Installer プロパティ リファレンスの「インストーラーが設定するシステム フォルダー」セクションを参照してください。*

インストール要件の設定

[インストール要件] ページでは、ターゲット システムのインストールの要件を簡単に設定することができます。たとえば、アプリケーションを適切に実行するために特定の OS が必要な場合、このパネルの最初の部分で示すことができます。

オペレーティング システム

アプリケーションがターゲット システムに 特定の Windows バージョン以降を必要とする場合、[はい] を選択してからアプリケーションに必要な OS を選択します。

チュートリアルでは *[いいえ]* のままにします

ソフトウェア要件

アプリケーションのインストールに、ターゲット システムへ特定のソフトウェアがインストールされていることが必要な場合、*[はい]* を選択して該当するソフトウェアを選択します。必要なソフトウェアがターゲット システム上で見つからない場合に表示されるランタイム メッセージをカスタマイズするには、そのメッセージをクリックして編集します。

メモ: *このセクションのランタイムメッセージはソフトウェア要件が選択されるまでは表示されません。*

チュートリアルでは *[いいえ]* のままにします。

インストール アーキテクチャのカスタマイズ

[インストール アーキテクチャ] ページで、インストールがエンド ユーザーに対して表示する機能を指定することができます。機能とは、ユーザーから見て個別にインストール可能な最小の製品構成単位のことです。インストール中に [カスタム] セットアップ タイプを選択すると、個々の機能がエンドユーザーに対して表示されます。

メモ: *機能には、サブ機能やサブサブ機能などを含めることも可能で、インストールの要求に応じて必要なレベルの多層構造化が可能です。*

インストールアーキテクチャは現在デフォルト機能、*Tutorial_Files* を含みます。デフォルトの機能は、エンドユーザーがインストールを実行するときに常にインストールされます。このステップでは、インストールアーキテクチャへ別の機能を追加します。

タスク ***チュートリアルでは、新しい Help_Files 機能を追加します。***

1. **インストール アーキテクチャをカスタマイズしますか？**には **[はい]** を選択します。
2. **[インストール アーキテクチャ]** ノードを右クリックして、**[新規作成]** を選択します。
3. 新規機能に *Help_Files* と名前をつけます。

このステップを完了すると、インストールアーキテクチャはこの様になります。

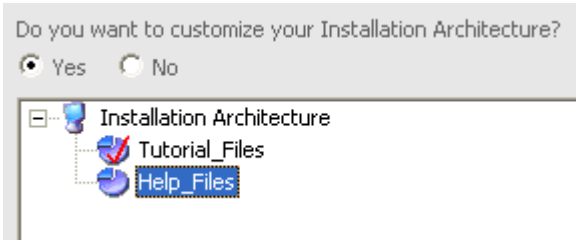


図 2: インストールアーキテクチャ

プロジェクトへファイルを追加する

次のステップでは、アプリケーションのファイルをインストール プロジェクトへ追加します。[アプリケーションファイル] ページでは、機能に関連付けるファイルを指定することができます。

このステップでは、Tutorial 実行可能ファイルを Tutorial_Files 機能へ追加します。

タスク ***Tutorial 実行可能ファイルを Tutorial_Files 機能へ追加するには、以下の手順を実行します。***

1. ページ上部の機能ドロップダウンリストから、Tutorial_Files 機能を選択します。
2. ツリー コントロール（トップノードは [インストール先コンピューター]）内にある **INSTALLDIR** ノードを選択します。
3. **[ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. チュートリアル ファイル ソース ディレクトリに置かれた Tutorial.exe を参照します。

5. **[開く]** をクリックして、Tutorial_Files 機能にファイルを追加します。
6. **「追加したファイル...は依存関係を持つ可能性があります」**メッセージが表示されたとき、**[いいえ]** をクリックします。Tutorial.exe には依存関係は含まれません。ファイルが機能へ追加され、右側のファイルリストパネル内に表示されます。

メモ: *ファイルの隣にあるキーアイコンは、ファイルが機能のコンポーネントのキーファイルであることを示します。Windows Installer ではほとんどの場合、コンポーネントにキーファイルが 1 つ含まれている必要があります。Windows Installer サービスは各種の目的でコンポーネントのキーファイルを使用します。たとえばキーファイルの存在を調べてコンポーネントを修復する必要があるかどうかを判断する場合や、ショートカットのデフォルトのターゲットとしてキーファイルを使うなどの用途があります。実行可能ファイルを基本の MSI プロジェクトの機能に追加するとき、プロジェクト アシスタントは自動的にそれをファイルの為に内部で作成されるコンポーネントのキーファイルに設定します。詳細については、「コンポーネントのキーファイルを設定する」を参照してください。*

ショートカットの作成

[アプリケーション ショートカット] ページでは、ターゲット システムのデスクトップ又は [スタート] メニュー上にあるアプリケーションファイルのショートカットを指定することができます。デフォルトで、このページはインストール プロジェクトに入れた各実行可能ファイルのショートカットを表示します。これらは削除することができ、インストール プロジェクトに入れたショートカットを他のファイルに追加することもできます。

タスク ***Tutorial.exe へのショートカットをアクティブにするには、以下の手順を実行します。***

[起動 Tutorial.exe] アイコンをクリックします。**[スタート] メニューにショートカットを作成する**を選択したデフォルト設定をそのままにします。InstallShield は、インストールが実行されたときに、ユーザーの **[スタート]** メニュー上に Tutorial.exe へのショートカットを作成します。

レジストリ データを構成する

[アプリケーション レジストリ] ページでは、アプリケーションに必要なレジストリ エントリを指定することができます。

チュートリアルでは、このページでレジストリ エントリの指定は行いません。レジストリ エントリについては、チュートリアルのステップ 2: ショートカットとレジストリ データ" で説明します。

インストール インタビューを使ってダイアログを選択する

[インストール インタビュー] ページでは、インストールの実行時にエンドユーザーに対して表示するダイアログボックスを選択します。このページでの回答によって、プロジェクト アシスタントは対応するダイアログをインストール プロジェクトに追加します。

タスク ***チュートリアル用のダイアログを指定するには、以下の手順を実行します。***

1. **使用許諾契約ダイアログを表示しますか?** – **[いいえ]** を選択します。このオプションに **[はい]** を選択した場合は、使用許諾契約書ファイルを参照します。
2. **[会社名] と [ユーザー名] を入力するプロンプトを表示しますか ?** – **[はい]** を選択します。インストールは、この情報を求めるダイアログを表示します。
3. **ユーザーがアプリケーションのインストール場所を変更することができるようにしますか?** – **[はい]** を選択します。詳細については、「エンドユーザーによるインストール先の変更を許可する」を参照してください。
4. **ユーザーがアプリケーションの一部のみを選択してインストールできるようにしますか?** – **[はい]** を選択します。詳細については、「インストールが選択できるインストールを作成する」を参照してください。
5. **インストールの完了時にアプリケーションを起動するオプションをユーザーに提供しますか?** – **[はい]** を選択して、Tutorial.exe ファイル ([ProgramFilesFolder]TutorialCo¥Tutorial にある) を参照します。このオプションが **[はい]** に設定されると、最後のダイアログにはチェック ボックスがあり、エンドユーザーが **[完了]** ボタンを押すとアプリケーションが直ちに起動します。

インストール用の言語を選択する

[インストールのローカライズ] ページで、インストールが対応する言語を指定することができます。文字列値と関連識別子を指定して、エンド ユーザーのインターフェイスで簡単に他の言語にローカライズすることができます。

エディション: *InstallShield をインストールした際に選択した言語に追加して他の言語を選択するには、InstallShield Premier Edition が必要です。*

タスク ***このチュートリアルでは英語(U.S.) を選択した状態にし、次の手順に従ってインストールの機能の表示名を変更します。***

1. リストボックスで、**"機能文字列データ"** を選択します。右の表は、すべての機能文字列エントリを表示します。
2. **[値]** 列で **Tutorial_Files** (識別子 IDS_FEATURE_DISPLAY_NAME2 と関連付けられた値) をクリックして、その値を **「チュートリアル ファイル」** に変更します。
3. **Help_Files** (識別子 IDS_FEATURE_DISPLAY_NAME3 と関連付けられた値) をクリックして、その値を**「ヘルプ ファイル」**に変更します。

メモ: *詳細については、「複数言語インストールの作成」を参照してください。*

インストールをビルドする

インストール プロジェクトのアーキテクチャを定義し、アプリケーションファイルを追加し、ショートカットを作成し、ダイアログを選択したあと、インストールのビルドの準備が整います。

[インストールのビルド] ページでは、ビルドするタイプを指定し、再配布可能ファイルをコピーする場所も指定することができます。

タスク ***作成したばかりのインストールをビルドするには、以下の手順を実行します。***

1. CD-ROM オプションを選択します。
2. **[インストールのビルド]** をクリックします。

[出力] ウィンドウが開き、ビルドの進行状況についての情報が表示されます。

メモ *ビルドが警告 -7235 を生成しました。これは予期する警告です。この警告を解決せずに次のステップに進むか、この警告を解決する場合は、必要に応じて [一般情報] ビューの [ソフトウェア識別タグ] 領域で設定を構成できます。"ソフトウェア識別タグの使用" 設定で [はい] が選択されている時に、4 つの必須識別設定 ([一般情報] ビューの "一意な ID"、"一意な登録 ID"、"タグ作成者"、および "タグ作成者 ID" 設定) のうち最低 1 つの値が設定されていなかった場合、空白の設定ごとに、ビルド警告 -7235 が一回発生します。詳細については、「製品のソフトウェア識別タグを含める」を参照してください。*

次のステップでは、インストールを実行します。

インストールを実行する

このチュートリアルでプロジェクト アシスタントのステップを完了すると、Tutorial 実行可能ファイルをインストールする完全なインストール が作成されます。

タスク ***インストールを実行するには、以下の手順を実行します。***

ツールバー上の **[実行]** ボタンをクリックするか、または、CTRL+F5 を押します。

インストールは、プロジェクト アシスタントの [インストール インタビュー] ページで選択したダイアログを表示します。プロジェクト アシスタントで入力した値は、適切なダイアログでエンドユーザーに対して表示されます。たとえば、プロジェクト アシスタントで指定した *INSTALLDIR* のデフォルト値は、インストーラーの実行時に [インストール先の選択] ダイアログ ボックス上に現れます。このときエンドユーザーが異なるインストール先ディレクトリを選ぶと、*INSTALLDIR* にはこの新しい値が格納されます。

インストールが完了した時点でインストール先のディレクトリを開き、インストールされたファイルを確認してください。インストールが成功していれば、チュートリアルファイルがインストールされているはずです。

メンテナンス モード

システム上にインストール済みのアプリケーションについて、ユーザーが 2 回目 (およびそれ以降) にインストールを実行すると、インストールは*メンテナンス モード*で実行されます。このメンテナンスモードでユーザーは、最初のインストールで選択した機能の変更、インストール済みの機能の修復、およびアプリケーション全体を削除することが可能です。

アプリケーションのアンインストール

タスク ***チュートリアルアプリケーションをアンインストールするには、以下の手順を実行します。***

[アンインストール] をクリックします。

基本のインストール プロジェクトを作成したあと、[インストール デザイナー] タブ (InstallShield ウィンドウの上部) をクリックして、チュートリアルの 次のステップ で説明されているようにインストールを拡張、または微調整します。

IDE で作業する

プロジェクトの作成後は、*IDE* とも呼ばれる InstallShield インストール開発環境でプロジェクトのプロパティを設定します。IDE は機能カテゴリー別に並べられているので、プロジェクトに情報を簡単に追加または編集できます。ここから先のチュートリアルでは IDE ビューのいくつかを紹介します。

メモ: *IDE で表示されるビューは、作成するプロジェクトの種類によって異なります。*

本ステップを一通り完了すると、以下の項目がマスターできます。

- プログラムの機能のプロパティの設定法。
- コンポーネントおよびファイル リンクの作成法。

機能のプロパティを設定する

まず最初に行うのは、プロジェクト アシスタントで作成した機能に対する追加プロパティの設定で、これには機能の表示名や説明などがあります。機能プロパティを編集するには、IDE の [機能] ビューへ移動します。

タスク ***このチュートリアルの機能のプロパティを設定するには、以下の手順を実行します。***

1. **[機能]** ビューを開きます。**[機能]** ビューは、**[ビューリスト]** の **[編成]** セクションにあります 。
2. [**機能**] ビューで、Tutorial_Files 機能を選択して機能のプロパティグリッドを右側に表示します。
3. **"説明"** フィールドに次のテキストを入力します:
4. **New_Feature** 機能を選択してそのプロパティ グリッドを表示します。
5. **"説明"** フィールドに次のテキストを入力します:

表示名と説明を入力すると、IDE はそれぞれの値を表すための文字列テーブルエントリ (*{ID_STRINGn}* と表示) を作成します。

実行時、エンド ユーザーが [カスタム] セットアップ タイプを選択した場合、インストールはユーザーがどの機能をインストールするかを選択できるダイアログを表示します。このダイアログには、ここで設定した機能の表示名および説明が表示されます。

コンポーネントおよびファイルリンクを作成する

[ファイルとフォルダー] ビューの追加ファイルにリンクを作成することができます。このステップでは、Help_Files 機能にファイルを 1 つ追加します。[ファイルとフォルダー] ビューでファイルを追加すると、セットアップベストプラクティス規則に従ってIDE がコンポーネントを作成します。

タスク ***Tutorial.html を Help_Files 機能にある新規コンポーネントへ追加するには、以下の手順を実行します。***

1. **[ファイルとフォルダー]** ビューを開きます。
2. **[インストール先コンピューターのフォルダー] ペイン**で、**[インストール先コンピューター]** アイコンを右クリックし、**[コンポーネントの表示]** が選ばれていることを確認します。
3. ビューの最上部にある機能のリストから、**Help_Files** を選択します。
4. [**インストール先コンピューターのフォルダー**] ペインにあるツリーを広げて [INSTALLDIR] フォルダーを探します。
5. [INSTALLDIR] フォルダーを右クリックして **[新規コンポーネント]** を選択します。コンポーネントに *Help_Component* と名前をつけます。
6. [**ソース コンピューターのフォルダー**] ペインで、TutorialHelp.html が置かれたチュートリアル ファイル ソース フォルダーを選びます。
7. **TutorialHelp.html** アイコンを **[ソース コンピューターのファイル] ペイン**から **インストール先コンピューターのフォルダー**にある **Help_Component** コンポーネントへドラッグします。InstallShield は、このファイルを Help_Files 機能の Help_Component コンポーネントへ追加します。
8. **Help_Component** アイコンをクリックして、**[インストール先コンピューターのファイル] ペイン**にコンポーネントのファイルを表示します。
9. 各コンポーネントにはキーファイルを含まなくてはならないので、TutorialHelp.html ファイルを右クリックして **[キーファイルの設定]** を選択します。

コンポーネントにリンクされるファイルのリストに変更が無い、この種のファイルリンクを「*スタティック ファイル リンク*」と呼びます。ビルド毎に内容に違いがあるディレクトリ(及びそのサブディレクトリ)にリンクするには、「ダイナミック ファイル リンク」を参照してください。

チュートリアルの次のステップでは、インストール プロジェクトのリリースイメージのビルド方法について説明します。

リリースのビルド

インストールをテストする前に、リリースをビルドしてプロジェクト設定の更新を行なう必要があります。リリースには、CD-ROM やネットワーク経由でエンド ユーザーに配布するすべてのファイルが含まれます。

リリース ウィザードを使って設定を構成すると、リリースを最も簡単に作成することができます。リリース ウィザードでは、使用するメディアの種類 (たとえば CD-ROM) や、メディア上のファイルを圧縮するかなど、リリースのプロパティを指定します。

タスク ***リリース ウィザードを開始するには、以下の手順を実行します。***

1. ツールバーにある **[リリース ウィザード]** ボタンをクリックするか、**[ビルド]** メニューから **[リリース ウィザード]** を選選択します。
2. **[ようこそ]** パネルで [次へ] をクリックしてリリース設定の定義を開始します。

製品構成とリリースに名前を付ける

[製品構成] パネル

[製品構成] パネルでは、現在の製品構成に対する名前を指定します。製品形状の名前とは、ビルドしたリリースを置く (プロジェクトフォルダー中の) フォルダーの名前です。

タスク ***チュートリアルでは、***

1. *Tutorial* と名づけた新規プロダクト構成を作成します。
2. **[次へ]** をクリックして、リリース名の指定の説明に進んでください。

[リリースの指定] パネル

[**リリースの指定**] パネルでは、リリース名を指定します。リリース名は、ビルドされたリリースを置く (製品構成フォルダー中の) フォルダーの名前として使用されます。

タスク ***チュートリアルでは、***

1. *CDROM* と名づけた新しい製品リリースを作成します。
2. **[次へ]** をクリックします。

フィルター設定と言語の指定

[フィルターの設定] パネル

[フィルターの設定] パネルでは、現在のリリースから除外しておく機能やコンポーネントを指定します。

タスク ***チュートリアルでは、***

デフォルト設定 (フィルタリング無し) のままとし、**[次へ]** をクリックしてください。

セットアップ言語

[セットアップ言語] パネルでは、インストールのユーザー インターフェイスで表示する言語を (プロジェクト言語の中から) 指定し、またユーザー用インストール言語の選択ダイアログ表示の有無を指定します。

タスク ***チュートリアルでは、***

デフォルト設定 (ユーザー インターフェイスに日本語を含める) のままとし、**[次へ]**をクリックしてください。

メディアの種類とディスク分割のオプションを選択する

[メディアの種類] パネル

[メディア タイプ] パネルでは、リリースをビルドするメディアの種類を指定します。ここで指定するメディアの種類は、[リリース ウィザード] が作成するディスクイメージフォルダーのサイズをデフォルトします。たとえば CD-ROM 用にリリースをビルドすると、リリース ウィザードがディスクイメージを複数のフォルダーに分割します。それぞれのフォルダーサイズは 650MB 以下です。

タスク ***チュートリアルでは、***

[メディアタイプ] メニューから **CD-ROM** を選択します。

[ディスク分割オプション] パネル

[ディスクス分割オプション] パネルでは、複数のディスクイメージが必要な場合に、プログラム ファイルをどのように配置するかを指定します。[カスタム] 分割タイプでは、特定の機能のファイルをどのディスク イメージ上に置くかを指定できます。

タスク ***チュートリアルでは、***

[自動] を選択し、各機能のファイルを置くディスクイメージを**リリース ウィザード**が決定します。

詳細については、「インストールを複数のディスクまたは CD に分割する」を参照してください。

圧縮の設定とセットアップランチャーのオプションを指定する

[リリース構成] パネル

[リリース構成] パネルでは、インストール プロジェクト内のファイルの圧縮について、すべて圧縮する、全く圧縮しない、一部のファイルのみを圧縮する、のどれかを指定することができます。

タスク ***チュートリアルでは、***

[すべてのファイルを圧縮] を選択します。

[セットアップランチャ] パネル

[セットアップランチャー] パネルでは、Setup.exe セットアップ起動ファイルを作成するか、インストールに Windows Installer 3.1 エンジンを含めるかどうかを指定します。Windows Installer エンジンのインストールは、古いバージョンの Windows Installer が搭載されたターゲット システムで必要です。

タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします。

このステップでは、すべてデフォルトの設定にしてください。

デジタル署名とパスワード保護を追加する

[デジタル署名] パネル

[デジタル署名] パネルで、アプリケーションにデジタル署名を付加することができます。アプリケーションにデジタル署名をすることで、アプリケーション内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証することができます。

タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします(デジタル署名を行わない)。

[パスワードと著作権] パネル

[パスワードと著作権] パネルでは、インストール プロジェクトのパスワード保護をアクティブにして、アプリケーションの著作権の特定の情報を指定することができます。

タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします(パスワード保護または著作権情報を含みません)。

.NET Framework のサポートを含める/詳細設定の選択

.NET Framework パネル

[.NET Framework] パネルで、リリースに .NET Framework サポートを含むかどうかを指定します。

タスク ***チュートリアルでは、***

デフォルトの設定をそのままにします(.NET Framework を含みません)。

[詳細設定] パネル

[詳細設定] パネルでは、圧縮のレベルや、SMS 配布用 PDF ファイルの作成など、現在のリリースに対する追加設定を行います。

タスク ***このチュートリアルでは、次のチェックボックスを選択します。***

- 長いファイル名を使用する
- サイズを最適化する
- Autorun.inf を作成する — これは CD-ROM イメージ用の AutoPlay を可能にするファイルを生成します。

その他の設定については、[詳細設定] パネルの [ヘルプ] ボタンをクリックしてください。

設定を確認する

概要

[概要] パネルでは、現在のリリースに関するリリース ウィザードの設定をすべて表示します。

タスク ***設定が正しい場合***

1. **[リリースをビルドする]** チェックボックスを選択します。
2. **[完了]** をクリックしてリリースをビルドします。

進行中のビルドに関するステータスメッセージは、アウトプットウインドウに表示されます。ビルドが完了すると、CD にコピーするファイルは、下記のディレクトリに置かれます。

<ProjectFolder>¥Tutorial¥cdrom¥DiskImages¥DISK1.

[リリース] ビューの [イベント] タブを使って、InstallShield で、ビルドされたディスク イメージを別のディレクトリにコピーすることができます。

プロジェクトの再ビルド

チュートリアル後半のステップで行うプロジェクトに対する変更が終了したら、最新リリースの再ビルドを行います。この処理は、[ビルド] ツールバーボタン をクリックするか、[ビルド] メニューから [ビルド] を選択するか、または F7 を押して実行します。

チュートリアルの次のステップでは、インストールのショートカットとレジストリデータの作成方法について説明します。

インストールのトラブルシューティング

インストールを実行してもファイルがインストールされなかった場合は、プロジェクトについて下記の点をチェックしてください。

- INSTALLDIR が適切な値に設定されていることを確認します。これは、[一般情報] ビューで設定されます。

  本チュートリアルの推奨値は、[ProgramFilesFolder]TutorialCo¥TutorialApp です。
- 機能にコンポーネントとファイルが関連付けられていることを確認します。
- インストールに対して何らかの変更を行った場合は、[ビルド] ボタンをクリックするか F7 キーを押して、プロジェクトの再ビルドが必要です。

ステップ 2: ショートカットとレジストリ データ

このステップでは、次の処理を行う場合の IDE 利用法を説明します。

- プログラムショートカットを作成する
- レジストリ情報の作成

その他のシステムデータの作成も、本ステップでの手順と同様に進めることができます。詳細については、下記にリストアップしたトピックを参照してください。

ショートカットの作成

[ショートカット] ビューでは、ショートカットの作成と変更を行うことができます。ショートカットのプロパティには、表示名、ターゲット実行可能ファイルおよび引数、表示用アイコンなどがあります。

ショートカットの作成

タスク

このステップでは、ユーザーのスタートメニューにあるプログラムフォルダーに Tutorial App のショートカットを作成します。

1. **[ショートカット]** ビューを開きます。**[ショートカット]** ビューは、**[ビュー リスト]** の **[システム構成]** セクションにあります 。
2. **[プログラム メニュー]** フォルダー アイコンを右クリックし、**[アドバタイズ ショートカットの新規作成]** を選択します。[**コンポーネントの参照**] ダイアログが表示されます。
3. ダイアログで、**[機能]** ドロップダウンメニューから **Tutorial_Files** を選択し、ファイルリストから Tutorial.exe を選択して **[開く]** をクリックしてダイアログを閉じます。
4. ショートカットアイコンの内部名を、*Tutorial* などに変更します。
5. 次のショートカットプロパティを設定します。

テーブル 2 ・ ショートカットのプロパティ

プロパティ	値	コメント
表示名	チュートリアルアプリケーション	長いファイル名をサポートしないターゲット システムに対応するため、IDE は “TUTORI~1|TutorialApp” のような短いファイル名を含む表示を作成します。必要であれば、“TUTORIAL|TutorialApp” のように短いファイル名表示の一部を変更することができます。
説明	チュートリアル アプリケーションを起動する	ツールヒントとして表示されます。

テーブル 2・ショートカットのプロパティ（続き.）

プロパティ	値	コメント
アドバタイズ	はい	実行時に、エンドユーザーがこのショートカットを含む製品または機能をアドバタイズすると、ショートカットが作成されますが、コンポーネントのファイルは、エンドユーザーがショートカットを起動するまでインストールされません。
ターゲット	[INSTALLDIR]Tutorial.exe へのアドバタイズ ショートカット	アドバタイズショートカットはコンポーネントのキーファイルへと自動的に設定されます。
アイコン ファイル	<TutorialSource>¥Tutorial.exe	ソースロケーションから Tutorial.exe を探し出し、そこにあるアイコンを選択します。
アイコン インデックス	0	実行可能ファイルに複数のアイコンリソースがある場合、アイコン インデックスは特定のアイコンを識別します。
作業ディレクトリ	[INSTALLDIR]	作業ディレクトリは [名前を付けて保存] および [開く] ダイアログに対するデフォルトのディレクトリに設定します。

ヒント: *エンドユーザーマシンの既存ファイルにショートカットを作成するには、[アドバタイズ] プロパティを [いいえ] に設定して、そのファイルへのパスを入力しますが、可能ならば Windows Installer のディレクトリプロパティを使用してこのファイルへのパスを指定をします。たとえば、ユーザーマシンの Windows または WinNT フォルダーに置かれた Windows Notepad のコピーを起動するには、ショートカット ターゲットに [WindowsFolder]Notepad.exe と入力します。*

レジストリ データを作成する

インストールに求められる機能の 1 つとして、ターゲット システムのレジストリへの情報の書き込みがあります。コンポーネントにレジストリを追加するには、[レジストリ] ビューを利用することができます。

タスク

HKEY_LOCAL_MACHINE¥SOFTWARE¥Tutorial Co¥Tutorial¥1.00.0000 の下に TutorialData というレジストリ値を作成するには、以下の手順を実行します。

1. **[レジストリ]** ビューを開きます。

   [レジストリ] ビューは、**[ビュー リスト]** の **[システム構成]** セクションにあります。

2. ビューの最上部にある **[ビューフィルター]** から、**Tutorial.exe** を選択します。

3. **[インストール先コンピューターのレジストリビュー]** ペインで、HKEY_LOCAL_MACHINE を右クリックして、**[新規作成]** を選択し、**[キー]** をポイントします。

4. キーの名前を *SOFTWARE* に変更します。

5. サブキー *Tutorial Co*、*Tutorial*、と *1.00.0000* に関しても同じ操作を繰り返します。

6. **[インストール先コンピューターのレジストリデータ]** ペインで、右クリックして、**[新しい文字列値]** を選択します。
7. 値を *TutorialData* に変更します。
8. TutorialData 値をダブルクリックして、"値のデータ" フィールドに *[INSTALLDIR]* と入力します。

これで [**レジストリ**] ビューは、次のように表示されます。

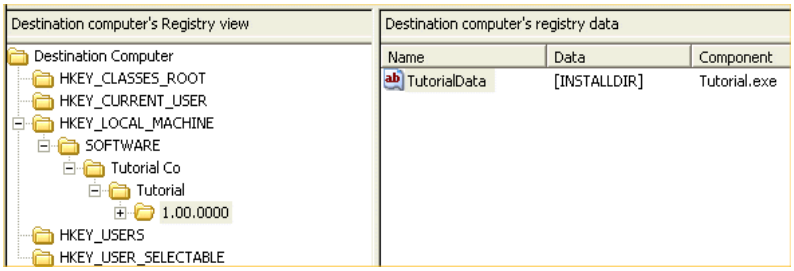


図 3: [レジストリ] ビュー

ヒント: *レジストリに Windows Installer プロパティの値を書き込む場合、その表記は [PropertyName] となります。この例の場合、[INSTALLDIR] というレジストリ値を作成すると、レジストリには INSTALLDIR という値が書き込まれます。*

インストール実行時に Tutorial.exe コンポーネントを含むセットアップの種類や機能のコレクションをユーザーが選択すると、ターゲット システム上にレジストリデータが作成されます。

ショートカットが作成されたことを確認する

タスク ***インストールがショートカットを作成したことを検証するには、以下の手順を実行します。***

1. **[ビルド]** ツールバーボタンをクリックするか、**F7** を押してプロジェクトを再ビルドします。
2. **[実行]** ボタンをクリックするか、CTRL+F5 を押してプロジェクトを実行します (最初にシステムからプログラムの既存バージョンをすべて削除します)。これでチュートリアルアプリケーションのショートカットが、**[スタート]** メニューの **[プログラム]** フォルダーに作成されているはずです。

レジストリデータが作成されたことを確認する

タスク ***インストールがレジストリデータを作成したことを検証するには、以下の手順を実行します。***

1. ショートカットから Tutorial App を起動します。
2. **Tutorial** メニューから **Verify Registry Data** を選択します。レジストリデータが作成されている場合、*INSTALLDIR* の値を表示するメッセージ ボックスが開きます。

チュートリアルの次のステップでは、COM サーバー (自己登録ファイル) の登録方法について説明します。

ステップ 3: COM サーバーの登録

ほとんどのファイルに関して、インストールが果たすべき唯一の役割は、ソース メディアからターゲット システムにファイルをコピーすることです。これに対し一部のファイルについては、ターゲット システムへのファイル登録作業をインストーラーが実行する必要があります。こうした例外的な処理が必要なファイルの 1 つが *COM サーバー*で、これは *自己登録ファイル*や *ActiveX コントロール*として知られれています。COM サーバーは通常 DLL や OCX の形を取りますが、こうした自動登録ファイルは、これらを使用するアプリケーションや Web ページが認識できるよう、あらかじめターゲット システムのレジストリへ書き込んでおくため、追加情報を必要とします。

COM サーバーコンポーネントの作成

これらのファイルのインストールと登録には、コンポーネントウィザードを利用することができます。コンポーネントウィザードでは、ファイルコピーや登録用の追加ステップを実行するコンポーネントを作成します。本ステップでは、Tutorial.ocx 本体およびこれが使用する HTML ファイルのインストールと登録を行うコンポーネントを作成します。

タスク

COM サーバーコンポーネントの作成は、次の手順を実行します。

1. **[ファイルとフォルダー]** ビューを開きます。**[ファイルとフォルダー]** ビューは、**[ビューリスト]** の **[アプリケーション データ]** セクションにあります 。
2. **[ファイルとフォルダー]** ビューの上部にある **[ビュー フィルター]** リストから **Tutorial_Files** 機能を選択します。
3. **[インストール先コンピューターのフォルダー]** ペインで、**[INSTALLDIR]** フォルダーを右クリックし、**[コンポーネントウイザードの起動]** を選択します。
4. **コンポーネント ウイザード**の **[ようこそ]** パネルで、[タイプを選択してコンポーネントを定義する] を選択し、**[次へ]** をクリックします。
5. **[コンポーネントのタイプ]** パネルで **COM サーバー** アイコンを選択し、**"コンポーネント名"** フィールドに *Tutorial.ocx* を入力してから **[次へ]** をクリックします。
6. **[COM サーバー – インストール先]** パネルで、インストール先が [INSTALLDIR] となっていることを確認します。
7. **[COM サーバー ファイル]** パネルで、**"COM サーバー ファイル"** フィールドの隣にある参照ボタンをクリックし、チュートリアル ファイル ソース ディレクトリの Tutorial.ocx を選択します。**[次へ]** をクリックします。
8. コンポーネントウイザード による COM 情報の抽出が完了したら、COM 情報を確認し、**[完了]** をクリックしてコンポーネントを作成します。

次のステップでは、作成したコンポーネントに HTML ファイルを追加します。

タスク ***コンポーネントに HTML ファイルを追加するには、以下の手順を実行します。***

1. **[ファイルとフォルダー]** ビューの **[インストール先コンピューターのフォルダー]** ペインで、新規に作成した Tutorial.ocx コンポーネントを選択します。
2. **[ソース コンピューターのファイル]** ペインから TutorialCtrl.html というファイルを、**[インストール先コンピューターのファイル] ペイン**へドラッグします。
3. Tutorial.ocx がコンポーネントのキーファイルとしてマークされていることを確認します。

メモ *リリースの再ビルドごとにインターフェイスを変更した場合の COM 情報の抽出や、ファイルの自己登録関数の呼び出しなど、自己登録ファイルの登録に関するその他のオプションについては、「COM サーバーの登録」を参照してください。*

COM サーバーが登録されていることを確認する

タスク ***リリースを再ビルドし (F7 キーを押す) インストールを実行 (CTRL+F5 を押す) をした後、COM サーバーが正常に登録されたかを次の手順で確認できます。***

1. **[プログラム]** メニューのショートカットか、アイコンをダブルクリックして、Tutorial App を起動します。
2. **Tutorial** メニューから **COM Server Test** を選択します。
3. COM サーバーが正常に登録されていれば、「登録成功」のメッセージを示す HTML ページが表示されます。

その他にもコンポーネント ウイザードでは、フォントや Windows NT サービスのインストールおよび設定を行うコンポーネントも作成できます。

チュートリアルの次のステップでは、ファイルの条件付きインストールについて説明します。

ステップ 4: 条件とプロパティ

このステップでは、ターゲット システムにデータを条件付きでインストールする方法について解説します。

オペレーティング システム条件

インストール通常求められる機能は、特定の条件が満たされた場合にのみ必要なファイルをシステムにインストールすることです。たとえばある種のファイルは、特定のオペレーティング システムや言語にのみ必要であったり、ユーザー必要な権限を持つ場合に限りインストールできる、ということがあります。

コンポーネント (およびそのファイルとその他のデータ) を特定のオペレーティング システムにのみインストールさせるには、コンポーネントの "オペレーティング システム" プロパティを使います。コンポーネントのプロパティの変更は、[セットアップのデザイン] ビューを開いて該当する機能のアイコンを展開し、必要なコンポーネントを選択して行います。

タスク

Windows 7 またはそれ以降を実行するシステム上でのみインストールが可能なコンポーネントを作成するには、以下の手順を実行します。

1. **[セットアップのデザイン]** ビューを開きます。**[セットアップのデザイン]** ビューは、**[ビューリスト]** の **[編成]** セクションにあります 。
2. **Help_Files** 機能を右クリックして、**[新しいコンポーネント]** を選択します。
3. コンポーネントの名前を *Windows_7_Files* に変更します。
4. Windows_7_Files コンポーネントを開き、コンポーネントの [**ファイル**] アイコンをクリックし、[ファイル] ペイン内で右クリックしてファイルを参照してチュートリアル ファイル ソース フォルダー から ReadmeNT.txt ファイルを追加します。
5. .txt ファイルを右クリックして **[キーファイルの設定]** を選択します。
6. Windows_7_Files コンポーネントをクリックして、コンポーネントのプロパティ グリッドを表示します。
7. コンポーネントの [条件] プロパティを選択して参照ボタンをクリックし、**[条件ビルダー]** ダイアログ を起動します。
8. 次の条件を作成します: *VersionNT>=601*条件の作成に関する詳細については、「条件ステートメントのビルド」を参照してください。
9. **[OK]** をクリックして **[条件ビルダー]** ダイアログを閉じ、条件を追加します。

(F7 を押して) 再ビルドの後、(CTRL+F5 を押して) インストールを実行する と、ターゲット システムが Windows 7 以降を実行している場合に限りコンポーネントに含まれるファイルおよびその他のデータがインストールされるようになります。

Windows Installer 条件

Windows Installer サービスは、作成したインストールおよびユーザーのオペレーティング システムに関して、一部のグローバル情報をプロパティとして記録します。こうしたプロパティには、MSI データベースの Property テーブルに用意されているものもあれば、ユーザーによるインストールの起動時に、Windows Installer エンジンが作成して設定するものもあります。

条件の判定でよく使用されるのは、次のプロパティです:

- AdminUser、これはインストールを実行するユーザーが管理者権限を持つ場合に設定されます。
- VersionNT、これはユーザーが使用しているオペレーティング システム バージョンを数値で示します。
- PhysicalMemory、これはユーザーが使用しているシステムの RAM をメガバイト単位で示します。

Windows Installer 条件は論理ステートメントで、プロパティ値を定数値と比較したり、プロパティがついているかの判定を行います。たとえば、Windows Installer は ScreenX および ScreenY というプロパティを定義しますが、これらはユーザーが使用するモニターの解像度を、ピクセル単位で示します。解像度が 800 × 600 以上であるかを調べる Windows Installer 条件の場合は、“(ScreenX>=800) And (ScreenY>=600)” と記述します。

定義済みのプロパティは、それだけで判定条件として使用できます。たとえば、AdminUser プロパティはユーザーが管理者権限を持つ場合に限り設定されるので、ユーザーが管理者権限を持っているかどうかを判断する条件は、単に ″AdminUser″ です。

タスク ***ユーザーが管理者権限を持つ場合にのみインストールされるコンポーネントを作成するには、次の手順を実行します。***

1. **Help_Files** 機能を右クリックして、**[新しいコンポーネント]** を選択します。
2. コンポーネントの名前を *Admin_Component* に変更します。
3. **Admin_Component** コンポーネントを開いて **[ファイル]** アイコンをクリックします。
4. チュートリアル ファイル ソース フォルダーから AdminOnly.txt ファイルを追加し、これをコンポーネントのキーファイルとして設定します。
5. コンポーネントの **"条件"** プロパティで参照ボタンをクリックして、**[条件ビルダー]** ダイアログ を表示します。
6. **[条件ビルダー]** ダイアログ ボックスで、**"条件"** フィールドに *AdminUser* と入力します。
7. **[OK]** をクリックします。

完成後のインストーラーを実行すると、このコンポーネントのデータは、ユーザーが管理者権限を持つ場合にのみインストールされます。

チュートリアルの次のステップでは、インストールのユーザー インターフェイスの変更方法について説明します。

ステップ 5: エンドユーザー インターフェイスを変更する

このステップでは、インストールのユーザー インターフェイスを変更する 3 種類の方法について説明します。

- インストール中に表示するダイアログを指定する。
- ダイアログ エディターを使用して、ダイアログのレイアウトとプロパティを変更する。

新しいダイアログの追加

基本の MSI プロジェクトは、インストールのユーザー インターフェイスで表示可能な多数のダイアログ ボックスが用意されています。このチュートリアルの「インストールを実行する」トピックでは、プロジェクト アシスタントの [インストール アーキテクチャ] ページで行った選択に基づいてインストールが表示するダイアログについて説明します。

タスク ***新しいダイアログを作成するには、以下の手順を実行します。***

1. **[ダイアログ]** ビューを開きます。**[ダイアログ]** ビューは、**[ビューリスト]** の **[ユーザー インターフェイス]** セクションにあります 。
2. **[すべてのダイアログ]** エクスプローラーを右クリックして、**[ダイアログの新規作成]** を選択します。**ダイアログ ウィザード**が開きます。.**[次へ]** をクリックして、**[ようこそ]** パネルをとばします。
3. **[ダイアログ テンプレート]** パネルで、**[内部ウィザード パネル]** をクリックして、**[このダイアログを [ユーザー インターフェイス] シーケンスに挿入する]** チェック ボックスを選択します。

4. **[ユーザー インターフェイス]** パネルで、**[ユーザー シーケンス]** リストから **[インストール]** を選択します。ダイアログのリストから、**InstallWelcome** を選択します。これらの選択に基づき、InstallShield は新しいダイアログをシーケンスの **InstallWelcome** ダイアログの直後に挿入します。
5. **[ダイアログの位置と条件]** パネルでは、デフォルト設定をそのままにして **[完了]** をクリックします。新しいダイアログが **[ダイアログ]** リストに表示されます。
6. ダイアログを右クリックして **[名前の変更]** を選択します。**WelcomeBitmap** ダイアログの名前を変更します。

同様の手順で、インストール プログラムのユーザー インターフェイスにダイアログを追加することができます。

ダイアログ エディターでダイアログのレイアウトを変更する

ダイアログ エディターを使って、インストールで表示されるダイアログの外観を修正することができます。

タスク ***このステップでは、作成したばかりの WelcomeBitmap ダイアログの修正を行います。***

1. まず、(Microsoft Paint のようなプログラムを使用して) 300 x 150 サイズのビットマップを作成します。
2. **[ダイアログ]** ビューを開きます。
3. WelcomeBitmap ダイアログのノードを開きます。**[日本語]** をクリックして、**ダイアログ エディター**を開きます。
4. ダイアログ上部にある [**ダイアログ太字タイトル**] テキスト ボックスをクリックします。**"テキスト"** フィールドに*「ようこそビットマップ」*と入力します。これでダイアログのメインタイトルが変更されます。
5. ダイアログ上部にある [**ダイアログ標準説明**] テキストボックスをクリックします。**"テキスト"** フィールドに*「ようこそビットマップを表示する」*と入力します。これでダイアログの説明が変更されます。
6. **"ダイアログ コントロール"** ツールバーにある **[ビットマップ]** ボタンをクリックして、カーソルを利用してダイアログ上にボックスをドラッグします。高さ 150、幅 300 に設定します。
7. **"ファイル"** フィールドで、ステップ 1 で作成したビットマップ ファイルを参照します。

プロジェクトを再ビルド (F7 キーを押す) してから実行すると (CTRL+F5 キーを押す)、Welcome Bitmap ダイアログは InstallWelcome ダイアログの後に表示されます。

ヒント: *CTRL+M を押すと、[ダイアログ エディター] ビューが最大化します。*

グローバリゼーション チュートリアル

グローバリゼーションチュートリアルでは、InstallShield が提供するグローバルインストール パッケージ作成のためのツールとオプションを紹介します。グローバルインストールは、多くの異なる言語で実行できるセットアップです。インストールのビルド方法に応じて、1 つのパッケージにすべての言語を含めてユーザーに言語の選択オプションを提供する、またはターゲットとするそれぞれの言語ごとに個別のインストールパッケージをビルドすることもできます。このチュートリアルでは、すべての言語を含むインストールの作成過程を説明します。

エディション: *このチュートリアルを完了するには、開発システム上で InstallShield Premier Edition のインストールが必要です。*

InstallShield チュートリアルをまだご利用になったことがない方は、インストールパッケージの作成方法を理解するために 基本の MSI チュートリアルを是非一度お試しいただくことをお勧めいたします。基本の MSI チュートリアルを完了すると、別の言語を追加するのに最適な MSI インストール プロジェクトを作成することができます。基本の MSI チュートリアルの完全プロジェクト ファイルは、InstallShield Program Files フォルダーの Samples サブフォルダーに製品と共にインストールされます。デフォルト保存先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Tutorial Project¥Tutorial

プロジェクト: *チュートリアル内の情報はほとんどすべて InstallScript インストール プロジェクトにも適用されます。異なる点についてはチュートリアル テキストに記述してあります。*

プロジェクトファイルを開く

チュートリアルでは、InstallShield を使ってインストールするサンプル プロジェクトファイルとして Othello.ism を利用します。

タスク ***チュートリアルを開始するには、以下の手順に従います:***

Othello.ism (InstallShield Program Files フォルダーの Samples サブフォルダーにあります) を開きます。デフォルト保存先は次の場所です:

C:¥Program Files¥InstallShield¥2022¥Samples¥WindowsInstaller¥Basic Installation Project

プロジェクト: *Othello.ism は基本の MSI インストール プロジェクトです。このチュートリアルに記述されている情報のほとんどは、InstallScript インストール プロジェクトにも適用されます。異なる点についてはチュートリアルテキストに記述してあります。*

ターゲット言語の選択

グローバルインストールを作成するには、まずターゲットとする言語を選択します。このチュートリアルでは、インストール プロジェクトに 2 つの言語 (ドイツ語とポーランド語) を追加します。

タスク ***プロジェクトに言語を追加するには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"セットアップ言語"** 設定で、省略記号ボタン (...) をクリックします。**[セットアップ言語]** ダイアログ ボックスが開きます。
3. **[英語(米国)]**、**[ドイツ語]**、および **[ポーランド語]** チェック ボックスを選択します。

プロジェクトに、英語、ドイツ語、およびポーランド語の文字列エントリが追加されます。各文字列エントリは、言語非依存の識別子と対応する言語固有の値で構成されます。文字列エントリには、翻訳済みのビルトイン ユーザー インターフェイス文字列リソースが含まれます。すべての文字列エントリを参照するには、[文字列エディター] ビューを使用します。([ユーザー インターフェイス] の下のビュー リストにある [文字列エディター] をクリックします。)

デフォルト言語に新しい文字列エントリを追加するたびに、プロジェクトに含まれているすべての別の言語用に、対応する文字列エントリが作成されます。

言語固有の文字列エントリを編集する

次のステップは、インストール プロジェクトの文字列エントリの編集です。文字列エントリを、3 つの異なる設定 (機能の表示名、ショートカットの表示名、およびショートカットの説明) で編集します。(ショートカットの説明は、チュートリアルの次のステップで作成します。)

機能の表示名

タスク ***機能の表示名を提供するには、以下の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、**ProgramFiles** をクリックします。
3. 右側のペインに表示される **"表示名"** 設定に「**プログラム ファイル**」が既に存在しない場合は、それを入力します。

[セットアップのデザイン] ビュー内の別の場所をクリックすると、入力したテキストの始まりに中括弧 ({}) に囲まれた文字列 ID が追加されていることが確認できます。"表示名" 設定をクリックしたときに表示される省略記号ボタン (...) をクリックすると、プロジェクトに含まれるすべての文字列エントリを参照できます。

ショートカットの表示名

タスク ***ショートカットの表示名を追加するには、以下の手順を実行します。***

1. **[セットアップのデザイン]** エクスプローラーで、**ProgramFiles** 機能を展開して、**Program_Executables** コンポーネントを展開します。

   Program_Executables コンポーネントには、Othello.exe ファイルへショートカットが含まれています。
2. **Program_Executables** コンポーネントの下にある **[ショートカット]** アイコンをクリックします。

3. **[ショートカット]** エクスプローラーで、**Othello** ショートカットをクリックします。右側のペインにショートカットの設定が表示されます。

   現在、**"表示名"** 設定は、**Othello** に設定されています。この値の始まりには、中括弧 ({}) で囲まれた文字列 ID が追加されています。

4. 表示名を変更するには、新しい名前を **"表示名"** 設定に入力します。

文字列エントリの作成

テキスト文字列の多くはプロジェクトの複数の場所で使用される可能性があるため、これらの文字列の各インスタンスをプロジェクトに保存することは非効率的です。その代わり、文字列を一度作成して必要な任意の場所でその文字列を使用することができます。プロジェクトのローカライズ処理を簡素化するために、インストール処理中、実行時に表示されるすべてのテキスト文字列は、統合された 1 つのビュー、[文字列エディター] ビューに表示されます。このビューを使って新しい文字列エントリを作成できます。

"ショートカットの説明" 設定には、[文字列エディター] ビューを使って新しい文字列エントリを入力します。次に、この文字列をショートカットの説明に関連付けます。

タスク ***新しい文字列エントリを作成して、それを "ショートカットの説明" 設定で使用するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 以下のいずれかを実行します。
   - **[新規]** ボタンをクリックします。
   - INSERT キーを押す。

   [文字列エントリ] ダイアログ ボックスが開きます。
3. **[文字列 ID]** ボックスで、次のように入力します:

   MYSTRING
4. **[値]** ボックスで、次の文字列を入力します:

   これは Othello ショートカットの説明です。
5. **[コメント]** ボックスに、オプションとして、文字列エントリについて内部で使用するメモを指定します。コメントは実行時には表示されません。
6. **[OK]** をクリックします。[文字列エディター] ビューで各言語ごと (英語、ドイツ語、ポーランド語) に新しい行が追加されます。
7. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
8. **[ショートカット]** エクスプローラーで、**Othello** ショートカットをクリックします。右側のペインにショートカットの設定が表示されます。
9. **"説明"** 設定で、省略記号 (...) をクリックします。**[文字列の選択]** ダイアログ ボックスが開きます。
10. **MYSTRING** 行を選択してから、**[OK]** をクリックします。

新しい文字列が**"説明"** 設定の値として入力されます。

言語別ファイルとコンポーネントを含める

次のステップでは、インストールの中に言語固有のファイルおよびコンポーネントを含むグローバルインストールを作成します。たとえば、作成したプログラムファイルの多くが特定言語に依存していない場合でも、ヘルプファイルおよびランタイム文字列は共に言語固有の情報を含みます。この実習の目的は、インストールに 3 つの新規コンポーネントを追加することです。各コンポーネントには、サポートするすべての言語にローカライズされた Readame ファイルが含まれます。

タスク ***プロジェクトにコンポーネントを追加するには、以下の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、**ProgramFiles** という名前の機能を右クリックしてから、**[新しいコンポーネント]** をクリックします。
3. コンポーネントの名前として *English_Readme* と入力します。
4. この手順を 2 回繰り返します。それらのコンポーネントに *Polish_Readme* および *German_Readme* と名前を付けます。

InstallShield には、翻訳済みのシンプルな Readme ファイルが添付されています。これらのファイルは、InstallShield Program Files フォルダーの Samples サブフォルダーにあります。

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Basic Installation Project¥Data Files¥Readme

タスク ***English_Readme コンポーネントにファイルを追加するには、以下の手順を実行します:***

1. 新しい **English_Readme** コンポーネントを展開して、**English_Readme** の下にある **[ファイル]** アイコンをクリックします。
2. **[ファイル]** リストを右クリックして、**[追加]** をクリックします。
3. English.txt ファイルを参照して、**[開く]** をクリックします。

上記のステップをドイツ語およびポーランド語のコンポーネントに対して繰り返し、それぞれ Deutsch.txt および Polski.txt を追加します。

コンポーネント インストール条件を指定する

言語固有のコンポーネントを作成した後は、論理式を指定することによって作成したどのコンポーネントをインストールするかをインストーラーに通知する必要があります。コンポーネント条件を指定して、ターゲット システムのデフォルト言語を判別し、該当するファイルをインストールできます。作成した 3 つの言語固有のコンポーネントのそれぞれに対して条件が必要です。この条件が True と評価された場合に、コンポーネントがインストールされます。

タスク ***コンポーネント条件を作成するには、以下の手順を実行します。***

1. German_Readme コンポーネントをクリックします。
2. 右側のペインで、**"条件"** 設定をクリックしてから、この設定の省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開きます。
3. **[プロパティ]** リストで、**SystemLanguageID** を選択してから、**[追加]** ボタンをクリックします。
4. **[演算子]** リストで、等号 (=) を選択してから、**[追加]** ボタンをクリックします。

   [条件] ボックスには、これまでに選択した内容を反映する **SystemLanguageID =** が含まれています。
5. 次に、インストールの実行時に調べる値を与える必要があります。現在ドイツ語コンポーネントを編集中のため、等号の後に **1031** と入力します。1031 はドイツ語の言語 ID です。コンポーネントは、この式が真の場合 (つまり、ターゲットシステムの言語がドイツ語の場合) にのみインストールされるため、ドイツ語で実行されていないマシンにはコンポーネントはインストールされません。

上記のステップにしたがって、同様に Polish_Readme コンポーネントに条件を追加します。言語値として、1031 ではなく、ポーランド語の言語 ID である 1045 を使用します。

デフォルトの言語として、1 つの言語を選択する必要があります。この例では、英語がデフォルトです。したがって、English_Readme コンポーネントに使用する条件は他の 2 つとは異なります。English_Readme コンポーネントの条件は、以下のとおりです:

SystemLanguageID◇1045 AND SystemLanguageID◇1031

この論理により、ターゲットマシンの言語がドイツ語またはポーランド語のどちらでもない場合に、English_Readme コンポーネントがインストールされます。

プロジェクト: *InstallScript と InstallScript オブジェクト プロジェクトで実行時にインストールされる言語依存コンポーネントを指定する方法については、「言語基づいてコンポーネントをインストールする」を参照してください。*

文字列の翻訳

インストール プロジェクトをビルドする前に、機能の表示名、ショートカットの表示名、およびショートカットの説明に入力した英語の文字列を翻訳する必要があります。このチュートリアルでは、翻訳済みの文字列を用意しました。ここでは、ドイツ語とポーランド語文字列エントリに正しいテキストを入力するだけです。[文字列エディター] ビューを使用できます。

ドイツ語文字列値

[文字列エディター] ビューを開きます。次の各ドイツ語の文字列 ID を見つけます。[検索グリッド] ボックスに文字列 ID を入力して、各文字列を簡単に見つけることができます。代りに、ID 列ヘッダーをクリックして、すべての文字列エントリを ID 順に並べ替えることもできます。3 つの文字列エントリのドイツ語の値を、以下のように更新します:

テーブル 1 - ドイツ語文字列エントリ

Identifier	値
FEAT_DISPLAYNAME	Programmdateien
SHORTCUT_DISPLAYNAME	Othello-Verknuepfung
MYSTRING	Dies ist die Beschreibung der Verknuepfung fuer Othello.

ポーランド語文字列値

3 つの文字列エントリのポーランド語の値を更新します:

テーブル 2 - ポーランド語文字列エントリ

Identifier	値
FEAT_DISPLAYNAME	Pliki Programu
SHORTCUT_DISPLAYNAME	Skrt do Othello
MYSTRING	Opis skrtu Othello.

通常は、翻訳用に文字列エントリをエクスポートします。詳細については、「文字列エントリの翻訳」を参照してください。このチュートリアルでは [文字列エディター] ビューで文字列エントリを編集した方が簡単です。

インストールのビルド

インストール プロジェクトは完全にグローバル化されており、直ぐにテストできる状態です。ただしインストールをテストする前にビルドしなくてはなりません。

タスク ***インストールをビルドするには、次の手順に従います:***

1. ツールバーの **[リリース ウィザード]** ボタンをクリックします。
2. 最初の 2 つのパネルでは、製品構成とリリース名を指定します。
3. **[セットアップ言語] パネル**を除き、他のすべてのパネルはデフォルト設定のままにします。
4. [**セットアップ言語**] パネルでは、インストールに含む言語を選択することができます。[一般情報] ビューで指定した言語 (英語、ポーランド語、およびドイツ語) のみが、使用可能言語のリストに表示されます。各言語のとなりにあるチェック ボックスを選択します。

5. また、必ず **[セットアップ言語ダイアログを表示する]** チェック ボックスも選択してください。このダイアログは、インストールを実行する言語をユーザーが選択できるようにします。
6. 残りのウィザードパネルはデフォルトの設定のままにします。
7. **[概要]** パネルに到達したら、**[リリースをビルドする]** チェック ボックスが選択されていることを確認して、**[完了]** ボタンをクリックします。
8. InstallShield がリリースをビルドします。

インストールの実行

タスク ***インストールを実行するには、以下の手順を実行します。***

1. ツールバーの **[セットアップの実行]** ボタンをクリックします。[**セットアップ言語の選択**] ダイアログが表示されます。
2. インストールを実行するには、**[ドイツ語 (標準)]** を選択して **[OK]** をクリックします。これ以降は、すべてのダイアログがドイツ語で表示されます。

メモ: *インストールが特定の言語で実行された後、Windows Installer はこの情報をキャッシュして、インストールを常に同じ言語で実行します。*

インストール ウィザードの処理が進むにつれ、いくつかのボタン サイズが適切でないことが確認できます。この問題の解決策として、ダイアログ エディター を開いてテキストが正しく収まるようにコントロールのサイズを変更して簡単に修正することができます。

[カスタム セットアップ] パネル (ドイツ語では Angepasstes Setup と呼びます) では、機能名が *Programmdateien* となります。ローカライズした文字列エントリが、インストールの一部となっています。

インストールのテスト

最後のステップは、作成したグローバライズされたインストールのテストです。

タスク ***インストールをテストするには、以下の手順を実行します。***

1. **[スタート]** メニューを開き、**[プログラム]** を選択します。Othello Verknpfung として Othello ショートカットが表示されます。ショートカットの説明もドイツ語で表示されていることを確認できます。
2. Othello のインストールディレクトリに移動します。ディレクトリは <Program Files フォルダー>¥Shakespeare Inc¥Othello になります。インストールされた readme ファイルの名前は Deutsch.txt です。

4

インストールの作成

コンピューターにアプリケーションをインストールしたことがある方は、実行中のインストールを既にエンドユーザーの視点でご覧になっています。インストールの主要な役割は、ソース メディアからローカル ドライブへファイルを転送することです。インストールはまた、ユーザー インターフェイスを表示してエンドユーザーの選択を取得し、ターゲット システムを構成し (たとえば、必要なレジストリ エントリやショートカットの作成)、またインストール済みアプリケーションの変更またはアンインストールを行います。インストールの作成は次のタスクの一部またはすべての処理を含みます。

インストール情報の指定

[一般情報] ビューで入力した基本情報は、インストールの様々な場所で利用されます。たとえば、製品名はアプリケーション情報 レジストリ キーの作成に利用されます。

ファイルの編成と転送

ファイル転送は CD または DVD のようなソース メディアからファイルをエンドユーザーのマシンにあるローカル ドライブへコピーします。エンドユーザーがセットアップの種類 (InstallScript または InstallScript MSI インストールの場合) または機能を選択して希望した構成に基づき、ファイルの一部またはすべてがローカルディスクに転送されます。

インストールするファイルをセットアップの種類および機能に編成して、エンドユーザーが最も適切なファイルを選択できるようにします。各機能について、たとえば同じターゲットフォルダーにインストールするファイルなど、ファイルをその種類と目的に応じてコンポーネントに編成します。

ターゲット システムの構成

ファイルをインストールするだけでなく、ショートカットやプログラムフォルダーの作成、レジストリの変更、初期化ファイル (.ini ファイル) データの変更、ODBC リソースの構成、環境変数の変更、XML ファイルの変更、テキスト ファイルの変更、タスクのスケジュール、および Windows サービスのインストールと制御が可能です。

インストール動作のカスタマイズ

InstallShield では、幅広いカスタマイズ オプションが提供されています。InstallScript インストールは InstallShield のシンプルかつパワフルな InstallScript プログラム言語によって実行が進められます。InstallScript では、ビルトイン関数の他に DLL および Windows API 関数の呼び出しも可能で、子インストールやその他のアプリケーションを

インストールから起動することも可能です。Windows Installer ベースのインストールでは、カスタム アクションを使った InstallScript、VBScript、または JavaScript コードの実行、DLL 関数の呼び出し、実行可能ファイルの実行、マネージ アセンブリ内にあるマネージ メソッドの呼び出し、プロパティやディレクトリの設定、エラーの起動とインストールの中止、他のインストール パッケージの実行が可能です。

エンドユーザー インターフェイスの定義

インストールのエンドユーザー インターフェイスは、エンドユーザーに対して情報およびインストール構成オプションを提供します。エンド ーザーはユーザー インターフェイスを通して、製品の一部をインストールするか、一部のファイルをソース メディアに残すか、使用許諾契約を表示するか、またはインストールを正しく構成するための情報をインストーラーに提供するかを選択できます。

ユーザー インターフェイスはインストールのニーズに合わせたカスタマイズが可能です。たとえば、ソフトウェアを不正な使用から保護するために、インストールの前にシリアル番号の入力をユーザーに求めるように指定できます。ファイル転送中、インストールは新機能や便利なヒントなどの製品情報を提供するビルボードを表示します。ファイル転送処理の進行状況がわかるステータス バーも表示されます。

サーバーの構成

サーバー側インストールには、新しい IIS Web サイトの作成および管理、COM+ アプリケーションおよびコンポーネントの管理、またはサーバー接続および設定による SQL スクリプトの管理および編成が必要な場合があります。

インストールにメンテナンスとアンインストール機能を追加する

アプリケーションのアンインストール、変更、または修復を行うためには、アプリケーションの存在がオペレーティング システムで示されなくてはなりません。これに対応するには、アプリケーションを簡単にメンテナンスまたはアンインストールできるように、インストールでアプリケーションをオペレーティング システムに登録します。

この処理で登録されるほとんどの情報は、コントロール パネルの [プログラムの追加と削除] でエンドユーザーに提供されます。たとえば、テクニカル サポートの連絡先情報、製品更新の情報、製品のバージョン、および製品発行元の情報はこのプロセスで登録されます。

インストールのビルド、テスト、および配布

インストール プロジェクトを作成した後は、インストールのビルド、テストおよび配布が必要です。つまりユーザーへリリースするファイルを作成し、インストールのエラーをテストし、オプションとしてローカル、ネットワークまたは FTP サイトへファイルをコピーします。

まず始めに

InstallShield ヘルプ ライブラリの「まず始めに」は、インストール作成者が InstallShield で新規のインストール プロジェクトを作成するときに役に立つ情報が盛り込まれています。各トピックでは、Windows Installer、Windows ロゴ プログラム、INSTALLDIR、TARGETDIR、およびインストール開発のその他の分野についてのバックグランド情報を読むことができます。

このセクションには、次のトピックが含まれています:

- Windows ロゴ プログラムの要件
- Windows Installer 入門

- インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する
- アプリケーションのディスクの使用量を設定する
- NSTALLDIR と TARGETDIR の違い
- 現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ
- 非管理者パッチのインストールを準備する
- プラットフォームとプラットフォーム スイートの設定

Windows ロゴ プログラムの要件

WWindows 8 デスクトップ アプリの認定を受けるために製品とそのインストールが満たさなくてはならない要件の一覧が、マイクロソフトによって確立されています。要件では、製品が Windows システムで実行されたとき、より高い互換性と安定性、および安全性を持つことができる基準が規定されています。Windows 8 デスクトップ アプリの認定要件を満たす製品は、Compatible with Windows 8 ロゴを表示できます。

Windows ロゴの取得について

Windows ロゴの取得に関する詳細は、「MSDN」を参照してください。この Web サイトには、Windows ロゴ プログラムに関する情報が掲載されています。

InstallShield の Certified for Windows 検証スイート

製品が Windows ロゴ プログラムの要件を満たしているかどうかを判断するために、インストール パッケージまたはマージ モジュールの検証が役に立つ場合があります。パッケージまたはマージ モジュールが 1 つ以上の検証規則に違反した場合、InstallShield は違反の対象となった特定の規則をレポートし、問題を解決するための追加情報を提供します。

このため、Windows ロゴの取得に関心があるユーザーは、次の両方のスイートを使ってインストール パッケージの検証を行うことをお勧めします:

- **InstallShield 検証スイート – Windows 8**—このスイートは、インストール時に潜在的に Windows システムで予期しない動作が発生する原因になる問題を識別する多数の InstallShield 内部整合性評価プログラム (ISICE) から構成されています。このスイートは、完全 MSI 検証スイートでは必ずしも発見できるとは限らない問題を確認します。
- **完全 MSI 検証スイート**—このスイートは、マイクロソフトが作成した ICE から構成されています。

InstallShield でマージ モジュールを作成した場合、次のスイートを利用して、マージ モジュールを検証することができます:

- **InstallShield マージ モジュール検証スイート – Windows 8** このスイートは、Windows システム上でマージ モジュールの予期しない動作が発生する原因になる問題を識別する多数の InstallShield (ISICE) から構成されています。このスイートは、マージ モジュール検証スイートでは必ずしも発見できるとは限らない問題を確認します。
- **マージ モジュール検証スイート**—このスイートは、マイクロソフトが作成した ICE から構成されています。

その他の検証スイートもあります。

詳細については、「プロジェクトの検証」を参照してください。

Windows ロゴ・*InstallShield ヘルプ ライブラリ全体を通して、情報が Windows ロゴ プログラム ガイドラインのコンプライアンスに関連する箇所では Windows Logo ガイドライン アラートが表示されています。*

Windows Installer and Logo Requirements (Windows Installer ヘルプ ライブラリ)

Windows Installer 入門

Windows Installer ベースのインストールは .msi パッケージとして配布され、パッケージは Windows Installer データベース (.msi データベース) と関連データ ファイル (.cab ファイル、非圧縮データ ファイルなど) から構成されます。.msi データベースは COM 構造のストレージとして実装され、ターゲット システム上で処理される変更について記述する何十ものテーブルで構成されます。次に、.msi テーブルのいくつかの例を挙げます:

- File – インストールされるファイルを記述する
- Registry – 書き込まれるレジストリ データを記述する
- Shortcut – ショートカットの設定を記述する

その他の .msi データベース テーブルは、インストールのユーザー インターフェイスの外観と動作を記述したり、Windows サービスおよび ODBC 情報を構成したり、ターゲット システムの特徴を確定したり、インストール中に使用されるアイコンやその他のバイナリ データを保管したりします。

開発者の視点から見て、Windows Installer インストール プログラムの最も大きな変更点は、スクリプトを明示的に書き込む必要がないという点です。その代わり、Windows Installer ベースのインストールは標準カスタムアクションを実行して、ダイアログを表示したり、ターゲット システムをクエリしたり、ターゲット システムに変更を加えたりします。これらのアクションはシーケンスに配置されていて、アクションの集合が順番に並べられています。

Windows Installer は製品のインストール管理専用のアプリケーション プログラム インターフェイス関数の集合 (または API) を含みます。Windows Installer で提供されている機能を活用するには、アプリケーションが Windows Installer API を呼び出す必要があります。

Windows オペレーティング システムに統合されている Windows Installer は、アプリケーションやシステム ツールを管理するためのインターフェイスだけでなく、コンポーネントの管理を行うための標準フォーマットも提供します。様々なバージョンの Windows Installer が Windows オペレーティング システムの再配布可能ファイルとして利用できます。

.msi データベース テーブルを直接編集して Windows Installer パッケージを作成することは可能ですが、多数のテーブルとその関連性を編集するには非常な困難を伴います。InstallShield は、Windows Installer 用のインストール開発過程を様々なビューに分け、.msi データベースにまつわる実装についての詳細の多くのを開発者の目から遮るグラフィカル エディターやウィザードを提供します。

Windows Installer テクノロジについての詳細は、Windows Installer ヘルプライブラリを参照してください。

インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する

プロジェクト・*この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

特定の詳細の一部は、これらのプロジェクト タイプの一部にのみ適用します。これらの違いについては、必要に応じて記述されています。

Windows Vista 以降およびユーザー アカウント制御 (UAC) の目的は、ユーザーが常に標準ユーザーとして実行できるようにすることです。昇格が必要になることは、ほとんどありませんが、必要となる場合、できるだけ短い時間に限られる必要があります。

InstallShield のいくつかの異なる領域では、昇格された権限を要求するために、インストールで UAC の同意または資格情報プロンプトを起動するかどうかが問題になります。Windows Vista 以降のシステムでエンド ユーザーがインストールを実行する際の UAC の動作を適切に定義するためには、これらの異なる設定をよく理解することが必要です。また、これはインストール中に表示される UAC プロンプトの数を最小限に抑えるための手助けとなります。

構成方法によって、InstallShield 前提条件を含むインストールがインストール中のいくつかの時点で、Windows Vista 以降のシステム上で昇格された権限のプロンプトを表示することができます:

1. エンド ユーザーが Setup.exe ファイルを起動するとき
2. Setup.exe ファイルが、昇格された権限を必要とするセットアップ前提条件を起動するとき
3. インストールされる 1 つまたは複数の機能に関連付けられた機能前提条件が含まれるため、ISInstallPrerequisites カスタム アクションが Setup.exe ファイルを機能前提条件インストール モードで再起動するとき

   ISInstallPrerequisites カスタム アクションは、昇格された権限のプロンプトが表示される前に、機能前提条件が昇格された権限を必要とするかどうかを検証しませんので、ご注意ください。また、ISInstallPrerequisites カスタム アクションは、機能前提条件のインストールが必要かどうかを決定するために機能前提条件の条件を確認することもありません。昇格された権限のプロンプトは常に表示されます。

   プロジェクト・*基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*
4. Windows Installer が .msi パッケージの [実行] シーケンスを開始するとき

   プロジェクト・*この最後のインストール ポイントは、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに適用しますが、InstallScript プロジェクトには適用しません。*

InstallShield における UAC 関連の設定

以下は、Windows Vista 以降のシステム上で、インストール中に UAC プロンプトを表示するかどうかを判断するための InstallShield の設定です。

- **必要実行レベル**—[リリース] ビューにあるこの設定を使用して、インストールの Setup.exe ファイルが必要とする最小実行レベルを指定します。InstallShield は、Setup.exe ランチャに埋め込まれるアプリケーション マニフェストで選択した値 (管理者、最高権限、または起動者) を使用します。詳細については、「Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する」を参照してください。
- **前提条件は管理者権限を必要とする**—インストールの InstallShield 前提条件を作成または変更している場合、このチェック ボックスを使用して、前提条件をインストールするために管理者権限が必要かどうかを指定し

ます。このチェック ボックスは、InstallShield 前提条件エディターの [動作] タブにあります。詳細については、「InstallShield 前提条件に管理者権限が必要であることを指定する」を参照してください。

- **管理者権限が必要**—[一般情報] ビューの [概要情報ストリーム] 領域にあるこの設定を使用して、インストールの .msi パッケージの [実行] シーケンスで管理者権限が必要かどうかを指定します。[いいえ] を設定した場合、InstallShield は Word Count Summary プロパティ の 3 ビット目を設定して、製品のインストールに昇格された権限が必要であることを示します。詳細については、「概要情報ストリーム データを入力する」を参照してください。

***プロジェクト・**"管理者権限が必要" 設定は、InstallScript プロジェクトには適用しません。*

- **前提条件が昇格必要時のアドバタイズ**—[リリース] ビューのこの設定を使用して、Windows Vista 以降のマシンにおけるインストールで InstallShield 前提条件が昇格された権限で正常にインストールされたあと、.msi パッケージをアドバタイズするかどうか指定します。アドバタイズすると指定した場合、それをサイレントまたは完全ユーザー インターフェイス (UI) で実行するかどうかも指定します。アドバタイズすることにより、エンドユーザーに UAC のプロンプトを避けることを許可できる場合があります。アドバタイズを行わない場合、昇格された権限を必要とする .msi パッケージに対して、UAC のプロンプトが表示されます。詳細については、「InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する」を参照してください。

***プロジェクト・**"前提条件が昇格必要時のアドバタイズ" 設定は InstallScript プロジェクトには適用しません。*

また、InstallShield 前提条件の種類 (セットアップ前提条件または機能前提条件)—は、UAC プロンプトが Windows Vista 以降のシステム上でのインストール中に表示されるかどうかに影響を及ぼす可能性があります。これら 2 種類の InstallShield 前提条件についての詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。

以下は、Windows Vista 以降における UAC 関連の動作についての注意点です:

- "必要実行レベル" が [起動者] に設定されていて、インストールが管理者権限を必要とする InstallShield 前提条件を含み、"管理者権限" が [いいえ] に設定されている場合、インストール中、プロンプトはエンドユーザーに表示されません。
- "必要実行レベル" が [起動者] に設定されていて、インストールが管理者権限を必要とするセットアップ前提条件を含み、"管理者権限" が [いいえ] に設定されている場合、インストール中、プロンプトはエンドユーザーに 1 回のみ表示されます。再起動がある場合、再起動ごとに、1 回のみ UAC プロンプトが表示されます。
- セットアップランチャーの完全ユーザー インターフェイスが表示される場合で、インストールする必要があるセットアップ前提条件がインストールに含まれているとき、セットアップランチャーは通常、メイン インストールが開始する前にセットアップ前提条件ダイアログを表示します。インストールが必要な 1 つまたは複数のセットアップ前提条件が管理者権限を必要とするとき、メッセージ ボックスの [インストール] ボタンが、エンドユーザーに昇格された権限が必要であることを通知するシールド アイコンと共に表示されます。
- 再起動後、インストールが続行し、権限の昇格が必要な場合、続行メッセージ ボックスの [OK] ボタンがシールド アイコンと共に表示されます。権限の昇格が必要がない場合、シールド ボタンは表示されません。
- インストールに、ターゲット マシンにインストールしなければならないセットアップ前提条件が複数あり、このうち 1 つまたは複数のセットアップ前提条件が管理者権限を必要とする場合、UAC プロンプトが、一番最初のセットアップ前提条件がインストールされる前に表示されます。これにより、それぞれの前提条件のインストールごとに別々の UAC プロンプトを要求することなく、昇格された権限をすべての前提条件に対し

て使用できるようになる場合があります。ただし、セットアップ前提条件のインストールで再起動がある場合、管理者権限は失われることになり、これにより、残りの前提条件が管理者権限を必要とする場合、UAC プロンプトが表示されることがありますので注意してください。

機能前提条件では動作が若干異なります。インストールが、前提条件に関連付けられている機能をインストールする場合、ISInstallPrerequisites カスタム アクションが機能前提条件インストール モードで Setup.exe を再起動するときに UAC プロンプトが表示されます。これは、機能前提条件のいずれかが昇格された権限を必要とするかどうかに関わらず発生します。また、これは機能前提条件のインストールが必要かどうかを決定するために機能前提条件のいずれかが評価される前に発生します。機能前提条件インストールによって再起動が行われる場合、管理者権限は失われますのでご注意ください。再起動のあと、再び ReadyToInstall ダイアログが表示され、エンド ユーザーは残りのインストールを続行するために [インストール] ボタンをクリックする必要があります。この場合、ISInstallPrerequisites カスタム アクションが機能前提条件インストール モードで Setup.exe を再起動するとき、UAC プロンプトが再び表示されます。

- "管理者権限が必要" が [いいえ] に設定されている状態で、適切な権限をもたずに .msi パッケージがタスクを実行しようとすると、Windows Installer によって実行時エラーが表示される場合があります。
- 権限がインストールの終わりで昇格され、SetupCompleteSuccess ダイアログが製品を起動すると、昇格された権限が製品に引き継がれます。通常、昇格された権限を使用したアプリケーションの実行は推奨されていません。

サンプル シナリオ

次のセクションは、InstallShield で上記の設定の値を異なる組み合わせで使用した場合の例です。これらの図では、Windows Vista 以降が標準ユーザーまたは限られた権限を持つ管理者ユーザーに昇格された権限を要求しています。これらの例は、Windows Vista 以降のシステム上の UAC のデフォルト設定に基づいています。

例 1: UAC プロンプトが、管理者権限が必要な前提条件に表示される – .msi ファイルがアドバタイズされる

例 1 の図表は、セットアップ前提条件、機能前提条件、および .msi パッケージの [実行] シーケンスで昇格が必要なインストールを説明します。Windows Vista 以降では、セットアップ前提条件に UAC プロンプトを、また機能前提条件に別のもう 1 つの UAC プロンプトを表示します。

機能前提条件が含まれなかった場合、または機能前提条件ではなくセットアップ前提条件であった場合、2 番目の UAC プロンプト (図表では *UAC prompt #2* と名付けられています) は表示されません。これらの場合、セットアップ前提条件が昇格された権限を使ってインストールされた後、.msi パッケージが正しくアドバタイズされるため、UAC prompt #2 は必要ありません。

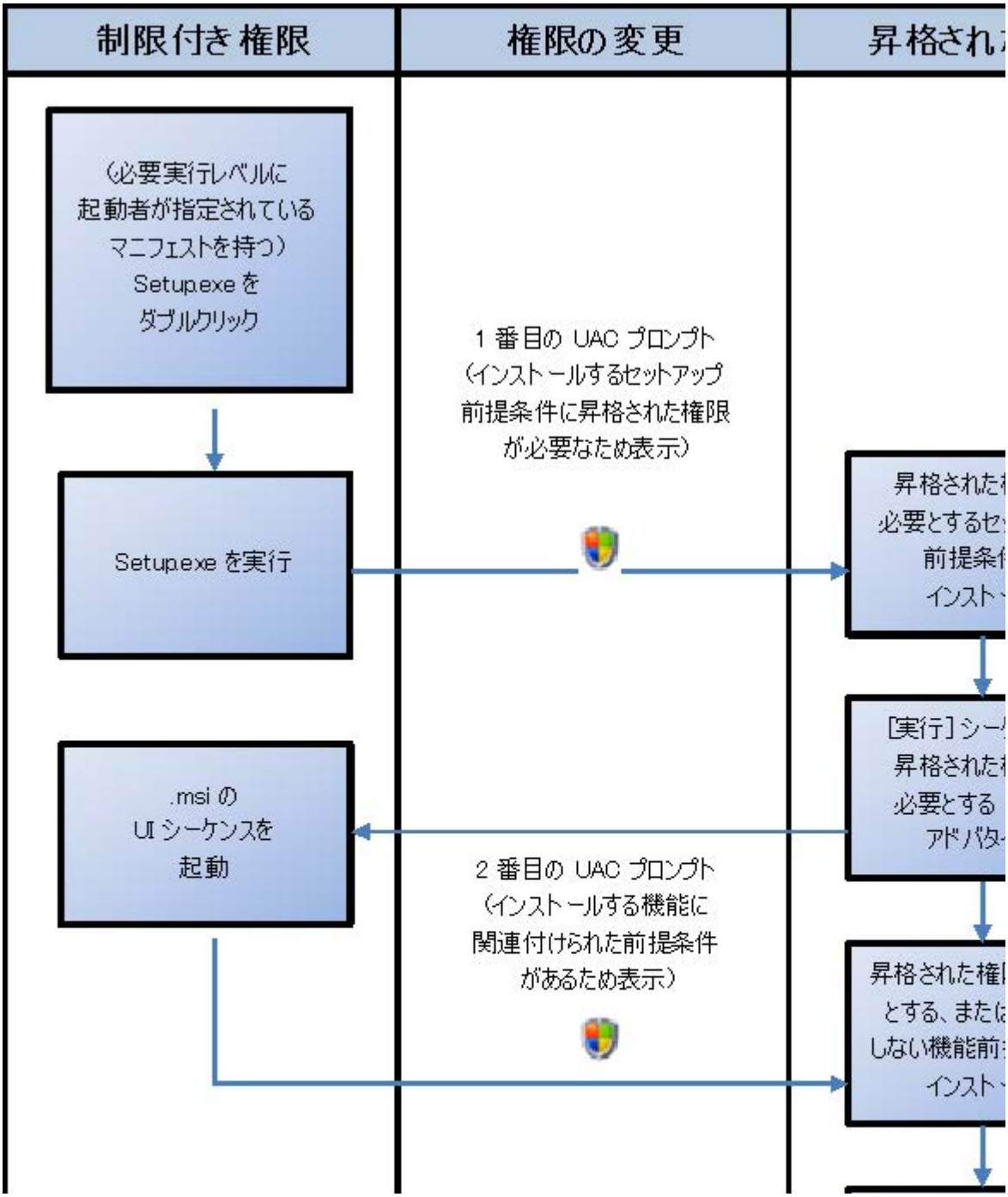


図 1: 例 1: "必要実行レベル" が [起動者] で、かつ .msi ファイルをアドバタイズするインストールのダイアグラム

例 2: UAC プロンプトが Setup.exe と、再起動後に管理者権限が必要な前提条件に表示される

例 2 の図表は、Setup.exe、2 つのセットアップ前提条件、機能前提条件、および .msi パッケージの [実行] シーケンスで昇格が必要なインストールを説明します。Setup.exe ファイルに、管理者を必要実行レベルとして指定するマニフェストがあるため、昇格された権限はインストールの各部分で使用されます。昇格された権限は再起動中に失われるため、2 つ目の UAC プロンプトが表示されます。

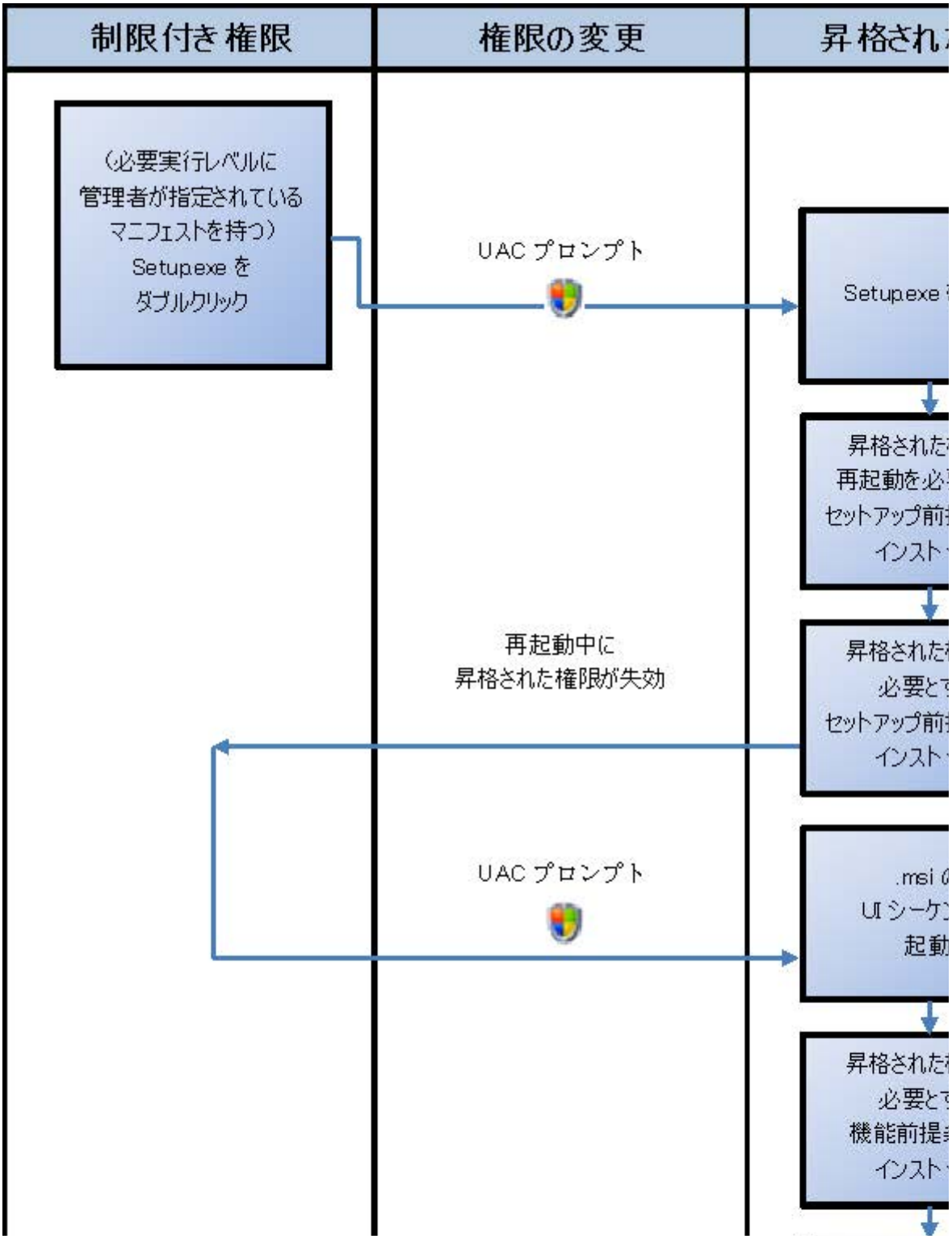


図 2: 例 2: "必要実行レベル" が [管理者] に設定されているインストールのダイアグラム

Using Windows Installer with UAC(Windows Installer ヘルプ ライブラリ)
Guidelines for Packages (Windows Installer ヘルプ ライブラリ)
Authoring Packages without the UAC Dialog Box (Windows Installer ヘルプ ライブラリ)

Installing a Package with Elevated Privileges for a Non-Admin (Windows Installer ヘルプ ライブラリ)

アプリケーションのディスクの使用量を設定する

Windows Installer はファイル コスティングと呼ばれるプロセスを使用して現在のインストールに必要な総ディスク容量を決定します。これには、インストールまたは削除されるファイル、レジストリ エントリ、ショートカット、およびインストールのその他のコンポーネントのコストが含まれます。

ファイル コスティング

ファイル コスティングは、いくつかのファイルが新規バージョンによって上書きされ、それによってファイル コストが削減されるという事実を考慮しています。これらの値は各ファイルがインストールまたは削除されるボリュームに依存し、コンポーネントのディレクトリ関連付けが変更される場合には再計算されます。

ファイルコストは、各コンポーネントについて、ローカルにインストールされるか、または CD などのソース メディアからインストールして実行するか、または削除されるかに応じて決定されます。

InstallShield では、アプリケーションのディスク使用量を設定することができます。これによって、ソース メディアから、またはローカルマシンからアプリケーションを実行するか、あるいは要求されたときにインストールするかを選択することができ、ファイルコストをコントロールすることが可能となります。ソースからアプリケーションを実行すると、アプリケーションの弾力性が拡張され、エンドユーザーのシステムのスペースが節約されます。

アプリケーションの復元性

Windows Installer がコンポーネントを提供できない状況で、該当するファイルが破損しているか、現在のファイルが使用可能なバージョンよりも古い場合には、Windows Installer 技術にを使ったアプリケーションによるコンポーネントの修復と再インストールの試行が可能です。

ソースの復元性

アプリケーションの復元性に加え、Windows Installer ではソースリストによってソースの復元性もサポートします。ソースリストには、必要時にアプリケーションをインストールするネットワークの場所、URL、CD などを含めることができます。管理者はグループ ポリシーエディターを使用してこの機能を無効にすることができます。

NSTALLDIR と TARGETDIR の違い

INSTALLDIR は、エンドユーザーによって起動される Setup.exe や .msi データベースなど、基本の MSI および InstallScript MSI のインストールで使用するメインの製品インストール ディレクトリを表します。

TARGETDIR は、InstallScript インストールまたは管理 Windows Installer ベースのインストール (/a コマンドラインスイッチを使って Setup.exe や MsiExec.exe をユーザーが実行した場合) のインストール ディレクトリを示します。

InstallScript MSI プロジェクトでは、InstallScript 変数の MSI_TARGETDIR が管理インストール時のインストール先を記録します。

現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンドユーザーが製品の現在のバージョンで同製品の将来のバージョンを上書きインストールするのを防止するためには、[アップグレード] ビューにメジャー アップグレード アイテムがあること、製品の現在のバージョンによって将来のバージョンが上書きインストールされないようにメジャー アップグレード アイテムが適切に構成されていること、および、製品に適切に構成、スケジュールされたタイプ 19 のカスタム アクションが含まれている必要があります。

新しい基本の MSI または InstallScript MSI プロジェクトを作成したとき、現在のインストールが将来のメジャー バージョンを上書きするのを防ぐためのサポートが自動的に追加されます:

- [アップグレード] ビューに、ISPreventDowngrade という名前のメジャー アップグレード アイテムが含まれています。

  [アップグレード コードを共有している製品] オプションは、メジャー アップグレード アイテムの [共通] タブで選択されています。[詳細] タブの "アップグレード コード" 設定の値は {000000000000-0000-0000-0000-00000000} です。リリースをビルドすると、InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、適切なアップグレード コード値を使用します。アップグレード コードに関する詳細は、「アップグレード コードを設定する」を参照してください。

  InstallShield は、このメジャー アップグレード アイテムの "検出プロパティ" 設定を ISFOUNDNEWERPRODUCTVERSION に設定し、他の設定を適切に構成します。

- [カスタム アクションとシーケンス] ビューには ISPreventDowngrade という名前のカスタム アクションが含まれています。ISPreventDowngrade は、エンドユーザーが現在のバージョンの製品で将来のメジャー バージョンを上書きインストールしようとしたとき Windows Installer が起動するタイプ 19 カスタム アクションです。

  InstallShield は、使用されているユーザー インターフェイスのレベルにかかわらず、適宜 Windows Installer が実行するように、ISPreventDowngrade カスタム アクションを [インストール] シーケンスの [ユーザー インターフェイス] シーケンスと [実行] シーケンスにスケジュールします。また、InstallShield は ISFOUNDNEWERPRODUCTVERSION をこのカスタム アクションの条件として使用します。

以下は、InstallShield 12 以前で作成し、InstallShield 2025 にアップグレードしたプロジェクトにこのサポートを手動で追加する方法です。

タスク

エンドユーザーが、製品の現在のバージョンで将来のメジャー バージョンを上書きインストールするためのサポートを手動で追加するには、以下の手順に従います:

1. **[アップグレード]** ビューを使用して、メジャー アップグレード アイテムをプロジェクトに追加します。
2. **[共通]** タブで、**[自分のアップグレード コードを使用する製品]** オプションを選択する。

3. メジャーアップグレードアイテムの **[詳細]** タブで設定を次のように構成します。

   a. **"最小バージョン"** 設定で、現在のプロジェクトに使用している製品バージョンを指定します。

      その代わりに、この設定に次の文字列を入力することもできます:

      *****ALL_VERSION*****

      その値はプレースホルダー値です。**"最小バージョン"** 設定で抽出された文字列を使うと、InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、現在開いているプロジェクトの製品バージョンを使用します。

   b. **"最大バージョン"** 設定は空白のままにします。この設定に値が表示されている場合、それを削除します。

   c. プロジェクトに複数の言語が含まれる場合、**"言語"** 設定で言語識別子を指定します。

   d. **"検出のみ"** 設定で、**[はい]** を選択します。他の [はい]/[いいえ] で答える設定は、すべて **[いいえ]** を選択します。

   e. **"検出プロパティ"** 設定で、次のような分かりやすい名前を入力します。

      FOUNDNEWERVERSION

4. プロジェクトで、エンド ユーザーが製品の現在のバージョンで将来のメジャー バージョンを上書きインストールしようと試みた場合に対応するために、タイプ 19 カスタム アクションを追加し、スケジュールします。

   a. **[カスタム アクションとシーケンス]** ビューで、**[カスタム アクション]** エクスプローラーを右クリックし、**[新しいエラー]** をクリックします。

   b. 新しいカスタム アクションを選択します。

   c. **"エラー メッセージ"** 設定で、エンド ユーザーが製品の現在のバージョンで将来のメジャー バージョンを上書きインストールしようを試みた場合に表示される、エラー メッセージ テキストを入力します。

   d. **"インストール UI シーケンス"** 設定と **"インストール実行シーケンス"** 設定で、**[次の後: FindRelatedProducts]** を選択します。

   e. **"インストール UI 条件"** と **"インストール実行条件"** 設定で、ステップ 3e で指定した値を入力します。

Preventing an Old Package from Installing Over a Newer Version (Windows Installer ヘルプ ライブラリ)

非管理者パッチのインストールを準備する

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者によるパッチは厳しい条件が満たされたときのみ利用することができます。たとえば、パッチが更新するベース インストールは、パッチ パッケージの署名に使用される証明書を含まなくてはなりません。満たされなければならない他の基準については、「非管理者パッチ」を参照してください。

タスク ***後で非管理者パッチによってアップデートが可能なベースインストールを作成するには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、デジタル署名の情報を構成するリリースをクリックします。

3. **[署名]** タブをクリックします。
4. デジタル署名情報を指定します。

必要な情報が自動的に、MsiDigitalCertificate テーブルおよび MsiPatchCertificate テーブルに追加されます。これによって、非管理者がインストールすることができるパッチの作成が可能になります。

ヒント: *[リリース] ビューでは、デジタル署名設定も利用できます。*

Windows Installer のデザインにより、パッケージの署名にある証明書を使用し、それとは異なる証明書で署名されたパッチを許可することができます。以下は、ベース インストールにパッチのための証明書を追加する方法です。

タスク ***追加のデジタル証明書を追加するには、以下の手順に従います:***

1. サンプル ファイルの署名には SignTool.exe 等のツールを使います。SignTool.exe は、Windows SDK に含まれています。
2. 元のインストール プロジェクトで、**ダイレクト エディター**を開きます。
3. **[テーブル]** エクスプローラーで、**MsiDigitalCertificate** をクリックします。
4. テーブルの最後の行をクリックして新規エントリを追加します。
5. **DigitalCertificate** フィールドに、証明書の一意の名前を入力します。
6. **CertData** フィールドをクリックします。**[バイナリ ストリームの編集]** ダイアログ ボックスが開きます。
7. **[ファイル名]** ボックスには、ステップ 1 で署名したファイルのパスと名前を入力します。参照ボタンをクリックしてファイルを指定することもできます。
8. **[OK]** をクリックします。
9. **[テーブル]** エクスプローラーで、**MsiPatchCertificate** をクリックします。
10. テーブルの最後の行をクリックして新規エントリを追加します。
11. **PatchCertificate** フィールドに、パッチ証明書の一意の名前を入力します。
12. **DigitalCertificate** フィールドで、ステップ 4 の MsiDigitalCertificate テーブルに作成したエントリを選択します。

メモ: *ベース インストールが期限切れのデジタル証明書で署名してあり、MsiDigitalCertificate および MsiPatchCertificate テーブルに以前の証明書と新しい証明書を含む中間パッチが適用されなかった場合、非管理者パッチは、ユーザー アカウント管理 (UAC) を表示します。*

User Account Control (UAC) Patching (Windows Installer ヘルプ ライブラリ)
MsiDigitalCertificate Table (Windows Installer ヘルプ ライブラリ)
MsiPatchCertificate Table (Windows Installer ヘルプ ライブラリ)

プラットフォームとプラットフォーム スイートの設定

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*

一部の設定は両方のプロジェクトの種類に適用しますが、他の一部はどちらか一方のプロジェクトの種類にのみ適用します。

InstallShield には、プラットフォーム、プラットフォーム スイートおよび言語の設定がいくつか含まれています。

プロジェクト レベルでプラットフォームを指定する

1 つのプロジェクト レベルの設定を使って、プロジェクトをサポートするプラットフォームを指定できます:

- **プラットフォーム フィルター**—この設定で、プロジェクトでコンポーネントまたはリリースにオペレーティング システム要件を選択するときの選択肢に加えるプラットフォームを指定します。一般的に、プロジェクト レベルのこの設定で表示されないプラットフォームがあるとき、プロジェクト内にある特定のコンポーネントまたはリリースをこのプラットフォームにターゲットするという指定ができなくなります。

  "プラットフォーム" 設定は InstallScript プロジェクトで提供されています。この設定にアクセスするには、[一般情報] ビューを開きます。[プロジェクトの設定] ダイアログ ボックスでもこの設定を構成することができます。その場合、[プロジェクト] メニューで [設定] をクリックして、[プラットフォーム] タブをクリックします。

メモ: *プロジェクト レベルでプラットフォームを指定しても、インストールを実行するときのターゲット システム要件は作成されません。InstallScript プロジェクトでターゲット システム要件を作成する場合、SYSINFO 構造を使用して、ターゲット システムのオペレーティング システムを識別します。*

InstallScript MSI プロジェクトにおけるターゲット システム要件の方法については、「プロジェクト アシスタントでオペレーティング システムを指定する」を参照してください。

コンポーネント レベルで、オペレーティング システムとプラットフォーム スイートを指定する

2 つのコンポーネント レベルの設定を利用して、コンポーネントのプラットフォーム情報を指定できます:

- **オペレーティング システム**—コンポーネントが 1 つまたは複数のオペレーティング システムに固有の場合、この設定を使用して、それらのオペレーティング システムを指定します。ターゲット マシンのオペレーティング システムがこの設定に指定されたオペレーティング システムの中にない場合、コンポーネントはインストールされません。

  デフォルトでは、コンポーネントはオペレーティング システムに依存しません。つまり、コンポーネントに特定のオペレーティング システム固有のデータがないことを意味します。

  "オペレーティング システム" 設定は、InstallScript プロジェクトと InstallScript MSI プロジェクトで使用できます。この設定にアクセスするには、[コンポーネント] ビューを開いて、コンポーネントを選択します。

- **プラットフォーム スイート**—コンポーネントが 1 つまたは複数のプラットフォーム スイートに固有の場合、この設定を使用して、それらのスイートを指定します。1 つ以上のスイートを指定する場合、コンポーネント

がインストールされるために、指定されたスイートすべて、または 1 つ以上のスイートがターゲット マシンに存在している必要があるかどうかを示すことができます。

デフォルトでは、コンポーネントはプラットフォーム スイートに依存しません。つまり、コンポーネントのデータ (ファイル、レジストリ エントリなど) はどれも特定のプラットフォーム スイートに固有ではありません。

この設定により、"オペレーティング システム" 設定よりもさらに細かくフィルターすることができます。"プラットフォームスイート" 設定は必要な場合のみ設定し、またアプリケーションが適切に機能するために必要なプラットフォーム スイートのみを確実に選択してください。たとえば、コンポーネントが WindowsXP の Home と Professional バージョンの両方にインストールされる必要がある場合、この設定の値を [スイート非依存] のままにしておくことができます。"オペレーティング システム" 設定に Windows XP を選択すると、両方のエディションが含まれます。

"プラットフォーム スイート" 設定は InstallScript プロジェクトで提供されています。この設定にアクセスするには、[コンポーネント] ビューを開いて、コンポーネントを選択します。

リリース レベルでプラットフォームを指定する

1 つのリリース レベルの設定を利用して、リリースのプラットフォーム情報を指定できます:

- **プラットフォーム**—リリースが 1 つまたは複数のプラットフォームに固有の場合、この設定を使用して、それらのプラットフォームを指定します。コンポーネントに指定されたプラットフォームが、この設定で選択されたプラットフォームのどれにも一致しない場合、そのコンポーネントはリリースに含まれません。

  この設定のデフォルト値は、[プロジェクト設定を使用] です。この値は、リリースがプロジェクト レベルで指定されたプラットフォームをサポートすることを示します。

  "プラットフォーム" 設定は InstallScript プロジェクトで提供されています。この設定にアクセスするには、[リリース] ビューを開いて、リリースを選択します。この設定は、リリース ウィザードの [プラットフォーム] パネルでも構成できます。

実行時にプラットフォームおよびプラットフォーム スイートのサポートを制御する (InstallScript プロジェクト)

InstallScript インストールの実行中に、**FeatureFilterOS** 関数を呼び出して、インストールがサポートするプラットフォームとプラットフォーム スイートを制御することができます。

OnFilterComponents イベント ハンドラーで、フレームワークは通常、適切なコンポーネントだけがインストールされるように、この関数をターゲット システムに一致するプラットフォームおよびプラットフォーム スイートで呼び出します。**FeatureFilterOS** を呼び出して、デフォルトの動作をオーバーライドし、指定したプラットフォームまたはプラットフォーム スイート基準に基づいてコンポーネントをインストールしたり、またはコンポーネントのインストールを防いだりすることができます。

詳細については、「FeatureFilterOS」を参照してください。

インストール情報を指定する

インストール プロジェクトを作成を開始するとき、まず最初にインストールの重要情報を指定する必要があります。これには、製品およびプロジェクト設定の指定と "プログラムの追加と削除" 設定の構成が含まれます。

このセクションには、次のトピックが含まれています:

- 一般的なプロジェクト設定を構成する
- 概要情報ストリーム データを入力する
- [プログラムの追加と削除] 情報を構成する
- 製品のソフトウェア識別タグを含める
- 製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める

一般的なプロジェクト設定を構成する

InstallShield は、プロジェクト設定を単一インストール プロジェクトファイル (.ism ファイル) に保管します。このファイルには、プロジェクトについてのすべての情報が格納されています。[一般情報] ビューを使うと、作成者名、プロジェクトがサポートする言語、および記入するコメントなど、インストール プロジェクトについての基本情報を編集することができます。

[一般情報] ビューではまた、製品名、製品コード (GUID)、およびバージョン番号といった一般的な製品情報も構成できます。製品は、インストール プロジェクトの構成中で最上位のものです。インストールは機能とコンポーネントに分割でき、これらが製品のサブセットになります。インストールに複数の機能とコンポーネントがある場合でも、製品は 1 つしかありません。

各プロジェクトとプロジェクト設定についての詳細は、「[一般情報] ビュー」を参照してください。

このセクションには、次のトピックが含まれています:

- InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存する
- 製品名の指定
- 製品バージョンを指定する
- InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおける製品バージョン番号
- Windows Installer ベースのプロジェクトで製品コードを設定する
- InstallScript ベースのプロジェクトで製品コードを設定する
- アップグレード コードを設定する
- 製品の条件を設定する
- デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定
- ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護
- Windows Installer インストールをログ記録するかどうかを指定する
- InstallScript インストールを複数回実行する
- "メンテナンスの有効化" 設定を構成する

- InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い

InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存する

InstallShield では、.ism プロジェクト ファイルを XML またはバイナリ形式で保存することができます。

タスク ***プロジェクト ファイルの形式を指定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. ***"プロジェクト ファイルの形式"*** 設定で、適切なオプションを選択します: 選択可能なオプションは以下のとおりです:
   - **バイナリ** – プロジェクト ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。
   - **XML** – プロジェクト ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。

メモ: *InstallShield プロジェクト ファイル (.ism) を XML またはバイナリ形式で保存した時、その拡張子は保持されます。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクト (.issuite) は、常に XML ファイルとして保存されます。

製品名の指定

[一般情報] ビューの "製品名" 設定に製品の名前を入力します。ここに入力する名前は、インストールを作成する製品の名前です。この値は、プロジェクト全体を通して、次のような名前に使用されます。

- プロジェクトの場所の下にあるソースファイルフォルダーの名前
- 基本の MSI および InstallScript MSI プロジェクトの場合: InstallShield がデフォルトでビルドする Windows Installer パッケージ (.msi file) の名前
- 実行時ダイアログ
- Windows ロゴ要件に従って登録される情報レジストリキー情報値は以下の場所にあり、エンド ユーザーが製品を変更または削除するのに使用できる [プログラムの追加と削除] で使用されます。

  HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥*ProductCode*

入力値はソース ファイルのパスに組み込まれるため、製品名には次のいずれの文字も含めることはできません: ¥ / : * ? " < > | –

ヒント: *InstallScript または InstallScript MSI プロジェクトでは、UNINSTALL_DISPLAYNAME 変数を使用して、[プログラムの追加と削除] に異なる製品表示名を指定することができます。*

基本の MSI および InstallScript MSI プロジェクトの場合、.msi パッケージ ファイルの名前に製品名以外を使用するとき、[リリース] ビューにある製品構成で [全般] タブの "MSI パッケージ ファイル名" 設定を使って、異なる .msi ファイル名を指定できます。製品のアップデートを行うためのマイナー アップグレードまたはスモール アップデートをリリースできるようにしたい場合は、インストールの以前のバージョンと最新バージョンの .msi パッケージ名が同じでなくてはなりません。.msi ファイル名が異なる場合にマイナー アップグレードまたはスモール アップデートを実行しようとすると、Windows Installer ランタイム エラー 1316 の原因となる可能性があります。

***注意 -** 製品名にアンパサンド (&) を含める場合、エンドユーザーダイアログで名前を正しく表示するには、2 つのアンパサンド (&&) を使用する必要があります。たとえば New & Improved Product と表示する場合、製品名を New && Improved Product と入力します。*

製品バージョンを指定する

製品のバージョン番号を指定するとき、必ず有効な製品バージョンを入力してください。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。

実行時に、インストールが、インストールされる製品のバージョン番号を登録します。完全なバージョン文字列が [プログラムの追加と削除] に表示されます。製品バージョン番号は、その一部をインストール エンジンがアップグレードの適用を判断するために使用するため、重要です。

製品バージョンを構成するには、[一般情報] ビューを使用します。この製品バージョンは、たとえば基本の MSI プロジェクトの [リリース] ビューにある製品構成など、InstallShield の別の領域でオーバーライドすることができます。また、基本の MSI または InstallScript MSI プロジェクトの [アップグレード] ビューなど InstallShield の別の領域で、アップグレードがターゲットとするバージョン番号を指定することもできます。

製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。

***プロジェクト-** 基本の MSI、InstallScript、および InstallScript MSI プロジェクトの場合―リリースに* Setup.exe *が含まれる場合、指定した製品バージョンが* Setup.exe *の [プロパティ] ダイアログ ボックスに表示されます。詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。*

InstallScript および InstallScript オブジェクト プロジェクトの場合、値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)

ProductVersion Property (Windows Installer ヘルプ ライブラリ)

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおける製品バージョン番号

***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*

- *InstallScript オブジェクト*

デフォルトで、InstallScript プロジェクトではパックされた DWORD 形式のバージョン番号を使用します。つまり 4 バイトの値で、最初のバイトはメジャー バージョン、2 番目のバイトはマイナー バージョン、最後の 2 つのバイトはビルド番号です。パックされた DWORDS が入力され、メジャー、マイナー、ビルド形式 (例、1.2.3 または 255.255.65535) で表示されます。インストール中の製品のバージョン番号を登録するデフォルトのスクリプト コードではパックされた DWORD バージョン番号が想定され、アップデートのインストール中にインストール済み製品のバージョン番号との比較に使用されます。

バージョン番号の中に パックされた DWORD 形式ではないものが存在する場合、次のセクションの内容に従ってスクリプト コードを変更する必要があります。

既にインストールされている製品のバージョン番号

デフォルトの **OnSetUpdateMode** イベント ハンドラー関数コードは、インストール中の製品のバージョン番号および、([リリース] プロパティ シートで指定されている要領で) アップデートを適用できるバージョン番号を、システム変数 *IFX_INSTALLED_VERSION* と比較します。インストールは、自動的に、このシステム変数の値を、アプリケーションアンインストールレジストリキーのVersion 値にあるデータに等しい文字列に初期化することを試みます。その値が存在しない場合、または、そのデータがパックされた DWORD 形式ではない場合、*IFX_INSTALLED_VERSION* の値はヌル文字列 ("") になり、その場合、エラー メッセージがデフォルトの **OnSetUpdateMode** コードによって表示され、インストールは中止 (abort) されます。これを解決する方法の 1 つとして、システムでバージョン情報をチェックし、*IFX_INSTALLED_VERSION* に適切なパックされた DWORD 値を設定するコードを挿入するという方法があります。たとえば、以前のインストールで HKEY_LOCAL_MACHINE¥Software¥MyCompany¥MyProduct の下に MyVersion の値にバージョン文字列が格納されていると、次のようなコードを挿入できます。

```
if (IFX_INSTALLED_VERSION="") then
    /* 登録されているバージョン情報を取得します。*/
    RegDBSetDefaultRoot ( HKEY_LOCAL_MACHINE );
    RegDBGetKeyValueEx( "Software¥¥MyCompany¥¥MyProduct",
        "MyVersion", REGDB_STRING, szVersionString, nSize );

    /*  IFX_INSTALLED_VERSION に値を割り当てます。*/
    switch (szVersionString)
        /* 登録されているバージョン文字列の "A" は、既存のバージョン番号の 1.0.0 に対応します。*/
        case "A":
            IFX_INSTALLED_VERSION = "1.0.0";
        /* 登録されているバージョン文字列の "B" は、既存のバージョン番号の 1.1.0 に対応します。*/
        case "B":
            IFX_INSTALLED_VERSION = "1.1.0";
        /* 不在のバージョン文字列は、既存のバージョン番号の 0.0.0 に対応します。*/
        デフォルト:
            IFX_INSTALLED_VERSION = "0.0.0";
    endswitch;
endif;
```

インストール中の製品のバージョン番号

デフォルトの **OnMoveData** イベント ハンドラー関数コードは、**RegDBSetVersion** を呼び出して、[一般情報] ビューにに入力された製品バージョン番号を使い、それをターゲット システムのレジストリに入力します。**RegDBSetVersion** は、製品バージョンをはパックされた DWORD 形式だと仮定します。パックされた DWORD 形式ではない製品バージョンを登録する場合、OnMoveData の **RegDBSetVersion** の呼び出しを次のようなコードで置き換えます:

```
/* セットアップは自動的にシステム変数
IFX_PRODUCT_VERSION の値を
[一般情報] ビューの "製品バージョン" 設定に入力された製品バージョンに初期化します。*/
RegDBSetItem(REGDB_UNINSTALL_VERSION, IFX_PRODUCT_VERSION );
RegDBSetItem(REGDB_UNINSTALL_DISPLAY_VERSION, IFX_PRODUCT_VERSION );
```

RegDBSetItem は、**MaintenanceStart** の後に呼び出される必要があります。

Windows Installer ベースのプロジェクトで製品コードを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

製品コードは製品を識別する固有の文字列です。インストールは実行時に製品コードを使用して、その製品が既にインストール済みであるかどうかを判断します。

製品を一意に識別する GUID を入力するか、[一般情報] ビューの "製品コード" 設定にある **[新しい GUID の生成]** ボタン ([...]) をクリックして、自動的に新しい GUID を生成します。インストールが、この GUID を実行時に登録します。

注意 - *この製品コードは製品を一意に識別するため、製品のリリースを配布してからはこのコードを変更しないことをお勧めします。*

InstallScript ベースのプロジェクトで製品コードを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

製品コードは製品を一意に識別する GUID 文字列です。インストールは実行時に製品コードを使用して、その製品が既にインストール済みであるかどうかを判断します。

製品を一意に識別する GUID を入力するか、[一般情報] ビューの "製品コード" 設定にある **[新しい GUID の生成]** ボタン ([...]) をクリックして、自動的に新しい GUID を生成します。インストールが、この GUID を実行時に登録します。

注意 - *プロジェクト GUID は、アンインストールまたはメンテナンスを元のインストールと関連付けるのに使用されます。新しい GUID は、(既存のプロジェクトのコピーを含む) 作成した各新規プロジェクトに対して自動的に生成されます。プロジェクトの GUID を変更すると、以前の GUID を回復することはできません。このため、プロジェクトの GUID の変更は、通常必要なく、変更する場合は注意が必要です。*

アップグレード コードを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *QuickPatch*
- *トランスフォーム*

アップグレード コードとは、製品が所属する製品ファミリを識別する一意な GUID です。ほとんどの場合、アップグレード コードは、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があり、これを使って、Windows Installer がインストール済みの製品の関連バージョンを検索します。

新しい製品の GUID を構成するには、[一般情報] ビューの "アップグレード コード" 設定を使用します。このアップグレード コードの値は、InstallShield の別の領域でオーバーライドすることが可能です。たとえば、[リリース] ビューの製品構成、または [リリース] ビュー内の製品構成にある [複数インスタンス] タブで定義されている各インスタンス。

製品の新しいバージョンのアップグレードを作成中の場合、[一般情報] ビューで構成されたアップグレード コード (または、前述の InstallShield のその他の領域でオーバーライドされたアップグレード コード) をアップグレード プロジェクトまたは QuickPatch プロジェクトの [アップグレード] ビューで指定する GUID として使用します。これらのアップグレード コードが一致するとき、アップグレードまたはパッチがベース バージョンをターゲットすることができるため、Windows Installer が必要に応じてアップデートすることが可能となります。

製品の条件を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト アシスタントの [インストール要件] ページでは、一般的に用いられるターゲット システムのインストール要件を指定できます。たとえば、アプリケーションを正しく実行するために特定の OS が必要な場合、[インストール要件] ページで指定することができます。

InstallShield ではまた、製品がインストールされる前に Windows Installer が評価しなくてはならない、独自のカスタム条件を定義できます。[一般情報] ビューと、プロジェクト アシスタントの [インストール要件] ページで作成する条件は、製品全体に適用します。ターゲット システム上で 1 つ以上の条件が False 評価された場合、インストールが終了してエラー メッセージが表示されます。

たとえば、製品を正しく実行するために RAM が 64 MB 必要な場合、[製品条件ビルだ] ダイアログ ボックスを使って、条件を作成できます。実行時、Windows Installer がターゲット システムをチェックして、インストールされている RAM のサイズを判断します。64 MB 未満の場合、またはその他の製品条件が False の場合、Windows Installer がエラーを表示して、インストールが終了します。

ヒント・ Windows Installer が製品の起動条件を評価する順番を保証することはできません。条件が評価される順番を制御する必要がある場合、[カスタム アクションとシーケンス] ビューで各条件に対してエラー カスタム アクションを作成して、それらを適切な順序にスケジュールすることができます。

タスク ***製品のインストールにカスタム起動条件を作成するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"インストール条件"** 設定をクリックしてから、省略記号ボタン (...) をクリックします。**[製品条件ビルダー]** ダイアログ ボックスが開きます。
3. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
4. 以下のいずれかを実行します。
   - **[条件]** 列で、起動条件を入力します。
   - **[プロパティ]** リスト、**[演算子]** リスト、および **[追加]** ボタンを使って、条件ステートメントをビルドします:
     a. **[プロパティ]** リストで、プロパティを選択してから、**[追加]** ボタンをクリックします。InstallShield は、プロパティを **[条件]** 列に追加します。
     b. 条件ステートメントに演算子を含める場合、**[演算子]** リストから演算子を選択してから、**[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。
     c. 条件ステートメントが値を含む場合、"条件" フィールドをダブルクリックしてから END を押し、挿入ポイントを条件ステートメントの終わりに移動して、値を入力します。
5. **[メッセージ]** 列に、条件が False 評価された場合に表示するエラー メッセージを入力します。

   この列に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定をダブルクリックしてから省略記号ボタン (...) をクリックして、既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

   ***メモ・** [メッセージ] 列で指定したテキストを表示する Windows Installer ダイアログは、¥r (改行)、¥n (新しい行) または ¥t (タブ) 等のエスケープ シーケンスを認識しません。*
6. **[OK]** をクリックします。

***重要・** InstallShield は、基本の条件検証を行いますが、条件ステートメントが予定通りの結果を評価することを確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクトの INSTALLDIR プロパティは、製品に含まれるすべてのファイル用のデフォルト フォルダーとして機能します。その値の割り当て先は、Windows Installer フォルダーのプロパティである INSTALLDIR となり、これはデフォルト機能およびコンポーネントのインストール先フォルダーの役割を果たします。

Windows ロゴ: *Windows ロゴ要件によると、ターゲット システムの言語にかかわらず、製品のファイルのデフォルトのインストール先を Program Files のサブフォルダー、またはエンド ユーザー のアプリケーション データ フォルダーにする必要があります。ProgramFilesFolder を製品のインストール先フォルダー設定の親フォルダーとして使用すると、ファイルは正しい場所にインストールされます。INSTALLDIR 設定のデフォルト値は次の通りです:*

[ProgramFilesFolder]*会社名¥製品名*

タスク ***製品の INSTALLDIR プロパティを設定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. ビルトイン Windows Installer ディレクトリをパスの一部として使用するには、**INSTALLDIR** 設定で、省略記号ボタン (...) をクリックします。.**[INSTALLDIR の設定]** ダイアログ ボックスが開きます。**[インストール先ディレクトリ]** ボックスで、インストール先のフォルダーを選択します。

   その代わりに、**INSTALLDIR** 設定に手動でパスを入力することもできます。

メモ: *[INSTALLDIR の設定] ダイアログ ボックスで新しいフォルダーを選択すると、INSTALLDIR 設定の値がオーバーライドされます。たとえば [ProgramFilesFolder]My Company¥Program のように、円記号でサブフォルダーを分割することによって、フォルダー プロパティのサブフォルダーを指定できます。*

その他のインストール先フォルダーについての考慮事項 (Windows Installer ベースのプロジェクトのみ)

機能の "リモート インストール" 設定が [ソースを優先] (または、サブ機能の親機能が [ソースを優先] に設定されている場合は [親を優先]) に設定すると、製品の INSTALLDIR プロパティに関わらず、機能のファイルはターゲット システムにインストールされません。

各機能およびコンポーネントには、"インストール先" 設定が含まれています。機能の "インストール先" 設定によって、製品の "インストール先フォルダー" 設定がオーバーライドされ、コンポーネントの "インストール先" 設定 によって、機能の "インストール先フォルダー" 設定がオーバーライドされます。したがって、製品のすべてのファイルをデフォルトで製品のインストール先フォルダーにインストールする場合は、すべての機能およびコンポーネントの "インストール先" 設定に **[INSTALLDIR]** を入力します。

INSTALLDIR などのインストールフォルダープロパティを使用している場合、*デフォルト*値を指定していることになります。エンド ユーザーは、Msiexec.exe 起動時に、コマンドラインでプロパティを設定したり、CustomSetup ダイアログで機能に対して新しいインストール先フォルダーを選択することにより、この値を変更できます。

ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護

InstallShield では、ロックダウン環境において製品を実行するエンド ユーザー向けに、ファイル、フォルダー、レジストリ キー、および Windows サービスを保護するためのいくつかの方法が提供されています:

- **従来型の Windows Installer 処理** — Windows Installer ベースのプロジェクトでは、実行時にファイル、フォルダー、およびレジストリ キーのアクセス許可を設定できるビルトイン Windows Installer サポートを使用できます。このオプションを使うと、InstallShield は .msi データベースの LockPermissions テーブルに製品のアクセス許可情報を格納します。

  この種類のアクセス許可処理を新しい Windows Installer 処理と組み合わせることはできません。MsiLockPermissionsEx テーブルおよび LockPermissions テーブルを含むリリースをビルドしようとすると、ビルド エラー -7207 が発生します。

- **新しい Windows Installer 処理** — Windows Installer ベースのプロジェクトでは、実行時にファイル、フォルダー、レジストリ キー、および Windows サービスのアクセス許可を設定できる最新の Windows Installer サポートを使用できます。このオプションを使うと、InstallShield は .msi データベースの MsiLockPermissionsEx テーブルに製品のアクセス許可情報を格納します。

  このオプションを使用するには、ターゲット システム上に Windows Installer 5 以降が必要です。以前のバージョンの Windows Installer は、この種類の処理に関する設定を無視します。

  この種類のアクセス許可処理を従来型の Windows Installer 処理と組み合わせることはできません。MsiLockPermissionsEx テーブルおよび LockPermissions テーブルを含むリリースをビルドしようとすると、ビルド エラー -7207 が発生します。

- **カスタム InstallShield 処理** — Windows Installer ベースのプロジェクトでは、実行時にアクセス許可を設定するためのカスタムサポートを選択できます。このオプションを使うと、InstallShield は .msi データベースのカスタム ISLockPermissions テーブルに製品のアクセス許可情報を格納します。InstallShield はまた、プロジェクトにカスタム アクションを追加します。

- **InstallScript 関数、SetObjectPermissions** — InstallScript イベントおよび InstallScript カスタム アクションで **SetObjectPermissions** 関数を使って、実行時にアクセス許可を設定できます。

これらの方法を使って、特定グループおよびユーザーにファイル、フォルダー、またはレジストリ キーのアクセス許可を割り当てることができます。たとえば、管理者グループに特定のファイルについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。新しい [Windows Installer 処理] オプションを使うと、Windows サービスのアクセス許可を割り当てることもできます。

使用するオプションを決定する

次のテーブルでは、アクセス許可を設定するため様々な方法を比較します。

テーブル 1 • ロックダウン環境においてオブジェクトをセキュリティ保護する異なる方法の比較

比較カテゴリ	利用できるサポートについて
プロジェクトの種類	• **従来型の Windows Installer 処理、新しい Windows Instller 処理、およびカスタム InstallShield 処理** — 基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト タイプで利用できます。 • **SetObjectPermissions 関数** — InstallScript プロジェクトおよび InstallScript MSI プロジェクト タイプの InstallScript イベントで利用できます。 基本の MSI、DIM、InstallScript MSI、および マージ モジュール プロジェクトでは、InstallScript カスタム アクションを使って利用することもできます。
広く知られているセキュリティ識別子 (SID)	• **従来型の Windows Installer 処理** — 限られた数の SID (Administrators、Everyone) をサポートします。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — 多くの SID (Administrators、Authenticated Users、Creator Owner、Everyone、Guests、Interactive、Local Service、Local System、Network Service、Power Users、Remote Desktop Users、および Users) をサポートします。
翻訳された SID の名前	• **従来型の Windows Installer 処理** — 翻訳された SID の名前をサポートしません。翻訳された名前を使用すると、インストールは失敗します。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートされている、よく知られる SID (Administrators、Authenticated Users、Creator Owner、Everyone、Guests、Interactive、Local Service、Local System、Network Service、Power Users、Remote Desktop Users、および Users) の翻訳されたすべての SID の名前をサポートします。
特定のアクセス許可を拒否できる機能	• **従来型の Windows Installer 処理** — サポートされていません。この処理を使って、特定のアクセス許可を設定できます。アクセス許可を拒否することはできません。したがって、ユーザーにファイルの読み取り専用アクセスを付与することができます。ただし、ユーザーが読み取り専用アクセスを所有することを防ぐことはできません。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートします。これらのオプションを使って、指定するアクセス許可を特定のユーザーまたはグループが所持することを拒否するかどうかを指定できます。

テーブル 1・ロックダウン環境においてオブジェクトをセキュリティ保護する異なる方法の比較 (続き.)

比較カテゴリ	利用できるサポートについて
既存するアクセス許可に対する影響	• **従来型の Windows Installer 処理** — 既存のアクセス許可を削除することができます。たとえば、ターゲット システム上のフォルダーに Everyone ユーザーのアクセス許可が既に設定されている場合で、インストールが Administrators ユーザーのアクセス許可を設定するとき、このオプションを使って Administrators ユーザー用のアクセス許可を設定できます。ただし、既存する Everyone のアクセス許可は削除されます。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — これらのオプションを使って、ターゲット システムに既存するファイル、フォルダー、またはレジストリ キーに、アクセス許可を追加できます。このとき、そのオブジェクトに既存するアクセス許可は削除されません。たとえば、ターゲット システム上のフォルダーに Everyone ユーザーのアクセス許可が既に設定されている場合で、インストールが Administrators ユーザーのアクセス許可を設定するとき、これらのオプションを使って既存する Everyone ユーザーのアクセス許可を削除することなく、Administrators ユーザー用のアクセス許可を設定できます。
子オブジェクト (サブフォルダー、ファイル、およびサブキー) にアクセス許可を反映させる機能	• **従来型の Windows Installer 処理** — サポートされていません。フォルダー内のサブフォルダーまたはファイル (またはレジストリ キーの下にあるサブ キー) にアクセス許可を構成する場合、ターゲット システム上で作成された親は、自動的に子のアクセス許可を継承します。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートします。これらのオプションを使って、フォルダー (またはレジストリ キー) のアクセス許可を構成し、そのフォルダーのサブフォルダーおよびファイル (またはレジストリ キーのサブキー) すべてに同じアクセス許可を適用するかどうかを指定できます。
インストールの一部としてインストールされないオブジェクトのアクセス許可を設定できる機能	• **従来型の Windows Installer 処理、新しい Windows Installer 処理、およびカスタム InstallShield 処理** — サポートしません。 • **SetObjectPermissions 関数** — サポートされています。インストールの一部としてインストールされた、またはターゲット システムに既存するファイル、フォルダー、またはレジストリ キーを、アクセス許可でセキュリティ保護することができます。
インストール中に作成された新しいユーザーに対するアクセス許可を設定できる機能	• **従来型の Windows Installer 処理** — サポートされていません。 • **新しい Windows Installer 処理、カスタム InstallShield 処理、および SetObjectPermissions 関数** — サポートします。インストール中に新しいユーザーが作成される場合、そのユーザーに対するアクセス許可を構成できます。

[カスタム InstallShield 処理] オプションまたは [従来型の Windows Installer 処理] オプションについての詳細

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、カスタム InstallShield 処理または Windows Installer 処理のどちらを使用するかを指定しなくてはなりません。詳しくは、「プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する」をご覧ください。

いずれかのオプションを使ってファイルまたはフォルダーのアクセス許可を設定する方法については、「ファイルとフォルダーのアクセス許可を構成する」を参照してください。これらのオプションの 1 つを使ってレジストリ キーのアクセス許可を設定する方法については、「レジストリ キーのアクセス許可を構成する」を参照してください。

新しい [Windows Installer 処理] オプションについての詳細

サービスに新しい Windows Installer 処理オプションを使用するには、サービスをプロジェクトに追加してから、その設定を構成します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。

ファイル、フォルダー、またはレジストリ キーに新しい Windows Installer 処理オプションを使用するには、[ダイレクト エディター] ビューの MsiLockPermissionsEx テーブルを使います。

InstallScript 関数、SetObjectPermissions についての詳細

SetObjectPermissions 関数についての詳細は、「SetObjectPermissions」を参照してください。

LockPermissions (Windows Installer ヘルプ ライブラリ)
MsiLockPermissionsEx (Windows Installer ヘルプ ライブラリ)

プロジェクトで、ロックダウン環境でのアクセス許可タイプを選択する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield には、ロックダウン環境にあるエンド ユーザー向けにファイル、フォルダー、およびレジストリ キーのアクセス許可をインストールがどのように構成するかを指定できる、プロジェクト全体に反映される設定があります。

タスク ***プロジェクトに対して、ロックダウン環境でのアクセス許可タイプを選択するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"ロックダウンの設定方法"** 設定で、適切なオプションを選択します:
   - **カスタム InstallShield 処理**—InstallShield は、プロジェクトにカスタム テーブルとカスタム アクションを追加して、ターゲット システム上のアクセス許可を設定します。これがデフォルトの値です。
   - **従来型の Windows Installer 処理** — InstallShield は、.msi データベースの LockPermissions テーブルを使って、製品のアクセス許可情報を格納します。

これらの 2 つのオプションについての詳細な比較情報は、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

プロジェクトで次にファイル、フォルダー、またはレジストリ キーのアクセス許可を構成するとき、InstallShield は選択済みのロックダウン環境でのアクセス許可を使用します。

- [従来型の Windows Installer 処理] オプションを選択した場合、InstallShield はプロジェクトで LockPermissions テーブルを使用します。
- [カスタム InstallShield 処理] オプションを選択した場合、InstallShield はプロジェクトで ISLockPermissions テーブルを使用し、さらに ISLockPermissionsCost と ISLockPermissionsInstall カスタム アクションをプロジェクトに追加します。

"ロックダウンの設定方法" 設定の値を変更する時に、プロジェクトにファイル、フォルダー、またはレジストリ キーのアクセス許可が既に含まれている場合、InstallShield は適切なテーブルにアクセス許可データを移行するかどうかを問い合わせるメッセージ ボックスを表示します。データの移行を選択すると、選択済みのオプションに対応するテーブルにデータが移動されます。[カスタム InstallShield 処理] オプションから [従来型の Windows Installer 処理] オプションに切り替えると、プロジェクトから ISLockPermissionsCost と ISLockPermissionsInstall カスタム アクションも削除されます。

LockPermissions (Windows Installer ヘルプ ライブラリ)

Windows Installer インストールをログ記録するかどうかを指定する

InstallShield では、Windows Installer 4.0 以降がインストールのログ記録を行うかどうかを、プロジェクト全体にわたって指定することができます。また、ログ記録されるメッセージの種類をカスタマイズすることもできます。

タスク ***Windows Installer 4.0 以降で、プロジェクト全体のログ情報を指定するには、次の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **[MSI ログの作成]** 設定をクリックして、省略記号ボタン (...) ボタンをクリックします。**[Windows Installer 4.0 以降のログ記録オプション]** ダイアログ ボックスが開きます。
3. 適切なオプションを選択します。カスタム オプションを選択する場合、MsiLogging 値を入力します。

   MsiLogging 値の有効なパラメーターのリストは、Windows Installer ヘルプ ライブラリの MsiLogging Property を参照してください。
4. **[OK]** をクリックします。

結果は、[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックスで行った選択によって異なります。

- [いいえ] を選択すると、ログ記録は行なわれません。これがデフォルトの値です。
- [はい] を選択すると、InstallShield は MsiLogging プロパティにデフォルト値である *voicewarmup* を挿入します。Windows Installer 4.0 が搭載されたターゲット システム上でインストールが実行されると、次の処理が行なわれます。
  - *voicewarmup* のデフォルト ログ モードに従って、インストーラーがログ ファイルを作成します。
  - インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。
  - SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。
- [カスタム] を選択すると、InstallShield は、このボックスに指定された値を MsiLogging プロパティに挿入します。Windows Installer 4.0 が搭載されたターゲット システム上でインストールが実行されると、次の処理が行なわれます。
  - このボックスに指定されたカスタム値に基づいて、インストーラーがログ ファイルを作成します。
  - インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。
  - SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。

以前のバージョンの Windows Installer は MsiLogging 設定を無視します。以前のバージョンの Windows Installer を実行中のシステム上で表示される実行時ダイアログでは、**[Windows Installer ログを表示]** チェック ボックスは表示されません。

重要: *MsiLogFileLocation プロパティは、読み取り専用のため、ログ ファイルの場所の設定および変更には使用できません。*

InstallScript インストールを複数回実行する

InstallScript プロジェクトの種類には、ターゲットマシンに製品がインストールされている場合、およびエンドユーザーがインストールを再試行した場合のインストール動作に関するいくつかのオプションがあります。

テーブル 2 ・ メンテナンス エクスペリエンスの設定

メンテナンス エクスペリエンス	メンテナンス UI 表示のタイミング	[プログラムの追加と削除] で作成されるエントリの数
標準	インストールを再試行した時	1
複数インスタンス	インストールが [プログラムの追加と削除] を通して実行された時	複数—各インストールごとに個別のエントリ
アンインストールまたはメンテナンスはなし	チェックしない	なし

これらのオプションは、[一般情報] ビューの "メンテナンス エクスペリエンス" 設定から選択できます。

デフォルトの動作 (標準オプション) では、エンドユーザーがマシン上でインストールを再実行すると、メンテナンス ユーザー インターフェイスが表示されます。

複数インスタンスのオプションを使うと、エンドユーザーはインストールをメンテナンスインストールではなく初回インストールとして複数回にわたって再試行することが可能となりす。このオプションを選択すると、デフォルトで、エンドユーザーがインストール実行ごとに異なる場所に製品をインストールすることができるようになります。インストールを初回インストールとして実行するたびに、直接選択またはセットアップの種類の選択に関わらず、選択されたコンポーネントが毎回インストールされます。

マシン上に製品の 1 つ以上のインスタンスが存在するとき、エンドユーザーがユーザー インターフェイスを使って複数インスタンス インストールを再実行した場合、インストールは [正規の製品が検出されました] ダイアログ ボックスを表示します。このダイアログでは、エンドユーザーはアップデートを適用するインスタンスを選択することができます。ただし、エンドユーザーがインストールをサイレントで再実行した場合、インストールは [正規の製品が検出されました] ダイアログ ボックスを抑制して、マシン上に新しいインスタンスを作成します。

プロジェクト: *基本の MSI プロジェクトにおける複数インスタンスのサポートに関する詳細は、「製品の複数のインスタンスをインストールする」を参照してください。*

"メンテナンスの有効化" 設定を構成する

プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

[一般情報] ビューにある "ンテナンスを有効にする" 設定を使って、製品が既にインストールされているシステム上で、エンド ユーザーがインストールを再度実行したときに、完全なメンテナンス ユーザー インターフェイス (UI) またはアンインストール UI を表示するかどうかを指定します。

次のテーブルは、"メンテナンスの有効化" 設定で選択できる設定です。

テーブル 3 · "メンテナンスの有効化" の設定のオプション

オプション	説明
はい	/removeonly コマンドライン パラメーターが Setup.exe に渡されない限り、エンド ユーザーがインストールを再度実行したとき、システム変数 REMOVEONLY は FALSE に設定され、標準メンテナンス UI が表示されます。

テーブル 3 - "メンテナンスの有効化" の設定のオプション (続き.)

オプション	説明
いいえ	エンド ユーザーがインストールを再度実行したとき、システム変数 REMOVEONLY は TRUE に設定され、アンインストール UI が表示されます。 *メモ- "ンテナンスを有効にする" 設定に [いいえ] を選択すると、OnUninstall イベント ハンドラの呼び出しが行われません。OnUninstall イベント ハンドラの呼び出しを行う場合は、*Setup.exe *に /*uninst *コマンドライン パラメーターを使用しなくてはなりません。これは、InstallScript UI スタイルが InstallScript エンジンを外部 UI ハンドラとして使用する従来型のスタイルの場合にのみ適用します。InstallScript UI スタイル が InstallScript エンジンを埋め込み UI ハンドラとして使用する新しいスタイルの場合、*/uninst *コマンドライン パラメーターは、サポートされていません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。*

メモ- InstallScript MSI インストールでは、デフォルトで、コントロール パネルの [プログラムの追加と削除] で [変更] ボタンと [削除] ボタンが別々に表示されます。[変更] ボタンをクリックするとメンテナンス UI が、また [削除] ボタンをクリックするとアンインストール UI が常に表示されます。

"メンテナンスの有効化" 設定は、エンド ユーザーが [プログラムの追加と削除] で製品のエントリに表示されている [変更] ボタンをクリックして製品のメンテナンスを開始した場合の動作には、効果を持ちません。

エンド ユーザーが [プログラムの追加と削除] 内からメンテナンスを実行できないようにするには、[一般情報] ビューの "変更ボタンを無効にする" 設定で [はい] を選択します。

InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い

***プロジェクト-** この情報は、InstallScript MSI プロジェクトに適用します。*

InstallScript MSI インストールは 2 つの異なるインストーラー エンジンを使用します:

- Windows Instaler エンジンは、.msi パッケージの標準 [実行] シーケンスを実行します。これは、通常ターゲット システムを変更するシーケンスです。
- InstallScript エンジンは、インストールのカスタム ユーザー インターフェイス (UI) ハンドラーとしての役割を果します。また、InstallScript コードを実行します。UI に InstallScript エンジンを使用する際の利点は、高度にカスタマイズされたランタイム ダイアログのサポートが提供されていることです。

従来、Windows Installer では、外部カスタム UI ハンドラーのみがサポートされていたため、InstallScript MSI インストールでは、常に Setup.exe セットアップランチャーを使用する必要がありました。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。

Windows Installer 4.5 では、埋め込みカスタム UI ハンドラーのサポートが追加されました。InstallScript エンジンが .msi パッケージ内に埋め込まれている場合、InstallScript MSI インストールでは、Setup.exe セットアップランチャーが必要なくなり、エンドユーザーは .msi パッケージを起動してインストールを開始することができます。このケースでは、.msi パッケージに、インストールの UI で InstallScript エンジンを起動するにあたって Windows Installer が知る必要がある情報が含まれています。

したがって、InstallShield では、InstallScript MSI インストールのUI に、次の 2 つの異なるスタイルが提供されています:

- 従来型スタイル (Setup.exe セットアップランチャーが必要) – InstallScript MSI インストールで、InstallScript エンジンを外部 UI ハンドラーとして使用することができます。これがデフォルトのスタイルになります。
- 新しいスタイル (ターゲット マシンで Windows Installer 4.5 が必要) – InstallScript MSI インストールで、InstallScript エンジンを埋め込み UI ハンドラーとして使用することができます。このスタイルでは、InstallShield が InstallScript エンジンを .msi パッケージ内に埋め込みます。このオプションには、いくつかの制限事項があります。

次のセクションは、これらの 2 つのスタイルに関する詳しい情報です。これらの情報を参考にして、ニースに最も適したスタイルを判別してください。

従来型スタイル (外部 UI ハンドラーとしての InstallScript エンジン)

初回インストールにおける従来型スタイルの InstallScript MSI パッケージの一般的な流れは、次の通りです:

1. InstallScript MSI が、.msi パッケージを開きます。
2. ExecuteAction を除く、[インストール]- UI シーケンスにあるすべてのアクションが実行されます。アクションは昇順に実行されます。
3. InstallScript エンジンは、自分で内部コンポーネントのコスト計算を行って、インストール内にあるすべての機能とコンポーネントに必要な空き容量を判別します。
4. InstallScript エンジンが、ISSetup.dll に含まれているコンパイル済みのスクリプトを起動します。.msi パッケージのインストールの前に実行されるイベントのイベント シーケンスは、次の通りです:
    a. OnBegin
    b. O*nXxx*UIBefore (Xxx は、First、Maint、Admin、Patch、Resume などが入ります)
    c. OnGeneratingMsiScript
    d. OnMoving
    e. OnFeaturesInstalling (機能の installing と uninstalling イベントは、すべてこの時点で実行されます。)
    f. OnInstallFilesActionBefore
    g. OnGeneratedMsiScript
5. InstallScript エンジンは、**MsiInstallProduct** を通して .msi パッケージ インストールを開きます。次の要因によって、InstallScript エンジンが **MsiInstallProduct** を通して渡すコマンドラインが判別されます: インストール モード (例、初回インストール、メンテナンス モード、マイナー アップグレード、パッチなど)、内部機能の選択、および現在のプロパティの値。
6. .msi パッケージ インストールが成功した場合、InstallScript エンジンは、セカンダリ アンインストール キー (InstallShield_{ProductCode}) をマシンに書き込みし、次のイベントを起動します:
    a. OnInstallFilesActionAfter

b. OnFeaturesInstalled (機能の installed と uninstalled イベントはすべてこの時点で実行されます。)

c. OnMoved

d. On*Xxx*UIAfter (*Xxx* は、First、Maint、Admin、Patch、Resume などが入ります)

e. OnEnd

7. スクリプトが完了すると、InstallScript エンジンは InstallScript ログをマシンに書き込みします。

8. InstallScript エンジンがシャットダウンします。

新しいスタイル (埋め込み UI ハンドラーとしての InstallScript エンジン)

InstallScript MSI の埋め込み UI のサポートは、従来型の InstallScript MSI インストールと同じ一般的なフローを従おうとします。ただし、一部のフローは、発生しないか、または、必要ありません。初回インストールにおける新しいスタイルの InstallScript MSI パッケージの一般的な流れは、次の通りです:

1. .msi パッケージは、直接、または Setup.exe (Setup.exe は、このあと、基本の MSI インストールで同じ動作をする Msiexec.exe を起動します) によって起動されます。

2. Windows Installer エンジンによって、MsiEmbeddedUI テーブルにあるすべてのファイルが抽出され、DLL 内の埋め込み UI ハンドラーの属性を含む "埋め込み UI の初期化" 関数を呼び出します。(InstallScript MSI ケースでは、これは ISSetup.dll です。)

3. InstallScript エンジンは、ISSetupFilesExtract アクションが存在し、InstallUISequence で条件が True に評価された場合、そのアクションを初期化し、手動で起動します。

4. InstallScript エンジンは、Directory テーブル内にあるすべてのエントリの解決を手動で解決します。

5. InstallScript エンジンは、自分で内部コンポーネントのコスト計算を行って、インストール内にあるすべての機能とコンポーネントに必要な空き容量を判別します。

6. InstallScript エンジンが、ISSetup.dll に含まれているコンパイル済みのスクリプトを起動します。.msi パッケージのインストールの前に実行されるイベントのイベント シーケンスは、次の通りです:

   a. OnBegin

   b. O*nXxx*UIBefore (*Xxx* は、First、Maint、Admin、Patch、Resume などが入ります)

   c. OnGeneratingMsiScript

   d. OnMoving

   e. OnFeaturesInstalling (機能の installing と uninstalling イベントは、すべてこの時点で実行されます。)

   f. OnInstallFilesActionBefore

   g. OnGeneratedMsiScript

7. InstallScript エンジンは、コントロールを Windows Installer エンジンに返します。そのあと、Windows Installer エンジンは、インストールの [実行] シーケンスの実行を開始します。

8. .msi パッケージ インストールが成功した場合、Windows Installer エンジンは ISSetup.dll に呼び戻します。そのあと、ISSetup.dll によって、次のイベントが起動されます。

   a. OnInstallFilesActionAfter

   b. OnFeaturesInstalled (機能の installed と uninstalled イベントはすべてこの時点で実行されます。)

   c. OnMoved

d. On*Xxx*UIAfter *(Xxx* は、First、Maint、Admin、Patch、Resume などが入ります)

e. OnEnd

9. InstallScript エンジンがシャットダウンし、コントロールを Windows Installer に返します。

新しいスタイルの InstallScript MSI インストールでは、[インストール]- UI シーケンスは実行されません。[インストール]- UI シーケンスにシーケンスされているカスタム アクションがある場合、これらのアクションは、/qb または /qn と共に起動された基本の MSI と同じ理由で、実行されません。

InstallScript エンジンは、InstallScript イベントからシステムに行われた変更をログ記録しません。

従来型の InstallScript MSI インストールでサポートされているほとんどのコマンドライン パラメーターは、新しいスタイルの InstallScript MSI インストールでもサポートされています。エンドユーザーが .msi パッケージを直接実行した場合、これらのパラメーターを ISSCRIPTCMDLINE プロパティに渡すことができません。

従来型スタイル (外部 UI ハンドラーとしての InstallScript エンジン) の制限

アンインストール可能なパッチ サポート

従来型のスタイルは、アンインストール可能なパッチを作成するためのサポートを含みません。

新しいスタイル (埋め込み UI ハンドラーとしての InstallScript エンジン) の制限事項と既知の問題

新しいスタイルを検討している場合、次の点に注意してください。

Windows Installer 4.5 の要件

新しいスタイルには、ターゲット マシン上に Windows Installer 4.5 が必要です。存在していないとき、Windows Installer 4.5 がインストールされるように、Windows Installer 4.5 の InstallShield 前提条件をプロジェクトに追加することができます。InstallShield 前提条件をインストールに含めると、Setup.exe セットアップランチャーを使用する必要があります。詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。Windows Installer 4.5 は、Windows の初期のバージョンの一部にはインストールできませんので注意してください。

実行時に Windows Installer 4.5 がターゲット システムにインストールされていなく、かつ、インストールが Setup.exe で起動された場合、エラー 1713 が表示され、インストールの実行に Windows Installer 4.5 が必要であることが通知されます。インストールは、エンドユーザーがこのエラー メッセージを閉じたあと中止されます。.msi パッケージが直接起動された場合、パッケージが Setup.exe ファイルから起動する必要があることを通知するエラーが表示されます。

アップグレードとパッチのサポート

InstallShield には、アップグレードが従来型のスタイルでインストールされた製品を更新する場合、新しいスタイルを使用するアップグレードを作成するためのビルトイン サポートは含まれていません。したがって、従来型スタイルを使って、製品のリリースをビルドしてから、後でアップグレードを作成する場合、そのアップグレードに対して従来型のスタイルを使用することを検討してください。これは、全ての種類のアップグレード (メジャー アップグレード、マイナー アップグレード、スモール アップデート) に適用します。

パッチがターゲットとする以前のセットアップが従来型スタイルを使用している場合、パッチも従来型スタイルを使用しなくてはなりません。また、QuickPatch プロジェクトは、元のインストールの UI スタイルを保持します。つまり、元のインストールが従来型スタイルを使用している場合、QuickPatch も従来型スタイルを使用します。元のインストールが新しいスタイルを使用している場合、QuickPatch も新しいスタイルを使用します。

メジャー アップグレード シナリオの回避策として、新しいスタイルのメジャー アップグレード インストールに、レジストリから以前の製品のアンインストール文字列を読み込ませ、新しいバージョンをインストールする前に、以前のバージョンの製品をアンインストールする InstallScript 関数 LaunchApplication を使ってそのインストールを起動することもできます。場合により、スクリプトで、**LaunchApplication** を使用する前に、LAAW_SHELLEXECUTEVERB を **runas** に設定する必要があります。

MsiDoAction

.msi パッケージのインストールの前に実行されるイベントの 1 つから **MsiDoAction** 関数を呼び出そうとした場合、エラーが返されます。このエラーは、Windows Installer が InstallScript エンジンに渡すハンドラーが原因で発生します。ハンドルは、.msi パッケージ内にある標準およびカスタム アクションを実行できません。この関数は、必要な場合、InstallScript カスタム アクションから関数を呼び出すことができます。

MsiGetTargetPath と MsiSetTargetPath

Windows Installer の **MsiGetTargetPath** と **MsiSetTargetPath** 関数は、これらがイベント ドリブン スクリプトにある場合、新しいスタイルで適切に動作しません。これらの関数は、Windows Installer のコスト計算アクションによって初期化されるディレクトリ マネージャーに依存します。新しいスタイルの InstallScript MSI インストールでは、Windows Installer のコスト計算は実行されません。したがって、**MsiGetTargetPath** はパス情報を返すことができず、**MsiSetTargetPath** も要求されたターゲット パスを設定することができません。両方のケースで、これらの関数が呼び出された場合、Windows Installer はメッセージを詳細ログにログ記録します。

インストール パスを更新するには、FeatureSetTarget を使用します。

MsiGetTargetPath と **MsiSetTargetPath** は、[インストール]–[実行] シーケンスで CostFinalize の後にシーケンスされた InstallScript カスタム アクションを通して呼び出すことができます。

FeatureTransferData と ComponentTransferData

InstallScript 関数 **FeatureTransferData** と **ComponentTransferData** は、InstallScript MSI インストールの新しいスタイルでは定義されていないため、使用できません。

programÉendprogram

新しいスタイルの InstallScript MSI インストールでは、手続きのスクリプト (programÉendprogram ブロックがあるスクリプト) を使用できません。このケースでは、プログラム関数は呼び出されません。イベント ドリブン スクリプトは、インストールの動作をカスタマイズするために使用します。

/uninst

新しいスタイルの InstallScript MSI インストールでは、/uninst コマンドライン パラメーターはサポートされていません。

BATCH_INSTALL

BATCH_INSTALL 変数を設定して再起動を試みても、正しく動作しません。再起動を実行するには、ScheduleReboot アクション または REBOOT プロパティを利用します。

Reboot のサポート

新しいスタイルの InstallScript MSI インストールでの再起動は、InstallScript ベースのインストールではなく、基本の MSI インストールと同じ要領で処理されます (InstallScript ベースのインストールでは、再起動のあと、インストールが続行し、**OnRebooted** イベント ハンドラーが呼び出されます)。

新しいスタイルの InstallScript MSI インストールは、基本の MSI インストールと同じ要領で再起動を処理するように作成されます。つまり、インストールは、スケジュール済みの再起動が発生した後実行されず、**OnRebooted** イベント ハンドラーが呼び出されません。

Msiexec.exe /x (for Uninstallation)

エンドユーザーが、Msiexec.exe /x {ProductCode} ステートメントを使ってコマンドラインから製品をアンインストールしようと試みたとき、アンインストールが成功することがあります。ただし、アンインストールの終わりのほうで、Windows Installer によってエラー ダイアログが表示されます。エラー ダイアログでは、Windows Installer サービスにアクセスできなかったというメッセージが表示されます。

コマンドラインからアンインストールを実行する場合、次のいずれかのメソッドを実行することが現在推奨されています:

msiexec.exe /i {ProductCode} REMOVE=ALL

msiexec.exe /x {ProductCode} /qn

これらの例では、Windows Installer エラーはトリガーされません。

推奨事項

新しいスタイルの InstallScript MSI インストールでは、UI 機能を管理者権限なしに実行することができます。したがって、次の点が推奨されます (および、一般的に、すべての Windows Installer ベースのインストールに適用します):

- 管理者権限が、InstallScript イベントから提供されると仮定することはできません。
- システムに行う必要がある変更は、[インストール]–[実行] シーケンスでカスタム アクションで行う必要があります。InstallScript カスタムアクションを使用することができます。権限が不足しないように、カスタム アクションは、システム コンテキストで "スクリプト内実行" 設定が遅延実行になっている必要があります。
- システムを変更するカスタム アクションには、インストールが失敗した場合、システムを以前の状態に復元する対応するロールバック アクションが必要です。
- システムを変更するカスタム アクションにも、システムからインストールされているアイテムを削除するためにアンインストールで実行する適切なアクションが必要です。
- InstallScript イベントから変更がシステムに行われた場合、変更は InstallScript エンジンによってログされないため、アンインストールでこれらのリソースをクリーンアップする追加のスクリプト コードが必要になります。

InstallScript を使って、実行時に 2 つのスタイルを識別する

INSTALLSCRIPTMSIEEUI 変数を利用して、異なる UI スタイルに異なる実行時の動作を生成する単一スクリプトを作成することができます。詳しくは、「INSTALLSCRIPTMSIEEUI」を参照してください。

MsiDoAction (Windows Installer ヘルプ ライブラリ)
MsiGetTargetPath (Windows Installer ヘルプ ライブラリ)
MsiSetTargetPath (Windows Installer ヘルプ ライブラリ)

InstallScript MSI インストールの InstallScript ユーザー インターフェイス タイプを指定する

プロジェクト: この情報は、InstallScript MSI プロジェクトに適用します。

InstallScript プロジェクトには、2 つの異なる種類の InstallScript ユーザー インターフェイス (UI) があります。制限事項を含む、これら 2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

タスク ***InstallScript MSI プロジェクトの InstallScript ユーザー インターフェイス タイプを指定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"InstallScript ユーザー インターフェイスのタイプ"** 設定で、適切なオプションを選択します。
   - **従来のスタイル (Setup.exe が必要)** — InstallScript エンジンを InstallScript MSI インストールの 外部 UI ハンドラーとして使用する場合は、このオプションを選択します。このスタイルを使用する場合は、インストールに Setup.exe セットアップランチャーを必ず含めてください。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。

     この設定では、このオプションがデフォルトで設定されています。
   - **新しいスタイル (Windows Installer 4.5 が必要)** — InstallScript エンジンを InstallScript MSI インストールの埋め込み UI ハンドラーとして使用する場合は、このオプションを選択します。このスタイルでは、InstallShield が InstallScript エンジンを .msi パッケージ内に埋め込みます。Windows Installer は InstallScript エンジンを呼び出して UI を表示します。Windows Instaler はまた、.msi パッケージの [実行] シーケンスを実行します。

     このオプションには、ターゲット マシン上に Windows Installer 4.5 が必要です。また、いくつかの制限事項があるため、このスタイルを使用する場合は綿密な計画が必要です。

概要情報ストリーム データを入力する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

概要情報ストリームの設定の一部は、DIM プロジェクトで使用できます。このプロジェクト タイプでは、これら設定の値は実行時に使用または表示されません。このプロジェクト タイプにおけるこれらの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。

Windows Installer データベース (.msi ファイルと .msm ファイル) は COM 構造化ストレージとして実装されており、通常 COM 構造化ストレージ ファイルには [概要情報ストリーム] が入っています。概要情報ストリームには、会社や、インストール中のソフトウェアに関する情報が含まれます。ファイルの概要情報の表示方法についての詳細は、「[概要情報ストリーム パネル] へのアクセス」を参照してください。

概要情報ストリームの各設定の説明については、「[一般情報] ビュー」を参照してください。

[概要情報ストリーム パネル] へのアクセス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***.msi ファイルまたは .msm ファイルの概要情報にアクセスするには、以下の手順をに従います:***

1. .msi ファイルまたは .msm ファイルを右クリックしてから、**[プロパティ]** を選択します。**[プロパティ]** ダイアログ ボックスが開きます。
2. [**概要**] タブをクリックして、概要情報を表示します。

ヒント: *[一般情報] ビューで、インストールの [プロパティ] ダイアログ ボックスに表示する情報を入力します。*

概要情報ストリームのプロパティを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

タスク ***プロジェクトの概要情報を指定するには、次の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. ビュー内で **[概要情報ストリーム]** 領域に移動します。
3. 必要に応じて設定の値を変更します。

ビルド時の概要情報ストリームのコメントを設定する

ビルド時にコメントを入力できるオプションもあります。MSBuild.exe タスクの SummaryInfoComments パラメーターを使って、次の例のようにビルド番号を含めるなど、ビルド時に **[概要情報ストリーム コメント]** を設定することができます。

MSBuild.exe c:¥installers¥Setup.sln /Property:SummaryInfoComments="ここにコメントを入力"

"テンプレート概要" プロパティを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[一般情報] ビューにある "テンプレートの概要" 設定を使って、プロセッサの種類と言語の基本的な条件を設定できます。ターゲット システムがこの要件を満たさない場合、インストーラーがエラー メッセージを表示してインストールを終了します。

ヒント: *[リリース] ビューで、製品構成について "テンプレートの概要" 設定を構成することもできます。[リリース] ビューに入力された値は、[一般情報] ビューに設定された値を上書きします。*

構文

プロセッサの種類を最初にリストしてから、セットアップのデフォルト言語に続きます。言語に 4 桁の言語 ID か、言語に中立にするにはゼロ (0) を入力します。

- 2 つのカテゴリは、間にセミコロンを入れて区切ります。
- インストールが複数の製品をサポートする場合で、言語カテゴリーに複数のエントリ (たとえば英語は 1033、ドイツ語は 1031) を含めるには、それらをコンマで区切ります。

注意・*インストールが 64 ビットマシンを対象にしている場合、Intel と Intel64 の両方をこのフィールドに入力しないでください。両方を入力するとインストール エラーが発生します。さらに、Intel64 を入力した場合、インストールは 32 ビットオペレーティング システムでは作動しません。*

有効なプロセッサ値は、次のとおりです:

- Intel
- Intel64
- x64

プロセッサタイプと言語に基づいた条件の設定

インストールが Intel プロセッサを持つ英語システムでのみ実行する場合、"テンプレートの概要" プロパティで次の構文を使用します。

Intel;1033

プロセッサの種類の条件が一致しないと、インストーラーはエラーメッセージを表示して終了します。

製品が x64 プロセッサを搭載した英語およびドイツ語システム上で実行する場合は、次のように入力します:

x64;1033,1031

パッケージを言語非依存に指定する場合、**Intel;** または **Intel;0** という構文を使用します。

言語のみに基づいた条件の設定

言語のみに基づいて条件を設定するには、セミコロンを入力して、セットアップのデフォルトの言語を指定します。

;1036

最初の部分が空の場合、ターゲット プロセッサが Intel であることを意味します。言語が指定されていない場合、InstallShield は [概要情報ストリーム] の リリース言語を提供します。

Template Summary Property (Windows Installer ヘルプ ライブラリ)

[プログラムの追加と削除] 情報を構成する

[一般情報] ビューの [プログラムの追加と削除] 領域にある設定は、[プログラムの追加と削除] ツールに表示される情報です。

特定の "プログラムの追加と削除" 設定に値を入力しなかった場合、サンプルの値が灰色テキストで表示されます。この値は例として表示されているため、最終的に Windows Installer データベースには含まれません。

[一般情報] ビューを使うと、[プログラムの追加と削除] を通じてエンド ユーザーに表示される情報の多くを設定することができます。

プロジェクト・*この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

プロジェクト: *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合 – 指定したほとんどの値は、ARP で始まる名前 (例、ARPPRODUCTICON または ARPHELPLINK など) を持つ Windows Installer のプロパティに格納されます。唯一の例外は "発行元" 設定です。この値は会社名と共に設定され、[製造業者] プロパティに格納されます。*

基本の MSI プロジェクトの場合 – 製品が [プログラムの追加と削除] に表示されないようにするには、プロパティ マネージャーで Windows Installer プロパティ ARPSYSTEMCOMPONENT *を 1 に設定します。このプロパティを設定すると、[プログラムの追加と削除] に製品を表示しないように抑制するだけです。エンド ユーザーは、コマンド ラインを使ってインストールをメンテナンス モードで実行することで、製品を削除することができます。*

InstallScript MSI プロジェクトの場合—製品が [プログラムの追加と削除] に表示されないようにするには、[リリース] ビューの "[追加/削除] パネルエントリを隠す" 設定で [はい] を選択します。

Readme ファイルを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[一般情報] ビューの README 設定では、製品の Readme ファイルの名前を入力します。Windows の一部のバージョンでは、この情報は、[プログラムの追加と削除] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。また、有効な URL を指定して、インターネット上の Readme ファイルのリンクを表示することもできます。

プロジェクト: *Windows Installer ベースのプロジェクト (基本の MSI または InstallScript MSI プロジェクト) では、Readme ファイルをインストールの一部としてインストールする時に、パスを適切に表示するための特別なフォーマットが必要です。次のセクションを参照してください。InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクトでは、ハードコード化されたパスまたは URLを入力するか、山かっこで括ったシステム変数の相対パス (たとえば、<TARGETDIR>¥Readme.txt) を指定します。このデータはデフォルトの OnMoveData イベント ハンドラー関数によってターゲット システムのレジストリに書き込まれます。*

基本の MSI および InstallScript MSI プロジェクトで Readme ファイル パスを表示する

Readme ファイルのパスを直接入力するか、または URL を入力します。ただし、[MyDirectory] のようなフォルダー名、または [$MyComponent] のようなコンポーネント名は解決されません。つまり、[サポート情報] ダイアログ ボックスの Readme 値が [INSTALLDIR]Readme.txt や [$MyComp]Readme.htm のように表示されます。このプロパティは Windows Installer プロパティ ARPREADME の初期値を単に設定するだけなので、パスは解決されません。したがって、書式設定された文字列が Windows Installer プロパティで解決されることはありません。

InstallShield の解決法はカスタム アクションを使用してARPREADME を README 設定 に入力した値で設定することです。SetARPReadme カスタム アクションで値を設定することにより、パスはインストール時に解決されます。たとえば、Readme.doc がコンポーネント Help_Files のファイルと仮定します。この場合、README 設定に **[$Help_Files]Readme.doc** を入力します。Help_Files が C:¥Program Files¥MyCompany¥MyProduct¥Help にインストールされると仮定した場合、ターゲット システムの [プログラムの追加と削除] で表示されるパスは、C:¥Program Files¥MyCompany¥MyProduct¥Help¥Readme.doc となります。

コンポーネント名の代わりに、ファイルキーを使用することもできます。実行時に解決できないため、Directory テーブルからフォルダーを使用しないでください。ファイルキーはコンポーネント名ほど信頼できません。これは、ファイルが 動的にリンクされている、またはリリースでパッチの最適化を使用している場合、ファイル名が変更される可能性があるからです。上記の例を続けると、Readme.doc が F501_Readme.doc のファイルキーを持っていれば、README 設定に **[#F501_Readme.doc]** と入力します。.同じフォルダーにインストールされた場合、ターゲット システム上で [プログラムの追加と削除] は次のパスを表示します:

C:¥Program Files¥MyCompany¥MyProduct¥Help¥Readme.doc

カスタム アクション SetARPReadme は、[一般情報] ビューの README 設定に文字列があり、[インストール] – [実行] シーケンスに CostFinalize アクションが含まれている場合に、毎回追加されます。SetARPReadme は、[インストール] の [実行] シーケンスと [ユーザー インターフェイス] シーケンスで CostFinalize のすぐ後に挿入されます。

製品のソフトウェア識別タグを含める

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ISO/IEC 19770-2 は、ソフトウェア識別タグを作成するための国際規格です。ソフトウェア識別タグは、製品名、製品エディション、パブリッシャーなど、ソフトウェアに関する説明的な情報を含むサイズが小さい XML ベースのファイルです。ソフトウェア資産管理ツールは、企業でインストールされているソフトウェアについての正確なアプリケーション ID を提供する目的で、タグ内のデータを収集するツールです。

ソフトウェア識別タグ機能は、業界標準として現在進化しつつあるもので、この機能により、独立系ソフトウェア ベンダーは、顧客に対して、ソフトウェア資産管理およびライセンス最適化イニシアチブに有用な、より適切な情報を提供することができる、より洗練されたアプリケーションを作ることができるようになります。製品のインストール パッケージに識別タグを持たせることで、顧客は、インストールした製品の内部的使用状況を監視できるツールが使えるようになります。これにより、顧客が入手済み製品のライセンスの数を把握、管理および最適化することができるようになります。

適切にタグを作成するためには、[一般情報] ビューにある ″製品名″ や ″製品バージョン″ 設定といった基本設定を構成する必要があります。また、[一般情報] ビューにある識別タグ関連の設定も構成しなくてはなりません。

タスク ***インストールにソフトウェア識別タグを含めるには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. このビュー内の **[ソフトウェア識別タグ]** 領域で、必要に応じて設定の値を変更します。

   "ソフトウェア識別タグの使用" 設定を使って、インストールにタグを含めるかどうかを指定できます。デフォルト値の **[はい]** を選択してから、必要に応じて **[ソフトウェア識別タグ]** 領域にあるその他の設定を構成します。

プロジェクトでタグ機能を使用する場合、InstallShield が作成する 2 つの新しいコンポーネントにタグが追加され、これらのコンポーネントが製品機能の 1 つに関連付けられます。コンポーネントは、以下のとおりです:

- INSTALLDIR をインストール先とする ISO19770_LocalTag
- CommonAppDataFolder をインストール先とする ISO19770_SystemTag

これらのコンポーネントをプロジェクト内の別の機能に関連付ける場合は、[セットアップのデザイン] ビューを使います。詳細については、「コンポーネントと機能の関連付け」を参照してください。

ビルド時に以下の条件が True 評価された場合、InstallShield がビルドするインストールにソフトウェア識別タグが追加されます:

- [一般情報] ビューの "ソフトウェア識別タグの使用" 設定に、デフォルト値の [はい] が選択されている。
- [一般情報] ビューの "一意な ID""タグ作成者"、および "タグ作成者 ID" 設定に値が設定されている。

タグ機能が有効である場合に、前述の 3 つのタグ識別設定のうち 1 つでも値が不足している場合、InstallShield はビルド警告を生成して、リリースにタグが含まれなかったことを通知します。この警告を解決するためには、必要に応じて [一般情報] ビューの [ソフトウェア識別タグ] 領域にある設定を構成してください。

プロジェクトにソフトウェア ID タグを含めて [リリース] ビューで .pfx ファイルを使ってリリースに署名を行うことを設定すると、InstallShield がビルド時にタグをデジタル署名します。タグ ファイルに署名するためには、.NET Framework 3.5 以降をビルドマシンにインストールする必要があります。

製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Microsoft System Center 2012 Configuration Manager は、IT 管理者が、エンタープライズ全体において、効率良くアプリケーションを管理および配信できるようにするためのソリューションです。ユーザー側に重点を置いたアプローチがとられているこのソリューションでは、それぞれのアプリケーションを、必要な依存アプリケーションと共に、最も適切なフォーマットで様々なデバイス上に配信できるように定義することが可能です。

配置メタデータを正確に識別することは、アプリケーションを Configuration Manager アプリケーション モデルに移行する上で必須です。InstallShield では、ソフトウェア識別タグを使って、アプリケーションのアプリケーション モデル メタデータの一部を指定できる機能が提供されています。AdminStudio のユーザーが AdminStudio のアプリケーション カタログにパッケージをインポートしたとき、パッケージの要素が解析されて、検出方法、依存

関係、要件などの配置データ、およびソフトウェア識別タグ内の情報が確認されます。AdminStudio では、この情報を Microsoft System Center 2012 Configuration Manager に公開する前に、確認およびテスト用として、ユーザーに提供しています。

タスク ***System Center Configuration Manager のアプリケーション モデル データをインストールに含めるには、以下の手順に従います:***

1. 製品に必ずソフトウェア識別タグを含めてください。詳細については、「製品のソフトウェア識別タグを含める」を参照してください。
2. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
3. **"SCCM アプリ モデル データの追加"** 設定で **[はい]** を選択します。
4. **"SCCM アプリ モデル データの追加"** 設定の下のサブ設定を適宜構成します。詳細については、「[一般情報] ビューの設定」を参照してください。

プロジェクトにタグ機能およびアプリケーション モデル データを含めると、InstallShield が作成する 2 つの新しいコンポーネントにタグがアプリケーション モデル データと共に追加され、これらのコンポーネントが製品機能の 1 つに関連付けられます。コンポーネントは、以下のとおりです:

- INSTALLDIR をインストール先とする ISO19770_LocalTag
- CommonAppDataFolder をインストール先とする ISO19770_SystemTag

これらのコンポーネントをプロジェクト内の別の機能に関連付ける場合は、[セットアップのデザイン] ビューを使います。詳細については、「コンポーネントと機能の関連付け」を参照してください。

ビルド時に以下の条件が True 評価された場合、InstallShield がビルドするインストールにソフトウェア識別タグが追加されます。また、タグにアプリケーション モデル データも含められます。

- [一般情報] ビューの "ソフトウェア識別タグの使用" 設定に、デフォルト値の [はい] が選択されている。
- [一般情報] ビューの "一意な ID""タグ作成者"、および "タグ作成者 ID" 設定に値が設定されている。
- [一般情報] ビューの "SCCM アプリ モデル データの追加" 設定で、[はい] が選択されている。

プロジェクトにソフトウェア ID タグを含めて [リリース] ビューで .pfx ファイルを使ってリリースに署名を行うことを設定すると、InstallShield がビルド時にタグをデジタル署名します。タグ ファイルに署名するためには、.NET Framework 2.0 以降をビルドマシンにインストールする必要があります。

インストールのファイルを編成する

インストールの一番重要なタスクは、ファイルを配布メディアからエンドユーザーのハード ディスクへ転送することです。InstallShield インストールでは、ファイルは階層構造で編成されていて、ファイルはコンポーネントに含まれています。コンポーネントは機能 (およびオプションでサブ機能) に関連付けられています。InstallScript と InstallScript MSI インストールでは、機能はセットアップの種類に関連付けられています。

実行時に、エンドユーザーは、単純にセットアップの種類、または、可能な場合、インストールする機能およびサブ機能を選択するだけです。エンドユーザーにはコンポーネントは見えません。コンポーネントは、ファイルをその種類および目的によってグループ分けするときに使われます。たとえば、同一のターゲットフォルダーにインストールされるファイルは、自己登録ファイルと同様、同一のコンポーネントに配置することができます。

あるファイルが、他のファイルの関数に依存してタスクを実行することがよくあります。ただし、インストールプロジェクトにアプリケーションファイルを含める際に、依存関係と呼ばれる他のファイルに気が付かない場合があります。依存関係ファイルが識別しやすくなるように、InstallShield には自動的に依存関係をプロジェクトに追加する 3 種類の依存関係スキャナーがあります。

プロジェクトへの個別ファイルの追加に加え、再配布可能ファイル (InstallShield 前提条件、マージ モジュールおよびオブジェクト) も含めることができます。再配布可能ファイルには、特定の機能性をインストールするのに必要なロジックとファイルが含まれています。たとえば、Java Runtime Environment (JRE) ファイルをインストールに含める場合、JRE の InstallShield 前提条件をインストール プロジェクトに追加します。

このセクションには、次のトピックが含まれています:

- インストールをデザインする
- パス変数を使用する
- ファイルとフォルダーを含める
- コンポーネントを使用する
- 機能の定義
- セットアップの種類について
- インストールに再配布可能ファイルを含める
- アプリケーションの依存関係を識別する
- COM サーバーの登録
- プロジェクトに DIM を追加する

インストールをデザインする

インストール プロジェクトには、開発者用とエンドユーザー用の 2 つの主要な観点があります。InstallShield では、それぞれ [コンポーネント] ビューと [機能] ビューで詳しくその観点を説明しています。

コンポーネント

コンポーネント は、製品の開発者用の観点を表します。コンポーネントは、ファイル、レジストリ エントリ、およびショートカットなど、開発者が類似したアプリケーション データを論理的にグループ分けするのに役立つインストール オーサリング ツールです。

エンドユーザーが、インストールする機能を選択できるようにするには、アプリケーションの機能に対応するコンポーネントにアプリケーションを分割する必要があります。詳細については、「アプリケーションをコンポーネントに分ける」を参照してください。

InstallShield を使用すると、プロジェクトのコンポーネントを公開することもできます。

メモ: *コンポーネント、製品、またはアクションのパブリッシュに指定した条件は、設計時には検証されません。必ず 正しい構文を使用して、条件によって出力が正常に生成されることを確認してください。*

機能

機能は、エンドユーザーから見たアプリケーションの構成要素です。各機能は、たとえばヘルプファイルのような製品の特定の機能を表します。エンドユーザーは、製品の別個の機能をインストールまたはアンインストールできます。

たとえば、ハードディスクの容量に制限のあるエンドユーザーは、製品チュートリアルをインストールしないよう選択できます。そのユーザーは、後で別のコンピューターを購入するか、または既存のコンピューター上のリソースに空きを作り、前にアンインストールした製品チュートリアルをインストールすることもできます。

アプリケーションのコンポーネントに対応して、アプリケーションを機能に分割する必要があります。詳細については、「アプリケーションを機能に分ける」を参照してください。

アプリケーションをコンポーネントに分ける

次のガイドラインを使用して、このアプリケーションをコンポーネントに分けます。

- グループとして機能するリソースを特定します。各グループをコンポーネントとすることができます。
- Windows Installer では、コンポーネント内のあらゆるファイルが同じディレクトリにインストールされる必要があります。ディレクトリ構造をインストールする必要がある場合、各サブディレクトリが個別のコンポーネントに対応している必要があります。
- コンポーネントのリソースは、同じ条件を必要とします。
- Windows Installer の修復機能を使用するためには、各 .exe、.dll および .ocx ファイルを、個別のコンポーネントに配置し、そのコンポーネントのキーファイルにする必要があります。(コンポーネントウィザードのベストプラクティスオプションでは、この規則を使ってコンポーネントを作成できます。)
- 同様に、各 .chm ファイルまたは .hlp ファイルも、対応する .chi または .cnt ファイルと一緒に、独自のコンポーネントに配置する必要があります。
- いくつかの製品全体を通して、2 つ以上のコンポーネントにリソースを含めることはできません。コンポーネントのリソースが 2 つ以上の製品で必要とされる場合は、共有リソースのマージ モジュール作成をご検討ください。
- コンポーネントを1つまたは複数の機能にマップする際は、かならずコンポーネントのすべての部分が機能にマップされるようにします。
- インストールをテストする際は、アプリケーションをいくつかのコンポーネントに分割して、そのパフォーマンスを徹底的に測定できます。

アプリケーションを機能に分ける

次のガイドラインを使用して、このアプリケーションを機能に分けます。

- アプリケーションを、ヘルプ ファイル、クリップ アート ファイル、プログラム ファイルなどの非依存な部分に分けます。これにより、エンドユーザーはアプリケーションをインストールする際オプションを組み合わせることができます。たとえば、アプリケーションにサイズが大きく画像の多いクリップ アート ファイルが含まれている場合、クリップ アート ファイルを機能にすることができます。これにより、エンドユーザーはファイルをインストールするかしないかを選択することができます。これは、使用できるリソースが限られている場合に極めて重要な機能です。
- アプリケーションを機能に分ける際、かならずエンドユーザーが特定のニーズを実現するため、分けた部分をいくつかの方法で再度組み合わせることができるようにします。これを行う際、システム管理者から顧客サービス担当者、開発者などすべてのユーザーのニーズについて考慮します。ユーザーのすべてのグループに対応することによって、アプリケーションの配布と使用の増加を促すことになります。
- 各機能は、ヘルプ ファイルなどの 1 つの機能性を持ち、この機能性にしたがって明確に定義を行うことで認識および理解されやすくします。機能には、ユーザーが選択で自ら機能をインストールして使用できる独立した機能性が必要です。
- ある機能が別の機能を必要とする場合、依存する機能をもう一方の子にします。
- 混乱を避けるため、システムやアプリケーションの管理に関するあらゆる情報をユーザーに透過にします。

パス変数を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

従来、ソースファイルをインストール プロジェクトにリンクするには、ハードコード化されたパスを使用して、そのファイルへのリファレンスを作成する必要がありました。たとえば、ソースファイルをインストールに含める場合は、C:¥Work¥Files にある Program.exe のように指定していました。但し、ハードコード化したパスを使用すると、ディレクトリからソースファイルを関連付けるときに毎回完全パスを入力する必要がありました。ファイルを別のディレクトリに移動すると、インストール プロジェクトで表示されるようにハードコード化したパスを変更する必要がありました。インストールに含まれているソースファイルの数が少ない場合は、このことは問題にならなかったかもしれません。残念ながら、インストールによっては、フォルダー構造を変えたり、別のマシンにプロジェクトを移行すると、これらすべてのファイルを再度マッピングし直す必要があり何千ものファイルを含んでいることがあります。

プロジェクトを移動したり、ディレクトリ構造を変更するたびに各ソースファイルのパスを変更する必要がないように、パス変数を使って共通して使用するパスを中央ディレクトリに定義することができます。

すべてのパス変数は、[パス変数] ビューで表示、修正することができます。ダイアログ エディター、ダイナミックファイルリンク、リリースフォルダーなど、ソースファイルにリンクされている InstallShield のほとんどすべてのフォルダーのパス変数を使用できます。パス変数を自分で入力するかわりに、パスを参照するたびに InstallShield の推奨値を使用することができます。InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示] オプションを選択します。このセクションには、次のトピックが含まれています:

- パス変数の種類
- カスタム パス変数の作成と定義
- リリースのカスタム パス変数の値をオーバーライドする
- プロジェクト内の絶対パスの参照をパス変数に変換する
- パス変数の削除

メモ: *パス変数は、インストール プロジェクトの開発中に使用されます。これらのパスは、アプリケーションがインストールされるターゲットマシンには適用されません。代わりに、インストール プロジェクトのソースファイルにリンクするために使用されます。プロジェクトをビルドする際、これらのリンクが評価され、ポイント先のファイルがインストールに組み込まれます。*

パス変数の種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

InstallShield では、次の種類のパス変数がサポートされています:

- 定義済みパス変数
- 標準パス変数
- レジストリ パス変数
- 環境パス変数

定義済みパス変数は、標準 Windows ディレクトリを定義します。この種類の変数は、名前を変更することができません。標準、レジストリ、および環境パス変数は、独自に作成および定義が可能なカスタム変数です。

定義済みパス変数

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

定義済みパス変数は特定の標準的の Windows ディレクトリに設定された変数です。これらの変数は修正したり名前を変更することはできませんが、あらかじめ決められたディレクトリをポイントするためにインストール プロジェクトで使用することができます。定義済みパス変数、Windows XP システムでのそれらの通常の値、および対応する InstallScript パス変数を以下のリストに示します。(InstallScript プロジェクトでは、InstallScript パス変数は、たとえば、コンポーネントの "インストール先" プロパティのリストのオプションとして InstallShield で使用されます。InstallScript パス変数はまた、インストールスクリプトでも使用できるシステム変数に対応しています。)

テーブル 1 • 定義済みパス変数

定義済みパス変数	値	InstallScript パス変数
<CommonFilesFolder>	C:¥Program Files¥Common Files¥	<COMMONFILES>
<ISProductFolder>	C:¥Program Files¥InstallShield¥2022	
<ISProjectDataFolder>	<ISProjectFolder>¥ProjectName	<ISPROJECTDIR>
<ISProjectFolder>	C:¥InstallShield 2022 Projects¥	
<ISRedistPlatformDependentExpressFolder>	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386 Express	
<ISRedistPlatformDependentFolder>	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386	
<ProgramFilesFolder>	C:¥Program Files¥	<PROGRAMFILES>
<SystemFolder>	C:¥Windows¥System32¥	<WINSYSDIR>

テーブル 1・定義済みパス変数 (続き.)

定義済みパス変数	値	InstallScript パス変数
<VSSolutionFolder>	状況によって異なる。この変数は、ハイレベルの基本ディレクトリを参照します。このサポートによって、InstallShield プロジェクトで Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルへのスタティック リンクを含むことができます。詳細については、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。	
<WindowsFolder>	C:¥Windows¥	<WINDIR>

デフォルトのパス変数の使用

プロジェクト: *Windows Installer ベースのプロジェクトのみ: [パス変数の推奨] ダイアログ ボックスで、定義済みパス変数を使用するかどうか指定することができます。このダイアログ ボックスは、ソース ファイルをインストールに挿入するときにハードコードされたパスを入力し、[オプション] ダイアログ ボックスにある [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示する] を選択した場合に起動します。定義済みのパス変数に含まれるパスを入力した場合、InstallShield はハードコード化されたパスではなく、パス変数を使用するように指示します。*

定義済みのパス変数は、[標準パス変数] の値を定義する場合にも利用することができます。定義済み変数のサブフォルダーに標準パス変数を定義する必要がある場合があります。たとえば、標準パス変数の値として <ProgramFilesFolder>¥InstallShield を使用することができます。

定義済みパス変数は *スマート* 変数です。つまり、Windows ディレクトリが C ドライブではなく D ドライブ上にある場合でも正しいディレクトリに設定されていることを意味します。[オプション] ダイアログ ボックスの [ファイルの場所] タブでデフォルトのプロジェクトの場所を変更すると、変数が更新され、新しいディレクトリに反映します。

標準パス変数

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*

- *スイート/アドバンスト UI*

標準パス変数は、ユーザー定義変数と呼ばれることもあります。これらの変数タイプは InstallShield に組み込まれています。つまり、これらはレジストリまたは環境変数のような外部リソースに依存しません。

標準パス変数の使用

標準パス変数を作成すると、ソースファイルをプロジェクトに追加するたびに、その変数を使用できます。たとえば、*<MyFiles>*という変数を作成し、その変数が C:¥Work¥Files をポイントしているとします。インストール先にその変数のパスが含まれているプロジェクトにソースファイルを追加する場合、パスをハードコード化するのではなく、変数を使用するようお勧めします。この推奨メッセージは、[パス変数の推奨] ダイアログ ボックスに表示されます。

InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示] オプションを選択します。

ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

レジストリ パス変数

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

レジストリ パス変数を使用すると、指定したレジストリキーのデフォルト値に基づいて独自の変数を定義することができます。

レジストリパス変数を作成すると、ソースファイルをプロジェクトに追加するたびに、その変数を使用できます。ファイルを *MyRegVar* の値を含むフォルダーからコンポーネントに追加する場合、ハードコード化されたパスよりもこのレジストリパス変数を使用することをお勧めします。この推奨メッセージは、[パス変数の推奨] ダイアログ ボックスに表示されます。

InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、[常に [パス変数の推奨] ダイアログを表示] オプションを選択します。

ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

環境パス変数

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

環境パス変数を使用すると、[環境] ダイアログ ボックスの特定の値に基づいて、独自のパス変数を定義できます。

タスク

[環境] ダイアログ ボックスを表示するには、以下の手順を実行します。

1. **[マイ コンピューター]** を右クリックして、**[プロパティ]** を選択します。**[システム プロパティ]** ダイアログ ボックスが開きます。
2. **[詳細]** タブで、**[環境変数]** をクリックします。

環境パス変数が役に立つのは、通常、コマンドラインからビルドを実行する場合です。専用のマシンで夜間ビルドを実行するときは、コマンドラインからパス変数を変更する必要があります。環境パス変数を使用している場合は、バッチファイルまたはコマンドラインからプロジェクトファイルのパスを定義できます。これらのパスは、インストール プロジェクトをビルドする際に評価され、ファイルへの正しいパスが使用されます。

環境パス変数の使用

環境パス変数を作成すると、ソースファイルをプロジェクトに追加するときはいつでもその変数を使用できます。たとえば、*<MyFiles>* という変数を作成し、その変数が C:¥Work¥Files をポイントしているとします。インストール先にその変数のパスが含まれているプロジェクトにソースファイルを追加する場合、パスをハードコード化するのではなく、変数を使用するようお勧めします。この推奨メッセージは、[パス変数の推奨] ダイアログ ボックスに表示されます。

InstallShield がパス変数を自動的に推奨するように設定するには、[オプション] ダイアログ ボックスの [パス変数] タブで、**[常に [パス変数の推奨] ダイアログを表示]** オプションを選択します。

ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

カスタム パス変数の作成と定義

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

[パス変数] ビューを使って、標準パス変数、環境パス変数、およびレジストリ パス変数を作成したり定義したりできます。

タスク ***パス変数を作成および定義するには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - QuickPatch プロジェクトの場合: **[パッチの設定]** の下にあるビュー リストで、**[パス変数]** をクリックします。
   - その他のプロジェクトの場合: **[メディア]** の下にあるビュー リストで、**[パス変数]** をクリックします。
2. 以下のいずれかを実行します。
   - 標準パス変数を作成するには、**[新しいパス変数]** ボタンをクリックします。
   - 環境変数を作成するには、**[新しいパス変数]** ボタンの隣りにある矢印をクリックしてから、**[環境]** をクリックします。
   - レジストリ変数を作成するには、**[新しいパス変数]** ボタンの隣りにある矢印をクリックしてから、**[レジストリ]** をクリックします。

   ビューの下に新しい行が追加されます。
3. 必要に応じて、新しい行の各設定を入力します。

各列の詳細については、「[パス変数] ビュー」を参照してください。

ヒント: *ビルド時に特定のリリースにおけるユーザー定義のパス変数の値、環境変数、またはレジストリ値をオーバーライドするには、そのリリースの [ビルド] タブにある "パス変数のオーバーライド" 設定を使います。詳細については、「リリースの [ビルド] タブ」を参照してください。*

リリースのカスタム パス変数の値をオーバーライドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield では、今回より、プロジェクト内の各リリースでプロジェクトの標準パス変数、環境パス変数、およびレジストリ パス変数の値をオーバーライドすることができます。この機能を使って、ビルドする特定のリリースごとに、ビルド時にプロジェクト内の特定のファイルとフォルダーを別のファイルとフォルダーに置換することができます。

たとえば、基本の MSI プロジェクトの場合、この機能を使ってカスタム アクション DLL ファイル (プロジェクトの 32 ビット リリース用の 32 ビット バージョン ファイルおよびプロジェクトの 64 ビット リリース用の 64 ビット バージョン ファイル) を置換することができます。あるいは、スイート/アドバンスト UI プロジェクトの場合、この機能を使って、プロジェクト内の異なるリリースでイメージや EULA などの UI 要素を置換することができます。これによって、プロジェクトの異なるエディションまたは異なるブランド バージョンのインストールを簡単に生成することができます。

タスク ***リリースのカスタム パス変数の値をオーバーライドするには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[ビルド]** タブにある **"パス変数のオーバーライド"** 設定で、省略記号設定 (...) をクリックします。 **[パス変数のオーバーライド]** ダイアログ ボックスが開きます。

   [パス変数のオーバーライド] ダイアログ ボックスには、[パス変数] ビューで構成されたパス変数、環境パス変数、およびレジストリ パス変数のチェック ボックスが表示されます。定義済みのパス変数、または [リリース] ビュー内の同じリリースで既にオーバーライドされているパス変数のチェック ボックスは含まれていません。
4. 選択されたリリースで、オーバーライドする各パス変数のチェック ボックスを選択します。
5. **[OK]** をクリックします。ダイアログ ボックスを閉じます。**"パス変数のオーバーライド"** 設定の下に、選択された各パス変数に新しいサブ設定が追加されます。
6. 各パス変数のサブ設定で、適切な新しい値を指定します。

ビルド時、InstallShield は [リリース] ビューで指定された新しい値を使って、[パス変数] ビューで設定されたデフォルト値をオーバーライドします。

パス変数のオーバーライドを行うその他の方法

[リリース] ビューのパス変数をオーバーライドする方法以外に、次の方法でパス変数をオーバーライドすることができます:

- IsCmdBld.exe を使って、コマンドラインからビルドする場合は、-I パラメーターを使って、パス変数と新しい値を指定します。
- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで PathVariables パラメーターを使います。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。

これらの方法は、[リリース] ビューの "パス変数のオーバーライド" 設定で設定された値よりも優先されます。

プロジェクト内の絶対パスの参照をパス変数に変換する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

インストール プロジェクトを始めた段階では、パス変数を使用する機会は多くはないかもしれませんが、作業を進めるにしたがって、その有用性が明らかになるでしょう。各ファイルリンクを確認してパス変数に変更する代わりに、ソースパス変換ウィザードを利用することができます。

このウィザードは、プロジェクトのスタティックリンクをスキャンして、すべてをパス変数に変換します。スタティックリンクどれかが既存のパス変数に置換できる場合は、ウィザードで自動的に置換されます。その他のリンクについては、ウィザードが指定可能な変数のリストを表示します。利用する変数を選択した後、ウィザードはリンクのすべてを変換します。

タスク ***ソースパス変換ウィザードを起動するには、以下の手順に従います:***

[プロジェクト] メニューで **[ソース パスの変換]** をクリックします。

パス変数の削除

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

タスク ***パス変数をプロジェクトから削除するには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - QuickPatch プロジェクトの場合: **[パッチの設定]** の下にあるビュー リストで、**[パス変数]** をクリックします。
   - その他のプロジェクトの場合: **[メディア]** の下にあるビュー リストで、**[パス変数]** をクリックします。
2. 削除するパス変数を選択します(複数選択可)。

   連続する複数のパス変数を選択するには、最初のパス変数を選択してから SHIFT キーを押しながら最後のパス変数を選択します。連続しない複数のパス変数を選択するには、最初のパス変数を選択してから CTRL キーを押しながらその他のパス変数を選択します。
3. **[選択したパス変数の削除]** ボタンをクリックします。

ファイルとフォルダーを含める

ファイルは、製品の核であるとともにインストールの核でもあります。ファイルは、コンポーネントレベルでのみプロジェクトに追加されます。インストール プロジェクトでは、コンポーネントは機能と関連付けられています。エンドユーザーがインストールを実行したときに表示されるのは機能です。したがって、インストールでコンポーネントの機能が選択された場合、そのコンポーネントのファイルがターゲット システムにインストールされます。

Windows Installer ベースのプロジェクトでコンポーネントにファイルを追加する場合には、セットアップベストプラクティスに留意する必要があります。デフォルトでは、セットアップベストプラクティスウィザードがセットアップのデザインを監視し、ベストプラクティスに反した場合には警告を表示します。

ファイルを [セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビュー、および [ファイルとフォルダー] ビューでプロジェクトに追加できます。

このセクションには、次のトピックが含まれています:

- プロジェクト ファイルに追加してターゲット システムにインストールする

- コンテキスト メニューを使ってファイルをドラッグ アンド ドロップする
- 64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する
- 新しいインストール先フォルダーの作成
- 空フォルダーの作成
- ハードコード化されたインストール先ディレクトリを指定する
- ターゲット システムからファイルとフォルダーを削除する
- プロジェクトでファイルとフォルダーを管理するときのヒント
- プロジェクトでファイルとフォルダーを検索する
- [ファイルとフォルダー] ビューにコンポーネントを表示する
- [ファイルとフォルダー] ビューを使ってコンポーネントを変更する
- ファイルの設定を構成する
- ファイルの関連付け
- ダイナミック ファイル リンク
- ターゲット システム上でファイルとコンポーネントを上書きする
- [ファイル] ビューで定義済みフォルダーを表示する
- コンパニオン ファイル
- ファイルとフォルダーのアクセス許可を構成する
- ファイルの追加時に COM データを抽出する
- 自己登録ファイルのインストール
- .NET アセンブリのプロパティおよび依存関係を識別する
- InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする

プロジェクト ファイルに追加してターゲット システムにインストールする

[ファイルとフォルダー] ビューを使用して、プロジェクトにファイルを追加することができます。

タスク ***プロジェクトにファイルを追加するには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[ビュー フィルター]** リストで、追加するファイルを含む機能を選択します。
3. Windows Installer ベースのプロジェクトのみ: **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を右クリックして、**[定義済みフォルダーの表示]** をポイントし、使用する定義済みフォルダーをクリックします。

4. **[インストール先コンピューターのフォルダー]** ペインで、ファイルを配置するフォルダーをクリックします。
5. **[ソース コンピューターのフォルダー]** ペインで、追加するファイルが入ったフォルダーに移動します。

   INSTALLDIR (Windows Installer ベースのプロジェクト) または Application Target Folder (InstallScript プロジェクト) は、アプリケーションのファイルのデフォルトのルートディレクトリであるため最もよく使用されるターゲット先となります。
6. 追加するファイルを **[ソース コンピューターのファイル]** ペインから選択して、**[インストール先コンピューターのファイル]** ペインにドラッグします。

ヒント: *INSTALLDIR の [ファイルとフォルダー] ビューでの [インストール先コンピューターのフォルダー] ペインでの表示方法を指定するには、[オプション] ダイアログ ボックスの [ディレクトリ] タブで設定を変更します。*

インストール先ディレクトリをハードコード化して指定することもできます。

[インストール先コンピューターのフォルダー] ペインでは、定義済みのインストール先 リストを使用できます。定義済みのインストール先または独自に作成したインストール先のどちらかにファイルを追加することもできます。

ファイル追加時に自動的に新しいコンポーネントを作成する

プロジェクトに新しいファイルを追加すると、InstallShield が新しいコンポーネントを作成して、これを適切に処理することができます。これを行うには、このビューの [ビュー フィルター] を使って、新しく作成したコンポーネントを関連付ける機能を選択します。

機能が存在しない場合、このビューにファイルを追加してから、機能を作成することもできます。

自動コンポーネントの作成におけるプロジェクト固有の違い

Windows Installer ベースのプロジェクトの場合、InstallShield はセットアップ ベスト プラクティスに基づいてコンポーネントを作成します。たとえば、すべてのポータブル実行可能ファイル (.exe、.dll、.ocx ファイル) に、独自のコンポーネントが与えられます。その他のすべてのファイルは、各インストール先ディレクトリのデフォルトのコンポーネントに追加されます。しかし、インストール先フォルダーの下にリスト表示されているコンポーネントの 1 つにファイルを直接ドラッグする場合、セットアップベストプラクティスの規則は無視されます。

InstallScript プロジェクトでは、InstallShield がファイルの種類に従ってコンポーネントを作成します。たとえば、すべての自己登録ファイルは 1 つのコンポーネントに含まれ、自己登録ファイル以外は別のコンポーネントに含まれ、そして .NET ファイルはすべて別のコンポーネントに含まれます。コンポーネントはサブフォルダーとして [ファイルとフォルダー] ビューに表示されます。

コンポーネントと機能の関係の変更

[インストール先コンピューターのフォルダー] ペインにあるコンポーネントを右クリックしてから [プロパティ] を選択することにより、コンポーネントと機能の関係を変更することができます。[プロパティ] ダイアログ ボックスにある [機能] タブを使って、必要に応じて情報を変更します。

ヒント: *[ファイルとフォルダー] ビューにコンポーネントを表示するには、インストール先フォルダー (ビューの左下ペイン) を右クリックして [コンポーネントの表示] を選択します。*

コンテキスト メニューを使ってファイルをドラッグ アンド ドロップする

[ファイルとフォルダー] ビューにある [ファイル] エクスプローラーを使うと、ソース コンピューターからターゲット システムのインストール先までフォルダーをドラッグ アンド ドロップすることができます。**[ソースコンピューターのフォルダー]** ペインから **[インストール先コンピューターのフォルダー]** ペインにファイルをドラッグする場合、多くのオプションがあります。

ヒント: *64 ビット システム上で InstallShield を使用している場合に、ソースの場所が開発マシン上の 64 ビット システム フォルダー (System32) であるシステム ファイルをプロジェクトに追加するとき、[ファイルとフォルダー] ビューの上部にあるソース コンピューターのペインからインストール先コンピューターのペインの適切な場所にドラッグすることはできません。詳細については、「64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する」を参照してください。*

タスク ***コンテキスト メニューを表示するには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[ソース コンピューターのフォルダー]** ペイン、または **[ソース コンピューターのファイル]** ペインでフォルダーまたはファイルを右クリックしてから、**[インストール先コンピューターのフォルダー]** ペインまたは **[インストール先コンピューターのファイル]** ペインにドラッグします。そしてマウス ボタンを離します。

   フォルダーまたはファイルを追加する定義済みフォルダーが **[インストール先コンピューターのフォルダー]** ペインに表示されていない場合、それを追加することができます。詳しくは、「[ファイル] ビューで定義済みフォルダーを表示する」をご覧ください。

InstallShield は、いくつかのコマンドを含むコンテキスト メニューを表示します。

テーブル 2 - [ファイルとフォルダー] ビューのコンテキスト メニューから使用できるコマンド

コマンド	説明
追加	選択されたフォルダー、サブフォルダーおよび/またはファイルを追加します。これはデフォルトのドラッグ アンド ドロップの動作と同じです。
ソース構造を保持したまま追加	ソース コンピューターのファイル/フォルダー構造を保持しながら、選択されたフォルダー、サブフォルダーおよび/またはファイルを追加します。 このコマンドは、ソース フォルダーが定義済みの Windows インストール先フォルダーと一致する場合にのみ使用できます。さらに、ファイルまたはフォルダーをドロップするインストール先は、インストール先コンピューターにある必要があります。
フォルダーのみを追加	選択されたフォルダーと、選択されたフォルダーに含まれるすべてのサブフォルダーのみを追加します。選択されたフォルダーまたはサブフォルダーに含まれているファイルは追加されません。

テーブル 2 • [ファイルとフォルダー] ビューのコンテキスト メニューから使用できるコマンド (続き.)

コマンド	説明
ソース構造を保持したままフォルダーのみを追加	選択されたフォルダーと、選択されたフォルダーに含まれるすべてのサブフォルダーのみを追加します。選択されたフォルダーまたはサブフォルダーに含まれているファイルは追加されません。このオプションは、また、ソース コンピューターで見つかったフォルダー構造を保持します。 このオプションは、ソース フォルダーが定義済みの Windows インストール先フォルダーと一致する場合にのみ使用できます。さらに、ファイルまたはフォルダーをドロップするインストール先は、インストール先コンピューターにある必要があります。
Cancel	変更を加えずにドラッグアンドドロップ操作を終了します。

64 ビット ソース マシンの 64 ビット System32 フォルダーからファイルを追加する

64 ビット システムでは、System32 フォルダーは、64 ビット アプリケーションに予約されています。InstallShield の [ファイルとフォルダー] ビューで、開発マシンの 64 ビット システム フォルダーを表示しようとした時、32 ビット バージョンのフォルダーである SysWOW64 フォルダーが代わりに表示されます。したがって、64 ビット システム上で InstallShield を使用している場合に、ソースの場所が開発マシン上の 64 ビット システム フォルダー (System32) であるシステム ファイルをプロジェクトに追加するとき、[ファイルとフォルダー] ビューの上部にあるソース コンピューターのペインからインストール先コンピューターのペインの適切な場所にドラッグすることはできません。

リダイレクトを回避して、開発マシン上の 64 ビット System32 ファイルを InstallShield プロジェクトに追加するには、マシン上の Sysnative フォルダーを参照してから、プロジェクトに適切なファイルを選択します。以下は、その手順の説明です。

注意 • *システム フォルダーは Windows によって保護されているため、通常、インストールにシステム ファイルを含めることは推奨されません。これらの場合、システム ファイルの配置/更新には、使用可能な場合、マイクロソフトの再配布可能ファイルを使用すること、または、Windows アップデートを使ってエンドユーザーに更新プログラムを取得してもらう方法を優先してください。*

タスク

64 ビット Windows が搭載された開発システム上の 64 ビット System32 フォルダーのファイルをプロジェクトに追加するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、ファイルを配置するフォルダーをクリックします。
3. **[インストール先コンピューターのフォルダー]** ペインを右クリックしてから **[ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 次のパスを指定します (ドライブ文字は適切なドライブ文字で適宜置き換えてください):

C:¥Windows¥Sysnative

5. プロジェクトに追加したい適切なファイルを選択してから、**[開く]** ボタンをクリックします。

ファイルがプロジェクトへ追加されます。Sysnative フォルダーが、追加したソース ファイルのパスの一部として使用されるようになります。WOW64 は Sysnative フォルダーを特殊なエイリアスとして認識するため、ファイル システムはこのフォルダーからアクセスのリダイレクトを行いません。

ヒント: *また、[コンポーネント] ビューまたは [セットアップのデザイン] ビューを使って 64 ビット System32 ファイルを追加することもできます。これらのビューで、64 ビット System32 ファイルを含むコンポーネントのノードを展開します。そのコンポーネントの下にある [ファイル] ノードを選択します。次に、右側の [ファイル] ペインを右クリックして、[ファイルの追加] をクリックします。前述の手順の通り、C:¥Windows¥Sysnative パスを指定します。*

Windows ベースのシステムにおける Sysnative フォルダー サポートについて

Sysnative フォルダーの使用は、32 ビット マシンでサポートされていません。64 ビット システム上の InstallShield プロジェクトで Sysnative フォルダーを使用していて、32 ビット システムで、その InstallShield プロジェクトのリリースをビルドしようとした時、ソース ファイルが見つからなかったことを通知する 1 つ以上のビルド エラーまたは警告が生成されます。

Sysnative のサポートがある 64 ビット マシン環境にある InstallShield または Standalone Build のオートメーション インターフェイスを使用して、インストールを作成、編集、またはビルドする場合、64 ビット システム フォルダー内にあるシステム ファイルのソース フォルダーを指定する時、Sysnative フォルダーをパスの一部として使用できます。

新しいインストール先フォルダーの作成

タスク ***定義済みフォルダーのサブフォルダーを作成するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのレジストリ]** ペインで、フォルダーを右クリックし、**[新規フォルダー]** をクリックします。
3. 新しいフォルダーを選択して F2 キーを押した後、新しい名前を入力して名前を変更します。

または、全体のフォルダーをインストール先のフォルダーの 1 つにドラッグして、新しいサブフォルダーをソースフォルダーファイルを含むターゲットフォルダーに追加します。この新しいサブフォルダーの名前は、ソース フォルダーの名前と同じです。

空フォルダーの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*

基本の MSI プロジェクト

ダイレクト エディター で表示される .msi データベースの CreateFolder テーブル使用して、空のディレクトリを作成できます。

例については、ナレッジベース記事 Q103218を参照してください。

InstallScript MSI プロジェクト

CreateDir 関数で、空のディレクトリを作成できます。

ハードコード化されたインストール先ディレクトリを指定する

[ファイルとフォルダー] ビューで、ハードコード化したインストール先ディレクトリを指定することができます。

インストール先ドライブの指定

タスク ***特定のドライブを指定するには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を右クリックし、**[新規フォルダー]** をクリックするか、**[インストール先コンピューター]** をクリックして、INSERT キーを押します。新しいフォルダーが追加されます。
3. フォルダーの名前は、ドライブ文字を入力し、続けてコロンを入力します (例 **C:**)。
4. ENTER を押します。

フォルダーおよびサブフォルダーを指定してインストール先パスを作成する

ドライブ文字フォルダーの下のフォルダーおよびサブフォルダーを指定してハードコード化されたインストール先パスを作成することができます。

1. フォルダーを追加するドライブ フォルダー (たとえば **C:**) またはサブフォルダーを追加するフォルダーを右クリックします。また、フォルダーをクリックして、INSERT キーを押しても同じことができます。
2. フォルダーの名前は、新しい名前を入力して ENTER キーを押します。

インストール先パスをすばやく作成する

タスク ***ネストされたインストール先パスを時間を掛けずに作成するには、以下の手順に従います:***

1. フォルダーを右クリックして **[新規フォルダー]** をクリックするか、フォルダーをクリックして INSERT キーを押します。
2. フォルダーの名前は、インストール先パスを入力します (例、**a¥b¥c**)。これにより、**a** を最上位に、**a** の下に **b**、**b** の下に **c** が来るフォルダー階層構造が作成されます。

ターゲット システムからファイルとフォルダーを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、実行時にターゲット システムから削除するファイルとフォルダーを簡単に指定することができるビルトイン サポートが提供されています。このファイルとフォルダーの削除機能は、アプリケーションによって作成されるファイルの削除など、インストールが追跡を行わない処理に使用すると便利です。

ファイルまたはフォルダーの削除は、次のイベントの 1 つにスケジュールできます:

- ファイルまたはフォルダーのコンポーネントがインストールされるとき
- ファイルまたはフォルダーのコンポーネントがアンインストールされるとき
- ファイルまたはフォルダーのコンポーネントがインストールまたはアンインストールされるとき

削除されるアイテムがフォルダーの場合、そのフォルダーが空の場合のみ削除されます。

InstallShield では、プロジェクトからのファイルまたはフォルダーのを削除を構成するさまざまな方法が提供されています:

[ファイルとフォルダー] ビューを使って、削除するファイルとフォルダーを指定します。

タスク

[ファイルとフォルダー] ビューを使って、ファイルとフォルダーの削除を構成するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、削除するファイルまたはフォルダーを含むフォルダーを選択します。
3. **[インストール先コンピューターのファイル]** ペインを右クリックしてから **[ファイルの削除を追加]** をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。
4. 必要に応じて設定を指定します。詳細については、「ファイル 削除の [プロパティ] ダイアログ ボックス」を参照してください。

InstallShield によって、**[インストール先コンピューターのファイル]** ペインにファイルまたはフォルダー アイコンが追加されます。アイコンに赤い X 印がついて、削除される項目を参照していることを示します。

[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使って、削除するファイルとフォルダーを指定する

タスク

[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使って、ファイルとフォルダーの削除を構成するには、次の手順に従います:

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトの場合) または **[コンポーネント]** をクリックします。
2. **[セットアップのデザイン]** または **[コンポーネント]** ツリーで、削除するファイルまたはフォルダーが含まれているコンポーネントのノードを拡張してから、**[ファイル]** サブノードをクリックします。
3. **[ファイル]** ペインを右クリックしてから **[ファイルの削除を追加]** をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。
4. 必要に応じて設定を指定します。詳細については、「ファイル 削除の [プロパティ] ダイアログ ボックス」を参照してください。

InstallShield によって、[ファイル] ペインにファイルまたはフォルダー アイコンが追加されます。アイコンに赤い X 印がついて、削除される項目を参照していることを示します。

ファイルまたはフォルダーの削除の設定を編集する

ファイルまたはフォルダーの削除の設定を変更するには、削除するアイテムを右クリックしてから、[プロパティ] をクリックします。[プロパティ] ダイアログ ボックス が開き、ここで必要に応じて設定を編集できます。

プロジェクトでファイルとフォルダーを管理するときのヒント

ドラッグ アンド ドロップ処理、CTRL+C や CTRL+P といった一般的なキーボード ショートカット、およびアイテムを右クリックすると表示されるコンテキスト メニューを使って、ファイルとフォルダーを異なるインストール先フォルダーや機能へ簡単に移動させることができます。

プロジェクト内のファイルとフォルダーのインストール先を管理する

タスク

プロジェクト内のファイルとフォルダーのインストール先を管理するには、次の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. 以下のいずれかを実行します:
   - ファイルまたはフォルダーの設定を変更するには、アイテムを右クリックしてから、**[プロパティ]** をクリックします。**[プロパティ]** ダイアログ ボックス が開き、ここで必要に応じて設定を編集できます。
   - あるインストール先フォルダーから別のインストール先フォルダーにファイルを移動させるには、**[インストール先コンピューターのファイル]** ペインの 1 つの場所から **[インストール先コンピューターのフォルダー]** ペイン内の適切なフォルダーにドラッグします。
   - あるインストール先フォルダーから別のインストール先フォルダーにフォルダーを移動させるには、**[インストール先コンピューターのフォルダー]** ペインの 1 つの場所から同じペイン内の適切なフォルダーにドラッグします。

- 1 つの場所から別の場所にファイルまたはフォルダーをコピーするには、CTRL を押しながら、アイテムを 1 つの場所から別の場所へドラッグします。

  別の方法として、アイテムをクリップボードにコピーしてから、それを適切な場所に貼り付けることもできます。アイテムを右クリックしてから **[コピー]** をクリックするか、それをクリックして CTRL+C を押します。 次にファイルを含める別のフォルダーを選択します。そのフォルダーを右クリックして **[貼り付け]** をクリックするか、フォルダーをクリックしてから CTRL+P を押します。

- プロジェクトからファイルまたはフォルダーを削除するには、それを右クリックしてから **[削除]** をクリックします。

プロジェクトに含まれるファイルと機能の関連付けを管理する

タスク ***プロジェクトに含まれるファイルと機能の関連付けを管理するには、次の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトの場合) または **[コンポーネント]** をクリックします。
2. 機能の関連付けを管理するファイルを含むコンポーネントのノードを展開してから、**[ファイル]** サブノードをクリックします。右側の **[ファイル]** ペインにファイルが表示されます。
3. 以下のいずれかを実行します:

   - ファイルを 1 つの機能から別の機能にコピーするには、ファイルをクリップボードにコピーしてから、適切な機能にペーストします。ファイルを右クリックしてから **[コピー]** をクリックするか、それをクリックして CTRL+C を押します。 次にファイルを含める別の機能を選択します。その [ファイル] ペインを右クリックして **[貼り付け]** をクリックするか、フォルダーをクリックしてから CTRL+P を押します。

   - ファイルを機能から削除するには、そのファイルを右クリックしてから **[削除]** をクリックします。

プロジェクトでファイルとフォルダーを検索する

多数のフォルダーとファイルをプロジェクトに追加したとき、特定のフォルダーまたはファイルの検索に手間が掛かることがあります。[ファイル] ビューでフォルダーおよびファイルの検索を行うことができます。InstallShield は、マッチしたものをすべて見つけ、最初に検索されたものをハイライトで表示します。検索条件にマッチしたものがすべて見つかるまで検索を続けることができます。

タスク ***プロジェクトでファイルとフォルダーを検索するには、以下の手順を従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を選択します。
3. **[編集]** メニューで、**[検索]** をクリックします。**[検索]** ダイアログ ボックスが開きます。

   また、CTRL+F を押しても同様の結果が得られます。
4. **[検索する文字列]** ボックスで、検索するテキストを入力します。***.exe** などのワイルドカードを利用することができます。
5. **[検索対象]** でファイル、フォルダー、またはその両方を検索するかどうかを指定します。
6. その他に必要な基準を指定します。

7. **[次を検索]** をクリックします。検索基準に一致したものがあったとき、最初のアイテムは、**[インストール先コンピューターのフォルダー]** ペインまたは **[インストール先コンピューターのファイル]** ペインのどちらかで選択されます。
8. 基準に適合する次のアイテムがある場合にそれを検索するには、F3 キーを押します。このステップを必要に応じて繰り返します。

[ファイルとフォルダー] ビューにコンポーネントを表示する

タスク ***インストールのコンポーネントとその関連ファイルを表示するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、[インストール先コンピューター] を右クリックし、**[コンポーネントの表示]** (Windows Installer ベースのプロジェクトの場合) または、**[コンポーネントとサブフォルダーを表示]** (InstallScript ベースのプロジェクトの場合) をクリックします。

機能に関連付けられていないインストール プロジェクト内のコンポーネントは、親のないコンポーネントのアイコン ( ) で表示されます。

[ファイルとフォルダー] ビューを使ってコンポーネントを変更する

[ファイルとフォルダー] ビューの [フォルダー] エクスプローラーで、コンポーネントを右クリックして [プロパティ] をクリックし、[コンポーネントのプロパティ] ダイアログ ボックスを表示します。このダイアログ ボックスでは、コンポーネントのプロパティの変更、ダイナミックファイルリンクの追加、およびコンポーネントを含む機能を変更することができます。

メモ: *(Windows Installer ベースのプロジェクトのみ) フォルダー アイコンを右クリックして、[コンポーネント ウィザードの起動] を選択することにより、コンポーネント ウィザードを起動することができます。*

ファイルの設定を構成する

ターゲット システムにインストールする各ファイルに多くのプロパティを設定することができます。

タスク ***ファイルのプロパティを指定するには、次の手順に従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、プロパティを構成するファイルを含むフォルダーをクリックします。
3. **[インストール先コンピューターのファイル]** ペインで、フォルダーを右クリックし、**[プロパティ]** をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。

設定可能な各プロパティについての説明は、「[ファイルのプロパティ] ダイアログ ボックス」を参照してください。

ファイルの関連付け

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ファイルの関連付けは、特定のタイプのファイルを開くのにどのアプリケーションを使用すべきかを Windows に指示するためのレジストリ設定です。たとえば、Windows では、テキスト(.txt)ファイルを開くと通常「メモ帳」が起動します。

システムで登録済みファイルを表示および変更するには、Windows エクスプローラーを開き、[ツール] メニューで、[フォルダー オプション] をクリックします。ファイルの関連付けの構成方法を見るには、[フォルダー オプション] ダイアログ ボックス の [ファイルの種類] タブを使用します。

似たような方法で、Windows エクスプローラーでファイルを右クリックしてから [プロパティ] をクリックして該当のファイルに関連付けられているアプリケーションを識別することもできます。

ファイルの関連付けは HKLM¥SOFTWARE¥Classes と HKCU¥SOFTWARE¥Classes の両方に格納され、マージされたデータは HKEY_CLASSES_ROOT の下に表示されます。

インストール プロジェクトのファイルの関連付けを作成する

ベストプラクティス ガイドラインは、プロジェクトで作成または使用されるすべての表示タイプのファイルに対してファイルの関連付けを作成することを推奨しています。[ファイル拡張子] ビューを使用すると、インストール プロジェクトにファイルの関連付けを素早く簡単に作成することができます。[ファイル拡張子] ビューは、[コンポーネント] ビューおよび [セットアップのデザイン] ビュー (インストール プロジェクトでのみ)内の詳細設定から使用できます。エンドユーザーがファイルの関連付けを含む機能をインストールと、ファイルの関連付けはターゲット マシンに登録されます。エントリが、レジストリの適切な場所に作成され、ProgID を通してファイルの種類が アプリケーションにリンク付けされます。ProgID は、時折、ファイル タイプのアプリケーション ID またはタグ名と呼ばれ、アプリケーションを一意に識別し、オペレーティング システムがファイルの関連付けを認識しやすくします。

ファイル拡張子のプロパティ構成の他、ファイル拡張子の追加と削除についての詳細は、次のセクションを参照して下さい:

- 新しいファイル拡張子の追加
- ファイル拡張子の削除
- ファイル拡張子のプロパティを設定する

新しいファイル拡張子の追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

アプリケーションがファイルの操作に一意のファイル拡張子を使用する場合、そのファイルの種類を登録することができます。たとえば、アプリケーションが .xyz という拡張子を持つファイルを操作する場合、このファイルの種類を登録しておくと、ユーザーがそのアイコンをダブルクリックしたときにファイルをそのアプリケーションで開くようにオペレーティング システムに指示することができます。

タスク ***新しいファイル拡張子を追加するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、**[拡張子]** を右クリックして、**[新しい拡張子]** をクリックします。InstallShield は **New ExtensionN** (ここで *N* は一意の番号です) というデフォルトの名前を持つ新しい拡張子を作成します。
6. 拡張子をピリオドを付けずに入力します (たとえば、**.ext** ではなく、**ext** と入力します)。

ファイル拡張子の削除

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

タスク ***拡張子の削除方法:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。

5. **[ファイルの種類]** エクスプローラーで、拡張子を右クリックし **[削除]** をクリックします。

ファイル拡張子のプロパティを設定する

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

拡張子のプロパティによって、ファイル拡張子のレジストリ エントリ、およびそれに関連付けられる ProgID が決まります。詳しいヘルプは、各プロパティをクリックすると、InstallShield のヘルプ ペインから見ることができます。

ファイル拡張子のプロパティ構成についての詳細は、次のセクションを参照してください:

- コマンド動詞の指定
- ファイル拡張子から動詞を削除する
- 新しい MIME タイプを追加する
- MIME タイプを削除する
- ProgID の作成
- ProgID の削除

コマンド動詞の指定

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

デフォルトで、[開く] という動詞がファイル拡張子に追加されています。

タスク ***新しい動詞を追加してそのプロパティを指定するには、以下の手順を実行します。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子を右クリックし **[新しい動詞]** をクリックします。InstallShield は **New VerbN** (ここで *N* は一意の番号です) というデフォルトの名前を持つ新しい動詞を作成します。
6. 新しい動詞の名前を入力します。**開く、印刷、検索、プロパティ** などのあらゆる 標準的な動詞を使用することができます。
7. プロパティを編集する新しい動詞を選択します。プロパティ シートが右側に開きます。

ファイル拡張子から動詞を削除する

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

タスク ***動詞を削除するには、以下の手順に従います:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. 動詞を右クリックして **[削除]** をクリックします。

新しい MIME タイプを追加する

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

タスク ***新しい MIME タイプを追加するには、次の操作を実行します。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子を右クリックし **[新しい MIME タイプ]** をクリックします。InstallShield は **MIME TypeN** (ここで *N* は一意の番号です) というデフォルトの名前を持つ新しい MIME タイプを作成します。
6. MIME タイプを入力します (たとえば, **image/jpeg**)。

MIME タイプをクリックしてその [クラス ID] プロパティを表示します。[クラス ID] プロパティをクリックして、下のペインで編集します。CTRL+V キー を押し、クラス ID をプロパティ フィールドに貼り付けます。

MIME タイプを削除する

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

タスク ***MIMEの種類を削除するには、以下の手順に従います:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. MIME タイプを右クリックして **[削除]** をクリックします。

ProgID の作成

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

[ファイルの種類] エクスプローラーの [ProgID] 項目には、ファイル拡張子と関連付けるすべてのプログラム識別子が入ります (ProgID は、ファイルの種類のサポート時にはアプリケーション識別子やタグ名とも呼ばれます)。ファイルの種類の ProgID は任意の文字列ですが、ターゲット システム上で一意である必要があります。ProgID の命名規則には、拡張子にドットを付けず、**file** という語を追加する、というものがあります。従って、.ext 拡張子は *extfile* という ProgID ともなりえます。別の命名規則には、SampleApp.Documentのように、特定の種類のファイルを開くのに使用されるアプリケーションの名前をとってファイルの種類 ProgID を命名する、というものがあります。

たとえば、*.xyz* ファイル拡張子は xyzfile progID をポイントすることができ、*.xyz* ファイルの種類情報はすべて xyzfile の下に登録されます。

タスク ***ProgID を作成するには、以下の手順に従ってください。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子をクリックします。拡張子のプロパティが、右のペインで表示されます。
6. **ProgID** プロパティで、ProgID の名前を入力します。ProgID が、**[ファイルの種類]** エクスプローラーの **ProgID** の下に追加されます。
7. **[ファイルの種類]** エクスプローラーで、新しい ProgID をクリックし、そのプロパティを設定します。

ProgID の削除

プロジェクト: *この情報は、InstallScript および InstallScript オブジェクト プロジェクトには適用しません。*

タスク ***ProgID を削除するには、以下の手順を実行します。***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ)、または **[コンポーネント]** ビューを開きます。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. **[ファイルの種類]** をクリックします。**[ファイルの種類]** エクスプローラーが開きます。
5. **[ファイルの種類]** エクスプローラーで、拡張子をクリックします。拡張子のプロパティが、右のペインで表示されます。
6. **ProgID** プロパティで、ProgID を削除します。

ダイナミック ファイル リンク

ディレクトリのコンテンツをすべてプロジェクトに追加する場合、ダイナミック ファイル リンクを使用できます。ダイナミック リンクにソース フォルダーを選択すると、ビルド時にそのフォルダー内のファイルがリリースに追加されます。ソース フォルダーは常にビルドの前にスキャンされ、すべての新規または変更ファイルが自動的にリリースへ組み込まれます。ダイナミック ファイル リンクは、フォルダー内のファイル一覧 (および該当する場合、サブフォルダー内のファイル一覧) がビルドとビルドの間で変わる可能性があるとき便利です。

重要: *ダイナミック ファイル リンクの使用は注意が必要です。ダイナミック リンクが参照するソース ファイルからダイナミック リンクがあるファイルを誤って削除してしまった場合、そのファイルは、次回リリースをビルドしたとき、リリースに含まれません。このとき、ビルドの警告やエラーも表示されません。製品は問題なくインストールされる場合がありますが、誤って削除されたダイナミック リンクがあるファイルがインストールされないため、適切に動作しない可能性があります。このため、重要な実行可能ファイル (例、.exe、、dll、.ocx ファイル) にダイナミック ファイル リンクを使用しないことが推奨されます (特に、製品でこれらのファイルが正常に実行される必要がある場合)。*

ダイナミック リンクがあるファイルのコンポーネントを作成するとき、可能なかぎり、ディレクトリごとメソッドではなく、ベスト プラクティス メソッドを使用することをお勧めします。ただし、どちらのメソッドでも、ターゲット イメージにあるファイルがアップグレードまたはパッチのダイナミック リンクから削除されると、マイナー アップグレード、スモール アップデート、またはパッチが正しくインストールされない場合があります。

ダイナミック リンクがあるファイルをフィルターする

ダイナミック ファイル リンクを構成するとき、ダイナミック リンクがあるフォルダーのサブフォルダーを含めるかどうかを指定することができます。ダイナミック リンクがあるファイルをさらに細かくフィルターする場合、ダイナミック リンクに含める、または除外するファイルの特定の名前を指定することができます。また、ワイルド カードを使用して、追加または除外する特定のファイルやファイルの種類のみを指定することができます。

たとえば、すべての画像ファイルがサウンド ファイルと共に 1 つのフォルダーの中にあるとき、画像ファイルのみダイナミック リンクを付加する場合、ダイナミック リンクがあるフォルダーに .bmp ファイルと .ico ファイルのみを含めるように指定することができます。これを行うには、以下の例のように、選択パターンにアスタリスク (*) を使用します:

***.bmp、*.ico**

特定のファイルを選択または除外する場合、選択また除外のパターン ボックスに完全なファイル名を入力します。詳細については、「ダイナミック リンクの作成」を参照してください。

InstallShield インターフェイスで、ダイナミック リンクがあるファイル/フォルダーを静的ファイル/フォルダーから識別する

InstallShield インターフェイスでダイナミック ファイルが表示されるとき、ファイルのアイコンに左隅に、ファイルにダイナミック リンクがあることを示すイメージが表示されます:

同じダイナミック ファイルのイメージが、ダイナミック ファイル リンクも含まれているサブフォルダーのアイコンにも含まれています。

また、ダイナミック ファイル リンクに含まれているサブフォルダーのコンポーネント アイコンにあるフォルダーのイメージには、ダイナミック ファイルのイメージのほかに、矢印も追加されています。

InstallShield インターフェイスが静的ファイルとフォルダーに表示するアイコンは、このダイナミック リンクのイメージを含みません。

このセクションには、次のトピックが含まれています:

- ダイナミック ファイル リンクの制限事項
- ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する
- ダイナミック リンクの作成
- ダイナミック ファイル リンクをコンポーネントに追加する
- ダイナミック ファイル リンクにキー ファイルを設定する

ダイナミック ファイル リンクの制限事項

重要: *ダイナミック ファイル リンクの使用は注意が必要です。ダイナミック リンクが参照するソース ファイルからダイナミック リンクがあるファイルを誤って削除してしまった場合、そのファイルは、次回リリースをビルドしたとき、リリースに含まれません。このとき、ビルドの警告やエラーも表示されません。製品は問題なくインストールされる場合がありますが、誤って削除されたダイナミック リンクがあるファイルがインストールされないため、適切に動作しない可能性があります。このため、重要な実行可能ファイル (例、.exe,、dll、.ocx ファイル) にダイナミック ファイル リンクを使用しないことが推奨されます (特に、製品でこれらのファイルが正常に実行される必要がある場合)。*

ダイナミック リンクがあるファイルのコンポーネントを作成するとき、可能なかぎり、ディレクトリごとメソッドではなく、ベスト プラクティス メソッドを使用することをお勧めします。ただし、どちらのメソッドでも、ターゲット イメージにあるファイルがアップグレードまたはパッチのダイナミック リンクから削除されると、マイナー アップグレード、スモール アップデート、またはパッチが正しくインストールされない場合があります。

Windows Installer ベースのプロジェクトの制限事項

基本の MSI プロジェクト、DIM プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトでダイナミック ファイル リンの使用を検討する際、次の制限事項に留意してください:

- ダイナミック リンク ファイル (DLF) へカスタム アクションを作成することはできません。
- ファイル拡張子を動的にリンクされたファイルに作成することはできません。
- 動的にリンクされたファイルから COM 情報を抽出することはできません。
- ″共有″、″パーマネント″、″上書き禁止″ といったプロパティを動的にリンクされたファイルに設定することはできません。
- .NET Installer クラス機能を、動的にリンクされたファイルに設定することはできません。
- COM Interop を動的にリンクされたファイルで有効にするように指定することはできません。
- デフォルトのファイル設定 ([読み取り専用]、[非表示] など) を変更することはできません。
- 動的にリンクされたファイルにファイルのアクセス許可を設定することはできません。
- 動的にリンクされたファイルへのショートカットを作成することはできません。
- 動的にリンクされたファイルについて、スタティック スキャンまたはダイナミック スキャンを実行することはできません。
- 動的にリンクされたフォルダーからサブフォルダーを除外することはできません。
- 基本の MSI プロジェクトの場合: SetupCompleteSuccess エンド ユーザー ダイアログからダイナミックにリンクされたファイルを起動することはできません。

プロジェクトに (ダイナミック リンクを使わずに) 直接追加したファイルは、すべて内部名 (FileKey) を持ちます。カスタム アクション、ファイル拡張子、ショートカット、または他の種類のアイテムを作成したとき、この内部名を実際にポイントします。

ダイナミック リンクを使ってファイルをプロジェクトを追加すると、そのファイルは物理的にプロジェクトに追加されていません。つまり、これらのファイルが、カスタム アクション、ファイル拡張子などに関連付けることができる FileKeys を含んでいないことを意味します。

InstallScript ベース プロジェクトの制限事項

InstallScript プロジェクトと InstallScript オブジェクト プロジェクトでは、単一のコンポーネントに、静的ファイルもダイナミック リンクがあるファイルも含めることはできません。

ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、ダイナミックにリンクされたファイルのコンポーネントを作成するとき、ベストプラクティス を使用する方法と 1 つのディレクトリに対して 1 つのコンポーネントを配置する 2 つの作成方法が提供されています。

ベスト プラクティス方式を使用する

コンポーネント作成のベスト プラクティスに従うと、ダイナミック リンクの選択と除外のフィルター基準を満たすすべてのファイルに対して、次のタスクがビルド時に実行されます。

- ダイナミック リンクがあるフォルダーにある各ポータブル実行可能 (PE) ファイルについてコンポーネントが別々に作成されます。各 PE ファイルは、そのコンポーネントのキー ファイルです。
- ダイナミック リンクのルート レベルにあるすべての非 PE ファイルがリンクを含むコンポーネントに追加されます。
- ダイナミック リンクにサブフォルダーが含まれている場合、サブフォルダー内にあるすべての非 PE ファイルに新しいコンポーネントが作成されます。ダイナミック リンクに複数のサブフォルダーが含まれている場合、各サブフォルダー内のすべての非 PE ファイルにコンポーネントが 1 つずつ別々に作成されます。

これは、すべての新しいダイナミック リンクについてのデフォルト機能です。

ヒント: *[オプション] ダイアログ ボックスの [ファイルの拡張子] タブでは、PE ファイルとして指定するファイルの種類を選択できます。*

ディレクトリごと方式を使用する

コンポーネントの作成にディレクトリごとメソッドが使用されると、ダイナミック リンクの選択と除外のフィルター基準を満たすすべてのファイルに対して、次のタスクがビルド時に実行されます。

- ファイルの種類に関わらず、ダイナミック リンクがあるソース フォルダーのルート レベルにあるすべてのファイルにコンポーネントが 1 つ作成されます。
- ダイナミック リンクに 1 つまたは複数のサブフォルダーが含まれている場合、ファイルの種類に関わらず、各サブフォルダーのすべてのファイルにコンポーネントが 1 つずつ作成されます。サブフォルダーのコンポーネント内にある最初のダイナミック リンクが付いたファイルが、そのコンポーネントのキー ファイルです。

コンポーネント作成のこのメソッドは、ベスト プラクティス メソッドが導入される前に InstallShield で提供されていた従来型のメソッドです。

利用するコンポーネントの作成方式を判別する

ほとんどのダイナミック リンクの場合、コンポーネントの作成に、ベスト プラクティス メソッドが好まれます。この メソッドを利用すると、ディレクトリごとメソッドを利用したときに比べ、パッチをより簡単に作成することができます。マイナー アップグレードとスモール アップデートの場合、コンポーネント、キー ファイル、および機能–コンポーネントの編成は、以前およびその後の .msi データベース間で保持されている必要があります。パッチの場合、File テーブル キーも保持される必要があります。各コンポーネント名とコンポーネント コード

（および、場合により、キー ファイル）はビルド時にダイナミック ファイル リンクのディレクトリごとメソッドによって変更されるため、問題が発生する場合があります。ベスト プラクティス メソッドの利点は、ディレクトリごとメソッドに比べ、予測可能性がより高くなることです。

ヒント: *アップグレードを構成するとき、ビルドの設定にあるパッチの最適化機能を利用します。パッチの最適化を行うと、コンポーネントの名前、コンポーネント コード、File テーブルのキーおよび Directory テーブルのキーを以前のリリースと統一しやすくなります。詳細については、「アップグレード関する考慮事項」を参照してください。*

初期のバージョンの製品にパッチを作成するとき、初期のインストールにディレクトリごとメソッドを使ったダイナミック リンクを含む場合、同ダイナミック リンクに対してディレクトリごとメソッドを続けて使用する必要があります。ただし、アップグレード プロジェクトに新しいダイナミック リンクを追加した場合、これらの新しいダイナミック リンクに対してベスト プラクティス メソッドを使用することができます。つまり、同じプロジェクトで両方の種類のダイナミック リンクを混合して使用し、アップグレードを配布するパッチを作成できるということです。

重要: *ダイナミック リンクがあるファイルのコンポーネントを作成するとき、可能なかぎり、ディレクトリごとメソッドではなく、ベスト プラクティス メソッドを使用することをお勧めします。ただし、どちらのメソッドでも、ターゲット イメージにあるファイルがアップグレードまたはパッチのダイナミック リンクから削除されると、マイナー アップグレード、スモール アップデート、またはパッチが正しくインストールされない場合があります。*

メモ: *パッケージに含まれているファイルがターゲット システムに既に存在するファイルを上書きするとき Windows Installer が使用する規則については、「ターゲット システム上でファイルとコンポーネントを上書きする」をご覧ください。*

利用するコンポーネントの作成方式を指定する

[コンポーネントのプロパティ] ダイアログ ボックスの [ファイル リンク] タブでは、利用するコンポーネントの作成メソッドを指定することができます。

ダイナミック リンクの作成

プロジェクト: *ダイナミック ファイル リンクを作成する手順は、使用するプロジェクトの種類によって異なります。一部の手順は、次の Windows Installer ベースのプロジェクトに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

一部の手順は、次の InstallScript ベースのプロジェクトに適用します:

- *InstallScript*
- *InstallScript オブジェクト*

[ファイルとフォルダー] ビューを利用して、ダイナミック リンクを作成することができます。

タスク ***ダイナミック リンクを作成するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[ビュー フィルター]** リストで、ダイナミック リンクを持つファイルがあるコンポーネントを含める機能を選択します。
3. コンポーネントが、**[インストール先コンピューターのフォルダー]** ペインに表示されていることを確認してください。

   コンポーネントが表示されない場合: [**インストール先コンピューターのフォルダー**] ペインで、**インストール先コンピューター**を右クリックし、[**コンポーネントの表示**] (Windows Installer ベースのプロジェクトの場合) または、[**コンポーネントとサブフォルダーを表示**] (InstallScript ベースのプロジェクトの場合) をクリックします。
4. Windows Installer ベースのプロジェクトの場合: コンポーネントを右クリックしてから **[ダイナミック ファイル リンク]** を選択します。**[コンポーネントのプロパティ]** ダイアログ ボックスが開き、[ファイルのリンク] タブが表示されます。

   InstallScript ベースのプロジェクトの場合: コンポーネントを右クリックしてから、**[ファイルのリンク]** を選択します。[リンク タイプ] ダイアログ ボックスが開きます。
5. ダイナミック リンクを定義して、**[OK]** をクリックしてください。

ヒント: *[コンポーネント] ビューを利用して、ダイナミック ファイル リンクを追加することもできます。詳細については、「ダイナミック ファイル リンクをコンポーネントに追加する」を参照してください。*

ダイナミック ファイル リンクをコンポーネントに追加する

[コンポーネント] ビューを利用して、ダイナミック ファイル リンクを Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトの両方でコンポーネントに追加することができます。ダイナミック ファイル リンクを作成する手順は、使用するプロジェクトの種類によって異なります。

Windows Installer ベースのプロジェクト

次の手順は、基本の MSI プロジェクト、DIM プロジェクト、InstallScript MSI プロジェクトおよびマージ モジュール プロジェクトに適用します。

タスク ***ダイナミック リンクをコンポーネントに追加するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、ダイナミック ファイル リンクを追加するコンポーネントを展開して、その下にある **[ファイル]** をクリックします。
3. **[ファイル]** ペインで右クリックして、**[ダイナミック ファイル リンク]** をクリックします。**[ダイナミック ファイル リンクの変更]** ダイアログ ボックスが開きます。

4. **[新しいリンク]** をクリックします。[ダイナミック ファイル リンクの設定] ダイアログ ボックスが開きます。このダイアログ ボックスで、リンクのソース フォルダーの設定、サブフォルダーを含めるかどうか、およびファイルが自己登録かどうかの指定を行うことができます。また、選択または除外するファイルの種類も指定できます。
5. **[OK]** をクリックします。ファイルの競合は、既存のファイルを新バージョンで上書きして解決します。上書きするときは **[はい]** を、既存のファイルを残す時は **[いいえ]** を、ダイナミック ファイル リンクの設定を保存しないでダイアログを終了する時は **[キャンセル]** をクリックします。フォルダーにファイルが入っていない場合や、他のダイナミックリンクですでに使用されている場合には、警告が表示されます。

新しいファイル リンクの詳細が [ファイル] ペインに表示されます。

InstallScript ベースのプロジェクト

次の手順は、InstallScript および InstallScript オブジェクト プロジェクトに適用します。

タスク ***ダイナミック リンクをコンポーネントに追加するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、ダイナミック リンクを含めるコンポーネントをクリックします。
3. **"リンクの種類"** プロパティの値を選択し、省略記号ボタン(...)をクリックします。[リンク タイプ] ダイアログ ボックスが開きます。
4. 含めるファイルを持つフォルダーを指定し、任意のオプションを指定します。
5. **[OK]** をクリックします。

ヒント: *[ファイルとフォルダー] ビューを利用して、ダイナミック ファイル リンクを追加することもできます。詳細については、「ダイナミック リンクの作成」を参照してください。*

ダイナミック ファイル リンクにキー ファイルを設定する

プロジェクト: *この情報は、次の Windows Installer ベースのプロジェクトで、コンポーネントの作成にディレクトリごとに 1 つのコンポーネント メソッド (ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する に説明があります) を使用したダイナミック リンクに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

この情報は、コンポーネントの作成にベスト プラクティス メソッドを使用するダイナミック リンクには適用しません。

コンポーネントの作成にディレクトリごとに 1 つのコンポーネント メソッドが使用されるダイナミック リンクあるファイルを、コンポーネントのキー ファイルに指定できません。

ダイナミックリンクされたコンポーネントにキーファイルを設定するには、目的のファイルにスタティックリンクを張り、次にこのスタティックリンクをコンポーネントのキーファイルに設定します。ダイナミック リンクの設定で、**"次の拡張子のファイルを除外する"** フィールドにキー ファイルの完全名を入力します。

たとえば、ソースディレクトリに複数の .txt ファイルがあり、Key.txt というファイルをキーファイルにする場合を例とします。まず最初に、Key.txt にコンポーネントへのスタティックリンクを張り、Key.txt をキーファイルに設定します。次に、**"次の拡張子のファイルを含める"** の設定を ***.txt** とし、**"次の拡張子のファイルを除外する"** の設定を **Key.txt** として、先のソース フォルダーへのダイナミック リンクを作成します。

サブディレクトリを含むダイナミックリンクの場合は、ビルドのプロセスで、サブディレクトリ内のコンポーネントの最初のファイルが、動的に生成されたコンポーネントのキーファイルに設定されます。

ターゲット システム上でファイルとコンポーネントを上書きする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

実行時、Windows Installer は次の質問事項を考慮して、ターゲット システム上でファイルを上書きするかどうかを決定します:

- コンポーネントをインストールするべきかどうか?
- コンポーネントをインストールする必要がある場合、そのファイルをインストールするのか、またはアップデートするのか?

次のセクションでは、Windows Installer がこれらの質問事項を評価する方法について説明します。

コンポーネントのインストールが必要かどうかを決定する

コスト分析中、Windows Installer はコンポーネントのキー パスを確認して、コンポーネントのインストールが必要かどうかを決定します。コンポーネントがディレクトリまたは ODBC キー パスを持つ場合、Windows Installer はそのコンポーネントをインストールします。さらに、コンポーネントにレジストリ キー パスが使用されていて、以下の条件の両方が True である場合、Windows Installer はコンポーネントをインストールします:

- コンポーネントの "上書きしない" 設定で [いいえ] が選択されている。
- キー パスがターゲット システムに存在しない。

レジストリ キー パスを持つコンポーネントでこれらの条件の 1 つまたは両方が False の場合、Windows Installer は実行時にそのコンポーネントをインストールしません。

コンポーネントがキー パスの代りにキー ファイルを持つ場合、Windows Installer はターゲット システムでそのファイルの存在を確認します。Windows Installer は、以下のすべての状況下でコンポーネントをインストールします:

- ターゲット システムにそのファイルが存在しない。
- ターゲット システムにそのファイルが存在し、コンポーネントのキー ファイルのバージョン番号が、ターゲット システム上のファイルのバージョン番号よりも新しい、またはそれに等しい。
- ターゲット システム上にそのファイルが存在し、コンポーネントのキー ファイルとターゲット システム上のファイルがバージョン指定されていない。

コンポーネントのキーファイルのバージョン番号がターゲット システム上のファイルよりも小さい場合、実行時に Windows Installer はそのコンポーネントをインストールしません。さらに、Windows Installer はコンポーネントのキーファイルにバージョンが指定されていないが、ターゲット システム上のファイルにはバージョンが指定されている場合、実行時にそのコンポーネントをインストールしません。つまり、Windows Installer は実行時にキーファイルがダウングレードされるような場合、コンポーネントをインストールしません。

ファイルのインストールが必要かどうかを決定する

Windows Installer がコンポーネントのインストールが必要であると判断したとき、そのコンポーネントのファイル確認して、それらをインストールする必要があるかどうかを決定します。キー ファイルの名前とインストール先がターゲット システムにあるファイルの名前と場所に一致すると、Windows Installer は、デフォルトで、次の規則を使って、インストール内のファイルでターゲット システムの対応するファイルを上書きするかどうかを判別します:

- **常に上書き** — ファイルに [常に上書き] チェック ボックスが選択されているとき、ターゲット マシンのファイルはバージョン番号に関わらず上書きされます。
- **バージョン付きファイル** — ターゲット システム上のファイルの方がインストールされるバージョンより新しい場合でも、常に最新バージョンのファイルが保持されます。さらに、バージョン指定されていないファイルではなく、バージョン指定されたファイルが保持されます。
- **ファイル言語** — 他の要素が同じ場合、インストールと同じ言語を持つファイルが他言語のバージョンより優先的に保持されます。この規則に対する例外はファイルが複数言語だった場合にのみ適用されます。単一言語のバージョンのファイルに対して複数言語のファイルが保持されます。
- **日付** — ターゲット システム上に既存するファイルの変更日がそのファイルの作成日以降である場合、ファイルは上書きされません。この規則によってアップグレードまたは再インストール中にユーザー環境設定ファイルが上書きされるのを防ぐことができます。

ファイルがインストールされるコンポーネントに含まれている場合で、それがターゲット システムに存在しないとき、Windows Installer はそのファイルを常にインストールします。

メモ: REINSTALLMODE *プロパティの設定により、デフォルトのファイル バージョン規則を変更することができます。詳細については、「ファイルの上書き規則について」を参照してください。*

File Versioning Rules (Windows Installer ヘルプ ライブラリ)

[ファイル] ビューで定義済みフォルダーを表示する

タスク ***定義済みのインストール先フォルダーを表示するには、次の手順に従います:***

1. **[アプリケーション データの指定]** の下にあるビュー リストで、**[ファイル]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[インストール先コンピューター]** を右クリックして、**[定義済みフォルダーの表示]** をポイントし、使用する定義済みフォルダーをクリックします。

コンパニオン ファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

コンパニオンファイルを使用することで、複数ファイルのインストールアクションを 1 つにバインドすることができます。たとえば、インストール プロジェクトに **FileA.exe** と **FileB.dat** という 2 つのファイルを含む場合、コンパニオン ファイルを使って **FileB.dat** を **FileA.exe** にバインドすることができ、**FileA.exe** をインストールまたは再インストールするときは、**FileB.dat** も同様に処理されるようになります。また、**FileA.exe** をアンインストールするときは、**FileB.dat** も同様にアンインストールされます。

コンパニオンファイルの使用

この機能は、Windows Installer がデフォルトで用いるファイルのバージョン規則をオーバーライドしたいときに便利です。たとえば、バージョン番号のないファイルに対するバージョン規則では、ターゲットマシン上にあるファイルで、その作成日以降に更新されたものはすべて、ユーザーデータのファイルと見なされ、上書き禁止にされます。しかし、こうした判定は必ずしも妥当でない場合があるため、コンパニオンファイルを使用してバージョン番号のないファイルをバージョン番号のあるファイルにバインドできるようになっています。

コンパニオンファイルの関連付け

タスク ***コンパニオンファイルの関連付けは、次の手順で行います。***

1. **[ファイルとフォルダー]** ビューで、プロジェクトにバージョン付きファイルを追加します。
2. **[コンポーネント]** ビューを開きます。
3. **[コンポーネント]** エクスプローラーで、追加したファイルを含むコンポーネントを展開してから **[ファイル]** をクリックします。そのファイルの**キー**列にある値に注目してください。
4. ファイルを右クリックして、**[追加]** をクリックします。
5. 選択したコンポーネントに追加するバージョンが付いていないファイルを選択します。
6. 2 番目のファイルを右クリックして、**[プロパティ]** をクリックします。[プロパティ] ダイアログ ボックスが開きます。
7. **"システム バージョンをオーバーライドする"** チェック ボックスを選択します。
8. **[バージョン]** ボックスには、ステップ 3 で見た **[キー]** 列の値の名前を入力します。

メモ: *ファイルを追加するたびに InstallShield は一意のファイル キーを生成します。ですから、バインド先に設定したファイルを削除すると、ファイルキーが存続し続けない場合があります。バージョンが付いていないファイルのバージョンをアップデートする必要があります。*

ファイルとフォルダーのアクセス許可を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield で、ロックダウンされた環境で製品を実行するエンドユーザーのために、ファイルとフォルダーを保護するための設定を構成することができます。ファイルまたはフォルダーのアクセス許可を特定のグループとユーザーに割り当てることができます。たとえば、管理者グループに特定のファイルについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。

タスク

ファイルまたはフォルダーのアクセス許可を構成するには、以下の手順に従います:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. ファイルの場合: **[インストール先コンピューターのファイル]** ペインで、ファイルを右クリックして**[プロパティ]** を選択します。**[プロパティ]** ダイアログ ボックスが開きます。

   フォルダーの場合: **[インストール先コンピューターのフォルダー]** ペインで、フォルダーを右クリックして**[プロパティ]** を選択します。**[プロパティ]** ダイアログ ボックスが開きます。
3. **[アクセス許可]** ボタンをクリックします。**[アクセス許可]** ダイアログ ボックスが開きます。
4. 必要に応じて、アクセス許可を追加/変更/削除します。詳細については、「ファイルとディレクトリの [アクセス許可] ダイアログ ボックス」を参照してください。

プロジェクトの [一般情報] ビューにある "ロックダウンの設定方法" 設定の選択に従って、InstallShield は ISLockPermissions テーブルまたは LockPermissions テーブルのどちらかにアクセス許可データを追加します。詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

ヒント: *コンポーネントのインストール先フォルダーのアクセス許可を構成することもできます。構成を行うには、まず [コンポーネント] ビューでコンポーネントをクリックします。次に "保存先のアクセス許可" 設定の値をクリックして、省略記号 (...) ボタンをクリックします。[アクセス許可] ダイアログ ボックスが開きます。このダイアログ ボックスで、必要なアクセス許可を構成できます。*

ファイルの追加時に COM データを抽出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[ファイルとフォルダー] ビュー、またはコンポーネントの下の [ファイル] サブビューに新しいファイルをドラッグアンドドロップすると、自己登録ファイルの COM データを静的に抽出できます。これは、ビルド時に COM 登録データを抽出しなかった場合の代替方法です。

このオプションは、[コンポーネント] ビューの [詳細設定] の [COM 登録] ノードからも使用できます。

ファイルの COM データを抽出する

タスク

COM データをビルド時ではなく、オン デマンドで抽出するには、以下の手順に従います:

ファイルを右クリック(または **COM** 登録ノードで **[COM 登録]** を右クリック)して、**[キーファイルの COM データを抽出]** をクリックします。

メモ: *このオプションは、ファイルが .exe ファイルか自己登録で、ファイルがコンポーネントのキーファイルである場合のみ有効になります。*

ヒント: *InstallShield を 64 ビット システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

ファイルの COM データを更新する

ファイルの COM データが既に静的に抽出されている場合、[キーファイルの COM データを更新する] を選択できます。COM データを更新すると、古い COM データはプロジェクトから削除されて、ファイルの現在の COM データがプロジェクトに追加されます。

自己登録ファイルのインストール

インストーラーは、ファイルが自己登録に指定されていて (Windows Installer ベースのプロジェクトの場合)、"ビルド時に COM 抽出" プロパティが [いいえ] に設定されていると、「従来の」自己登録方法を使用して自己登録ファイルを登録します。ファイルはアンインストール時に登録解除されます。サポートされている自己登録関数は **DllRegisterServer** および **DllUnregisterServer** です。

ファイルの "登録" プロパティを設定する前に、ファイルが自己登録に指定されていることを確認してください。セットアップ ベスト プラクティスに従い、.dll、.ocx、または .exe は 1 つのコンポーネントにつき 1 つだけにする必要があります。

ヒント: *Windows Installer を使って自己登録ファイルをインストールするには、.msi データベース テーブル (クラス、ProgID およびその他) に登録情報を書き込む方法をお勧めします。ファイルを自己登録にしない場合、代わりに、コンポーネントの詳細設定を使用するか、またはCOM 情報をビルド時に抽出するか、あるいは[ファイルとフォルダー] ビューにファイルを追加したときに COM 情報を抽出して、progIDs やタイプライブラリなどを登録できます。詳細設定は、インストール中にファイルをインストールする場合でも、「ジャスト イン タイム インストール」としてファイルをアドバタイズする場合でも有効です。その他に有利な点は、インストールが失敗した場合にファイルの登録を解除できることです。*

コンポーネントの [ビルド時に COM 抽出] プロパティを いいえ に設定しても、COM 登録の詳細設定に格納されている情報はインストール時に登録されます。COM 登録の詳細設定とコンポーネントのレジストリデータをチェックすることにより、予定通りのエントリが使用されており自己登録機能で作成されたエントリと競合していないことを確認することができます。

.NET アセンブリのプロパティおよび依存関係を識別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield は、.NET アセンブリのプロパティおよび依存関係の識別をサポートします。使用するプロジェクトの種類によって、その機能は異なります。

Windows Installer ベースのプロジェクト

基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、プロジェクトに .NET アセンブリを追加する際に、それをコンポーネントのキー ファイルとして設定することが推奨されます。その他の .NET アセンブリは、そのコンポーネントに含まないでください。

.NET アセンブリがそのコンポーネントのキーファイルである場合、以下の方法の 1 つを使ってその依存関係を識別できます:

- .NET アセンブリで必要なファイルとマージ モジュールが必要が判明している場合、プロジェクトの .NET アセンブリ コンポーネントを含む機能にそれらを手動で追加することができます。プロジェクトにどのファイルを含めるのかを大幅に制御できるこの方法が推奨されます。
- .NET アセンブリに必要なファイルおよびマージ モジュールが不明な場合には、以下のビルトイン処理の 1 つを行ってください:
  - .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別するには、スタティック スキャン ウィザードを使います。このウィザードで検出された依存関係のリストが表示され、それぞれを

プロジェクトに含めるかどうかを指定することができます。この方法は基本の MSI および InstallScript MSI プロジェクトで使用できますが、DIM またはマージ モジュール プロジェクトでは使用できません。

- プロジェクトをビルドする度に .NET アセンブリの依存関係を識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [依存関係およびプロパティ] を選択します。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

.NET アセンブリのプロパティを構成することで、Windows Installer が必要に応じてアセンブリをアップデートおよびアンインストールできるようになります。.NET アセンブリがそのプロパティのキーファイルである場合、以下の方法のいずれかを使ってその依存関係を識別できます:

- 手動でプロパティおよびその値を入力する: [コンポーネント] ビューで、.NET アセンブリを含むコンポーネントの [詳細設定] 領域にある [アセンブリ] ノードの下の [.NET アセンブリ] サブノードを選択してから、必要に応じてプロパティを構成します。入力するプロパティと値は、アセンブリのマニフェスト ファイル内の情報と一致しなくてはなりません。一致しなかった場合、.NET コンポーネント含まれる機能がアンインストールされる時に、アセンブリがターゲット システムに残る可能性があります。
- プロジェクトをビルドする度に .NET アセンブリのプロパティを識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [プロパティ] のみまたは [依存関係およびプロパティ] を選択します。InstallShield がビルド時に不足している可能性のあるプロパティを検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

InstallScript プロジェクト

InstallScript プロジェクトの場合、プロジェクトのコンポーネントに .NET アセンブリを追加するとき、コンポーネントの ".NET アセンブリ" 設定を使って、アセンブリがローカル .NET アセンブリであることを識別することができます。また、以下の方法の 1 つを使って、.NET アセンブリの依存関係を識別できます:

- .NET アセンブリで必要なファイルとマージ モジュールが必要が判明している場合、プロジェクトの .NET アセンブリ コンポーネントを含む機能にそれらを手動で追加することができます。プロジェクトにどのファイルを含めるのかを大幅に制御できるこの方法が推奨されます。
- .NET アセンブリに必要なファイルおよびマージ モジュールが不明な場合には、以下のビルトイン処理の 1 つを行ってください:
  - .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別するには、スタティック スキャン ウィザードを使います。このウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。
  - プロジェクトをビルドする度に .NET アセンブリの依存関係を識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [依存関係] を選択します。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

コンポーネントの ".NET アセンブリ" 設定で [アセンブリではない] が選択されている場合、InstallShield はコンポーネントのアセンブリの依存関係をスキャンしません。

コンポーネントの ".NET アセンブリ" 設定で [ローカル アセンブリ] が選択されている場合、インストールは実行時にコンポーネントの COM の相互運用機能への登録および .NET Installer クラス情報の構成を行います。

依存関係の 64 ビット .NET アセンブリをスキャンする

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用していて、以下のどちらかのビルトイン処理によって依存関係を検出する場合、InstallShield はプロジェクトに含まれる 64 ビット .NET アセンブリの 64 ビット依存関係をスキャンすることができます。

- 64 ビット .NET アセンブリの依存関係の可能性があるファイルをオンデマンドで識別するには、スタティック スキャン ウィザードを使います。このウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。
- プロジェクトをビルドする度に 64 ビット .NET アセンブリの依存関係を識別するには、コンポーネントの "ビルド時に .NET をスキャン" 設定で [依存関係] オプションの 1 つを選択します。InstallScript プロジェクトの場合、コンポーネントの ".NET アセンブリ" 設定を [ローカル アセンブリ] に設定しなくてはなりません。InstallShield がビルド時に不足している可能性のある依存関係を検出した場合、InstallShield が生成するリリースにはそれが組み込まれます。

これらの処理は、プロジェクトに含まれる 32 ビット .NET アセンブリの 32 ビット依存関係のスキャンも行います。

Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのビルトイン スキャンはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみを検出できます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

Windows Installer ベースのプロジェクトで、ビルトイン スキャン処理がプラットフォーム依存またはプラットフォーム非依存 .NET 依存関係を検出する方法

基本の MSI、DIM、InstallScript MSI およびマージ モジュール プロジェクトで、前述のビルトイン処理を使って .NET 依存関係を検出する場合、InstallShield は次の表に示される特定の順序に従って検出を行います。

テーブル 3 • .NET アセンブリの依存関係を検出する順序

スキャンが行われるファイルの種類	32 ビット Windows システム上のスキャン結果	64 ビット Windows システム上のスキャン結果
GAC 内のプラットフォーム非依存 .NET アセンブリ	InstallShield が依存関係をスキャンする順序: 1. 32 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係	InstallShield が依存関係をスキャンする順序: 1. 64 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係

テーブル 3 • .NET アセンブリの依存関係を検出する順序 (続き.)

スキャンが行われるファイルの種類	32 ビット Windows システム上のスキャン結果	64 ビット Windows システム上のスキャン結果
GAC 内の 64 ビット .NET アセンブリ	InstallShield は、依存関係を検出しません。	InstallShield が依存関係をスキャンする順序: 1. 64 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係
GAC 内の 32 ビット .NET アセンブリ	InstallShield が依存関係をスキャンする順序: 1. 32 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係	InstallShield が依存関係をスキャンする順序: 1. 32 ビット固有 .NET 依存関係 2. プラットフォーム非依存 .NET 依存関係

.NET アセンブリが GAC に存在しない場合、InstallShield は依存関係をスキャンするときにアセンブリが含まれているフォルダーを検索してから、サブフォルダーを検索します。

.NET 依存関係スキャナー結果の確認

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、様々な .NET アセンブリの依存関係を識別するためのビルトイン処理 が提供されています。これらのビルトイン処理を使用すると、スキャンによって製品が必要としないファイルが依存関係またはマージ モジュールとして識別される可能性があります。スタティック スキャン ウィザードを使用する際にこの状況が発生した場合、プロジェクトに特定の依存関係を追加しないことをウィザードで指定することができます。ビルド時に .NET 依存関係の可能性があるすべてのファイルをプロジェクトに自動的に含める、[ビルド時に .NET をスキャン] 機能を使用する場合、.NET アセンブリが自動的にスキャンされないように、コンポーネントの設定値を変更することができます。

- 基本の MSI、InstallScript MSI およびマージ モジュール プロジェクトの場合、InstallShield はデフォルトでビルド時にコンポーネントの .NET アセンブリの依存関係を自動的にスキャンします。必要に応じて、コンポーネントの "ビルド時に .NET をスキャン" 設定の値を変更して、このデフォルト動作を上書きすることができます。
- InstallScript プロジェクトの場合、InstallShield はデフォルトでビルド時に各コンポーネントの .NET アセンブリの依存関係をスキャンしません。必要に応じて、コンポーネントの ".NET アセンブリ" 設定およびその "ビルド時に .NET をスキャン" 設定の値を変更して、このデフォルト動作を上書きすることができます。

さらに、InstallShield を使って依存関係スキャンを行うときは常に自動的に含めたり除外したりするファイルをマシン全体で指定することも可能です。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

依存関係を識別するときに最も良い結果を得るために、クリーン マシン上で製品とそのインストールを十分にテストすることが推奨されます。製品が予定通りに動作しなかった場合、マシン上で足りない依存関係がないか、またそれをインストールに含むべきかどうかを判断してください。

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでフォントをインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

([コンポーネント] ビューまたは[ファイルとフォルダー] ビューを使用して) インストール プロジェクトにフォント ファイル(.ttf、 .ttc または .fon ファイル) 含めるとき、グローバルフォント登録を有効にしておくと、ファイルは自動的にターゲットマシンに登録されるよう内部的にマークされます。実行時に、**OnInstalledFontFile** イベント ハンドラー関数が各フォントファイルのインストールされた直後に呼び出されます。フォントが登録されるよう内部的にマークされている場合、このイベント ハンドラー関数のデフォルトコードは、 **RegisterFontResource** 関数を呼び出してフォントを登録します。

フォントタイトル

InstallShield は、インストールが登録するフォントタイトルを以下のように判断します。

- .fon ファイルの場合、InstallShield は、デフォルトで、フォント タイトルをソース システムのレジストリから読み込みます。タイトルがレジストリに見つからなかった場合、InstallShield は、ファイル名をフォント タイトルとして格納します。"リンク イプ" プロパティに [スタティック] が設定されているコンポーネントにある .fon ファイルの場合、InstallShield はそのファイルの [ファイルのプロパティ] ダイアログ ボックスの [フォントのタイトル] ボックスにフォントのタイトルを表示します。
- TrueType ファイル (.ttf または .ttc ファイル)の場合、デフォルトでは、インストールは実行時にそのファイルからフォント タイトルを読み込みますが、デザイン時には、[フォント タイトル] ボックスにはテキストは表示されません。
- 3つすべてのタイプのファイルにおいて、ファイルが リンクタイプ プロパティに スタティックが設定されているコンポーネントにあるとき、[フォントタイトル] ボックスに非デフォルトテキストを入力した場合、インストールはそのテキストをフォントタイトルとして登録します。

Windows Fonts Folder にフォントをインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*

- *InstallScript オブジェクト*

タスク ***次のいずれかを実行して、Windows Fonts フォルダーにフォントファイルをインストールします:***

- **[ファイルとフォルダー]** ビューで、**[インストール先コンピューターのフォルダー]** ペインの **Fonts Folder** フォルダーにフォント ファイルを置きます。Fonts Folder フォルダーは、**Windows** フォルダーのサブフォルダーです。
- **[コンポーネント]** ビューで、**"インストール先"** プロパティに **<FOLDER_FONTS>** が設定されているコンポーネントにフォント ファイルを置きます。

グローバル フォント登録を有効または無効にする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

フォントファイルの自己登録をグローバルに有効または無効にすることができます。

タスク ***グローバルフォント登録を有効または無効にするには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"フォントの登録"** 設定で、適切なオプションを指定します:
   - グローバル フォント登録を有効にするには、**[有効]** を選択します。
   - グローバル フォント登録を無効にするには、**[無効]** を選択します。

フォント ファイルの登録を有効または無効にする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

個々のファイルに対して、フォント ファイルの自己登録を有効または無効にすることができます。

タスク ***個々のフォントファイルの自己登録を有効または無効にするには、以下の手順を実行してください。***

1. **[コンポーネント]** ビュー、**[セットアップのデザイン]** ビューまたは **[ファイルとフォルダー]** ビューを開きます。
2. フォント ファイルを右クリックして、**プロパティ**をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。

3. [**フォントファイルを登録する**] チェック ボックスを選択して、フォントファイルの自己登録を有効にします。

   自己登録を無効にするには、[**フォント ファイルの登録**] チェック ボックスをクリアしてください。

4. [OK] をクリックします。

コンポーネントを使用する

インストール開発者の視点からみると、コンポーネントはアプリケーションの要素です。コンポーネントはエンドユーザーには表示されません。ユーザーがインストール用の機能を選択すると、インストーラーによってその機能に関連したコンポーネントが判断され、それらのコンポーネントがインストールされます。アプリケーションのコンポーネントには実行可能バイナリファイル、データファイル、ショートカット、ヘルプ システムファイル、およびレジストリ エントリが含まれます。

コンポーネントは、[セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビューで修正できます。

コンポーネントと機能の関係

プロジェクト: *機能は DIM プロジェクトまたはマージ モジュール プロジェクトでは使用されません。*

コンポーネントは、[セットアップのデザイン] ビューで機能に関連付けられます。詳細については、「新しいコンポーネントを機能に関連付ける」を参照してください。

ファイル

[セットアップのデザイン] ビューまたは [コンポーネント] ビューでコンポーネントのツリーを展開して、ファイル をクリックし、コンポーネントに関連付けられた全ファイルの一覧を表示します。ファイルの追加方法に関する情報は、「ファイルをコンポーネントに追加する」をご覧ください。

セットアップ ベスト プラクティス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield は、[セットアップのデザイン] ビューおよびコンポーネント ウィザードでコンポーネントを作成中に発生したベスト プラクティス違反に対して警告を行い、セットアップ ベスト プラクティスの厳守を容易にします。これらのガイドラインに従うことにより、再使用可能なコンポーネントを有効に配布して、ファイルの非互換性問題を伴わないクリーンなインストールを作成することができます。

[セットアップのデザイン] ビューでコンポーネントを作成する際、セットアップベストプラクティスウィザードは、コンポーネントに追加するファイルを監視し、ベストプラクティスとの競合がある場合に警告し、ウィザードでアクションを修正できるようにします。ウィザードのセットアップのデザインの自動スキャン機能をオフにするには、[ツール] メニューの [オプション] を選択します。

InstallShield は、次のセットアップ ベストプラクティスに準拠しているかどうかを確認します。

テーブル 4 • InstallShield がモニターするセットアップベストプラクティス

ベスト プラクティス	説明
コンポーネントに複数の .exe、.dll、.ocx、.chm、または .hlp ファイルを含めない	各コンポーネントに含めるのは、ポータブル実行可能ファイル (.exe、.dll、または .ocx ファイル) または WinHelp ファイル (.hlp ファイル) の 1 つだけにする必要があります。Windows Installer のコンポーネントは、GUID コード など、すべての詳細設定とコンポーネントの設定が、理想的には単一のポータブル実行可能ファイルまたはヘルプ ファイルを参照するように設計されています。他の .exe、.dll、.ocx、または .hlp ファイルを新しいコンポーネントに配置します。 この理由は、インストール実行時にユーザーがインストール済み製品に対する修復を選択した場合、Windows Installer は、インストール済みの各コンポーネントのキー ファイルの存在をチェックするためです。そして欠落しているキーファイルがあれば、Windows Installer はコンポーネントを再インストールします。そのため、コンポーネントのキーファイルではないファイルが欠落してる場合、修復モードでは復旧されない可能性があります。
マージ モジュールの使用	マージ モジュールには、個別の機能をインストールするために必要な、すべてのファイル、レジストリ エントリ、および論理が含まれています。マージ モジュールが可能なファイルを配布しないでください。また、マージ モジュールを使用すると、1 つのファイルを複数のコンポーネントに関連付けないというベストプラクティスと、いかなるコア コンポーネントも外に出さないという Windows ロゴ ガイドラインの 2 つの関連要件に準拠することができます。 ベストプラクティスをモニターするオプションを有効にすると、InstallShield が提供するマージ モジュールの一部となっているファイルをコンポーネントに追加したとき、常にセットアップベストプラクティスウィザードが警告を発します。

InstallShield では自動的に警告が出ませんが、以下のコンポーネント作成におけるセットアップベストプラクティスについても注意してください。

テーブル 5 • InstallShield が自動的に警告を発しないセットアップ ベスト プラクティス

ベスト プラクティス	説明
ショートカット ターゲットをそれ自身のコンポーネントへ配置する	ショートカットのターゲットになるファイルは、それ自身のコンポーネントが必要です。このファイルは、コンポーネントのキーファイルである必要があります。

テーブル 5 • InstallShield が自動的に警告を発しないセットアップ ベスト プラクティス (続き.)

ベスト プラクティス	説明
他のファイルをコンポーネントへグループ化する	共通のインストール先フォルダーやバージョン確認などファイルの要件に従って、上記のどのコンポーネントのカテゴリにも該当しないすべてのファイルを整理してください。
ファイルを自己登録しない	自己登録機能を呼び出して COM サーバー情報の登録および登録解除を行う代わりに、インストール時にコンポーネント用のこの情報を登録するようにしてください (インストーラーはアンインストール時にこの登録を解除します)。 自己登録はファイルの登録や登録解除には信頼性に欠ける方法です。インストールが情報を登録することの 1 つの利点は、ファイルがアドバタイズされるかすぐにインストールされない場合に、後でインストーラーによりファイルが要求されたときでも、登録情報をすぐに取得できるということです。このため、InstallShield は COM サーバーの登録についてだけ、詳細設定をサポートします。

コンポーネントの作成

InstallShield でのコンポーネントの作成には、いくつかの方法があります。

- [セットアップのデザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューを使って、新しいコンポーネントを作成してから手動で構成する。詳細については、「[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使ってコンポーネントを作成する」を参照してください。
- コンポーネント ウィザードを使用して、InstallShield が自動的にセットアップ ベスト プラクティスに従ってコンポーネントを定義付ける。このウィザードは、基本の MSI、DIM、InstallScript、InstallScript MSI、およびマージ モジュール プロジェクト タイプで利用できます。詳細については、「コンポーネント ウィザードのベスト プラクティス オプションを使用する」を参照してください。
- COM サーバーまたはフォント用のコンポーネントを作成する場合、コンポーネント ウィザードを使用することが推奨されます。このウィザードは、基本の MSI、DIM、InstallScript、InstallScript MSI、およびマージ モジュール プロジェクト タイプで利用できます。詳細については、「コンポーネント ウィザードの [コンポーネント タイプ] オプションを使用する」を参照してください。

コンポーネントをプロジェクトにエクスポートする手順は、「他のプロジェクトへコンポーネントをエクスポートする」を参照してください。

プロジェクト内のコンポーネントを削除するには、「プロジェクトのコンポーネントを削除する」を参照してください。

[セットアップのデザイン] ビューまたは [コンポーネント] ビューを使ってコンポーネントを作成する

次に説明するように、[コンポーネント] ビューで新しいコンポーネントを作成して機能と関連付けたり、[セットアップのデザイン] ビューで特定の機能のコンポーネントを作成することができます。[セットアップのデザイン] ビューは、DIM プロジェクトまたはマージ モジュール プロジェクトには使用できません。

[セットアップのデザイン] ビューまたは [機能] ビューの InstallShield オブジェクト インストーラー機能の上に配置する機能に必要なファイルを含めてください。

インストール プロジェクトのみ、以下の手順に従って [セットアップのデザイン] ビューでコンポーネントを作成します。

タスク ***[セットアップのデザイン] ビューで、コンポーネントを作成するには、次の手順に従います。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、機能またはサブ機能を右クリックして、**[新しいコンポーネント]** をクリックするか、CRTL+INSERT を押します。InstallShield が、**New Component n** (ここで *n* は連続番号です) というデフォルトの名前で新しいコンポーネントを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

選択した機能と新規コンポーネントが自動的に関連付けられます。

[コンポーネント] ビューを使用して、コンポーネントを作成する

DIM プロジェクトとマージ モジュール プロジェクトの場合、以下の手順に従って [コンポーネント] ビューでコンポーネントを作成します。

タスク ***[コンポーネント] ビューでコンポーネントを作成するには、以下の手順を実行します。***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーを右クリックして、**[新しいコンポーネント]** をクリックするか、INSERT を押します。InstallShield が、**New Component n** (ここで *n* は連続番号です) というデフォルトの名前で新しいコンポーネントを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. インストール プロジェクトの場合、1 つまたは複数の機能とコンポーネントの関連付けを行う必要があります。

メモ: *インストール プロジェクトでは、機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコン ( ) で表示されます。*

InstallScript プロジェクトでは次の文字をコンポーネント名に使用できません:

*¥ / : * ? " ' < > |*

コンポーネントを作成すると、そのプロパティ シートが右側に表示されます。必須のコンポーネント コード (Windows Installer ベースのプロジェクトのみ) やインストール先プロパティなど、コンポーネントのプロパティを直ちに指定します。

コンポーネントを作成しプロパティを指定した後、これに、アプリケーションのファイルやレジストリ エントリ、ショートカットをコンポーネントに関連付けることができます。

コンポーネント ウィザードのベスト プラクティス オプションを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントを作成する最も速い方法は、コンポーネントウィザードを起動して InstallShield によってファイルをコンポーネントに構成する方法です。**"ベストプラクティスを使用してコンポーネントを作成する"** オプションを選択すると、ウィザードがセットアップ ベスト プラクティス を指定されたファイルに適用して、コンポーネントを作成します。

タスク ***コンポーネント ウィザードを起動するには、以下のいずれかを実行します。***

- インストール プロジェクトのみ: [**セットアップのデザイン**] ビューの機能を右クリックして、**コンポーネントウィザード**を選択します。
- インストール プロジェクト、DIM プロジェクト、およびマージ モジュール プロジェクトの場合: **[コンポーネント]** ビューのエクスプローラーで **[コンポーネント]** を右クリックしてから、**[コンポーネント ウィザード]** を選択します。

コンポーネント作成におけるベスト プラクティス規則

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネント ウィザードを使ってコンポーネントを作成するときに [ベストプラクティス] オプションを選択すると、ウィザードがセットアップ ベスト プラクティスに基づいて、自動的にファイルをコンポーネントを編成します。その場合、次のルールにしたがってコンポーネントを作成します。

- ウィザードは、追加されるすべてのファイルが既にプロジェクトにインクルードされていないか調べます。複製ファイルの場合、ウィザードはこのファイルのコンポーネントを作成しません。

- 新しいコンポーネントは、各 ポータブル実行可能ファイル (.exe、.dll、または .ocx ファイル)に対して作成する必要があります。コンポーネントには、ポータブル実行可能形式の名前が付きます。ウィザードは GUID を [コンポーネントコード] プロパティに作成し、ポータブル実行可能ファイルを、コンポーネントのキーファイルとして設定します。
- コンポーネントウィザードは、各々のポータブル実行可能形式の登録情報を抽出しようとします。正常に抽出されると、ウィザードは COM サーバーコンポーネントを作成して、データをコンポーネントの COM サーバーの詳細設定に書込みます。
- 指定されるすべてのヘルプ (.hlp) ファイルはそのコンポーネントの中に、関連のコンテンツ (.cnt) ファイルと共に存在します。コンポーネントには、そのヘルプファイルの名前がつけられます。ウィザードは GUID を [コンポーネントコード] プロパティに作成して、ヘルプファイルをコンポーネントのキーファイルとします。コンポーネントを伴う HTML ヘルプ(.chm) ファイルと .chi ファイルにも同じ規則が適用されます。
- コンポーネントウィザードは、すべてのフォント (.ttf と .ttc) ファイルを、フォントファイルと呼ばれる1つのコンポーネントに挿入します。ウィザードは GUID を [コンポーネントコード] プロパティに作成し、リストの最初のファイルをキーファイルに設定します。指定する .fon ファイルは、作成される AllOtherFiles コンポーネントに自動的に含まれます。ただし .fon ファイルにはタイトルが付かないので、これらは .msi パッケージのフォントテーブルに追加されません。これらのファイルをフォントとして追加するには、コンポーネント ウィザードの"フォント オプション"を使います。
- 上記のカテゴリに含まれないファイルは、「その他のファイル」という名の1つのコンポーネントに分類されます。ウィザードは GUID を [コンポーネントコード] プロパティに作成し、リストの最初のファイルをキーファイルに設定します。

InstallShield による COM サーバー コンポーネントの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネント ウィザードの [ようこそ] パネル内の **[ベストプラクティスを使用してコンポーネントを作成する]** オプションを選択した場合、InstallShield は指定した各ポータブル実行可能ファイルに対して個々にコンポーネントを自動作成します。ウィザードはまた、各々のポータブル実行可能形式ファイルの登録情報を抽出します。

正しく抽出されると、COM 登録詳細設定にマップする登録情報のすべてがそこに書き込まれます。ウィザードは、InProcServer32 ThreadingModel 値など、すべての追加エントリをコンポーネントのレジストリ データに作成します。

メモ: *COM サーバーが実行可能な場合、値 1 の OLESelfRegister 属性を、コンポーネントのリソースファイルのバージョンセクションに置く必要があります。*

InstallShield によるフォント コンポーネントの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネント ウィザードの [ようこそ] パネル内の **[ベスト プラクティスを使用してコンポーネントを作成する]** オプションを選択した場合、InstallShield は指定したすべてのフォント ファイル (.ttf および .ttc) に対して個々にコンポーネントを自動作成します。すべての .fon ファイルは、埋め込みフォントタイトルを持っていないために作成される AllOtherFiles コンポーネントに追加されます。.fon ファイルをセットアップにフォントとして追加するには、コンポーネントウィザードの "フォント オプション" を使用します。

フォントコンポーネント用のインストール先フォルダーは、コンポーネントウィザードで選択したインストール先フォルダーに設定されます。コンポーネントを作成した後、そのインストール先フォルダーをフォント用の通常のデフォルトの場所である Windows Installer フォルダー プロパティ [FontsFolder](つまり、[FontsFolder] をインストール先として使う)に設定できます。FontsFolder の両側に [INSTALLDIR] を指定するときと同様に角かっこをつけます。

タスク ***システムにフォントが登録されていない場合は、以下の手順でフォントタイトルを指定する必要があります。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、フォント ファイルを含むコンポーネントを展開し、**ファイル**をクリックします。**[ファイル]** ビューが開きます。
3. フォント ファイルを右クリックして、**プロパティ**をクリックします。**[プロパティ]** ダイアログ ボックスが開きます。
4. **[フォント タイトル]** ボックスで、**FontName (FontType)** の形式で、フォントタイトルを入力します (例、**Roman (All res)**)。
5. **[OK]** をクリックします。

コンポーネント ウィザードの [コンポーネント タイプ] オプションを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

特殊なインストール要件を持つファイルがある場合、コンポーネントウィザードの **[タイプを選択してコンポーネントを定義する]** オプションを使用して、これらのファイルをコンポーネントにまとめることをお勧めします。

タスク ***コンポーネント ウィザードを起動するには、以下のいずれかを実行します。***

- インストール プロジェクトのみ: [**セットアップのデザイン**] ビューで機能を右クリックして、**コンポーネントウィザード**を選択します。
- インストール プロジェクト、DIM プロジェクト、およびマージ モジュール プロジェクトの場合: **[コンポーネント]** ビューで **[コンポーネント]** エクスプローラーを右クリックしてから、**[コンポーネント ウィザード]** を選択します。

コンポーネント ウィザードの**タイプを選択してコンポーネントを自分で定義する** オプションは、次のコンポーネント 他オプションをサポートします。

- COM サーバー
- フォントファイル
- サービスのインストール
- サービスのコントロール

メモ: *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

COM サーバーをインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントを作成する方法はいくつもありますが、COM サーバ (.exe、.dll または .ocx) ファイルをセットアップにインストールするには、コンポーネント ウィザードで COM サーバーのコンポーネントを作成することをお勧めします。

表示されるウィザードのパネルは、前のパネルで選択した内容によって異なります。たとえば、ウィザードが [COM サーバー実行可能ファイル] パネルの登録情報を正常に抽出できた場合は、[クラス] パネルは表示されません。

ヒント: *InstallShield では、64 ビット COM 抽出をサポートします。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

フォントのインストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

さまざまな方法でコンポーネントを作成できますが、インストールにフォント (.ttf、.fon、または.ttc ファイル) をインストールする最も簡単な方法は、コンポーネントウィザードでフォントコンポーネントを作成することです。

コンポーネントウィザードでは、各パネルで行う設定によって、次に表示されるパネルが異なります。たとえば、[インストールされているフォントの追加] パネルで **"このシステムにインストールされていないフォントを追加する"** を選択しなかった場合は、[新規フォントの追加] パネルは表示されません。

他のプロジェクトへコンポーネントをエクスポートする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***コンポーネントを別のプロジェクトへ保存するには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. コンポーネントを右クリックして、**コンポーネントのエクスポートウィザード**を選択します。

ヒント: *コンポーネントのエクスポート ウィザードは、InstallShield のプロジェクト メニューからアクセスできます。*

コンポーネントを他のプロジェクトにエクスポートする場合、このコンポーネントのコピーが、ファイル、ショートカット、レジストリ エントリ、詳細設定など、すべてのコンポーネントのデータと共に、指定した ism ファイルに追加されます。ダイアログで使用されるすべての文字列テーブル、プロパティ、およびパス変数も新規プロジェクトにコピーされます。

対象の .ism ファイルが既に異なるプロパティで同じ名前のコンポーネントを持つ場合は、[競合の解決] ダイアログ ボックスが表示され、競合を解決するオプションを選択できます。

プロジェクトのコンポーネントを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューおよび[セットアップのデザイン] ビューの両方で、コンポーネントを永久に削除することできます。

タスク ***コンポーネントを削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. コンポーネントを右クリックして、[**プロジェクトから削除**] をクリックします。

コンポーネントはこれでプロジェクトから永久に削除されます。

コンポーネントと機能の関連付け

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

コンポーネントを必要に応じて任意の数の機能またはサブ機能に関連付けることができます。たとえば、エディターとスペルチェックの 2 つの機能を持つテキストエディターを例とします。エディターとスペルチェックは両者ともシステム .dll コンポーネントに依存しています。このインストールを設計する場合には、システム .dll コンポーネントをエディター機能とスペルチェック機能の両方に関連付ける必要があります。

[セットアップのデザイン] ビューで、コンポーネントを機能に関連付けることができます。[セットアップのデザイン] ビューでは、コンポーネントが既に存在するかしないによって、コンポーネントを機能に関連付ける方法が 2 つあります。詳細は、次を参照してください:

- 新しいコンポーネントを機能に関連付ける.
- 既存コンポーネントを機能に関連付ける.

コンポーネントを関連付けられた機能から削除する手順は、「コンポーネントと機能の関連付けを解除する」を参照してください:

メモ: *機能に関連付けられていないコンポーネントは、次の親のないコンポーネントのアイコンで表示されます:*

新しいコンポーネントを機能に関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

タスク ***新しいコンポーネントを機能に関連付けるには、以下の手順に従ってください。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. 機能またはサブ機能を右クリックして、**[新しいコンポーネント]** または **[コンポーネント ウィザード]** をクリックします。InstallShield により新しいコンポーネントが作成され、それが選択された機能に関連付けられます。

既存コンポーネントを機能に関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

タスク ***既存のコンポーネントを機能に関連付けるには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. 機能、またはサブ機能を右クリックして **[コンポーネントを挿入]** をクリックします。
3. この機能に関連付けたいコンポーネントをリストから選択して **[OK]** をクリックします。

ある機能から別の機能へドラッグアンドドロップを使って既存のコンポーネントを移動またはコピーすることができます。

- 既存のコンポーネントを移動するには、コンポーネントをある機能から別の機能へドラッグします。
- 既存のコンポーネントをコピーするには、CTRL を押しながら、ある機能から別の機能へコンポーネントをドラッグします。

コンポーネントと機能の関連付けを解除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

タスク ***コンポーネント－機能の関連付けを削除するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. 機能との関連を解除するコンポーネントを右クリックしてから、**[機能から削除]** を選択します。

コンポーネントとこの機能の関連が削除されても、コンポーネントはプロジェクト中に存在しています。

データをコンポーネントに追加する

[コンポーネント] ビューを使って、ファイル、レジストリ データ、およびショートカットをコンポーネントに追加できます。

このセクションには、次のトピックが含まれています:

- ファイルをコンポーネントに追加する
- コンポーネントのファイルの [リンク] 列で値を変更する
- コンポーネントからファイルを削除する
- レジストリ データをコンポーネントに追加する
- [コンポーネント] ビューでショートカットを作成する
- 静的にリンクされたコンポーネントにサブフォルダーを追加する

ファイルをコンポーネントに追加する

コンポーネントの中にあるすべてのファイルは、同じコンポーネント設定を共有する必要があります。

コンポーネントへファイルを追加する際の条件

以下の条件を満たしている場合のみ、コンポーネントにファイルを追加することができます。

- すべてのファイルのインストール先フォルダーが同じであること。
- すべてのファイルを同じ条件 (オペレーティング システム、言語を含む) でインストールすること。

この条件を満たしていない場合は、ファイルのインストールのニーズやコンポーネントプロパティに合わせて新しいコンポーネントを作成します。また、ファイルをコンポーネントに追加する際は、セットアップ ベスト プラクティスにも注意する必要があります。セットアップベストプラクティスを使うと、新しくインストールを作成したり、再使用可能コンポーネントを効果的に配布しやすくなります。

コンポーネントへのファイルの追加方法

コンポーネントへファイルを追加する方法は、コンポーネントの作成の仕方によって 多少の違いがあります。ファイルをコンポーネントに関連付ける方法を、コンポーネントの作成方法別に、以下に説明します。

InstallShield で新しいコンポーネントを作成する

[セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビューで右クリックしてコンポーネントを作成する場合は、まずコンポーネントの [ファイル] の項目を選択してファイルリストを表示してください。

タスク ***次に、以下のいずれかの方法で、ファイルをコンポーネントに関連付けます。***

- Windows エクスプローラーからファイル リストにファイルをドラッグアンドドロップする。
- ファイルリスト内を右クリックして **[追加]** をクリックする。表示されるダイアログ ボックスで参照し、そのフォルダーから追加するファイルを必要なだけ選択してください。**[開く]** をクリックします。

コンポーネントウィザードを使用したコンポーネントの新規作成

コンポーネント ウィザードを使用してコンポーネントを作成する際、[ファイル] パネルでコンポーネントにファイルを追加することができます。

タスク ***次に、以下のいずれかの方法で、ファイルをコンポーネントに関連付けます。***

- Windows エクスプローラーからファイル リストにファイルをドラッグアンドドロップする。
- ファイルリスト内を右クリックして **[追加]** をクリックする。表示されるダイアログ ボックスで参照し、そのフォルダーから追加するファイルを必要なだけ選択してください。**[開く]** をクリックします。

InstallShield はアプリケーションのファイルにリンクを作成します。リンクは、インストールのリリースをビルドする際、ファイルを探すのに使用されます。リンクが無効になると、"ファイルが見つかりません" がファイルの横に表示されます。

ファイルを追加する上記の方法はすべて、ファイルを スタティック (静的)にリンクします。これは、新しいファイルを追加したり、ソースフォルダーから除去しても、ファイルのリンクは変わらないということを意味します。スタティック リンクに代わる方法は、「ダイナミック ファイル リンクをコンポーネントに追加する」を参照してください。

コンポーネントのファイルの [リンク] 列で値を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューで、コンポーネントを拡張して[ファイル] ノードを表示することができます。[ファイル] ノードを表示して、コンポーネントに関連付けられているファイルを見ることができます。[ファイル] エクスプローラーの、[リンク] 列はディレクトリまたはファイルを追加したときに使用したパス変数を提供します。[ダイレクトエディター] を利用してこの列に表示される内容を変更することができます。

タスク ***ディレクトリを編集するには、以下の手順を実行します。***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、**[ファイル]** をクリックします。
3. ファイルを参照して **ISBuildSourcePath** 列の値を編集します。

コンポーネントからファイルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***コンポーネントからファイルを削除するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、削除するファイルを含むコンポーネントを展開してから、その下の **ファイル**アイテムをクリックします。コンポーネントに関連付けられているファイルは、右のペインに表示されます。
3. 削除するファイルを右クリックして、[**削除**] をクリックします。

レジストリ データをコンポーネントに追加する

[レジストリ] ビューを使用して、レジストリキーとレジストリ値をコンポーネントに追加することができます。この情報は、コンポーネントがインストールされると、ターゲット システムのレジストリに書き込まれます。レジストリ キーと値の追加方法については、次を参照してください。

- レジストリ キーの作成
- レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する
- レジストリ値の作成
- レジストリ ファイルをインポートする
- レジストリ ファイルのエクスポート

[コンポーネント] ビューでショートカットを作成する

プロジェクト: *インストール プロジェクトでは、[ショートカット] ビューでショートカットを作成できます。*

ショートカットを作成する前に、まずショートカットをリンクさせるファイルが入ったコンポーネントを作成する必要があります。

タスク ***新しいショートカットを作成するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、作成するショートカットに関連付ける必要のあるコンポーネントを展開したあと、**[ショートカット]** をクリックします。**[ショートカット]** エクスプローラーが新しいペインに開きます。
3. **[ショートカット]** エクスプローラーで、インストール先フォルダーを右クリックしてから、**[新しいフォルダー]** または **[新しいショートカット]** をクリックします。たとえば、会社名の下にショートカットを表示したい場合は、プログラム フォルダーを作成することができます。

   ショートカット用のフォルダーを作成した後、そのフォルダーを右クリックし **[新しいショートカット]** をクリックして、ショートカットを作成します。
4. ショートカットのプロパティ を定義します。

静的にリンクされたコンポーネントにサブフォルダーを追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

タスク ***静的にリンクされたコンポーネントにサブフォルダーを追加するには以下の手順に従います:***

[コンポーネント] または **[セットアップのデザイン]** ビューでコンポーネントの **[スタティック ファイル リンク]** サブノードを右クリックして、**[新しいフォルダー]** を選択します。

コンポーネントのキーファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

コンポーネントのファイルの 1 つはキーファイルとなります。キーファイルは、ターゲットマシン上にコンポーネントが存在するかどうかを検出し、またそのアップデートが必要かどうかを決定するために Windows Installer が使用するファイルです。コンポーネントの詳細設定やショートカットを作成するには、キーファイルを指定する必要があります。

このセクションには、次のトピックが含まれています:

- コンポーネントのキーファイルを設定する
- コンポーネントからキー ファイルをクリアする

注意 *・プロジェクトにダイナミック リンクがあるファイルが含まれている場合、それらのファイルに一部がコンポーネントのキー ファイルとして自動的に設定される場合があります。詳細については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。*

コンポーネントのキーファイルを設定する

プロジェクト *・この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***コンポーネントのファイルの 1 つをキーファイルに指定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開し、**ファイル**をクリックします。
3. ファイル リストを右クリックして、**[キー ファイルの設定]** をクリックします。そのファイルのファイルアイコン ( ) が、キーアイコン ( ) に置換されます。

コンポーネントは、1 つのキー パス、または、1 つのキーフ ァイルを持つことができます。キーパスまたはキーファイルがコンポーネントに既に割り当てられている場合、他のキーファイルを設定しようとすると、警告メッセージ ボックスが表示されます。既存のキーファイルまたはキーパスを新規キーファイルで置き換えるには、このメッセージ ボックスで [はい] をクリックします。

メモ *・[ファイルとフォルダー] ビューのキーファイルは指定できません。キーファイルは、[セットアップのデザイン] ビューまたは [コンポーネント] ビューのいずれかで設定できます。*

コンポーネントからキー ファイルをクリアする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

このファイルをコンポーネントのキーファイルとして使用しない場合、コンポーネントからキー ファイルをクリアすることができます。

タスク ***コンポーネントからファイルをクリアするには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開し、**ファイル**をクリックします。
3. キー ファイルを右クリックして [**キー パスのクリア**] をクリックします。

コンポーネントの設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントを作成すると、コンポーネントをビューで作成したかウィザードで作成したかによって、デフォルトの値が設定されます。

タスク ***コンポーネントの設定を編集するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成を行うコンポーネントをクリックしてから、その設定を必要に応じて変更します。

各設定についての詳細は、「コンポーネントの設定」を参照してください。

このセクションには、次のトピックが含まれています:

- コンポーネントのインストール先と機能のインストール先の違い
- アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する
- コンポーネント条件を構成する
- 再インストール中にコンポーネント条件を再評価する
- 共有ファイルの参照カウントを管理する
- インストール前にファイルのバージョンを確認する
- 同じ名前のファイルをインストールする
- コンポーネントの "リモート インストール" 設定を構成する
- ビルド時に COM 登録データを抽出する
- .NET 依存関係とプロパティを確認するためのスキャン
- .NET アプリケーション ファイルを指定する
- .NET Installer クラスへ渡されたプロパティを読み込む
- レジストリ リフレクションの有効化と無効化
- コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する

コンポーネントのインストール先と機能のインストール先の違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

DIM、マージ モジュール、および MSM データベース プロジェクトは機能を含みません。これらのプロジェクトの種類は、基本の MSI プロジェクトの機能に追加されます。したがって、DIM、マージ モジュール、および MSM

データベース プロジェクトのコンポーネントは、それらを使用するインストール プロジェクトに含まれる機能に関連付けられます。

コンポーネントおよび機能 のどちらにも "インストール先" 設定がありますが、エンド ユーザーがインストールを実行した時に利用される方法が異なります。

エンドユーザーによるインストール先の変更を許可する

機能のインストール先設定を使って、ユーザーによるターゲット システムでの機能のインストール先変更を可能にできます。機能のインストール先はエンド ユーザーによって変更されるため、機能のインストール先にはパブリック プロパティを利用しなくてはなりません。パブリック プロパティの名前には、大文字しか使用できません (例、INSTALLDIR)。

メモ: *機能のすべてのコンポーネントを機能のインストール先にインストールするには、コンポーネントのすべてのインストール先が機能のインストール先と一致するように設定する必要があります。*

特定のインストール先を設定する

コンポーネントをエンドユーザーによる変更が不可能な特定の場所にインストールする場合、コンポーネントのインストール先を機能のインストール先として利用されないディレクトリに設定しなくてはなりません。

コンポーネントのファイルのインストール先フォルダーを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

各コンポーネントのファイルに対し、異なるインストール先を指定できます。"インストール先" 設定のデフォルト値は、次のようになります:

- 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合は INSTALLDIR で、製品の INSTALLDIR プロパティでの初期値は [ProgramFilesFolder]Company Name¥Product Name です。
- InstallScript および InstallScript オブジェクトプロジェクトの場合は TARGETDIR で、**OnFirstUIBefore** イベント ハンドラー関数のデフォルトのスクリプト コードでのデフォルトの初期値は PROGRAMFILES ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME です。

*Windows ロゴ・Windows ロゴ要件によると、アプリケーションのファイルのデフォルトのインストール先は、エンド ユーザーの Program Files フォルダーのサブフォルダーである必要があります。*INSTALLDIR *または* ProgramFilesFolder *を機能の "インストール先" 設定の親フォルダーとして使用すると、ファイルは正しい場所にインストールされます。*

タスク ***コンポーネントのインストール先フォルダーを変更するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. インストール先を変更するコンポーネントを選択します。
3. **"インストール先"** 設定で、一覧からオプションの 1 つを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。

インストール先フォルダーに関するその他の考慮事項

プロジェクト・*この情報は、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの "リモート インストール" 設定

コンポーネントの "リモート インストール" 設定を [ソースを優先] (または、コンポーネントの機能が [ソースを優先] に設定されている場合は [オプション]) に設定すると、コンポーネントの "インストール先" 設定に関わらず、コンポーネントのファイルはターゲット システムにインストールされません。

機能の "リモート インストール" 設定

各機能にも "インストール先" 設定があります。異なる場合、コンポーネントの "インストール先" 設定は、機能のインストール先をオーバーライドします。機能の "インストール先" 設定はオプションですが、コンポーネントの "インストール先" 設定は必須です。

デフォルトのインストール先としての INSTALLDIR の使用

すべての機能およびコンポーネントに対し [インストール先] 設定のデフォルトが INSTALLDIR に設定されているのは、すべてのアプリケーションのファイルを同じルートフォルダーにインストールするためです。この結果、エンドユーザーがある機能のインストール先フォルダーを Custom Setup ダイアログで変更すると、INSTALLDIR に含まれるパスにインストールするよう設定されたすべての機能のインストール先フォルダーも変更されます。

メモ: *コンポーネントのインストール先が* INSTALLDIR *以外に設定されていると、コンポーネントは各コンポーネントに指定されたインストール先にインストールされるため、*INSTALLDIR *を変更してもコンポーネントのインストール先には影響がありません。*

INSTALLDIR *などのインストールフォルダープロパティは、デフォルト値として指定されます。エンドユーザーは、*Msiexec.exe *起動時に、コマンドラインでプロパティを設定したり、*CustomSetup *ダイアログで機能に対して新しいインストール先フォルダーを選択することにより、この値を変更できます。*

.NET アセンブリを含むコンポーネント用の [GlobalAssemblyCache]

コンポーネントに .NET アセンブリが含まれる場合、コンポーネントのインストール先を [GlobalAssemblyCache] に設定できます。.NET アセンブリがターゲット システムのグローバルアセンブリキャッシュ (GAC) にインストールされている場合、アセンブリにシステムの他のアプリケーションからアクセスできます。

一般に、アセンブリはローカルアプリケーション ディレクトリにインストールすることをお勧めします。こうするとアプリケーション分離が向上します。

メモ: *アセンブリをグローバル アセンブリ キャッシュにインストールするには、アセンブリ コンポーネントの "ビルド時に .NET をスキャン" 設定を [プロパティのみ] または [依存関係とプロパティ] に設定しておく必要があります。*

スクリプトからコンポーネントのインストール先を指定する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

コンポーネントのファイルのターゲットのインストール先をスクリプトから指定すると、エンド ユーザーの入力またはその他の条件に従ってランタイム時にそのインストール先を変更できます。

タスク ***コンポーネントのファイルのターゲット インストール先をスクリプトから指定するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、**[スクリプト定義のフォルダー]** を右クリックし、**[新規フォルダー]** をクリックします。サブフォルダーが、**<NEW VARIABLE N>** というデフォルト名で追加されます (ここで *N* は連続番号です)。このサブフォルダーの名前を **<スクリプト定義フォルダー>** などのように山かっこで囲って任意の文字列に変更できます。
3. 既存のコンポーネントのファイルのターゲットインストール先を指定するには、以下の手順を実行します。
   a. **[編成]** のビュー リストにある **[コンポーネント]** または **[セットアップのデザイン]** をクリックします。
   b. 構成するコンポーネントを選択します。
   c. コンポーネントのプロパティグリッドで "**インストール先**" プロパティの値をクリックし、リストから手順 1 で作成したサブフォルダーの名前を選択します。

   新しいコンポーネントを作成してそのファイルのターゲットのインストール先を指定するには、以下の手順を実行します。

a. **[ファイルとフォルダー]** ビューで、希望のファイルを **[ソース コンピューターのフォルダー]** ペインからステップ 1 で作成したサブフォルダーにドラッグアンドドロップします。デフォルト名に **FilesN** という名前を持つ新しいコンポーネントが作成されます。

b. スクリプトで **FeatureSetTarget** を呼び出して手順 1 で作成したサブフォルダーにターゲット先を割り当てます。下に例を示します。

```
AskDestPath( "" , "XYZ ファイルの場所を選択してください。", svEndUserSelectedPath, 0 );
FeatureSetTarget( MEDIA , "<スクリプト定義フォルダー>" , svEndUserSelectedPath);
```

大文字のディレクトリ識別子とコンポーネントのインストール先

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ディレクトリ識別子を大文字で指定すると、その値がユーザー インターフェイスから設定できる パブリックプロパティになります。エンドユーザーまたは管理者がユーザー インターフェイスまたはコマンドラインからインストール先を変更するには、コンポーネントのインストール先のディレクトリ識別子をパブリック プロパティにする必要があります。

大文字と小文字を混ぜたり、小文字でディレクトリ識別子を指定すると、ディレクトリエントリが プライベートプロパティ として定義されます。プライベートプロパティは、ユーザー インターフェイスから変更できません。

アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

エンド ユーザーが製品をアンインストールするとき、場合によって、コンポーネントのファイル、レジストリ エントリ、ショートカットおよび他のデータがアンインストールされないようにする必要があります。以下は、コンポーネントのデータがアンインストールされないようにする必要がある場合のシナリオ例です。

- 他の製品に使用される可能性があるファイルのアンインストールを防ぎたい。
- コンポーネントにフォントが含まれている。
- コンポーネントが [SystemFolder] にインストールされる。検証規則によると、この場所にインストールされるコンポーネントは、アンインストール時にアンインストールされるべきではないとあります。

コンポーネントのデータをアンインストールするかどうかを制御する設定は、使用しているプロジェクトの種類によって異なります。

タスク ***コンポーネントのファイル、レジストリ エントリ および他のデータをアンインストールするかどうかを指定するには、以下の手順に従います:***

1. **[セットアップのデザイン]** ビュー (インストール プロジェクトのみ) または **[コンポーネント]** ビューで、構成するコンポーネントを選択します。InstallShield の右側に設定が表示されます。
2. 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合― **"パーマネント"** 設定を必要に応じて次のように構成します:
   - エンドユーザーが製品をアンインストールするとき、コンポーネント データがアンインストールされないようにするには、**[はい]** を設定します。
   - コンポーネントデータがアンインストールされるようにする場合、**[いいえ]** を選択します。

   InstallScript および InstallScript オブジェクト プロジェクトの場合 ― **"アンインストール"** 設定を必要に応じて次のように構成します:
   - エンドユーザーが製品をアンインストールするとき、コンポーネント データがアンインストールされないようにするには、**[いいえ]** を設定します。
   - コンポーネントデータがアンインストールされるようにする場合、**[はい]** を選択します。

"パーマネント" 設定と "アンインストール" 設定は、コンポーネントに関連付けられているすべてのデータ型に適用します。これには、ファイル、レジストリ エントリ、ショートカット、XML ファイルの変更および SQL スクリプトが含まれます。

コンポーネント条件を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

試用版と完全版など、同じプログラムの異なるバージョンを作成する場合、コンポーネントの条件付きインストールが役立ちます。トライアル版では、一部の機能を提供しないようにするため、一部のコンポーネントをインストールしません。また、コンポーネントの条件付きインストールによって、ディスク容量を節約することもできます。ターゲット システムにすべてのコンポーネントをインストールするのに十分なディスク容量がない場合、必ずしも必要でないコンポーネントを条件付きでインストールするよう設定できます。

"コンポーネントの条件" 設定を使用すると、ターゲット システムにコンポーネントのデータをセットアップする前にインストールが評価するステートメントを入力できます。このコンポーネントは、条件が False に評価された場合にはインストールされません。条件が True と評価された場合には、インストールする機能が選択されているとして、コンポーネントがインストールまたはアドバタイズされます。

タスク ***プロジェクトのコンポーネントに条件を設定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開きます。
4. 以下のいずれかを実行します。
   - **[条件]** ボックスで、コンポーネントの条件を入力します。
   - **[プロパティ]** リスト、**[演算子]** リスト、および **[追加]** ボタンを使って、条件をビルドします:
     a. **[プロパティ]** リストで、プロパティを選択してから、**[追加]** ボタンをクリックします。InstallShield は、プロパティを **[条件]** 列に追加します。
     b. 条件ステートメントに演算子を含める場合、**[演算子]** リストから演算子を選択してから、**[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。
     c. 条件ステートメントに値を含める場合は、その値を入力します。
5. **[OK]** をクリックします。

重要: *InstallShield は、基本の条件検証を行いますが、条件ステートメントが予定通りの結果を評価することを確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*

再インストール中にコンポーネント条件を再評価する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *MSM データベース*
- *トランスフォーム*

コンポーネントの "条件の再評価" 設定で [はい] を選択すると、Windows Installer は、コンポーネントの条件を再評価します。*再インストールについての条件が True と評価された場合、コンポーネントは再インストールされます。また、条件が最初に False と評価されていた場合、あるいはコンポーネントがインストール用に選択されていなかった場合は、インストールされます。

移行コンポーネント

インストール済みのコンポーネントで、再インストール時に条件が False と評価されたコンポーネントは削除されます。この特別機能によって、再インストール時にコンポーネントを交換することができます。"再評価条件" 設定で [はい] が選択されたコンポーネントは*移行コンポーネント*と考えられます。

Windows XP または Windows Vista のどちらにインストールされるかによって、異なる .dll ファイルを必要とするアプリケーションを例とします。各ファイルにコンポーネントを作成して、各コンポーネントにオペレーティング システムのバージョンをチェックする条件を追加します。Windows Vista 用のコンポーネントの条件の例を以下に示します:

VersionNT=600

WindowsXP 専用コンポーネントの条件の例を以下に示します:

VersionNT<600

製品を Windows XP にインストールする場合、適切なバージョンの .dll インストールされ、Windows Vista バージョンはインストールされません。エンドユーザーが Windows Vista にアップグレードした場合はどうなるのでしょうか。両方のコンポーネントが移行可能であれば、製品の再インストール時に Windows XP 専用コンポーネントはアンインストールされ、Windows Vista 専用のコンポーネントがインストールされます。

Using Transitive Components (Windows Installer ヘルプ ライブラリ)

共有ファイルの参照カウントを管理する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ファイルの参照カウント (*refcount* とも呼ばれます) は、特定のファイルを使用するターゲット システム上にある製品の数です。参照カウントを使って、あるファイルが複数の製品で共有されている場合に、それを共有する製品すべてが削除されるまで、ターゲット システムに保持することができます。

共有ファイルの参照カウントは、次のレジストリ キーに格納されます:

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥SharedDLLs

Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトの両方で、共有ファイルの参照カウントを管理することができます。プロジェクトの種類によって、その機能は多少異なります。

Windows Installer ベースのインストールでの動作

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、キーファイルのコンポーネントの "共有" 設定で [はい] を選択して、キー ファイルが共有されていることを指定できます。インストールがキー ファイルをインストールする場合で、コンポーネントの "共有" 設定で [はい] が選択されているとき、レジストリに参照カウントが既存しない場合はそれが作成され、既存する場合はそのカウントが増加されます。キーファイルのアンインストール中、参照カウントが減少されます。

***Windows ロゴ・** Windows ロゴガイドラインでは、共有ファイルをインストールするときに参照カウントを増加し、アンインストールするときに減少することが必要です(インストールしないように推奨されている) コア コンポーネントファイルは参照カウントされません。*

InstallScript ベースのインストールでの動作

InstallScript および InstallScript オブジェクト プロジェクトでは、コンポーネントを共有としてマークできます。インストールがコンポーネントのファイルをインストールする場合で、コンポーネントの "共有" 設定で [はい] が選択されているとき、レジストリに各ファイルの参照カウントが既存しない場合はそれが作成され、既存する場合はそのカウントが増加されます。コンポーネントのアンインストール中、参照カウントが減少されます。

InstallScript 関数 **GetFileInfo** を FILE_SHARED_COUNT 定数を使って呼び出して、既存ファイルの参照カウントを判断できます。

コンポーネントを共有としてマークする方法

タスク

コンポーネントのキー ファイルを共有として指定する (Windows Installer ベースのプロジェクトの場合) 、または コンポーネントを共有として指定する (InstallScript ベースのプロジェクトの場合) には、以下の手順に従います:

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. **"共有"** 設定で **[はい]** を選択します。

***ヒント・** コンポーネントを常に共有としてマークしなくてはならない一例として、そのコンポーネントのファイルが System フォルダーまたは Common Files フォルダーといった共有ディレクトリにインストールされる場合が挙げられます。*

***メモ・** コンポーネントを共有としてマークしているいないにかかわらず、インストーラーによってコンポーネントのすべてのファイルの参照カウントが増分されますので、ご注意ください。ただし、参照カウントが存在しない場合、この "共有" 設定が [はい] に設定されていない限り、インストールはこれを作成しません。*

インストール前にファイルのバージョンを確認する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript プロジェクトの場合、この機能はコンポーネントの "上書き" 設定によって処理されます。この設定で省略記号ボタン (...) をクリックすると開く [上書き] ダイアログ ボックス で、コンポーネントがターゲット システム上の既存ファイルを上書きするかどうかを指定できます。

コンポーネントの **"上書きしない"** 設定では、ファイルが既にターゲット システムに存在する場合、インストールでファイルを上書きするかどうか指定できます。

- [はい] を選択すると、(ターゲット システムに存在する場合) ファイルは、ファイルのバージョンに関係なく、決して上書きされません。この設定で [はい] を選択すると、ファイルのバージョン規則がオーバーライドされます。
- [いいえ] を選択しても、ターゲット システムのファイルバージョンがインストールされるバージョンより新しい場合、ターゲット システムのファイルは上書きされません。インストールされるバージョンの方が新しい場合、ターゲット システム上のファイルは 上書きされます 。

Windows Installer は、コンポーネントをインストールするかどうか決定する際、コンポーネントの キーファイルの存在を確認します。詳細については、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。

同じ名前のファイルをインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

コンポーネントの "ソースの場所" 設定は、コンポーネントのファイルが圧縮されない場合に、コンポーネントファイルがソース ディスク イメージに格納されるサブフォルダーを識別します。コンポーネントのファイルはリリースイメージでこのフォルダーにコピーされます。

"ソースの場所" 設定の値は必須ではなく、たいていの場合は、空白のままで構いません。ただし値を指定する場合は、有効な Windows フォルダー名を入力する必要があります。

″ソースの場所″設定が使用される例の 1 つは、複数の言語を持つインストールを作成する場合です。こうしたケースでは、複数のファイルに共通の名前をつけておきたい状況も考えられます。各言語用のコンポーネントを作成して、それぞれのファイルに対して必要に応じて ″ソースの場所″ 設定を構成できます。この ″ソースの場所″ 設定を使用することで、同じ名前を持つファイルを上書きしてしまう危険を冒すことなく、ディスク上の異なる場所にコピーすることができます。

たとえば、*German* および *English* という 2 つのコンポーネントを作成したとします。最初のコンポーネントの ″ソースの場所″ 設定には、**GermanVersion** を入力します。2 番目のコンポーネントの ″ソースの場所″ 設定には、**GermanVersion** を入力します。そして *Test.txt* という名前で、内容が少しだけ異なるファイルを、2 つ用意します。各ファイルをコンポーネントに割り当てます。

非圧縮ファイルを使用してインストールをビルドすると、ディスクイメージ上に *GermanVersion* と *EnglishVersion* という名前の異なるフォルダーが作成されます。上記フォルダーに別々のバージョンの Test.txt が、どちらも上書きされることなしにコピーされます。

メモ: *″ソースの場所″ 設定と「インストール先」は異なります。2 つのバージョンのファイルを両方ともエンドユーザーのマシンにコピーするような場合もありますが、通常はファイルに言語でフィルターをかける場合の方が多くなります。*

コンポーネントの ″リモート インストール″ 設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

″リモート インストール″ 設定によって、このコンポーネントのファイルをターゲット システムにインストールするか、または CD-ROM やネットワークサーバーのようなソース メディアから実行するかが決定されます。新規コンポーネントのデフォルト値は [ローカルを優先] です。これは、コンポーネントのファイルがターゲット システムにインストールされることを意味します。

タスク ***この値を変更してコンポーネントのファイルがソース メディアからのみ実行されるようにするには、以下のようにします。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. ″リモート インストール″ 設定で、**[ソースを優先]**を選択します。**[オプション]** を選択すると、このコンポーネントにその機能の **″リモート インストール″** 設定を与えることになります。

コンポーネントの "リモート インストール" 設定は、機能のそれをオーバーライドします。詳細については、「コンポーネントの "リモート インストール" 設定と機能の "リモートインストール" 設定の違い」を参照してください。

***注意** - コンポーネントに Windows サービスが含まれる場合、[ローカルを優先する] オプションを選択します。エンドユーザーは CustomSetup ダイアログを使用して機能のインストール状態を変更することができますが、Windows Installer はサービスをリモート インストールすることはできません。*

コンポーネントの "リモート インストール" 設定と機能の "リモートインストール" 設定の違い

***プロジェクト** - この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *トランスフォーム*

コンポーネントの "リモート インストール" 設定は、常に機能の "リモート インストール" 設定をオーバーライドします。たとえば、コンポーネントの "リモート インストール" 設定が [ローカルを優先する] に設定されていると、機能の "リモート インストール" 設定に関係なくターゲット システムにそのファイルがインストールされます。

また、コンポーネントの "リモート インストール" 設定が [ソースを優先] に設定されている場合には、ファイルは常にソース メディアから実行されます。コンポーネントの機能が、ファイルをソース メディアから実行するかどうかを決定する場合、コンポーネントの "リモート インストール" 設定で [オプション] を選択します。

コンポーネントが複数の機能に関連付けられていて、コンポーネントの "リモート インストール" 設定が [オプション] に設定されているとき、その機能で [ローカルを優先] が設定されてる場合、ファイルはローカルにインストールされます。コンポーネントが [ローカルを優先] または [ソースを優先] に設定されていれば、その設定に従ってファイルがインストールされます。

ビルド時に COM 登録データを抽出する

***プロジェクト** - この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

コンポーネントの "ビルド時に COM 抽出" 設定で [はい] を選択すると、InstallShield はそのコンポーネントを含むリリースをビルドするたびに、コンポーネントのキー ファイルをスキャンして COM 登録データを抽出します。抽出された情報はリリースに配置され、インストールまたはアドバタイズされるときに Windows Installer が COM サーバーを登録できるようにします。コンポーネントによって作成されたすべての必要なレジストリ設定は、この設定で [はい] を選択すると抽出されます。

コンポーネント ウィザードと異なり、ビルドプロセスではプロジェクト (.ism) ファイルに抽出された COM 情報が書き込まれません。代わりに、ビルドし直すたびにアップデートされるというように、動的になります。これはまた、プロジェクトファイルへの書き込みアクセス許可を持たなくても、InstallShield やコマンドラインから既存のリリースを再ビルドできることを意味します。

メモ: *ビルドされたマシンの PATH システム変数は、COM サーバーのリンク先となるすべての .dll ファイルのディレクトリを含むように設定する必要があります。この設定を怠った場合、ファイルの登録は失敗し、COM 情報の抽出が行なわれません。*

ビルドのフィードバック (「出力」ウィンドウおよびビルドログファイルで表示される) では、抽出された登録情報が詳細に表示されます。

ヒント: *InstallShield を 64 ビット システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

競合の解決

"ビルド時に COM 抽出" 設定に [はい] が選択されていても、コンポーネントの [COM 登録] 詳細設定とコンポーネントの [レジストリ] エクスプローラーで、COM 情報を指定することができます。既存の情報は、コンポーネントがインストールされるときに常に登録されます。

エントリが COM 登録の下で検出されると、InstallShield は "ビルド時に COM 抽出" 設定に [はい] が設定されたときにそれらを削除するかどうかを問い合わせます。ビルド時に競合が見つかった場合、動的に取得したデータで上書きされた詳細設定の項目に関して警告メッセージが表示されます。

これらの安全策の他にも、COM 登録の詳細設定とコンポーネントのレジストリ データをチェックすることによって、予定通りのエントリが使用されており、ビルド時に抽出されたエントリと競合していないことを確認することができます。

.NET 依存関係とプロパティを確認するためのスキャン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallShield では、ビルド時にコンポーネントの .NET 依存関係とプロパティをスキャンするかどうかを指定できます。使用するプロジェクトの種類によって、その機能は異なります。

Windows Installer ベースのプロジェクト

基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、"ビルド時に .NET をスキャン" 設定を使って、ビルド時にコンポーネントのキー ファイルについて .NET 依存関係とプロパティをスキャンするかどうかを指定できます。

メモ: *キーファイルが .NET アセンブリ ファイルではない場合、ビルド時のスキャンではコンポーネントのキー ファイルはチェックされません。*

コンポーネントの "ビルド時に .NET をスキャン" 設定で利用可能なオプションは、次のとおりです:

テーブル 6 - Windows Installer ベースのプロジェクトにおける "ビルド時に .NET をスキャン" 設定のオプション

オプション	説明
なし	InstallShield、このコンポーネントのキーファイルをスキャンして .NET 依存関係またはプロパティのチェックを行いません。
プロパティのみ	ビルド時に、このコンポーネントのキー ファイルをスキャンして、.NET 属性をチェックします。必要に応じて、InstallShield は MsiAssembly と MsiAssemblyName テーブルにアセンブリ プロパティを挿入します。
依存関係およびプロパティ	ビルド時に、このコンポーネントのキー ファイルをスキャンして、.NET 依存関係とプロパティをチェックします。必要に応じて、InstallShield は MsiAssembly と MsiAssemblyName テーブルにアセンブリ プロパティを挿入します。さらに InstallShield は、.NET アセンブリが必要とする、不足しているファイル、コンポーネント、およびマージ モジュールをリリースに追加します。

メモ: *アセンブリをグローバル アセンブリ キャッシュにインストールするには、**"ビルド時に .NET をスキャン"** 設定を [プロパティのみ] または [依存とプロパティ] に設定しておく必要があります。*

この設定は、プロジェクトでスタティック スキャニング ウィザードがどのようにファイルをスキャンするかに影響します。

InstallScript ベースのプロジェクト

InstallScript および InstallScript オブジェクト プロジェクトでは、"ビルド時に .NET をスキャン" 設定を使って、ビルド時にコンポーネントのキー ファイルについて .NET 依存関係をスキャンするかどうかを指定できます。

コンポーネントの "ビルド時に .NET をスキャン" 設定で利用可能なオプションは、次のとおりです:

テーブル 7 • InstallScript ベースのプロジェクトにおける "ビルド時に .NET をスキャン" 設定のオプション

オプション	説明
なし	InstallShield は、このコンポーネントのファイルをスキャンして .NET 依存関係のチェックを行いません。
依存関係	InstallShield は、このコンポーネントのファイルをスキャンして .NET 依存関係をチェックします。InstallShieldは、不足している依存関係をリリースに追加します。

MsiAssembly テーブル (Windows Installer ヘルプ ライブラリ)
MsiAssemblyName テーブル (Windows Installer ヘルプ ライブラリ)

.NET アプリケーション ファイルを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

".NET アプリケーション ファイル" 設定は、このコンポーネントがビルド時にスキャン ([ビルド時に.NETをスキャン] 設定による) されるか、スタティック スキャン ウィザード でスキャンされたときに使用されます。スキャナーはコンポーネントのインストール先と一緒にこの設定を使用して、アセンブリの "ファイル アプリケーション" 設定の値を決定します。

.NET アセンブリの "ファイル アプリケーション" 設定を構成するためのスキャンのアルゴリズムは次のとおりです:

1. アルゴリズムをスキャンすると、コンポーネントの "インストール先" 設定をチェックします。この値が [GlobalAssemblyCache] の場合、.NET アセンブリの "ファイル アプリケーション" 設定はヌルに設定されます。
2. コンポーネントのインストール先が [GlobalAssemblyCache] 以外の場合、スキャンのアルゴリズムはコンポーネントの ".NET アプリケーションファイル" 設定をチェックします。この値がヌル値でない場合、この設定の値を使ってアセンブリの "ファイル アプリケーション" 設定が構成されます。
3. コンポーネントのインストール先が [GlobalAssemblyCache] 以外で、".NET アプリケーション ファイル" 設定がヌルの場合、コンポーネントのキーファイルを使用してアセンブリの "ファイル アプリケーション" 設定が構成されます。

.NET Installer クラスへ渡されたプロパティを読み込む

これは Installer クラスの実装し、インストールからInstaller クラスへ渡されるプロパティの読み出し方をデモンストレーションする .NET クラスの例です。

```
using System;
using System.Configuration.Install;
using System.Windows.Forms;
using System.Collections;

namespace MyInstall
{
    ///
    /// Class1 の概要説明。
    ///
    [System.ComponentModel.RunInstallerAttribute(true)]
    public class MyInstallClass : Installer
    {
        public MyInstallClass()
        {
        }
        public override void Install(IDictionary stateSaver)
        {

            base.Install(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }

        public override void Commit(IDictionary stateSaver)
        {
            base.Commit(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }

        public override void Uninstall(IDictionary stateSaver)
        {
            base.Uninstall(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }

        public override void Rollback(IDictionary stateSaver)
        {
            base.Rollback(stateSaver);
            foreach(string strKey in Context.Parameters.Keys)
            {
                MessageBox.Show(strKey + " は " + Context.Parameters[strKey] + "です");
            }
        }
    };
}
```

レジストリ リフレクションの有効化と無効化

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *トランスフォーム*

レジストリ リフレクションを使用して、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期を保つことができます。

メモ: *Windows Installer 4 以降がある 64 ビット システムでのみレジストリ リフレクションがサポートされています。また、Windows Vista 以降と、Windows Server 2008 以降でのみサポートされています。*

エンド ユーザーがレジストリ リフレクションが有効にされたコンポーネントを持つ 64 ビットのアプリケーションをインストールすると、まず Windows Installer によりレジストリの 64 ビット ビューで (関連付けられた) レジストリの変更が行われ、その後リフレクタによりそのレジストリの変更が 32 ビットのレジストリ ビューにコピーされます。同様に、エンド ユーザーが同レジストリのキーまたは値を変更する 32 ビットのアプリケーションをインストールすると、まず Windows Installer によりレジストリの 32 ビット ビューで (関連付けられた) レジストリの変更が行われ、その後リフレクタによりそのレジストリの変更が 62 ビットの [レジストリ] ビューにコピーされます。

コンポーネントが各キーにアクセスする際、Windows Installer が **RegDisableReflectionKey** 関数を呼び出します。この関数は、指定されたキーのレジストリ リフレクションを無効にします。キーのリフレクションを無効にしても、サブキーのリフレクションには影響ありません。

タスク

コンポーネントによって影響が及ぼされるすべての新しいまたは既存のレジストリキーに対してレジストリ リフレクションを有効化するには、次の手順に従います。

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、レジストリ リフレクションの設定を構成するコンポーネントを選択します。
3. レジストリ リフレクションを有効にするには、**[レジストリ リフレクションを無効にする]** 設定を **[いいえ]** にします。これがデフォルト値です。

   レジストリ リフレクションを無効にするには、**[レジストリ リフレクションを無効にする]** 設定を **[はい]** にします。

レジストリ リフレクションの詳しい情報については、MSDN ライブラリの「Registry Reflection」を参照してください。

コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、コンポーネントについて共有コンポーネントのパッチを有効にするかどうかを指定できます。デフォルトでは無効です。

ターゲット システムにインストールされている少なくとも 1 つのパッケージについて、この複数パッケージの共有機能が有効な場合、Windows Installer 4.5 はこれらすべてのパッケージ間でコンポーネントが共有されているものとして処理します。このコンポーネントを共有するパッチがアンインストールされると、Windows Installer はシステム上で、コンポーネントのファイルの最も上位バージョンを共有し続けることができます。

複数パッケージ コンポーネントの共有を行う目的は、1 つまたは複数の他のインストール パッケージによって共有されているコンポーネントを含むパッチのアンインストール中に、ファイルがダウングレードされないように防ぐことです。これによって、そのパッチがアンインストールされた後でも、コンポーネントのファイルの最も上位バージョンがマシン上で保持されます。

メモ: *ターゲット システムで DisableSharedComponent ポリシーが 1 に設定されている場合、Windows Installer はすべてのパッケージについて、この設定を無視します。*

Windows Installer 4.0 以前では、この設定は無視されます。

次の図表は、file.dll と呼ばれるファイルを含むコンポーネントを共有する ABC と XYZ という 2 つの製品の例を示します。エンド ユーザーは、最初に製品 ABC をインストールし、共有コンポーネントの file.dll バージョン 1.0.0.0 がインストールされます。次に、エンド ユーザーは file.dll バージョン 1.1.0.0 を含む製品 XYZ をインストールします。このバージョンは製品 ABC 用にインストールされたファイルよりも上位バージョンであるため、Windows Installer は現在のバージョンを 1.1.0.0 で上書きします。その後、エンド ユーザーが製品 ABC 用のアンインストール可能なパッチをインストールします。このパッチは、file.dll の 1.2.0.0 バージョンを含みます。これは、ターゲット システムに既存するファイルよりも上位バージョンであるため、Windows Installer は現在のバージョンを 1.2.0.0 で上書きします。エンド ユーザーがパッチをアンインストールすると、次の結果のどちらかが発生します:

- 製品 ABC または 製品 XYZ のどちらかについて、"複数パッケージの共有コンポーネント" 設定が [はい] に設定されている場合で、Windows Installer 4.5 が存在するときは、パッチを製品 ABC からアンインストールすると、ターゲット システムでバージョン 1.1.0.0 が復元されます。
- 製品 ABC と製品 XYZ でこの設定の値が [いいえ] の場合、製品 XYZ が 1.1.0.0 を使用するのにも関わらず、ファイルのバージョンは 1.0.0.0 にダウングレードされます。このため、製品 XYZ がバージョン 1.1.0.0 を必要とするときに、XYZ が適切に動作しない可能性があります。

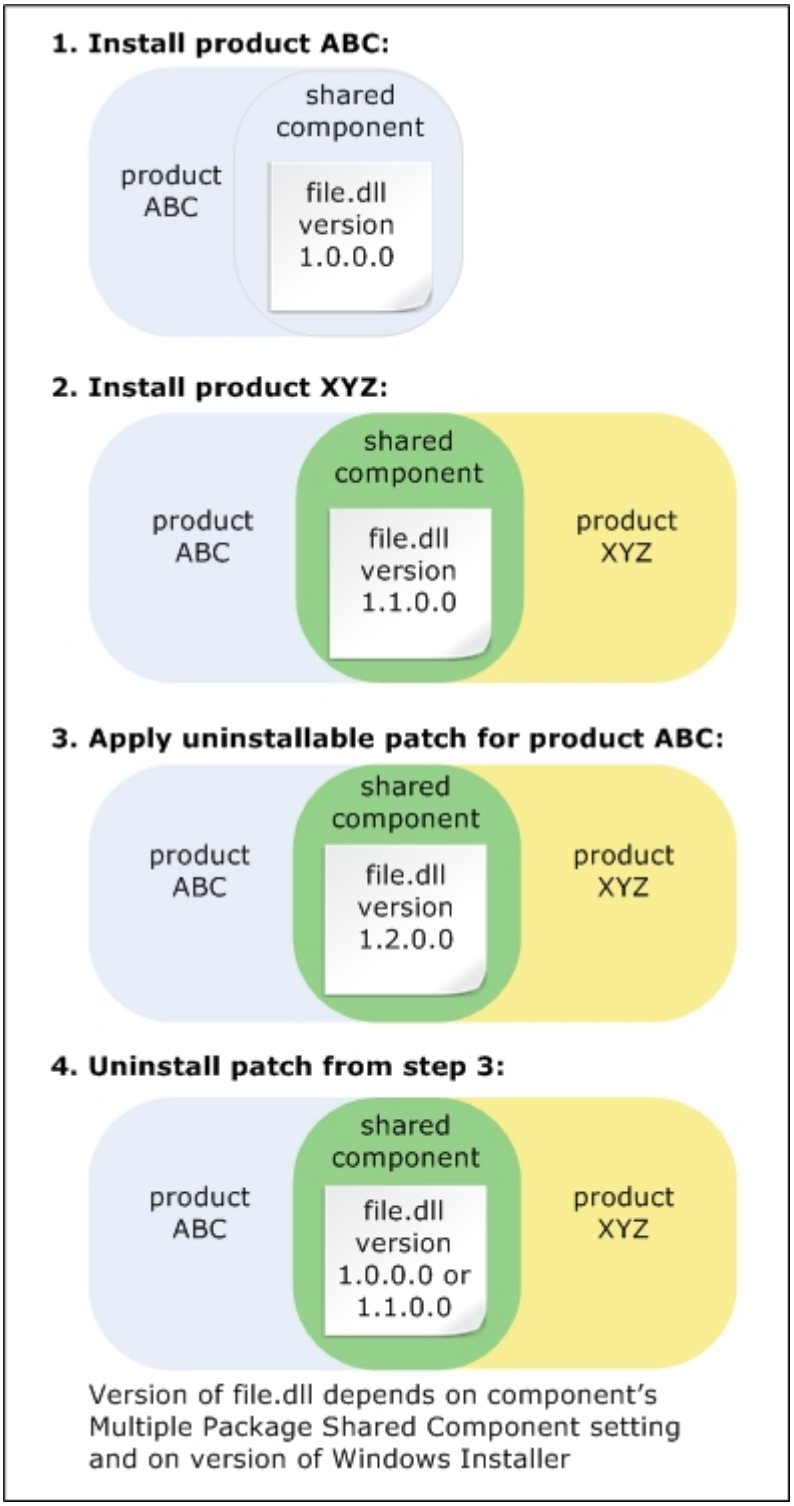


図 1: どちらか 1 つのパッケージで共有コンポーネントの "複数パッケージの共有コンポーネント" 設定が [はい] に設定されている場合で、Windows Installer 4.5 が存在するとき、ファイルのバージョン 1.1.0.0 の復元が可能な場合があります。そうでない場合、ファイルが意図せずにバージョン 1.0.0.0 にダウングレードされる可能性があります。

タスク ***コンポーネントについて、共有コンポーネントのパッチを有効にするかどうかを指定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択します。
3. ***"複数パッケージの共有コンポーネント"*** に適切な値を選択します:
   - 共有コンポーネントのパッチを有効にするには、**[はい]** を選択します。InstallShield が選択されたコンポーネントの msidbComponentAttributesShared 属性を設定して、このコンポーネントが共有されることを示します。
   - 共有コンポーネントのパッチを無効にするには、**[いいえ]** を選択します。InstallShield は選択されたコンポーネントの msidbComponentAttributesShared 属性を設定しません。ただし、コンポーネントが別のパッケージと共有される場合、そのパッケージに含まれる同じコンポーネントに msidbComponentAttributesShared 属性が設定されている可能性があります。その場合、Windows Installer 4.5 は、複数パッケージのコンポーネントが共有されていると見なします。

コンポーネントの詳細設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビュー (および [セットアップのデザイン] ビュー) 内にあるコンポーネントの [詳細設定] 領域では、特定のコンポーネント タイプのインストール条件を満たすことができます。たとえば .ocx ファイルをターゲット システムにコピーする場合、ファイルのメソッドが適切にアクセスされるようにクラス、ProgID、およびタイプ ライブラリを登録する必要があります。詳細設定は Windows Installer のビルトイン機能を利用して COM サーバーの登録、ODBC ドライバー、データソース、およびトランスレーターの設定、Windows サービスのインストール、制御、および構成、並びにファイル関連付けの登録を行います。

詳細設定を指定してコンポーネントのパブリッシュ、COM サーバー、ファイル拡張サーバー、および MIME の種類の登録を行なうことができます。コンポーネントが選択されている場合、コンポーネントがインストールまたはアドバタイズされるときにターゲット システム上で詳細設定が行なわれます。したがって、ファイルはインストールされた直後に実行可能となります。アドバタイズの一種であるコンポーネントのパブリッシュは、パブリッシュの詳細設定を通して行います。

このセクションには、次のトピックが含まれています:

- COM 登録の設定を手動で構成する
- ファイル拡張子を登録する

- アセンブリのインストール
- コンポーネントのアプリケーションパスを指定する
- デバイスドライバー設定を構成する
- コンポーネントをパブリッシュする

COM 登録の設定を手動で構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

重要: *COM サーバーをインストールする場合、Windows Installer でそのクラス、ProgID などを登録する方法をお勧めします。自己登録機能を呼び出すことは、セットアップベストプラクティスの違反になります。*

TypeLib テーブルの使用は避けることが推奨されます。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「TypeLib Table」を参照してください。

COM 登録詳細設定を組み込む最も簡単な方法は、コンポーネントウィザード に必要な情報を抽出させるか、キーファイルの COM データを抽出する方法です。(ビルド時にダイナミック (動的) に抽出することもできます。その場合には、この詳細設定を使用する必要はありません。)コンポーネントのレジストリエクスプローラーで COM 登録詳細設定を変更または COM エントリを作成する作業は、ファイル登録に関する技術詳細に十分精通している開発者向けです。

タスク

COM サーバーを、コンポーネントの詳細設定の編集のみで、インストールするには、以下の手順に従います:

1. COM サーバーが 1 つコンポーネントに追加されていることを確認します。セットアップベストプラクティスによると、ファイルは単一のポータブル実行可能ファイル (.exe、.dll、または.ocx ファイル) である必要です。
2. COM サーバーをコンポーネントのキーファイル にします。
3. コンポーネントの"ビルド時に COM 抽出" 設定を **[いいえ]** に設定します。
4. コンポーネントの**詳細設定**項目を展開して、すべての詳細設定を表示します。
5. **COM の登録** をクリックして、COM サーバー情報を構成します。

[COM 登録] エクスプローラーにあるアイテムの 1 つを右クリックし、COM サーバーの登録情報を変更または作成します。アイテムを右クリックして[名前の変更] をクリックし、新しいアイテムの名前を変更します。

メモ- ビルドされたマシンの PATH システム変数は、COM サーバーのリンク先となるすべての .dll ファイルのディレクトリを含むように設定する必要があります。この設定を怠った場合、ファイルの登録は失敗し、COM 情報の抽出が行なわれません。

作成した各登録アイテムまたはサブアイテムの設定を構成します。この他のヘルプは、各登録設定をクリックすると、InstallShield のヘルプ ペインから見ることができます。

COM クラス、プログラム識別子、および **COM 登録** の構成手順については、次を参照してください:

- COM 登録の COM クラスを手動で構成する
- COM 登録の ProgID を手動で構成する
- COM 登録のタイプ ライブラリを手動で構成する

COM 登録の COM クラスを手動で構成する

***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***クラス ID を登録するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。
3. **[COM 登録]** エクスプローラーで、**[COM クラス]** を右クリックしてから、**[新しい COM クラス]** をクリックします。
4. 必要に応じて、COM クラスの新しい名前を入力します。COM クラスに指定する名前は、デフォルト値として HKEY_CLASSES_ROOT¥CLSID¥<GUID> に登録されます。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。
5. 新しい COM クラスをクリックして、設定の構成を行います。
6. このクラスのコンテキストの種類を指定します。以下のリストに、COM サーバーの種類に適したサーバー コンテキストを示します。
   - **LocalServer32** — 32 ビット .exe ファイル
   - **LocalServer** — 16 ビット .exe ファイル

- **InprocServer32** ― 32 ビット .dll または .ocx ファイル
- **InprocServer** ― 16 ビット .dll または .ocx ファイル

サーバー コンテキストが LocalServer または LocalServer32 の場合、コンテキストをクリックして "デフォルトの Inproc ハンドラー" 設定と "引数" 設定を構成します。

COM 登録の ProgID を手動で構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

progID は、**Component.Class.N** の形式で示される、クラス識別用の文字列です。

タスク ***新しいプログラム上の識別子を指定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。
3. **[COM 登録]** エクスプローラーで、**[ProgID]** を右クリックしてから、**[新しい ProgID]** をクリックします。
4. ProgID の新しい名前を入力します。**Component.Class.N** 形式を使用します。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。
5. 新しい ProgID をクリックして、その設定を構成します。

バージョン非依存のProgID

バージョン非依存の progID はクラスを **Component.Class**の形式で識別する文字列で、クラスのすべてのバージョンに対して一定です。

タスク ***バージョン非依存の新しいプログラム上の識別子を指定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。

3. **[COM 登録]** エクスプローラーで、**[バージョン非依存 ProgID]** を右クリックしてから、**[新しい ProgID]** をクリックします。
4. ProgID の新しい名前を入力します。**Component.Class** 形式を使用します。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。

COM 登録のタイプ ライブラリを手動で構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

重要: *TypeLib テーブルの使用は避けることが推奨されます。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「TypeLib Table」を参照してください。*

タスク ***タイプ ライブラリまたは libID を指定するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを開き、**[詳細設定]** の下で、**[COM 登録]** を選択します。**[COM 登録]** エクスプローラーが別のウィンドウで表示されます。
3. **[COM 登録]** エクスプローラーで、**[タイプ ライブラリ]** を右クリックしてから、**[新しい タイプ ライブラリ]** をクリックします。
4. この COM サーバーによって参照されるタイプライブラリ または libID の新しい名前を入力します。タイプライブラリのアイテムに指定する名前は、デフォルト値として HKEY_CLASSES_ROOT¥TYPELIB¥<libID>¥<version> の下に登録されます。**Component.Class** 形式を使用します。クラスに新しい名前を入力するには、それを右クリックしてから [**名前の変更**] を選択してください。
5. 設定を構成するタイプ ライブラリをクリックします。

ファイル拡張子を登録する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

アプリケーションがファイルの操作に一意のファイル拡張子を使用する場合、[ファイルの種類] 領域で、そのファイル タイプを登録することができます。[ファイルの種類] 領域は、[コンポーネント] ビューおよび [セットアップのデザイン] ビュー (インストール プロジェクトでのみ)内の詳細設定からアクセスできます。たとえば、アプリケーションが .xyz という拡張子を持つファイルを操作する場合、このファイルの種類を登録しておくと、ユーザーがそのアイコンをダブルクリックしたときにファイルをそのアプリケーションで開くようにオペレーティング システムに指示することができます。

この詳細設定は、コンポーネントがインストールまたはアドバタイズされている時に、ターゲット システム上でファイルの種類に関する次の情報を登録します。

テーブル 8 • ファイルの種類情報

登録されている情報	説明
ファイル拡張子	ファイル拡張子 (.doc や .txt など) をコンポーネントのキーファイル に関連付けることができます。
ProgID	ProgID プロパティを拡張子に設定すると、たとえばファイルタイプの登録を含む extfile などに ProgID を指定することができます。
動詞	エンドユーザーが現在の拡張子を持つファイルを右クリックしたときに Windows Explorer が表示するコンテキストメニューの中に表示されるコマンド動詞 (Open や Print など) を登録できます。
MIME タイプ	多目的で、メディアの種類やコンテンツの種類として知られるインターネットメール拡張子 (MIME) タイプもコンポーネントのキーファイルに登録できます。MIMEの種類をクラス ID に関連付けることもできます。

メモ• *ファイルの関連付けは HKLM¥SOFTWARE¥Classes と HKCU¥SOFTWARE¥Classes の両方に格納され、マージされたデータは HKEY_CLASSES_ROOT の下に表示されます。Windows Installer による機能のアドバタイズをサポートする場合、レジストリに直接書き込むのではなく、ファイルの種類エディターを使用されることをお勧めします。*

また、Windows Installer はファイル拡張子アドバタイズ情報を、一見ランダムな文字列としてレジストリに書き込みます。しかし、これは正常な動作です。

アセンブリのインストール

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

メモ: *プロジェクトに .NET アセンブリを追加 する場合、アセンブリをコンポーネントのキー ファイルとして追加して、コンポーネントの "ビルド時に .NET をスキャン" 設定を選択する方法が推奨されます。*

コンポーネントの詳細設定の [アセンブリ] セクションでは、現在のコンポーネントをインストールしたときに、プライベートまたはグローバルの Win32 アセンブリや .NET アセンブリを追加することができます。アセンブリは、システム上の他のアプリケーションに影響を与えない、自己完結型の製品をインストールするときに便利です。

1 つのコンポーネントに複数のアセンブリを含めることはできません。

このセクションには、次のトピックが含まれています:

- アセンブリを追加する
- アセンブリを削除する
- ターゲット システムで .NET アセンブリ サポートをテストする

アセンブリを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

.NET アセンブリをコンポーネントのキーファイルとして追加すると、コンポーネントがビルド時にスキャン されたとき、または スタティック スキャニング ウィザードを実行したときに InstallShield は自動的に .NET アセンブリの設定に値を追加します。

タスク ***コンポーネントに .NET または Win32 アセンブリを追加するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開します。

3. **[詳細設定]** 項目をクリックして展開します。
4. **アセンブリ** を右クリックして、**[新しい Win32 アセンブリ]** または **[新しい .NET アセンブリ]** をクリックします。
5. アセンブリをクリックして、**マニフェスト**、**ファイル** **アプリケーション**および関連する設定を構成します。

InstallShield が、アセンブリに入力された情報を Windows Installer データベースの MsiAssembly および MsiAssemblyName テーブルに追加します。これらのテーブルは、ダイレクトエディターで表示、編集することができます。

アセンブリを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***コンポーネントからアセンブリを削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、コンポーネントを展開します。
3. **[詳細設定]** 項目をクリックして展開します。
4. アセンブリを右クリックして [**削除**] をクリックします。

ターゲット システムで .NET アセンブリ サポートをテストする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ターゲット システムが .NET アセンブリをサポートするかどうかを確認するには、MsiNetAssemblySupport プロパティをテストします。ターゲット システムが Win 32 アセンブリをサポートするかどうかを確認するには、MsiWin32AssemblySupport プロパティをテストします。(Win32 のアセンブリは、Windows XP 以降のみでサポートされています。)アセンブリテーブル、アクションおよびプロパティを使用するには、Windows Installer バージョン 2.0 以降が必要です。

コンポーネントのアプリケーションパスを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[App Paths] レジストリキーは、実行可能ファイルが .dll ファイルを見つけるときに使用するインストール関連レジストリキーで、これがあると PATH 環境変数を変更する必要がありません。実行可能ファイルの App Paths キーは次のような形をしています:

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥App Paths¥*ファイル名*.exe

プログラムの App Paths キーには、通常、"Path" という名前の値が含まれます。ここにはセミコロンで区切られたディレクトリの一覧が含まれており、プログラムの .dll ファイルはこの中から見つけることができます。アプリケーションの場所がシステムのパスに含まれていない場合、Windows ではこのキーを使用してアプリケーションとその .dll ファイルを探します。さらにエンドユーザーがエクスプローラー シェルによってアプリケーションの実行可能ファイルを移動したり名前を変更したりすると、Windows は自動的にファイルの App Paths キーを更新します。

タスク ***コンポーネントのアプリケーションパスを指定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 構成するコンポーネントを選択して、その **[詳細設定]** 項目を展開します。
3. **[詳細設定]** の下の **[アプリケーションのパス]** 項目をクリックします。
4. キーを作成するファイルのチェック ボックスを選択します。
5. **[アプリケーション パス]** 列で、依存関係へのパスを入力するか、パスをハードコーディングしない場合は、Windows Installer フォルダーのプロパティをリストから選択します。複数のパスはセミコロン (;) で区切ります。
6. **[OK]** をクリックします。

デバイスドライバー設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallScript プロジェクトにデバイス ドライバーをインストールするための詳しい情報は、「デバイス ドライバーのインストール」を参照してください。

デバイス ドライバー ウィザード...を実行して、インストールでデバイスドライバーを含めるのに必要なテーブルやエントリ、カスタム アクション、機能、およびコンポーネントを追加した後、デバイスドライバーを含むコンポーネントと関連付けられたプロパティを設定することができます。次の情報は InstallShield 内で設定可能な様々なオプションについて説明します。

[コンポーネント] ビューにあるデバイスドライバー詳細設定の [共通] タブを利用して、現在のコンポーネントにデバイス ドライバーを含めるかどうか、また含める場合、希望のランタイム インストール オプションを選択することができます。[シーケンス] タブでは、プロジェクトのデバイス ドライバー（現在のコンポーネントのデバイス ドライバーだけでなくすべてのデバイス ドライバー）をインストールする順番を指定することができます。

コンポーネントをパブリッシュする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの [パブリッシュ] 詳細設定では、コンポーネントのパブリッシュ情報を指定できます。パブリッシュ（公開）は、アドバタイズ（ジャストインタイム インストール）の一種です。インストール中にコンポーネントのユーザー インターフェイス要素は作成されませんが、コンポーネントは [プログラムの追加と削除] アプレットからインストールすることができます。また、インストールされたコンポーネントがインストーラーからパブリッシュされたコンポーネントを要求した際にもインストールできます。

各アドバタイズ コンポーネントに対し、1 つ以上のコンポーネント ID を作成しなくてはなりません。

重要: *コンポーネント ID と コンポーネントの "コンポーネント コード" で入力する GUID とは異なりますので、ご注意ください。これらは両方とも一意の値でなくてはなりません。"パブリッシュ" 詳細設定で使用するコン*

ポーネント ID は、修飾コンポーネントとしてグループ化された複数のコンポーネントのカテゴリを表すカテゴリ IDです。

各コンポーネント ID には、少なくとも 1 つの修飾子が必要です。修飾子とは、たとえば言語を指定する場合に、この言語またはコンポーネントのバージョンを他から区別するために使用する文字列です。修飾子は、コンポーネントに対して一意である必要があります。

実行時にインストーラーはコンポーネント ID と修飾子を登録して、これらの固有の値を使って、パブリッシュされたコンポーネントを管理します。Windows Installer 関数を呼び出すことにより、インストールされたコンポーネントまたは別のインストールされたコンポーネント（「クロスプロダクト」アドバタイズという名前で知られています）はアドバタイズされたコンポーネントに関する情報を要求し、それをインストールすることができます。

このセクションには、次のトピックが含まれています:

- パブリッシュ情報を指定する
- パブリッシュされるコンポーネントのコンポーネント ID を追加する
- ComponentID を削除する
- componentID へ修飾子を追加する
- componentID から修飾子を削除する
- 修飾子の設定を構成する

パブリッシュ情報を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの [パブリッシュ] 詳細設定では、コンポーネントのパブリッシュ情報を指定できます。パブリッシュ（公開）は、アドバタイズ（ジャストインタイム インストール）の一種です。インストール中にコンポーネントのユーザー インターフェイス要素は作成されませんが、コンポーネントは [プログラムの追加と削除] アプレットからインストールすることができます。また、インストールされたコンポーネントがインストーラーからパブリッシュされたコンポーネントを要求した際にもインストールできます。

タスク ***作成したコンポーネントをパブリッシュするには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]**（インストール プロジェクトのみ）または **[コンポーネント]** をクリックします。
2. エクスプローラーで、パブリッシュするコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。

3. **[パブリッシュ]** エクスプローラーを右クリックし、**[新しい ComponentID]** をクリックして新しいコンポーネントの ID を生成します。一意のコンポーネント ID および対応する修飾子が追加されます。
4. [**パブリッシュ**] エクスプローラーで、新しい修飾子をクリックして、その値を構成します。

インストール プロジェクトの [カスタム アクションとシーケンス] ビューで、ターゲット システムで製品をアドバタイズするために必要な条件を設定できます。

タスク ***インストール プロジェクトでアドバタイズ条件を設定するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[アドバタイズ]** アイテムを展開して、**[実行]** アイテムを展開します。
3. 適切な **[実行]** アクションをクリックして、必要に応じて **"条件"** 設定を構成します。

パブリッシュされるコンポーネントのコンポーネント ID を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントをパブリッシュする場合、少なくとも 1 つのコンポーネント ID を [パブリッシュ] 詳細設定に追加する必要があります。

タスク ***componentID を追加するには、以下の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、新しいコンポーネント ID が必要なコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーを右クリックしてから、**[新しい componentID]** をクリックします。一意のコンポーネント ID および対応する修飾子が追加されます。

ComponentID を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***componentID を削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、componentID を右クリックし **[削除]** をクリックします。

componentID へ修飾子を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

各 ComponentID には少なくとも 1 つの修飾子が必要です。コンポーネント ID を作成すると、デフォルト修飾子が指定されます。この修飾子の名前を変更するには、右クリックしてから、[名前の変更] をクリックします。

タスク ***新しい修飾子を追加するには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、componentID を右クリックし **[新しい修飾子]** をクリックします。InstallShield は、デフォルト名を持つ新しい修飾子を追加します。
4. 修飾子の名前を入力します。

componentID から修飾子を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***修飾子を削除するには、以下の手順に従います:***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、修飾子を右クリックし **[削除]** をクリックします。

修飾子の設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

各修飾子について、*アプリケーション データ*と呼ばれる情報文字列を指定するオプションがあります。

タスク ***修飾子のプロパティを設定するには、次の操作を実行します。***

1. **[編成]** の下にあるビュー リスト、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーでコンポーネントを開き、**[詳細設定]** の下で、**[パブリッシュ]** を選択します。**[パブリッシュ]** エクスプローラーが別のウィンドウで表示されます。
3. **[パブリッシュ]** エクスプローラーで、修飾子をクリックします。

4. **"アプリケーション データ"** 設定に適切な値を入力します。

   この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

機能の定義

機能は、エンド ユーザーから見て、個別にインストール可能な最小の製品構成単位です。これは、ヘルプ ファイルや製品スイートの一部などの製品の特定機能を表し、エンド ユーザーがこれらをインストールまたはアンインストールするかどうかを決定できます。インストール全体は特定の目的を実行する各機能ごとに分ける必要があります。

サブ機能は機能をさらに分割したものです。機能は、ユーザーが選択してインストールできる製品または製品スイートに含まれた自己完結型の要素である必要があります。そのため、インストールの各部分をある "親" 機能のサブ機能として構成するのが、合理的な方法と言えます。

ヒント: *多くのサブ機能を作成できますが、編成のためにデザインを可能な限り単純にしておくようお勧めします。*

このセクションには、次のトピックが含まれています:

- 機能の作成
- 機能の設定を構成する
- 機能のインストール先を設定する
- 機能の条件を設定する
- 機能をエンドユーザーへ表示する
- 機能の条件付き選択
- 機能の条件付き非表示
- 条件付きで実行時にスイート機能の脆弱性を設定する
- 機能のインストールを必須にする
- 機能のアドバタイズ
- 機能の "インストール レベル" 設定を構成する
- 機能の "リモート インストール" 設定を設定する
- 機能にリリース フラグを使用する
- 機能の並べ替え
- "必要な機能" 設定を使用する

機能の作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

[セットアップのデザイン] ビューまたは [機能] ビューを使って、プロジェクトの機能およびサブ機能を作成できます。

タスク

機能を作成するには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. エクスプローラーで最上部のアイテムを右クリックして、[**新しい機能**]をクリックします。InstallShield が、**New Feature_n** (ここで *n* は連続番号です) というデフォルトの名前で新しい機能を追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. 右側のペインで機能の設定を構成します。

プロジェクト: *基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合、機能名には必ずアルファベット、数字、アンダースコア (_)、およびピリオド (.) のみを使用し、名前の先頭にはアルファベット、またはアンダースコアを使用しなくてはなりません。*

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは、次の文字を機能名に使用できません。

*¥ / : * ? ″ ′ < > |*

ヒント: *サブ機能を追加するには、その親機能を右クリックして [新しい機能] をクリックします。*

*新しい機能を追加し、機能の名前に K***Feature 1¥Feature 2¥Feature 3** *と入力することで、ネストされた複数の機能を一度に作成することができます。InstallShield は、***Feature 3** *が* **Feature 2** *のサブ機能であり、Feature 2 が* **Feature 1** *のサブ機能である、ネストされた機能構造を作成します。*

プロジェクトの機能やサブ機能のすべてを作成した後、コンポーネントを作成して、機能と関連付ける必要があります。

機能の設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

タスク ***機能の設定を構成するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. 右側ペインに表示されているグリッド内の設定を構成します。

機能のインストール先を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

各機能およびサブ機能のファイルに対し、別のインストール先を指定できます。新しい機能の "インストール先" 設定のデフォルト値は INSTALLDIR で、これは [一般情報] ビューで設定できます。

Windows ロゴ: *Windows ロゴ要件によると、ターゲット システムの言語にかかわらず、製品のファイルのデフォルトのインストール先を Program Files のサブフォルダー、またはエンド ユーザー のアプリケーション データ フォルダーにする必要があります。*

タスク ***機能のインストール先フォルダーを変更するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. インストール先を変更する機能を選択します。
3. **"インストール先"** 設定で、一覧からオプションの 1 つを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。

インストール先を実行時に構成できるようにするには、選択するインストール先フォルダーに、必ずパブリック プロパティ (すべて大文字で表記) を選択してください。

メモ: *機能の "インストール先" 設定は空白にしておくことができます。*

インストール先フォルダーに関するその他の考慮事項

実行時における機能のインストール先の変更

機能のインストール先ディレクトリを、実行時にエンド ユーザーの機能選択に基づいて変更する場合、Windows Installer カスタム アクション タイプ 35 を利用することができます。

ディレクトリのターゲット先を変更する方法に関する詳細は、Windows Installer ヘルプ ライブラリの「Changing the Target Location for a Directory」を参照してください。

機能の "リモート インストール" 設定

機能の "リモートインストール" 設定を [ソースを優先] (または、サブ機能の親機能が [ソースを優先] に設定されている場合は [親を優先]) に設定すると、機能の "インストール先" 設定に関わらず、機能のファイルはターゲット システムにインストールされません。

"コンポーネントのインストール先" 設定

各コンポーネントにも "インストール先" 設定があります。機能の "インストール先" 設定とコンポーネントの "インストール先" 設定の値が異なる場合、コンポーネントの "インストール先" 設定が機能のそれをオーバーライドします。機能の "インストール先" 設定はオプションですが、コンポーネントの "インストール先" 設定は必須です。

デフォルトのインストール先としての INSTALLDIR の使用

すべての機能とコンポーネントのデフォルトのインストール先プロパティとして INSTALLDIR を設定しているのは、すべてのアプリケーションのファイルを同じルートフォルダーにインストールするためです。この結果、エンド ユーザーがある機能のインストール先フォルダーを CustomSetup ダイアログで変更すると、INSTALLDIR に含まれるパスにインストールするよう設定されたすべての機能のインストール先フォルダーも変更されます。

メモ: *コンポーネントのインストール先が INSTALLDIR 以外に設定されていると、コンポーネントは各コンポーネントに指定されたインストール先にインストールされるため、INSTALLDIR を変更してもコンポーネントのインストール先には影響がありません。*

INSTALLDIR などのディレクトリ プロパティは、デフォルト値を指定します。エンドユーザーは、Msiexec.exe 起動時に、コマンドラインでプロパティを設定したり、CustomSetup ダイアログで機能に対して新しいインストール先フォルダーを選択することにより、この値を変更できます。

機能の条件を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

試用版と完全版など、同じプログラムの異なるバージョンを作成する場合、機能の条件付きインストールが便利です。トライアル版では、一部の機能を提供しないようにするため、一部の機能をインストールしません。また、機能の条件付きインストールによって、ディスク容量を節約することもできます。ターゲット システムにすべての機能をインストールするのに十分なディスク容量がない場合、不要な機能を条件付きでインストールするよう設定できます。

タスク ***プロジェクトの機能に条件を設定するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[機能条件ビルダー]** ダイアログ ボックスが開きます。
4. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
5. **[レベル]** 列に、条件が満たされている場合に機能に使用するインストール レベルを入力します。

   条件が満たされている場合、この値は機能の "インストール レベル" 設定で指定された値を上書きします。
6. 以下のいずれかを実行します。
   - **[条件]** 列で、機能の条件を入力します。
   - **[プロパティ]** リスト、**[演算子]** リスト、および **[追加]** ボタンを使って、条件をビルドします:
     a. **[プロパティ]** リストで、プロパティを選択してから、**[追加]** ボタンをクリックします。InstallShield は、プロパティを **[条件]** 列に追加します。
     b. 条件ステートメントに演算子を含める場合、**[演算子]** リストから演算子を選択してから、**[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。
     c. 条件ステートメントが値を含む場合、"条件" フィールドをダブルクリックしてから END を押し、挿入ポイントを条件ステートメントの終わりに移動して、値を入力します。
7. **[OK]** をクリックします。

重要: *InstallShield は、基本の条件検証を行いますが、条件ステートメントが予定通りの結果を評価することを確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*

機能条件の実行時の動作

機能のデフォルトのインストール レベルは、[機能] ビューまたは [セットアップのデザイン] ビューの "インストール レベル" 設定で構成された値です。この値は、機能の条件が True 評価された場合に上書きされます。その場合、機能のインストール レベルは、True 条件ステートメントに関連付けられている "レベル" 値に設定されます。

各機能のインストール レベル値は、グローバル パブリック プロパティ INSTALLLEVEL の値と比較され、インストール レベル値が INSTALLLEVEL と等しいかそれ以下の機能だけがインストール用に選択されます。

たとえば、ユーザーが昇格された権限を持つ場合のみデフォルトで選択されるようにする機能がある場合、この機能に対して、条件 **Not Privileged** およびインストール レベル **200** を設定できます。エンドユーザーが昇格された権限を持たない場合、この条件が True 評価され、機能にインストール レベル 200 が与えられます。200 は、デフォルトの製品インストール レベル (100) より大きいため、機能は選択されません。

ヒント: *インストール レベルを数値 0 に設定する条件を機能に与えることによって、機能を条件付きで非表示にすることができます。条件が True 評価されると、機能は選択解除されて、CustomSetup ダイアログにも表示されません。*

機能をエンドユーザーへ表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、プロジェクトの種類に対応するカスタム セットアップ ダイアログで、エンド ユーザー向けに機能が表示される方法を指定することができます。

- 基本の MSI、MSI データベース、およびトランスフォーム プロジェクトの場合 – CustomSetup ダイアログ
- InstallScript MSI プロジェクトの場合 – **SdFeatureDialog2**、**SdFeatureMult** または **SdFeatureTree**

[機能] ビューまたは [セットアップのデザイン] ビューにある機能の "表示" 設定を使って、機能を表示するかどうか、およびどのように表示するかを指定できます。選択可能なオプションは以下のとおりです:

テーブル 9 ・ "表示" 設定で使用できるオプション

オプション	説明
閉じて表示する	デフォルトで、実行時ダイアログにサブ機能が閉じた状態で、機能が表示されます。
展開して表示する	デフォルトで、実行時ダイアログにサブ機能が展開された状態で、機能が表示されます。
非表示	実行時ダイアログに、機能およびそのサブ機能は表示させません。

メモ: *この設定で [表示しない] を選択しても、機能がインストールされるかどうかには影響しません。非表示にした機能は、自動的にすべてインストールされるというわけではなく、インストールが必要な機能の場合は選択解除できず、インストールするべきではない機能の場合は選択できないようになります。*

機能の条件付き選択

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

実行時に機能を条件付きで選択する手順は、使用するプロジェクトの種類によって異なります。

基本の MSI、MSI データベース、およびトランスフォーム プロジェクト

機能の [条件] 設定を使って、指定した条件が満たされた場合のデフォルト以外のインストールレベル値を、指定できます。機能のインストールレベル値が、プロジェクトの INSTALLLEVEL プロパティ以下の場合、その機能は選択されてインストールされます。

タスク ***たとえば、管理者権限を持たないエンドユーザーに対しては非選択にしたい機能があれば、次の手順で設定をします。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[機能条件ビルダー]** ダイアログ ボックスが開きます。
4. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
5. **[レベル]** 列で、**200** と入力します。
6. **[条件]** 列に、**Not AdminUser** と入力します。
7. **[OK]** をクリックします。

実行時、エンドユーザーが管理権限を持たない場合(つまり条件が成立すると)、機能の "インストール レベル" プロパティは 200 に設定されます。プロジェクトのデフォルト INSTALLLEVEL プロパティは 100 なので、機能は選択解除されます。

タスク ***INSTALLLEVEL プロパティをプロパティマネージャー で変更することができます。***

InstallScript および InstallScript MSI プロジェクト

FeatureSelectItem 関数を使うと、**SdFeatureTree** などの機能選択ダイアログ ボックスに表示される機能の選択/非選択を設定できます。

機能の条件付き非表示

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

実行時に機能を条件付きで隠す手順は、使用するプロジェクトの種類によって異なります。

基本の MSI、MSI データベース、およびトランスフォーム プロジェクト

インストールレベルをゼロとした機能は、すべて非表示 (かつ非選択) にされます。

タスク

たとえば、インストールの実行エンドユーザーが管理者権限を持たない場合に、特定の機能を非表示にしたければ、次の手順で設定をします。

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"条件"** 設定をクリックして、省略記号ボタン (...) をクリックします。**[機能条件ビルダー]** ダイアログ ボックスが開きます。
4. **[新しい条件]** ボタンをクリックします。InstallShield によって、**"条件"** ボックスの下に新しい条件が追加されます。
5. **[レベル]** 列で、**0** と入力します。
6. **[条件]** 列に、**Not AdminUser** と入力します。
7. **[OK]** をクリックします。

このプロジェクトを再ビルドしてインストールを実行すると、管理者権限を持たないエンドユーザーに対しては、ここで設定した機能は非表示にされインストールもされません。

InstallScript および InstallScript MSI プロジェクト

FeatureSetData 関数は FEATURE_FIELD_VISIBLE 定数を受け取り、これを使って特定の機能を表示するかどうかをコントロールすることができます。たとえば、HiddenFeature という名前の機能を非表示にする場合は、スクリプトに次の関数呼び出しを置きます。

```
FeatureSetData(MEDIA,
   "HiddenFeature",
   FEATURE_FIELD_VISIBLE, FALSE,
   "");
```

メモ: *機能を非表示にしても、その機能の選択は解除されません。機能のデータがインストールされないように選択を解除するには、以下を呼び出します:*

```
FeatureSelectItem(MEDIA, "FeatureName", FALSE);
```

条件付きで実行時にスイート機能の脆弱性を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShield 2018 では、インストール デザイナーの **[機能]** ビューで **"可視"** プロパティの下にある **[条件]** オプションを使って、実行時プロパティに基づいて条件付きで機能を表示または非表示することができます。

"条件" 設定を使って、**InstallationFeatures** ウィザード ページで機能をインストール可能としてデフォルトで表示するかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。

たとえば、Windows の特定バージョンが搭載されたターゲット システム上では特定の機能をデフォルトで表示する場合、その Windows バージョンを指定する条件を作成します。

タスク ***アドバンスト UI またはスイート/アドバンスト UI インストールで条件付きで機能を表示する場合、以下の手順に従います:***

1. **[機能]** ビューで **"表示"** プロパティの下にある **[条件]** 行をクリックします。緑色の + 記号 (**[新しい条件]** ボタン) が行の終わりに表示されます。
2. **[新しい条件]** ボタンをクリックします。**[条件]** 行の下に新しい行が追加されます。
3. **[新しい条件]** ボタンの横にある下矢印をクリックして、一覧から適切なオプション (**[All]**、**[Any]**、または **[None]**) を選択します。
4. 次に同じ行で、**[新しい条件]** ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。

   1 つ以上の条件ステートメントが構成されると、**"条件"** プロパティには **(条件)** と表示されます。何も構成されていない場合、**"条件"** プロパティには **(空白)** と表示されます。

詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

条件付きの表示をサポートするための、オートメーション インターフェイスのメソッド

オートメーション インターフェイスで次のメソッドを使って、実行時に条件付きでスイート プロジェクト内の機能の表示/非表示を設定することもできます:

テーブル 10 ・ スイート プロジェクトで条件付き表示をサポートするオートメーション インターフェイス メソッド

メソッド	構文
AddVisibleCondition	AddVisibleCondition() As ISWiSuiteCondition

テーブル 10 • スイート プロジェクトで条件付き表示をサポートするオートメーション インターフェイス メソッド

メソッド	構文
DeleteVisibleCondition	DeleteVisibleCondition()
VisibleCondition	読み取り専用オブジェクト プロパティ

機能のインストールを必須にする

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

機能の "必須" 設定に [はい] を選択すると、エンド ユーザーは CustomSetup ダイアログ (基本の MSI 、MSI データベース、およびトランスフォーム プロジェクトの場合)、あるいは **SdFeatureDialog2**、**SdFeatureMult**、または **SdFeatureTree** ダイアログ (InstallScript MSI プロジェクトの場合) でそれを選択解除できません。その機能はターゲット システムにインストールされます。

"必須" 設定が [いいえ] に設定されていると、機能はデフォルトでインストールされますが、エンドユーザーは選択を解除できます。

InstallScript MSI プロジェクトでは、"必須" 設定は初回インストール中にルート レベルの機能に適用します。アップグレードまたはパッチに含まれるサブ機能にも適用できます。この設定は、InstallScript MSI インストールの初回インストール中、サブ機能には適用しません。

機能のアドバタイズ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、機能のアドバタイズを選択的に有効または無効にすることができます。アドバタイズされた機能は、インストール過程ですぐにはインストールされません。要求されたときにインストールされます。機能を割り当てると、その機能はすでにインストールされているように見えますが、エンドユーザーが要求するまではインストールされません。(機能を割り当てるとショートカットがインストールされ、コントロール パネルの [プログラムの追加と削除] アプレットからインストールできます。ただし割り当てられた機能は、ユーザーが要求するまではアドバタイズされるだけです。)パブリッシュされた機能は、インストーラーから要求されるまでターゲット システムに表示されません。(パブリッシュされた機能には、エンド ユーザー インターフェイス要素がありません。これらのインストールは、プログラム上で行うか、MIMEの種類を関連付けて行います。)

[機能] ビューの "アドバタイズ" 設定を使って、アドバタイズを許可するかどうかを指定します。この設定で選択できるオプションは、次のとおりです:

テーブル 11 • "アドバタイズ" 設定で使用できるオプション

オプション	説明
アドバタイズを許可する	エンド ユーザーは、CustomSetp ダイアログで、この機能のアドバタイズ オプションを選択できます。アドバタイズは許可されますが、インストール実行時のデフォルトのオプションではありません。
アドバタイズを優先する	機能は、デフォルトでアドバタイズされます。エンド ユーザーは、CustomSetup ダイアログの機能のアドバタイズ オプションを変更できます。
アドバタイズを許可しない	この機能には、アドバタイズは許可されません。エンドユーザーは、CustomSetup ダイアログで機能のアドバタイズを選択できません。
サポートされていない場合にアドバタイズを無効にする	アドバタイズメントは、Internet Explorer 4.01 以上のシステムでのみ機能します。この条件を満たさないターゲット システムでは、アドバタイズは許可されません。ターゲット システムがアドバタイズをサポートする場合、アドバタイズは許可されます。

機能のアドバタイズを有効にすると、アドバタイズを阻止するその他の要素がない限り、インストールが実行されているモードにかかわらず機能はアドバタイズされます。CustomSetup ダイアログで、エン ドユーザーは直ぐにインストールする機能と、後で使用する機能を制御できます。

アドバタイズする場合には、通常、アプリケーションでこの機能をサポートしている必要があります。たとえば、ある製品でスペルチェック機能をアドバタイズするとします。アプリケーションインターフェイスは、メニュー コマンドまたはツールバーボタンから、スペルチェックを提供します。さらに、機能のインストール状態を確認し、カスタマーが [スペルチェック] コマンドまたはボタンをクリックするとそれがインストールされるようにアプリケーションに書き込む必要があります。

機能の "インストール レベル" 設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

*InstallScript MSI プロジェクトでは、この情報は*セットアップの種類*が定義されていない場合にのみ適用します。*

"インストールレベル" 設定は、実行時に INSTALLLEVEL プロパティと比較され、どの機能をインストールできるかを決定します。この設定を使って、特定の機能設定を作成することもできます。

エンド ユーザーが CustomSetup ダイアログで機能を選択解除しない限り、パッケージの INSTALLLEVEL プロパティの値と同等またはそれ以下のインストールレベルの機能がすべてターゲット システムにインストールされます。

***メモ・** パッケージの* INSTALLLEVEL *プロパティをプロパティマネージャーで変更することができます。*

タスク ***機能のインストールレベルプロパティを設定するには、次の操作を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"インストール レベル"** 設定で、この機能のインストール レベルに整数を入力します。推奨値は **100** です。

機能の "リモート インストール" 設定を設定する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

機能の "リモート インストール" 設定によって、この機能のファイルをターゲット システムにインストールするか、または CD-ROM やネットワークサーバーのようなソース メディアから実行するかが決定されます。新しい機能のデフォルト値は "ローカルを優先" です。これは、選択された機能のファイルがターゲット システムにインストールされることを意味します。

タスク ***"リモート インストール" 設定を変更して、機能のファイルがソース メディアからのみ実行されるようにするには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"リモート インストール"** 設定で、**[ソースを優先]**を選択します。

***ヒント・** [**親を優先**] を選択すると、サブ機能にその親機能と同じ値が指定されます。*

***注意・** 機能のコンポーネントに Windows サービスが含まれる場合は、"リモート インストール' 設定で [ローカルを優先する] を選択します。エンド ユーザーは CustomSetup ダイアログを使用してインストール状態を変更することができますが、Windows Installer はサービスをリモート インストールすることはできません。*

機能にリリース フラグを使用する

プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースフラグを使用すると、複数のプロジェクトを作成せずに、異なるバージョンの製品を作成できます。機能のフィルタリングには、リリースフラグに応じて 2 つのステップ (リリース フラグの割り当てと、リリースに含めるフラグの指定) があります。

リリース フラグを機能に割り当てる

プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースフラグは、特定のビルドから除外する機能に設定する必要があります。たとえば、製品の特別版だけに含みたいアドオンという機能がある場合、この機能にフラグを付けて、必要なときだけ含めることができます。

タスク ***リリースフラグを機能に追加するには、次の手順で行ってください。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **"リリース フラグ"** 設定で、文字列を入力します。文字列には、文字または数字を使用できます。複数のフラグをこの機能に付ける場合は、カンマを使用してフラグを区切ります。

リリース フラグに従って機能をフィルターにかける方法については、「リリース フラグ」を参照してください。

リリース フラグを削除する

プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

タスク ***機能からリリースフラグを削除するには、以下の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 構成する機能を選択します。
3. **[リリース フラグ]** 設定で、値を削除します。

機能の並べ替え

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンド ユーザーが製品をインストールするとき、CustomSetup ダイアログ (基本の MSI、MSI データベース、およびトランスフォーム インストールの場合) または機能選択ダイアログの 1 つ (InstallScript または InstallScript MSI インストールの場合) では、InstallShield の [機能] ビューで表示されるのと同じ順序で機能が表示されます。機能が表示される順番を変更することができます。

タスク ***機能の順序を変更するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. 移動する機能を右クリックして、**[上に移動]** または **[下に移動]** をクリックします。機能を右または左に移動したり、それによって別の機能のサブ機能やトップレベルの機能に変更することもできます。

ヒント: *ドラッグアンドドロップ操作で、機能の順序を変更できます。機能またはサブ機能は、すべてこの方法で移動できます。*

"必要な機能" 設定を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

"必要な機能" 設定では、選択された機能に必要な機能を指定できます。たとえば、**ProgramFiles** と **HelpFiles** という 2 つの機能を持つインストールがあると想定します。**HelpFiles** 機能が選択されたときにエンドユーザーが **ProgramFiles** をインストールしなければならない場合は、"必要な機能" プロパティを使用する必要があります。

タスク ***必要な機能を設定するには、次の操作を実行します。***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** または **[機能]** をクリックします。
2. **HelpFiles** 機能を選択します。
3. **必要な機能** プロパティをクリックして、省略記号 (...) ボタンをクリックします。**[必要な機能]** ダイアログ ボックスが開きます。
4. **ProgramFiles** チェック ボックスを選択します。
5. **[OK]** をクリックします。

実行時に、ユーザーがカスタム セットアップタイプを選択すると、**SdFeatureTree** などの機能選択ダイアログは、**HelpFiles** 機能が選択されているときにユーザーが **ProgramFiles** 機能の選択を解除できないようにします。

セットアップの種類について

プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。

セットアップ タイプを使って、エンド ユーザーに様々なバージョンの製品を提供できます。たとえば、デフォルトのセットアップの種類は「完全」と「カスタム」です。「完全」セットアップは、インストールに含まれるすべてのファイルをインストールします。カスタム セットアップの種類は、どの機能をインストールするかエンド ユーザーが選択することができます。

セットアップの種類は機能に基づいています。各セットアップ タイプに関連付ける機能を選択します。次に、エンドユーザーが特定のセットアップの種類を選択すると、そのセットアップの種類に関連付けた機能だけがインストールされます。

デフォルトでは、作成する各プロジェクトは定義済みセットアップの種類 を含みます。[セットアップの種類] ビューで、セットアップの種類を追加/削除したり、既存のセットアップの種類名を変更できるほか、各セットアップの種類に関連付けられている機能を変更することもできます。詳細は、次のセクションを参照してください。

- セットアップの種類を追加する
- セットアップの種類を編集する
- セットアップの種類の名前を変更する
- セットアップの種類を削除する

セットアップの種類を追加する

プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。

タスク ***セットアップの種類を追加するには、次の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーを右クリックし、**[追加]** をクリックします。InstallShieldが、以前のセットアップを新規追加します。
3. セットアップのタイプの新しい名前を入力します。

セットアップの種類を編集する

プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。

タスク ***セットアップの種類に関連付けられている機能を変更するには、次の操作を実行します。***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーで、編集するセットアップの種類をクリックします。インストール プロジェクトに含まれるすべての機能はペインの左下に表示されます。
3. 選択したセットアップの種類に含めない機能の横にあるチェック ボックスを選択解除します。含める機能の横にあるチェック ボックスを選択します。

[セットアップの種類] ペインにリストされているすべてのセットアップのタイプは、自動的にインストール プロジェクトに追加されます。

メモ: **SetupType2** *関数は、それぞれに規定の説明テキストがある標準のセットアップの種類 (完全およびカスタム) のみを表示します。より幅広い柔軟性が必要な場合は、スクリプトで* **SetupType2** *の代わりに* **SdSetupTypeEx** *を呼び出します。(***OnFirstUIBefore** *イベント ハンドラーのデフォルト コードには、***SetupType2** *関数への呼び出しが含まれています。)*

ヒント: *セットアップの種類にアクセラレータ キーを提供する場合、名前の文字の前にアンパサンド (&) を入力してください。たとえば、*Cu&stom *という名前は、*Custom *というラベルになります。エンドユーザーがセットアップ時に S キーを押すことでそのオプションボタンを選択することができるようになります。*

セットアップの種類の名前を変更する

プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。

タスク ***セットアップの種類の名前を変更するには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーで、編集するセットアップの種類をクリックし、**[名前の変更]** をクリックします。
3. セットアップの種類の新しい名前を入力します。

これで、選択したセットアップ タイプの "表示名" 設定が更新されます。この名前は、実行時に [セットアップの種類] ダイアログに表示されます。

セットアップの種類を削除する

プロジェクト: *セットアップ タイプは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" 設定を使用します。

タスク ***セットアップの種類を削除するには、次の手順を実行します。***

1. **[編成]** のビュー リストにある **[セットアップの種類]** をクリックします。
2. **[セットアップの種類]** エクスプローラーで、削除するセットアップの種類をクリックし、**[削除]** をクリックします。

インストールに再配布可能ファイルを含める

InstallShield は、一般によく利用されるサードパーティ再配布可能ファイルを含み、.NET Framework のような技術に対するサポートをインストールに簡単に組み込むことができます。プロジェクトへ再配布可能ファイルを追加すると、再配布可能ファイルおよびすべての関連ファイルがインストールへ追加されます。これによって再配布可能ファイルのパッケージ処理が簡素化され、内部または外部での利用で一貫性を保ちやすくなります。

[再配布可能ファイル] ビュー(InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは [オブジェクト] ビュー) には、すべての InstallShield オブジェクト並びに InstallShield に含まれているサードパーティ マージ モジュールが含まれています。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、このビューにはインストールに追加可能な InstallShield 前提条件も含まれます。InstallScript プロジェクトでは、[前提条件] ビューを使ってインストールに InstallShield 前提条件を追加できます。

InstallShield 前提条件

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

InstallShield は InstallShield 前提条件のベース セットを含みます。また、では InstallShieldInstallShield 前提条件エディターを利用してカスタム InstallShield 前提条件を定義したり、既存の InstallShield 前提条件の設定を編集したりできます。

InstallShield では、次の 2 つのタイプの InstallShield 前提条件がサポートされています:

- **セットアップ前提条件** – この種類の前提条件のインストールは、インストールの実行の前に実行されます。
- **機能前提条件** – この種類の前提条件は、1 つまたは複数の機能に関連付けられています。機能前提条件は、前提条件を含む機能がインストールされたときに、その前提条件がシステム上に既にインストールされていない場合にインストールされます。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

マージ モジュール

マージ モジュール (.msm ファイル) には、個別機能をインストールするために必要なロジックとファイルのすべてが含まれています。たとえば、一部のアプリケーションには、Visual Basic ランタイム ライブラリが必要です。機能にファイルを含めてインストール要件を調べる必要はなく、プロジェクトに含まれる機能の 1 つに Microsoft C++ ランタイム ライブラリを添付するだけでこれを実行できます。

メモ: *[再配布可能ファイル] ビューに含まれているマージ モジュールの多くは Microsoft またはその他のサードパーティによるものです。InstallShieldでは、これらのモジュールを無料配布することによって、インストール プロジェクトの作成を支援します。ただし、サードパーティが作成したモジュールに存在する問題を InstallShield が*

修正したり直すことはできません。サード パーティが作成したモジュールに関する問題は、ベンダーへお問い合わせください。

オブジェクト

オブジェクトには、マージ モジュール同様、個別の機能をインストールするために必要なロジックとファイルがすべて含まれています。InstallShield に含まれる DirectX オブジェクトなどのオブジェクトは、ウィザードを使ってカスタマイズする必要があります。オブジェクトをインストールに追加すると、すぐに該当するカスタマイズ ウィザードが開きます。オブジェクトを追加時点でカスタマイズすることも、ウィザードをいったんキャンセルし、後でオブジェクトを右クリックして [オブジェクトの設定変更] を選択することによってカスタマイズすることもできます。

ライブ再配布可能ファイルギャラリー

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

構成可能マージ モジュール

構成可能な再配布可能ファイルは、ModuleSubstitution テーブル内の少なくとも 1 行によって参照される ModuleConfiguration テーブル内の少なくとも 1 行を含むマージ モジュールまたはオブジェクトです。これによって再配布可能ファイルの値を変更することができます。[再配布可能ファイル] ビューで構成可能モジュールを選択した場合、表示される [マージ モジュール構成可能値] ダイアログ ボックスでモジュール追加時にそれを構成することができます。マージ モジュールを後でカスタマイズするには、それを右クリックして [マージ モジュールの構成]を選択します。

このセクションには、次のトピックが含まれています:

- 再配布可能ファイルの出荷
- 再配布可能ファイルギャラリーを管理する
- InstallShield 前提条件、マージ モジュール、およびオブジェクトをプロジェクトに組み込む
- インストール プロジェクトに含まれている InstallShield 前提条件を利用する
- インストール プロジェクトに含まれているマージ モジュールを使用する
- Windows Installer 再配布可能ファイルをプロジェクトに追加する
- .NET Framework 再配布可能ファイルをプロジェクトへ追加する
- MySQL Connector ODBC 前提条件を含める
- Oracle 11g Instant Client の InstallShield 前提条件を含める
- DirectX 9.0 オブジェクトを含める

ModuleConfiguration テーブル (Windows Installer ヘルプ ライブラリ)
ModuleSubstitution テーブル (Windows Installer ヘルプ ライブラリ)

再配布可能ファイルの出荷

InstallShield は、インストール プロジェクトに組み込むことが可能なサードパーティ再配布可能ファイルを提供します。プロジェクトに、たとえば Crystal Reports といった再配布可能技術を含む場合、その再配布可能ファイルについてベンダーからライセンスが付与されていなくてはなりません。適切なライセンス無しにこれらの技術を再配布することは法的に許可されていません。詳細は、ベンダーのマニュアルを参照してください。

InstallShield プロジェクトでリリースをビルドする時、InstallShield によってビルド出力に様々な InstallShield 再配布可能ファイルが含まれます。InstallShield の使用許諾契約に基づいて、ビルド出力に含まれるこれらの InstallShield 再配布可能ファイルを使用することができます。これらのファイルのほとんどは *InstallShield Program Files フォルダー*¥Redist フォルダーにインストールされ、必要に応じてビルドに含まれます。以下はInstallShield 再配布可能ファイルのリストです。

- _isres_LanguageID.dll
- AppxStub.exe
- ClrSuitePSHelper.dll
- ClrWrap.dll
- CommonHelper.dll
- corecomp.ini
- default.pal
- DLLWrap.dll
- dotnetfx.exe
- DotNetInstaller.exe
- EulaScrollWatcher.dll
- FileBrowse.dll
- IISHelper.dll
- IISRT.dll
- InstallShield.ClrHelper.dll
- InstallShield.Interop.Msi.dll
- ISBEW.exe
- ISChain.exe
- ISChainPackages.dll
- ISComSrv.dll
- ISExpHlp.dll
- isexternalui.dll
- IsLockPermissions.dll
- ISNetAPI.dll
- ISNetApiRT.dll

- ISNetworkShares.dll
- ISRegSvr.dll
- Isrt.dll
- ISScheduledTasks.dll
- IsSchRpl.dll
- ISSetup.dll
- ISSQLSrv.dll
- IsWebDeploy.dll
- ISWindowsFeaturesAction.dll
- ISWindowsFeaturesAction64.dll
- ISXmlCfg.dll
- Layout.bin
- PowerShellWrap.dll
- PrqLaunch.dll
- QuickPatchHelper.dll
- SerialNumCAHelper.dll
- SetAllUsers.dll
- Setup.exe
- Setup.ini
- setup.inx
- setup.isn
- setup.ocx
- setup.skin
- Setup_UI.dll
- setupPreReq.exe
- SetupSuite.exe
- SetupSuite64.exe
- SFHelper.dll
- SQLRT.dll
- SuiteAppxHelper.exe
- SuiteSQLRT.dll
- XMLRT.dll
- 次のフォルダーのサブフォルダーにインストールされるイメージ ファイル

InstallShield Program Files Folder¥Support¥Themes

- 次のフォルダー、およびこのフォルダーの scale-150 および scale-200 サブフォルダーにインストールされるイメージとアイコン ファイル

  InstallShield Program Files フォルダー¥Redist¥Language Independent¥OS Independent

再配布可能ファイルギャラリーを管理する

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

アイコンの種類から再配布可能ファイルのステータスを判断することができます。以下は、再配布可能ファイルビュー内 (または、InstallScript プロジェクトの場合、[オブジェクト] ビューまたは [前提条件] ビュー内) の利用可能なアイコンのリストとそれぞれの説明です。

テーブル 12 • 再配布可能ファイルのアイコン

アイコン	プロジェクトの種類	説明
	基本の MSI、InstallScript、InstallScript MSI	この InstallShield 前提条件はコンピューターにインストールされます。
	基本の MSI、InstallScript、InstallScript MSI	この InstallShield 前提条件ははコンピューターにインストールされず、ダウンロードして利用することができます。
	基本の MSI、InstallScript、InstallScript MSI	この InstallShield 前提条件はプロジェクトに含まれていますが、その場所は [オプション] ダイアログ ボックスの [前提条件] タブ で指定されたディレクトリの 1 つとしてリストされません。
	基本の MSI、InstallScript、InstallScript MSI	このマージ モジュールはコンピューターにインストールされます。
	基本の MSI、InstallScript MSI	このマージンジュールはリポジトリに格納され、プロジェクトに含めることが可能です。リポジトリに関する詳細については、「リポジトリを使用してプロジェクトの要素を共有する」を参照してください。
	基本の MSI、InstallScript、InstallScript MSI	このマージ モジュールはコンピューターにインストールされず、ダウンロードして利用することができます。
	基本の MSI、InstallScript MSI	ご利用のコンピューターにはこのマージ モジュールの古いバージョンがインストールされています。最新版のダウンロードが可能です。
	基本の MSI、InstallScript、InstallScript MSI	このオブジェクトはコンピューターにインストールされます。

テーブル 12 • 再配布可能ファイルのアイコン (続き.)

アイコン	プロジェクトの種類	説明
	基本の MSI、InstallScript MSI	このオブジェクトはコンピューターにインストールされず、ダウンロードして利用することができます。
	基本の MSI、InstallScript MSI	ご利用のコンピューターにはこのオブジェクトの古いバージョンがインストールされています。最新版のダウンロードが可能です。

プロジェクト *利用中のコンピューターにインストールされていない再配布可能ファイルをプロジェクトに追加する場合、リリースをビルドしたときに 1 つまたは複数のビルドエラーが発生します。ビルドエラーを避けるには、プロジェクトから再配布可能ファイルを削除するか、リリースを再ビルドする前にダウンロードします。再配布可能ファイルがコンピューターにインストールされていない場合、その再配布可能ファイルの [場所] 列で* ***[ダウンロードが必要]*** *が指定されます。*

InstallScript プロジェクトに追加する [オブジェクト] ビューの再配布可能ファイルがコンピューターにインストールされていない場合、InstallShield は追加を許可しません。

ライブ再配布可能ファイル ギャラリーについて

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの [再配布可能ファイル] ビューで表示される再配布可能ファイル ギャラリーには、インストールに含むことができるマージ モジュール、オブジェクト、および InstallShield 前提条件が含まれています。また、InstallScript プロジェクトの [前提条件] ビューには、プロジェクトに追加可能な InstallShield 前提条件のギャラリーが表示されます。

InstallShield 前提条件

InstallShield では、多数の InstallShield 前提条件が提供されています。さらに InstallShield のInstallShield 前提条件エディターでは、これらの前提条件を変更したり、独自の前提条件を作成したりできます。すべての InstallShield 前提条件ファイル (.prq) は、次の場所に格納されています:

InstallShield Program Files フォルダー¥SetupPrerequisites

マージ モジュール

マージ モジュールは、さまざまなソースから利用することができます。InstallShield には多くの再配布可能モジュールが含まれていますが、新しいバージョンがある場合や、他のソフトウェア開発会社から必要なモジュールがリリースされている場合があります。さらに InstallShield では、独自のマージ モジュールを作成して再配布可能ファイルギャラリーに追加することも可能です。

[再配布可能ファイル] ビューに一覧表示されているマージ モジュール ファイルのソースは、[オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されたフォルダーです。[オプション] ダイアログ ボックスにアクセスするには、[ツール] メニューから [オプション] を選択します。

次のディレクトリは InstallShield に付属されているモジュールのデフォルトの場所です。

InstallShield Program Files フォルダー¥Modules¥i386

オブジェクト

InstallShield　は多くの再配布可能オブジェクトを提供すると同時に、インストールに含める再配布可能ファイルを独自に作成することも可能です。さらに、他の開発者が InstallShield を使って作成したオブジェクトをプロジェクトに追加することも考えられます。

InstallShield に付属されているオブジェクトのデフォルトの場所は以下の通りです。

InstallShield Program Files フォルダー¥Objects

上記の場所に含まれるオブジェクトは 再配布可能ファイルビューに一覧表示されています。

再配布可能ファイル ギャラリーの管理についての詳細は、次セクションを参照してください:

- 再配布可能ファイルをコンピューターにダウンロードする
- 再配布可能ファイルギャラリーに InstallShield 前提条件を追加する
- 再配布可能ファイルギャラリーから InstallShield 前提条件を削除する
- マージ モジュールを参照する
- マージ モジュールを参照した場合に起きること
- 再配布可能ファイル ギャラリーにマージ モジュールを追加する
- 再配布可能ファイルギャラリーからマージ モジュールを削除する
- InstallScript プロジェクトでオブジェクトを登録する

再配布可能ファイルをコンピューターにダウンロードする

再配布可能ファイルをコンピューターへダウンロードする手順は、使用するプロジェクトの種類によって異なります。

基本の MSI プロジェクトと InstallScript MSI プロジェクトの場合

[再配布可能ファイル] ビューを使用して、最新の InstallShield 前提条件、マージ モジュールとオブジェクトを Revenera Web サイトからコンピューターへダウンロードすることができます。再配布可能ファイルがコンピューターにインストールされていない場合、その再配布可能ファイルの [場所] 列で **[ダウンロードが必要]** が指定されます。

タスク

特定の InstallShield 前提条件、マージ モジュール、オブジェクトをダウンロードするには、以下の手順を実行します:

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. ダウンロードする InstallShield 前提条件、マージ モジュール、または、オブジェクトを右クリックして、**[選択したアイテムをダウンロード]** をクリックします。

タスク ***インストール プロジェクトに必要なすべての InstallShield 前提条件、マージ モジュール、およびオブジェクトをダウンロードするには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. InstallShield 前提条件、マージ モジュール、、またはオブジェクトのどれかを右クリックして **[必要なアイテムをすべてダウンロード]** をクリックします。

InstallScript プロジェクトの場合

[再配布可能ファイル] ビューを使って、最新の InstallShield 前提条件を Revenera Web サイトからコンピューターへダウンロードすることができます。再配布可能ファイルがコンピューターにインストールされていない場合、その再配布可能ファイルの [場所] 列で **[ダウンロードが必要]** が指定されます。

タスク ***特定の InstallShield 前提条件をダウンロードするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. ダウンロードする InstallShield 前提条件を右クリックしてから、**[選択したアイテムをダウンロード]** をクリックします。

タスク ***インストール プロジェクトに必要なすべての InstallShield 前提条件をダウンロードするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. InstallShield 前提条件の 1 つを右クリックしてから、**[必要なアイテムをすべてダウンロード]** をクリックします。

[オブジェクト] ビューを利用して、最新のマージ モジュールとオブジェクトを Revenera Web サイトからコンピューターへダウンロードすることができます。マージ モジュールがコンピューターにインストールされていない場合、インストールされていないことがそのアイコン（ ）によって表示されます。InstallScript プロジェクトに追加するマージ モジュールがコンピューターにインストールされていない場合、InstallShield は追加を許可しません。オブジェクトがコンピューターにインストールされていない場合、[オブジェクト] ビューには表示されません。

タスク ***特定のマージ モジュールをダウンロードするには、以下の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. **[InstallShield オブジェクト/マージ モジュール]** ペインで、ダウンロードするマージ モジュールを右クリックして **[選択したアイテムのダウンロード]** をクリックします。

タスク ***オブジェクトをダウンロードするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. **[InstallShield オブジェクト/マージ モジュール]** ペインで、オブジェクトを右クリックし **[Web をチェック]** をクリックします。Revenera Web サイトの [ダウンロード] ページがインターネット ブラウザーで開きます。
3. ダウンロード/インストールするオブジェクトを選択します。インストール中に、InstallShield を閉じてオブジェクトのインストールを完了するようプロンプトが表示されます。

再配布可能ファイルギャラリーに InstallShield 前提条件を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイル ギャラリーへ InstallShield 前提条件を追加するには、以下の手順を実行します。***

1. 新規または更新された InstallShield 前提条件 (.prq) ファイルを取得します。
2. Windows Explorer を利用して、新しい前提条件を次の場所にコピーします。

   InstallShield Program Files フォルダー¥SetupPrerequisites
3. 現在 InstallShield が開いている場合は閉じます。
4. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) および [前提条件] ビュー (InstallScript プロジェクトの場合) に反映されます。

再配布可能ファイルギャラリーから InstallShield 前提条件を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイル ギャラリーから InstallShield 前提条件を削除するには、以下の手順を実行します。***

1. InstallShield を閉じます。
2. Windows Explorer を利用してギャラリーから削除する InstallShield 前提条件を探して削除します。InstallShield セットアップ前提条件ファイルは、次のディレクトリに格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites
3. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) または [前提条件] ビュー (InstallScript プロジェクトの場合) に反映されます。

マージ モジュールを参照する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクトに追加するマージ モジュールが [再配布可能ファイル] ビューに表示されていない場合、それを検索してプロジェクトおよび [再配布可能ファイル] ビューに追加することができます。

タスク ***マージ モジュールを参照するには、次の操作を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. アイテムを右クリックして、**マージ モジュールの参照**をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. マージ モジュールファイルを参照します。
4. **[OK]** をクリックします。

マージ モジュールを参照した場合に起きること

InstallShield は、マージ モジュールへの参照を明示パスとして維持しません。そのかわり、マージ モジュール GUID とマージ モジュールロケールに基づいてマージ モジュールのキーを生成します。InstallShield がマージ モジュールにアクセスする時、そのキーと一致するファイルを [マージ モジュールの場所] ボックスで指定したフォルダー内で探します。[マージ モジュールの場所] ボックスは、[オプション] ダイアログ ボックスの [マージ モジュール] タブにあります。

マージ モジュールを参照する場合、マージ モジュールを含むフォルダーへのパスがマージ モジュールの場所ボックス内のパスのリストへ追加されます。さらに GUID:ロケールキーが選択されたファイルに基づいてインストール プロジェクトに追加されます。

インストールへの影響

マージ モジュールの場所ボックスにある 2 つのマージ モジュールが GUID:ロケール キーを持つ場合、ファイル名が異なる場合でも 1 つだけがインストールに含まれます。InstallShield による [マージ モジュールの場所] ボックスの検索方法のため、どのマージ モジュールが含まれるかを予期するのは不可能です。

マージ モジュールの場所ボックス内のディレクトリ数の限定

共有マージ モジュールギャラリを使用する場合、ターゲットマシンに存在するバージョンより古いか新しいバージョンのマージ モジュールが存在する可能性があります。このため、マージ モジュールの場所ボックスのディレクトリ数を制限した方が賢明な場合があります。

タスク ***ディレクトリ数を制限するには、次のいずれかを実行します。***

- Windows エクスプローラーを使用して、**[マージ モジュールの場所]** ボックスに既にリストされているフォルダーの 1 つにマージ モジュールをコピーします。
- 検索パスからデフォルトのフォルダーを削除して、共有の場所だけが参照されるようにします。

再配布可能ファイル ギャラリーにマージ モジュールを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイルギャラリーへマージ モジュールを追加するには、以下の手順に従います:***

1. 新規、またはアップデートされたマージ モジュールを取得します。
2. Windows Explorer を利用して、**[オプション]** ダイアログ ボックスの **[マージ モジュール]** タブで指定されたフォルダーの 1 つに新しいモジュールをコピーします。

   InstallShield に付属されているモジュールのデフォルトの場所は以下の通りです:

   InstallShield Program Files フォルダー¥Modules¥i386
3. 現在 InstallShield が開いている場合は閉じます。
4. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) および [オブジェクト] ビュー (InstallScript プロジェクトの場合) に反映されます。

再配布可能ファイルギャラリーからマージ モジュールを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***再配布可能ファイル ギャラリーからモジュールを削除するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. Windows Explorer を利用してギャラリーから削除するマージ モジュールを探して削除します。**[オプション]** ダイアログ ボックスの[マージ モジュール] タブで指定されたディレクトリをすべて検索して下さい。
3. InstallShield を起動します。

変更内容が [再配布可能ファイル] ビュー (基本の MSI および InstallScript MSI プロジェクトの場合) および [オブジェクト] ビュー (InstallScript プロジェクトの場合) に反映されます。

メモ: *インストールに現在関連付けられているマージ モジュールを削除すると* ***[マージ モジュールが見つかりません]*** *というメッセージが表示され、モジュールをインストールに追加できないことを通知します。*

InstallScript プロジェクトでオブジェクトを登録する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクト内からオブジェクトを登録すると、そのオブジェクトが再配布可能ファイル ギャラリに追加され、他のプロジェクトに含めることができるようになります。

タスク ***オブジェクトを登録するには、次の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. 項目を右クリックして **[詳細]** をポイントしてから、**[新しいオブジェクトの登録]** をクリックします。**[プロジェクトの設定]** ダイアログ ボックスが開きます。
3. [言語非依存オブジェクトのプロパティ] タブで、メディア ファイル (Data1.hdr ファイル) を指定します。
4. その他希望の設定を指定して **[OK]** をクリックします。

情報は、次のセクションを参照してください:

- InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する
- InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する

- InstallShield 前提条件をプロジェクトから削除する
- プロジェクトからマージ モジュールとオブジェクトを削除する
- InstallShield 前提条件、マージ モジュールおよびオブジェクトのファイルを決定する

InstallShield 前提条件、マージ モジュール、およびオブジェクトをプロジェクトに組み込む

InstallShield では、InstallShield 前提条件、マージ モジュール、および InstallScript オブジェクトとしてパッケージ化されている多数のサード パーティ再配布可能ファイルが提供されています。これらのビルトイン再配布可能ファイルをインストール プロジェクトに追加することができます。詳しくは、ドキュメントの該当セクションをご覧ください。

ヒント: *InstallShield 前提条件、マージ モジュール、およびオブジェクトを自分自身で作成するときの情報については、「InstallShield 前提条件およびその他の再配布可能ファイルをデザインする」をご覧ください。*

InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*

InstallScript プロジェクトに再配布可能ファイルに追加するときの情報は、「InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する」を参照してください。

2 つの種類の再配布可能ファイル (マージ モジュールおよびオブジェクト) は、インストールされるために機能への関連付けが必要です。マージ モジュールまたはオブジェクトの関連付けを行うとき、関連付けることができる機能やサブ機能の数に制限はありませんマージ モジュールまたはオブジェクトを追加するときにインストール プロジェクトに機能が存在しない場合は、機能を作成できる [新しい機能の作成] ダイアログ ボックスが開きます。機能を作成しない場合、これら 2 種類の再配布可能ファイルはインストール プロジェクトには追加されません。

InstallShield 前提条件を InstallScript MSI プロジェクトに追加した場合、それらを 1 つまたは複数の機能と関連付けることはできません。

InstallShield 前提条件を基本の MSI プロジェクトに追加した場合、その前提条件はメインのインストールが開始される前に実行されるため、デフォルトでどの機能にも関連付けられていません。これらの前提条件は、*セットアップ前提条件*と呼ばれます。必要に応じて、InstallShield 前提条件を現在プロジェクトに存在する 1 つ以上の機能に関連付けることができます。

タスク ***InstallShield 前提条件、マージ モジュールまたはオブジェクトを基本の MSI プロジェクトまたは InstallScript MSI プロジェクトに追加するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. リポジトリへパブリッシュされたマージ モジュールを表示するには、再配布可能ファイルを右クリックして **"リポジトリのマージ モジュールを表示する"** が有効となっていることを確認してください。
3. 追加する再配布可能ファイルの前にあるチェック ボックスを選択します。オブジェクトを選択した場合、関連するウィザードが開いてカスタマイズ処理の手順を案内します。
4. マージ モジュールまたはオブジェクトの場合: **[条件付きインストール]** ペインで、この再配布可能ファイルを含める必要がある各機能のチェック ボックスを選択します。

   基本の MSI プロジェクトで、前提条件を機能と関連付ける場合: **[条件付きインストール]** ペインで、この前提条件を含める必要がある各機能のチェック ボックスを選択します。機能に前提条件を関連付けない場合、**[機能の選択の前にインストールする]** チェック ボックスを選択状態のままにしておきます。このチェック ボックスは、InstallShield 前提条件を基本の MSI プロジェクトに追加したとき、デフォルトで選択されています。

ヒント: *[再配布可能ファイル] ビューの右側のペインに、提供されている再配布可能ファイルの一覧から選択されたマージ モジュール、オブジェクト、または InstallShield 前提条件の詳細が表示されます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。このビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。*

メモ: *プロジェクトに追加した InstallShield 前提条件の [場所] 列で **[ダウンロードが必要]**を指定した場合、その InstallShield 前提条件はコンピューターにインストールされません。InstallShield 前提条件をプロジェクトに含める必要がある場合、コンピューターにインターネットからその前提条件をダウンロードすることができます。1 つまたは複数の InstallShield 前提条件をダウンロードしないでリリースをビルドした場合で、さらに InstallShield 前提条件を* Setup.exe *から抽出する、または (Web からエンド ユーザーのコンピューターにダウンロードするのではなく) ソース メディアからコピーするを指定した場合、1 つまたは複数のビルドエラーが生成される可能性があります。ビルドエラーを除去するには、プロジェクトから InstallShield 前提条件を削除してコンピューターにダウンロードするか、またはリリースの InstallShield 前提条件の場所をダウンロード オプションに変更してから、リリースを再ビルドします。*

InstallShield 前提条件とオブジェクトの取得

一部の InstallShield 前提条件とオブジェクトは、InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

また、インストール時に起動するインストール (例、Setup.exe ファイルまたは .msi パッケージ) がある場合、カスタム InstallShield 前提条件を自分で作成して、それを必要に応じてプロジェクトに追加することができます。InstallShield 前提条件を自分で作成する方法については、「InstallShield 前提条件を定義する」をご覧ください。

InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに再配布可能ファイルを追加する方法については、「InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する」を参照してください。

機能に関連付けられていないマージ モジュールとオブジェクトは、InstallScript インストールの実行時にインストールされません。マージ モジュールまたはオブジェクトの関連付けを行うとき、関連付けることができる機能やサブ機能の数に制限はありませんマージ モジュールまたはオブジェクトを追加するときにインストール プロジェクトに機能が存在しない場合は、機能を作成できる [新しい機能の作成] ダイアログ ボックスが開きます。機能を作成しない場合、これら 2 種類の再配布可能ファイルはインストール プロジェクトには追加されません。

InstallShield 前提条件を InstallScript プロジェクトに追加する場合、それを 1 つまたは複数の機能と関連付けることはできません。

タスク ***InstallShield 前提条件を InstallScript プロジェクトに追加するには、以下の手順を実行します:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 追加する InstallShield 前提条件の前にあるチェック ボックスを選択します。

ヒント: *[前提条件] ビューの右側のペインに、提供されている再配布可能ファイルの一覧から選択された InstallShield 前提条件の詳細が表示されます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。このビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。*

メモ: *プロジェクトに追加した InstallShield 前提条件の [場所] 列で **[ダウンロードが必要]**を指定した場合、その InstallShield 前提条件はコンピューターにインストールされません。InstallShield 前提条件をプロジェクトに含める必要がある場合、コンピューターにインターネットからその前提条件をダウンロードすることができます。1 つまたは複数の InstallShield 前提条件をダウンロードしないでリリースをビルドした場合で、さらに InstallShield 前提条件を (Web からエンド ユーザーのコンピューターにダウンロードするのではなく) メディアに含めることを指定した場合、1 つまたは複数のビルドエラーが生成される可能性があります。ビルドエラーを除去するには、プロジェクトから InstallShield 前提条件を削除してコンピューターにダウンロードするか、またはリリースの InstallShield 前提条件の場所をダウンロード オプションに変更してから、リリースを再ビルドします。*

タスク ***マージ モジュールまたはオブジェクトを InstallScript プロジェクトへ追加するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[オブジェクト]** をクリックします。
2. **[機能]** ペインで、オブジェクトまたはマージ モジュールを追加する機能を選択します。

3. 追加するオブジェクトまたはマージ モジュールを右クリックして、**[選択した機能に追加する]** を選択します。(オブジェクトまたはマージ モジュールを機能にドラッグアンドドロップすることもできます。)一部のオブジェクトについては、関連したウィザードが表示されるので、これに従ってカスタマイズします。

メモ: *InstallScript プロジェクトにある機能に追加されたマージ モジュールは、マージ モジュール ホルダー オブジェクトのサブアイテムとして機能ペインに表示されます。このオブジェクトには、Windows Installer エンジンが必要です。詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。*

ヒント: *インストールされるファイルや実行されるその他の操作など、オブジェクトについての詳細を表示するにはオブジェクト名またはマージ モジュール名を選択します。右側のペインに情報が表示されます。*

注意 - *他の企業のマージ モジュールは変更しないでください。*

オブジェクトを取得する

一部のオブジェクトは InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

InstallShield 前提条件をプロジェクトから削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***InstallShield 前提条件をプロジェクトから削除するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下のビュー リストで **[再配布可能ファイル]** (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合)、または **[前提条件]** (InstallScript プロジェクトの場合) をクリックします。
2. 削除する InstallShield 前提条件の前にあるチェック ボックスをクリアします。

ヒント: *InstallShield 前提条件が別の InstallShield 前提条件の依存関係としてインストールに含まれているが、インストールから依存前提条件を削除したい場合、対応する依存関係を InstallShield 前提条件から削除する必要があります。詳細については、「InstallShield 前提条件の依存関係を削除する」を参照してください。*

プロジェクトからマージ モジュールとオブジェクトを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***プロジェクトからマージ モジュールやオブジェクトを削除するには、以下の手順に従います:***

- 基本の MSI および InstallScript MSI プロジェクトの場合: **[再配布可能ファイル]** ビューで、再配布可能ファイルの前にあるチェック ボックスをクリアします。
- InstallScript プロジェクトの場合: **[オブジェクト]** ビューで、**[機能]** ウィンドウの再配布可能ファイルを右クリックして、**[プロジェクトから削除]** をクリックします。

プロジェクトからマージ モジュールやオブジェクトが移動されます。また、再配布可能ファイルに関連付けられている依存関係もすべて自動的に削除されます。

InstallShield 前提条件、マージ モジュールおよびオブジェクトのファイルを決定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件、マージ モジュール、オブジェクト内にあるファイルを確認するには、基本の MSI および InstallScript MSI プロジェクトの [再配布可能ファイル] ビューでそれらを表示できます。このビューの右側のペインに、提供されている再配布可能ファイルの一覧から選択された InstallShield 前提条件、マージ モジュール、またはオブジェクトの詳細が表示されます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。このビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。InstallScript プロジェクトの [前提条件] ビューで [詳細の表示] ボタンをクリックすると、選択した InstallShield 前提条件に含まれるファイルが表示されます。

InstallScript プロジェクトで、[オブジェクト] ビューに表示されているマージ モジュールまたはオブジェクトの詳細を表示する場合、そのオブジェクトを選択すると、右のペインに再配布可能ファイル内のファイルの一覧が追加の情報と共に表示されます。

ヒント: *マージ モジュールまたはオブジェクトに含まれるファイルを参照する他の方法は、ナレッジベース記事 Q106474 を参照してください。この記事はダウンロードが可能な [マージ モジュール依存関係] ビューアーへのリンクを含んでいます。*

インストール プロジェクトに含まれている InstallShield 前提条件を利用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。InstallShield で提供されている InstallShield 前提条件の一例として、Java Runtime Environment (JRE) および SQL Server Express Edition があります。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

プロジェクト: *Windows Installer ベースのプロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖させることができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。*Setup.exe *セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。*

メモ: *Windows Installer 4.5 のチェーンとは異なり、InstallShield 前提条件のインストールは単一のトランザクションとして処理されません。つまり、正常に完了したインストールは、後続の前提条件のインストールが失敗した場合でもロールバックされることはありません。Windows Installer 4.5 のチェーン サポートの詳細については、「トランザクション処理を使って複数パッケージ インストールを構成する」を参照してください。*

[再配布可能ファイル] ビューで、InstallShield 前提条件を基本の MSI および InstallScript MSI プロジェクトに追加することができます。[前提条件] ビューでは、InstallShield 前提条件を InstallScript プロジェクトに追加します。

このセクションには、次のトピックが含まれています:

- セットアップ前提条件と機能前提条件の違い
- 基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける
- 基本の MSI プロジェクトで、機能から InstallShield 前提条件の関連付けを解除する
- InstallShield 前提条件のインストール順を指定する
- 前提条件を含むリリースを構成する
- InstallShield 前提条件を含むディレクトリを指定する
- 特定の InstallShield 前提条件の実行時の場所を指定する
- リリース フラグを InstallShield 前提条件に割り当てる
- InstallShield 前提条件を含むリリースをビルドする
- InstallShield 前提条件を含むインストールの実行時の動作
- インストールに InstallShield 前提条件が含まれていたアプリケーションをアンインストールする

セットアップ前提条件と機能前提条件の違い

プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

次のプロジェクト タイプでは、セットアップ前提条件のさポートが含まれていますが、機能前提条件は含まれていません:

- *InstallScript*
- *InstallScript MSI*

メインのインストールの [ユーザー インターフェイス] シーケンスが開始する前に実行される InstallShield 前提条件は、*セットアップ前提条件*と呼ばれます。セットアップ前提条件は、インストールされている製品のすべての構成に対してインストールする必要がある、または、それ自身のインストールで使用する機能を提供するベースアプリケーションおよびテクノロジ フレームワークに適しています。プロジェクトに InstallShield 前提条件を追加すると、デフォルトでセットアップ前提条件タイプの InstallShield 前提条件になります。

基本の MSI プロジェクトでは、InstallShield 前提条件をメイン インストールに含まれる機能に関連付けることができます。InstallShield 前提条件が 1 つまたは複数の機能と関連付けられている場合、それは *機能前提条件*と呼ばれます。機能の前提条件は、エンドユーザーがインストールする機能を選択したときインストールされます。つまり、マージ モジュール同様、機能前提条件は、それを含む 1 つまたは複数の機能がインストールされたときのみインストールされます。このため、機能前提条件は、インストールされている製品の一部の構成に対してのみインストールする必要がある、および、それ自身のインストールでは使用されないアプリケーションまたはコンポーネントに適しています。

ニーズに最も適した InstallShield 前提条件の種類を判別するときに役に立つ情報は、次のセクションで読むことができます:

セットアップ前提条件の特別考慮

次は、1 つまたは複数のセットアップ前提条件をプロジェクトに含めるとき、次のヒントを参考にしてください:

.NET Framework の要件

製品の実行に、.NET Framework がターゲット システムにインストールされている必要がある場合、.NET Framework 再配布可能ファイルをプロジェクトに含めることができます。.NET Framework がターゲット システムにないとき、インストール中にインストールされます。詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

インストールに .NET Framework 再配布可能ファイルと、ターゲット マシンで .NET Framework が必要なセットアップ前提条件 (ファイルを GAC にインストール場合など) が含まれている場合、セットアップ前提条件がインストールされる前に NET Framework がインストールされる必要があることを指定できます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

エンドユーザーがセットアップランチャーの代わりに .msi パッケージを起動したとき、エラーを表示する

セットアップ前提条件が含まれている基本の MSI または InstallScript MSI インストールで、エンドユーザーが、製品の Setup.exe セットアップランチャーを起動する代わりに、.msi パッケージを直接起動した場合、セットアップ前提条件のインストールは実行されません。前提条件がターゲット システムに存在しないとき、製品が適切に動作しないことがあります。これは、.msi パッケージが Setup.exe ファイルにストリームされない非圧縮のリリースをビルドしたとき発生することがあります。

この問題が発生するのを防ぐために、タイプ 19 カスタム アクションを基本の MSI または InstallScript MSI プロジェクトに追加することもできます。このカスタム アクションは、InstallShield 前提条件エディターの [条件] タブで前提条件について構成された条件を使用して評価を行います。カスタム アクションによって、セットアップ前提条件が引き続き必要かどうかが検証され、必要な場合、タイプ 19 エラー カスタム アクションによって、エラー メッセージが表示され、インストールが終了します。

機能前提条件の特別考慮

1 つまたは複数の機能前提条件を基本の MSI プロジェクトに含めるとき、次のヒントを参考にしてください。

Windows Installer の要件

プロジェクトに Windows Installer をインストールする前提条件が含まれている場合、前提条件は機能前提条件ではなくセットアップ前提条件になります。したがって、この前提条件を機能に関連付けることはできません。

.NET Framework の要件

Windows Installer をインストールする前提条件が含まれているプロジェクトで、インストールで .NET Framework が存在している必要がある場合 (ファイルを GAC にインストール場合など)、.NET Framework の前提条件は機能前提条件ではなくセットアップ前提条件である必要があります。したがって、この前提条件を機能に関連付けることはできません。

機能前提条件の再起動に関する潜在的な問題

プロジェクトに InstallShield 前提条件が含まれていて、潜在的に再起動の必要がある場合、この前提条件を機能に関連付けないことをお勧めします。機能前提条件によって再起動がトリガーされた場合、再起動のあと再び ReadyToInstall ダイアログが表示され、エンド ユーザーは [インストール] ボタンをクリックして残りのインストールを続行する必要があります。

必要ディスク容量の計算

Windows Installer がファイルのコスト計算関連のアクションを実行するとき、機能の前提条件で必要なディスク容量は自動的には含まれません。したがって、CustomSetup ダイアログが実行時に表示された場合、異なる機能に表示された空きディスク領域の容量は、機能前提条件に必要なディスク容量が考慮されないため正確ではない場合があります。また、可能性として、メイン インストールに機能前提条件を付加したとき、ターゲット システムのディスク空き容量が足りなくなるということも発生します。このような場合、インストールの途中でターゲット システムの空きディスク容量が足りなくなることがあります。

プロジェクトの機能の前提条件でファイルのコスト計算関係の問題を防ぐために、InstallShield 前提条件エディターの [条件] タブで前提条件について構成された条件を使用して評価を行うカスタム アクションを追加することができます。このカスタム アクションは、関連付けられている機能のインストールが選択されたとき、機能前提条件を実行する必要があるかどうかを判別します。

機能前提条件の実行が必要な場合、カスタム アクションによって、一時的な行が ReserveCost テーブル に追加されます。

基本の MSI プロジェクトで、InstallShield 前提条件を機能に関連付ける

プロジェクト: *基本の MSI プロジェクトは、前提条件と機能の関連付けをサポートします。*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

プロジェクトで、InstallShield 前提条件が機能と関連付けられている場合、それは *機能前提条件*と見なされます。機能に関連付けられていない場合、それは *セットアップ前提条件*と見なされます。

インストール プロジェクトに InstallShield 前提条件を追加したとき、デフォルトで、その前提条件がセットアップ前提条件として追加されます。セットアップ前提条件は、それをプロジェクトに既に存在する 1 つまたは複数の機能に関連付けることで機能前提条件にすることができます。

タスク ***InstallShield 前提条件を機能に関連付けるには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 再配布可能ファイルの一覧で、機能に関連付ける InstallShield 前提条件を選択します。

   メモ・ *InstallShield 前提条件のチェック ボックスは、既に選択されている必要があります。選択されたチェック ボックスは、前提条件がプロジェクトに含められることを示します。詳細については、「InstallShield 前提条件、マージ モジュール、オブジェクトを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加する」を参照してください。*
3. **[条件付インストール]** ペイン内で、この InstallShield 前提条件を追加するすべての機能のチェック ボックスを選択します。

前提条件を新しい機能に関連付ける場合、まずその機能を作成する必要があります。新しい機能の作成方法については、「機能の作成」をご覧ください。

前提条件をプロジェクト内のすべての機能に関連付けたあとで新しい機能を追加した場合、その機能前提条件は新しい機能に自動的に関連付けられません。

メモ・ *機能前提条件には、セットアップ前提条件には適用されない制限事項があります。詳細については、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

基本の MSI プロジェクトで、機能から InstallShield 前提条件の関連付けを解除する

プロジェクト・ *基本の MSI プロジェクトは、前提条件と機能の関連付けをサポートします。*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

プロジェクトで、InstallShield 前提条件が機能と関連付けられている場合、それは *機能前提条件*と見なされます。機能に関連付けられていない場合、それは *セットアップ前提条件*と見なされます。

タスク ***機能から InstallShield 前提条件を削除するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 再配布可能ファイルの一覧で、機能との関連付けを解除する InstallShield 前提条件を選択します。
3. **[条件付きインストール]** ペインで、**[機能の選択の前にインストールする]** チェック ボックスを選択します。このチェック ボックスは、InstallShield 前提条件を基本の MSI プロジェクトに追加したとき、デフォルトで選択されています。

メモ: *セットアップ前提条件には、機能前提条件にはない利点がいくつかあります。詳細については、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

InstallShield 前提条件のインストール順を指定する

プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

次のプロジェクト タイプでは、セットアップ前提条件のさポートが含まれていますが、機能前提条件は含まれていません:

- *InstallScript*
- *InstallScript MSI*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

[再配布可能ファイル] ビューでは、基本の MSI プロジェクトまたは InstallScript MSI プロジェクトに複数の InstallShield 前提条件を含める場合、それらをインストールする順番を指定することができます。[前提条件] ビューでは、InstallScript プロジェクトに複数の InstallShield 前提条件を含める場合、それらをインストールする順番を指定することができます。

タスク ***InstallShield 前提条件がターゲット マシン上にインストールされる順番を指定するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下のビュー リストで **[再配布可能ファイル]** (基本の MSI プロジェクトまたは InstallScript MSI プロジェクトの場合)、または **[前提条件]** (InstallScript プロジェクトの場合) をクリックします。
2. 必要な InstallShield 前提条件をプロジェクトに追加していない場合、それを行います。
3. 任意の再配布可能ファイルを右クリックして、**InstallShield 前提条件の順番を設定**をクリックします。**InstallShield[前提条件のインストール順]** ダイアログ ボックスが開きます。
4. リストから前提条件を選択してから上下矢印をクリックしてインストールの順に並べます。

プロジェクト: *基本の MSI プロジェクトで順番を指定するとき、InstallShield では、セットアップ前提条件と機能前提条件の違いは識別されません。したがって、プロジェクトにセットアップ前提条件と機能前提条件が混在す*

る場合、[InstallShield 前提条件のインストール順] ダイアログ ボックスで、それらがすべて同じ一覧内に表示されます。実行時に、メインのインストールが起動される前、Setup.exe セットアップランチャーはセットアップ前提条件のみ評価し (必要時のみ)、[InstallShield 前提条件のインストール順] ダイアログ ボックスで指定した順番でそれらをインストールします。このあと、インストールの後半で、Windows Installer エンジンによって機能前提条件のみが評価され (必要時のみ)、それらが指定された順番でインストールされます。

ヒント: *InstallShield 前提条件がインストールされる前に Windows Installer エンジンまたは .NET Framework、または両方 をインストールしなくてはならない場合、InstallShield 前提条件エディターで InstallShield前提条件を開いて、この要件を指定することができます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。*

前提条件を含むリリースを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件を含む基本の MSI インストールまたは InstallScript MSI インストールをパッケージする場合、以下のいずれか 1 つの方法を使ってエンド ユーザーに対して InstallShield 前提条件ファイルを提供することができます。

- InstallShield 前提条件ファイルを、ソース メディアに格納する。
- InstallShield 前提条件ファイルを Setup.exe に圧縮し、実行時に必要に応じて抽出されるように設定する。
- 必要な場合、インストールはプロジェクトに含まれた InstallShield 前提条件ファイルを、必要に応じて各前提条件の InstallShield 前提条件ファイル (.prq) で指定した URL からダウンロードすることができます。

InstallShield 前提条件を含む InstallScript インストールの場合、利用できる方法は多少異なります:

- リリースの構成に従って、InstallShield 前提条件ファイルを、ソース メディアまたは Setup.exe に格納します。
- インストールはプロジェクトに含まれた InstallShield 前提条件ファイルを、必要に応じて各前提条件の InstallShield 前提条件ファイル (.prq) で指定した URL からダウンロードすることができます。

インストールに含まれる各 InstallShield 前提条件の提供方法を指定することができます。詳細については、「特定の InstallShield 前提条件の実行時の場所を指定する」を参照してください。

リリースに含まれるすべての InstallShield 前提条件を同じ方法で利用可能にする場合、リリース レベルで個別の方法をオーバーライドすることも可能です。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。

Windows Installer エンジンおよび .NET Framework のインストールの前または後でインストールされるように InstallShield 前提条件を構成することができます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

InstallShield 前提条件を含むディレクトリを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield前提条件ファイル (.prq) のデフォルトの場所は、以下のとおりです:

InstallShield Program Files フォルダー¥SetupPrerequisites

InstallShield では、ローカル マシンまたはネットワーク上の追加の場所または代替の場所を指定することができます。この柔軟な機能を使って、InstallShield 前提条件をソース コード管理システムに格納し、InstallShield 前提条件の共通のセットを他のチーム メンバーと共有することができます。

InstallShield では、InstallShield 前提条件ファイル (.prq) の検索パスを指定するいくつかの方法があります:

- InstallShield 内部から編集またはビルドを行う場合、[ツール] メニューで [オプション] をクリックすると表示される [オプション] ボックスにある [前提条件] タブを使って、マシン共通および現在のユーザーのフォルダをコンマ区切りのリストで指定できます。

  InstallShield は、[前提条件] タブで指定したパスをマシン上のレジストリに保存します。現在のユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥PrerequisiteSearchPath

  すべてのユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥PrerequisiteSearchPath

  Standalone Build を使ってリリースをビルドする場合、[オプション] ダイアログ ボックスは使用できません。ただし、必要であれば Standalone Build が搭載されているマシン上のレジストリへのパスを手動で追加することができます。

- ISCmdBld.exe を使って、コマンドラインからビルドする場合は、-prqpath パラメーターを使ってフォルダのコンマ区切りのリストを指定します。

  .ini ファイルを使って ISCmdBld.exe パラメーターを指定する場合、.ini ファイルの [Mode] セクションで新しい PrerequisitePath パラメーターを使用して、フォルダのコンマ区切りのリストを指定できます。

- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで PrerequisitePath パラメーターを使います。このパラメーターは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPrerequisitePath として露出されます。複数のパスを指定するには、順序指定されたパスの配列を使用します。

ハードコード化されたパスの代わりに、以下の例のようにパスにパス変数を使用することができます:

<ISProductFolder>¥SetupPrerequisites,<ISProjectFolder>¥MyCustomPrerequisites

[再配布可能ファイル] ビューと [前提条件] ビューには、[オプション] ダイアログ ボックスの [前提条件] タブで指定されている様々な検索パスに含まれている .prq ファイルに対応する InstallShield 前提条件の名前がリストされます。同じ .prq ファイルが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみを表示します。InstallShield は、最初に [前提条件] タブでユーザーごとの設定にリストされた各パスをチェックする。次に、マシン共通の設定にリストされている各パスをチェックします。

ビルド時、プロジェクトに 1 つ以上の InstallShield 前提条件が含まれている場合、InstallShield (または Standalone Build) が指定された場所を検索して、必要に応じて適切な InstallShield 前提条件をリリースに含みます。同じ .prq ファイルが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみをビルドに含みます。.prq ファイルの検索は以下の順で行われます:

1. InstallShield (または Standalone Build)が –prqpath コマンドライン パラメーター、PrerequisitePath .ini ファイル パラメーター、または InstallShield タスクの PrerequisitePath パラメーターを通して指定されたパスをチェックする。
2. InstallShield が [前提条件] タブでユーザーごとの設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥PrerequisiteSearchPath

   Standalone Build で [前提条件] タブを使用することはできませんが、レジストリ パスは使用できます。
3. InstallShield が [前提条件] タブのマシン共通の設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥PrerequisiteSearchPath

   Standalone Build で [前提条件] タブを使用することはできませんが、レジストリ パスは使用できます。
4. 前述の場所のいずれにもパスが指定されていない場合、InstallShield はデフォルトの場所 (*InstallShield (または Standalone Build) Program Files フォルダー*¥SetupPrerequisites) をチェックします。

ヒント: *Standalone Build を使って InstallShield 前提条件を含むが、1 つ以上の検索パスが指定されていないリリースをビルドすると、Standalone Build はデフォルト パス (<ISProductFolder>¥SetupPrerequisites) で InstallShield 前提条件ファイルを検索します。ただし、1 つ以上の検索パスを指定して、デフォルト パスを明記しなかった場合、Standalone Build はデフォルト パスを検索しません。*

特定の InstallShield 前提条件の実行時の場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、プロジェクトの各 InstallShield 前提条件に異なる実行時の場所を指定することができます。

タスク ***インストールの各 InstallShield 前提条件に異なる実行時の場所を指定するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下のビュー リストで **[再配布可能ファイル]** (基本の MSI プロジェクトまたは InstallScript MSI プロジェクトの場合)、または **[前提条件]** (InstallScript プロジェクトの場合) をクリックします。
2. インストールに含める InstallShield 前提条件の 1 つのチェック ボックスを選択します。

3. InstallShield 前提条件を右クリックして、**[プロパティ]** をクリックします。InstallShield**[前提条件のプロパティ]** ダイアログ ボックスが開きます。
4. **[ビルドの場所]** リストで、適切なオプションをクリックします。

指定した場所は、リリース レベルでオーバーライドすることができます。個々の InstallShield 前提条件に選択した値のオーバーライドを禁止するには、リリース レベルにある "InstallShield 前提条件の場所" 設定を [個々の選択に従う] に設定します。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。

ヒント: *InstallShield 前提条件が別の前提条件の依存関係としてプロジェクトに追加される場合、前提条件の依存関係の場所は、それを必要とする InstallShield 前提条件の場所設定に従います。*

リリース フラグを InstallShield 前提条件に割り当てる

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

特定のビルドから除外する InstallShield 前提条件にリリース フラグを設定することができます。たとえば、InstallShield 前提条件を必要とする特別なアドオンを含んだ製品の特別なエディションにのみ含める InstallShield 前提条件がある場合、その InstallShield 前提条件にフラグをつけることで、必要な場合にのみそれがビルドに含まれるようにします。

タスク ***インストール プロジェクトに追加した InstallShield 前提条件にリリース フラグを追加するには、次の手順に従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. InstallShield 前提条件を右クリックして、**[プロパティ]** をクリックします。InstallShield**[前提条件のプロパティ]** ダイアログ ボックスが開きます。
3. **[リリース フラグ]** ボックスで、文字列を入力します。文字列には、文字または数字を使用できます。複数のフラグを前提条件追加する場合は、コンマを使ってフラグを区切ります。

リリース フラグに基づいて InstallShield 前提条件をフィルターする方法について、さらに詳しい情報は「リリース フラグ」を参照してください。

InstallShield 前提条件を含むリリースをビルドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield は、前提条件を含まないプロジェクトの Setup.exe ファイルをビルドするとき、以下の場所に格納されたベース Setup.exe ファイルからビルドを始めます。

InstallShield Program Files フォルダー¥redist¥Language Independent¥i386

ただし、InstallShield が前提条件を含むプロジェクトの Setup.exe ファイルをビルドするとき、前述のファイルをベースとして利用することはできません。これは、Setup.exe ファイルに前提条件を含むことができないためです。代わりに SetupPrereq.exe と呼ばれる、多少サイズが大きいファイルを利用します。ベース SetupPrereq.exe ファイルはベース Setup.exe ファイルと同じディレクトリに配置されています。異なるベースファイル (Setup.exe および SetupPrereq.exe) が使用されるため、エンドユーザーに配布される最終ビルド Setup.exe ファイルの追加サイズ オーバーヘッドは、実際プロジェクトに前提条件を含む作業を行うインストール作成者にのみ負担が掛かることになります。

InstallShield 前提条件を含むインストールの実行時の動作

プロジェクト: *次のプロジェクト タイプでは、セットアップ前提条件と機能前提条件が両方含まれています:*

- *基本の MSI*

次のプロジェクト タイプでは、セットアップ前提条件のさポートが含まれていますが、機能前提条件は含まれていません:

- *InstallScript*
- *InstallScript MSI*

セットアップ前提条件と機能前提条件 (2 つの異なる種類の InstallShield 前提条件) の違いについては、「セットアップ前提条件と機能前提条件の違い」を参照してください。

InstallShield 前提条件を含むインストールの概要

次のプロシージャは、エンドユーザーがセットアップ前提条件と機能前提条件を含むインストールを起動したときに、一般的に実行時に発生する事柄の説明です。一部の手順は、特定の種類のプロジェクトにのみ適用します。

1. セットアップランチャー (通常、Setup.exe と呼ばれます) は、必要に応じて言語の選択ダイアログを表示します。
2. セットアップランチャーは、必要に応じてセットアップの前提条件ダイアログを表示し、セットアップ前提条件インストールを起動します。
3. インストールは、エンド ユーザーが機能を選択したり、項目を構成したりできるインストール UI を表示します。インストール UI は、進行状況ダイアログを表示します。
4. 基本の MSI インストールでは、適切な場合、セットアップランチャーが機能前提条件のインストールを起動します:
    a. ビルトイン InstallShield カスタム アクション ISInstallPrerequisites (SetupProgress ダイアログと ExecuteAction アクションの間にスケジュールされる) は、インストールすることが選択された機能を Windows Installer プロパティ IsPrerequisiteFeatures のリストと比較します。全く一致しなかった場合、機能前提条件はインストールされません。
    b. ISInstallPrerequisites アクションは、セットアップランチャーを検索および起動し、またインストール中の機能のリストを提供します。セットアップランチャーのパスは、Windows Installer プロパティ SETUPEXEDIR と SETUPEXENAME によって識別されます:

[SETUPEXEDIR]¥[SETUPEXENAME]

ISInstallPrerequisites は、その場所でセットアップランチャーが見つからなかった場合、別の場所を検索します。初回のインストールでは、SourceDir が確認されます。メンテナンス モードでは、インストール ソース パスに関連するパスが確認されます。

ISInstallPrerequisites がセットアップランチャーが見つけられなかった場合、または、複数の .exe ファイルが見つかった場合、エンドユーザーにプロンプトが表示され、セットアップランチャー ファイルを参照するように要求します。エンドユーザーがファイルを識別すると、インストールが続行します。そうでない場合、インストールは終了します。

c. セットアップランチャーは、機能のリストを評価してインストールする機能前提条件を選択し、また適切なインストールを起動します。

5. インストールはエンド ユーザーの選択にしたがって、ターゲット システム上で変更を完了します。

6. インストールは進行状況ダイアログから SetupCompleteSuccess ダイアログに切り替わります。

InstallShield 前提条件を含むインストールのユーザー インターフェイス

ターゲット システムにインストールする必要があるセットアップ前提条件が 1 つ以上ある場合、通常、メイン インストールの実行が開始される前に、セットアップ前提条件ダイアログが表示されます。このセットアップ前提条件によって、ターゲット システムに不足している非表示のセットアップ前提条件がすべて表示されます。エンド ユーザーがこのダイアログで [インストール] ボタンをクリックすると、必要なセットアップ前提条件のインストールが起動されます管理者権限が必要とマークされているセットアップ前提条件が 1 つ以上あり、かつ、インストールがユーザー アカウント制御 (UAC) が有効にされたシステムで実行される場合、このダイアログの [インストール] ボタンが、昇格された権限が必要であることをエンドユーザーに通知するシールド アイコンと共に表示されます。

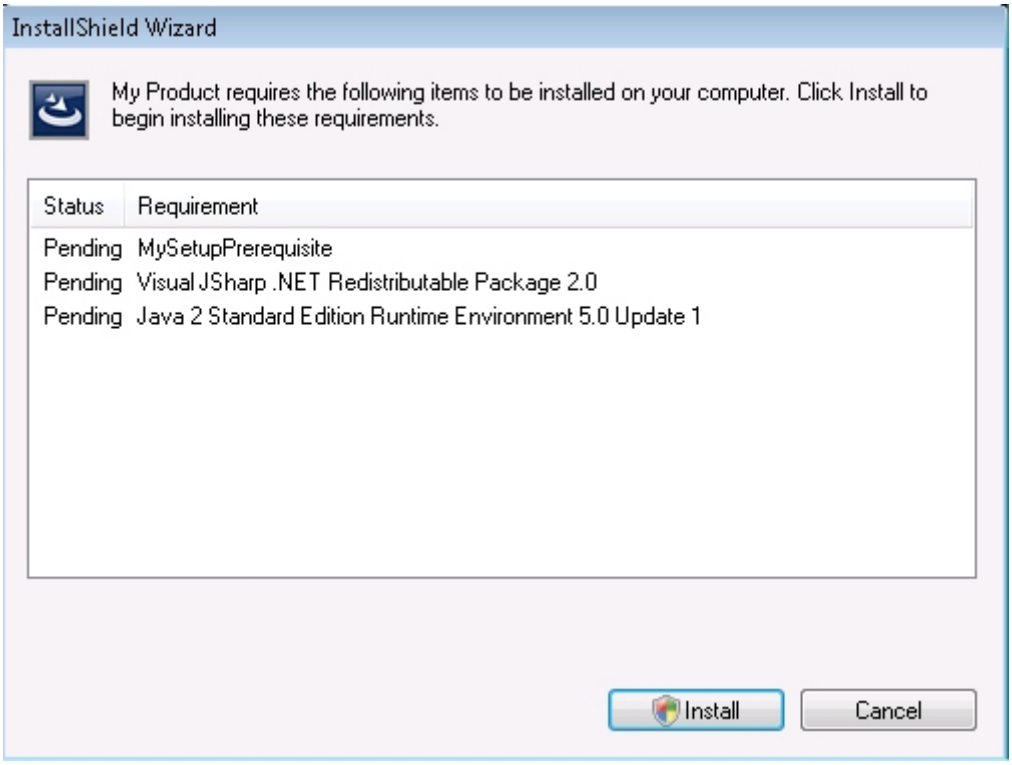


図 2: [インストールするセットアップ前提条件] リストを表示するサンプル [セットアップ前提条件] ダイアログ

セットアップ前提条件が非表示と構成された場合、この前提条件はセットアップ前提条件ダイアログに表示されませんが、インストールはされます。インストールに含まれるすべてのセットアップ前提条件が隠されている場合、セットアップ前提条件ダイアログの代わりに、セットアップランチャーの標準初期ダイアログがインストールで表示されます。

ヒント: セットアップ前提条件ダイアログでセットアップ前提条件の表示と非表示を切り替える方法については、「ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する」をご覧ください。

セットアップ前提条件のインストールが起動するファイルが .msi パッケージで、前提条件が進行状況を表示するようにマークされている場合、前提条件のインストール中、ユーザー インターフェイスで、Windows Installer からのインストールの進行状況メッセージおよびステータス バージョンが表示されます。これは、メイン インストールが基本の MSI インストール、または InstallScript MSI インストールの場合のみ適用します。詳細については、「実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する」を参照してください。

セットアップ前提条件がエンドユーザーによってオプションでインストールされると構成されている場合、エンド ユーザーが前提条件をインストールするかどうかを選択することができるメッセージ ボックスが表示されます。詳細については、「InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする」を参照してください。

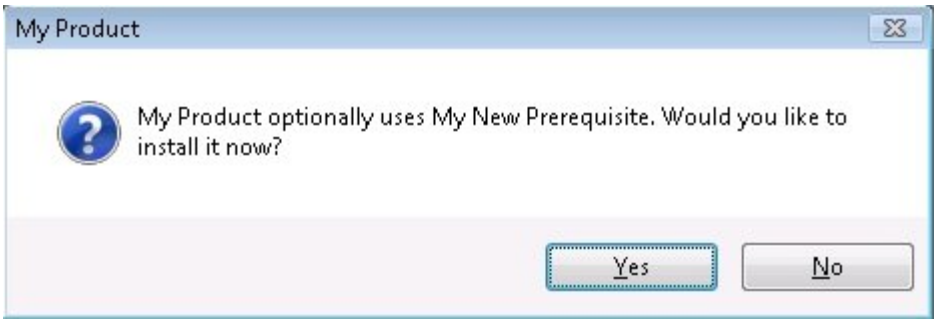


図 3: オプション前提条件のメッセージ ボックス

インストールに機能前提条件が含まれている場合、セットアップランチャーが表示するすべての前提条件のダイアログでそれらが表示されません。ただし、ユーザー インターフェイスでは、必要に応じて進行状況メッセージが表示されます。また、機能前提条件がオプションとマークされている場合、オプションの前提条件メッセージ ボックスが表示されます。

サイレント シナリオ – 基本の MSI インストールで抑制されたユーザー インターフェイスの場合

セットアップ前提条件と機能前提条件は、インストールがサイレントで実行されるときもインストールされます。InstallShield 前提条件は、次のすべてのシナリオでサポートされています:

- **サイレント セットアップランチャーと可視の .msi パッケージ** – セットアップランチャーのユーザー インターフェイスは抑制されますが、.msi パッケージのユーザー インターフェイスは表示されます。たとえば、エンドユーザーは次のコマンドライン ステートメントを使用できます:

  Setup.exe /s

  このシナリオでは、言語の選択ダイアログとセットアップ前提条件ダイアログは表示されません。

- **可視のセットアップランチャーとサイレント .msi パッケージ** – セットアップランチャーのユーザー インターフェイスは表示されますが、.msi パッケージのユーザー インターフェイスは抑制されます。たとえば、エンドユーザーは次のコマンドライン ステートメントを使用できます:

  Setup.exe /v"/qn"

  このシナリオでは、メイン インストールの言語の選択ダイアログとセットアップ前提条件ダイアログは表示されません。ただし、エンドユーザーは、インストールする機能を指定するとき、コマンドラインから ADDLOCAL、ADDSOURCE、ADDDEFAULT、ADVERTISE などの Windows Installer プロパティを設定することができます。

- **サイレント セットアップランチャーとサイレント .msi パッケージ** – セットアップランチャーと .msi パッケージのユーザー インターフェイスは抑制されます。たとえば、エンドユーザーは次のコマンドライン ステートメントを使用できます:

  Setup.exe /s /v"/qn"

  このシナリオでは、すべてのセットアップランチャーと .msi パッケージのダイアログが抑制されます。

メイン インストール内にある .msi パッケージの UI シーケンスがスキップされると、セットアップランチャーは ADDLOCAL、ADDSOURCE、ADDDEFAULT、ADVERTISE などの Windows Installer プロパティを評価して、インストールが必要な機能前提条件があるかどうかを判別し、その結果にしたがって、機能前提条件をインストールします。

サイレント シナリオ – InstallScript および InstallScript MSI インストールで抑制されたユーザー インターフェイスの場合

InstallScript または InstallScript MSI インストールでは、インストールがサイレントで実行される場合でもセットアップ前提条件をインストールできます。つまり、エンド ユーザーが以下のコマンドライン ステートメントを入力してインストールを起動した場合に、InstallShield 前提条件がサポートされます。

Setup.exe /s

これには、応答ファイル (Setup.iss) が必要です。詳細については、「応答ファイルを作成する」を参照してください。

言語の選択ダイアログとセットアップ前提条件ダイアログは表示されません。

UAC のプロンプト

構成方法によって、InstallShield 前提条件を含むインストールがインストール中のいくつかの時点で、Windows Vista 以降のシステム上で昇格された権限のプロンプトを表示することができます:

1. エンド ユーザーが Setup.exe ファイルを起動するとき
2. Setup.exe ファイルが、昇格された権限を必要とするセットアップ前提条件を起動するとき
3. Setup.exe ファイルが、昇格された権限を必要とする機能前提条件を起動するとき
4. Windows Installer が .msi パッケージの [実行] シーケンスを開始するとき

詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

InstallShield 前提条件の動作を変更する

インストールの動作を変更したい場合、InstallShield 前提条件エディターを使ってInstallShield 前提条件を変更することができます。たとえば、ユーザーが前提条件のインストールをスキップするかどうか、前提条件には管理者権限が必要かどうか、またターゲットマシンの再起動の要否に応じて前提条件インストールをどのように処理するかを指定することができます。詳細については、「InstallShield 前提条件のインストール動作を指定する」を参照してください。

Windows Installer エンジンおよび .NET Framework のインストールの前または後でインストールされるように InstallShield 前提条件を構成することができます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

インストールに InstallShield 前提条件が含まれていたアプリケーションをアンインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールがアプリケーションおよび 1 つまたは複数の InstallShield 前提条件から構成される場合があります。エンドユーザーがコントロール パネルのプログラムの追加と削除を使ってアプリケーションをアンインストールした場合、InstallShield 前提条件はマシン上にインストールされたままの状態です。InstallShield 前提条件インストールによって、[プログラムの追加と削除] にエントリが追加された場合、エンドユーザーは [プログラムの追加と削除] を通して InstallShield 前提条件を削除することができます。

インストール プロジェクトに含まれているマージ モジュールを使用する

ドキュメントのこのセクションでは、インストール プロジェクト内からマージ モジュールを使用する方法について説明します。

自分自身のマージ モジュールを作成または変更するときの情報については、「マージ モジュールのデザイン」をご覧ください。

このセクションには、次のトピックが含まれています:

- マージ モジュールを含むディレクトリを指定する
- マージ モジュールのインストール先をオーバーライドする
- マージ モジュールに関するトラブルシューティング
- マージ モジュールをインストール プロジェクト内からリポジトリにパブリッシュする

マージ モジュールを含むディレクトリを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、ローカル マシンまたはネットワーク上でマージ モジュール (.msm ファイル) を格納する場所を指定できます。この柔軟な機能を使って、マージ モジュールをソース コード管理システムに格納し、マージ モジュールの共通のセットを他のチーム メンバーと共有することができます。

InstallShield では、マージ モジュールの検索パスを指定するいくつかの方法があります:

- InstallShield 内部から編集またはビルドを行う場合、[ツール] メニューで [オプション] をクリックすると表示される [オプション] ボックスにある [マージ モジュール] タブを使って、マシン共通および現在のユーザーのフォルダをコンマ区切りのリストで指定できます。

  InstallShield は、[マージ モジュール] タブで指定したパスをマシン上のレジストリに保存します。現在のユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥MMSearchPath

  すべてのユーザーに指定されたパスは、以下の場所に保存されます:

  HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥MMSearchPath

  Standalone Build を使ってリリースをビルドする場合、[オプション] ダイアログ ボックスは使用できません。ただし、必要であれば Standalone Build が搭載されているマシン上のレジストリへのパスを手動で追加することができます。

- ISCmdBld.exe を使って、コマンドラインからビルドする場合は、-o パラメーターを使ってフォルダのコンマ区切りのリストを指定します。

  .ini ファイルを使って ISCmdBld.exe パラメーターを指定する場合、.ini ファイルの [Mode] セクションで新しい MergeModulePath パラメーターを使用して、フォルダのコンマ区切りのリストを指定できます。

- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで MergeModulePath パラメーター を使います。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldMergeModulePath として露出されます。複数のパスを指定するには、順序指定されたパスの配列を使用します。

ハードコード化されたパスの代わりに、以下の例のようにパスにパス変数を使用することができます:

<ISProductFolder>¥MergeModules,<ISProjectFolder>¥MyCustomMergeModules

[再配布可能ファイル] ビューと [オブジェクト] ビューには、[オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されている様々な検索パスに含まれているマージ モジュールファイルに対応するマージ モジュールの名前がリストされます。同じマージ モジュールが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみを表示します。InstallShield は、最初に [マージ モジュール] タブでユーザーごとの設定にリストされた各パスをチェックします。次に、マシン共通の設定にリストされている各パスをチェックします。

ビルド時、プロジェクトに 1 つ以上のマージ モジュールが含まれている場合、InstallShield (または Standalone Build) が指定された場所を検索して、必要に応じて適切な マージ モジュールをリリースに含みます。同じマージ モジュールが複数の検索パスに含まれている場合、InstallShield は最初に検出されたインスタンスのみをビルドに含みます。マージ モジュールの検索は以下の順で行われます:

1. InstallShield (または Standalone Build)が –o コマンドライン パラメーター、MergeModulePath .ini ファイル パラメーター、または InstallShield タスクの MergeModulePath パラメーターを通して指定されたパスをチェックする。
2. InstallShield が [マージ モジュール] タブでユーザーごとの設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_CURRENT_USER¥Software¥InstallShield¥*Version*¥Professional¥Project Settings¥MMSearchPath

   Standalone Build で [マージ モジュール] タブを使用することはできませんが、レジストリ パスは使用できます。
3. InstallShield が [マージ モジュール] タブのマシン共通の設定にリストされた各パスをチェックする。パスは、レジストリの次の場所に保存されます。

   HKEY_LOCAL_MACHINE¥Software¥InstallShield¥*Version*¥Professional¥MMSearchPath

   Standalone Build で [マージ モジュール] タブを使用することはできませんが、レジストリ パスは使用できます。
4. 前述の場所のどこにもパスが指定されていない場合、InstallShield は次の順序でデフォルト ディレクトリをチェックします:
   a. *InstallShield (または Standalone Build) Program Files フォルダー*¥System
   b. *InstallShield (または Standalone Build) Program Files フォルダー*¥Modules¥i386
   c. *InstallShield (または Standalone Build) Program Files フォルダー*¥Objects
   d. *InstallShield (または Standalone Build) Program Files フォルダー*¥Modules¥i386¥Japanese
   e. *InstallShield (または Standalone Build) Program Files フォルダー*¥Modules¥i386¥German
   f. *Program Files フォルダー*¥Common Files¥Merge Modules

ヒント: *Standalone Build を使ってマージ モジュールを含むリリースをビルドするときに、ユーザー定義による方法で (つまり、コマンドラインまたは .ini ファイルを通して、あるいは InstallShield タスクを通してレジストリで) 1 つ以上の検索パスを指定しなかった場合 、Standalone Build 前述のデフォルト ディレクトリでマージ モジュールを検索します。ただし、1 つ以上の検索パスを指定して、デフォルト ディレクトリを明記しなかった場合、Standalone Build はデフォルト ディレクトリを検索しません。*

マージ モジュールのインストール先をオーバーライドする

サードパーティ マージ モジュールは変更すべきではありませんが、いくつかのサードパーティ マージ モジュールおよび InstallShield 作成のマージ モジュールのインストール先を上書きすることは可能です。

メモ: *この手順は、マージ モジュールの* TARGETDIR *ディレクトリ、または* TARGETDIR *から直接派生したディレクトリのみを転送します。マージ モジュールが定義済みフォルダー (たとえば、SystemFolder) へファイルを送るように構成されている場合 、モジュールのインストール先をオーバーライドすることはできません。*

タスク ***マージ モジュールのインストール先をオーバーライドするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. マージ モジュールの横にあるチェック ボックスを選択して、インストールに追加します。
3. モジュールを右クリックして、**プロパティ**をクリックします。[マージ モジュールプロパティ] ダイアログ ボックス が表示されます。
4. **[インストール先]** ボックスで、インストール先を入力するか、定義済みのインストール先から 1 つ選択します。
5. **[OK]** をクリックします。
6. **[条件付きインストール]** ペインで、マージ モジュールを含める機能を選択します (複数可)。

マージ モジュールに関するトラブルシューティング

関連付けられているマージ モジュールの削除

インストールに現在関連付けられているマージ モジュールを削除すると、[マージ モジュールが見つかりません] というメッセージが表示されます。このことにより、そのモジュールをインストールに追加できないことがわかります。

マージ モジュールの言語プロパティの変更

ユーザー定義または別の会社のマージ モジュールが再配布可能ファイルギャラリーに追加された後、その言語プロパティを変更すると、変更後のマージ モジュールをプロジェクトに関連付けようとしたときにビルドエラーが発生します。このエラーが発生するのは、InstallShield がモジュールファイルを識別する際にモジュール識別子の他に言語識別子を使用するためです。

マージ モジュールファイルが再配布可能ファイルギャラリーに追加された後、そのファイルの言語識別子を変更すると、InstallShield はファイルを見つけられなくなります。

正しいマージ モジュール ファイルの場所を特定するには、[再配布可能ファイル] ビューから参照することができます。詳しくは、「マージ モジュールを参照する」をご覧ください。

マージ モジュールをインストール プロジェクト内からリポジトリにパブリッシュする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer ベースのインストール プロジェクトで再利用可能な、または他のユーザーと共有可能な既存のマージ モジュールがある場合、それをリポジトリへパブリッシュすることができます。

タスク ***Windows Installer ベースのインストール プロジェクトで作業中に、マージ モジュールをリポジトリへパブリッシュするには、次の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. マージ モジュールを右クリックして、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

ビルド時にマージ モジュールがインストールへ組み込まれます。リポジトリ マージ モジュールに変更を加えてからリポジトリへ再パブリッシュすると、プロジェクトのインストールが次回再ビルドされたときに、古いバージョンのリポジトリマージ モジュールを持つプロジェクトはすべて更新されます。

Windows Installer 再配布可能ファイルをプロジェクトに追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Windows Installer は Windows のほとんどのバージョンに組み込まれていますが、Windows Inslaller ベースのインストールは Windows Installer の最新バージョンでのみ動作する特定の機能に依存している可能性があります。また、一部の InstallScript インストールでは、特定のバージョンの Windows Installer が必要な場合もあります。

InstallShield では、Windows Installer の再配布可能ファイルをプロジェクトに含めることができます。Windows Installer 実行可能ファイルの追加方法は、インストールに必要な Windows Installer のバージョンだけでなく、使用中のプロジェクトの種類によって異なります。

Windows Installer の各バージョンごとの最小オペレーティング システムの一覧は、「ターゲット システムの要件」を参照してください。

Windows の各バージョンで一緒にリリースされた Windows Installer のバージョン一覧については、Windows Installer ヘルプ ライブラリの「Released Versions of Windows Installer」をご覧ください。

Windows Installer 5 と Windows Installer 4 は、再配布可能ファイルとして提供されていません。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

InstallShield には、Windows Installer 再配布可能ファイルを自己展開型実行可能ファイル (Setup.exe) としてインストールに含めるためのオプションがあります。

Windows Installer の配布

デフォルトで、InstallShield は Setup.exe をセットアップ パッケージと同時に作成します。セットアップランチャーは、インストールで Windows Installer エンジンをインストールする場合、必ず必要です。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに Windows Installer 再配布可能ファイルを含める場合、次のいずれかを実行します:

- **Windows Installer 4.5 の場合**—プロジェクトに 1 つまたは複数の Microsoft Windows Installer 前提条件を追加します。InstallShield は、異なるバージョンの Windows をターゲットとするいくつかのバージョンを含みます。詳細については、「Microsoft Windows Installer の前提条件を含める」を参照してください。
- **Windows Installer 3.1、3.0、または 2.0 の場合**—[リリース] ビューの Setup.exe タブでは、Setup.exe ランチャーを使用するかどうか、これらの Windows Installer 再配布可能ファイルの 1 つのバージョンを含めるかどうか、および含める Windows Installer のバージョンなどの情報を指定することができます。詳細については、「リリースの [Setup.exe] タブ」を参照してください。

  代りに、プロジェクトに 1 つまたは複数の Microsoft Windows Installer 前提条件を追加することともできます。詳細については、「Microsoft Windows Installer の前提条件を含める」を参照してください。

ヒント・セットアップランチャーの要件は、リリース ウィザードの [セットアップランチャー] パネルでも指定できます。

インストール プロセスの概要

実行時、Setup.exe は Windows Installer が既にターゲット システムにインストールされているかどうかを判別します。ターゲット システムに Windows Installer が見つかり、かつ、最小バージョンの要件を満たした場合、インストール パッケージが起動されます。Windows Installer がインストールされていない場合や、または新しいバージョンをインストールする必要がある場合、 Setup.exe は Windows Installer をインストールしてからインストール パッケージを起動します。Windows Installer を更新するためにシステムを再起動する必要があることに注意してください。

InstallScript プロジェクト

場合によって、InstallScript プロジェクトに、Windows Installer 再配布可能ファイルを追加したほうが良い場合があります。たとえば、InstallScript プロジェクトを使って、Windows Installer の特定の最小バージョンが必要な複数の Windows Installer ベースのインストールを連鎖させることがあります。また、マージ モジュールを InstallScript プロジェクトに追加した場合、そのマージ モジュールを [オブジェクト] ビューに マージ モジュール ホルダー– オブジェクトのサブアイテムとして追加する必要があります。このマージ モジュール ホルダー– オブジェクトは Windows Installer エンジンを必要とするため、ターゲット システム上に Windows Installer が存在しない可能性がある場合、Windows Installer 再配布可能ファイルを InstallScript プロジェクトに追加する必要があります。

Windows Installer 再配布可能ファイルを InstallScript プロジェクトに含めるには、1 つ以上の Microsoft Windows Installer 前提条件をプロジェクトに追加します。InstallShield は、異なるバージョンの Windows をターゲットとするいくつかのバージョンを含みます。詳細については、「Microsoft Windows Installer の前提条件を含める」を参照してください。

Microsoft Windows Installer の前提条件を含める

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield には、複数の Windows Installer バージョン用の InstallShield 前提条件が含まれています。[再配布可能ファイル] ビュー (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合) または [前提条件] ビュー (InstallScript プロジェクトの場合) を使って、プロジェクトにこれらの InstallShield 前提条件を追加できます。

基本の MSI インストールまたは InstallScript MSI インストールの実行中に Windows Installer 4.5 の存在をチェックする

Windows Installer 4.5 がインストールされているかどうかを判別するために、インストールがターゲット システムをチェックする必要があるかもしれません。Windows Installer 4.5 がインストールされていない場合、インストールはエラー メッセージ ボックスを表示して、処理の続行を阻止します。この動作は、以下の状況下で役立ちます:

- インストールが Windows Installer 4.5 の機能に依存している場合。
- 製品のインストール時に、必要に応じて Windows Installer 4.5 をインストールする Setup.exe セットアップランチャーではなく、.msi ファイルが使用される可能性がある場合。

  例えば、インストールを非圧縮リリースとして配布した場合に、エンド ユーザーが Setup.exe ファイルではなく .msi ファイルを直接起動できるとき。さらに、システム管理者が製品を企業環境に配布するためにインストールをカスタマイズする必要がある場合、管理インストール バージョンを作成および配布する可能性があります。

Windows Installer 4.5 がインストールされているかどうかを判別して、インストールされていない場合にエラーメッセージを表示するには、エラー カスタム アクション (タイプ 19 カスタム アクション) を作成します。以下は、その手順の説明です。

タスク

Windows Installer 4.5 がインストールされているかどうかを判別して、インストールされていない場合にエラーメッセージを表示するには、以下の手順に従います:

1. **[カスタム アクションとシーケンス]** ビューで、**[カスタム アクション]** エクスプローラーを右クリックし、**[新しいエラー]** をクリックします。
2. 新しいカスタム アクションを選択します。
3. **"エラー メッセージ"** 設定に、Windows Insatller 4.5 がインストールされていない状態でエンド ユーザーがインストールを起動した場合に表示するエラー メッセージ テキストを入力します。
4. **"インストール UI シーケンス"** と **"インストール 実行シーケンス"** 設定で、エラー メッセージを LaunchConditions アクションの前にスケジュールするオプションを選択します。たとえば、このエラー アクションの一般的なシーケンスは **<最初のアクション>** です。
5. **"インストール UI 条件"** と **"インストール実行条件"** 設定で、次の条件を入力します:

   VersionMsi < "4.05"

ロックされたファイル/再起動の問題

InstallShield 前提条件が Windows Installer 4.5 エンジンをインストールするとき、既存のエンジン ファイルがロックされている場合がよくあります。ターゲット システムを再起動する前に、インストールが更新済みエンジンを必要とする場合は、以下の動作に注意してください。

Windows XP と Windows Server 2003 システムでは、Windows Installer 4.5 エンジン ファイルが直ぐにアップデートされ、再起動の要求が記録されて、メインのインストールが終了するまで (またはその前に起こる再起動が発生するまで) 遅延されます。メイン インストールはアップデート済みの Windows Installer 4.5 エンジンを使用し、インストールの最後で再起動を要求します。

Windows Vista と Windows Server 2008 システムでは、Windows Installer 4.5 エンジンは再起動後までアップデートされないため、即時、再起動が要求されます。エンド ユーザーがマシンの再起動を受け入れた場合、インストールは再起動の後に続行します。ただしエンド ユーザーがマシンの再起動を許可しなかった場合、インストールが終了します。再起動の保留中は、Windows Installer 4.5 エンジンは未インストールの状態となり、続けて前提条件を実行しても失敗して、インストールの続行が不可能となります。マシンの再起動を行うと、Windows Installer 4.5 のインストールが完了し、インストールを予定通りに続行できます。

.NET Framework 再配布可能ファイルをプロジェクトへ追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

製品の実行に、.NET Framework がターゲット システムにインストールされている必要がある場合、.NET Framework 再配布可能ファイルをプロジェクトに追加することができます。.NET Framework がターゲット システムにないとき、インストール中にインストールされます。

プロジェクトに、.NET Framework の言語パックの再配布可能ファイルを含めることもできます。言語パックには、英語以外の言語のための翻訳済みテキスト (エラー メッセージなど) が含まれています。

.NET Framework および .NET Framework 言語パックをプロジェクトに追加する方法は、使用しているプロジェクト タイプ、および製品が必要とする .NET Framework のバージョンによって異なります。

メモ: *一部の .NET Framework バージョンは、以前の .NET Framework バージョンを含みます:*

- *.NET Framework 3.5 は、.NET Framework 3.0 SP1 と .NET Framework 2.0 SP1 を含みます*
- *.NET 3.0 Framework SP1 は .NET Framework 2.0 SP1 を含みます。*
- *.NET 3.0 Framework RTM は .NET Framework 2.0 RTM を含みます。*

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

基本の MSI プロジェクトまたは InstallScript MSI プロジェクトに .NET サポートを含める場合、適切な Microsoft .NET Framework 前提条件を追加します: 詳細については、「Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める」を参照してください。

メモ: *.NET Framework 4.5 Full、4.5 Web、4.0 Full、4.0 Client、3.5 SP1、3.5、3.0 SP1、3.0、2.0 SP1、または 2.0 (x64、IA64 のみ) 再配布可能ファイルが含まれます。*

ターゲット システムに .NET Framework が既にインストールされているかどうかをテストするには、ビルトインの MsiNetAssemblySupport プロパティを使用することができます。.NET Framework がインストールされている場合、プロパティには、特定の .NET DLL (fusion.dll) のバージョンに設定され、インストールされていない場合、何も設定されません。

InstallScript プロジェクト

InstallScript プロジェクトで .NET Framework 再配布可能ファイルをプロジェクトに追加する方法は 2 種類あります:

- [前提条件] ビューを使って、プロジェクトに 1 つまたは複数の .NET Framework 前提条件を追加します。
- [オブジェクト] ビューを使って、インストールに Microsoft .NET Framework オブジェクトを追加します。このオブジェクトを追加すると、InstallScript プロジェクトに 1 つ以上の言語パックを追加することもできます。詳細については、「Microsoft .NET Framework オブジェクト ウィザード」を参照してください。

特定のバージョンの .NET Framework または言語パックがインストールされているかどうかを判別するには、Is 関数 を使用して、DOTNETFRAMEWORKINSTALLED 定義済み定数を渡します。

InstallShield 前提条件とオブジェクトの取得

一部の InstallShield 前提条件とオブジェクトは、InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

Microsoft .NET Framework および Microsoft .NET Framework 言語パックの前提条件を含める

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield には、一部のバージョンの .NET Framework と .NET Framework 言語パックの InstallShield 前提条件が含まれています。これらのバージョンの .NET Framework と言語パックを再配布する場合、基本の MSI、InstallScript および InstallScript MSI プロジェクトに、これらの InstallShield 前提条件を含めることができます。

以下は、InstallShield 前提条件として提供されている .NET Framework 再配布可能ファイルの一覧です。関連する言語パック 前提条件も、配布されている場合、含まれています。

- Microsoft .NET Framework 4.5(完全パッケージに 1 つ、Web パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。)
- Microsoft .NET Framework 4 Full。これは、.NET Framework 4 をターゲットにするアプリケーションを実行および開発するのに必要な .NET Framework ランタイムと関連ファイルをインストールします。(完全パッケージに 1 つ、Web ダウンロード パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。)
- Microsoft .NET Framework 4 Client。ほとんどのクライアントアプリケーションを実行するために必要な .NET Framework ランタイムおよび関連ファイルをインストールします。(完全パッケージに 1 つ、Web ダウンロー

ド パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。）

- Microsoft .NET Framework 3.5 SP1（完全パッケージに 1 つ、Web ダウンロード パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。）
- Microsoft .NET Framework 3.5（完全パッケージに 1 つ、Web ダウンロード パッケージに 1 つの InstallShield 前提条件が含まれています。サイズは Web ダウンロード パッケージのほうが小さいですが、実行時にターゲット システムでインターネット接続が必要になります。）
- Microsoft .NET Framework 3.0 SP1（これは、実行時にターゲット システムでインターネット接続が必要になる Web ダウンロード パッケージです。）
- Microsoft .NET Framework 3.0
- Microsoft .NET Framework 3.0 (x64)
- Microsoft .NET Framework 2.0 SP2
- Microsoft .NET Framework 2.0 SP2 (x64)
- Microsoft .NET Framework 2.0 SP2 (IA64)
- Microsoft .NET Framework 2.0 SP1 (x86)
- Microsoft .NET Framework 2.0 SP1 (x64)
- Microsoft .NET Framework 2.0 SP1 (IA64)
- Microsoft .NET Framework 2.0 (x64)
- Microsoft .NET Framework 2.0 (IA64)

ヒント: *.NET Framework 再配布可能ファイルの他のバージョンに関する詳細は、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。*

これらの InstallShield 前提条件のインストールは、サイレント モードで実行されます。したがって、.NET Framework インストールが実行されるときの言語は、問題になりません。

現在、Windows XP を実行している 64 ビット Itanium に Windows Installer 3.x をインストールできないことと、.NET Framework 2.0 以降に Windows Installer 3.x 以降が必要であるということのために、64 ビット .NET Framework 2.0 前提条件は、Windows XP を実行している 64 ビット Itanium にインストールすることはできません。

InstallShield 前提条件のインストールは、特定の HKEY_LOCAL_MACHINE キーの Install または InstallSuccess の値データを確認して、.NET Framework の対応するバージョンが既にターゲット マシンにインストールされているかどうかを確認します。詳細については、InstallShield 前提条件エディターで任意の InstallShield 前提条件を開いて、設定されている条件をご覧ください。

.NET Framework 再配布可能ファイルをプロジェクトへ追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクトが、ターゲット システム上に .NET Core のインストールが必要な場合、そのプロジェクトに .NET Core 再配布可能ファイルを追加することができます。ターゲット システムに .NET Core が搭載されていない場合、インストール中に .NET Core もインストールされます。

.NET Core をプロジェクトに追加する方法は、そのプロジェクトの種類およびプロジェクトで必要な .NET Core ランタイムのバージョンによって異なります。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

基本の MSI または InstallScript MSI プロジェクトに .NET サポートを含みたい場合、適切な Microsoft .NET Core 前提条件を追加してください。詳細については、「Microsoft .NET Core 前提条件を含める」を参照してください。

InstallShield 前提条件とオブジェクトの取得

一部の InstallShield 前提条件とオブジェクトは、InstallShield と共にインストールされませんので注意してください。これらについては、場合により、ダウンロードする必要があります。詳細については、「InstallShield のアップデートを取得する」を参照してください。

Microsoft .NET Core 前提条件を含める

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield には、.NET Core の一部のバージョンの InstallShield 前提条件が含まれています。これらのバージョンの MSI .NET Core を再配布したい場合、基本の MSI MSI、InstallScript MSI プロジェクトにこれらの InstallShield 前提条件を含めることができます。

InstallShield 前提条件として、次の .NET Core 再配布可能ファイルを使用できます:

- Microsoft .NET 8 Desktop Runtime 8.0.6 (x64)
- Microsoft .NET 8 Desktop Runtime 8.0.6 (x86)

ヒント: *.NET Core 再配布可能ファイルのその他のバージョンについての情報は、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。*

InstallShield 前提条件インストールは、特定の HKEY_LOCAL_MACHINE キーの Install または InstallSuccess 値データを確認して、対応するバージョンの .NET Core がターゲット マシン上に既にインストールされているかどうかを判別します。詳細は、InstallShield エディターで InstallShield 前提条件を開いて、設定されている条件を確認することができます。

MySQL Connector ODBC 前提条件を含める

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

MySQL Connector/ODBC 3.51 をインストールする InstallShield 前提条件をインストールに含めることができます。これにより、エンドユーザーよる ODBC を使用する MySQL データベース サーバーへの接続が可能になります。この InstallShield 前提条件をプロジェクトに追加する前に、MySQL Connector/ODBC ドライバーをダウンロードして、システムで InstallShield 前提条件を構成する必要があります。

タスク ***MySQL Connector ODBC 3.51 前提条件をシステムに追加して、プロジェクトに追加できるようにするには、以下の手順に従います:***

1. Windows エクスプローラーを開いて、InstallShield 前提条件テンプレート フォルダーを参照します。デフォルト保存先は次の場所です:

   C:¥Program Files¥InstallShield¥2022¥SetupPrerequisites¥Templates

2. Templates フォルダーにある MySQL Connector ODBC 3.51.prq ファイルをコピーし、InstallShield 前提条件フォルダーに貼り付けます。デフォルト保存先は次の場所です:

   C:¥Program Files¥InstallShield¥2022¥SetupPrerequisites

3. http://dev.mysql.com/downloads/connector/odbc/3.51.html にアクセスして、MySQL Connector/ODBC 3.51 driver for Windows 用の MSI インストーラーをダウンロードしてください。

4. ファイルを次の場所に保存します。

   InstallShield Program Files フォルダー¥Objects¥MySQL¥Redist

次回 InstallShield を起動したとき、MySQL Connector ODBC 前提条件が、[再配布可能ファイル] ビューに表示されます。

マシン上で MySQL Connector/ODBC 3.51 ドライバーのインストーラーを保存する場所を変更する場合、InstallShield 前提条件エディターで MySQL Connector ODBC 3.51.prq ファイルを開いて、設定を変更します。InstallShield 前提条件エディターを開くには、[ツール] メニューで、[前提条件エディター] をクリックします。詳細については、「InstallShield 前提条件のファイルを指定する」を参照してください。

Oracle 11g Instant Client の InstallShield 前提条件を含める

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールが実行される前に、Oracle 11g Instant Client をインストールする場合、Oracle 11g Instant Client 前提条件をプロジェクトに含めます。この InstallShield 前提条件をプロジェクトに追加する前に、Oracle Instant Client をダウンロードして、システムで .msi パッケージを作成する必要があります。詳細については、「Oracle のインスタンスに接続して SQL スクリプトを実行する」を参照してください。

メモ: *Oracle 11g InstaInt Client がターゲット システム上で必要な時にインストールがこれをダウンロードする場合、Oracle に問い合わせて、それが可能かどうかを確認してください。可能な場合、独自の Web サイトでダウンロードをホストして、InstallShield 前提条件にそのダウンロード パスを追加しなくてはなりません。Revenera では、このインストールのダウンロードを提供していません。*

DirectX 9.0 オブジェクトを含める

DirectX は、最新のグラフィック カードを含む、マルチメディア アプリケーションおよびハードウェアの API ライブラリがサポートとしています。製品の実行に DirectX がターゲット システムにインストールされている必要がある場合、DirectX オブジェクト (基本の MSI と InstallScript MSI プロジェクトの場合、Windows Installer ベースのオブジェクト、InstallScript プロジェクトの場合、InstallScript オブジェクト) をプロジェクトに追加することができます。ターゲット システムに DirectX がない場合、DirectX がインストール時にインストールされます。

インストール後、DirectX ランタイムはアンインストールできません。DirectX はシステム コンポーネントであるため、エンドユーザーが DirectX をアンインストールするには、オペレーティング システムの再インストールが必要になります。

ヒント: *DirectX オブジェクトは InstallShield と共にインストールされないため、ダウンロードする必要があります。基本の MSI プロジェクト または InstallScript MSI プロジェクトに DirectX オブジェクトを含めるには、MSI オブジェクトのダウンロードを入手します。基本の MSI プロジェクト または InstallScript MSI プロジェクトに DirectX オブジェクトを含めるには、MSI オブジェクトのダウンロードを入手します。詳細については、「InstallShield のアップデートを取得する」を参照してください。*

再配布可能ファイル

DirectX オブジェクトは、32 ビットおよび 64 ビット固有のコンポーネントを含む、最新の DirectX 9.0c コアおよびオプションのコンポーネントをすべてインストールします。

基本の MSI プロジェクトと InstallScript MSI プロジェクトに DirectX オブジェクトを含める

DirectX オブジェクトを基本の MSI または InstallScript MSI プロジェクトに含めると、DirectX オブジェクト ウィザードが起動されます。

DirectX オブジェクトは、圧縮、非圧縮の両方のインストールで使用することができます。DirectX オブジェクト ウィザードを利用して、DirectX ファイルを Disk1 フォルダーに含めるか、または .msi ファイルにストリームするかを指定することができます。

- ファイルを Disk1 フォルダーにあるフォルダーに含めるように指定すると、ビルド時に作成中のインストールに対して DirectX フォルダーが作成され、リリースの Disk1 フォルダーに配置されます。DirectX フォルダーは [サポート ファイル] ビュー (基本の MSI プロジェクトの場合)、または [サポート ファイル/ビルボード] ビューの Disk1 領域に表示されます。
- ファイルを Disk1 フォルダーに含めないように指定すると、ファイルはインストールの .msi ファイルに埋め込まれます。これらのファイルは、[サポート ファイル] ビューの [言語非依存] 領域 (基本の MSI プロジェクトの場合) または [サポート ファイル/ビルボード] ビュー (InstallScript MSI プロジェクトの場合) で一覧表示されます。

メモ: *DirectX インストールを起動するカスタム アクションは Windows Vista 以降のシステムで昇格された権限を使って実行できるように、[実行] シーケンスにスケジュールされ、遅延システム コンテキストで実行されます。*

基本の MSI プロジェクトと InstallScript MSI プロジェクトで DirectX オブジェクト ファイルを更新する

ある DirectX ファイルのアップデートを入手して、DirectX オブジェクトに含める場合、それらを適切な InstallShield Program Files サブフォルダーに他の DirectX ファイルと共に配置します。保存先:

InstallShield Program Files フォルダー¥Objects¥DirectX9c¥Redist

一部のファイルが製品で必要なく、インストールに含める必要がない場合、それらをフォルダーから削除することもできます。InstallShield の現在のバージョンがリリースされてからマイクロソフトがリリースしている可能性があるアップデートに関する詳細など、DirectX 再配布可能ファイルの関する詳しい情報は、最新の DirectX SDK または マイクロソフトの MSDNWeb サイト をご覧ください。

ビルド時に、InstallShield がリリースのビルドをするとき、DirectX フォルダー内の再配布可能ファイルはすべて使用されます。

アプリケーションの依存関係を識別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

あるファイルが、他のファイルの関数に依存してタスクを実行することがよくあります。ただし、インストールプロジェクトにアプリケーションファイルを含める際に、*「依存関係」*と呼ばれる他のファイルに気が付かない場合があります。InstallShieldでは、これらのファイルを見つけて作業を行うスキャン ウィザードが用意されています。スキャナーは、依存関係スキャナー ビューでアクセスできます。

テーブル 13 • 利用可能なスキャン ウィザード

スキャナー	説明
スタティック スキャン ウィザード	プロジェクトのポータブル実行可能ファイル（例、.exe、.ocx、.com、.tlb、.hlp、および .chm）を確認して、必要な依存関係を検出します。
ダイナミック スキャン ウィザード	実行可能ファイルが実行中にシステムを監視して、実行可能ファイルで必要となる可能性がある .dll または .ocx ファイルを確認します。

これらのスキャナーのどれか 1 つを使って基本の MSI または InstallScript MSI プロジェクトに追加されるファイルは、セットアップベストプラクティス に従って追加されます。

スタティックおよびダイナミック スキャン ウィザードを使用する場合、InstallShield を使ってスタティックまたはダイナミック スキャンを実行するとき必ず自動的に選択または除外されるファイルを指定することができます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

スタティック スキャン

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザードを利用して、プロジェクトに既に追加されているファイルをスキャンして必要な依存関係があるかどうかをチェックすることができます。このウィザードはプロジェクトに含まれるすべてのポータブル実行可能ファイル（.exe、.dll、.ocx、.sys、.com、.drv、.scr、および .cpl ファイル）をスキャンして、必要な可能性がある依存関係を検出します。ウィザードで検出された依存関係のリストが表示され、それぞれをプロジェクトに含めるかどうかを指定することができます。

プロジェクトに追加された新規ファイルは、ファイルが依存している同じ機能に追加されるので、インストールが必要なときに確実に実行されます。

[ビルド時に .NET スキャン] 設定によるスタティック スキャンへの影響

基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで、コンポーネントの "ビルド時に .NET をスキャン" 設定で選択された値は、スタティック スキャン ウィザードがそのコンポーネントに含まれるファイルをスキャンする方法に影響します。

テーブル 14 ・ スタティック スキャン ウィザードがコンポーネントに含まれるファイルをスキャンする方法を判断する

コンポーネントの "ビルド時に .NET をスキャン" 設定の値	説明
なし	スタティック スキャン ウィザードは、コンポーネントの .NET 部分の依存関係もプロパティもスキャンしません。たとえば、コンポーネントに Notepad.exe と .NET アセンブリファイルが含まれている場合、スタティック スキャン ウィザードは Notepad.exe の依存関係を調べますが、.NET アセンブリ ファイルの依存関係とプロパティの*いずれもスキャンしません*。
プロパティのみ	スタティック スキャン ウィザードは、.NET アセンブリを含むコンポーネントのすべてのファイルをスキャンしてプロパティを調べますが、依存関係を調べてコンポーネントの .NET 部分の特定はしません。たとえば、コンポーネントに Notepad.exe と .NET アセンブリファイルが含まれている場合、スタティック スキャン ウィザードは両方のファイルの依存関係を調べますが、依存関係の .NET アセンブリファイルはスキャンしません。
依存関係およびプロパティ	スタティック スキャン ウィザードは、.NET アセンブリを含むコンポーネントのすべてのファイルをスキャンして、依存関係とプロパティを調べます。スキャン中に見つかった依存関係は、[ファイル選択] パネルに表示されます。.NET ファイルにプロパティが検出された場合、InstallShield によってそのプロパティがコンポーネントの [詳細設定] 領域にある [アセンブリ] ノードの下の [.NET アセンブリ] サブノードに追加されます。

ダイナミック スキャン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードは、実行可能ファイルの実行中にシステムを監視する使い易いツールです。ウィザードでは実行可能ファイルが必要とする可能性のある .dll および .ocx ファイルのリストが表示され、それぞれをプロジェクトに含めるかどうかを指定できます。

スキャンする実行可能ファイルは、既にプロジェクトに含められているものでも、後からウィザードで追加するものでも構いません。

このウィザードに関する詳しい情報は、「ダイナミック スキャン ウィザード」をご覧ください。

64 ビット依存関係のスキャン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Windows Vista 以降の 64 ビット バージョン、または Windows Server 2008 以降の 64 ビット バージョンで InstallShield を使用する場合、スタティック スキャン ウィザードおよびダイナミック スキャン ウィザードは、プロジェクトに含まれる 64 ビット ファイルの 64 ビット依存関係をスキャンすることができます。これらのウィザードは、プロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のスキャンも行います。

Windows の 32 ビット バージョンで InstallShield を使用する場合、これらのウィザードはプロジェクトに含まれる 32 ビット ファイルの 32 ビット依存関係のみをスキャンできます。プロジェクトに 64 ビット ファイルが含まれている場合、必要に応じてプロジェクトに依存関係を手動で追加することができます。

依存関係スキャナー結果の確認

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザード およびダイナミック スキャン ウィザードを使ってプロジェクトをスキャンして、製品が必要とする可能性のあるその他のファイルを識別することができます。それぞれのウィザード パネルには、プロジェクトに追加する必要がある可能性の高いファイルおよびマージ モジュールのリストが表示されます。スキャン結果をよく確認して、指定された各ファイルやマージ モジュールをプロジェクトに追加する必要があるかどうかを判断してください。

両方のスキャナーは、異なる方法で依存関係を識別します。1 つのキャナーでは識別されなかった依存関係が、もう 1 つのスキャナーによって識別される場合もあります。そのため、スタティック スキャナーおよびダイナミック スキャナーの両方を使って、依存関係の可能性があるファイルのより完全なリストを作成することをお勧めします。一部の状況において、依存関係スキャナーが製品で必要のないファイルまたはマージ モジュールを依存関係として識別する場合もあります。その場合、ウィザードでは識別された各依存関係を含めたり除外したりすることが可能なので、不要なファイルまたはマージ モジュールを除外することができます。さらに、InstallShield を使ってスタティックまたはダイナミック スキャンを行うときは常に自動的に含めたり除外したりするファイルをマシン全体で指定することも可能です。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

依存関係を識別するときに最も良い結果を得るために、クリーン マシン上で製品とそのインストールを十分にテストすることが推奨されます。製品が予定通りに動作しなかった場合、マシン上で足りない依存関係がないか、またそれをインストールに含むべきかどうかを判断してください。

依存関係スキャナーでファイルをフィルターする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティックおよびダイナミック スキャン ウィザードを実行すると、インストールに追加したくない依存関係ファイルが一覧表示されることがあります。スキャナーを実行するたびにこれらのファイルが追加されないようにするために、Filters.xml を編集できます。このファイルを使うと、スキャナーが無視する、または含めるファイルを指定できます。

Filters.xml は、以下の場所にあります:

InstallShield Program Files フォルダー¥Support

ファイルは編集後もこの場所に残る必要があります。そうでない場合、スキャナーが正しく動作しません。

ヒント: Filters.xml *ファイルを使って、COM 抽出中にどのレジストリ項目を除外するのかを制御することもできます。詳細については、「COM 抽出のレジストリ変更をフィルターする」を参照してください。*

Filters.xml *ファイルを使って、IIS Web サイトをインポートするときに除外する IIS 設定を制御できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

ファイルを除外する

Filters.xml ファイルの <Exclude> 要素には、スキャナーが除外する各ファイルのサブ要素を追加します。ここにリストされたファイルは、スキャナーによってインストール プロジェクトに追加されることはありません。

デフォルトで <Exclude> 要素には、すべての Windows ベースのマシン上に存在する一般的なシステム ファイルのサブ要素が含まれています。

ファイルを含める

Filters.xml ファイルの <Include> 要素を使うと、Exclude 要素のサブ要素である個別のファイルをオーバーライドすることができます。スキャナーは、<Include> 要素のサブ要素にリストされているファイルをすべてインストール プロジェクトに追加します。これは <Exclude> 要素のサブ要素にリストされているファイルにも適用されます。

メモ: *次の重要なオペレーティング システム ファイルは、<Include> 要素のサブ要素に追加してスキャナーでは認識されません。*

- *kernel32.dll*
- *ntdll.dll*
- *user32.dll*
- *gdi32.dll*
- *advapi32.dll*

- *shell32.dll*
- *ole32.dll*

<Exclude> および <Include> 要素にファイルを指定する

<Exclude> または <Include> 要素にファイルをリストする場合、そのファイルをサブ要素として追加しなくてはなりません。適切にフォーマットされたサブ要素のサンプルは次の通りです。

```
<File name="myfile.dll" path="[SystemFolder]" We="needthis"/>
```

テーブル 15 · <File> サブ要素で利用可能な属性

属性	説明
名前	この属性は小文字のみ使用可能です。この属性の値 (例えば、前述の **myfile.dll**) は、含めるまたは除外するファイルの名前を示します。
パス	この属性はオプションです。この属性の値 (例えば、前述の **[SystemFolder]**) は、ファイルのパスを示します。

その他の属性はすべてオプションで、スキャナーが認識することはありません。例えば前述の **We** 属性と、それに対応する **"needthis"** 値のように、特定の項目が含まれるまたは除外される理由を説明する属性を追加したい場合があります。

重要: *その場合、XML コードが適切に書かれていることを再確認してください。不適切な場合には、すべてのフィルターが失敗します。多くの場合、インターネット エクスプローラーで* Filters.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Filters>、<Include>、および <Exclude> 要素は縮小および展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。*

<Exclude> または <Include> 要素にサブ要素を追加した場合、それらが誤ってコメントアウトされたセクションに配置されていないことを確認してください。InstallShield は Filters.xml *ファイルのコメントアウト部分を無視します。*

次のサンプル XML コードで、Filters.xml ファイルの形式を説明します。

```
<Filters>
 <Include>
  <!—この要素にファイルを追加する方法
  -->
  <File name="mfc42.dll" We="needthis"/>
 <Include>
 <Exclude>
  <!—この要素にファイルを追加する方法
  -->
  <Registry key="HKEY_CLASSES_ROOT¥Interface¥{00020404-0000-0000-C000-000000000046}"/>
  <File name="12520437.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
  <File name="12520850.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 </Exclude>
</Filters>
```

COM サーバーの登録

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ほとんどのアプリケーションでは、正常に機能するために特定の COM サーバーが必要です。COM サーバーがオペレーティング システムで認識されるようにするには、登録が必要です。

従来型の COM を使用すると、COM でーたは、レジストリに直接書き込まれます。レジストリ フリー COM (Reg-Free COM)、従来型の方法に代わる簡単な方法を提供します。Reg-Free COM を使用すると、COM データは、アプリケーション マニフェスト ファイルに書き込まれます。

- 従来型の COM 登録
- レジストリ フリー COM 登録

従来型の COM 登録

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

InstallShield では、ターゲット マシンへ COM サーバー登録するため方法が複数サポートされています。以下の方法は、従来型の方法です。

- ビルド時 (またはデザインタイム) に COM 情報をファイルから抽出し、.msi データベースの COM 関連データベースに書き込む方法。
- COM サーバーの自己登録関数をインストール時に呼び出す方法。

リストされている最初の方法 (COM 情報を抽出して、.msi データベースに書き込む) は、自己登録登録方法よりも好まれています。最初の方法は、.msi データベースの Class、ProgID、および **Registry** テーブルに COM クラス情報を書き込みます。

このセクションには、次のトピックが含まれています:

- COM サーバーが自己登録をサポートするかどうかを判断する
- COM 情報を COM サーバーから抽出する

- COM 抽出のレジストリ変更をフィルターする
- COM サーバーの自己登録

COM サーバーが自己登録をサポートするかどうかを判断する

すべての COM サーバーが自己登録をサポートするとは限りません。通常、インストール内でどのファイルが自己登録かどうかは明らかな場合が多いですが、そうではない場合もあります。自己登録ファイルの扱いは、非自己登録ファイルと異なるため、特定のファイルが自己登録かどうか判断できる必要があります。

通常、InstallShield プロジェクトにファイルを追加する際、ファイルが自己登録かどうか判別できます。InstallShield は、COM サーバー ファイルがプロジェクトに追加されると、それが自己登録ファイルがどうか判断するためにテストを行います。InstallShield が、プロジェクトに追加された COM サーバーが自己登録であると判断した場合、次のような操作が行われます。

- Windows Installer ベースのプロジェクトの場合: ファイルのコンポーネントの "ビルド時に COM 抽出" プロパティは [はい] に設定されます ([オプション] ダイアログ ボックスの [プリファレンス] タブで、[ビルド時に COM 抽出が行われます] が指定されているとき)。
- InstallScript ベースのプロジェクトの場合: ファイルは、自己登録としてマークされます。

COM 情報を COM サーバーから抽出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

MSI データベースと MSM データベース プロジェクトの場合、COM 情報をビルド時に抽出することは不可能なため、デザイン時に抽出します。

COM サーバーの登録には、.msi データベース内の Windows Installer テーブルを使って必要なレジストリ エントリを作成する方法が推奨されます。この方法を利用すると、Windows Installer はファイルをアドバタイズされたときに登録することが可能で、またインストールが失敗した場合に登録情報を確実にロールバックすることができます。

各コンポーネントはポータブル実行可能ファイルを 1 つしか持たないため、すべての登録情報はコンポーネントのキーファイルとしてマークされた単一のファイルに属すると想定されます。

InstallShield は、COM サーバーから COM 情報を抽出するいくつかの方法をサポートします。

- コンポーネント ウィザードを使用します。このウィザードを利用して、新規 COM サーバーコンポーネントを作成します。この方法を使うと、ウィザードはコンポーネントのキーファイルから COM 情報のスキャンを 1 回だけ行います。

- コンポーネントの"ビルド時に COM 抽出" 設定を [はい] に設定します。リリースをビルドする度に COM データが抽出されるため、この方法は COM サーバーのインターフェイスが頻繁に変更される場合に有効です。この場合、COM サーバーがそのコンポーネントのキーファイルでなくてはなりません。
- [ファイルとフォルダー] ビューにあるファイルを右クリックしてから、[キーファイルの COM データを抽出] をクリックし、情報を抽出します。COM サーバーファイルをコンポーネントへ追加済みの場合は、この方法を使います。

InstallShield が COM データを抽出する方法の代わりに、手動で コンポーネントの詳細設定を使って COM 情報をコンポーネントに追加する こともできます。

リリースをビルドする度に自動的に COM 情報を抽出したくない場合は、コンポーネントの "ビルド時に COM 抽出" 設定で {いいえ] を選択します。InstallShield に、コンポーネントの COM 登録詳細設定に静的に含まれる情報に従って自動的にファイルを登録させるには、この方法を使います。この詳細設定には、コンポーネントウィザードで抽出されたか、または詳細設定に手動で入力された、ファイルの COM クラス、PprogID などの情報が保存されます。

InstallShield が COM サーバーから COM 情報を抽出する時、COM クラス情報は .msi データベースの Class、ProgId および Registry テーブルに書き込まれます。

InstallShield を 64 ビット オペレーティング システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

InstallShield では、登録せずに COM サーバー上のすべての COM 情報を見つける特別な技術を採用しています。したがって COM 抽出をしてもシステムレジストリは影響を受けません。ただし一部の COM サーバーの登録プロセスは既存のレジストリ エントリ値によって変わってきます。InstallShield はこのようなシナリオを回避するアルゴリズムを作成しましたが、極端な場合は絶対安全とは言えません。

注意 *・WinRunner など一部のアプリケーションは、COM 抽出エンジンにフック .dll ファイルを挿入します。その結果、COM 抽出が失敗してメッセージ「ISRegSpy は、次のモジュール %1 がこのプロセスにフックされ、そのために ISRegSpy の誤動作が発生していることを検出しました。アプリケーションをシャットダウンして COM 抽出を再起動してください。」が表示されます。このメッセージが表示された場合、ダイアログ ボックスの指示通りアプリケーションをシャットダウンして COM 抽出を再開します。*

自己登録ではない .exe ファイルには、"自己登録" プロパティを選択しないでください。.exe ファイルを自己登録するには、/regserver コマンドを使って .exe ファイルを起動する必要があります。しかし .exe ファイルがコマンドラインスイッチをサポートしない場合、ビルド時の抽出中に .exe が起動されます。

メモ *・COM 関連の Windows Installer テーブルの定義は、COM サーバーが 2 つ以上の機能に配置されるのを防ぎます。1 つのCOM サーバーを 2 つ以上の機能に配置する必要がある場合は、次のオプションのどれかを使用してください。*

- *その COM サーバーが最初の機能の子となるような 2 つ目の機能を作成し、親機能にファイルを配置する。*
- *COM 関連の Windows Installer データベーステーブルを使用せずに、COM サーバーの自己登録を使用する。*

COM 抽出のレジストリ変更をフィルターする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

InstallShield による COM サーバーからの不必要な COM データの抽出 (ビルド時またはデザイン時) を阻止するために、Filters.xml ファイルを編集して特定のレジストリ キーを除外することができます。Filters.xml は、以下の場所にあります:

InstallShield Program Files フォルダー¥Support

ファイルは編集後もこの場所に残る必要があります。そうでない場合、COM 抽出が正しく動作しません。

ヒント: Filters.xml *ファイルを使って、依存関係のスキャン中にどのファイルを除外するまたは含めるのかを制御することもできます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。*

Filters.xml *ファイルを使って、IIS Web サイトをインポートするときに除外する IIS 設定を制御できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

COM 抽出からレジストリ キーを除外する

Filters.xml ファイルの <Exclude> 要素には、COM 抽出プロセスが除外する各レジストリ キーのサブ要素を追加します。ここにリストされているすべてのキーは、製品がアンインストールされるのと同時にアンインストールされることはありません。

デフォルトで、<Exclude> 要素は必要とされる一般的なシステム レジストリ キーのサブ要素を含みます。

<Exclude> 要素にレジストリ キーを指定する

<Exclude> 要素にキーをリストする場合、そのキーをレジストリ サブ要素として追加しなくてはなりません。

InprocServer32 レジストリ キー、そのすべての値、およびそのサブキーのすべてに対する変更を阻止する、適切にフォーマットされたレジストリ サブ要素のサンプルは次の通りです。

```
<Registry key="HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32" />
```

InprocServer32 レジストリ キーのデフォルト値のみに対する変更を阻止する、適切にフォーマットされたレジストリ サブ要素は次の通りです。

```
<Registry key="HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32" value="" />
```

InprocServer32 レジストリ キーのThreadingModel 値名のみに対する変更を阻止する、適切にフォーマットされたレジストリ サブ要素は次の通りです。

<Registry key="HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32" value="ThreadingModel"/>

テーブル 16 ・ <Registry>サブ要素で利用可能な属性

属性	説明
キー	この属性は小文字のみ使用可能です。この属性の値 (例えば、前述の HKEY_LOCAL_MACHINE¥SOFTWARE¥Classes¥CLSID¥{00000231-0000-0010-8000-00AA006D2EA4}¥InprocServer32) は、フィルターを行うレジストリ キーの名前を示します。
値	この属性はオプションです。 ・ レジストリ キー全体に対する変更を阻止するには、値属性を含まないでください。 ・ 特定のキーのデフォルト値に対する変更を阻止するには、この属性の値をヌルに設定します。 ・ 特定のキーの値名に対する変更を阻止するには、この属性の値をそのレジストリ値の名前に設定します。

<Registry> サブ要素のその他の属性はオプションで、COM 抽出プロセスでは認識されません。特定の項目が除外される理由を説明するために、属性を追加することもできます。

重要・ *その場合、XML コードが適切に書かれていることを再確認してください。不適切な場合には、すべてのフィルターが失敗します。多くの場合、インターネット エクスプローラーで* Filters.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Filters>、<Include>、および <Exclude> 要素は縮小および展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。*

<Exclude> または <Include> 要素にサブ要素を追加した場合、それらが誤ってコメントアウトされたセクションに配置されていないことを確認してください。InstallShield は Filters.xml *ファイルのコメントアウト部分を無視します。*

次のサンプル XML コードで、Filters.xml ファイルの形式を説明します。

```
<Filters>
 <Include>
  <!—この要素にファイルを追加する方法
  —>
 <Include>
 <Exclude>
  <!—この要素にファイルを追加する方法
  —>
  <Registry key="HKEY_CLASSES_ROOT¥Interface¥{00020404-0000-0000-C000-000000000046}"/>
  <File name="12520437.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
  <File name="12520850.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 </Exclude>
</Filters>
```

COM サーバーの自己登録

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

自己登録の COM サーバーをお持ちの場合、インストール時に COM サーバーの自己登録関数を呼び出して、ターゲットマシンに COM サーバーを登録することができます。

InstallShield では、COM 関連ファイル (.dll、.ocx、.exe、.tlb、.olb) が自己登録であることを示すための異なるメソッドがサポートされています。

- 特定のファイルが自己登録であることを指定できます。詳細については、「[ファイルのプロパティ] ダイアログ ボックス」を参照してください。
- ダイナミック リンクが自己登録であることを指定できます。詳細については、「[ダイナミック ファイル リンクの設定] ダイアログ ボックス」を参照してください。

ヒント: *自己登録ファイルが 64 ビット コンポーネントの一部の場合、64 ビット自己登録がターゲット マシンで実行されます。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

COM サーバーの .dll ファイルまたは .ocx ファイルを自己登録としてマークすると、ファイルの独自の登録関数 (**DllRegisterServer**) がインストール中に呼び出されてターゲット システムに登録され、アンインストール中には登録解除関数 (**DllUnregisterServer**) が呼び出されてファイルが登録解除されます。

DllRegisterServer を指定してファイルを登録すると、いくつかの制限があります。

- **DllRegisterServer** に登録した COM 情報はアドバタイズできない。
- **DllRegisterServer** は .dll コードであるため、インストールに失敗した場合にそれによる変更を正しくロールバックできるという保証がない。
- **DllRegisterServer** は、ユーザーごとまたはマシンごとの COM 情報の違いを判別できません。
- 自己登録時、COM サーバーは、場合によって、特定の順番で登録される必要があります。InstallShield 自己登録 (ISSelfReg) を使用すると、この制限を回避できます。
- 通常、**DllRegisterServer** は相対パスを使用してファイルを登録しない。相対パスはサイドバイサイド共有をサポートしているシステムに必要なものです。

また、パッチで任意の種類の自己登録を使用すると、パッチはアンインストール不可能となります。

1 つまたは複数のファイルを自己登録としてマークすると、InstallShield はデータを .msi データベースのテーブルにデータを追加し、自己登録に関連したカスタム アクションを [インストール]–[実行] シーケンスに追加します。詳細については、「自己登録メソッド」と「InstallShield 自己登録 (ISSelfReg)」を参照してください。

マシンごとの自己登録情報はレジストリキー HKLM¥Software¥Classes¥CLSID に格納され、ユーザーごとのデータは HKCU¥Software¥Classes¥CLSID に格納されます。マージされたデータのビューは、HKCR¥CLSID の下に表示されます。

自己登録メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[オプション] ダイアログの [プリファレンス] タブで、InstallShield 自己登録メソッド (ISSelfReg) または Windows Installer 自己登録メソッド (SelfReg) のどちらを使用するかを選択できます。

メモ: *パッチで任意の種類の自己登録を使用すると、パッチはアンインストール不可能となります。*

Windows Installer 自己登録

Windows Installer 自己登録は次の特徴があります。

- .dll ファイルと .ocx ファイルのみを登録する。
- ファイルの登録順は任意である。
- 自己登録ファイルが 64 ビット コンポーネントの一部である場合、64 ビット ファイルを登録できる。

InstallShield 自己登録

InstallShield 自己登録は次の特徴があります。

- .exe ファイル、.tlb ファイル、.dll ファイルおよび .ocx ファイルを登録できる。
- バッチモード登録を使用してランタイムのすべての登録依存を処理する。
- オーバーヘッドで70 KB の自己登録エンジンを含める必要がある。
- 自己登録ファイルが 64 ビット コンポーネントの一部である場合、64 ビット ファイルを登録できる。

メモ: *InstallShield 自己登録の場合、実行時に登録するファイルの順番を指定できます。順番は、ダイレクト エディター の ISSelfReg テーブル(Order 列)で設定できます。*

InstallShield 自己登録 (ISSelfReg)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

COM サーバーの自己登録方法として ISSelfReg を選択した場合で、プロジェクトに自己登録としてマークされているファイル (またはダイナミックリンク) が含まれている場合、InstallShield はこれらのファイルについての情報を自動的に .msi データベースの ISSelfReg テーブルへ追加します。ISSelfReg テーブルは、ダイレクト エディターで参照および編集できます。このテーブルのフィールドについての詳細は、「ISSelfReg テーブル」をご覧ください。

さらに、プロジェクトに自己登録指定のファイル (またはダイナミックリンク) が含まれている場合、InstallShield は次のカスタム アクションを [インストール]-[実行] シーケンスに追加します。

テーブル 17 ・ 自己登録 COM サーバーがあるプロジェクトに追加されたカスタム アクション

アクション	説明
ISSelfRegisterCosting	**ISSelfReg** テーブルを読み取って、登録または登録解除するファイルを決定する、[即時実行] アクション。ファイルはそのコンポーネントがインストールされる予定に合わせて登録され、アンインストールされる予定に合わせて登録解除されます。
ISUnSelfRegisterFiles	コンポーネントの削除が予定されている各ファイルを登録解除する [遅延実行] カスタム アクション。
ISSelfRegisterFiles	コンポーネントのインストールが予定されている各ファイルを登録する [遅延実行] カスタム アクション。
ISSelfRegisterFinalize	自己登録に失敗したファイルのエラー情報を表示します。

レジストリ フリー COM 登録

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Reg-Free COM を使用すると、COM データは、アプリケーション フォルダーに格納されているアプリケーション マニフェスト ファイルに書き込まれます。マニフェスト ファイルは、アプリケーションおよびそれに関連付けられているライブラリに関する情報を含む XML ファイルです。Reg-Free COM マニフェスト ファイル、実行可能ファイル、COM ライブラリはすべて、ターゲット マシン上の同一のフォルダーにインストールする必要があるので注意してください。

Reg-Free COM の利点

Reg-Free COM には、従来型 COM にはなかったいくつかの利点があります。たとえば、Reg-Free COM を使用すると、コンポーネントは、アプリケーション自体のスコープ内で定義されます。同一の COM コンポーネント (異なるバージョンも含む) を使用する他のアプリケーションがそれの登録を要求してきた場合も、このアプリケーションには干渉しません。

複数のバージョンの共有ライブラリがターゲット システムに存在する場合、従来型の COM 登録に関して問題が発生する可能性があります。たとえば、インストールが共有ライブラリの新しいバージョンを古いバージョン、または、古いバージョンと後方互換性を持たない新しいバージョンで上書きしたりするということが起こりえます。このようなことが発生した場合、特定のバージョンの機能が必要なアプリケーションがクラッシュする可能性があります。このような状況は、一般に *DLL Hell (DELL 地獄)* と呼ばれます。Reg-Free COM を使用すると、他のアプリケーションは作成したアプリケーションの COM コンポーネントにアクセスできないため、これらの問題を回避することができます。

さらに、Reg-Free COM はアップグレードおよびアンインストール処理を簡素化します。アップグレードの場合、単純にアプリケーション フォルダーを置き換えるだけです。アンインストールの場合、単純に同フォルダーを削除します。

Reg-Free COM の制限

Reg-Free COM は、一部のソリューションには適しません。以下のようないくつかの制限があります。

- Reg-Free COM は、Windows XP 以降でのみ動作します。
- コンポーネントは、それがシステム コンポーネントまたはオペレーティング システムの一部であるとき、Reg-Free COM には適しません。また、MDAC (Microsoft Data Access Components) などのデータ アクセス コンポーネントである場合も、不適当です。これらのタイプのコンポーネントは、分離することができません。
- COM コンポーネントは、1 つのアプリケーションにつき一回のみ分離することができます。このような制限の対処策として、単一クラス ライブラリにある複数の COM コンポーネントをグループ化することも考えられます。

このセクションには、次のトピックが含まれています:

- インストールに Reg-Free COM ファイルを作成および変更する
- Ref-Free COM ファイルのサンプル マニフェスト ファイル

インストールに Reg-Free COM ファイルを作成および変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*

Reg-Free COM ウィザードを使用して、インストールに含める Reg-Free COM マニフェスト ファイルを作成および変更することができます。Reg-Free COM ウィザードを使用する前に、COM ライブラリ (.dll および .ocx ファイル) と、それを使用する実行可能ファイルを InstallShield プロジェクトに追加する必要があります。

ウィザードは、COM 情報をアプリケーション マニフェスト ファイルに指定および追加したライブラリから抽出します。また、ウィザードは、新規のコンポーネントを、マニフェスト ファイルがキーファイルとして設定された状態で作成します。マニフェスト コンポーネントは、関連付けられた実行可能ファイル コンポーネントと同じインストール先および条件を持ちます。

Ref-Free COM ファイルのサンプル マニフェスト ファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

マニフェスト ファイルのコンテンツには、実行可能ファイル名、実行可能ファイルに関連付けられたライブラリの名前、および、COM 情報が含まれます。次のマニフェスト ファイルのサンプルでは、実行可能ファイル名は **Sample.exe**、ライブラリ名は **SampleCircle.dll** になっています。

```
<?xml version="1.0" encoding="utf-8"?>
<assembly xsi:schemaLocation="urn:schemas-microsoft-com:asm.v1 assembly.adaptive.xsd" manifestVersion="1.0"
xmlns:asmv1="urn:schemas-microsoft-com:asm.v1" xmlns:asmv2="urn:schemas-microsoft-com:asm.v2" xmlns:dsig="http://www.w3.org/
2000/09/xmldsig#" xmlns="urn:schemas-microsoft-com:asm.v1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <assemblyIdentity type="win32" name="Sample.exe" version="1.0.0.10" />
    <file asmv2:size="24576" name="SampleCircle.dll">
        <hash xmlns="urn:schemas-microsoft-com:asm.v2"> ...</hash>
        <typelib flags="HASDISKIMAGE" helpdir="" resourceid="0" tlbid="{6A2304BC-F200-49B8-955E-0ACDE272B6E0}" version="1.0" /
>
        <comClass clsid="{C621F4D3-8463-4B0C-9BEC-84D979C41385}" description="SampleCircle.Server"
progid="SampleCircle.Server" threadingModel="Apartment" tlbid="{6A2304BC-F200-49B8-955E-0ACDE272B6E0}" />
    </file>
</assembly>
```

Windows XP システム上にマニフェスト ファイルがあると、アプリケーションは、ライブラリがターゲット システムのレジストリに登録されていなくても COM ライブラリからメソッドを使用することができます。Windows XP は、ターゲット システムに登録されている可能性がある他のバージョンのライブラリを使用する前に現在のディレクトリにある COM ライブラリを使用します。

プロジェクトに DIM を追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

機能と同じサイズのデベロッパー インストール マニフェスト (DIM) は、インストール パッケージの別個の部分を構成している製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM プロジェクトによって、リリース エンジニアは、機能別に分かれたインストールの異なる部分を効率的に再利用することができます。

また、DIM を利用することにより、複数の開発者が、インストールの開発を同時に携わることができます。各ソフトウェア開発者は、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。

このセクションでは、基本の MSI インストール プロジェクトへの DIM プロジェクトの追加の仕方について説明します。また、1 つ以上の DIM を含む基本の MSI プロジェクトの使用方法についても説明されています。

このセクションには、次のトピックが含まれています:

- インストール プロジェクトに DIM を統合するための適切な方法を判別する
- プロジェクトで DIM ファイルを参照する
- DIM をプロジェクトにインポートする
- インストール プロジェクトで DIM の要素を識別する
- インストール プロジェクトで使用する DIM プロジェクト内のパス変数をオーバーライドする

ヒント: *DIM プロジェクトを作成方法については、「開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化」をご覧ください。*

インストール プロジェクトに DIM を統合するための適切な方法を判別する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

機能と同じサイズのデベロッパー インストール マニフェスト (DIM) は、インストール パッケージの別個の部分を構成している製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM を作成したあと、2 つある方法のいずれかで基本の MSI プロジェクトに追加することができます:

- **参照** — DIM プロジェクトの参照を基本の MSI プロジェクトに追加することができます。この方法を使用すると、DIM の要素は、ビルド時に基本の MSI プロジェクトにマージされます。基本の MSI インストールがビルドされるたびに、DIM プロジェクトの最新バージョンが参照され、生成されたインストールに含められます。

  基本の MSI プロジェクトで DIM リファレンスを使用すると、複数のソフトウェア開発者が、インストール開発作業に同時に携わることができます。各ソフトウェア開発者は、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。

  DIM プロジェクトのある箇所を変更する場合、まず InstallShield で DIM プロジェクトを開き、必要な変更を行った後、その DIM を参照する任意の基本の MSI プロジェクトを再ビルドします。

  この方法は、最も頻繁に使われる方法です。

  詳細については、「プロジェクトで DIM ファイルを参照する」を参照してください。

- **インポート** – DIM プロジェクトの参照を基本の MSI プロジェクトにインポートすることができます。この方法は、デザイン時に、DIM データを基本の MSI プロジェクトにマージするときに行う目的で一回切り行われ、一度行われると元に戻すことはできません。

  DIM データにさらに変更を行う必要がある場合は、基本の MSI プロジェクトで行います。元の DIM プロジェクトを変更しても、基本の MSI プロジェクト内の対応するデータは変更されます。

  DIM プロジェクトをインポートして、製品またはそのインストール パッケージの問題をトラブルシュートすることを考えているとします。DIM プロジェクトをインポートする時、インポートする前に、基本の MSI プロジェクト (.ism) のバックアップを作成するオプションがあります。このオプションを使うことにより、DIM プロジェクトがインポートされていない元の基本の MSI プロジェクトに復元することができます。

  詳細については、「DIM をプロジェクトにインポートする」を参照してください。

DIM プロジェクトによって、開発者とリリース エンジニアは、機能別に分かれたインストールの異なる部分を効率的に再利用することができます。

プロジェクトで DIM ファイルを参照する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

基本の MSI プロジェクトに、DIM プロジェクトのリファレンスを追加すると、DIM の要素が、ビルド時に、基本の MSI プロジェクトにマージされます。基本の MSI インストールがビルドされるたびに、DIM プロジェクトの最新バージョンが参照され、生成されたインストールに含められます。

タスク

DIM のリファレンスを、基本の MSI プロジェクト内の特定機能に追加するには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM リファレンスの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. 参照する DIM プロジェクト ファイル (.dim) を参照し、見つかったファイルを選択します。
4. **[開く]** ボタンをクリックします。

適切な機能に、DIM プロジェクト内の各コンポーネントに対するコンポーネント ノードが追加されます。このセクションには、次のトピックが含まれています:

- DIM リファレンスを機能に関連付ける
- ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する
- DIM リファレンスのビルドの手順を表示する
- DIM リファレンスのインストール先のオーバーライド
- DIM リファレンスからのカスタム アクションとダイアログをスケジュールする
- インストール プロジェクト内から参照された DIM プロジェクトを開く

ヒント: *[DIM リファレンス] ビューを使って、基本の MSI プロジェクトにDIM のリファレンスを追加することもできます。このビューでは、[DIM] エクスプローラーを右クリックして、[DIM リファレンスの追加] をクリックします。この方法を通してリファレンスを追加すると、DIM は基本の MSI プロジェクト内にあるどの機能にも関連付けられません。関連付けの設定方法については、「DIM リファレンスを機能に関連付ける」をご覧ください。参照された DIMはそれぞれ、基本の MSI プロジェクト内の機能に関連付けられる必要があります。関連付けられなかった場合、ビルド時にそれを生成する基本の MSI インストールに含められません。*

DIM リファレンスを機能に関連付ける

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM リファレンスを、基本の MSI プロジェクトに含める場合、その DIM リファレンスは最低 1 つの機能に関連付けられている必要があります。関連付けられていなかった場合、ビルド時にそれを生成する基本の MSI インストールに含められません。

DIM リファレンスは、基本の MSI プロジェクトの [DIM リファレンス] ビルドまたは [セットアップのデザイン] ビューで、1 つまたは複数の機能に関連付けることができます。

タスク

基本の MSI プロジェクトの [DIM リファレンス] ビューを使って、DIM リファレンスを機能に関連付けるには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
2. **[DIM]** エクスプローラーで、構成する DIM を選択します。
3. **[機能]** タブをクリックします。
4. 選択した DIM を含める機能のチェック ボックスをすべて選択します。選択した DIM を含めない機能のチェック ボックスをすべてクリアします。

タスク

基本の MSI プロジェクトの [セットアップのデザイン] ビューを使って、DIM リファレンスを機能に関連付けるには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM リファレンスを関連付ける]** をクリックします。**[DIM リファレンスを関連付ける]** ダイアログ ボックスが開き、基本の MSI プロジェクト内の機能に関連付けられていない DIM リファレンスがすべて表示されます。
3. 選択した機能に関連付ける DIM ファイルをそれぞれ選択して、**[OK]** をクリックします。

ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM リファレンスを含む基本の MSI リリースをビルドした時、DIM データが生成された基本の MSI リリースに組み込まれます。データの競合が存在する場合、DIM プロジェクトの値か、または基本の MSI プロジェクトの値か、優先されるべきデータが判別される必要があります。

InstallShield では、DIM プロジェクト内の設定と、その DIM が参照されている基本の MSI プロジェクト内の設定間の競合を解決する方法を指定できます。次は、発生する可能性があるいくつかの競合の例です:

- 同一の名前のプロパティが両方のプロジェクトで使用されていて、異なる値が設定されている場合。
- 同一の文字列識別子がある文字列が両方のプロジェクトでエントリされていて、特定の言語に対して異なる値が設定されている場合。
- 同一の名前が指定されているが、異なる値が設定されているパス変数が両方のプロジェクトで使用されている場合。
- 同一名のコンポーネントが両方のプロジェクトで使用されていて、1 つまたは複数の設定で異なる値が設定されている場合。

ほとんど場合、プロパティ、文字列エントリ、パス変数、コンポーネントなどのアイテムを DIM プロジェクトで作成して、基本の MSI プロジェクトでは作成しなかった場合、DIM プロジェクト固有の GUID がアイテム名の一部として自動的に使用されるため、競合は発生しません。基本の MSI プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは MYPROPERTY という名前になります。DIM プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは内部的に MYPROPERTY.DIM_GUID という名前で使用されます。*DIM_GUID* は、DIM プロジェクトの [一般情報] ビューにある "DIM GUID" 設定で定義されている識別子を表しています。例えば、DIM でプロパティ名は、**MYPROPERTY.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC** のようになります。この特定の例では、ダイレクト エディターを使って、DIM プロジェクトの Property テーブルで、プロパティの内部名を確認することができます。

タスク

基本の MSI プロジェクトと使われている DIM リファレンスの 1 つ間で発生した競合を自動解決するように設定するには、以下の手順に従います:

1. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
2. **[DIM]** エクスプローラーで、構成する DIM を選択します。
3. **[ビルド オプション]** タブをクリックします。
4. **"競合の解決"** 設定で、適切な値を選択します。選択可能なオプションは以下のとおりです:
   - **ベース プロジェクト値を使用する** – ビルド時に DIM データを基本の MSI インストールに結合した時に競合が発生した場合、DIM プロジェクト内の値の代わりに、基本の MSI プロジェクト内の値が使用されます。
   - **DIM プロジェクト値を使用する** – ビルド時に DIM データを基本の MSI インストールに結合した時に競合が発生した場合、基本の MSI プロジェクト内の値の代わりに、DIM プロジェクト内の値が使用されます。

可能性は低いけれども競合が発生する可能性があるケースとして、DIM を基本の MSI プロジェクトにインポートして、アイテムの 1 つを DIM プロジェクトで変更し、その DIM のリファレンスを基本の MSI プロジェクトに追加した場合が考えられます。このシナリオでは、競合を避けるために "競合の解決" 設定がどのように構成されているかによって、DIM の設定の値または基本の MSI の設定の値が上書きされます。

DIM リファレンスのビルドの手順を表示する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM リファレンスを基本の MSI プロジェクトに追加する場合、DIM の作成者によって、ビルドの手順が提供されているかどうかを確認することをお勧めします。この手順で提示されているガイダンス、ヒント、コメントなどによって、追加の手続きが必要かどうか判別できることがあります。

タスク ***基本の MSI プロジェクトで DIM リファレンスのビルドの手順を表示するには、以下の手順に従います:***

1. DIM リファレンスを含む基本の MSI プロジェクトを開きます。
2. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
3. **[DIM]** エクスプローラーで、構成する DIM を選択します。
4. **[手順]** タブをクリックします。

[手順] タブで、作成または編集時に DIM プロジェクトの作成者が入力したコメントが表示されます。

DIM リファレンスのインストール先のオーバーライド

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトを作成した場合、DIM プロジェクト内のファイル用に、ターゲット システム上で、デフォルトのインストール先を指定することができます。この場所は通常、INSTALLDIR プロパティの値か、このフォルダーのサブフォルダーです。DIM リファレンスを基本の MSI プロジェクトに追加した時、DIM プロジェクトの作成者が構成したインストール先を使うか、または、既存の場所を上書きして、別の場所を指定するかを選択できます。

タスク ***基本の MSI プロジェクトで、DIM リファレンスのファイル用に指定されたデフォルトのインストール先を表示して、それを必要に応じて上書きするには、以下の手順に従います:***

1. DIM リファレンスを含む基本の MSI プロジェクトを開きます。
2. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
3. **[DIM]** エクスプローラーで、構成する DIM を選択します。
4. **[ビルド オプション]** タブをクリックします。**"インストール先"** 設定で、DIM プロジェクトの作成者が構成した値が表示されます。
5. デフォルトの値を上書きするには、適切な値を指定します。

   パスをハードコード化する代わりに、パスの一部としてディレクトリ プロパティを入力することができます。ディレクトリ プロパティを選択するには、この設定で省略記号ボタン (...) をクリックします。ここで、適切なディレクトリをリストから選択するか、定義済みディレクトリ内に新しいディレクトリを作成できます。円記号を使用して、サブディレクトリの下位レベルを **[ProgramFilesFolder]MyApp¥Bin** のように区切ります。

機能およびコンポーネント レベルで、インストール先を上書きすることもできます。詳細については、「デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定」を参照してください。

DIM リファレンスからのカスタム アクションとダイアログをスケジュールする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトを作成した時、カスタム アクションとダイアログを作成することができます。DIM リファレンスを基本の MSI プロジェクトに追加した時、基本の MSI インストールの実行中、どのタイミングで、その DIM 内のカスタム アクションを実行するかをスケジュールできます。また、基本の MSI インストールの実行中、DIM のダイアログをいつ実行するかも指定できます。

タスク ***DIM リファレンスからのカスタム アクションとダイアログを基本の MSI プロジェクトに挿入するには、以下の手順に従います:***

1. DIM リファレンスを含む基本の MSI プロジェクトを開きます。
2. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
3. **[DIM]** エクスプローラーで、構成する DIM を選択します。
4. **[シーケンス]** タブをクリックします。
5. **[シーケンス]** エクスプローラーで、新しいアクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。**[アクションの挿入]** ダイアログ ボックスが開き、シーケンスに追加することができるアクションおよびダイアログが一覧表示されます。
6. ダイアログ ボックスの上にあるリストから、挿入するアクションの種類を選択します。
7. アクションの一覧があるボックスで、挿入するアクションの種類を選択します。
8. **[OK]** をクリックします。

インストール プロジェクト内から参照された DIM プロジェクトを開く

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

1 つまたは複数の DIM リファレンスを含む基本の MSI プロジェクトを変更中、DIM プロジェクトを編集、またはその詳しい情報を表示する場合、リンクをクリックすることで、そのプロジェクトを開くことができます。DIM プロジェクトは、基本の MSI プロジェクトを変更しているマシンからアクセスできる必要があるので注意してください。

タスク ***基本の MSI プロジェクトから参照された DIM プロジェクトを開くには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[DIM リファレンス]** をクリックします。
2. **[DIM]** エクスプローラーで、開く DIM を選択します。

3. **[全般]** タブで、**[DIM プロジェクトを起動]** リンクをクリックします。

InstallShield の新しいインスタンスが起動し、DIM プロジェクトが InstallShield 内で開きます。

DIM をプロジェクトにインポートする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトを基本の MSI プロジェクトにインポートすると、DIM データが基本の MSI プロジェクトに結合されます。インポートは 1 回きりの手続きで、復元不可能な操作です。

DIM プロジェクトをインポートして、製品またはそのインストール パッケージの問題をトラブルシュートすることを考えているとします。DIM プロジェクトをインポートする時、インポートする前に、基本の MSI プロジェクト (.ism) のバックアップを作成するオプションがあります。このオプションを使うことにより、DIM プロジェクトがインポートされていない元の基本の MSI プロジェクトに復元することができます。

タスク ***基本の MSI プロジェクトで、DIM を特定機能にインポートするには、以下の手順に従います:***

1. **[編成]** のビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM のインポート ウィザード]** をクリックします。**[DIM のインポート ウィザード]** ダイアログ ボックスが開きます。
3. ウィザードの各パネルを完成します。

DIM を基本の MSI プロジェクトにインポート中、デザイン時の競合を解決する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトをインストール プロジェクトにインポートすると、DIM データがインストール プロジェクトに結合されます。データの競合が存在する場合、DIM プロジェクトの値か、または基本の MSI プロジェクトの値か、優先されるべきデータが判別される必要があります。

InstallShield では、インポート中、2 つのプロジェクトの設定間の競合を解決することができます。次は、発生する可能性があるいくつかの競合の例です:

- 同一の名前のプロパティが両方のプロジェクトで使用されていて、異なる値が設定されている場合。
- 同一の文字列識別子がある文字列が両方のプロジェクトでエントリされていて、特定の言語に対して異なる値が設定されている場合。
- 同一の名前が指定されているが、異なる値が設定されているパス変数が両方のプロジェクトで使用されている場合。
- 同一名のコンポーネントが両方のプロジェクトで使用されていて、1 つまたは複数の設定で異なる値が設定されている場合。

DIM プロジェクトのインポート中に競合が発生した時、[競合の解決] ダイアログ ボックスが開きます。このダイアログ ボックスで、任意の対応方法を指定できます。詳細については、「[競合の解決] ダイアログ ボックス」を参照してください。

ほとんど場合、プロパティ、文字列エントリ、パス変数、コンポーネントなどのアイテムを DIM プロジェクトで作成して、基本の MSI プロジェクトでは作成しなかった場合、DIM プロジェクト固有の GUID がアイテム名の一部として自動的に使用されるため、競合は発生しません。基本の MSI プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは MYPROPERTY という名前になります。DIM プロジェクトで MYPROPERTY という名前のプロパティを作成した場合、このプロパティは内部的に MYPROPERTY.DIM_GUID という名前で使用されます。*DIM_GUID* は、DIM プロジェクトの [一般情報] ビューにある "DIM GUID" 設定で定義されている識別子を表しています。例えば、DIM でプロパティ名は、**MYPROPERTY.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC** のようになります。この特定の例では、ダイレクト エディターを使って、DIM プロジェクトの Property テーブルで、プロパティの内部名を確認することができます。

インストール プロジェクトで DIM の要素を識別する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield で DIM プロジェクトを作成すると、プロジェクトに GUID が割り当てられます。この GUID によって、DIM プロジェクトを個別に識別できます。InstallShield では、DIM プロジェクトおよびこの DIM を含むすべての基本の MSI プロジェクトの様々箇所で、DIM に属するデータを識別するために、この GUID が使われます。

たとえば、**MyFile.exe** という名前のファイルを **8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC** という GUID を持つ DIM プロジェクトに含めると、このファイルの値が次のように、DIM プロジェクトの File テーブルに書き込まれます:

MyFile.exe.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC

異なる DIM プロジェクトに、同一の MyFile.exe ソース ファイルを含めた場合、ファイルの File テーブル エントリの終わりに、別の GUID が付加されます。DIM プロジェクトを基本の MSI プロジェクトにインポートすると、このファイルの GUID エントリも基本の MSI プロジェクトの File テーブルにインポートされます。**MyFile.exe** ディレクトリを基本の MSI プロジェクトに追加した場合、GUID は、ファイル名の終わりに付加されません。**MyFile.exe** を含む DIM を基本の MSI プロジェクトにインポートした場合、

ファイル、コンポーネント、パス変数、および他のインストール データに対して、DIM の GUID が使用されます。これにより、基本の MSI プロジェクトのどの部分が、基本の MSI プロジェクトから直接来ているかが簡単に判別できます。また、基本の MSI プロジェクトで参照されている、または、インポートされた様々な DIM プロジェクトに元々属している箇所も簡単に判別できます。

GUID の使用は、競合の予防にもなります。たとえば、複数の DIM プロジェクトで特定のパス変数に対して異なる値を使用していて、それらの DIM を、その同じパス変数に対してさらに別の異なる値を使用している基本の MSI プロジェクトにインポートした時も、競合は発生しません。基本の MSI インストールがビルドされた時、それぞれのパス変数のインスタンスに対して適切な値が使用されます。

インストール プロジェクトで使用する DIM プロジェクト内のパス変数をオーバーライドする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM プロジェクトの作成者によって、プロジェクトのパス変数がどう定義されたかにより、場合によって、その DIM を含む基本の MSI のリリースをビルドする時、それらの値をオーバーライドする必要があります。この手順を省くと、インストールをビルドした時、ファイルが不足しているためにビルド エラーが発生したり、間違ったファイルがビルドに取り込まれたりすることがあります。

InstallShield では、DIM プロジェクトのパス変数をオーバーライドする方法がいくつか提供されています:

- 基本の MSI リリースをビルドする時、そのプロジェクトにインポートされた DIM のパス変数をオーバーライドする場合、[リリース] ビルドの [ビルド] タブにある "パス変数のオーバーライド" 設定を使うことができます。
- ISCmdBld.exe を使って、コマンドラインからビルドする場合は、新しい -I パラメーターを使って、パス変数の名前と新しい値を指定します。1 つまたは複数の DIM を含む基本の MSI プロジェクトのインストールをビルドする場合、この方法を使用します。
- MSBuild または Team Foundation Server (TFS) を使ってビルドする場合、InstallShield タスクで PathVariables パラメーターを使います。このパラメーターは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として露出されます。
- DIM プロジェクトを開き、必要に応じて、パス変数の値を更新します。DIM プロジェクトを開いている基本の MSI プロジェクト内から開く方法については、「インストール プロジェクト内から参照された DIM プロジェクトを開く」を参照してください。

ビルド時に製品のソース ファイルを取得して、リリースをビルドする時、設定されたパス変数のオーバーライドが使用されます (これは Standalone Build にも適用します)。

DIM プロジェクトのパス変数に関する情報およびヒントに関しては、「DIM でパス変数を使用する」を参照してください。

ターゲット システムの構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

また記述の通り、このトピックの一部は特定のプロジェクトの種類には適用しません。

すべてのインストールは、何らかの方法でターゲット システムを変更します。簡単なインストールでは、ファイルをコピーするだけのものもあります。より複雑なインストールでは、レジストリの変更、.ini ファイルの編集、ショートカットの作成、ODBC リソースの構成、環境変数の使用、XML ファイルの変更、およびテキスト ファイルの変更、タスクのスケジュール、および Windows サービスの制御が行われます。ターゲット システムの構成方法に関する詳しい情報は、ドキュメントのこのセクションを参照してください。

このセクションには、次のトピックが含まれています:

- ショートカットおよびプログラム フォルダーの作成
- レジストリの編集
- .ini ファイル データの変更
- ODBC リソースの構成
- 環境変数を使用する
- XML ファイルの変更
- テキスト ファイルの変更
- スケジュール タスク
- Windows サービスのインストール、制御、および構成
- 実行時に既定のユーザー アカウントを作成する
- ユーザーごとのインストールとマシンごとのインストールの違い
- 管理者権限なしに InstallScript インストールを実行する

ショートカットおよびプログラム フォルダーの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ショートカット] ビューの [ショートカット] エクスプローラーを使って、Windows デスクトップと [スタート] メニュー上に、ショートカットおよびプログラム フォルダーを作成できます。ファイルと同様に、ショートカットはコンポーネントに関連しています。インストール プロジェクトでは、コンポーネントが所属する機能がインストールに選択された場合にのみ、ターゲット システムに作成されます。ショートカットおよびプログラム フォルダーについての詳細は、次を参照してください:

- ショートカットの種類
- ショートカットの作成
- ショートカットの設定を構成する
- ショートカットのアイコンを指定する
- インターネット ショートカットの作成
- フォルダーにショートカットを作成する
- ショートカットにアクセスできるキーボード ショートカットを指定する
- ショートカット名の変更
- ショートカットのシェル プロパティを設定する
- 基本の MSI プロジェクトのアンインストール ショートカットを作成する
- InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットを作成する
- スタート画面上のデスクトップ アプリのタイルの外観を構成する

ショートカットの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*

- *MSM データベース*
- *トランスフォーム*

InstallShield では、いくつかのタイプのショートカットが提供されています。

テーブル 1 - ショートカットの種類

ショートカットの種類	プロジェクトの種類	説明
新しいショートカット	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	プロジェクトに存在するファイルへの新しいショートカットを作成します。 ***メモ-** このオプションは、コンポーネントのファイルが指定されていない場合は無効です。*
新しいアドバタイズ ショートカット	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	アドバタイズ ショートカットの作成コンポーネントのファイルはエンドユーザーがショートカットを起動するまでターゲット システムにインストールされません。
新しいインターネット ショートカット	InstallScript	"インターネット ショートカット" 設定が [はい] に設定されているインターネット ショートカットを作成します。"ターゲット" 設定は、デフォルト値の **http://www.YourCompanyName.com** になっていますが、この値を希望の URL に変更します。
既存ファイルへの新しいショートカット	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ターゲット システムに既に存在するファイルへのショートカットを作成します。
新しいフォルダー	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	プログラム フォルダーを作成します。たとえば、会社名の下にショートカットを表示したい場合は、プログラム フォルダーを作成することができます。

ショートカットの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ショートカットを作成する前に、最低でも 1 つの機能とコンポーネントを作成する必要があります。最初のショートカットを作成するときに、コンポーネントが作成されていない場合は、InstallShield によってコンポーネントが作成されます。ショートカットが属するコンポーネントは、ショートカットの "コンポーネント" 設定を構成することによって変更できます。

タスク

ショートカットを作成するには、以下の手順に従ってください。

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、インストール先ディレクトリの 1 つを右クリックして、適切なコマンドをクリックします。使用可能なコマンドの一覧は、「ショートカットの種類」をご覧ください。

   新しいショートカットが、**新しいショートカットN** というデフォルト名で追加されます (ここで *N* は連続番号です)。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. ショートカットの設定を構成します。

ショートカットとフォルダーについて構成可能な各設定についての詳細は、以下を参照してください:

- ショートカットの設定
- フォルダーの設定

メモ: *たとえば、会社名の下にショートカットを表示したい場合は、プログラム フォルダーを作成することができます。ショートカットのフォルダーを作成した後、そのフォルダーを右クリックし、[新しいショートカット] を選択して、ショートカットを作成できます。*

動的にリンクされたファイルへのショートカットを作成することはできません。詳細については、「ダイナミック ファイル リンクの制限事項」を参照してください。

ヒント: *[コンポーネント] ビューまたは、インストール プロジェクトの場合は、[セットアップのデザイン] ビューを使ってショートカットを作成することもできます。[コンポーネント] ビューまたは [セットアップのデザイン] ビューを使用する場合、エクスプローラーで [ショートカット] 項目をクリックします。[ショートカット] エクスプローラーが新しいペインで開きます。*

ショートカットの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ショートカットを作成後、ショートカットの設定を構成します。ショートカットの設定を利用して、ショートカットをファイルにリンクさせたり、ショートカットの説明を提供したり、または引数を渡したりすることができます。

タスク

ショートカットの設定を構成するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ショートカットをクリックします。ショートカットの設定が、右のペインに表示されます。
3. 必要に応じて設定を指定します。

ショートカットとフォルダーについて構成可能な各設定についての詳細は、以下を参照してください:

- ショートカットの設定
- フォルダーの設定

ショートカットのアイコンを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、実行時にターゲット システムで作成されるショートカットに使用されるアイコンを指定することができます。

タスク ***ショートカットのアイコンを指定するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、アイコンを指定するショートカットをクリックします。ショートカットの設定が、右のペインに表示されます。
3. **"アイコン ファイル"** 設定で、作成するショートカットのアイコンが含まれているファイルを指定します。アイコン リソースを含む .ico ファイル、または実行可能ファイル (.dll または .exe) を指定する必要があります。

   基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: アイコンを含むファイルの完全修飾パスを入力するか、または省略記号 (...) ボタンをクリックして参照します。

   InstallScript プロジェクトの場合: アイコンを含むターゲット システム上にあるファイルへの完全修飾パスを入力します。
4. 指定したアイコン ファイルに 1 つ以上のアイコン リソースがある場合、**"アイコン インデックス"** 設定にインデックスを入力します。

   負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば、0 はファイル内の最初のアイコン、1 は 2 番目のアイコン、2 は 3 番目のアイコンを参照します。

次のいずれかの条件が True でない限り、InstallShield は [ショートカット] エクスプローラーのショートカットに表示されたアイコンを指定したアイコンにアイコンに変更します。

- プロジェクトの種類が InstallScript である。
- ショートカットが、ターゲット システム上に既に存在するファイルである。

上記の両条件の場合、アイコン ファイルは実行時まで認識されません。したがって、[ショートカット] エクスプローラーでは、ターゲット システム上で実行時に使用されるアイコンの代わりに、各ショートカットに次のアイコンが表示されます。

"アイコン ファイル" 設定で選択されたファイルがアイコンを含まない場合、このアイコンが [ショートカット] エクスプローラーのショートカットに使用されます。

プロジェクト: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: Windows Installer ではコンポーネントがアドバタイズされるときに別のアイコンが必要なため、InstallShield は指定した実行ファイルからアイコンを取り出します。*

ヒント: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: [ショートカット] ビューのアイコンを右クリックしてから、[ショートカットの変更] をクリックして、異なるショートカット アイコンを指定することもできます。この方法で指定された値でアイコンの設定内の値が更新されます。*

インターネット ショートカットの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

インターネット ショートカットとは、拡張子 .url を持つテキスト ファイルのことです。

タスク ***Web サイトを開くインターネット ショートカットを作成するには、次の手順を実行します。***

1. 次の内容を含む **MySite.url** という名前のテキスト ファイルを作成します:

   [InternetShortcut]

   URL=http://www.MySite.com

2. このファイルをコンポーネントに追加します。

単純に右クリックして [追加] をクリックした場合、ショートカットそのものではなくショートカットがポイントしているものを追加することになります。

ユーザーがこのショートカットを起動すると、指定された Web サイト (**www.MySite.com**) がユーザーのデフォルトのブラウザーで開きます。

フォルダーにショートカットを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、フォルダーへのショートカットを作成できます。フォルダーへのショートカット作成方法は、使用するプロジェクトの種類によって異なります。

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト

ショートカットの "ターゲット" 設定には、角括弧の中にディレクトリ識別子を含めることができます。たとえば、INSTALLDIR にショートカットを作成するには、次の設定でショートカットを作成します。

- 表示名: INSTALLDIR へのショートカット
- アドバタイズ: なし
- ターゲット: [INSTALLDIR]

InstallScript および InstallScript オブジェクト プロジェクト

AddFolderIcon 関数は 3 番目の (szCommandLine) パラメーターでフォルダーへのパスを渡すと、そのフォルダーへのショートカットを作成します。

たとえば、このコードはエンド ユーザーの [プログラム] フォルダーに Common Files フォルダーへのショートカットを作成します。

```
ProgDefGroupType(COMMON)

szFolder = TARGETDIR;
LongPathToQuote(szFolder, TRUE);

AddFolderIcon(
   ProgramMenuFolder,        // ショートカットの作成場所
   "TARGETDIR へのショートカット",// ショートカット表示名
   szFolder,        // ショートカットによって起動されるもの
   ""// 作業ディレクトリ
   TARGETDIR ^ "Sample.exe", 0,       // アイコンファイル、インデックス
   "",    // ショートカット キー
   NULL);              // 特殊設定
```

InstallScript MSI プロジェクト

前の InstallScript セクションでの作業と同様に基本の処理を行い、サンプルコード内のすべての TARGETDIR を INSTALLDIR へ変更します。

ショートカットにアクセスできるキーボード ショートカットを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

キーボード ショートカット (ホット キーとも呼ばれます) を使って、マウスを使う代わりに CTRL+ALT+A などのキーの組み合わせを押さえることで、素早く処理を行うことができます。キーボード ショートカットを製品のショートカットに割り当てると、エンド ユーザーは適切なホット キーを押してショートカットを起動できます。

注意 - *ターゲット システム上の既存のキーボード ショートカットと競合する可能性があるため、ショートカットのキーボード ショートカットを構成することは避けることをお推めします。*

タスク ***キーボード ショートカットをプロジェクト内のショートカットに割り当てるには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、ホットキーを指定するショートカットを選択します。
3. **"ホット キー"** 設定で、省略記号ボタン (...) をクリックします。**[ホットキー]** ダイアログ ボックスが開きます。
4. このショートカットに使用するキーボード ショートカットを押します。
5. **[OK]** をクリックします。

"ホット キー" 設定に、押されたキーの組み合わせを示す適切な 10 進数の値が表示されます。

たとえば、キーの組み合わせが CTRL+ALT+A の場合、この設定には 1601 と表示されます。この数値は、CTRL の 16 進数の値 (200) と ALT の 16 真数の値 (400)、および論理 Or 演算子を組み合わせて取得されます。次に、この数値 (600) に A キーの 16 進数値 (41) を追加し、最後に 10 進数値に変換します。この例では、10 進数に変換される数値は 641 で、変換後は 1601 となります。

ショートカット名の変更

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

新しいショートカットを作成する、デフォルトの内部名で表示されます。この名前はエンドユーザーに表示されませんが、プロジェクトに関連した名前に変更することができます。

タスク ***ショートカットの名前を変更するには、次の操作を実行します。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、名前を変更するショートカットを右クリックして、**[名前の変更]** をクリックします。
3. 新しい名前を入力します。

ショートカットのシェル プロパティを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、インストール実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。たとえば、InstallShield には次の動作を制御するシェル プロパティを設定するためのビルトイン サポートが搭載されています:

- Windows 8 スタート画面への初回のショートカットのピン留めを抑制する
- エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する
- [スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ

また、Windows シェルがサポートする追加プロパティを設定することもできます。詳細については、「カスタム アクション シェルのプロパティを設定する」を参照してください。

Windows 8 スタート画面への初回のショートカットのピン留めを抑制する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、Windows 8 ターゲット システム上で、スタート画面への初回のショートカットのピン留めを抑制するかどうかを指定できます。

メモ: *Windows 8.1 では、新しくインストールされたアプリケーション/ショートカットの自動的なピン留めが無効化されました。*

ショートカットのピン留めを抑制する場合、インストールは Windows 8 で使用可能になった Windows シェル プロパティを設定します。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。

以前のバージョンの Windows はこのショートカット プロパティを無視します。

タスク

Windows 8 スタート画面へのショートカットのピン止めをデフォルトで抑制するかどうかを指定するには、次の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[Windows 8 スタート画面への初回のピン止めを抑制する]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定の追加行を追加して、必要に応じてそれらを構成します。

Windows Installer 5 で、ショートカット プロパティがサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。

Windows 8 は、アプリケーションのアンインストールによってショートカットが削除された後でも、ショートカットのスタート画面へのピン留めに関する情報を保持します。そのため、ショートカットがインストール済みの場合、ターゲット システム上でこの設定は効果を持ちません。この機能をテストする際、ショートカットとそのターゲットが既にインストールされていない、クリーン マシン上でテストするようにして下さい。

Windows Installer ベースのプロジェクト (基本の MSI、DIM,InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム) で、Windows Installer がシェル プロパティを構成するショートカットをインストールするとき、次のようなメッセージと共に表示されるランタイム警告またはエラーが、一部のエンド ユーザーから報告されました。

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript 言語のサポート

次の InstallScript 関数を使って、ショートカットをスタート画面にピン留めしないように防ぐことができます:

- CreateShortcut
- SetShortcutProperty

エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、エンド ユーザーが製品をインストールした後に、ショートカットをタスクバーおよび [スタート] メニューにピン留めするためのコンテキスト メニュー コマンドを表示するかどうかを指定できます。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。このピン留めを行わないようにショートカットを構成した場合、[スタート] メニューの最もよく使われている製品のリストに、ショートカットのターゲットを含められなくなります。

このピン留め機能を行わないようにショートカットを構成した場合、インストールは Windows シェル プロパティを設定します。デフォルトで、新しいショートカットにプロパティは設定されないため、エンド ユーザーはショートカットをタスクバー、および [スタート] メニューにピン留めすることができます。

特定の文字列を含むショートカットは、タスクバーまたは [スタート] メニューにピン留めすることができません。また、それらを最もよく使う製品リストに表示することもできません。例:

- Documentation
- Help

- Install
- Remove
- Setup
- Support

使用するプロジェクトの種類によって、コンテキスト メニュー コマンドを隠す方法は異なります。

Windows Installer ベースのプロジェクトの手順

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、"シェル プロパティ" 設定を使ってコンテキスト メニュー コマンドを隠します。

タスク

ショートカットをタスクバーまたは [スタート] メニューにピン留めするためのコンテキスト メニューコマンドを隠すには、次の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[ピン留めをしない]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定の追加行を追加して、必要に応じてそれらを構成します。

コンテキスト メニュー コマンドの表示を許可するには、次のいずれかを含む Property サブ設定を持つ "キー名" 設定を探してから、"キー名" 設定で [**このショートカット シェル プロパティを削除する**] ボタンをクリックします。

- System.AppUserModel.PreventPinning
- {9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3}, 9

Windows Installer 5 で、このショートカット プロパティがサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。

Windows Installer ベースのインストールでは、一部のエンド ユーザーから、Windows Installer がシェル プロパティを構成するショートカットをインストールしているときに発生する、以下のようなメッセージを含むランタイム警告またはエラーが報告されています:

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript プロジェクトの手順

InstallScript プロジェクトの場合、"ピン留めを禁止する" 設定を使います。

タスク

ショートカットをタスクバーまたは [スタート] メニューにピン留めするためのコンテキスト メニューコマンドを隠すには、次の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"ピン留めを禁止する"** 設定で **[はい]** を選択します。

Windows Installer 7 から、この設定がサポートされています。以前のバージョンの Windows はこの設定を無視します。

InstallScript 言語のサポート

次の InstallScript 関数を使って、ショートカットをタスクバーまたは [スタート] メニューにピン留めするためのコンテキスト メニューコマンドを隠すことができます:

- CreateShortcut
- SetShortcutProperty

[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

オプションで InstallShield では、エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐことができます。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。インストールの一部であるツールまたは従属的な製品のショートカットのプロパティを設定したい場合があります。

ショートカットの強調表示機能の抑制を指定した場合、インストールは Windows シェル プロパティを設定します。デフォルトで、新しいショートカットのプロパティは設定されていません。

使用するプロジェクトの種類によって、強調表示をしないように防ぐ方法が異なります。

Windows Installer ベースのプロジェクトの手順

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、"シェル プロパティ" 設定を使って強調表示をしないように防ぐ動作を設定します。

タスク

ショートカットの [スタート] メニューのエントリを、新しくインストールされたアイテムとして強調表示しない場合は、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[新規として強調表示しない]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定の追加行を追加して、必要に応じてそれらを構成します。

[スタート] メニュー エントリの強調表示を有効化するには、次のいずれかを含む "プロパティ" サブ設定を持つ "キー名" 設定を探してから、"キー名" 設定で [**このショートカット シェル プロパティを削除する**] ボタンをクリックします。

- System.AppUserModel.ExcludeFromShowInNewInstall
- {8F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3}, 9

Windows Installer 5 で、このショートカット プロパティがサポートされています。以前のバージョンの Windows Installer はこのプロパティを無視します。

Windows Installer ベースのインストールでは、一部のエンド ユーザーから、Windows Installer がシェル プロパティを構成するショートカットをインストールしているときに発生する、以下のようなメッセージを含むランタイム警告またはエラーが報告されています:

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript プロジェクトの手順

InstallScript プロジェクトの場合、"新規として強調表示しない" 設定を使います。

タスク

ショートカットの [スタート] メニューのエントリを、新しくインストールされたアイテムとして強調表示しない場合は、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"新規として強調表示しない"** 設定で **[はい]** を選択します。

Windows Installer 7 から、この設定がサポートされています。以前のバージョンの Windows はこの設定を無視します。

InstallScript 言語のサポート

次の InstallScript 関数を使って、強調表示しないように防ぐことができます:

- CreateShortcut
- SetShortcutProperty

カスタム アクション シェルのプロパティを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。シェルが設定できるプロパティは、Windows SDK に含まれている propkey.h で定義されています。

使用するプロジェクトの種類によって、強調表示をしないように防ぐ方法が異なります。

Windows Installer ベースのプロジェクトの手順

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、"シェル プロパティ" 設定を使って 1 つ以上のシェル プロパティを設定します。

タスク

プロジェクトに含まれるショートカットの "シェル" プロパティを設定するには、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラーで、構成するショートカットを選択します。ショートカットの設定が、右のペインに表示されます。
3. **"シェル プロパティ"** 設定で、**[新しいシェル ショートカット プロパティ の追加]** ボタンをクリックしてから、**[カスタム プロパティ]** を選択します。InstallShield は、新しい **"キー名"** 設定と、その下に関連する設定とプレースホルダー テキストを含む追加行を追加します。
4. 必要に応じて、各設定を構成します。

必要に応じて、ショートカットに複数のプロパティを追加できます。

Windows Installer 5 では、シェル ショートカット プロパティがサポートされています。以前のバージョンの Windows Installer は、これらのプロパティを無視します。

ショートカットのプロパティの構成に関する詳細は Windows Installer ヘルプ ライブラリの「MsiShortcutProperty Table」を参照してください。

Windows Installer ベースのインストールでは、一部のエンド ユーザーから、Windows Installer がシェル プロパティを構成するショートカットをインストールしているときに発生する、以下のようなメッセージを含むランタイム警告またはエラーが報告されています:

ショートカット '[2].lnk' のプロパティ [1] を設定できませんでした。

このメッセージで、[1] は Windows Installer が設定しようとしているシェル プロパティの名前で、[2].lnk はショートカット ファイルの名前です。その場合、.lnk ファイルが別のプロセスによってロックされているかのように見られます。

InstallScript 言語のサポート

次の InstallScript 関数を使って、InstallScript プロジェクトおよびその他のプロジェクト タイプでシェル プロパティを構成できます:

- CreateShortcut
- SetShortcutProperty

基本の MSI プロジェクトのアンインストール ショートカットを作成する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Windows ロゴ: *現在の Windows ロゴ ガイドラインによると、ベスト プラクティスは [スタート] メニューでアプリケーションを削除するショートカットを配置しないこととなっています。アンインストール ショートカットは、アプリケーションのアンインストーラが [プログラムの追加と削除] にあるので不要です。*

エンドユーザーが簡単にシステムから製品をアンインストールできるように、アンインストールのショートカットを作成することができます。このショートカットは、起動すると自動的にアンインストール プロセスを開始します。

タスク ***アンインストール ショートカットを作成するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[ショートカット]** をクリックします。
2. **[ショートカット]** エクスプローラで、ショートカットのインストール先を右クリックしてから、**[既存ファイルへの新しいショートカット]** をクリックします。新しいショートカットが、**新しいショートカットN** というデフォルト名で追加されます (ここで *N* は連続番号です)。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。
4. [**引数**] フィールドに、**/x [ProductCode]** とタイプします。2 つの引数は、間にスペースを入れて区切ります。

Msiexec.exe は、Windows Installer サービスのコマンドライン エンジンです。/x 引数は、Windows Installer サービスに、製品コードによって参照された製品をアンインストールするよう指示します。

5. **"アドバタイズ"** フィールドを **[いいえ]** に設定します。
6. **"ターゲット"** フィールドで、リストから **[SystemFolder]** を選択し、この場所に **msiexec.exe** を付加します。
7. **"コンポーネント"** フィールドで、このショートカットを関連付けるコンポーネントを選択します。ショートカットが常にインストールされるようにするには、アプリケーションのメイン実行可能ファイルが含まれるコンポーネントにショートカットを関連付けます。

InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *アンインストール ショートカットの作成は必要ないと思うユーザーの方もいるでしょう。アンインストール ショートカットは、アプリケーションのアンインストーラが [プログラムの追加と削除] にあるので不要です。*

エンドユーザーが簡単にシステムから製品をアンインストールできるように、アンインストールのショートカットを作成することができます。エンドユーザーがショートカットを起動すると、自動的にアンインストール プロセスが開始します。

次の InstallScript コードは、ファイルの転送後、デスクトップに **[Uninstall Application (アプリケーションのアンインストール)]** という名前のショートカットを作成します。このショートカットを起動すると、インストールはメンテナンス モードで実行されます。このコードは、InstallScript プロジェクトと InstallScript MSI プロジェクトで使用できます。

```
function OnMoved()
begin
  if( !REMOVEALLMODE ) then
    AddFolderIcon( FOLDER_DESKTOP, "Uninstall Application", UNINSTALL_STRING + ADDREMOVE_STRING_REMOVEONLY, "", "", 0,
"", REPLACE );
  endif;
end;
```

ヒント: *ADDREMOVE_STRING_REMOVEONLY を含めると、-removeonly がコマンドラインに追加されます。この追加によって、REMOVEONLY が設定され、OnMaintUIBefore イベント ハンドラーがアンインストール オプションのみを表示するようになります。標準のメンテナンス ユーザー インターフェイスを表示する場合、ADDREMOVE_STRING_REMOVEONLY を含めないでください。*

スタート画面上のデスクトップ アプリのタイルの外観を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

Windows 8 からアプリケーション タイルのグリッドをスタート画面に表示できるようになりました。これは、今までのショートカットの一覧に取って代わるもので、ショートカットの代わりにタイルを配置します。InstallShield は、スタート画面上のデスクトップ アプリのタイルの外観をカスタマイズすることができます。次のタイル構成設定が使用できます:

- アプリケーション名を中サイズ (150x150) のタイルに含めるとき、明色または暗色のテキストを切り替える
- タイル背景色を選択
- カスタム タイル イメージ (小: 70x70、中:150x150) の使用オプション
- アプリケーション名を中サイズ タイルに表示または非表示を選択

[タイルの構成] ノードは、メインの **[ショートカット]** ビューおよび各コンポーネントの **[ショートカット]** サブビューに表示されます。すべての該当するタイル構成が一覧表示されます。コンポーネントの場合、**[タイルの構成]** は、そのコンポーネントに含まれる .exe ファイルの構成にのみ 適用します。メイン ビューの場合、フィルターに構成済みの .exe ファイルをターゲットとするコンポーネントが含まれる場合にのみ適用します。

メモ: *ショートカットおよびタイルは関連していますが、これらは異なります。ショートカットを個別に構成して、異なるアイコンを使用する同じ .exe ファイルに対して複数のショートカットを作成することもできますが、1 つの .exe ファイルに対して 1 つのタイル構成のみ作成することが可能です。*

タスク ***スタート画面上のデスクトップ アプリのタイルの外観を構成するには、以下の手順に従います:***

1. **[ショートカット]** ビュー、または**[セットアップのデザイン]** あるいは **[コンポーネント]** ビューの **[ショートカット]** ノードで、**[ショートカット]** をクリックします。
2. **[タイルの構成]** を右クリックしてから、**[タイル構成の追加]** コンテキスト メニュー オプションを選択します。.

   メモ・ *この手順は、.exe ファイルがプロジェクトに追加されていることを前提とします。.Exe ファイルがプロジェクトに含まれていない場合、**[構成の追加]** コンテキスト メニュー オプションは無効になります。*

   ヒント・ *別の方法として、スタート画面タイルを構成する .exe ファイルをターゲットとするショートカットを右クリックしてから、**[タイルの構成]** コンテキスト メニュー オプションを選択します。新しいタイル 構*

*成が追加および選択されます。ショートカットに構成が既に含まれている場合、**[タイルの構成]** オプションは無効化されます。*

3. [タイル ターゲットの参照] ダイアログ ボックスで、タイルの外観を構成するアプリケーションの .exe ファイルを参照して選択します。新しいタイル 構成が追加されます。
4. **[タイルの構成]** で使用可能な任意の設定を使って、スタート画面上のデスクトップ アプリのタイルをカスタマイズします。使用可能な設定についての詳細は、「[タイル構成] の設定」を参照してください。

レジストリの編集

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

Windows レジストリは、アプリケーションとオペレーション システムで使用される構成情報を含むシステム全体のデータベースです。このレジストリは、次を含むすべての情報を格納します。

- 会社名、製品名、バージョン番号等のアプリケーション情報
- アプリケーションの実行を可能にするパス情報
- エンドユーザーが、システム上にある他のアプリケーションを妨げることなく簡単にアプリケーションをアンインストールができるアンインストール情報
- アプリケーションによって作成される文書のためのシステム全体のファイル関連付け
- ライセンス情報
- ウィンドウの位置等のアプリケーション オプションのデフォルトの設定

キー、値名、および値

レジストリは、My Computer エクスプローラーの下に階層的に編成された 1 セットのキーで構成されます。My Computer のすぐ下には、いくつかのルートキーがあります。インストールは、レジストリのどのキーにもキーと値を追加することができます。以下は、通常インストールによって影響を受けるルートキーです。

- HKEY_LOCAL_MACHINE
- HKEY_USERS
- HKEY_CURRENT_USER
- HKEY_CLASSES_ROOT

キーは、レジストリ内の名前が付けられた場所です。キーは、サブキー、値名、値のペア、およびデフォルト（名前が付いていない）値を含むことができます。値名と値のペアは、キーの下の2 つの部分からなるデータ構造です。値名はキーの下のストレージの値を見分けます。また、その値は値名に関連付けられた実際のデータです。値名が値に特定されていない場合、その値はそのキーのデフォルト値になります。各キーは、デフォルト（名前の付いていない）値を 1 つのみ持つことができます。

terms キーとサブキーはお互い関連していることに注意してください。レジストリでは、別のキーの下のキーを、レジストリ階層の別のキーに関連してそれをどう参照するかによって、サブキーまたはキーとして参照することもできます。

InstallShield プロジェクトとレジストリ

InstallShield の [レジストリ] ビューを利用して、エンド ユーザーのレジストリを変更する作業を簡単に行うことができます。Windows のレジストリ エディターとほぼ同じようにこのビューを使用して、キーおよび値を作成します。

すべてのレジストリ データ（InstallScript プロジェクトの <Default> レジストリ セットを除く）をコンポーネントに関連付ける必要があります。インストール プロジェクトでは、コンポーネントの機能がインストールに選択されると、そのコンポーネントのレジストリ データがターゲット システムに設定されます。

***注意・**レジストリは Windows オペレーティング システムの重要な部分なので、レジストリをむやみに変更または削除することは避けてください。不可欠なレジストリ キーが変更されると、システムが機能しなくなることがあります。*

***メモ・**また、インストーラは、[一般情報] ビューに指定した値に基づいて、特定のレジストリ エントリも自動的に作成します。*

***Windows ロゴ・**これらの「情報キー」は、Windows ロゴ プログラムの要件を満たすために必要です。*

また、コンポーネントの詳細設定はすべて、ターゲット システム上でファイルの登録に使用されます。

コンポーネントまたは機能によるレジストリ エントリのフィルタリング

***プロジェクト・**この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

いくつかのプロジェクト タイプ (DIM、マージ モジュール、MSM データベース) では、機能はサポートされていません。これらのプロジェクト タイプでは、コンポーネント別にフィルターすることはできますが、機能別にフィルターすることはできません。

[レジストリ] ビューには、ビュー フィルターが含まれています。このフィルターで、レジストリ データをビューに表示するコンポーネントまたは機能を選択することができます。

ビューフィルターは、プロジェクトの機能、サブ機能、およびコンポーネントの階層をリスト表示します。機能を選択すると、機能に含まれるすべてのコンポーネントのレジストリ データが表示されますが、既存エントリの変更と削除のみ可能です。新しいレジストリキーを追加するにはコンポーネントを選択しなくてはなりません。

機能の階層にサブ機能が含まれる場合、親機能を選択するとその機能のレジストリ エントリのみが表示されます。ここにサブ機能のレジストリ エントリは表示されません。

コンポーネントを参照する

[レジストリ] エクスプローラー内からコンポーネントを参照することができます。ビューフィルターの右側の省略ボタンをクリックして、[コンポーネント] ダイアログ ボックスの参照を起動しますダイアログ ボックスから、[機能] リストにある機能を選択してから、既存コンポーネントを選択するか、新規コンポーネント ボタンをクリックして新しいコンポーネントを作成します。

ビュー フィルターが [すべてのアプリケーション データ] に設定されている場合、インストール先コンピューターの [レジストリ] ビュー ペインでレジストリ ハイブを右クリックして、[コンポーネントを参照] を選択してダイアログ ボックスを表示することができます。

プロジェクトでのすべてのレジストリ エントリの表示

インストールのすべてのレジストリ エントリを表示するには、ビュー フィルターで [すべてのアプリケーション データ] オプションを選択します。レジストリ キーを右クリックした後、コンテキスト メニューから [すべてエクスポート]、または、[選択したブランチのエクスポート] を選択して、レジストリ データをエクスポートできます。

メモ: *[すべてのアプリケーションデータ] でビューをフィルターして、レジストリキーと値を変更、名前の変更、または削除することができます。*

[レジストリ] ビューの [インストール先コンピューターのレジストリ] ビュー ペインでレジストリ キーをクリックすると、InstallShield は、そのキーのすべてのレジストリ データを [レジストリ] ビューの右下のペインに表示します。[コンポーネント] 列には、レジストリ データが関連付けられているコンポーネントが表示されます。同じ値が複数のコンポーネントに存在する場合は、レジストリデータに関連付けられている最後のコンポーネントが表示されます。

キーに値を設定していない場合、ビュー フィルターで [すべてのアプリケーション データ] を選択しても、そのキーのレジストリ データは一切表示されません。

[レジストリ] ビューのリフレッシュ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***[レジストリ] ビューをリフレッシュするには、以下の手順を実行します。***

F12 キーを押します。

レジストリ キーの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***コンポーネントのインストール時に、ターゲット システムで作成されるレジストリ キーを指定するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム、および QuickPatch プロジェクト*—**[ビュー フィルター]** リストで、レジストリ データを関連付けるコンポーネントをクリックします。

   メモ: ***[すべてのアプリケーションデータ]*** *が選択されている場合、****[レジストリ]*** *ビューは読取り専用になります。*

   InstallScript および InstallScript Object プロジェクトの場合 – **[インストール先コンピューターのレジストリ] ビュー** ペインで既存のレジストリ セットを開くか、**[インストール先コンピューター]** フォルダーを右クリックしてレジストリ セットを作成します。レジストリ セットをクリックして **[レジストリ セットのインストール条件]** ペインで適切なコンポーネントを選択し、レジストリ セットを 1 つまたは複数のコンポーネントに関連付けます。

3. **[インストール先コンピューターのレジストリ] ビュー** ペインで、レジストリ キーを右クリックして、**[新規作成]** をポイントして、**[キー]** をクリックします。InstallShield が、**新しいキー –n***n* (ここで n は連続番号です) という名前で新しいキーを追加します。
4. ここでキー名をわかりやすい名前に変更するか、または、後でキーを右クリックして **[名前の変更]** をクリックし、新しい名前を付けます。

InstallShield が、空白のデフォルトの文字列値を持つ新しいキーを追加します。

デフォルトでは、作成したすべてのキーは、自動インストールおよびアンインストールに設定されています。つまり、所属するコンポーネントがインストールされるとキーもインストールされ、そのコンポーネントがアンインストールされるとキーもアンインストールされます。レジストリ キー フラグの詳細については、「レジストリ フラグ」を参照してください。

ヒント: *新しいキーを作成し、キーの名前に* K**Key 1¥Key 2¥Key 3** *などと入力することによって、ネストされた複数のキーを 1 度に作成することができます。InstallShield は、Key 3 が Key 2 のサブキーであるところにネストされたキー構造を作成します。Key 2 は Key 1 のサブ キーになります。*

レジストリ エントリをドラッグアンドドロップしてレジストリ キーを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

インストール プロジェクトにレジストリ エントリを追加する最も手軽な方法は、[レジストリ] ビューの [ソース] ペインからレジストリ エントリのひとつをドラッグし、[インストール先] ペインにドロップする方法です。すべてのキーを **[インストール先コンピューターのレジストリ] ビュー** ペインへドロップした場合、そのキーのすべてのサブキーおよび値が選択したコンポーネントに追加されます。また InstallScript プロジェクトでは、エントリをドロップしたレジストリセットに追加されます。

コンテキスト メニューを使ってキーをドラッグ & ドロップする

コンテキスト メニューを使って、複数のキーや値を一度に移動することができます。レジストリ エントリを右クリックし、それをインストール先にドラッグし、コンテキスト メニューから [オプション] をクリックします。

テーブル 2 ・ レジストリ エントリのコンテキスト メニューから使用できるコマンド

オプション	説明
すべてのキーと値	すべての選択されたキー、サブキーおよび値を追加します。
キーとその値のみ	選択されたキーとそのキーの値のみを追加します。サブキーは追加されません。
このキーのみ	選択されたキーのみ追加し、そのサブキーまたは値は追加されません。
キャンセル	変更を加えずにドラッグアンドドロップ操作を終了します。

64 ビット開発システム上で、ソースマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示する

InstallShield を 64 ビット開発システム上で使用する場合、InstallShield が表示する [レジストリ] ビューは、使用中のマシンのレジストリの 32 ビットおよび 64 ビット領域の両方を表示します:

- HKEY_LOCAL_MACHINE¥Software
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node

このサポートによって、プロジェクトのレジストリ データの変更を構成する際、これらのソース領域からこのビューのインストール先ペインの適切な領域にエントリーをドラッグ アンド ドロップすることが可能となります。

インストールがレジストリ データを、32 ビット領域にリダイレクトせずに、64 ビット ターゲット システム上のレジストリの 64 ビット領域にインストールする場合、レジストリ データを 64 ビットとしてマークされたコンポーネントに配置します。[レジストリ] ビューのソースペインから 64 ビット データを、ビュー内のインストール先ペインにドラッグするだけでは、そのコンポーネントが 64 ビットであるとマークされません。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

データを他のマシンからインポートする

ただし、ドラッグ アンド ドロップ操作は、レジストリ エントリがインストール開発システムに存在する場合にのみ有効です。別のマシンからのレジストリ データがある場合、REG ファイルのインポート ウィザードを使って、データをインポートできます。

レジストリ キーの削除

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***レジストリ キーを削除するには、以下の手順を実行します:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. *基本の MSI*、*DIM*、*InstallScript MSI*、*マージ モジュール*、*MSI データベース*、*MSM データベース*、*トランスフォーム*、*および QuickPatch プロジェクトの場合* – **[ビュー フィルター]** リストでレジストリ キーを含むコンポーネントをクリックするか、**[すべてのアプリケーション データ]** を選択して、製品のレジストリ キーをすべて表示します。

   InstallScript と InstallScript Object プロジェクトの場合–**[インストール先コンピューターのレジストリ] ビュー** ペインで、レジストリ キーを含む既存のレジストリ セットを開きます。
3. **[インストール先コンピューターのレジストリ]** ビュー ペインで、削除するレジストリ キーを右クリックして、**[削除]** をクリックします。

レジストリ値の作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***ターゲット システムで作成されるレジストリ キーを指定するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、値を追加するキーをクリックします。そのキーのすべての既存のレジストリ値が **[インストール先コンピューターのレジストリ データ]** ペインにリストで表示されます。

3. 値リスト内で右クリックしてから、登録するデータ型をクリックします。使用可能なオプションは次の通りです。

テーブル 3 • レジストリ値の種類

オプション	説明
デフォルト値	キーのデフォルト値。
文字列値	規定の長さのテキスト文字列。
バイナリ値	この値は 16 進数値として解析され、格納されます。
DWORD 値	4 バイト (32 bit) の長さの数値で表示されるデータ。
複数文字列値	ヌル文字で終わる文字列配列、および 2 つの ヌル文字で終わる文字列配列としてフォーマットされた複数テキスト文字列。このコマンドを選択すると、[複数行文字列値] ダイアログ ボックスが起動します。
展開可能な文字列値	この値は展開可能な文字列として解析され、格納されます。MSDN (Microsoft Developer's Network) によると、展開可能な文字列レジストリ値は、環境変数への非展開参照を含むヌル文字で終わる文字列です (例、**%PATH%**)。

InstallShield は、New Value (新しい値) (ここで *n* は連続番号です) という名前で新しい値を追加します。ここで値名をわかりやすい名前に変更するか、または後で値を右クリックして [名前の変更] を選択し、新しい名前を付けます。

プロジェクト: *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム および QuickPatch プロジェクトでは、すべてのレジストリの値を、コンピューターのキー パスとして使用できます。*

レジストリ値データの変更

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***ターゲット システムで作成されるレジストリ値のデータを変更するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、変更する値を含むレジストリ キーをクリックします。すべてのレジストリ値が、**[インストール先コンピューターのレジストリ データ]** ペインに一覧表示されます。
3. 変更する値をダブルクリックします。[レジストリ データの編集] ダイアログ ボックス または [複数文字列値] ダイアログ ボックス が開きます。
4. ダイアログ ボックスで情報をすべて入力し、**[OK]** をクリックします。

プロジェクト: *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム、および QuickPatch プロジェクトの場合　角かっこ ([]) を含む値データを追加するには、各かっこの前に円記号 (¥) を入れて、それをかっこで括ります。これを省略すると、Windows Installer はこの値をプロパティと認識します。たとえば、レジストリに* **[stuff]** *と書き込む場合、値データに* **[¥[]stuff[¥]]** *を使用します。*

レジストリ値の削除

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***レジストリ値をプロジェクトから削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、削除する値を含むレジストリ キーをクリックします。すべてのレジストリ値が、**[インストール先コンピューターのレジストリ データ]** ペインに一覧表示されます。
3. **[インストール先コンピューターのレジストリ データ]** ペインで、削除するレジストリ値を右クリックしてから **[削除]** をクリックします。

レジストリ フラグ

プロジェクト: *レジストリ フラグのサポートは、使用しているプロジェクトの種類によって異なります。*

レジストリ フラグを使用すると、レジストリ エントリのインストールを制御できます。

Windows Installer ベースのプロジェクトにおけるレジストリ フラグ

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

レジストリ エントリは、所属先のコンポーネントがインストールされるとデフォルトでターゲットシステム上に作成されます。ターゲット システムからそのコンポーネントが削除されたときは、これらのレジストリ エントリも削除されます。製品をアンインストールした後もレジストリ エントリをターゲット システムに残しておく場合や、既に存在しない場合にのみレジストリ エントリを作成する場合には、そのキーのインストールフラグを設定する必要があります。

InstallShield では、インストール動作はサブキー レベルで設定されます。あるキー以下のすべての値には同じインストール動作およびアンインストール動作が設定されなければなりません。

キーのレジストリ フラグを変更するには、[レジストリ] ビューでプロジェクトのキーの 1 つを右クリックしてから、以下のテーブルにリストされている任意のコマンドをクリックします。

テーブル 4 · Windows Installer ベースのプロジェクトにおけるレジストリ フラグの種類

アイコン	コマンド	説明
	自動	これはすべてのレジストリキーのデフォルトのオプションです。キーが既に存在しない場合、インストールがそれを作成します。アンインストール中、キーが空白の場合は、それが削除されます。
	インストールのみ (+)	キーが存在しない場合は、作成されます。このキーが所属するコンポーネントがアンインストールされるとき、このキーはターゲット システムに残ります。 このオプションは、サブキーまたは値を含まないキーにのみ使用できます。
	すべてのキーをアンインストール (–)	このフラグが空のキーに割り当てられた場合、キーはインストール時に作成されません。このフラグが値を含むキーに割り当てられていて、そのキーがターゲット システムに存在しない場合、インストール時にそのキーと値が作成されます。どちらの場合においても、キー、すべてのサブキー、および値はアンインストール時に削除されます。これはサブキーや値がインストール後に追加された場合も同様です。

テーブル 4 • Windows Installer ベースのプロジェクトにおけるレジストリ フラグの種類 (続き.)

アイコン	コマンド	説明
	存在しない場合はインストール、存在する場合はアンインストール (*)	このオプションは、1 つの例外 (自動レジストリ フラグ) を除いて、デフォルトの動作に似ています。自動レジストリ フラグの場合、アンインストール中にそのキーが空白でないとき、それは削除されません。**[存在しない場合はインストール、存在する場合はアンインストール (*)]** フラグの場合、アンインストール中にレジストリ キーが存在するときは、そのサブキーまたは値が残されているかどうかに関わらず、キーが削除されます。

InstallScript ベースのプロジェクトにおけるレジストリ フラグ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

インストールがターゲット システムにレジストリ セットをインストールしてレジストリ キーを作成した後、これらのキーの下に他のインストールまたはアプリケーションによって (あるいは直接エンド ユーザーによって) サブキーまたは値が作成される場合があります。インストールされたレジストリ キーに、そのインストール以外によって作成されたサブキーまたは値がある場合、そのレジストリ キーがアンインストールされることを防ぐことができます。そのためには、該当するキーのレジストリ フラグを変更を行います。まず、[レジストリ] ビューにあるプロジェクトのキーの 1 つを右クリックしてから、次のコマンドをクリックします。

テーブル 5 • InstallScript ベースのプロジェクトにおけるレジストリ フラグ

アイコン	コマンド	説明
	複数のアプリケーション間で共有	このキーが属するコンポーネントがアンインストールされるとき、キーにサブキーまたは値がある場合、そのキーはターゲット システムから削除されません。(インストールで作成された値は、**[アンインストール中に削除]** フラグの設定を解除していなければ、既にアンインストールされているはずです。)キーにサブキーまたは値がない場合、キーがインストールによって作成されたかどうか、つまり、インストール時に既に存在していたかどうかに関わらず、ターゲット システムから削除されます。

たとえば、オペレーティング システムが使用するレジストリ値にインストールが変更を行っている場合、アンインストール時にエンド ユーザーのシステム上に製品のレジストリ値の一部をそのまま残したい場合があります。その場合、アンインストールしないレジストリ値を含むキーを右クリックします。そのキーの **[複数のアプリケーション間で共有]** フラグを選択します。キーの値を右クリックしてから、**[アンインストール中に削除]** コマンドが選択されていないことを確認します。

プロジェクト内のレジストリ エントリの検索

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

タスク ***(Windows Installer プロジェクト内の)特定のコンポーネントまたは(InstallScript プロジェクト内の)レジストリ キー内でレジストリ エントリを検索するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. *基本の MSI*、*DIM*、*InstallScript MSI*、*マージ モジュール*、*MSI データベース*、*MSM データベース*、*トランスフォーム*、*および QuickPatch プロジェクト*—**[ビュー フィルター]** リストで、検索するコンポーネントを選択します。**[すべてのエントリを表示]** が選択されている場合、**[レジストリ]** ビューは読取り専用になります。

   InstallScript と InstallScript Object プロジェクトの場合—検索するレジストリ セットを開きます。
3. レジストリ エントリを右クリックして、**[検索]** を選択します。**[検索]** ダイアログ ボックスが開きます。
4. 検索する文字列を入力します。
5. **[検索]** をクリックして、検索を開始します。

Windows Installer プロジェクトの場合、InstallShieldでは、プロジェクトのすべてのレジストリ データではなく、現在のコンポーネントのレジストリ エントリのみが検索されます。

検索順序は、選択されている項目に関係なく、一番上の項目から開始して下へと移動します。一致するとその項目で停止し、キーがハイライト表示されます。検索を続行するには、右クリックして [次を検索] を選択するか F3 を押します。[次を検索] を選択すると、エクスプローラーを下方向に移動しながら、次に一致する項目を検出します。

レジストリ キーのアンインストール動作を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

- *QuickPatch*

InstallShield では、レジストリ エントリのアンインストール動作はキー レベルで設定されます。

タスク ***レジストリ キーのアンインストール動作を設定するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、キーを右クリックして、適切なアンインストールの動作をクリックします。

アンインストール動作のオプション

InstallShield では、インストール動作はサブキー レベルで設定されます。あるキー以下のすべての値には同じインストール動作およびアンインストール動作が設定されなければなりません。使用可能なオプションの一覧は、「レジストリ フラグ」をご覧ください。

レジストリ値で環境変数を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

REG_EXPAND_SZ 文字列値を使用すると、レジストリに格納されているパスに環境変数を使用できます。これらのエントリは、オペレーティング システムで環境変数として認識されるために、特別な形式を必要とします。レジストリに表示される REG_EXPAND_SZ 値の形式は、%TEMP% です。TEMP は、TEMP ディレクトリの標準環境変数です。

構文

このタイプのレジストリ エントリを作成するときは、エントリをシャープ記号で開始し、その次にパーセント記号を付ける必要があります (#%)。次に、環境変数を入力できます。環境変数の前後にはパーセント記号を付けます。したがって、の [レジストリ] ビューで **#%%TEMP%** と入力すると、レジストリに書き込まれたときに **%TEMP%** と表示されます。

このエントリの "種類" フィールドには REG_EXPAND_SZ、"データ" フィールドには %TEMP% と表示されます。

レジストリにプロパティ値を書き込む

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

Windows Installer ベースのプロジェクトでは、Windows Installer のプロパティをレジストリ値に使って、製品が後で使用する情報を保存することができます。実行時に、Windows Installer は 自動的にレジストリ データ中の **[PropertyName]** のような式は、インストーラーの実行時に Windows Installer によって、**PropertyName** というプロパティの値に自動的に置き換えられます。[プロパティ マネージャー] ビューで定義したすべてのプロパティを同じように使用できます。

たとえば、ソフトウェアのインストール先を保存する場合は、**[INSTALLDIR]** というデータをもつレジストリ値を作成します。実行時に、インストールがレジストリ値を作成するとき、そのレジストリ値のデータは、アプリケーションがインストールされる場所に設定されます。

同じフォーマットを、レジストリキーの名前および値名にも使用できます。たとえば、キー名に会社名を付ける場合、キーの名前に **[Manufacturer]** と入力します。レジストリ キーがターゲット システムで作成されると、このキーの名前が、[一般情報] ビューの で入力されたとおりに ″パブリッシャー″ 設定の値になります。

レジストリ ファイルをインポートする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield では、別のインストール プロジェクトのレジストリ (.reg) ファイル、または InstallShield の外で作成した既存のレジストリ ファイルをインポートすることができます。

タスク ***.reg ファイルをインポートするには、以下の手順を実行します。***

1. REG ファイルのインポート ウィザードを起動します。
2. ウィザードパネルの指示に従って .reg ファイルをインポートします。

コンポーネントまたはレジストリ セットに .reg ファイルをインポートすると、そのレジストリ データがコンポーネントまたはレジストリ セットのレジストリ データに追加され、コンポーネントまたはレジストリ セットがエンド ユーザーのシステムにインストールされている場合に、そのシステムに書き込まれます。

レジストリ ファイルのエクスポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

レジストリ ファイル (.reg) をエクスポートすると、プログラムのレジストリ データの一部がレジストリ ファイルにコピーされます。

タスク ***レジストリファイルをエクスポートするには、次の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、エクスポートするレジストリ エントリを右クリックして、**[REG ファイルのエクスポート]** または **[選択したブランチのエクスポート]** のどちらかを選択します。
   - **REG ファイルのエクスポート** では、コンポーネントのレジストリ データ全体がエクスポートされます。
   - **[選択したブランチのエクスポート]** は、現在のレジストリ キーとサブキーだけをエクスポートします。

コンポーネントのキーパスを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

キー パスは、コンポーネントの有無を検出するために Windows Installer が使用する各コンポーネントの一意のファイルです。この値には、ファイルまたはフォルダーが含まれていなければなりません。

タスク ***コンポーネントのキーパスを設定するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインで、キーパスに使用するレジストリ データを含むレジストリ エントリをクリックします。
3. **[インストール先コンピューターのレジストリ データー]** ペインで、レジストリ キーの値を右クリックして **[キー パスの設定]** を選択します。

値の文字列、バイナリ、または DWORD アイコンは、キー アイコンに置き換えられます。

タスク ***コンポーネントのキーパスを削除するには、以下の手順に従います。***

キー パスを右クリックして [**キー パスのクリア**] をクリックします。

メモ: *コンポーネントは、1 つのキー パス、または、1 つのキーフ ァイルを持つことができます。キーパスまたはキーファイルがコンポーネントに既に割り当てられている場合、他のキーパスを設定しようとすると、警告メッセージが表示されます。ダイアログ ボックスで [はい] をクリックして、既存のキー ファイルまたはキー パスを新しいキー パスに置き換えます。*

レジストリ キーのアクセス許可を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield で、ロックダウンされた環境で製品を実行するエンドユーザーのために、レジストリ キーを保護するための設定を構成することができます。レジストリ キーのアクセス許可を特定のグループとユーザーに割り当てることができます。たとえば、管理者グループに特定のレジストリ キーについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。

タスク ***レジストリ キーのアクセス許可を構成するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[レジストリ]** をクリックします。
2. **[インストール先コンピューターのレジストリ] ビュー** ペインでレジストリ キーを右クリックし、**[アクセス許可]** ボタンをクリックします。**[アクセス許可]** ダイアログ ボックスが開きます。
3. 必要に応じて、アクセス許可を追加/変更/削除します。詳細については、「レジストリ キーの [アクセス許可] ダイアログ ボックス」を参照してください。

プロジェクトの [一般情報] ビューにある "ロックダウンの設定方法" 設定の選択に従って、InstallShield は ISLockPermissions テーブルまたは LockPermissions テーブルのどちらかにアクセス許可データを追加します。詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

レジストリ テーブルのプライマリ キー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows Installer では、Registry テーブルに追加するレジストリ キーと値それぞれについて、一意のプラマリ キーが必要です。レジストリ エントリの作成状況がすべて見えるように、InstallShield では、データベースの [レジストリ] テーブルのエントリをビルドする時に各エントリに一意の名前が割り当てられます。

カスタム アクションを作成するときに、エントリのプライマリ キーが必要になるときがあります。InstallShield では、[セットアップのデザイン] ビューまたは [コンポーネント] ビューの [レジストリ データ] エクスプローラーで、レジストリ キーまたは値のプライマリ キーを指定することができます。[レジストリ] ビューを使って、キー、値、またはプライマリ キーを割り当てることはできません。

Registry のプライマリ キーを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***レジストリ キーまたは値のプライマリ キーを指定するには、以下の手順に従います:***

1. キーまたは値を右クリックして、**[MSI 値]** を選択します。[MSI 値] ダイアログ ボックスが開きます。
2. キーの名前を入力します。プライマリ キーは Windows Installer 識別子でなければならないので、名前には文字、数字、下線 (_)、およびピリオド (.) だけを使用し、名前の最初には文字または下線を使用します。

プライマリ キーを表示または変更するには、レジストリ キーまたは値を右クリックし、もう一度 [MSI 値] をクリックします。

値を指定しない場合、InstallShield によって、Registry テーブルのこのエントリに一意のプライマリ キーが生成されます。

レジストリ キーに複数行の文字列値を入力する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

特定のレジストリ キーに値を追加するとき、[レジストリ] ビューでは複数行の文字列を指定することができます。

タスク ***複数行の文字列値を追加するには、以下の手順に従います。***

1. 複数行の値を追加するレジストリ キーを右クリックして、**[新規]** をポイントし、**[複数行の値]** を選択します。[複数行文字列値] ダイアログ ボックスが開きます。
2. レジストリ値の変更方法を選択してから、各ヌル区切り文字列の行を入力します。

3. [OK] をクリックします。

メモ: *文字列にはスペースだけを含めることはできますが、空白にしたり、文字列の区切り文字である ["] を使うことはできません。*

ユーザーごとのインストールで HKCU の下にエントリを書き込む

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

現在のユーザーが HKEY_LOCAL_MACHINE の下にあるキーを変更するための十分な権限を持っているとはかぎりません。そのような場合、HKEY_CURRENT_USER の下にエントリを書きこむ必要があります。

[レジストリ] ビューで HKEY_USER_SELECTABLE を選択すると、インストールのタイプとエンドユーザーのアクセス権にしたがって、適切なレジストリ ハイブにエントリが作成されます。ユーザーごとのインストール (ALLUSERS プロパティが 2 に設定されているか、またはインストールがユーザー レベルのアクセス権限を持つユーザーによって実行されている場合)では、これらのエントリは HKEY_CURRENT_USER に作成されます。マシンごとのインストールでは、つまり、ALLUSERS がヌル値ではなく、エンドユーザーが管理者である場合には、エントリは HKEY_LOCAL_MACHINE に書き込まれます。

メモ: *Windows では、HKEY_LOCAL_MACHINE の下に直接新しいキーを作成することはできません。このため、HKEY_USER_SELECTABLE の下に作成した情報は、HKEY_LOCAL_MACHINE および HKEY_CURRENT_USER に共通した唯一のサブキーである、SOFTWARE サブキーに配置する必要があります。*

たとえば、HKEY_USER_SELECTABLE¥SOFTWARE¥MyCompany というキーは有効ですが、HKEY_USER_SELECTABLE¥MyCompany というキーは有効ではありません。

レジストリで複数行文字列の設定または取得を行う

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

複数行文字列の設定

複数行文字列を設定するには、**RegDBSetKeyValueEx** 関数を呼び出します。szValue パラメーターには、ヌル文字で区切られた、追加されるサブ文字列が必要です。メインの文字列の終わりには、その終わりを示すのに 2 つのヌル文字を追加する必要があります。今述べた文字列を作成するには、以下の手順を実行します。

1. 文字列を初期化して各サブ文字列をスペース文字で区切って含めます。
2. 各行のあとに、ヌル文字列の ASCII 値を文字列部分に割り当てます。
3. 文字列の終わりにヌル文字 (ASCII 値 0) を追加で割り当てます。

例:

```
svValue = "これは第 1 行目です。これは第 2 行目です。これは第 3 行目です。";
svValue[17] = 0;
svValue[35] = 0;
svValue[55] = 0;
svValue[56] = 0;
```

複数行文字列の取得

複数行文字列を取得するには、**RegDBGetKeyValueEx** 関数を呼び出します。サブ文字列は、メインの文字列の中でヌル文字で区切られて戻されます。サブ文字列を抽出するには、次のコードを使用して文字列を文字列リストに読み込んで **SdShowInfoList** ダイアログに表示させます。

```
// svValue は、RegDBGetKeyValueEx で読み込まれた文字列です。
listID = ListCreate (STRINGLIST);
if (listID != LIST_NULL) then
    StrGetTokens( listID, svValue, "" );
    SdShowInfoList (szTitle, szMsg, listID);
endif;
```

レジストリ関数の使い方

プロジェクト: *InstallScript MSI と基本の MSI プロジェクトの場合、InstallScript コードを使ってレジストリのキーと値を作成する代わりに、InstallShield の [レジストリ] ビューを利用することが推奨されます。この方法ですべてのレジストリの変更を処理すると、Windows Installer サービスを通したクリーン アンインストールが可能となります。*

InstallScript 言語には、次のようなレジストリへアクセスするためにインストールを簡単に構成できる多数のビルトイン関数があります : レジストリ キーの読み取り、作成、および削除、アンインストール用のレジストリ関連パラメーターの設定。

たとえば、アプリケーションで使われているレジストリキーを検出することでターゲット システムにあるアプリケーションが存在するかどうかを判断できることもあります。多数のアプリケーションは、データをレジストリ キー HKLM¥SOFTWARE¥CompanyName¥ProductName¥ProductVersion に格納します。レジストリ キーが存在するかどうかを確認するには、次の InstallScript スクリプトで **RegDBKeyExist** 関数を呼び出します:

```
// 他の RegDB 関数を呼び出す前に常にルートキーを設定する
RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

if (RegDBKeyExist("Software¥¥ThisCo¥¥ThisApp") = 1) then
   MessageBox("ThisApp はシステムに存在します。", INFORMATION);
```

endif;

InstallScript と共に利用できるビルトインレジストリ関数について詳しく学ぶには、次をご覧ください。

- レジストリ関数
- レジストリ関連の特殊関数

レジストリからデータを読み込む

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

基本の MSI プロジェクト

AppSearch および RegLocator テーブルには、レジストリの値を読み込むことができます。上記の例で使用した RegisteredOwner の値を読み込むには、ダイレクト エディターを使用して AppSearch テーブルに次のレコードを追加します。

テーブル 6 ・ AppSearch メソッドの値

フィールド	値	コメント
プロパティ	REGISTERED_OWNER	パブリック プロパティにする必要があります。
Signature_	registry_sig	

次に、RegLocator テーブルに次のレコードを追加します :

テーブル 7 ・ RegLocator テーブルの値

フィールド	値	コメント
Signature_	registry_sig	上記の Signature_ と同じ
ルート	2	HKEY_LOCAL_MACHINE
キー	Software¥Microsoft¥Windows NT¥CurrentVersion	
名前	RegisteredOwner	
種類	2	レジストリ データ

AppSearch アクションの実行後、REGISTERED_OWNER プロパティには、レジストリから読み込んだデータが含まれます。値が見つからない場合、プロパティが未定義 (空) です。

AppSearch および RegLocator テーブルに関する詳細については、Windows Installer ヘルプ ライブラリを参照してください。

Windows Installer ベースのプロジェクトでは、データの検索に システム検索ウィザードを使用します。

InstallScript MSI プロジェクト

InstallScript では、**RegDBGetKeyValueEx** 関数を使用できます。たとえば、HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion キーから RegisteredOwner の値を読み込むには、次のようなコードを記述します。

```
function readRegisteredOwner( )
    STRING svRegisteredOwner;
    NUMBER nvType, nvSize;
begin
RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

RegDBGetKeyValueEx(
    "Software¥¥Microsoft¥¥Windows NT¥¥CurrentVersion",
    "RegisteredOwner",
    nvType,
    svRegisteredOwner,
    nvSize);

MessageBox(
    "登録された所有者は: " + svRegisteredOwner,
    INFORMATION);
end;
```

Windows Installer プロパティは、**MsiSetProperty** 関数を使って読み込んだ値と等しく設定することができます。

.ini ファイル データの変更

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

初期化 (.ini) ファイルは、アプリケーションの次回の使用までの間において、情報の保存と取得を行うデータベースの役割を果たします。Boot.ini や Wininit.ini などの .ini ファイルは、オペレーティング システムで使用されます。[INI ファイル変更] ビューでは、ターゲット システム上で .ini ファイルに行う変更を指定することができます。ターゲット システムにあるすべての .ini ファイルを編集することができますが、システム .ini ファイルの変更は推奨しません。

タスク ***.ini ファイルを編集するには、以下の手順を実行します。***

1. .ini ファイル リファレンスを作成します。
2. .ini ファイルにセクションを追加します。
3. .ini ファイルにキーワードを追加します。

.ini ファイル リファレンスを作成するには、少なくとも 1 つのコンポーネントを作成している必要があります。.ini ファイル リファレンスを作成したときにコンポーネントがなかった場合は、[新しいコンポーネントの作成] ダイアログ が表示され、ここでコンポーネントを作成することができます。

.ini ファイルのリファレンスを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

.ini ファイル変更作成の最初の手順は、編集するファイルへのリファレンスを作成することです。そのためには、編集するターゲット システム上のファイルの場所と名前が必要です。指定した場所にファイルがない場合、インストールがこのファイルを変更することはできません。

タスク ***.ini ファイルにリファレンスを作成するには、次の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. **[INI ファイル]** エクスプローラーを右クリックして、**[INI ファイルの追加]** をクリックします。

[INI ファイル] エクスプローラーに .ini ファイルの新しいリファレンスが追加されます。.ini ファイルの設定を右側のペインで構成します。各設定の詳細については、「.ini ファイルの設定」を参照してください。

.ini ファイルへのリファレンスを作成したら、セクションを .ini ファイルに追加する次のステップに進みます。

既存の .ini ファイルをインポートする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

InstallShield では、プロジェクトに既存する .ini ファイルをインポートできます。.ini ファイルをインポートした後、必要に応じて [INI ファイルの変更] ビューを使って、ターゲット システムに加える変更を構成することができます。

タスク ***既存する .ini ファイルをインポートするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. **[INI ファイル]** エクスプローラーを右クリックしてから、**[INI ファイルのインポート]** を選択します。**[開く]** ダイアログ ボックスが開きます。
3. インポートする .ini ファイルを選択してから、**[開く]** をクリックします。

InstallShield が .ini ファイルとそのセクション、キーワード、および値をプロジェクトにインポートします。

.ini ファイルのセクションを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

編集する .ini ファイルを指定した後の次のステップは、そのファイル内で変更するセクションの指定です。.ini ファイルは、セクションに分割され、それぞれのセクションにはキーワードがあります。セクションは、**[SectionName]** のように角かっこ [ ] で囲まれて識別されます。

タスク ***.ini ファイルのセクションを指定するには、次の操作を実行します。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. 存在しない場合は、.ini ファイルへの参照を作成します。
3. **[INI ファイル]** エクスプローラーで、セクションを含める .ini ファイルを右クリックして、**[セクションの追加]** をクリックします。

[INI ファイル] エクスプローラーにセクションが追加されます。右側のペインでセクションの設定を構成します。各設定の詳細については、「.ini ファイルのセクションの設定」を参照してください。

.ini ファイル内のセクションを指定した後、キーワードを追加できます。

.ini ファイルのキーワードを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、.ini ファイルを変更するための .ini ファイル関数 が含まれています。InstallScript プロジェクトで、これらの関数を使用できます。

.ini ファイルのキーワードは、.ini ファイルの構造の最下位レベルです。キーワードには、アプリケーションの終了から次回の起動までの間に保持しておく必要のあるデータを格納します。

プロジェクトに .ini ファイルに追加してから 1 つ以上のセクションを設定した後、セクションにキーワードを追加して、キーワードのプロパティを構成できます。キーワードのプロパティには、キーワードの値のほかに実行するアクション (データ値の置換、既存のデータ値への追加など) が含まれます。

タスク ***.ini ファイルにキーワードを指定するには、次の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[INI ファイルの変更]** をクリックします。
2. **[INI ファイル]** エクスプローラーで、セクションを右クリックして、**[キーワードの追加]** をクリックします。

[INI ファイル] エクスプローラーにキーワードが追加されます。右側のペインでキーワードの設定を構成します。各設定の詳細については、「.ini ファイルのキーワードの設定」を参照してください。

.ini ファイルからデータを読み込む

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

キー名などのデータを .ini ファイルから読み取るには、.ini ファイルがどこにあるか(例、ネットワーク上)にかかわらず、**GetProfString** 関数を利用することができます。次のスクリプト例を参考にしてください:

```
/*

.ini ファイルの名前が Test.ini で、以下を含むと想定します:

[ProductSettings]
Key1=One
Key2=2
Key3=III
Key4=....

[OtherSettings]
Key1=Value1
Key2=Value2

*/

function OnBegin()
    STRING svKey1Value; //  GetProfString が入力
begin

// 単一値を読み込む
GetProfString(
    "¥¥¥¥Server¥¥Config¥¥Test.ini", // ファイル名。2 つの円記号を
実際の円記号の前につけます
    "ProductSettings", // 角括弧なしのセクション名
    "Key1", // キー名
    svKey1Value); // キー値をキャプチャする STRING 変数

MessageBox("Key1 の値は: " + svKey1Value, INFORMATION);

end;
```

.ini ファイルからすべてのキー名を取得する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

.ini ファイルのセクションからキー名を読み取るには、次のスクリプト使用します:

```
function OnBegin()
    STRING svAllKeyNames; //  GetProfString が入力
    LIST   listKeyNames;  // キー名を含む文字列リスト
begin

// 単一値を読み込む
GetProfString(
    "¥¥¥Server¥¥Config¥¥Test.ini", // ファイル名。2 つの円記号を
実際の円記号の前につけます
    "ProductSettings", // 角括弧なしのセクション名
    "", // キー名; ヌル文字列 "" はすべてのキー名を入手するの意
    svAllKeyNames); // キー名に保存する STRING 変数

listKeyNames = ListCreate(STRINGLIST);

// キー名を含む文字列を分離する
StrGetTokens(listKeyNames, svAllKeyNames, "");

SdShowInfoList("キー名", "こちらです:", listKeyNames);

// 必要であれば、リストの中のキー名をループして各キー値を
キー値

ListDestroy(listKeyNames);

end;
```

ODBC リソースの構成

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム構成のより複雑な領域の 1 つに、ODBC ドライバー、データソース名 (DSN)、トランスレーターの設定があります。ODBC リソースは、すべての必須属性とともにシステムに正しく登録されている必要があります。また、ドライバーおよびトランスレーターの場合、インストール .dll ファイルなどの必要なファイルをインストールする必要があります。このプロセスは、ODBC リソース ビューを使って単純化できます。このビューでは、開発システムにインストールされているドライバー、データ ソース、およびトランスレーターを選択できます。

ODBC リソースのインストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ODBC リソース] ビューには、すべての ODBC ドライバー、データソース名 (DSN)、およびシステムに登録されているトランスレーターが表示されます。DSN は、関連ドライバーの「子」として表示されます。エクスプローラーを展開すると、既存の ODBC リソースがすべて表示されます。

インストールに ODBC リソースを組み込むには、ビューの左上にある [ODBC リソース] ペインで名前の隣にあるチェック ボックスをクリックします。インストール プロジェクトでは、リソースを少なくとも 1 つの機能に割り当てる必要があります。そのあと、選択したリソースについて属性を設定することができます。

追加の ODBC リソースを含める

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

エクスプローラーのリストにない ODBC リソースをインストールすることもできます。これらを追加する場所は、リソースのタイプにより異なります。

タスク ***[ODBC リソース] ビューの [ODBC リソース] エクスプローラーにリストされていないドライバーを追加するには、以下の手順を実行します。***

1. **[ODBC リソース]** エクスプローラーで、**[ドライバーと DSN]** を右クリックして、**[新しいドライバー]** をクリックします。
2. ドライバーの新しい名前を入力するか、後で F2 キーを押して名前を変更します。
3. 新しいドライバーの隣りにあるチェック ボックスを選択して、インストールに含めます。

タスク ***[ODBC リソース] ビューの [ODBC リソース] エクスプローラーにリストされていないデータ ソースを追加するには、以下の手順を実行します。***

1. **[ODBC リソース]** エクスプローラーで、ドライバーを右クリックして、**[新しいDSN]** をクリックします。
2. DSN の新しい名前を入力するか、後で F2 キーを押して名前を変更します。
3. 新しい DSN の隣りにあるチェック ボックスを選択して、それをインストールに含めます。

メモ: *トランスレーターをリストに追加することはできません。トランスレーターは開発システムにインストールしてから選択する必要があります。*

ODBC リソースを機能に関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクトのほとんどのデータ同様に、ODBC リソースも機能に関連付ける必要があります。機能がターゲット システムにインストールされると、ODBC リソースは機能の一部としてインストールされます。

インストール用のドライバー、DSN、トランスレーターをインストールするように選択した後、[機能] リストが [ODBC リソース] エクスプローラーの右側で使用できます。ODBC リソースが属する機能のすべてを選択します。InstallShield により新しいコンポーネントが作成され、それが選択された各機能に関連付けられます。リソースは、複数の機能に関連付けられている場合でも一度だけインストールされます。しかし、関連付けられている機能がインストールされていない場合、リソースはインストールされません。

InstallScript MSI プロジェクトでは、その他の機能が存在しない場合は DefaultFeature にリソースが追加されます。基本の MSI プロジェクトでは、インストールに ODBC リソースを追加するときに機能が存在しない場合、[新しい機能の作成] ダイアログ ボックスが表示されます。このダイアログ ボックスは、機能の作成をプロンプトします。1 つの ODBC リソースに関連付けられている機能が 1 つだけの場合、別の機能を少なくとも 1 つ選択しない限り、その機能の選択を解除することはできません。

ODBC リソース属性の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ODBC リソースが開発システムに登録されている場合、その現在の属性は、[ODBC リソース] ビューのプロパティリストに表示されます。ドライバー、データソース名 (DSN)、またはトランスレーターを選択すると、[プロパティ] ペインで属性を編集できます。

新しい属性エントリの追加と削除

トランスレーターに属性を追加することは、サポートされていません。

タスク ***新しい属性をドライバーまたは DSN に追加するには、以下の手順に従います:***

1. **[ODBC リソース]** ビューを開きます。
2. **[ODBC リソース]** エクスプローラーで、新しい属性を含めるドライバー、または DSN をクリックします。
3. プロパティ シートの最後の行をクリックして、**[プロパティ]** または **[値]** 列に適切な情報を追加します。

タスク ***ドライバーまたは DSN から属性を削除するには、以下の手順を実行します。***

1. **[ODBC リソース]** ビューを開きます。
2. **[ODBC リソース]** エクスプローラーで、削除する属性を含む ODBC リソースをクリックします。
3. プロパティ シートで、削除する行を右クリックしてから、[**削除**] をクリックします。

環境変数を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。

環境変数は、インストールと共にターゲット システム上に設定できる名前と値の組み合わせで、アプリケーションおよび実行されているその他のプログラムによってアクセスできます。環境変数はレジストリに格納されます。

[環境変数] ビュー を使うと、インストールを使ってターゲット システム上に環境変数を作成、設定 (または変更) したり、システム上の環境変数を削除することができます。このビューで環境変数プロパティを指定することもできます。

環境変数の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。

タスク

新しい環境変数を作成するか、または既存の環境変数を変更するには、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[環境変数]** をクリックします。
2. **[環境変数]** を右クリックして、**[環境変数の追加]** をクリックします。
3. InstallShield は、NewEnvironmentVariable*X* (*X* は数字) を持つ新しい環境変数を追加します。変更、削除または作成する変数名を入力します。
4. 右側のペインで、環境変数の設定を構成します。各設定の詳細については、「環境変数の設定」を参照してください。

環境変数の例

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

```
/*****************************************************************¥
* 次のコードは、Windows の
*  全システムで環境変数を作成します。OnEnd イベント ハンドラー関数ブロック
* (またはその他の関数ブロック) を変更して、このサンプルコードを含めることができます。
*
*
* メモ: また、現在のユーザーがこのコードを使用するには、
* 管理者特権が必要です。
¥*****************************************************************/

  #define WM_WININICHANGE 0x001A
  #define HWND_BROADCAST 0xffff
  NUMBER nResult;
  STRING szKey, szEnv;
  WPOINTER pEnv;

  begin

     szKey = "SYSTEM¥¥CurrentControlSet¥¥Control¥¥Session Manager¥¥Environment";
     RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);
     nResult = RegDBSetKeyValueEx(szKey, "Fame", REGDB_STRING, "C:¥¥Test", -1);
     if (nResult < 0) then
        MessageBox("環境変数の設定に失敗しました", WARNING);
     else
        MessageBox("環境変数は正常に設定されました", INFORMATION);
        // レジストリをすべてのアプリケーションにフラッシュ。
        szEnv = "環境";
        pEnv = &szEnv;
        SendMessage (HWND_BROADCAST, WM_WININICHANGE, 0, pEnv );
     endif;
     // RebootDialog("", "", SYS_BOOTMACHINE);
  end;


/*****************************************************************¥
* 次のコードは、Windows の
*  現在のユーザーの環境変数を作成します。OnEnd イベント ハンドラー関数ブロック
* (またはその他の関数ブロック) を変更して、このサンプルコードを含めることができます。
*
```

```
*
* メモ: また、現在のユーザーがこのコードを使用するには、
* 管理者特権が必要です。
¥*****************************************************************/

 #define WM_WININICHANGE 0x001A
 #define HWND_BROADCAST 0xffff

 NUMBER nResult;
 STRING szKey, szEnv;
 WPOINTER pEnv;

 begin
    szKey="環境";
    RegDBSetDefaultRoot(HKEY_CURRENT_USER);
    nResult=RegDBSetKeyValueEx(szKey,"Fame",REGDB_STRING,"C:¥¥test",-1);
    if (nResult < 0) then
       MessageBox("環境変数の設定に失敗しました",WARNING);
    else
       MessageBox("環境変数は正常に設定されました", INFORMATION);
       // レジストリをすべてのアプリケーションにフラッシュ。
       szEnv = "環境";
       pEnv = &szEnv;
       SendMessage (HWND_BROADCAST, WM_WININICHANGE, 0, pEnv );
    endif;
    //RebootDialog("","",SYS_BOOTMACHINE);
 end;
```

XML ファイルの変更

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

製品に関連する設定を保存する .xml ファイルの変更が必要な場合、または web.config および machine.config などの標準構成ファイルの変更が必要な場合があります。InstallShield には、実行時にターゲット システムで XML ファイルを変更する機能が搭載されています。XML ファイルは、インストールの一部であっても、またターゲット システム上に既に存在するファイルであっても構いません。

インストール中に XML ファイルを変更する方法については、ドキュメントのこのセクションを参照してください。

XML に関する詳細は、次の Web サイトをご覧ください。

- World Wide Web Consortium (http://www.w3.org)

- W3 Schools (http://w3schools.com)

ヒント: XML ファイルは、InstallShield 製品の機能の他の場所でもサポートされています。InstallShield の [システム検索] 機能では、指定した XML ファイルで、要素の属性値、内容、または有無を検索することができます。詳細については、「XML データの検索」を参照してください。

XML ファイルの変更の概要

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

デザイン時のタスク

[XML ファイル変更] ビューでは、実行時に XML ファイルに加えられる変更を定義することができます。変更を定義するときの一般的な手順は、以下の通りです。

1. 変更する XML ファイルがインストールの一部の場合、[ファイルとフォルダー] ビューでプロジェクト内にあるコンポーネントにベースの XML ファイルを追加します。
2. [XML ファイルの変更] ビューで、変更するファイルの XML ファイルの参照を作成します。
3. ターゲット マシン上の XML ファイルの保存場所を指定します。
4. XML ファイルの変更を含める機能を指定します (複数可)。ここで、ステップ 1 でコンポーネントを追加した機能と同じ機能を指定することもできますし、別の機能を指定することもできます。
5. 実行時に発生する変更を構成します。

場合によって、MSXML 再配布可能ファイルをプロジェクトに追加する必要があります。詳細については、「XML ファイルの変更の実行時の要件」を参照してください。

実行時の動作

[XML ファイル変更] ビューでは、実行時の動作を次のように構成することができます。

- 名前空間のマッピングを XML ファイルに追加し、名前空間のプレフィックスを要素と属性に追加する。
- 指定された要素が存在しない場合、それを作成する。
- 存在する場合、要素を変更するが、存在しない場合は、作成しない。
- 指定された XPath 式に一致する XML ファイルの最初の要素のみを変更するか、すべての一致するインスタンスを変更する。

- アンインストール時に要素を削除する。
- 要素にコンテンツを設定する。
- インストールまたはアンインストール時 (または両方) に、新しい属性を作成する。
- インストールまたはアンインストール時 (または両方) に、既存の属性を削除する。
- インストールまたはアンインストール時 (または両方) に、文字列を既存の属性値に付加する。
- 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクトの場合: 実行時に、要素、属性、属性値、および要素のコンテンツの Windows Installer のプロパティを適切な値で置換する。
- InstallScript プロジェクトの場合: 実行時に、要素、属性、属性値、および要素のコンテンツの InstallScript の文字列変数を適切な値で置換する。

XML ファイルの変更を含むインストールがターゲット システムで実行されたとき、MSXML が XML ファイルを解析し、構成した変更に関連付けられている XPath 式を実行します。[XML ファイルの変更] ビューにある [詳細] タブで、XML ファイルのターゲット システムで実行される XPath 式を表示することができます。

MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。

基本的な XPath 式の作成方法については、「XPath 式を使って、XML ファイル内の XML データを検索する」を参照してください。

メモ: *実行時に XML ファイルが存在しなく、XML ファイルの [詳細] タブで* ***[XML ファイルが存在しない場合、常に作成]*** *チェック ボックスが選択されているとき、XML ファイルが [XML ファイルの変更] ビューで構成された XML データで作成される。*

XML ファイルの変更の実行時の要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルの変更の実行時機能には、ターゲット マシンに Microsoft XML Core Services 3.0 (MSXML) 以降が搭載されいることが必要です。

XML ファイルの変更を含むインストールがターゲット システムで実行されたとき、MSXML が XML ファイルを解析し、構成した変更に関連付けられている XPath 式を実行します。MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。

InstallShield には、MSXML の異なるバージョンについてそれぞれ再配布可能ファイルが用意されています。ターゲット システムに MSXML がない、または適切なバージョンの MSXML ことが考えられる場合、[再配布可能ファイル] ビュー (基本 MSI プロジェクトと InstallScript MSI プロジェクト) または [前提条件] ビュー (InstallScript プロジェクト) のいずれかで、プロジェクトに適切な MSXML 再配布可能ファイルを追加することができます。

MSXML についての詳細は、「MSDN ライブラリ」を参照してください。

XML ファイル リファレンスの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

実行時に変更する XML ファイルを追加する場合、プロジェクトで [ファイルとフォルダー] ビューを使用して、ベースの XML ファイルをコンポーネントに追加することをお勧めします。次いで、その XML ファイル リファレンスを [XML ファイル変更] ビューに追加し、実行時にファイルに加える変更を指定します。XML ファイル リファレンスを追加するときの最も簡単な方法は、ファイルを [XML ファイルの変更] ビューにインポートする方法です。

重要: *[XML ファイルの変更] ビューは、XML ファイルにあるすべてのノードについてノードを表示するようにはデザインされていません。パフォーマンス速度を向上させるために、[XML ファイルの変更] ビューでは、ベースの XML ファイルとは異なる設定のみが表示されます。*

- *変更する XML ファイルがインストールの一部である場合、[XML ファイル変更] ビューには、XML ファイルが実行時にインストールされた後に追加、変更、または削除されるノード、またはノード セットのみが表示されます。*
- *変更する XML ファイルがターゲット システムに既に存在するファイルの場合、[XML ファイル変更] ビューには、実行時に追加、変更、または削除されるノードおよびノード セットのみが表示されます。*

したがって、ファイルをインポートするとき、実行時に変更する XML ファイル内のノードのみをインポートします。

[XML ファイルの変更] ビューでは、新しい要素が XML ファイルに表示されるときの順番を指定することはできませんので注意してください。従って、要素が特定の順番で表示されている必要がある製品の場合、実行時に変更するノードのみをインポートすると、潜在的な問題を回避することができます。

タスク ***XML ファイルを [XML ファイルの変更] ビューへインポートするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーを右クリックし、**[インポート]** をクリックします。**XML の設定のインポート ウィザード**が開きます。
3. 変更する XML ファイルのノードのみを選択して、ウィザードのパネルを完成します。
4. **[インポート]** をクリックします。

インポートした XML ファイルの新しいノードが追加されます。XML ファイル ノードには、ウィザードで選択したノードがそれぞれ含まれています。各ノードは、実行時に発生する XPath クエリを示します。[XML ファイルの変更] ビューでインポートした XML ファイルの新しいコンポーネントも追加されます。

XML ファイルの設定を構成して、要素、属性、コンテンツを任意に変更することができます。

ヒント: *XML ファイルは、[XML ファイル] エクスプローラーを右クリックして、[新規ファイル] をクリックしても、追加することができます。ほとんどの場合、サードパーティの XML エディターまたはテキスト エディターを利用して XML ファイルを作成する方法が一般的です。次に、このファイルを [ファイルとフォルダー] ビューからプロジェクトに追加します。次いで、ファイルを上記の手順にしたがってインポートし、実行時にファイルに加える変更を構成します。*

XML ファイルの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューに XML ファイル リファレンスを追加すると、その XML ファイルの場所を指定する必要があります。XML ファイルがインストールの一部の場合、その場所は、[ファイルとフォルダー] ビューでプロジェクトにベースの XML ファイルを追加したときに指定したインストール先に一致していなければなりません。

タスク ***ターゲットマシン上の XML ファイルの場所を指定するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、場所を指定する XML ファイルをクリックします。
3. **[全般]** タブをクリックします。

4. **[XML ファイルのインストール先]** 領域で、**[参照]** ボタンをクリックします。**[ディレクトリの参照]** ダイアログ ボックスが開きます。
5. **[インストール先ディレクトリ]** ボックスで、場所を選択します。
6. **[OK]** をクリックします。

[ターゲットマシン上の XML ファイルの保存場所を指定します] ボックスに指定したインストール先が追加されます。

XML ファイルの変更のリファレンスを機能に関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューに XML ファイル リファレンスを追加すると、そのリファレンスの新しいコンポーネントが自動的に作成されます。コンポーネントは、またデフォルトで、機能一覧の最上位にある機能にも関連付けられます。この関連付けを変更する場合は、[XML ファイルの変更] ビューを使用します。

メモ: *XML ファイルのリファレンスを追加したとき、プロジェクトに機能が存在しない場合、新しい機能を作成することができる [新しい機能の作成] ダイアログ ボックスが表示されます。*

タスク

プロジェクトで XML ファイルの変更のリファレンスを機能に関連付けるには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、コンポーネントを機能に関連付ける XML ファイルをクリックします。
3. **[全般]** タブをクリックします。
4. **[XML ファイルが属する機能を選択します]** ボックスで、選択した XML ファイルの変更のリファレンスを含める機能のチェック ボックスを選択します(複数可)。選択する機能は、[ファイルとフォルダー] ビューでプロジェクトに追加したベースの XML ファイルを含む機能を選択することもできますし、別の機能を選択することもできます。

エンドユーザーが XML ファイル変更を含む機能のインストールを選択した場合、その XML ファイルの変更が実行時に実行されます。

ルート要素の追加

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューで新しい XML ファイルのリファレンスを追加するとき、ルート要素を追加することができます。XML ファイルに含めることができるルート要素は、1 つだけですので注意してください。

タスク ***ルート要素を XML ファイルに追加するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、新しいルート要素を追加する XML ファイルを右クリックして、**[新しいルート要素]** をクリックします。

新しいルート要素が追加されます。右のペインにあるタブを利用して、設定を構成します。

ヒント: *ルート要素および 1 つ以上のレベルのサブ要素を一度に追加するには、ルート要素の名前とサブ要素をそれぞれスラッシュで区切って入力します。たとえば、***Root** *というルート要素と、***Root** *要素の下にある* **Sub** *要素と、***Sub** *要素の下の* **Sub2** *要素を追加するには、次のように入力します:*

Root/Sub/Sub2

エクスプローラーで自動的に展開されます。

新しい要素の追加

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

新しい要素は、XML ファイルのルート要素または任意の要素に追加することができます。

タスク ***新しい要素を追加するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、新しい要素を追加する XML ファイルを右クリックして、**[新しい要素]** をクリックします。

新しい要素が追加されます。要素の名前を必要に応じて変更し、右のペインに表示されているタブを使って、設定を構成します。

ヒント: *要素とサブ要素を [XML ファイルの変更] ビューのルート要素の下に追加するとき、要素名の完全 XPath 式を入力することができます。完全 XPath は、エクスプローラーで自動的に拡張します。*

*ルート要素および 1 つ以上のレベルのサブ要素を一度に追加するには、ルート要素の名前とサブ要素をそれぞれスラッシュで区切って入力します。たとえば、**Root** というルート要素と、**Root** 要素の下にある **Sub** 要素と、**Sub** 要素の下の **Sub2** 要素を追加するには、次のように入力します:*

Root/Sub/Sub2

各要素がそれぞれのノードと共にエクスプローラーで自動的に展開されます。

要素を .NET 構成ファイルに追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

web.config ファイルで製品について .NET 構成の設定を定義する場合、これらの設定を [XML ファイルの変更] ビューで指定することができます。

タスク ***要素を .NET 構成ファイルに追加するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、構成要素を追加する XML ファイルを右クリックし、**[定義済み要素の追加]** をクリックします。次いで、**[.NET 構成ファイル]**、**[Web 構成ファイル]** と順番にポイントし、表示された一連の設定をポイントして、適切なコマンドをクリックします。

エクスプローラーに、新しい要素が追加されます。

.NET 構成ファイルの異なる要素および属性に関する詳しい参照情報は、MSDN Web サイトの「Configuration File Schema (構成ファイル スキーマ)」(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/gngrfnetframeworkconfigurationfileschema.asp) をご覧ください。

属性を要素に追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルで、属性を要素に追加して、属性を実行時に作成するかどうか、もしくは、削除するかどうか、または、実行時に属性値を既存の値に追加するかどうかを指定することができます。インストールまたはアンインストール、もしくはその両方で実行するタスクをスケジュールすることもできます。

追加した属性がターゲット マシン上にある XML ファイル内に既に存在するが、属性の値が [XML ファイルの変更] ビューで指定した値と異なる場合、その値はインストールの実行時に更新されます。

タスク

属性を XML 要素に追加するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、属性を追加する XML ファイルをクリックします。
3. **[全般]** タブをクリックします。
4. **[追加]** ボタンをクリックします。

新しい行が、属性テーブルに [属性]、[値]、[操作]、および [スケジュール] 列にデフォルト値が入った状態で追加されます。必要に応じて設定を変更します。それぞれの列についての詳しい情報は、「XML 要素の [全般] タブ」を参照してください。

ヒント: *コンポーネントがアンインストールされたとき XML ファイルがそのままターゲット システムに残るようにインストールを構成する場合、属性テーブルでインストール/アンインストールの属性の組み合わせを作成したほうが良い場合があります。*

たとえば、インストール中に **key** *属性を追加して、アンインストール中にその同じ* **key** *属性を削除する場合、* **key** *属性を持つ 2 つの行を属性テーブルに追加します。つまり、片方の行をインストールにスケジュールして、もう 1 つの行をアンインストールにスケジュールします。*

属性の値を編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、属性が実行時にターゲット マシンに存在する場合、その属性の値を編集することができます。属性を編集するには、まず属性を [XML ファイルの変更] ビュー内の要素に追加してから、その属性の値を適宜構成します。属性値の編集方法は、属性値を追加する方法と同じです。詳細については、「属性を要素に追加する」を参照してください。

実行時に、属性に異なる値が存在する場合、その値は [XML ファイルの変更] ビューで構成した値に置き換えられます。

コンテンツを要素に追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML 要素に、テキスト コンテンツを含めることができます。たとえば、次の XML コードでは、**feature** は、要素 **product** のコンテンツです:

```
<product>feature</product>
```

タスク

コンテンツを [XML ファイルの変更] ビューで一覧表示されている要素の 1 つに追加するには、以下の手順に従います。

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、コンテンツに追加する要素を選択します。
3. **[詳細]** タブを選択します。
4. **[要素のコンテンツを設定する]** チェック ボックスを選択します。

5. **[コンテンツ]** ボックスで、テキスト コンテンツを入力します。

   この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

XPath 式を使って、XML ファイル内の XML データを検索する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

実行時に、インストールが新しい要素または属性を XML ファイルに追加したり、または XML ファイルに他の変更を実行したりするために、MSXML は [XML ファイルの変更] ビューで定義された XPath 式を使って、XML ファイルの中を移動し、変更が必要な箇所を見つけます。

XPath 式の書き方については、次の Web サイトをご覧ください:

- World Wide Web Consortium (http://www.w3.org)
- W3 Schools (http://w3schools.com)

以下のセクションは、基本的な XPath 式を作成するときの例です。

例 1: 2 つの要素を追加する

例 1 では、インストール時に、次の XML 文書に太線が追加されます:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Biographies>
    <Book Author="Bill Smith" Copyright="2007">Bill's Great Biography</Book>
    <Book Author="John Smith" Copyright="2006">John's Great Biography</Book>
  </Biographies>
</Books>
```

これらの要素を追加するために、次の XPath 式が [XML ファイルの変更] ビューの Biographies ノードの下に追加されます:

```
Book[@Author="John Smith"]
Book[@Author="Bill Smith"]
```

次のスクリーン ショットは、[XML ファイルの変更] ビューにある設定です。

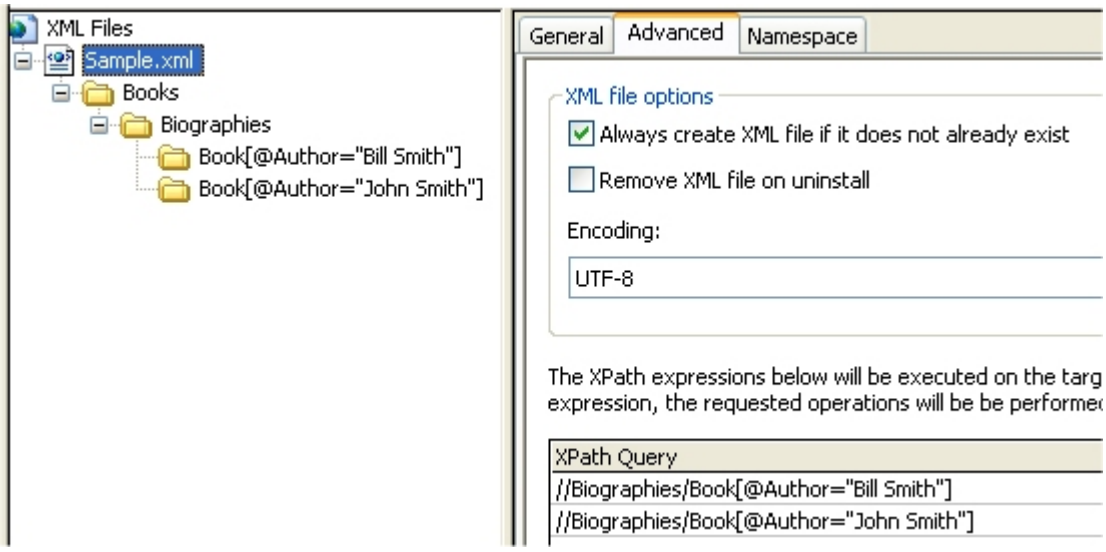


図 1: Sample.xml ファイルの設定

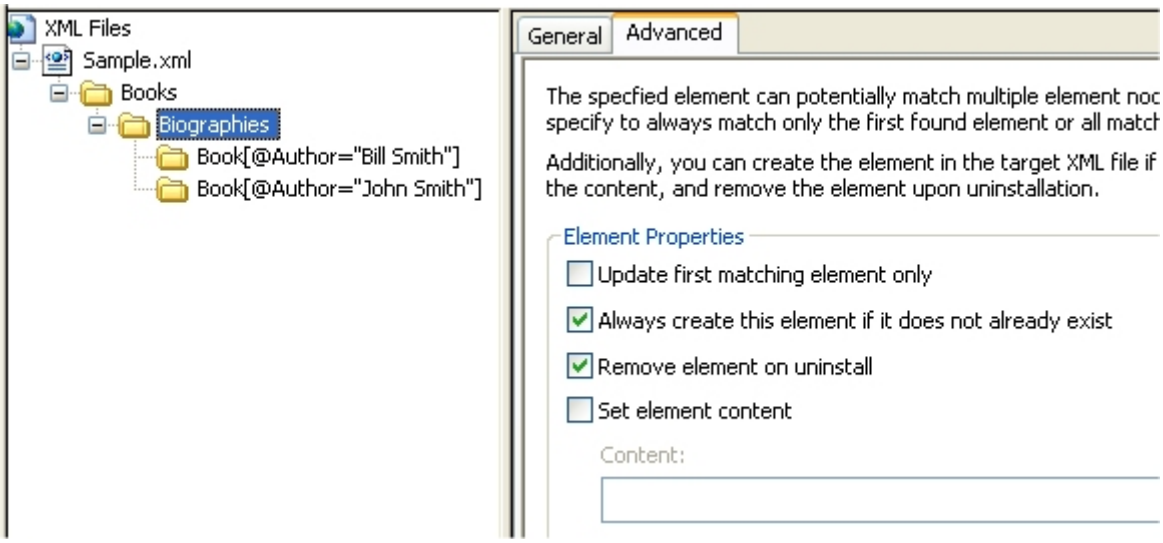


図 2: 追加される子ノードが含まれている Biographies ノードの設定

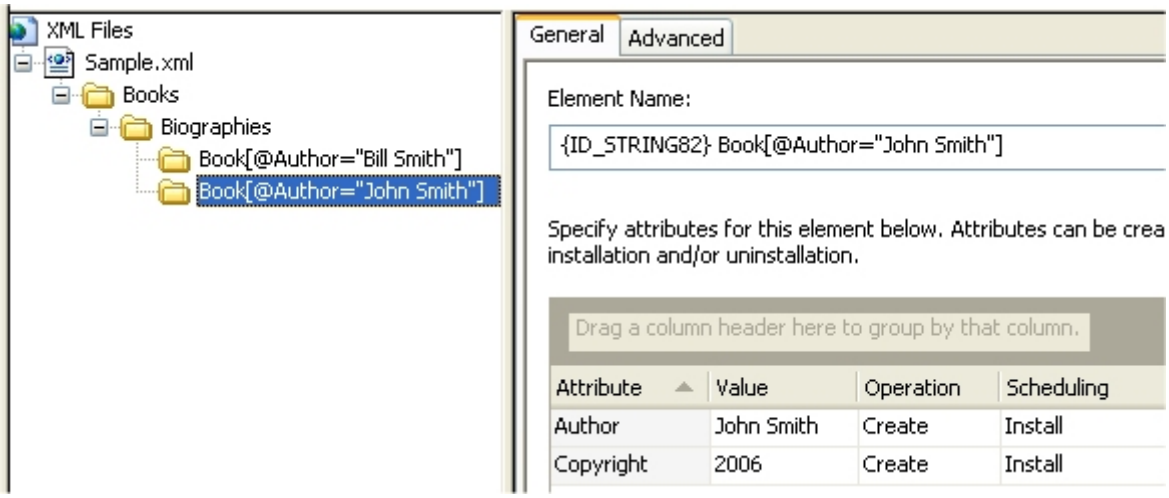


図 3: 追加される子ノードの 1 つの全般設定

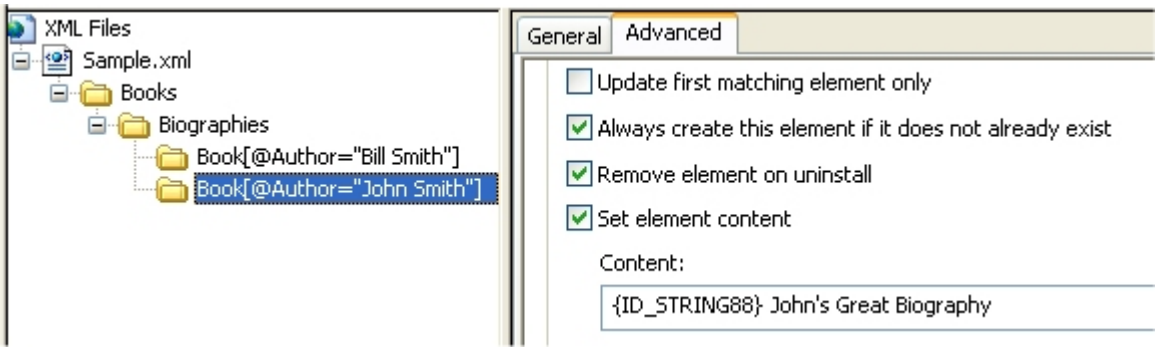


図 4: 追加される子ノードの 1 つの詳細設定

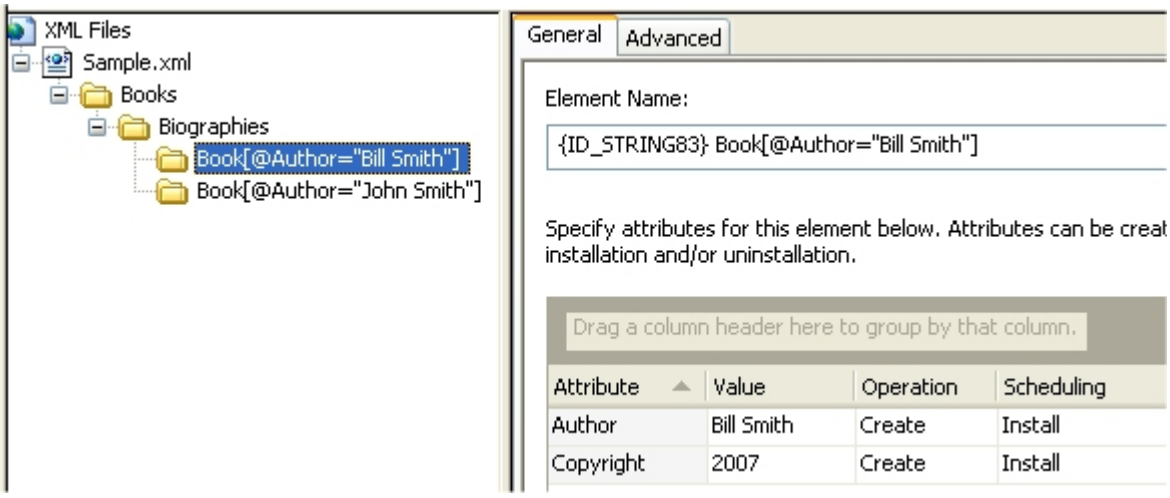


図 5: 追加される子ノードの 1 つの全般設定

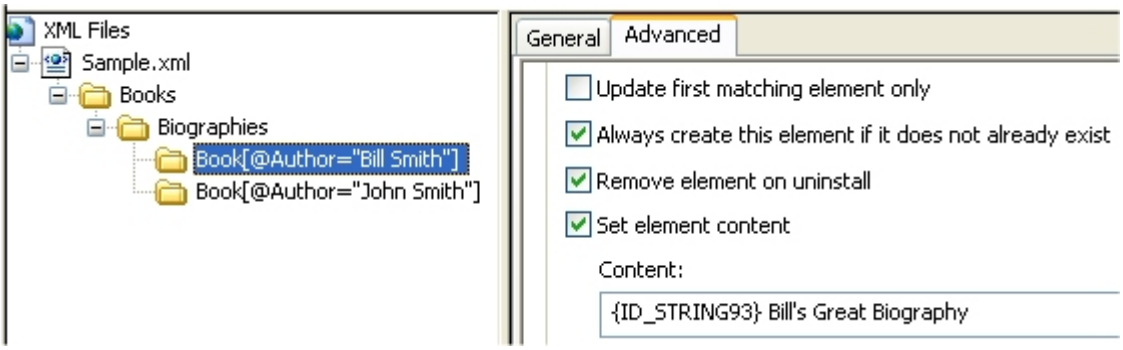


図 6: 追加される子ノードの 1 つの詳細設定

例 2: 要素に属性を追加する

例 2 では、インストール時に、太字属性が次の XML 文書に追加されます:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Biographies>
    <Book Author="Bill Smith" Copyright="2007" Publisher="Bill & John's Publish Co.">Bill's Great Biography</Book>
    <Book Author="John Smith" Copyright="2006" Publisher="Bill & John's Publish Co.">John's Great Biography</Book>
  </Biographies>
</Books>
```

次のスクリーン ショットは、[XML ファイルの変更] ビューにある設定です。

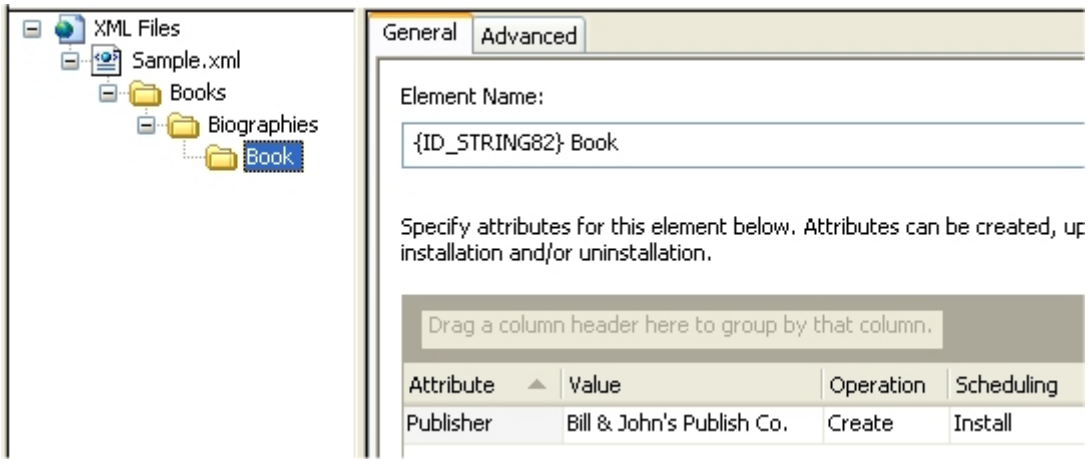


図 7: Book ノードの設定

例 3: 属性の値を更新する

例 3 では、インストール時に、2006 Copyright (次の XML 文書の中で太字で表示されています) という値がある属性が検索され、2007 Copyright という値で置き換えられます:

```
<?xml version="1.0" encoding="UTF-8"?>
<Books>
  <Biographies>
    <Book Author="Bill Smith" Copyright="2007" Publisher="Bill & John's Publish Co.">Bill's Great Biography</Book>
    <Book Author="John Smith" Copyright="2006" Publisher="Bill & John's Publish Co.">John's Great Biography</Book>
  </Biographies>
```

</Books>

これらの値を更新するために、次の XPath 式が [XML ファイルの変更] ビューの Biographies ノードの下に追加されます:

Book[@Copyright="2006"]

次のスクリーン ショットは、[XML ファイルの変更] ビューにある設定です。

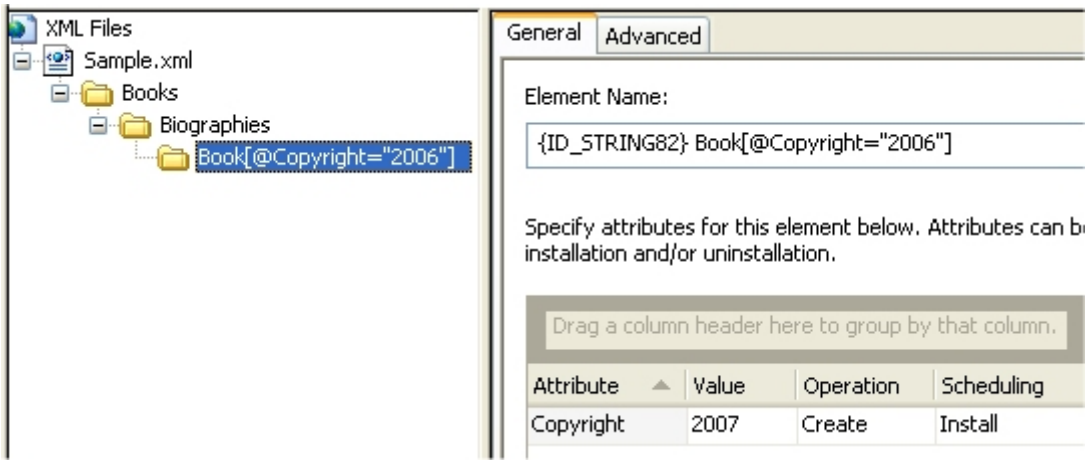


図 8: 更新される属性の全般設定

Windows Installer のプロパティを使用して、XML ファイルを動的に変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、およびトランスフォーム プロジェクトの場合、Windows Installer プロパティを使って、XML 要素、属性、属性値、および要素のコンテンツを指定することができます。実行時に、Windows Installer は **MsiFormatRecord** を使ってプロパティの値を解決し、それを XML ファイルのデータとして使用します。これにより、エンドユーザーがダイアログで入力したデータや、インストール時に判別された他の構成情報を製品の XML ファイルで使用できるようになります。

たとえば、インストールにエンドユーザーが SQL Server を選択することができる [SQL ログイン] ダイアログが含まれていると想定します。エンドユーザーが選択したサーバーの名前は、通常 IS_SQLSERVER_SERVER プロパティに格納されます。XML ファイルにエンドユーザーが選択した SQL サーバーの名前を含める場合、[XML ファイル変更] ビューで XML ファイルの変更を構成する際に、このプロパティをかっこで囲んで使用します (例、[IS_SQLSERVER_SERVER])。実行時に、このプロパティは Windows Installer によって自動的に関連付けられている値に置換されます。

ヒント: プロパティの値が、整形式の XML 文書になる場合のみ、要素、属性、または要素のコンテンツに Windows Installer プロパティを使用してください。プロパティの値が、XML 文書で構文エラーになる場合、製品またはインストールが期待通りに動作しない場合があります。

たとえば、要素の名前を [INSTALLDIR] (値がターゲット システム上のパスの Windows Installer) に設定した場合、実行時の結果は、円記号を含む要素名です。要素名に、円記号は使用できません。円記号がある要素名を含む XML 文書は無効です。

メモ: [XML ファイル変更] ビュー内から XML ファイルをテストする場合、XML データに使用している Windows Installer プロパティは適切な値で置き換えられません。テストに関する詳細については、「XML ファイルに加えられるインストール時の変更をテストする」または「XML ファイルに加えられるアンインストール時の変更をテストする」を参照してください。

例

以下は、エンドユーザーが [SQL ログイン] ダイアログで選択した SQL Server の名前を XML ファイル内のある要素のコンテンツとして使用するための手順です。[XML ファイル変更] ビューの要素、属性、属性値、要素のコンテンツのプロパティの設定でハードコード化されている値を代入することができます。このビューでプロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

タスク ***エンドユーザーが選択した SQL Server の名前を XML 要素のコンテンツとして使用するには、以下の手順に従います:***

1. SQL 接続と関連付けられている SQL スクリプトが未構成の場合、それらをまず構成し、サーバー プロパティ名の名前を決定します:
   a. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
   b. SQL 接続と SQL スクリプトを適宜追加します。手順については、「SQL サポートの構成」を参照してください。
   c. **[SQL スクリプト]** エクスプローラーで、SQL 接続をクリックします。
   d. **[詳細]** タブをクリックします。
   e. **"ターゲット サーバー プロパティ名"** 設定で選択された名前を確認してください。通常、デフォルト値は **IS_SQLSERVER_SERVER** です。
2. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
3. **[XML ファイル]** エクスプローラーで、コンテンツに追加する要素を選択します。
4. **[詳細]** タブを選択します。
5. **[要素のコンテンツを設定する]** チェック ボックスを選択します。
6. **[コンテンツ]** ボックスに、1e で選択したファイルを入力し、それをかっこで囲みます。例:

   [IS_SQLSERVER_SERVER]
7. リリースをビルドします。

InstallScript テキスト置換を使用して、XML ファイルを動的に変更する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトの場合、XML 要素、属性、属性値、および要素のコンテンツにテキスト置換文字列変数を利用することができます。実行時に XML ファイルの変更が発生すると、InstallScript ランタイム コードは **TextSubSubstitute** 関数を使って、文字列変数を適切な値で置き換えます。これにより、エンドユーザーがダイアログで入力したデータや、インストール時に判別された他の構成情報を製品の XML ファイルで使用できるようになります。

ヒント: *プロパティの値が、整形式の XML 文書になる場合のみ、要素、属性、または要素のコンテンツにテキスト置換を使用してください。プロパティの値が、XML 文書で構文エラーになる場合、製品またはインストールが期待通りに動作しない場合があります。*

たとえば、要素の名前を 1 つまたは複数のスペースを含むテキストで置換されるテキスト置換文字列変数に設定すると、エンドユーザーが製品またはインストールを使用するときに問題が発生する場合があります。要素名に、スペースは使用できません。したがって、スペースがある要素名を含む XML 文書は無効になります。

メモ: *[XML ファイル変更] ビュー内から XML ファイルをテストする場合、XML データに使用しているテキスト置換文字列変数は適切な値で置き換えられません。テストに関する詳細については、「XML ファイルに加えられるインストール時の変更をテストする」または「XML ファイルに加えられるアンインストール時の変更をテストする」を参照してください。*

例

以下は、エンドユーザーが [SQL ログイン] ダイアログで選択した SQL Server の名前を XML ファイル内のある要素のコンテンツとして使用するための手順です。[XML ファイル変更] ビューの要素、属性、属性値、要素のコンテンツのテキスト置換文字列変数の設定でハードコード化されている値を代入することができます。このビューで指定する文字列変数は山かっこで囲む必要があります; 例、**<MYPROPERTY>**。指定する文字列変数は大文字と小文字を区別します。

タスク

エンドユーザーが選択した SQL Server の名前を XML 要素のコンテンツとして使用するには、以下の手順に従います:

1. SQL 接続と関連付けられている SQL スクリプトが未構成の場合、それらを構成します:
   a. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
   b. SQL 接続と SQL スクリプトを適宜追加します。手順については、「SQL サポートの構成」を参照してください。
2. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
3. **[XML ファイル]** エクスプローラーで、コンテンツに追加する要素を選択します。
4. **[詳細]** タブを選択します。
5. **[要素のコンテンツを設定する]** チェック ボックスを選択します。

6. **[コンテンツ]** ボックスで、次のように入力します:

   <MYPROPERTY>

7. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

8. SQL Server コントロールを含む [SQL ログイン] ダイアログの OnSQLLogin イベントでダイアログ コードを見つけ、InstallScript 関数 **TextSubSetValue** の呼び出しを追加します。例:

   ```
   // ダイアログ ボックスを表示する (接続名なし)
   // ダイアログを入れ替える場合、コメントアウト
   nResult = SQLServerSelectLogin2( szConnection, szServer, szUser, szPassword, bWinLogin, szDB, FALSE, TRUE );
   TextSubSetValue("<MYPROPERTY>", szServer, TRUE);
   ```

9. リリースをビルドします。

要素内で予約されている文字 (<、>、&、'、") を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML 要素内で、"小なり" 記号 (<) などの予約されている文字を使用すると、MSXML パーサーが、実行時にターゲット システムでそれを定義済みエンティティ (<) に変換します。このエンティティの代わりに "小なり" 記号を使用すると、XML パーサーが新しい要素の始まりとして解釈し、それが無効な XML コードと見なされるため、エラーが発生する原因となります。結果の XML ファイルを Internet Explorer のようなブラウザーで開くと、エンティティではなく、この文字が、XML 要素内で表示されます 。

次のテーブルは、予約されている XML 文字と、それらに対応するエンティティの一覧です。テーブルにはまた、それぞれの予約されている文字について、実行時にエンティティによって置換されるかどうかが示されています。

テーブル 8 ・ 予約されている文字と定義済みエンティティ

文字	エンティティ	説明	注
<	<	より小さい	この文字は、XML 要素の始まりを示すために予約されているため、XML 要素のコンテンツとして使用することはできません。 この文字は、実行時に自動的にエンティティによって置換されます。
>	>	より大きい	この文字は、実行時に自動的にエンティティによって置換されます。

テーブル 8・予約されている文字と定義済みエンティティ (続き.)

文字	エンティティ	説明	注
&	&	And	この文字は、エンティティの始まりを示すとき以外は、XML 要素のコンテンツとしては使用できません。 この文字が、XML 要素のコンテンツとして使用されているときに、エンティティの始まりを示すものではない場合、実行時にそのエンティティによって自動的に置換されます。
'	'	アポストロフィ	この文字は、実行時に自動的にエンティティによって置換されません。
"	"	引用符	この文字は、実行時に自動的にエンティティによって置換されません。

XML ファイルで名前空間を使用する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML 名前空間を利用して、要素名の競合を避けることができます。InstallShield で XML ファイルの変更を指定するとき、XML ファイルで宣言する名前空間のマッピングを指定して、任意のファイルの要素について、名前空間プレフィックスを指定することができます。

***ヒント・** XML 設定のインポート ウィザードを使用して、XML ファイルを [XML ファイルの変更] ビューへインポートすると、ファイルに宣言された名前空間がインポートされます。*

XML ファイルに名前空間のマッピングを宣言する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューで、XML ファイルに名前空間のマッピングを宣言することができます。

タスク ***XML ファイルに名前空間のマッピングを宣言するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、名前空間を含める XML ファイルをクリックします。
3. **[名前空間]** タブをクリックします。
4. テーブル内の行をクリックして、新しい名前空間を追加します。
5. **[プレフィックス]** 列で、対応する名前空間に関連付けられているすべての要素に使用されるプレフィックスを入力します。
6. **URL**1 列で、URL、またはインターネット リソースを識別する文字列を入力します。

名前空間プレフィックスを要素に追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルに名前空間のマッピングを宣言すると、そのファイル内の要素を、その要素に対応するプレフィックスを追加することで、名前空間に関連付けることができます。

タスク ***名前空間プレフィックスを要素に追加するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、プレフィックスを追加する要素を右クリックして、以下のいずれかを実行します。
   - プレフィックスを選択した要素にのみ追加する場合、**[名前空間プレフィックス]** をポイントしてから、適切な名前空間マッピングをクリックします。
   - プレフィックスを要素とそのすべてのサブ要素に追加する場合、**[名前空間プレフィックス (すべてのサブ要素を含む)]** をポイントしてから、適切な名前空間マッピングをクリックします。

プレフィックスが、[XML ファイル] エクスプローラー内の要素 (および、該当する場合、そのすべてのサブ要素) に追加されます。

ヒント: *また、要素を右クリックして、[名前の変更] をクリックし、要素名に前にプレフィックスとコロン (:) を追加しても、名前空間プレフィックスを追加することができます。*

名前空間プレフィックスを属性に追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

XML ファイルに名前空間のマッピングを宣言すると、そのファイル内の属性を、その属性に対応するプレフィックスを追加することで、名前空間に関連付けることができます。

タスク ***名前空間プレフィックスを属性に追加するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、属性を含む要素をクリックします。
3. 右側のペインで **[全般]** タブをクリックします。
4. グリッドで、プレフィックスを追加する属性をダブルクリックしてから、属性名の始まりにカーソルを配置します。
5. 属性名の前にプレフィックスとコロン (:) を追加します。

名前空間プレフィックスを属性から削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *トランスフォーム*

タスク ***名前空間プレフィックスを属性から削除するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、属性を含む要素をクリックします。
3. 右側のペインで **[全般]** タブをクリックします。
4. グリッドで、プレフィックスを削除する属性をダブルクリックしてから、属性名の始まりにカーソルを配置します。
5. 属性名の前からプレフィックスとコロン (:) を削除します。

名前空間プレフィックスを要素から削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

タスク ***名前空間プレフィックスを要素から削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、プレフィックスを削除する要素を右クリックして、以下のいずれかを実行します。
   - プレフィックスを選択した要素からのみ削除する場合、**[名前空間プレフィックス]** をポイントして、**<なし>** をクリックします。
   - プレフィックスを要素とそのすべてのサブ要素から削除する場合、**[名前空間プレフィックス (すべてのサブ要素を含む)]** をポイントして、**<なし>** をクリックします。

プレフィックスが、[XML ファイル] エクスプローラー内の要素 (および、該当する場合、そのすべてのサブ要素) から削除されます。

XML ファイルから名前空間のマッピングを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

タスク ***名前空間マッピングを XML ファイルから削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、削除する名前空間属性を含む XML ファイルをクリックします。
3. **[名前空間]** タブをクリックします。
4. 削除する名前空間マッピングを含む行をクリックして、**[削除]** ボタンをクリックします。

テーブルから名前空間が削除されます。

XML ファイルに加えられるインストール時の変更をテストする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

InstallShield では、インストール全部をビルドして実行する手間をかけずに、[XML ファイルの変更] ビューでプロジェクトで構成された XML ファイルの変更のみをテストすることができます。インストールの変更をテストするとき、InstallShield はマシン上にある最新バージョンの MSXML を使用して、XML ファイルの解析と構成した XPath 式の実行を行います。MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。

メモ: *XML ファイルの変更に、Windows Installer のプロパティまたは InstallScript テキスト置換が含まれている場合、それらはテスト中、適切な値で置換されません。*

タスク ***XML ファイルに加えられるインストール時の変更をテストするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、テストする XML ファイルを右クリックして、**[XML ファイルのインストール時の変更のテスト]** をクリックします。**[テスト XML ファイルの選択]** ダイアログ ボックスが開きます。
3. **[ファイル名]** ボックスで、インストール時の変更を適用するターゲット ファイルを選択するか、または、新しいターゲット ファイル名と場所を指定します。
   - インストールの一部としてインストールされる XML ファイルを変更する場合、そのファイルのコピーを選択します。(XML のインストール時の変更をテストしたとき、XML ファイルに変更が加えられるため、インストール内のオリジナルのファイルは、選択しないでください。)
   - ターゲット システムに既に存在する XML ファイルを変更する場合、そのファイルのコピーを選択します。
   - XML ファイルが実行時に存在しなく、且つ、それがインストールの一部としてインストールされない場合に何が起こるかをテストする場合、新しいファイル名と場所を指定します。

   デフォルトのファイル名は、ステップ 2 で右クリックしたテスト ファイルの名前です。
4. **[開く]** をクリックします。

ターゲットファイルが既に存在する場合、テスト ファイルからの変更がターゲット ファイルに適用されます。指定したターゲット ファイルが存在しないときに、**[XML ファイルが存在しない場合、常に作成]** チェック ボックスが、その XML ファイルについて [詳細] タブで選択されている場合、ファイルが作成され、テスト ファイルからの変更が適用されます。

出力ウィンドウの [結果] タブで、インストール テストについての詳細が表示されます。詳細には、テスト ファイルへのハイパーリンクが含まれています。

ヒント: *テストの実行中に、ターゲット ファイルがブラウザーで開かれている場合、ブラウザーをリフレッシュして、場合によって、テスト変更を確認する必要があります。*

XML ファイルに加えられるアンインストール時の変更をテストする

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューで構成した XML ファイルのインストール時の変更をテストしたあと、アンインストール時に発生する XML ファイルの変更をテストすることもお勧めします。このテストを行うことにより、構成した変更が、アンインストール時に期待通りに動作するかどうかを確認できます。

アンインストールの変更をテストするとき、InstallShield はマシン上にある最新バージョンの MSXML を使用して、XML ファイルの解析と構成した XPath 式の実行を行います。MSXML が XPath 式に一致する XML ファイルの領域を検出したとき、[XML ファイルの変更] ビューで構成された変更が加えられます。

メモ: *XML ファイルの変更に、Windows Installer のプロパティまたは InstallScript テキスト置換が含まれている場合、それらはテスト中、適切な値で置換されません。*

タスク ***XML ファイルに加えられるアンインストール時の変更をテストするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、テストする XML ファイルを右クリックして、**[XML ファイルのアンインストール時の変更のテスト]** をクリックします。**[テスト XML ファイルの選択]** ダイアログ ボックスが開きます。
3. **[ファイル名]** ボックスで、アンインストール時の変更を適用するターゲット ファイルを選択します。

   デフォルトの値は、インストール時のテストを最後に行ったファイルの名前です。
4. **[開く]** をクリックします。

[XML ファイルの変更] ビューで構成されたアンインストールの変更がテスト ファイルに適用されます。

出力ウィンドウの [結果] タブで、アンインストール テストについての詳細が表示されます。詳細には、テスト ファイルへのハイパーリンクが含まれています。アンインストール時に削除されるように XML ファイルを構成すると、そのファイルが存在しなくなる可能性があるため、ハイパーリンクが動作しない場合があります。

ヒント: *テストの実行中に、ターゲット ファイルがブラウザーで開かれている場合、ブラウザーをリフレッシュして、場合によって、テスト変更を確認する必要があります。*

[XML ファイルの変更] ビューから要素または XML ファイルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

タスク ***[XML ファイルの変更] ビューから要素または XML ファイルを削除するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[XML ファイルの変更]** をクリックします。
2. **[XML ファイル]** エクスプローラーで、削除する XML ファイルまたは XML 要素を右クリックし、**[削除]** をクリックします。

XML ファイルを削除すると、XML ファイルに関連付けられたコンポーネントもファイルと共に削除されますというメッセージが表示されます。

テキスト ファイルの変更

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

InstallShield では、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。

[テキスト ファイルの変更] ビューでは、テキスト ファイルに行う変更を定義できます。このビューでは以下の処理が可能です:

- プロジェクトに 1 つ以上のテキスト変更セットを追加する。テキスト変更セットは、実行時に検索する 1 つ以上のテキスト ファイルへのリファレンスです。
- テキスト変更セットに 1 つ以上のテキスト変更アイテムを追加する。テキスト変更アイテムは、検索/置換の基準を識別します。

タスク ***テキスト ファイルの変更を構成するには、以下の手順に従います。***

1. テキスト 変更セットを作成する
2. テキスト ファイルの変更を指定する。

実行時にテキスト ファイルを変更する方法については、ドキュメントのこのセクションを参照してください。

テキスト 変更セットを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

テキスト ファイルの変更を構成するときの最初の手順は、編集するファイル (複数可) への参照を作成することです。この参照は、*[テキスト変更セット]* と呼ばれます。ファイルのフォーマットとしては、非バイナリファイル (たとえば .txt、.htm、.xml、.config、.ini、または .sql) のどれでも可能です。ファイルはインストールの一部 (つまり、[ファイルとフォルダー] ビューでプロジェクトに追加したファイル)、またはターゲット システム上に既存するファイルでも構いません。

メモ: *各テキスト変更セットは、必ずプロジェクト内のコンポーネントに関連付ける必要があります。このため、テキスト変更セットを作成する前に、プロジェクトに 1 つ以上のコンポーネントが必要です。テキスト変更リファレンスを作成するときにコンポーネントがなかった場合は、[新しいコンポーネントの作成] ダイアログ が表示され、ここでコンポーネントを作成することができます。*

タスク

実行時に変更する 1 つ以上のテキスト ファイルを参照するテキスト変更セットを作成するには、以下の手順に従います:

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーを右クリックして、**[変更セットの追加]** をクリックします。

   InstallShield は、デフォルト名を持つ新しい変更セットを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはプロジェクトに含まれる様々な変更セットを区別するのに内部で使用される名前です。
4. 右側のペインで、変更セットの設定を構成します。各設定の詳細については、「変更セットの設定」を参照してください。

テキスト ファイルへのリファレンスを作成してから、その設定を構成した後、次の検索/置換基準を指定するステップに進みます。

ヒント: *Windows Installer パブリック プロパティを使って、検索に含める/除外するテキスト ファイルの名前を指定することができます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別され*

た他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。

テキスト ファイルの変更を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

編集するテキスト ファイルを指定した後、テキスト ファイルの変更を構成するための次の「検索/置換基準を指定する」ステップに進みます。各基準セットは、*変更アイテム* と呼ばれます。

タスク ***テキスト ファイルに行う変更を指定するには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーで、定義を行うてテキストの変更を含む変更セット アイテムを右クリックしてから、**[変更の追加]** をクリックします。

   InstallShield は、デフォルト名を持つ新しい変更アイテムを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはプロジェクトに含まれる様々な変更アイテムを区別するのに内部で使用される名前です。
4. 右側のペインで、変更の設定を構成します。各設定の詳細については、「変更セットの設定」を参照してください。

テキスト ファイルで追加文字列を変更するには、このビューで変更セットに追加の変更アイテムを 1 文字列につき 1 アイテム追加します。

ヒント: *Windows Installer パブリック プロパティを使って、検索文字列と置換文字列を指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。*

テキスト ファイルの変更が行われる順番を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

[テキスト ファイルの変更] ビューで一覧されている順番で、ターゲット システム上のテキスト ファイルに変更が行われます。

タスク ***テキスト ファイルが実行時に変更される順番を変更するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーで、変更セット アイテムまたは変更アイテムの 1 つ右クリックして、**[上に移動]** または **[下に移動]** をクリックします。

すべてのテキスト ファイルの変更が正しく並べ替えられるまで最後のステップを繰り返します。

Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

Windows Installer プロパティを使って、検索/変更を行うテキスト 文字列を指定します。プロパティを使って、検索に含める、または検索から除外するテキスト ファイルを指定することもできます。

実行時に、Windows Installer は **MsiFormatRecord** を使ってプロパティの値を解決し、その値を使ってテキスト ファイルを変更します。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。

例

以下は、エンド ユーザーがインストール中に、実行時に XML ベースの web.config ファイルに書き込む IP アドレスを指定できるようにするための手順を説明します。web.config ファイルが製品と共に INSTALLDIR にインストールされていて、以下のような XML を含む:

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
   <appSettings>
      <add key="IP アドレス" value="デフォルト" />
   </appSettings>
</configuration>
```

太字表示のデフォルト値は、エンド ユーザーが入力する IP アドレスで置換します。

[テキスト ファイルの変更] ビューにある次の変更セットの設定では、ハードコード化された値をプロパティで置換できます:

- 含めるファイル
- 除外するファイル

[テキスト ファイルの変更] ビューで、次の変更アイテムの設定にはプロパティも使用できます:

- 検索する文字列
- 置換後の文字列
- テキストの挿入

これらの設定のいずれかでプロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

手順のステップ 4 はプロジェクトの種類によって若干異なります。これは、基本の MSI インストールのユーザー インターフェイスは Windows Installer が制御し、InstallScript MSI インストールのユーザー インターフェイスは InstallScript エンジンが制御するためです。

タスク ***エンドユーザーが IP アドレスを指定できるようにするには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. インストールが検索を行うファイルを識別する、変更セット アイテムを追加および構成します:
   a. **[テキスト ファイルの変更]** エクスプローラーを右クリックして、**[変更セットの追加]** をクリックします。

      InstallShield が、新しい変更セット アイテムを追加します。ステップ 2b から 2d までは、右側のペインに表示される設定を構成する方法を説明します。

   b. **"ターゲット フォルダー"** 設定で、**[INSTALLDIR]** ディレクトリ プロパティを選択します。

   c. **"含めるファイル"** 設定に、次のように入力します:

      web.config

   d. 残りの設定のデフォルト値はそのままにします。
3. 検索/置換の基準を識別する変更アイテムを追加および構成します:

a. **[テキスト ファイルの変更]** エクスプローラーで、ステップ 2 で作成した変更セット アイテムを右クリックしてから、**[変更の追加]** をクリックします。

   InstallShield が、新しい変更アイテムを追加します。ステップ 3b から 3e までは、右側のペインに表示される設定を構成する方法を説明します。

b. **"アクション"** 設定で **[置換]** を選択します。

c. **"検索する文字列"** 設定に、次のように入力します:

   <add key="IP アドレス" value="デフォルト"

d. **"置換後の文字列"** 設定に、次のように入力します:

   <add key="IP アドレス" value="[MYPROPERTY]"

e. 残りの設定のデフォルト値はそのままにします。

4. ダイアログでプロパティを使用します。この部分の手続きは、使用しているプロジェクトの種類によって異なります。

   - 基本の MSI プロジェクトの場合:

     a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

     b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、User Name コントロールを含めるダイアログの下にある言語をクリックします。代わりに、新しいダイアログを追加することもできます。

     c. **Edit Field** コントロールをダイアログに追加し、**Property** プロパティを次にように設定します:

        MYPROPERTY

   - InstallScript MSI プロジェクトの場合:

     a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

     b. User Name コントロールを含めるダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、Windows Installer API 関数 **MsiSetProperty** の呼び出しを追加します。たとえば、エンド ユーザーがプロジェクトに追加されている SdShowDlgEdit1 ダイアログにある編集ボックスに IP アドレスを入力できるようにするには、次のコード行で示されているように、**MsiSetProperty** の呼び出しを追加します。

        ```
        Dlg_SdShowDlgEdit1:
            nResult = SdShowDlgEdit1 (szTitle, szMsg, szField1, svEdit1);
            MsiSetProperty (ISMSI_HANDLE, "MYPROPERTY", svEdit1);
            if (nResult = BACK) goto Dlg_SdWelcome;
        ```

5. リリースをビルドします。

ヒント: *複数言語プロジェクトを作成する場合で、インストールが使用する言語に基づいて異なる値を持つプロパティを使用するには、ローカライズ可能なプロパティを使います。詳細については、「ローカライズ可能なプロパティを作成する」を参照してください。*

MsiFormatRecord (Windows Installer ヘルプ ライブラリ)

ANSI テキスト ファイルを開く時に使用するコード ページを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

インストールが ANSI テキスト ファイルを開いて、[テキスト ファイルの変更] ビューで構成された変更を行うとき、インストールはプロジェクトの ISSearchReplaceSet テーブルにある CodePage 列で指定されたコードページを使用します。この列のデフォルト値は数値 0 です。これは、CP_ACP コード ページで、ターゲット システム上で、システム Windows ANSI コード ページとして現在構成されているものです。[ダイレクト エディター] ビューを使って、この値を特定のコード ページでオーバーライドできます。

CodePage 列は 2 バイト (署名付き) です。32,767 よりもい大きい CodePage 値を提供する場合、2 の補数で指定します。

例:

名前	識別子	CodePage
UTF-8	65001	-535
UTF-7	65000	-536
Windows-1252	1252	1252

タスク

ANSI テキスト ファイルを開く時に使用するデフォルトのコード ページをオーバーライドするには、以下の手順に従います:

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]**エクスプローラーで、ISSQLDBMetaData テーブルをクリックします。InstallShield の右側に設定が表示されます。
3. グリッドで、コード ページを構成する変更セット アイテムに対応する行を見つけます。

   このグリッドの ISSearchReplaceSet 列には、**[テキスト ファイルの変更]** ビューの **[テキスト ファイルの変更]** エクスプローラーで利用可能な変更セットの名前がすべて表示されます。
4. 適切な行で **CodePage** フィールドの値を変更します。

[テキスト ファイルの変更] ビューから変更アイテムまたは変更セットを削除する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

この情報は InstallScript プロジェクトには適用しませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための 文字列関数 が含まれます。InstallScript プロジェクトで、これらの関数を使用できます。

あるテキスト ファイルに関して、特定の変更を今後行わないようにするには、[テキスト ファイルの変更] ビューからその変更アイテムを削除します。また、ファイルまたはテキスト ファイルのグループを今後検索しないようにするには、[テキスト ファイルの変更] ビューから対応する変更セットを削除します。

タスク ***[テキスト ファイルの変更] ビューから変更アイテムまたは変更セットを削除するには、以下の手順に従います。***

1. **[システム構成]** の下のビュー リストにある **[テキスト ファイルの変更]** をクリックします。
2. **[テキスト ファイルの変更]** エクスプローラーで、削除する変更セット アイテムまたは変更アイテムを右クリックしてから、**[削除]** をクリックします。

[テキスト ファイルの変更] ビューからアイテムが削除されます。

スケジュール タスク

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

ターゲット システムでインストールが実行された時、Windows タスク スケジューラーを通して、自動タスクを作成および構成することができます。スケジュールされたタスクは、インストールの一部であるファイルや、ターゲット システムに既に存在するファイルを起動することができます。

スケジュール タスクの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

メモ: *それぞれのスケジュール タスクは、必ずプロジェクト内のコンポーネントに関連付ける必要があります。したがって、スケジュール タスクを追加する前に、プロジェクトにコンポーネントが最低 1 つ存在している必要があります。スケジュール タスクを追加するときにコンポーネントが存在しない場合は、[新しいコンポーネントの作成] ダイアログ ボックスが表示され、ここでコンポーネントを作成することができます。*

タスク

プロジェクトにスケジュール タスクを追加するには、以下の手順を実行します。

1. **[システム構成]** の下のビュー リストにある **[スケジュール タスク]** をクリックします。
2. **[スケジュール タスク]** エクスプローラーを右クリックして、**[スケジュール タスクの追加]** をクリックします。InstallShield は、デフォルト名を持つ新しいタスクを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはプロジェクトに含まれる様々なスケジュール タスクを区別するのに内部で使用される名前です。
4. 右側のペインで、タスクの設定を構成します。各設定の詳細については、「スケジュール タスクの設定」を参照してください。

Windows Installer プロパティを使って、スケジュール タスクをダイナミックに構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

Windows Installer プロパティを使って、スケジュール タスクによって起動されるファイルを実行するために必要なアカウント情報などの情報を指定することができます。

実行時に、Windows Installer は **MsiFormatRecord** を使って、プロパティの値を解決し、その値を使ってスケジュール タスクを構成します。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のスケジュール タスクが作成される時に使用できるようになります。

例

以下の手順では、インストール時に、エンドユーザーに、どのようにスケジュール タスクの実行に必要なアカウントとパスワードを指定させるかが示されています。プロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

手順のステップ 4 はプロジェクトの種類によって若干異なります。これは、基本の MSI インストールのユーザー インターフェイスは Windows Installer が制御し、InstallScript MSI インストールのユーザー インターフェイスは InstallScript エンジンが制御するためです。

タスク ***エンドユーザーに、スケジュール タスクのアカウントとパスワード情報を指定させるには、以下の手順に従います:***

1. **[システム構成]** の下のビュー リストにある **[スケジュール タスク]** をクリックします。
2. **[スケジュール タスク]** エクスプローラーで、構成するタスクを選択します。
3. **"別のユーザーとして実行"** 設定に、次のように入力します:

   [DOMAINNAME]¥[USERNAME]
4. **"パスワード"** 設定に、次のように入力します:

   [PASSWORD]
5. ダイアログでプロパティを使用します。この部分の手続きは、使用しているプロジェクトの種類によって異なります。

   - 基本の MSI プロジェクトの場合:

     a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

     b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、Domain Name、User Name、および Password コントロールを含めるダイアログの下にある言語をクリックします。代わりに、新しいダイアログを追加することもできます。

     c. **Edit Field** コントロールをダイアログに追加し、**Property** プロパティを次にように設定します:

     DOMAINNAME

     d. **USERNAME** と **PASSWORD** プロパティに対し、ステップ 5c を繰り返します。

   - InstallScript MSI プロジェクトの場合:

     a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

     b. Domain Name、User Name、および Password コントロールを含めるダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、Windows Installer API 関数 **MsiSetProperty** の呼び出しを追加します。たとえば、エンド ユーザーがプロジェクトに追加されている **SdShowDlgEdit3** ダイアログにあ

る編集ボックスに情報を入力する場合、次のコード行で示されているように、**MsiSetProperty** の呼び出しを追加します。

```
Dlg_SdShowDlgEdit3:
    nResult = SdShowDlgEdit3
        (szTitle, szMsg, szField1, szField2, szField3, svEdit1, svEdit2, svEdit3);
    MsiSetProperty (ISMSI_HANDLE, "DOMAINNAME", svEdit1);
    MsiSetProperty (ISMSI_HANDLE, "USERNAME", svEdit2);
    MsiSetProperty (ISMSI_HANDLE, "PASSWORD", svEdit3);
    if (nResult = BACK) goto Dlg_SdWelcome;
```

6. リリースをビルドします。

スケジュール タスクの削除

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

タスク ***スケジュール タスクをプロジェクトから削除するには、以下の手順を実行します。***

1. **[システム構成]** の下のビュー リストにある **[スケジュール タスク]** をクリックします。
2. **[スケジュール タスク]** エクスプローラーで、削除するインスタンスを右クリックして **[削除]** をクリックします。

タスクがプロジェクトから削除されます。

Windows サービスのインストール、制御、および構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows サービスは、ログイン済みのユーザーが居ない場合でも、Windows ベースのシステム上で様々なシステムタスクを管理するためにバックグラウンドで実行される実行可能ファイルです。サービスは実行可能ファイルですが、これはサービスとして設計されたものであり、任意の実行可能ファイルをサービスとして使用することはできません。Windows サービスは、システム起動時に毎回実行することも、必要に応じてオンデマンドで実行することも可能です。InstallShield を使って、新しい Windows サービスをインストール、または既存のサービスを構成することができます。Windows で提供されているサービス管理ツールを使って、システムにインストール済みのサービスを参照および構成できます。

[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを構成できます。また、このビューを使って、Windows Installer 5 で提供されている拡張サービスのカスタマイズ オプションを構成することもできます。[サービス] ビューの他にも、[コンポーネント] ビューまたは [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [サービス] 領域を使うこともできます。これらの領域のいずれかを使ってサービス情報を構成した場合、対応する領域も自動的に更新されます。

サービスに関する作業について以下の点についてご注意ください:

- 開始、停止、削除、または構成を行うサービスは、インストールまたはアンインストール中にターゲット システムに既存するものか、インストールの一部としてインストールされるものかを問いません。
- サービス実行可能ファイルは、そのコンポーネントのキー ファイルでなくてはなりません。詳細については、「コンポーネントのキーファイル」を参照してください。
- サービスのコンポーネントの "リモート インストール" 設定は [ローカルを優先] でなくてはなりません。詳細については、「コンポーネントの "リモート インストール" 設定を構成する」を参照してください。

ヒント: *[サービス] ビューの上部にある [ビュー フィルター] を使って、ビューで表示するサービス データを含むコンポーネントまたは機能を選択し、ビューで非表示にするサービス データを含むコンポーネントまたは機能を隠すことができます。ビューフィルターは、プロジェクトの機能、サブ機能、およびコンポーネントの階層をリスト表示します。*

タスク ***サービスをインストール、開始、停止、削除、アンインストール、または構成するには、以下の手順に従います:***

1. インストールがサービスをインストールする場合、プロジェクトに含まれるコンポーネントにサービス実行可能ファイルを追加して、それをコンポーネントのキーファイルとします。詳細については、「ファイルをコンポーネントに追加する」を参照してください。

   サービスがターゲット システム上に既存する場合、この手順は省略します。

2. **[システム構成]** の下のビュー リストにある **[サービス]** をクリックします。
3. **[サービス]** ノードを右クリックして、**[サービスの追加]** をクリックします。新しいサービスが追加されます。
4. サービスの新しい名前を入力するか、後で F2 キーを押して名前を変更します。

   ここで入力する名前は、サービスの [プロパティ] ダイアログ ボックスに表示される名前と一致しなくてはなりません。(インストール済みサービスのプロパティにアクセスするには: [サービス] 管理ツールで、サービスを右クリックしてから [プロパティ] を選択します。)
5. 追加したサービスを選択してから、右側のペインに表示される設定を必要に応じて構成します。各設定についての詳細は、「[サービス] ビュー」を参照してください。

このコンポーネントのキーファイルの各サービスについて、この手順を繰り返します。

メモ: サービスの設定を構成するにあたっては、サービスの技術的な詳細についてよく知っておく必要があります。

ヒント: 実行時にサービスの構成が失敗すると、インストールが失敗してエラーメッセージを表示することがあります。ICE 102 は、サービス関連の設定の一部を検証して、構成エラーの回避に役立ちます。したがって、リリースに対する検証を行うことが推奨されます。

実行時に既定のユーザー アカウントを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield では、ログオン ダイアログを使わずに、1 つまたは複数の Windows ユーザー アカウントをグループとして作成することができます。このサポートには、実行時に設定されるアカウントごとに、ユーザー アカウント、グループ、およびパスワードの 3 つの Windows Installer プロパティが必要です。

ユーザー アカウントを作成するためのプロパティ

実行にユーザー アカウントを作成するには、次のプロパティを使用します:

- ISNetApiLogonUsername—このプロパティの値を、インストール時に作成するユーザー アカウントに設定します。次のいずれかのフォーマットを使用します:
  - *マシン名¥ユーザー名*
  - *ドメイン名¥ユーザー名*
- ISNetApiLogonGroup—このプロパティの値を、ユーザー アカウントが属するグループに設定します。
- ISNetApiLogonPassword—このプロパティの値を、ユーザー アカウントに使用されるパスワードに設定します。

必須のプロパティとその値を構成するには、[プロパティ] ビューを使用します。その手順は、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

ユーザー アカウントを作成するためのガイドライン

複数のユーザー アカウントを作成するには、**ISNetApiLogonUsername** プロパティ値のそれぞれのエントリを角かっこで囲んだチルダ ([~]) を使って区切ります。たとえば、**Domain1** という名前のドメインに **User1** という名前のユーザーを作成し、**Machine2** という名前のマシン上で **User2** という名前のユーザーを作成する場合、次のフォーマットを使用します:

Domain1¥User1[~]Machine2¥User2

ISNetApiLogonUsername プロパティで指定したユーザー アカウントにはそれぞれ、**ISNetApiLogonGroup** プロパティで対応するグループと、ISNetApiLogonPassword プロパティで対応するパスワードが指定されている必要があります。これらの 3 つのプロパティで、同数のエントリが指定されていなかった場合、ランタイム エラーが発生します。

以下は、3 つの異なるユーザー アカウントを作成するときのサンプル プロパティとそれらの対応する値です:

テーブル 9 ・ ユーザー アカウントとグループのプロパティ

プロパティ名	プロパティ値
ISNetApiLogonUsername	Domain1¥User1[~]Domain1¥User2[~]Domain2¥User3
ISNetApiLogonGroup	Users[~]Users[~]Administrators
ISNetApiLogonPassword	Password1![~]Password1![~]Password1!

この例では、**User1** は、**Users** グループの一部で、**Domain1** ドメイン用に **Password1!** というパスワードが設定されています。**User2** は、**Users** グループの一部で、**Domain1** ドメイン用に **Password1!** というパスワードが設定されています。**User3** は、**Administrators** グループの一部で、**Domain2** ドメイン用に **Password1!** というパスワードが設定されています。

ISNetApiLogonUsername、**ISNetApiLogonGroup**、および ISNetApiLogonPassword プロパティは、フォーマット済みのプロパティです。このため、これらのプロパティの値には、異なるプロパティ、環境変数、ファイル パス、またはコンポーネント ディレクトリ パスが指定可能です。例:

- **[PROPERTYNAME]**—実行時に、指定されたプロパティの値を解決します。
- **[%EnvironmentVariable]**—実行時、指定された環境変数の値に解決されます。
- **[#FileKey]**—実行時に、File テーブルで指定されたファイル キーがあるファイルのファイルの完全パスに解決されます。
- **[$ComponentName]**—実行時に、指定されたコンポーネントのインストール場所に解決されます。

ユーザーごとのインストールとマシンごとのインストールの違い

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

2 つの Windows Installer プロパティおよび現在のユーザーの権限によって、製品のショートカットやレジストリ エントリなどの構成情報がターゲット マシン上で All Users プロファイルまたは現在のユーザーのプロファイルのどちらに格納されるべきかが決定されます。

- ALLUSERS によって、構成が格納される場所が判別されます。
- MSIINSTALLPERUSER は、Windows Installer によって、パッケージが現在のユーザーに対してのみインストールされることを示します。

  MSIINSTALLPERUSER プロパティは、Windows Installer 5 および Windows 7、または Windows Server 2008 R2 で使用できます。以前のバージョンの Windows Installer と Windows は、このプロパティを無視します。

プロパティ マネージャーを使って、プロジェクトに ALLUSERS と MSIINSTALLPERUSER プロパティを設定できます。また、つぎの方法を使ってこれらのプロパティを設定することもできます。

- コマンドラインを使用する
- カスタム アクションを使用する
- CustomerInformation および ReadyToInstall ダイアログを使用する

ALLUSERS、MSIINSTALLPERUSER、および Windows 7 または Windows Server 2008 R2

ALLUSERS プロパティが 1 に設定されていて、**MSIINSTALLPERUSER** に 1 が設定されている場合、Windows Installer は、ユーザーごとにインストールを実行します。

マシンごとのインストール時、Windows Installer は昇格された権限を必要とし、ファイルとレジストリ エントリがマシンごとの場所に配置します。ユーザー アカウント制御 (UAC) がターゲット システムで提供されている場合、マシンごとのインストールでは、通常、ユーザーの権限レベルに応じて、同意または認証情報を求めるプロンプトが表示されます。ユーザーごとのインストールでは、Windows Installer によって認証情報を求めるプロンプトは表示されず、ファイルとレジストリ エントリはユーザーごとの場所に配置します。

詳しい情報は、MSDN Web サイトの「Single Package Authoring」を参照してください。

Windows Vista 以降における ALLUSERS の効果

スクリプト内実行設定がシステム コンテキストで遅延実行になっているカスタム アクションは、Windows で LocalSystem アカウントに与えられた権限を使ってアクションを実行するときに使用されます。これは、Windows Installer がシステム コンテキストで実行されるためです。システム コンテキストで遅延としてマークされていないアクションは、ユーザーを偽装して実行され、インストールを起動するユーザーが所有する権限を持ちます。

ユーザーごとのインストール (ALLUSERS が設定されていないインストール) が実行されると、"システム コンテキストで遅延" のアクションは、通常の遅延または即時カスタム アクションと同じコンテキストで (ユーザーを偽装して) 実行されます。これにより、次のような状況で、カスタム アクションの実行時に問題が起きる可能性があります。

- Windows Installer インストールを起動するユーザーが管理者ではないとき、またはユーザーがインストールを Windows Vista 以降で実行していて、ユーザーが管理者グループに属し、ユーザーがデフォルトで管理者権限を持たないとき。
- カスタム アクションは、マシンのマシンごとの場所でリソース (Program Files フォルダーにあるファイル、HKEY_LOCAL_MACHINE のレジストリ キーまたは値) を変更しようと試みるとき。

これは、Windows XP または Windows の初期のバージョンでは問題にはならない可能性がありますが、デフォルトで Windows Vista 以降では、ユーザーは完全管理者権限は与えられません。したがって、ALLUSERS が設定されていない場合、"システム コンテキストで遅延" のアクションがユーザー偽装で実行されると、カスタム アクションは失敗する可能性があります。

この動作を防ぐには、[プロパティ マネージャー]で ALLUSERS を 1 に指定して、常にマシンごとのインストールを実行することをお勧めします。マシンごとのインストールは通常、ユーザーごとのインストールよりも管理が簡単です。

ALLUSERS のデフォルト値

すべての基本の MSI プロジェクトでは、デフォルトで、ALLUSERS プロパティは 1 に設定されています。管理者権限を使わないでユーザーごとにインストールできるようにインストールを構成する場合、ALLUSERS プロパティの値を変更するか、このプロパティをプロジェクトから削除したほうが良い場合があります。

ReadyToInstall ダイアログのデフォルトのコントロール

[一般情報] ビューの "[ユーザーごと] オプションの表示" 設定を使って、エンド ユーザーが製品をすべてのユーザー、または現在のユーザーのみにインストールするかを選択できるオプションを提供するかどうかを指定します。以下は、"All Users オプションの表示" 設定に選択できるオプションです:

- **いいえ**—InstallShield は関連プロパティを設定しません。実行時に、ReadyToInstall はエンド ユーザーが製品をインストールする方法を指定できるボタンを表示しません。これがデフォルトの値です。
- **はい** InstallShield は **ISSupportPerUser** プロパティを 1 に設定します。

実行時に以下の条件が True 評価されると、ReadyToInstall にはユーザーごとまたはマシンごとのボタンが表示されます:

- **ISSupportPerUser** プロパティが 1 である。
- ターゲット システムに Windows 7 以降、または Windows Server 2008 R2 以降が搭載されている。
- 製品がターゲット システムに既にインストールされていない。

ReadyToInstall ダイアログのボタンを使って、エンド ユーザーはプロジェクトをインストールする方法を指定できます。昇格された権限が必要な場合、[すべてのユーザー] ボタンにシールド アイコンが含まれます。エンド ユーザーが [ユーザーごと] ボタンを選択した場合、ALLUSERS プロパティが 2 に、MSIINSTALLPERUSER プロパティが 1 に設定されます。エンド ユーザーが [すべてのユーザー] ボタンを選択した場合、ALLUSERS プロパティが 1 に設定され、MSIINSTALLPERUSER プロパティは設定されません。

メモ: *[一般情報] ビューの "[ユーザーごと] オプションの表示" 設定に [いいえ] を選択しても、エンド ユーザーがインストールを実行するときに、コマンドラインから* MSIINSTALLPERUSER *を設定することを阻止することはできません。インストールがこれをサポートしない場合、起動条件を追加するか、その他の実行時のチェックを行って、これを回避することができます。*

CustomerInformationダイアログのデフォルトのコントロール

デフォルトで、すべての基本の MSI プロジェクトの CustomerInformation ダイアログは、エンドユーザーが製品をすべてのユーザーにインストールするか、または現在のユーザーのみにインストールするかを指定できるラジオ ボタン グループを表示しないようになっています。このダイアログについては、これが推奨される実装です。

タスク

エンド ユーザーが CustomerInformation ダイアログから ALLUSERS を設定できるようにするラジオ ボタン グループを表示するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** の下にある **CustomerInformation** ダイアログを展開して、**[動作]** をクリックします。CustomerInformation ダイアログのコントロール リストが表示されます。
3. 右下のペインの下部にある、**[条件]** タブをクリックします。右上のペインに、選択されたコントロールの条件が表示されます。
4. コントロール リストから、**DlgRadioGroupText** を選択します。右側のペインにコントロールの条件が表示されます。
5. 右上のペインで、値が 1 で、かつアクションが Hide になっている条件を含む行を右クリックし、**[削除]** をクリックします。

6. **RadioGroup** コントロールについて、ステップ 4 と 5 を繰り返します。

ALLUSERS Property (Windows Installer ヘルプ ライブラリ)

管理者権限なしに InstallScript インストールを実行する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallShield では、管理者権限を持たないエンドユーザーでも実行可能な InstallScript インストールを作成することができます。レジストリ エントリ、フォルダー、[スタート メニュー] のアイテムの適切な場所は、実行に動的に判別されます。InstallScript システム変数 ALLUSERS は、このようなインストールのキーとなります。つまり、インストールが管理者権限を持たないエンドユーザーによって実行された場合、ALLUSERS は FALSE の値で初期化され、スクリプトで新しい値を割り当てても効果はありません。

次の事項に注意してください。

- ファイルの自己登録には管理者権限が必要です (ファイルの自己登録は、"自己登録" 設定で [はい] が選択されているコンポーネントに含まれています)。
- 通常 InstallScript オブジェクトでは、ファイルを管理者権限が必要な場所にファイルをインストールまたは登録するため、権利者権限が必要です。
- DIFx ドライバーのインストールには、管理者権限が必要です。
- One-Click Install インストールのセットアップ プレーヤーのインストールには、管理者権限が必要です。
- Windows サービスのインストールには、管理者権限が必要です。
- リモート レジストリに接続するとき、管理者権限が必要です。
- 管理者またはパワーユーザー以外のエンドユーザーが、ALLUSERS が TRUE に設定された状態でインストールされた InstallScript インストールを保持しようと試みた場合、初期化中にエラー メッセージが表示されます。エラー メッセージでは、製品は管理者権限を持つユーザーによってインストールされており、製品の変更およびアンインストールには同様の権限が必要であることが示されます。

インストール動作のカスタマイズ

インストール作成の重要な要素は、それをエンドユーザーのニーズに合わせてカスタマイズすることです。ドキュメントの「インストール動作のカスタマイズ」セクションは、インストールの機能を拡張するのに役立つ InstallShield の様々な機能について説明します。たとえば、Windows Installer が直接サポートしていない機能を追加するカスタム アクションを作成するのに役立ちます。さらに、InstallScript ビューのスクリプト エディターで書いたスクリプトを呼び出すカスタム アクションを設定することができます。このセクションには、次のトピックが含まれています:

- InstallScript を使用する
- カスタム アクションを使用
- アクション テキストを使う
- シーケンスを定義する
- サポート ファイルを使用する

InstallScript を使用する

InstallScript の強力で使いやすい機能を活用することにより、インストールパッケージの機能を拡張できます。InstallScript ビューでは、スクリプト エディター ペインを使用して InstallScript コードを作成できます。

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

エディション: *InstallScript を使用する場合は* Setup.rul *というファイルがプロジェクトに含まれていなければなりません。InstallScript と InstallScript MSI プロジェクトにはデフォルトで* Setup.rul *ファイルが含まれています。基本の MSI プロジェクト、DIM、およびマージ モジュール プロジェクトで InstallScript カスタム アクションを使用する場合は、プロジェクトに* Setup.rul *ファイルを追加する必要があります。*

InstallScript とInstallScript MSI プロジェクトで、イベント ドリブン型 InstallScript を使用する

InstallScript と InstallScript MSI プロジェクトには、イベント ドリブン型スクリプトが含まれています。これらのプロジェクト タイプで InstallScript を使用すると、多くの関数がアンインストール用にログに記録されます。

カスタム アクションでの InstallScript の使用

基本の MSI、DIM、およびマージ モジュール プロジェクト

基本の MSI、DIM、またはマージ モジュール プロジェクトでは、カスタム アクションを使って、実行時に InstallScript を実行できます。これらのプロジェクト タイプでは、イベントドリブン型スクリプトはサポートされていません。

InstallScript MSI プロジェクト

InstallScript MSI プロジェクトでは、[実行] シーケンスで、デフォルトのイベント ハンドラーがインストールのニーズに応じて適切にスケジュールされていない場合に、InstallScript カスタム アクションを使って機能を拡張することができます。

カスタム アクションの作成と使用

タスク ***InstallScript カスタム アクションを作成して、インストールで実行するには、以下の手順を実行します。***

1. 既存の Setup.rul がない場合、InstallScript ビューでプロジェクトに Setup.rul を追加します。
2. エントリ ポイント関数を記述します。エントリ ポイント関数は、InstallScript MSI インストール プロジェクトの [ユーザー インターフェイス] シーケンスでは呼び出すことができないことに注意してください。
3. スクリプトをコンパイルします。
4. InstallScript 関数を呼び出すカスタム アクション を作成します。
5. InstallScript カスタム アクションを、シーケンス内に含める (InstallScript MSI、基本の MSI、およびマージ モジュール プロジェクトの場合) か、コントロール イベントとして実行 (基本の MSI、およびマージ モジュール プロジェクトの場合のみ) することにより呼び出します。
6. 必要に応じてデバッグします。

メモ: *他のカスタム アクションと同様、InstallScript カスタム アクションによってシステムに加えた変更は、パッケージをアンインストールするときに自動的に復元されません。アンインストーラーは、InstallScript カスタム アクションをログに記録したり、削除することがないため、カスタム アクションで加えたすべての変更をアンインストールするには、アンインストール用のカスタム アクションを記述する必要があります。*

ISSetup.dll の概要

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ISSetup.dll は、すべての InstallScript スクリプト ランタイム エンジンを含む C++ MSI DLL です。InstallScript MSI、基本の MSI、DIM、およびマージ モジュール プロジェクトでは、ISSetup.dll が InstallScript カスタム アクションを実行します。InstallScript プロジェクトでは、ISSetup.dll を必ず Disk1 フォルダーに配置します。

別のインストールによってターゲット マシン上に最新版の InstallScript スクリプト ランタイム エンジン がある場合でも、インストールは常にビルドに使われたエンジンと同じバージョンの InstallScript スクリプト ランタイム エンジンを利用します。

重要 *・次の事項に注意してください。*

- *ISSetup.dll を使用する場合、InstallScript MSI、基本の MSI、DIM、またはマージ モジュール プロジェクトに追加することができる InstallScript カスタム アクションの数は 1,000 までです。プロジェクトに InstallScript カスタム アクションを 1,000 以上含めると、ビルドエラーが発生します。*
- *InstallScript プロジェクトに使われる ISSetup.dll のバージョンは、InstallScript MSI、基本の MSI、DIM、およびマージ モジュール プロジェクトで使われるバージョンとは異なります。これらの 2 つのバージョンを置き換えることはできません。*

スクリプト ファイル

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

インストール プロジェクトを作成した場合、InstallShield では 2 つのスクリプト ファイルが作成され、[プロジェクト] フォルダーの [スクリプト ファイル] フォルダーに格納されます。

- Setup.rul はグローバル イベント ハンドラーと例外ハンドラーのために作成されます。
- FeatureEvents.rul は機能イベント ハンドラーのために作成されます。

元々これら 2 つのファイルは空白で、InstallShield の InstallScript ビューから選択しない限り、InstallShield が定義したデフォルトのイベント ハンドラーはこれらのファイルには含まれません。選択すると、適切なスクリプト ファイルに挿入されて InstallScript ビューのスクリプト ペインに表示されるので、そこで編集が可能です。

FeatureEvents.rul でデフォルトの機能のイベント ハンドラーコードを変更した場合、Setup.rulに次のステートメントを入れてインストールに変更を含める必要があります。

```
#include "FeatureEvents.rul"
```

InstallScript ファイルの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallScript を使用する場合は Setup.rul *というファイルがプロジェクトに含まれていなければなりません。InstallScript と InstallScript MSI プロジェクトにはデフォルトで* Setup.rul *ファイルが含まれています。基本の MSI プロジェクト、DIM、およびマージ モジュール プロジェクトで InstallScript カスタム アクションを使用する場合は、プロジェクトに* Setup.rul *ファイルを追加する必要があります。*

タスク ***プロジェクトに新規 InstallScript ファイルを追加するには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[新しいスクリプト ファイル]** を選択します。
3. ファイルに名前をつけます。

デフォルトでは、新規スクリプト ファイルに Setup.rull という名前が付けられます。Setup.rul が既にある場合は、新規ファイルが Setup*n* (*n* は連続番号) という名前で追加されます。ファイル名を右クリックして [名前の変更] をクリックすると、ファイル名を変更できます。

新しい スクリプト ファイルは Link To (リンク先) フォルダーに配置されます。プロジェクトを移動した場合、InstallShield はパス変数の使用を試みます。Link To 値を編集することはできません。

追加のヘッダー ファイル (.h files) やスクリプト ファイルを含めることもできます。上の手順を繰り返して新規スクリプト ファイルをプロジェクトに追加します。

InstallScript ファイルを開く

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

タスク ***編集するプロジェクト内のスクリプト ファイルを開くには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、開くファイルをクリックします。

スクリプト ファイルの挿入とインポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallShieldでは、複数のプロジェクトで InstallScript ファイル (.rul) および InstallScript ヘッダーファイル (.h) の再利用が可能です。InstallScript ビューで、プロジェクトにスクリプト ファイルを挿入したりインポートしたりすることができます。

- スクリプト ファイルを挿入すると、ファイルの現在の保存場所へのリンクが作成されます。
- スクリプト ファイルをインポートすると、プロジェクトのスクリプト ファイルを格納しているフォルダーにそのファイルがコピーされます。インポートしたスクリプト ファイルは、システム上またはリポジトリに保存することができます。

InstallShield は、これらのスクリプト ファイルの場所用に [パス変数] ビューで定義されたすべてのパス変数タイプをサポートします。ただし、ソース コード管理 ソフトウェアがパスを解決できるよう、対応するフォルダーの構成がソース コード データベースに確実に存在していることが重要です。

スクリプト ファイルの挿入

タスク ***スクリプト ファイルを挿入するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[スクリプト ファイルの挿入]** を選択します。**[開く]** ダイアログ ボックスが開きます。
3. 挿入する **InstallScript** ファイル (.rul) または InstallScript ヘッダーファイル (.h) を選択します。
4. **[開く]** をクリックします。

スクリプト ファイルのインポート

タスク ***スクリプト ファイルをインポートするには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[スクリプト ファイルのインポート]** を選択します。[InstallScript ファイルのインポート] ダイアログ ボックスが開きます。
3. 以下のいずれかを実行します。
   - **[リポジトリ アイテム]** ボックスで、プロジェクトに追加する InstallScript ファイル (.rul) または InstallScript ヘッダーファイル (.h) をクリックします。
   - インポートしたいスクリプト ファイルがリポジトリに格納されていない場合、**参照**ボタンをクリックしてファイルを選択します。
4. **[OK]** をクリックします。

スクリプトのコンパイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

インストール中に InstallScript コードを呼び出すには、事前にスクリプトをコンパイルする必要があります。

スクリプトのコンパイル時に InstallShield が検索するのは、Setup.rul という名前のファイルだけです。異なる名前のファイルを含めることは可能ですが、そのファイルを Setup.rul に含めるか、または #include プリプロセッサ ステートメントを使用したインクルード ファイルに含める必要があります。

タスク ***スクリプトをコンパイルするには、以下のいずれかを行います。***

- **[ビルド]** メニューで **[コンパイル]** をクリックします。
- ツールバー上の **[コンパイル]** ボタンをクリックします。
- CTRL +F7 を押す。

コンパイルの前に、InstallShield は、スクリプト ファイルに加えた変更をすべて保存します。エラーメッセージまたは警告メッセージなどのコンパイラの状態が、出力ウィンドウに表示されます。スクリプト中のエラーが見つかった行に移動するには、コンパイラ メッセージをダブルクリックします。

変更を加えてからスクリプト ファイルをコンパイルする場合、リリースをビルドし直す必要はありません。InstallShield では、リリースのビルド時には常にスクリプトが自動的にコンパイルされます。

スクリプトが正常にコンパイルされた場合は、Setup.inx (セットアップ エンジンが実行するオブジェクト コード) が作成され、リリースのビルド時に Windows Installer パッケージにストリーム入力されます。

スクリプトの変更を確かめる前に、以前実行し InstallScript カスタム アクションをテストしたリリースをアンインストールしなければならない場合もあります。

[設定] ダイアログ ボックス の [コンパイル/リンク] タブでコンパイラ オプションを設定することができます。

スクリプトのデバッグ

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript コードを順番に確認にしながら、コードの進行をチェックする場合は、InstallShield デバッガーを使用すると便利です。

スクリプトのデバッグを行う前に、まずスクリプトをコンパイルして、カスタム アクションとして実行し (該当する場合)、リリースをビルドしておく必要があります。

タスク ***InstallShield から直接スクリプトをデバッグするには、以下のいずれかを実行します。***

- **[ビルド]** メニューで **[デバッグ]** をクリックします。
- F5 キーを押します。
- ツールバー上の [**デバッグ**] ボタンをクリックします。

InstallShield はインストールを実行し、カスタム アクションの実行時に InstallScript デバッガーを開きます。

InstallScript デバッガの使用中に、任意の場所で F1 を押すと InstallScript デバッガー ヘルプが表示されます。

テスト システムのデバッグについては、「任意のコンピューターでのインストールのデバッグ」を参照してください。

プリプロセッサ ステートメントを使用してスクリプトをデバッグする

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

スクリプトの内部デバッガー作成には、#define ステートメントと #ifdef ステートメントを使用します。

タスク ***プリプロセッサ ステートメントを使用してスクリプトをデバッグするには、以下の手順を実行します。***

1. スクリプトにデバッグステートメントを挿入するときは常に、次の #ifdef 命令で始めます:

   ```
   #ifdef DEBUG
   ```

2. その後に続く行で、デバッグ ステートメントを入力します。
3. デバッグステートメントの後の別の行に、次を入力します:

   ```
   #endif
   ```

4. デバッグには、次のコンパイラ設定をコンパイルします。

   ```
   DDEBUG=1
   ```

これは、#ifdef ステートメントを利用したデバッグセクションの例です:

```
#ifdef DEBUG
   if nResult < 0 then
      WriteLine (LogFileHandle, "PlaceBitmap が失敗しました");
   endif;
#endif
```

InstallScript デバッガーを使用して、インストールの実行と同じ状態で、プロジェクトの実行を追跡して、変数を検証することができます。

InstallScript ファイルの名前を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

タスク ***InstallScript ファイルの名前を変更するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、ファイルを右クリックし **[名前の変更]** をクリックします。
3. ファイルの新しい名前を入力します。

プロジェクト: *InstallScript プロジェクトには、*Setup.rul *という名前のソースファイルが必要です。このファイルはインストール スクリプトのメイン コンパイル ユニットです。InstallScript プロジェクトに、この名前のスクリプト ファイルが含まれていない場合、スクリプトのコンパイル時、またはリリースをビルドした時に、エラー C8503 が発生します。*

スクリプト中で文字列エントリを使用する

文字列識別子を、スクリプトで文字列リテラルを受け入れる任意の値の代わり使用できます。カスタム アクションが実行される時、文字列識別子はインストールを実行中の言語に対応する文字列値で置換されます。

スクリプト中の文字列 ID の先頭には @ マークを置く必要があります。[文字列の選択] ダイアログ ボックスを使って、プロジェクト内の文字列 エントリの一覧を参照できます。ここではまた、選択した文字列識別子をスクリプトに挿入する前に、デフォルト言語の文字列エントリを変更することもできます。

たとえば、プロジェクトに MSG_ACTION_SUCCEEDED という文字列識別子が含まれているとき、その値を次のようにメッセージ ボックスに表示することができます:

```
szMsg = @MSG_ACTION_SUCCEEDED;
nType = INFORMATION;
MessageBox (szMsg, nType);
```

プロジェクトから InstallScript ファイルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

インストールからスクリプト ファイルを削除する際には、ファイルそのものではなくそのファイルへの参照をプロジェクトから削除します。これにより、後でインストールにそのスクリプト ファイルを挿入することになった場合、スクリプトを書き直す必要がありません。

ただし、スクリプト ファイルを別のファイル中に含めると、コンパイルが可能な場合もあります。

タスク　　プロジェクトから InstallScript ファイルを削除するには、以下の手順を実行します。

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、ファイルを右クリックし **[削除]** をクリックします。

注意・スクリプト ファイルが上記手続きで削除された場合でも、ファイルの名前を、プロジェクトの場所に残っているそのファイルと同じ名前に変更することはできません。たとえば、プロジェクトに Setup.rul という名前のスクリプト ファイルがあるとします。次に、そのファイルを削除して、デフォルトで Script1.rul という名前の新しいスクリプト ファイルを追加します。このとき、Script1.rul を Setup.rul という名前に変更することはできません。それによって最初の Setup.rul が上書きされてしまうからです。これを回避するには、元の Setup.rul をプロジェクト フォルダーの外へ移動してから名前を変更するか、Setup.rul を Windows のエクスプローラーで削除してください。

スクリプト ライブラリ (.obl ファイル) を作成する

InstallScript のビルトイン関数はライブラリ ファイル (.obl ファイル) で、スクリプトがコンパイル時にリンクされるように定義されています。

タスク　　定義した関数のライブラリ ファイルを作成するには、以下の手順を実行します。

1. 関数の定義を含む 1 つまたは複数の .rul ファイルを作成します。
2. コマンドラインで、各 .rul ファイルに対して、-c スイッチで Compile.exe を実行し、既存のライブラリ ファイルとリンクさせずに .rul ファイルをコンパイルします。これにより .obs ファイルを作成します (-c スイッチなしのコンパイルで .inx ファイルを作成する方法と異なります)。たとえば、次のコマンドラインは現在のフォルダーに MyFunc.obs ファイルを作成します。

   Compile MyFunc.rul -c

   .obs ファイルを別の名前や場所で作成するには、-o スイッチを使用します。
3. -l スイッチを使用して Compile.exe と 1 つまたは複数の .obs ファイルをパラメーターとして実行して、ライブラリ ファイルを作成します。たとえば、次のコマンドラインは現在のフォルダーに MyFunc.obl ファイルを作成します。

   Compile MyFunc.obs -l

   次のコマンドラインは現在のフォルダーに MyFunc1.obl ファイルを作成します。

Compile MyFunc1.obs MyFunc2.obs -l

.obs ファイルを別の名前や場所で作成するには、-o スイッチを使用します。

ヒント: *.obs ファイルを多数保持している場合、次の例のようにコマンドファイルを使用してコマンドラインを短縮できます :*

Compile @MyObsFiles.txt -l

コマンドファイルをすばやく作成するために、MS-DOS コマンド DIR をその /b *(ベアフォーマット) スイッチと共に使って、出力をファイルにリダイレクトすることができます。例:*

DIR *.obs /b > MyObsFiles.txt

ライブラリ ファイルを使用してインストール スクリプトをコンパイルするには、コマンドライン (または InstallShield 内でコンパイルしている場合は [設定] ダイアログ ボックスの [コンパイル/リンク] タブ) でライブラリ ファイルを指定します。

InstallScript ファイル (.rul および .h) をリポジトリにパブリッシュする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

別のプロジェクトで再利用、または他のユーザーと共有したい既存 InstallScript ファイル (.rul) または InstallScript ヘッダー ファイル (.h) がある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***スクリプト ファイルをリポジトリにパブリッシュするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **[InstallScript]** エクスプローラーでパブリッシュするスクリプト ファイルを右クリックしてから、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへスクリプト ファイルをインポートした後、現在のスクリプト ファイルと既存のリポジトリ スクリプト ファイルとは関連性を持ちません。スクリプト ファイルを変更してリポジトリへ再パブリッシュしても、プロジェクト内のインポート済みスクリプト ファイルには影響しません。ただし、リポジトリからプロジェクトへスクリプト ファイルを再インポートすることができます。

InstallScript リスト

リストは、文字列や番号といった関連情報を格納するのに利用されます。InstallScript リストは C 言語の単方向リストと非常によく似ています。InstallScript リスト関数は柔軟性に富み、情報をそれらが格納されている順番とは異なる順序で戻すことが可能であり、さらにそれらにアクセスして様々な方法で利用することができます。

リスト関数

InstallShield ではリストを作成または操作するための関数がいくつか提供されています。InstallScript リスト関数には 3 つのタイプがあります:

- 文字列リストとのみ動作する **–String** で終わる関数
- 番号リストとのみ動作する **–Item** で終わる関数
- 文字列リストと番号リストのどちらとも共に動作する関数

InstallShield はまた、リストを作成または利用する従属的なリスト関連関数を提供します。

リスト構造

リストは、文字列や番号のいずれかに関連する情報を格納するのに利用されます。リスト内のすべての情報は同じデータ型であり、要素の数は利用可能なメモリ容量によってのみ制限されます。

InstallScript リストは 2 つの部分で構成されます。最初の部分はヘッドで、InstallShield が内部で利用します。リストのヘッドにはリストについて、文字列または番号を含んでいるか否かといった一般情報が含まれています。ヘッドにはまたリストの開始と終了へのポインターが含まれます。

リストの 2 番目の部分はリスト本体です。リスト本体には実際の文字列や番号が含まれます。システムのメモリが有効な限り、リストには文字列や番号を含むことができます。

リストは文字列と番号を同時に含むことができない点に注意してください。リストは文字列のみ、または番号のみを含まなくてはなりません。

リストを代表する変数は、type LIST または type LONG として宣言できます。リストはメモリにのみ格納されます。つまり、インストールが完了したときに、リストは破棄されます。リストが関数のローカルであった場合、関数が呼び出しコードへコントロールを戻したときにリストは破棄されます。

リストの作成と破棄

リスト作成前に、ビルドするリストが文字列リストと数値（項目）リストのどちらかを決めます。リストを作成するには、**ListCreate** 関数を呼び出します。

```
// これは文字列リスト用のリスト ヘッドをビルドします。
listID1 = ListCreate (STRINGLIST);
```

または

```
// これは数値（項目）リスト用のリスト ヘッドをビルドします。
listID2 = ListCreate (NUMBERLIST);
```

ListCreate は自動的にリストのヘッドをビルドし、その ID 番号を戻します。ID は、リストで動作する後に続く関数すべてに利用されます。したがって、別のリスト関数を利用する前に常に **ListCreate** を使ってリストを作成しなくてはなりません。**ListCreate** からの戻り値を LIST タイプまたは LONG タイプの値で格納しなくてはなりません。

この部分が数値リストと、そのあと文字列リストを作成します。また、リストが正しく作成されたことを確認するためにそれぞれをテストします。

```
// 文字列の空リストを作成します。
listID1 = ListCreate (STRINGLIST);

if (listID1 = LIST_NULL) then
    MessageBox ("文字列リストを作成することができませんでした", SEVERE);
endif;

// 数値の空リストを作成します。
listID2 = ListCreate (NUMBERLIST);

if (listID2 = LIST_NULL) then
   MessageBox ("数値リストを作成することができませんでした", SEVERE);
endif;
```

リストを利用し終わったとき、その他の用途のためにメモリーを解除するため、通常はリストを破棄します。**ListDestroy** がリストとその内容を破棄します。この例は listID が参照するリストを作成し、リストへ文字列を追加してリスト全体を破棄します。

```
listID = ListCreate (STRINGLIST);

if (listID = LIST_NULL) then
   MessageBox ("リストを作成できませんでした。", SEVERE);
   abort;
endif;

ListAddString (listID, "これはリストの文字列です", AFTER);
ListDestroy(listID);
```

ListDestroy を使ってリストを破棄しなかったなかった場合、インストールが完了したときにリストは破棄されます。リストが関数のローカルであった場合、関数が呼び出しコードへコントロールを戻したときにリストは破棄されます。

リストへ要素を追加する

InstallShieldではリストへ要素を追加するための関数がいくつか提供されています：

テーブル 1・リストへ要素を追加するときに使用する関数

関数	説明
ListAddItem	項目をリストに追加します。
ListAddString	リストへ文字列を追加します。
ListReadFromFile	テキストファイルをリストへ読み込みます。

ListAddString および **ListAddItem** は、単一の要素を指定されたリストへ追加します。リスト内での新規文字列の配置場所にかかわらず、それが現在の文字列となることに注意してください。BEFORE と AFTER オプションを使って、リスト内で現在の要素に関連して新規要素を配置する場所を指定します。新しく作成されたリストを使って作業中の場合、BEFORE または AFTER のいづれかを使用するとリストの最初の要素位置に文字列が配置されます。

リストへの要素の追加と、それがリスト順や現在の位置にある要素へ与える影響は、例をみると簡単に理解することができます。以下の例では文字列リストと **ListAddString** を使用しますが、**ListAddItem** や数値リストを使用する場合にも同じ原理と手順が当てはまります。次のシナリオを参考にしてください:

- 空白リストへ要素を追加する
- 現在の要素の前に要素を追加する
- 現在の要素の後に要素を追加する
- 現在の要素の前と後に要素を追加する

空白リストへ要素を追加する

リストへ追加した最初の文字列はリストのヘッド部分の直後に配置されます。この文字列 (サンプル コードの *String 1*) がリストで現在の文字列になります。下に表示された文字列部分は、その後に表示されるリストの様な結果となります。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);
endif;
```

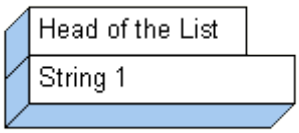


図 1: String 1 がリストへ追加されます。

現在の要素の前に要素を追加する

現在の文字列がリストのトップにある場合で、新規文字列をその前に追加するとき、新しい文字列がリストのトップになります。その結果、元々最初の要素位置にあった文字列は二番目の要素位置へと移ります。最初の要素位置にある新規文字列が現在の文字列となります:

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);

    szString = "String 2";
    ListAddString (listID, szString, BEFORE);
```

```
endif;
```

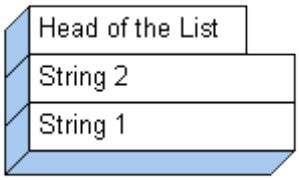


図 2: String 2 が String 1 の前へ追加されます。

現在の要素の後に要素を追加する

現在の文字列がリストのトップにある場合で、現在の文字列の後に新規文字列を追加した場合、リストで 2 番目の文字列となり新しい現在の文字列となります。次の文字列と、後に続く解説を参考にして下さい:

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);

    szString = "String 2";
    ListAddString (listID, szString, AFTER);
endif;
```

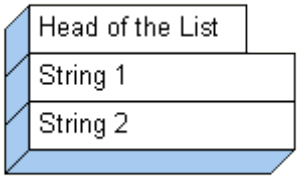


図 3: String 2 が String 1 の後へ追加されます。

現在の要素の前後に要素を追加する

別の例として、以下に示されたコード部分は新規リストを作成し String 1 を先頭位置に配置します。そして String 2 が String 1 の前に追加され、String 2 は先頭位置に現在の文字列として残ります。次に、String 3 が現在の文字列の後に追加され、以下に表示された結果となります。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);
```

```
    szString = "String 2";
    ListAddString (listID, szString, BEFORE);

    szString = "String 3";
    ListAddString (listID, szString, AFTER);
endif;
```

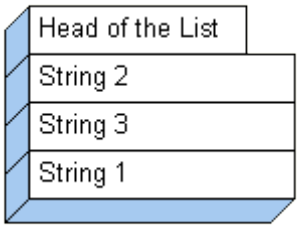


図 4: String 1 が追加されます。その後、String 2 が String 1 の前へ追加されます。最後に、String 3 が String 2 の後へ追加されます。

上の例では、String 3 が現在の文字列の前に追加された場合、下のような結果となり String 3 が現在の文字列となります。コード部分は次の通りです。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト
if (listID = LIST_NULL) then
    MessageBox ("リストは作成されません", SEVERE);
else
    // リストへ文字列を追加する。
    szString = "String 1";
    ListAddString (listID, szString, AFTER);

    szString = "String 2";
    ListAddString (listID, szString, BEFORE);

    szString = "String 3";
    ListAddString (listID, szString, BEFORE);
endif;
```

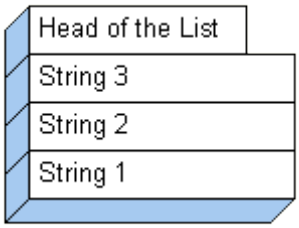


図 5: String 1 が追加されます。その後、String 2 が String 1 の前へ追加されます。最後に、String 3 が String 2 の前へ追加されます。

リスト内にある既存の要素を変更する

ListSetCurrentString 関数を呼び出して、文字列リスト内にある要素の値を変更します。現在の要素のみ変更が可能で、アップデートしたい文字列がリスト内で現在の文字列であることを確認して下さい。

ListSetCurrentItem 関数を呼び出して数値リスト内にある要素の値を変更します。上記と同じく、アップデートする項目はリスト内で現在の要素でなくてはなりません。

下の例では、**ListSetCurrentItem** を呼び出して数値リスト内の現在の項目の値を変更する方法をデモンストレーションします。**ListSetCurrentString** 関数も同じ要領で処理を行いますが、文字列リストと文字列変数を扱います。

```
// リストを作成し、その結果を確認します。
listID = ListCreate(NUMBERLIST);
if (listID = LIST_NULL) then
    MessageBox ("リストを作成できませんでした。", SEVERE);
    abort;
endif;


// 項目 (1078 と 304) をリストに追加します。
nItem = 1078;
ListAddItem (listID, nItem, AFTER);
nItem = 304;
ListAddItem (listID, nItem, AFTER);

// 現在の項目は 2 番目の項目 (304) です。
// ここで現在の項目を新しい値 (304) に設定します。
nItem = 305;
ListSetCurrentItem (listID, nItem);
ListDestroy(listID);
```

リストから要素を削除する

ListDeleteString 関数を呼び出して、リストから現在の文字列を削除します。または **ListDeleteItem** 関数を呼び出して、リストから現在の番号を削除します。他に削除する要素がない場合、これらの関数は END_OF_LIST 定数を戻します。

ListDeleteString と **ListDeleteItem** は現在の要素を削除するため、現在の要素を削除する要素へリセットしなくてはなりません。削除のあと、リストの次の要素が現在の要素となります。「リストのスキャン」で説明されているいずれかの方法で要素をリセットします。

以下の例は **ListDeleteString** を含むいくつかのリスト関数を具体的に説明します。**ListDeleteItem** 関数も同じ要領で利用されますが、異なる点はリストが数値リストであり、変数が数値変数であることです。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

if (listID = LIST_NULL) then  // 有効リストのテスト。
    MessageBox ("リストは作成されません", SEVERE);
endif;

// リストへ文字列を追加する。
szString = "String 1";
ListAddString (listID, szString, AFTER);
szString = "String 2";
ListAddString (listID, szString, AFTER);
szString = "String 3";
ListAddString (listID, szString, AFTER);

// 現在の文字列を削除します。
ListDeleteString (listID);
```

```
// リストにある現在の文字列を再設定します。
lResult = ListCurrentString (listID, svString);

// svString contains "文字列 2."
```

リストの特定要素を検索する

リストを降順にスキャンすることができます。リスト内の特定文字列あるいは番号要素を検索するには、**ListFindString** 関数または **ListFindItem** 関数を呼び出します。これら 2 つの関数は現在の要素から検索を開始し、その位置から続けて検索を行います。

リストの最初から検索を開始するには、**ListFindString** 関数または **ListFindItem** 関数を呼び出す前に **ListGetFirstString** 関数または **ListGetFirstItem** 関数を呼び出します。

ListFindString 関数または **ListFindItem** 関数が指定した文字列または番号を検出したとき、それがリストの現在の要素となります。

下の部分スクリプトでは、数値リストが作成され、(nItem 内の) 番号 1 が最初の要素として追加されます。そして、**ListFindItem** が番号 1 をリスト内で検索し、検出した場合はそれを削除します。最後に、リストが破棄されます。

```
listID = ListCreate(NUMBERLIST);

if (listID = LIST_NULL) then
    MessageBox ("リストを作成できませんでした。", SEVERE);
    abort;
endif;

nItem = 1;
ListAddItem (listID, nItem, AFTER);

if (ListFindItem (listID, nItem) = 0) then
     ListDeleteItem (listID);
endif;

ListDestroy(listID);
```

メモ: ***ListFindString** 関数と **ListFindItem** 関数は、現在の要素位置またはその後にある、指定した文字列または番号の最初のインスタンスのみを検索します。*

リストの 1 番目と 2 番目の要素を取得する

ListGetFirstString 関数または **ListGetFirstItem** 関数を呼び出して、リストから最初の文字列要素、または数値要素を戻します。読み出した要素が、リストの現在の要素になります。

ListGetNextString 関数または **ListGetNextItem** 関数を呼び出してリストの現在の要素の後に最初の文字列要素、または数値要素を戻します。読み出した要素が、リストの現在の要素になります。

```
// 空白文字列リストを作成する。
listID = ListCreate (STRINGLIST);

// 有効リストのテスト。
if (listID = LIST_NULL) then
```

```
    MessageBox ("リストは作成されません", SEVERE);
endif;

// リストへ文字列を追加する。
ListAddString (listID, "String 1", AFTER);
ListAddString (listID, "String 2", AFTER);
ListAddString (listID, "String 3", AFTER);

// リストをスキャンし、文字列をメッセージボックスに表示します。
lResult = ListGetFirstString(listID, szDriveName);

while (lResult !=  END_OF_LIST)
    MessageBox (szDriveName, INFORMATION);
    lResult = ListGetNextString (listID, szDriveName);
endwhile;
```

リストへファイルを読み込む

ファイル全体を文字列リストへ読み込むには、**ListReadFromFile** 関数を呼び出します。ファイルの各行はリストの要素となります。**ListReadFromFile** 関数を利用して、アイテムをひとつずつビルドする方法よりも簡単にリストをロードすることができます。

リストのインデックスを設定する

InstallShield が提供する **ListSetIndex** 関数を利用すると、インデックス番号を使ってある要素を現在の要素にすることができます。リストの特定の要素がどこにあるかが分かっている場合、**ListSetIndex** 関数を呼び出してその要素に直接アクセスすることが可能です。特定の要素を現在の要素へ設定するために、インデックスを利用してリストをどちらかの方向へトラバースすることができます。リストのインデックスは 0 (ゼロ) で始まります。

ListSetIndex 関数は、文字列リストでも数値リストでも使用できます。インデックスを付けた要素を現在の要素に設定した後で、**ListCurrentItem** 関数または **ListCurrentString** 関数を呼び出して、インデックスが付いた項目の値を返します。

この例は **ListSetIndex** を使ってリストを降順にスキャンする方法を説明します。

```
listID = ListCreate (STRINGLIST);
GetGroupNameList (listID);
nCheck = ListSetIndex (listID, LISTFIRST);

while (nCheck != END_OF_LIST)
    ListCurrentString (listID, svString);
     MessageBox (svString, INFORMATION);
     nCheck = ListSetIndex (listID, LISTNEXT);
 endwhile;

ListDestroy(listID);
```

ListCount 関数はリスト内の要素の数を報告します。**ListCount** 関数は **ListSetIndex** と共に上位インデックス値の設定に利用することもできますが、主に一般情報の目的の為に利用されます。たとえば、**ListCount** を呼び出してリストの数々の要素を取得し、その値を **ListSetIndex** と共に利用してリストをスキャンことができます。while ループを利用する前述の例は、リストの要素番号に基づいたループ用 InstallScript を使って下に説明されています。

```
listID = ListCreate (STRINGLIST);
GetGroupNameList (listID);
```

```
// リストの要素の数を取得します。
nItems = ListCount (listID);

// リストの要素の数を表示します。
SprintfBox (INFORMATION, "", "i = %d", nItems);

// ゼロで始まるループを nItems 回行い、
// メッセージボックスに順番に各リスト要素を表示します。
for i = 0 to (nItems - 1)
    ListSetIndex (listID, i);
    ListCurrentString (listID, svString);
    MessageBox (svString, INFORMATION);
    nCheck = ListSetIndex (listID, LISTNEXT);
endfor;

ListDestroy(listID);
```

リストのスキャン

InstallShield は増加順にまたは減少順に検討するためにこれらの関数を提供します:

テーブル 2 ・ リストをスキャンするときに使用する関数

関数	説明
ListCount	指定したリストに含まれている文字列または数値要素の数を確認します。
ListCurrentItem	現在の項目をリストに戻します。
ListCurrentString	現在の文字列をリストに戻します。
ListFindItem	リストで数値要素を検出します。要素が見つかった場合、リストの現在の要素になります。
ListFindString	リストで文字列要素を検出します。要素が見つかった場合、リストの現在の要素になります。
ListGetFirstItem	番号リストから 1 番目のアイテムを読み出します。
ListGetFirstString	文字列リストから 1 番目の文字列を読み出します。
ListGetNextItem	番号リストから現在のアイテムの後のアイテム取得します。
ListGetNextString	文字列リストから現在のアイテムの後のアイテムを取得します。
ListSetIndex	リストの現在の要素をインデックスとします。

InstallShield は単一リンクリストを利用します。つまりインデックスを設定またはリスト内の特定の要素を検索する関数を利用しない限り、一番目の要素から最後の要素への一定方向のみにリストを増加してスキャンすることができます。

InstallShield では、索引と、リストの特定の要素を検索することによって、非増分的にリストを検討することができます。詳細については、それぞれの関数の説明を参照してください。

ほとんどのリストのトラバースとリストへのアクセス処理は、現在のリスト要素と関連付けて実行されます。さらに、リストのトラバースとリストへのアクセスに利用されるほとんどの関数は、現在の要素をそのアクションの結果として確立します。したがって、要素をリストの現在の要素とするのは、独立したアクションではなく、別のアクションの副産物といえます。

リストが空の場合にリストへ要素を追加すると、現在の要素を確立することになります。リストが空白でない場合にひとつの要素を現在の要素にするには、リストの検討またはリスト内の特定の要素の検索を行うのが最も有効です。

ヒント: *リストはしばしば while ループで処理され、通常 END_OF_LIST を確認します。無限ループはリストが無効な場合に起こります。リストを while ループで処理する場合、***ListCreate** *関数を利用してリストを作成したこと、そして* **ListDestroy** *関数を利用してリストを破棄していないことを確認してください。*

ファイルへリストを書き込む

文字列リストの内容をファイルへ書き込むには、**ListWriteToFile** 関数を呼び出します。リストの各要素はファイルの行となります。

下のスクリプト例は、 Autoexec.bat ファイルを listFile 文字列リストへ読み込み、その文字列リストを Autoexec.bak と名づけられたファイルへ書き込みます。

```
listFile   = ListCreate (STRINGLIST);
szPath     = "C:¥¥";
szFileOld  = "C:¥¥Autoexec.bat";
szFileNew  = "C:¥¥Autoexec.bak";

if (ListReadFromFile (listFile, szFileOld) < 0) then
   MessageBox ("ListReadFromFile が失敗しました。.", SEVERE);
endif;

ListWriteToFile (listFile, szFileNew);
```

InstallScript ファイルの保存

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallShield の [ファイル] メニューから [保存] をクリックすると、すべての開いているスクリプト ファイルが自動的に保存されます。

システムの復元

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

システムの復元は、エンドユーザーがソフトウェアインストール中に破損した PC を回復できる Windows の機能です。システムの復元機能は、エンド ユーザーの PCの 重要なシステム変更を自動的に監視し、記録します。システムの復元は、システムに危険をもたらす変更を簡単に取り消したり、システムが理想的に実行されていたときに戻すことを可能にすることにより、サポートコストを削減し、カスタマーの満足を向上させます。

InstallScript インストールではセットアップはファイル転送が始まる前に「*復元ポイント*」を設定してシステムの復元をサポートします。エンドユーザーは [システムの復元] を使用して、ファイル転送前の状態にシステムを復元できます。

メモ: *セットアップアクション (たとえばレジストリ変更やファイル変更) でファイル転送前に起きるものはシステムの復元でやり直しできません。*

ご使用のインストールはデフォルトでシステムの復元と互換性があります。(スクリプトの OnBegin イベント ハンドラーに以下のコードを挿入することにより、セットアップをシステムの復元機能に対応しないようにすることができます。

```
Disable( PCRESTORE );
```

Wininit.ini というファイルがターゲット マシンの Windows フォルダーに存在する場合、インストールは復元ポイントを設定できません。Wininit.ini を処理するには、**OnFirstUIBefore** と **OnMaintUIBefore** イベント ハンドラー関数に次のようなコードを挿入します。

```
/* Windows フォルダーで Wininit.ini を検索します。場所は...*/
if FindFile( WINDIR, "Wininit.ini", svResult )=0 then
    bRebootForSystemRestore = TRUE;
    /* ... サイズを取得します。*/
    GetFileInfo( WINDIR ^ "Wininit.ini", FILE_SIZE, nvSize, svResult );
    /* サイズがゼロバイトの場合 ...*/
    if nvSize=0 then
        /* ... Wininit.ini を削除します。*/
        if DeleteFile( WINDIR ^ "Wininit.ini" )=0 then
            bRebootForSystemRestore = FALSE;
        endif;
    endif;
    /* Wininit.ini がゼロ以外のサイズの場合、または削除不可能な場合はエンドユーザーに対して通知して再起動を可能にします。*/
    if bRebootForSystemRestore then
        szQuestion = "[Windows システム回復] を利用して 、コンピューターの変更を" +
            "元に戻すことができます。[システム回復] を利用して" +
            "このインストールを下に戻すようにできるようにしたい場合、 " +
            "コンピューターを今すぐ再起動してください。¥n¥n" +
            "コンピューターを今すぐ再起動しますか?"
        if AskYesNo( szQuestion, YES )=YES then
            System( SYS_BOOTMACHINE );
        endif;
    endif;
endif;
```

プロパティの取得と設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Windows Installer プロパティには、製品、セットアップ、オペレーティング システム、ユーザーに関する有用な情報が保持できます。**MsiGetProperty** および **MsiSetProperty** を呼び出して、InstallScript カスタム アクション中でこれらのプロパティと直接対話することができます。

メモ: *MsiGetProperty プロパティおよび* MsiSetProperty *プロパティは、遅延 InstallScript カスタム アクションに使用することはできません。これは、アクティブな .msi データベースへのアクセスが不可能であり、Windows Installer プロパティを認識しないためです。実行スクリプトに書かれた情報のみにアクセスすることができます。*

次の例では、Windows Installer セットアップパッケージからユーザー名を取得して、値を確認し、ユーザーにそれを変更する機会を与え、データベースに新しい値を書き込んでいます。

メモ: *プロパティ値が 1024 文字を超える場合、適切なバッファ サイズを指定する必要があります。この状況におけるコードの記述例については、「プロジェクトを InstallShield 2011 以前からアップグレードする」トピック内の「InstallScript カスタム アクションで呼び出される一部の MSI API の動作を変更する」を参照してください。*

```
//ビルトイン関数のヘッダー ファイルを含める
#include "isrt.h"
//MSI API 関数と定数のヘッダー ファイルを含める
#include "iswi.h"

export prototype Func1(HWND);

function Func1(hMSI)
   STRING svName;
   NUMBER nvSize, nResponse;
begin
   //MSI データベースからユーザーの名前を読み出す
   nvSize = 256;
   MsiGetProperty (hMSI, "USERNAME", svName, nvSize);

   nResponse = AskYesNo ("登録名は、" +
                  svName + "です。間違いないですか?", YES);

   if nResponse = NO then
      AskText ("この製品の登録名を" +
            "入力してください。", svName, svName);
      MsiSetProperty(hMSI, "USERNAME", svName);
   endif;
end;
```

MsiGetProperty (Windows Installer ヘルプ ライブラリ
MsiSetProperty (Windows Installer ヘルプ ライブラリ)

ビット フラグの使い方

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ビットフラグは、単一数値変数に格納されているひとつまたは複数 (最大 32) の Boolean 値です。

一般的に各ビット フラグには、関連付けられた対応の定義済み定数があります。この定数はフラグ用のビットが 1 に設定されており、その他のビットは 0 です。たとえば、次の定数は、0 ビット用のビット フラグを認識します。つまり、番号の一番右側にあるビットです。

```
#define BITFLAG_EXAMPLE_1 0x00000001
```

その他のビット フラグは次の通りです:

```
#define BITFLAG_EXAMPLE_2 0x00000002
#define BITFLAG_EXAMPLE_3 0x00000004
```

0x00000003 は有効なビット フラグでない点に注意してください。なぜならこの値は 1 に設定された番号にある 2 つのビットに対応するからです。

変数でビットフラグを設定する

変数でビット フラグを設定するには、OR 演算子 (|) を利用します。たとえば、数値変数 *nFlags* の 値 BITFLAG_EXAMPLE_1 および BITFLAG_EXAMPLE_2 を割り当てる場合、次のようなコードを使用します。

```
nFlags = nFlags | BITFLAG_TEST_1 | BITFLAG_TEST_2;
```

変数からビット フラグをクリアする

変数からビット フラグをクリアするには、ビット単位の AND 演算子 (&) とビット単位 NOT 演算子 (~) を組み合わせます。たとえば、BITFLAG_TEST_1 フラグを nFlags から削除するには、次のようなコードを使用します。

```
nFlags = nFlags & ~BITFLAG_TEST_1;
```

ビットフラグ用の変数をテストする

ビットフラグ用の変数をテストするには、ビット単位 AND 演算子 (&) を利用して、テストするビット以外のビットフラグ定数をすべてゼロにします。たとえば、*nFlags* が現在 **BITFLAG_TEST_1 | BITFLAG_TEST_2** に設定されている場合、後に続く式は true 評価 (つまり、0 以外の結果) を行います。

```
nFlags & BITFLAG_TEST_1
```

& 演算子によって、一番右にあるバイト以外はすべて 0 に設定されています。定数 BITFLAG_TEST_1 と *nFlags* の両方が 1 なので、一番右のビットは 1 となります。

次の式は、nFlags の 3 番目のビットが 0、そしてその他のビットが 演算子によって 0 に設定されているので、FALSE 評価します。

nFlags & BITFLAG_TEST_3

文字列比較

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallShield が 2 つの文字列を比較するとき、まず最初の文字列の最初の文字と 2 番目の文字列の最初の文字とを比較します。これらの文字が等しい場合、InstallShield は各文字列の次の位置にある文字を比較します。これらの文字も等しい場合は、その次の位置にある文字に移動し、下の条件のひとつに合致するまでシーケンスを続けます:

- 2 つの文字列の対応する位置にある 2 つの文字が一致しません。この場合、InstallShield はこれら 2 つの文字の比較に基づいて解決します。文字列 1 の文字が文字列 2 の対応する位置にある文字よりも大きな値を持つとき、文字列 1 が大きくなります。それ以外の場合は文字列 2 が大きくなります。
- 対応する位置で異なる文字を検出せずに文字列 1 の終わりに到達しました。この場合、文字列の長さは異なり、つまり等しくないことになります。
- 対応する位置で異なる文字を検出せずに両方の文字列の終わりに到達しました。この場合、文字列の長さは同じで、文字のすべてが一致します。つまり、文字列が等しいことになります。

次の例を参考にして下さい:

```
svString1 = "trusting";
   svString2 = "TRUTHFUL";

   if svString1 = svString2 then
      MessageBox("等しい", INFORMATION);
   else
      MessageBox("等しいくない", INFORMATION);
   endif;
```

文字列比較で大文字と小文字は区別されません。大文字が対応する小文字と等しいため、InstallShield は **trusting** の最初の 3 文字と **TRUTHFUL** の最初の 3 文字が同等であると判断します。比較は各文字列の 4 番目の文字のテストで終了します。ASCII 文字テーブルで **s** は、**t** よりも下位にあるため、svString1 と svString2 は等しくありません。残りの文字は比較されず、else ブランチが実行されます。

メモ: *各文字の値は、その ASCII 値に基づきます。特定の文字や記号の ASCII 値に関する情報については、基本のプログラミングマニュアルを参照してください。*

ヌル区切り文字列の使用

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

*「複数ヌル区切り文字列」*または*「複数ヌル文字で終わる文字列」*とも呼ばれる、ヌル区切り文字列、2 つのヌルで終わる文字列 (例、**abc¥0def¥0ghi¥0¥0**) を利用するときは次の点に注意してください。

- ヌル区切り文字列値へ設定される変数を宣言する際、変数のサイズを明示します。例:

  STRING szString[1024];

  オートサイズは使用しないで下さい。インストールエンジンはオートサイズされた文字列はヌル文字を文字列内に含まないものとみなします。

- ヌル区切り文字列の配列を作成しないで下さい。配列要素は常にオートサイズされなくてはならないため、文字列配列は現在この種類の文字列をサポートしません。

- ヌル区切り文字列の指定されたサイズは保存される文字数以上と、さらにヌル文字 2 つ分を含まなくてはなりません。

- 文字列は自動的にヌル文字に初期化されるため、以前に最後から 2 番目の文字をヌル以外に設定していない限り、これをヌルと明示する必要はありません (明示したところで悪影響がもたらされる訳ではありません)。

- ヌル区切り文字列の長さを判断する最も有効な手段は、**CharReplace** 関数を呼び出してヌル文字を置換してから、**StrLengthChars** 関数を呼び出して結果文字列のサイズを判断する方法です。

- **StrGetTokens** 関数と **StrPutTokens** 関数は、ヌル区切り文字列を使った作業に便利です。

- 前述の関数以外、ほとんどの InstallScript 関数は複数ヌル区切り文字列と共に利用することはできません。ビルトイン関数と共に複数のヌル区切り文字列を利用する場合、まず **CharReplace** 関数を利用して、たとえば、アンダースコア (_) のような別の文字を使ってヌル文字を置換します。

相対パス

相対パスは、現在のドライブ上の現在のフォルダーを起点にして、ファイル検索に必要な情報をすべて含みます (例、Support¥Validation)。フォルダーは、現在のディレクトリに存在する場合のみ、相対パスで検索することができます。

長いファイル名

32 ビット Windows オペレーティング システムは、長いファイル名をサポートします。長いファイル名を使うと、エンドユーザーはディレクトリやファイルにより分かりやすい意味のある名前をつけることができます。用語 "*長いファイル名*" は、長いファイル名と長いパスの両方を意味します。

InstallShield が提供する長いファイル名関数は、長いファイル名を認識しない 16-bit アプリケーションや 32-bit アプリケーションのインストールを可能にします。アプリケーションの要件を判断するのは開発者の判断に委ねられています。InstallShield は、あらゆる種類のアプリケーションのインストールに役立つツールを装備しています。

長いファイル名と 16 ビット アプリケーション

長いファイル名を利用して 16 ビット アプリケーションを起動することができますが、長いファイル名を 16 ビット アプリケーションへ引数として渡すことはできません。16 ビット アプリケーションが適切に機能するには、長いファイル名の短いファイル名バージョンが必要です。

インストールがファイル名を 16-bit アプリケーションへ渡す場合、アプリケーションへの有効な短いファイル名を用意しなくてはなりません。長いファイル名では動作しません。

たとえば .ini ファイルや 16-bit アプリケーションが利用するファイルへインストールがファイル名を書き込む場合、16-bit アプリケーションが利用する短いファイル名を書き込まなくてはなりません。

長いファイル名と二重引用符

32 ビット オペレーティング システムで、コマンドラインに 1 つ以上の空白文字を含む長いファイル名を渡した場合 (例: DOS シェル内や、アイコンの "コマンドライン" 設定内)、長いファイル名を二重引用符で囲まなくてはなりません。これは、コマンドラインが空白文字を他の式からコマンドを区別する区切り文字として認識するために必要となります。二重引用符は長いファイル名を文字列リテラルへ変換して、コマンドラインがそれを単独の引数として受け取る事を可能にします。

32 ビット オペレーティング システムで、1 つまたは複数の空白文字を含む長いファイル名を渡す場合、それを二重引用符で囲まなくてはなりません。しかしながら、Windows NT によるアイコン ファイルへのアクセス形態のため、二重引用符で囲んだ長いファイル名がアイコンの "コマンドライン" 設定で利用されている場合、アプリケーションのアイコンの代わりにデフォルトの Windows アイコンが表示される場合もあります。

製品のアイコンを表示するには、プログラム フォルダーへアイコンを追加するときに、

関数 AddFolderIcon のパラメーター szIconPath でアイコンのパスを指定することができます。

InstallShield で短いファイル名へ変換する前に、長いファイル名から二重引用符を削除しなくてはなりません。LongPathToQuote 関数と LongPathToShortPath 関数を参照してください。

長いファイル名の形式

長いファイル名は従来の 8.3 (8 文字と 3 桁拡張子) の短いファイル名制限よりも長い名前を持ちます。長いファイル名では短いファイル名で利用される文字すべてを利用することができます。さらに、長いファイル名にはプラス記号 (+)、コンマ (,)、セミコロン (;)、等号 (=)、左右角括弧 ([ ])、そして空白を含むことができます。

先頭と行末の空白は無視されます。完全修飾の (最後のヌル文字を含む) 長いファイル名は、NTFS ファイルシステム上では 256 文字まで、そして VFAT ファイルシステム上では 260 文字まで含むことができます。

デフォルトでは、オペレーティング システムが長いファイル名それぞれに短いファイル名を作成します。短いファイル名は長いファイル名の最初の 6 文字と、チルダ (~)、そして数字で構成されます。

コンパイラで定数を定義する

InstallScript 定数をスクリプト内ではなく、[設定] ダイアログ ボックスの [コンパイル/リンク] タブで定義することができます。

このタブで定数を定義する際、次のような制限が適用します :

- 数値定数のみを定義できます。
- スクリプト内で #define ステートメントを使って定義した定数を定義すると、エラーが発生します。
- スクリプト内で変数として定義した定数を定義すると、定数の値が実行時に失われます。

スクリプトで Windows 定数を使用する

一部の Windows 定数は、*InstallShield Program Files フォルダー*¥Script¥Isrt¥Include フォルダーにある ISRTWindows.h ファイルで既に定義されています。このファイルは、スクリプトに Ifx.h を含めると、自動的にインストールに含まれます。ISRTWindows.h に定義されている定数を再定義する必要はありません。再定義すると、コンパイラ警告が発生します。どの定数が定義済みかを判別するには、ISRTWindows.h ファイルを参照します。

ISRTWindows.h に定義されていない定数を使用する場合は、インストールスクリプトの宣言ブロックに必ずそれらを定義しなければなりません (#define を使用)。通常 C++ プログラムの一部である Windows.h ファイルを、単純挿入することはできません。未定義定数に割り当てる必要がある値は、通常は、該当する Windows SDK や開発ツール付属のインクルードファイルに含まれています。(Microsoft Visual C++ の場合、ほとんどの定数は *InstallShield Program Files フォルダー*¥Script¥Resource フォルダーにある Winuser.h ファイルに含まれています。)

長い文字列リテラルのコーディング

スクリプトで長い文字列リテラルを読み易くするため、文字列を 2、3 行に分割して順に並べ、連結させます。InstallScript では、連結にはプラス記号 (+) が使われます。次の例では、長い文字列リテラルを前述の方法でいくつかのコードラインに分割する方法を示します:

```
szMonths = "1 月、2 月、3 月、4 月、" +
        "5 月、6 月、7 月、8 月、9 月、" +
        "10 月、11 月、12 月";
```

絶対パス

***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

指定されたドライブのルート ディレクトリを起点にしてファイル検索をするのに必要な情報をすべて含む絶対パス。たとえば、C:¥Program Files¥InstallShield¥2022 は、C ドライブにインストールされたときの InstallShield フォルダーへの絶対パスです。

関数のビルド

一般的に、関数はファイルの最初で、また本文はファイルの最後で宣言されます。関数プロトタイプを宣言した後で、関数ブロックで関数そのものを定義する必要があります。各関数ブロックには 1 つの関数だけが含まれています。

タスク ***関数をビルドするには、以下の手順を実行します。***

1. 関数本体をキーワード **function** で始めます。
2. 関数プロトタイプに利用されている戻り値データタイプを入力する、または関数プロトタイプが **void** の戻りタイプを指定する場合、void を入力します (これは関数が値を戻さないことを示します。)。関数プロトタイプが戻り値の型を指定しない場合、空白のままにします。
3. 関数名を入力します。
4. 関数名の右にかっこの中で使用する引数名を入力します。引数は宣言ブロックに宣言するパラメーターのデータタイプに対応している必要があります。

   ***メモ・** InstallScript 関数では割り当てステートメントをパラメータとして渡すことができません。さらに && や || 演算子を関数への引数の中で使用することはできません。*

   ステップ 1 と 2 は関数ヘッダー を作成します。関数ヘッダーは InstallScript ではセミコロンで終わりません。

5. 関数で使用するローカル変数を指定します。次に **begin** というキーワードを区切り記号を入れずに入力します。

6. begin の行の次に、特定のタスクの実行に必要なステートメントを追加します。

   関数から特定の値を返す場合は特に、return ステートメントを使用することもできます。関数から値を戻す方法については次を参照して下さい。

7. 関数はキーワード **end** で終了します。

サンプル関数ブロックを以下に示します。

```
function GetPathParts(szFullPath, svDrv, svPath, svName)
    LONG lResult;
begin
    lResult = ParsePath(svDrv, szFullPath, DISK);
    if (lResult = 0) then
        lResult = ParsePath(svPath, szFullPath, DIRECTORY);
    endif;

    if (lResult = 0) then
        lResult = ParsePath(svName, szFullPath, FILENAME);
    endif;

    return lResult;
end;
```

メモ: *ユーザー定義関数は return ステートメントで値を戻すことが可能です。その他のデータ型は BYREF 演算子を使って宣言したパラメーターで戻すことが可能です。*

関数の呼び出し

すべての関数 (ユーザー定義関数、ビルトイン関数、Sd 関数、外部関数)は、同じ方法で呼び出されます。関数名を入力し、その次に関数に渡すパラメーターを入力します。例:

```
MyFunction (MyAge, MyHeight, MyWeight);
```

メモ: *InstallScript 関数では割り当てステートメントをパラメータとして渡すことができません。さらに && や || 演算子を関数への引数の中で使用することはできません。*

リファレンスによって関数に渡されるオートサイズ文字列変数は呼び出された関数の中では自動サイズ調整されません。関数が現在のパラメーターのサイズより大きい長さの値を割り当てようとすると、ランタイム エラー エラー401 が発生します。このエラーを回避するためには、リファレンスが関数に値を渡すときの特定の文字列サイズを宣言します。

DLL ファイル関数の呼び出し

InstallScript インストール スクリプトから DLL ファイル関数を呼び出す際に、注意する 3 つの規則があります。

- DLL ファイル名の長さは、最大 33 文字で、関数名は 63 文字です。

- .dll ファイルを呼び出す際、インストールは幅が 4 バイトを超える合成パラメーターを処理することができません。ただし、パラメーターを合成構造を指すポインターにすることは可能です。
- アドレスがポインター変数に保存されている文字列変数を、ポインターを DLL ファイル関数に渡して変更することはできません。DLL ファイル関数で文字列変数値を変更できるようにするには、BYREF 演算子を指定して引数のデータ型を宣言してから、変数そのものを引数として関数に渡します。

タスク　スクリプトから DLL ファイル関数をスクリプトから呼び出すには、以下の手順に従います:

1. DLL ファイルをサポート ファイルとしてプロジェクトに追加していない場合、まずこれを行います。詳細については、「サポート ファイルを追加する」を参照してください。

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. **InstallScript** エクスプローラーで、DLL 関数を呼び出す InstallScript ファイル (.rul) をクリックします。
3. スクリプトの先頭で、次の構文を使用して関数のプロトタイプを宣言します。

   prototype [CallingConvention] [ReturnType] DLLName.FunctionName(ParamType1, ParamType2, ...);

   例:

   prototype BOOL MyDLL.MyFunction(INT, INT, INT);

   呼び出し規則は cdecl または stdcall に指定できます。呼び出し規則を特に指定しなかった場合、InstallShield は stdcall を使用します。

   DLL ファイル名は大文字と小文字を区別することに注意してください。たとえば、スクリプト コンパイラは *MyDLL.MyFunction* および *mydll.MyFunction* を異なる関数として処理します。User32.dll、Gdi32.dll、または Kernel32.dll から関数のプロトタイプを宣言する際、USER、GDI、または KERNEL のキーワードを DLL 関数名の代わりに使用できます。

   戻り値には、LIST 以外のすべての InstallScript データ タイプを指定できます。ワイド文字の文字列引数または Unicode 文字列引数が予想される DLL ファイル関数呼び出しを宣言するとき、wstring データ タイプを利用してください。戻り値の型を指定しなかった場合、インストールは DLL ファイル関数が 4 バイト値を返すものと想定します。

4. **UseDLL** 関数を呼び出して DLL ファイルをロードします。例:

   UseDLL(SUPPORTDIR ^ "MyDLL.dll");

   メモ・ *_isuser.dll や _isres.dll、または User32.dll、Gdi32.dll、および Kernel32.dll などの Windows API DLL ファイルをロードする必要はありません。これらの DLL ファイルのロードまたはアンロードのために、* **UseDLL** *および* **UnUseDLL** *を呼び出さないで下さい。*

5. 他の関数と同様に、関数を呼び出します。例:

   bResult = MyDLL.MyFunction(nInt1, nInt2, nInt3);

   上記の例のように、関数呼び出しに DLL ファイル名を含めることをお勧めします。スクリプトが別の DLL から同じ名前の関数を宣言した場合、DLL 名を含まずに関数を呼び出した時にコンパイル エラーの原因となります。

   呼び出された関数を DLLファイルで検出できなかった場合、インストールは例外を発生します。これは try...catch...endcatch block を使って処理することができます。

6. DLL へのスクリプトの呼び出しがすべて完了すると、**UnUseDLL** を呼び出して DLL をアンロードします。例:

```
UnUseDLL( SUPPORTDIR ^ "MyDLL.dll" );
```

メモ: *IS_NULLSTR_PTR 変数を使って、ヌル ポインターを InstallScript 文字列としてプロトタイプされているパラメーターを通して 外部 DLL 関数または Windows API に渡すことができます。詳細については、「IS_NULLSTR_PTR」を参照してください。*

DLL ファイル関数に配列を渡す

InstallScript 配列は、内部的にはビルトイン C または C++ 配列ではなく OLE オートメーション セーフ配列としてフォーマットされます。InstallScript 配列を C/C++ 配列が必要な DLL 関数に渡すためには、GetCArrayFromISArray を呼び出すことによって配列データを要求フォーマットに配置しなければなりません。

関数の宣言

ユーザー定義関数を作成する最初のステップは、関数を宣言することです。キーワード *prototype* は、その行に関数定義があることを InstallShield のスクリプトコンパイラに通知します。

タスク ***関数を宣言するには、以下の手順を実行します。***

1. キーワード **prototype** を入力します。
2. 関数の戻り値タイプを入力します。(この手順はオプションです。戻り値タイプを入力しなかった場合、関数が NUMBER 値を戻す、あるいは値を戻さないものと判断します。)関数が値を戻さないようにするには、**void** を入力します。
3. 同じ行で、関数名を入力します。
4. 関数名の後、かっこ内にパラメーターのデータ タイプをコンマで区切って入力します。
5. パラメーターがない場合は、関数名の右側に空の括弧を挿入します。
6. セミコロン (;) で行を終了します。

次の例で、**FunctionName** は、3 つのパラメーターを含む関数です。**FunctionName** を呼び出す際に渡される引数は、順に INT、STRING、SHORT です。**CopyBitmapExample** にはパラメーターがありません。**FileTransfer** には 5 つのパラメーター (3 つの LONG 変数と 2 つの STRING 変数) があり、NUMBER 値を戻します。

```
prototype FunctionName (INT, STRING, SHORT);
prototype CopyBitmapExample( );
prototype NUMBER FileTransfer (LONG, LONG, LONG, STRING, STRING);
```

DLL 関数を宣言する際、DLL ファイル関数の名前に <DLL ファイル名>.<関数名> という形式を使用します。例:

```
prototype MyDLL.MyFunction (INT, INT);
```

上記の宣言は、プログラムが 2 つの INT パラメーターを使用して **Mydll.dll** というファイルにある **MyFunction** という関数を呼び出すよう、InstallScript コンパイラに通知します。

オプションとして、DLL ファイル関数を宣言するとき、呼び出し規則の cdecl または stdcall を宣言することもできます。例:

prototype cdecl MyDLL.MyFunction(INT, INT);

呼び出し規則を明示的に宣言しなかった場合、InstallShield は stdcall を使用します。呼び出し規則とり戻り値の型の両方を宣言する場合、戻り値の型の前に呼び出し規則を配置します。

メモ *Windows API 関数のほとんどは stdcall 呼び出し規則を利用しますが、C または C++ 開発環境の中には、C または C++ 関数を _stdcall 修飾子と共にプロトタイプ化しない限り、cdecl 呼び出し規則を使って DLL ファイル関数をビルドします。さらに詳しい情報は、コンパイラについての資料をご覧下さい。*

多くの Windows API 関数はヘッダーファイル ISRTWindows.h *で宣言され、*Ifx.h *スクリプトに含んだときに自動的に含まれます。(*[設定] *ダイアログ ボックスの* [コンパイル/リンク] *タブにある* [プリプロセッサ定義] *ボックス内にプリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を配置することで Windows API が自動的に定義されないようにすることが可能です。)*

関数から値を戻す

InstallScript に組み込まれた関数と同様に、ユーザー定義関数も呼び出し側に値を戻すよう設計することができます。関数から値を戻すには、関数の end ステートメントの前に return ステートメントとそれに続く値を挿入します。return ステートメントを含まない場合、またはキーワード return の後に値を指定しない場合、関数によって戻される値は予測できません。(関数プロトタイプが void 戻り値タイプを指定する場合、関数は値を戻すことができません。)

多くのプログラマは、return ステートメントを使用して、関数呼び出しが成功したか失敗したかを示すエラーコードを戻します。InstallScript のビルトイン関数の多くは、この目的のためだけに return ステートメントを使用します。また、return ステートメントは、関数に渡されたパラメーターに行われる処理の結果を戻す関数を作成するためにも使用されます。以下に例として、長方形の面積を戻す場合を示します。

```
function RectangleArea (nLength, nWidth)
    INT nVal;
begin
    nVal = (nLength * nWidth);
    return nVal;
end;
```

キーワード return の後には、定数、変数、数値、文字列式、または関数呼び出しを使用できます。以下の例では、**RectangleArea** を変更し、代入ステートメントを省いてキーワード return の後に算術式を使用しています。

```
function RectangleArea (nLength, nWidth)
begin
    return (nLength * nWidth);
end;
```

関数から戻された値は、呼び出し側のプログラムや関数で無視することもできますが、条件式でテストしたり変数に代入することもできます。以下の例では、**RectangleArea** からの戻り値を変数 *nArea* に割り当てます。

```
nArea = RectangleArea (nLong, nWide);
```

次の例では、**RectangleArea** の結果が条件式でテストされます。

```
if (RectangleArea(nLong, nWide) > nMaxArea) then
        MessageBox("範囲が制限を越えています。", INFORMATION);
    endif;
```

複数の値、または数値以外の値を戻すには、BYREF 演算子を使用して、参照から渡されるパラメーターを定義します。

サポートされていない関数

InstallShield Professional および InstallShield–Windows Installer Edition で利用可能な関数の一部は、InstallShield ではサポートされていません。

コンポーネント関数

InstallShield では、機能はエンド ユーザーの視点から見たプロジェクトで最も上位にあたる構成区分です。このため、従来のコンポーネント関数は、機能関数となります。たとえば、**ComponentDialog** は現在、**FeatureDialog** になっています。では、従来のコンポーネント関数のほぼすべてを機能関数として使用できます。

プロジェクト: *機能関数が使用できるのは、InstallScript および InstallScript MSIプロジェクトのみです。基本の MSI プロジェクトについては、インストールに機能関数を使用する場合、InstallScript MSI プロジェクト タイプにプロジェクトを変換する必要があります。*

PreShowComponentDlg および PostShowComponentDlg

InstallShield – Windows Installer Edition 2.x の場合、コンポーネント関連の関数 (現行の機能関連の関数) を使用するには、**PreShowComponentDlg** および **PostShowComponentDlg** 関数を呼び出す必要がありました。InstallShield の場合、機能関連の関数が使用できるのは、InstallScript および InstallScript MSI プロジェクトのみです。基本の MSI プロジェクトについては、スクリプトで機能関数を使用する場合、InstallScript MSI プロジェクトにプロジェクトを変換する必要があります。

機能関数は基本の MSI プロジェクトでの使用をサポートしていないため、**PreShowComponentDlg** および **PostShowComponentDlg** 関数は不要となり、サポートされなくなりました。

エントリポイント関数の書き込み

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript カスタム アクションを含む基本の MSI*
- *InstallScript カスタム アクションを含む InstallScript MSI*
- *InstallScript カスタム アクションを含むマージ モジュール*

Windows Installer が InstallScript カスタム アクションを実行する際に、InstallShield はカスタム アクションを作成したときに指定した関数を呼び出します。すべての InstallScript カスタム アクションは、スクリプトへのエントリポイントとして、エクスポートされたユーザー定義関数が必要です。

関数のプロトタイプ宣言と定義

エントリ ポイント関数は、他の関数と同じようにプロトタイプ宣言され、定義されます。ただし、この関数には以下の要件があります。

- プロトタイプには、エクスポートされた関数であることを宣言するキーワード export が含まれます。

- 関数の引数は 1 つだけで、引数は .msi データベースへのハンドルでなければなりません。
- カスタム アクションが戻り値を待つように設計されている場合は、Windows Installer に解釈できる値を戻します。上記のカスタム アクションの戻り値は IsMsiQuery.h に定義されており、スクリプトに IsMsiQuery.h または Iswi.h をインクルードすれば使用できます。

以下のスクリプト例では、正常終了した場合に成功値を戻すエントリポイント関数が宣言されています。

```
//ビルトイン InstallScript 関数プロトタイプの Isrt.h のインクルード
#include "isrt.h"

キャッシュには次の 2 種類あります:
#include "iswi.h"

export prototype MyFunction(HWND);

function MyFunction(hMSI)
    STRING szKey, svValue, svPath;
    NUMBER nvType, nvSize, nReturn;
begin
    RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);

    szKey = "SOFTWARE¥¥Microsoft¥¥Windows¥¥CurrentVersion¥¥App Paths¥¥isdev.exe";
    nReturn = RegDBGetKeyValueEx (szKey, "Path", nvType, svValue, nvSize);

    //App Paths キーは、InstallShield がインストールされたフォルダーと、
    //その後に続くセミコロンで構成されています。
    StrSub (svPath, svValue, 0, StrFind(svValue, ";"));

    if nReturn = 0 then
        MessageBox ("InstallShield のインストールされている場所 " + svPath, INFORMATION);
        return ERROR_SUCCESS;
    else
        MessageBox("InstallShield をインストールした場所が不明です。", SEVERE);
        return ERROR_INSTALL_FAILURE;
    endif;
end;
```

複数のエントリポイント

複数のカスタム アクションを実行するために、スクリプトに複数のエントリポイント関数を持たせることができます。ただし、InstallShield は各 InstallScript カスタム アクションに対して指定された 1 つのエントリポイント関数のみ呼び出します。

関数ウィザードで関数の呼び出しを追加する

関数ウィザードを使用して関数の選択、引数の指定、スクリプトへの関数の挿入を行うことができます。

宣言済みの Windows API 関数

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Windows API 関数はヘッダーファイル ISRT.h で宣言され、Ifx.h スクリプトに含むと自動的にインストールに含まれます。スクリプト コードでこの関数が明示的に宣言された場合、次のどれかを行ってください。

- 関数宣言を削除し、InstallShield が提供する宣言を使用します。
- InstallShield 2025 および InstallShield Professional 7 より前のバージョンの両方と互換性のあるスクリプトが必要な場合、次のようなコードを使って宣言を囲みます:

  #if _ISCRIPT_VER < 0x700prototype ...#endif

また、InstallShield が定義した宣言と利用する宣言とが異なる場合、Windows API 関数を呼び出すコードもアップデートする必要があります。

[設定] ダイアログ ボックスの [コンパイル/リンク] タブにある [プリプロセッサ定義] ボックス内にプリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を配置することで Windows API が自動的に定義されないようにすることが可能です。

デフォルト以外の機能イベント ハンドラー関数を指定する

デフォルトで、Setup.rul が InstallScript ビューに存在し、<機能名>_<イベント名> 関数を定義する場合 (例 **NewFeature1_Installing**)、その関数はその機能のイベントのハンドラーです。

機能イベントのハンドラーを別の関数を選択することも可能です。これには、InstallScript ビューの任意のスクリプト ファイルで定義された任意の関数を利用できます。ただし、引数を必要とせず次のようなエキスポートキーワードを利用して宣言されていない場合に限ります:

```
export prototype MyFeatureHandler();
```

タスク ***デフォルト以外の機能のイベント ハンドラー関数を指定するには、次の操作を行います:***

1. スクリプト ファイル (.rul ファイル) を **InstallScript** ビューで開きます。
2. 次のいずれかを スクリプト エディター ペインで実行します。
   - 機能イベント ハンドラーとして指定する関数を宣言および定義します。
   - スクリプト エディター ペインの上部にある [イベント カテゴリ] と [イベント] リストで、それぞれ機能とイベント ハンドラーを選択します。InstallShield は自動的にイベント ハンドラーを作成します (たとえば Feature1Installing)。

メモ: *機能名に括弧を使うと、デフォルト形式 <機能名>_<イベント名>の名前をもつイベント ハンドラー関数は正常にコンパイルしません。*

グローバル変数へのアクセス

プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

グローバル変数は、関数の外で宣言され、スクリプト中の任意のモジュールから使用できるデータです。グローバル変数を使用することにより、イベント ハンドラー、エントリ ポイント関数、ユーザー定義関数による持続データを共有できます。

たとえば、OnBegin イベント ハンドラーでオペレーティング システムのバージョン値を得た場合、エントリポイント関数でそのグローバル値にアクセスできます。次に引用したスクリプトは、この設定について説明しています。

```
//ビルトイン関数のヘッダー ファイルを含める
#include <isrt.h>
//イベント ハンドラーと MSI API のヘッダーファイルをインクルード
#include <iswi.h>

    //グローバル変数の宣言
    NUMBER nvResult;

    //エントリポイント関数のプロトタイプ宣言
    export prototype MyFn(HWND);

function OnBegin()
    //ローカル変数の宣言
    STRING svResult;
begin
    GetSystemInfo (WINMAJOR, nvResult, svResult);
end;

function MyFn(hMSI)
begin
    if nvResult > 4 then
        //Windows 4.0 より新しいバージョンのコード
    else
        //Windows 4.0 のコード
    endif;
end;
```

重要 *・グローバル変数の状態は、すべてのイベント ハンドラーで共有されます。ただし、プロジェクト用の InstallScript カスタム アクションを作成した場合、グローバル変数は InstallScript カスタム アクションからイベントドリブン型 InstallScript コードへ、またはその逆に状態が共有されることはありません。したがって、イベントドリブン型 InstallScript コードでグローバル変数を宣言した場合、InstallScript カスタム アクションでその変数を利用することはできません。同様に、InstallScript カスタム アクションで変数を宣言した場合、イベントドリブン型 InstallScript コードでその変数を利用することはできません。*

初期化 (.ini) ファイル エントリのアンインストール

アンインストールは、ログ記録が有効な状態で InstallScript 初期化ファイル関数を使って加えられた変更を削除します。さらに詳しい情報は、説明書のこのセクションをご覧ください。

初期化 (.ini) ファイル エントリのアンインストールについての一般制限事項

.ini ファイルエントリおよび変更点を自動的にアンインストールする際、適用される以下の制限事項が適用されます:

- .ini ファイル関数が呼び出されたときにログ記録が有効でなくてはなりません。
- .ini ファイルが .ini ファイル関数を使って作成された場合、それは削除されません。

- セクション名が .ini ファイル関数を使って作成された場合、キーが存在しなくてもそれは削除されません。
- InstallScript 関数によってキー値が (前後に追加されるのではなく) 完全に置き換えられている場合、インストールの前に存在したキー値は 修復されません。この場合、InstallShield は置き換えられたキーを新規に作成されたキーとみなし、インストールが作成した新しいキーと同様にアンインストールします。
- アンインストールは **WriteProfString** を使って削除したキーおよび値を修復しません。
- 既存するキーに値を追加するには (たとえば、[386Enh] の下にあるネットワーク)、新しいエントリは既存する文字列の終わり又は一番最初に追加しなくてはならず、文字列に挿入することはできません。
- コンマ (,) およびセミコロン (;) のみが有効な区切り文字です。アンインストールする文字列値が長い文字列の一部として検出された場合、区切り文字がコンマまたはセミコロンの場合のみ文字列の位置に基づいてその値および前後の区切り文字も適切に削除されます。たとえば、アンインストールが文字列 **Key=pqr,rst,uvw** から **pqr** を削除する場合、**pqr** の後に続くコンマも削除されます。コンマおよびセミコロン以外の文字は削除されません。基本的に、区切り文字の前後に空白を使わないようにしてください。

AddProfString の初期化 (.ini) ファイル エントリをアンインストールする

次の条件が満たされていると、アンインストールによってキー名と値のペアが完全に削除されます。

- **AddProfString** によってキーが正しく作成されている。
- アンインストール実行時に存在したキー名と値のペアがインストールされたキー名と値のペアと完全に一致する。インストールとアンインストールの間に、別のインストールまたはプログラムによってインストールされたキーが変更されると、キー名と値のペアは完全に削除されません。元のインストールによって作成された値だけが削除され、キー自体とその他の値はアンインストーラーによって削除されません。

元の System.ini ファイルが次のような場合 :

[386Enh]

device=votc.386

device=*vmpcd

AddProfString を呼び出した後は次の様になります :

[386Enh]

device=*vmp32

device=votc.386

device=*vmpcd

アンインストールによって device=*vmp32 が削除されます。

別のインストールが既存の行に値を追加した場合、アンインストール中のインストールからの値のみが削除されます。オリジナルキー名および新しい値はファイルに残ります。たとえば、[Test] セクション内の Test.ini に **Test Values=votc.386** 行を追加した場合は、以下のとおりです。

[Test]

Test Values=votc.386

Continuous Test=No

また、インストールの実行後に別のインストールが Test Values に新しい値を追加している場合は、以下のとおりです。

[Test]

Test Values=votc.386,pctcp.386

Continuous Test=No

アプリケーションがインストールされるとき、votc.386 が削除されて TestValues=pctcp.386 がファイル内に残ります。

[Test]

Test Values=pctcp.386

Continuous Test=No

ReplaceProfString の初期化 (.ini) ファイル エントリをアンインストールする

次の条件が満たされていると、アンインストールによってキー名と値のペアが完全に削除されます。

- **ReplaceProfString** によってキーが正しく作成されている。
- 以前、キーが存在していなかった。
- アンインストール実行時に存在したキー名と値のペアが、インストールされたキー名と値のペアと完全に一致する。インストールとアンインストールの間に、別のインストールまたはプログラムによってインストールされたキーが変更されると、キー名と値のペアは完全に削除されません。元のインストールによって作成された値だけが削除され、キー自体とその他の値はアンインストーラーによって削除されません。

元の System.ini ファイルがこのような場合 ;

[386Enh]

device=votc.386

device=*vmpcd

ReplaceProfString を呼び出した後は、以下のようになります。

[386Enh]

device=*vmp32

device=votc.386

device=*vmpcd

アンインストールによって device=*vmp32 が削除されます。

WriteProfString の初期化 (.ini) ファイル エントリをアンインストールする

次の条件が満たされていると、アンインストールによってキー名と値のペアが完全に削除されます。

- **WriteProfString** によってキーが正しく作成されている。
- 以前、キーが存在していなかった。
- アンインストール実行時に存在したキー名と値のペアが、インストールされたキー名と値のペアと完全に一致する。インストールとアンインストールの間に、別のインストールまたはプログラムによってインストールされたキーが変更されると、キー名と値のペアは完全に削除されません。元のインストールによって作成された値だけが削除され、キー自体とその他の値はアンインストーラーによって削除されません。

COM オブジェクトを使用してインストールを拡張する

重要・ *COM オブジェクトと InstallShield オブジェクトは混在しやすいので注意してください。2 つのオブジェクトはまったく別の機能です。*

DLL または他の実行可能ファイルの代わりに、COM オブジェクトを使用してインストールを拡張することができます。COM オブジェクトは、比較的簡単にインストールに統合することができます。次の手順に従って COM オブジェクトをスクリプトに割り当てます。

タスク ***COM オブジェクトを割り当てるには、以下の手順に従います:***

1. オブジェクト変数を宣言します。例:

   ```
   OBJECT oMyCOMObject
   ```

2. COM への参照を取得して、SET キーワードを使用して、それを変数に割り当てます。例:

   ```
   set oMyCOMObject = CreateObject ( szMyProgID );
   ```

szMyProgID の値が、COM オブジェクトの有効なプログラム ID になります。(SET キーワードを省略すると、スクリプト エンジンは oMyCOMObject のデフォルト プロパティを szProgID に設定します。デフォルトとなるプロパティがないため、oMyCOMObject はCOM オブジェクトへの参照を含まないので、スクリプトは失敗します。)

重要・ *デフォルトでは、インストールが* **CoCreateObject**、**CoCreateObjectDotNet**、**CoGetObject**、*または* **DotNetCoCreateObject** *を使って COM オブジェクトをロードした後、インストールが終了するまでロードされた状態が保たれます。COM オブジェクトは、COM オブジェクト変数がスコープ外に出るか、NOTHING に設定されているかに関わらず、ロードされた状態に保たれます。*

そのため、COM オブジェクトが参照するライブラリもインストールが終了するまでリリースされません。ただし、Windows API の ***CoFreeLibrary*** *を呼び出して COM オブジェクトが参照するライブラリをアンロードすることは可能です。* ***CoFreeLibrary*** *は、ライブラリがオブジェクト変数によって参照されている状態かどうかに関わらず、これをアンロードします.したがって、ライブラリを参照するすべてのオブジェクトがスコープ外に出た後、または これらが NOTHING に設定されている場合に* ***CoFreeLibrary*** *を呼び出します。次のサンプル コードを参照してください。*

```
prototype void ole32.CoFreeLibrary( byval int );
int     hModule;

program

   //ライブラリを開放します。
   hModule = GetModuleHandle( "CSharpClassLibrary.dll" );

   if( !hModule ) then
      MessageBox( "モジュール ハンドルを取得できませんでした", INFORMATION );

   else

      CoFreeLibrary( hModule ); // ライブラリをリリースします

   endif;
```

endprogram

***CoFreeUnusedLibraries** を使ってライブラリをアンロードすることもできます。ただし、デフォルトではライブラリのアンロードを行う前にシステムが 10 分間待機するように設定されています。ほとんどの場合、インストールはライブラリをすぐにアンロードする必要があります。インストールが Windows XP 以降でのみ実行される場合、**CoFreeUnusedLibrariesEx** を呼び出すことができます。これにはライブラリをすぐにアンロードするオプションが含まれます。*

以下は、シリアル番号を検証する COM オブジェクトの例です。Validate というメソッドと、IsEval というプロパティがあります。

```
function OnFirstUIBefore()
    OBJECT oObj;
    STRING szProgID, szMsg, szTitle, svName, svCompany, svSerial;
    BOOL bValidSerialNumber, bEval, bValidate;
    NUMBER nResult;
begin
    szProgID = "MySerialValidation";
    set oObj = CreateObject( szProgID );
    if ( !IsObject( oObj ) ) then
        MessageBox( "オブジェクト " + szProgID + " は無効です。", SEVERE );
        return ISERR_GEN_FAILURE;
    endif;

    bValidSerialNumber = FALSE;
    while (!bValidSerialNumber)
        szMsg = "名前、会社名、および製品のシリアル番号を入力してください。";
        szTitle = "情報の入力";
        nResult = SdRegisterUserEx( szTitle, szMsg, svName, svCompany, svSerial );
        if ( nResult < 0 ) then
            MessageBox( "ダイアログ ボックスを表示できませんでした", INFORMATION );
            return ISERR_GEN_FAILURE;
        endif;

        /* オブジェクトの IsEval プロパティの値を確認します。*/
        bEval = oObj.IsEval;
        /* オブジェクトの Validate メソッドを実行して
        メソッドからの戻り値を取得します。*/
        bValidate = oObj.Validate( svSerial );
        if ( bEval || bValidate ) then
            bValidSerialNumber = TRUE;
        else
            MessageBox( "シリアル番号が無効です。もう一度入力してください。", INFORMATION );
        endif;
    endwhile;

    return ISERR_SUCCESS;
end;
```

***メモ:** COM オブジェクトが期待するメンバーがあれば、データ構造を COM オブジェクトに渡すことができます。次は、データ構造を渡す例です。*

```
#define PERSON_NAME_SIZE 1024
```

```
typedef PERSON
    begin
        STRING Name[PERSON_NAME_SIZE];
        NUMBER Age;
    end;

function OnBegin()
    PERSON pPerson;
    NUMBER nPhoneNumber;
    OBJECT oMyCOMObject;
    STRING szMyProgID;
begin
    /* この行で szMyProgID に値を割り当てます。*/
    set oMyCOMObject = CreateObject ( szMyProgID );
    if ( !IsObject( oMyCOMObject ) ) then
        MessageBox( "オブジェクト " + szMyProgID + " は無効です。", SEVERE );
        return ISERR_GEN_FAILURE;
    endif;
    nPhoneNumber = oMyCOMObject.GetPhoneNumber( pPerson );

    return ISERR_SUCCESS;
end;
```

メモ: 標準の COM 戻り値 HRESULT は、InstallScript などのオートメーション クライアントには表示されません。COM オブジェクト メソッドから値を戻すには、次のサンプル コードで説明されているように、メソッドのパラメーター リストに [out,retval] パラメーターが必要です。

IDL:

```
interface IMyInterface : IDispatch
{
    [id(1)] HRESULT DoSomeWork([in] long nInputValue, [out,retval] long *pReturnValue);
}
```

C++:

```
STDMETHODIMP MyInterfaceImpl::DoSomeWork(long nInputValue, long *pReturnValue)
{
    // 実装コード
    *pReturnValue = 1; //クライアントに値を戻します
    return S_OK;
}
```

InstallScript:

```
function OnBegin()
    NUMBER nObjReturn;
begin
    set oMyCOMObject = CreateObject("MyProgID");
    if ( !IsObject( oMyCOMObject ) ) then
        nObjReturn = oMyCOMObject.DoSomething(10);
        /*nObjReturn には、実装から戻された値 1 が含まれています*/
    endif;
end;
```

関数呼び出しで機能やサブ機能を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

*機能*とは、InstallShield で一連のコンポーネントまたはサブ機能を参照する一般的な名称です。*サブ機能*とは別の機能の下にある機能で、フォルダーとサブフォルダーの関係に似ています。

トップレベルの機能とは、階層の最も上にある機能です。トップレベルの機能がサブ機能と呼ばれることはありません。

InstallScript コード内の機能およびサブ機能の参照方法

一部の機能関数およびセットアップの種類 ダイアログ関数では、単一の機能を参照し、他では複数の機能を参照します。

単一機能の参照

1 つの機能を参照するには、機能名を使用します。サブ機能を参照するには、パスと同様に階層順に各機能を 2 つのバックスラッシュで区切って記述します。たとえば、トップレベルの機能 **Help Files** 内にあるサブ機能 **Tutorials** を指定するには、インストール スクリプトで以下の式を使用します。

```
szFeature = "Help Files¥¥Tutorials";
```

Tutorials 内にあるサブ機能 **CBT** を参照するには、以下を使用します。

```
szFeature = "Help Files¥¥Tutorials¥¥CBT";
```

機能の名前に円記号は使用できません。

複数機能の参照

InstallScript MSI インストールでは、**SdFeatureMult** などの機能およびセットアップの種類ダイアログ関数で、複数の機能およびそれらのサブ機能が表示されることもあります。その場合、階層のすぐ上の機能を指定して、複数の機能を参照します。複数のトップレベルの機能を参照するには、ヌル文字列 ("") を使用します。

たとえば、**SdFeatureMult** 関数にヌル文字列を渡すと、MEDIA システム変数の値により、**SdFeatureMult** ダイアログの左のウィンドウに設定されているスクリプト作成コンポーネントのトップレベル機能がすべて、対応するダイアログに表示されます。すべてのサブ機能は、このダイアログの右側のウィンドウに表示されます。

Windows API 関数を呼び出す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

Windows API 関数は、InstallScript コード内から呼び出すことができます。

タスク ***Windows API 関数を呼び出すには、以下の手順を実行します。***

1. スクリプトの先頭で、次の構文を使用して関数のプロトタイプを宣言します。

   prototype ReturnType DLLName.FunctionName(ParamType1, ParamType2, ...)

   例:

   prototype INT User.LoadString(INT, SHORT, BYREF WSTRING, INT);

   Dumpbin.exe を /EXPORTS オプションと共に使用して、Windows API DLL Kernel32.dll、User32.dll、および GDI32.dll のいずれかにある関数を判別します。関数の戻り値の型およびパラメーターのタイプを判別するには、MSDL (Microsoft Development Library) などの Windows プログラミング リファレンスを参照してください。

   メモ・戻り値の型を、InstallScript データ型 BOOL、CHAR、HWND、INT、LONG、 LPSTR、NUMBER、POINTER、または SHORT に指定することができます。戻り値の型が指定されなかった場合、InstallShield は DLL 関数が 4 ビットの値を戻したと仮定します。

2. 他の関数と同様に、関数を呼び出します。例:

   nResult = LoadString(iInstance, nStringID, szMyString, MAX_SIZE);

3. 関数が失敗した際、より詳しいエラー情報が必要な場合、Err オブジェクトの LastDllError プロパティを確認してください。(セットアップは制御をスクリプトに戻す前にこの値をそれ自身で変更するため、Windows API 関数 **GetLastError** を呼び出して詳しいエラー情報を取得することはできません。)

カスタム ファイルの転送操作を埋め込む

InstallShield では、インストールの標準ファイル転送中に追加のファイル転送操作を埋め込むことができます。進行状況バーも、これらの操作中にスムースかつ適切に更新されます。この機能の使用に関して、いくつかの異なる一般的なシナリオがあります。

以下は、各シナリオで呼ばれる機能の概要を説明したテーブルです。

テーブル 3・共通カスタム ファイルの転送操作に適切な関数を判別する

シナリオ	呼び出す関数
FeatureMoveData	なし
XCopyFile または CopyFile	FeatureAddCost
LaunchApplication(外部アプリケーション)	FeatureAddCost、FeatureSpendCost、FeatureAddUninstallCost、または FeatureSpendUninstallCost

これらの手続きには、各シナリオのカスタムファイル転送操作の利用に関する手順を追っての説明が含まれています。

タスク ***標準ファイル転送操作中に、XCopyFile または CopyFile を使用して追加のファイルをインストールする場合、以下の手順を実行します。***

1. ファイル転送前に、ファイルの転送先フォルダーのクラスタサイズを考慮して、インストールされるファイルのサイズを判別します。ファイルのサイズは、**GetAndAddFileCost**、**CalculateAndAddFileCost**、および/または **GetAndAddAllFilesCost** 関数を使用することで判別できます。
2. **FeatureAddCost** 関数を利用して、手順 1 で判別したコストをこれらのファイルが関連付けられている機能のコストへ加えます。すべてのコストは特定の機能に関連付けられています。したがって、インストールされるファイルが既存の機能に関連付けられていない場合、サンプル機能を作成するか、既存の機能を使用する必要があります。
3. ファイル転送中、**XCopyFile** 関数を呼び出して、ファイルをインストールします。通常、これは追加のファイルに関連付けられている機能のイベントのインストール中、または OnMoving または OnMoved イベントで実行します。進行状況バーも、呼び出し中にスムースかつ適切に更新されます。XCopyFile 操作中、ステータス ダイアログの [キャンセル] ボタンを無効にする必要があります。詳細については、XCopyFile 関数の説明を参照してください。

メモ: *アンインストールに必要な追加のアクションはありません。ファイルは、アンインストール進行状況が適切に更新されている間にアンインストールされます。*

タスク ***標準ファイル転送操作中に、追加のファイルをインストールする外部インストールを呼び出す場合、以下の手順を実行します。***

1. ファイル転送前に、外部インストールを検査し、発生する操作を調べて、外部インストールによってインストールされるファイルのサイズおよび数 (または他の操作) を判別します。
2. **FeatureAddCost** 関数を利用して、手順 1 で判別したコストをこのインストールが関連付けられている機能のコストへ加えます。すべてのコストは特定の機能に関連付けられています。したがって、インストールされるファイルが既存の機能に関連付けられていない場合、サンプル機能を作成するか、既存の機能を使用する必要があります。以下の 2 つのシナリオを確認してください。
3. 外部インストールがアンインストール中に呼び出された場合、**FeatureAddUninstallCost** を利用して操作の数および操作のサイズをアンインストールのコストに加えます。
4. ファイル転送中、**LaunchApplication** イベントを使用してインストールを起動 (通常、サイレントモード) します。通常、これは追加のファイルに関連付けられている機能のイベントのインストール中、または OnMoving または OnMoved イベントで実行します。スクリプトが、インストールおよびアンインストール中に、定期的に制御を再取得するように LAAW_USE_CALLBACK オプションを利用します。
5. **OnLaunchAppAndWaitCallback** イベントをオーバーライドし、インストールの実行をポールするコードを追加して進行状況を判別します。それから、**FeatureSpendCost** (インストール) 関数または **FeatureSpendUninstallCost** (アンインストール) 関数を呼び出して、インストールが完了した適切なコストを使用します。この処理を、外部インストールが実行されている間繰り返します。進行状況バーも、呼び出し中にスムースかつ適切に更新されます。**OnLaunchAppAndWaitCallback** イベントが適切な回数呼ばれて進行状況バーがスムースに更新されるように、LAAW_PARAMETERS.nCallbackInterval を調整しなければならない場合がありますので注意してください。

InstallScript で大きい数値を表現する

ある数値が InstallScript 整数で格納される際、31 ビットデータだけが値の格納に使用することができます。したがって、表現可能な最大値は 2^31 (2GB) となります。InstallScript 整数は、常に符号付きと見なされるため、32 ビット目は符号ビットとして解釈され、数値データの格納に使用することはできません。この結果、符号なしデータ型で格納が可能な 2^32 (4GB) の代わりに 2^31 (2GB) に表現することができる値が制限されます。

場合においては、より大きい値を指定できることが必要もしくは望ましいことがあります。これを InstallShield で実現するため、いくつかの関数では、大きい数値の指定を各数値の符号ビット (32 ビット) を持つ 1 組の 32 ビットの数値として指定できます。符号ビットは常に正の数値を示すように 0 が設定されます。このフォーマットを使用することにより、数値を 62 ビットまでのデータまたは 2^62 までのサイズと共に指定または取得することができます。

ただし、この言語は数値を単一変数として表現できるデータ型を持たないため、このデータに実行可能な操作は限られています。このデータは通常、これらの値をサポートする関数の間でやり取りされます。データはまた、**ConvertSizeToUnits** 関数を利用してより大きいサイズ単位 (KB または MB など) で表現された単一値に変換することも可能ですし、より大きいデータ型を処理できる外部 DLL 関数に渡すことも可能です。

ある数値が上位/下位の 1 組で指定された場合、下位 32 ビットの値は、完全値の下位 31 ビットデータを指定します (符号ビットは常に 0 に設定され無視されます) (符号ビットは常に 0 に設定され無視されます)。高位 32 ビット値は、高位 31 ビット データ (同様に符号ビットは常に 0 に設定され無視されます) を指定します。たとえば、4 の上位値は実際、4 * 2^31 または 8589934592 (8 GB) サイズを示します。これが、下位値 100 と組み合わさって、合計サイズ 8589934692 単位を示します。

数値が 2^31 未満の場合、高値は必ず 0 となり、値のサイズが 2^31 を超えないと明らかな場合は、これを無視することができます。ただし、このデータを計算に使用する必要がある場合、**ConvertSizeToUnits** 関数を呼び出して、このデータを単一値で表現するとき四捨五入したときの誤差が最小になるように最も適した単位の単一値に変換することを推奨します。

メモ: *下位値 (low-value) が下位 32 ビットデータを格納し、上位値 (high value) がデータの上位 32 ビットを格納する Windows によって使用される符号なし上位/下位値の組み合わせとは規則が異なりますので注意してください。* **ConvertWinHighLowSizeToISHighLowSize** *関数は、Windows API 関数によって戻されるデータの簡易変化用に提供されています。*

次の関数は、上位/下位値の組み合わせをサポートしています。

- GetDiskInfo
- GetFileInfo
- GetAndAddFileCost
- CalculateAndAddFileCost
- GetAndAddAllFilesCost
- FeatureFileInfo
- FeatureGetCostEx
- FeatureAddCost
- FeatureSpendCost
- ConvertWinHighLowSizeToISHighLowSize

- ConvertSizeToUnits

デバイス ドライバーのインストール

InstallShield では、DIFxAPI コンポーネントを含む Microsoft Windows Driver Install Framework (DIFx) を使って InstallScript インストール内のデバイス ドライバーをインストールことがサポートされています。このコンポーネントがあると、Windows Installer を使わずにドライバーをインストールまたはアンインストールすることができます。

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。デバイス ドライバーのインストールは、DIFx Windows Installer 機能を使用した InstallScript MSI および基本の MSI プロジェクトでサポートされています。これらのプロパティの使用方法については、「デバイスドライバー設定を構成する」を参照してください。*

タスク ***InstallScript プロジェクトに 32 ビット DIFx ドライバーをインストールするには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"DIFx (32 ビット プラットフォーム用)"** 設定で **[有効]** を選択します。
3. DIFx ドライバー ファイルをプロジェクトに追加します。
4. DIFx ドライバー ファイルを含む機能の機能イベントをオーバーライドし、プラグ アンド プレイ (PnP) ではないドライバー用の **DIFxDriverPackageInstall** 関数、または、PnP 用の **DIFxDriverPackagePreinstall** 関数を呼び出します。
5. インストールのビルドと実行を行います。

タスク ***InstallScript プロジェクトに 64 ビット DIFx ドライバーをインストールするには、以下の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"DIFx サポート (64 ビット Itanium プラットフォーム用)"** 設定または **"DIFx サポート (64 ビット AMD プラットフォーム用)"** 設定で **[有効]** を選択します。
3. DIFx ドライバー ファイルをプロジェクトに追加します。
4. DIFx ドライバー ファイルを含む機能の機能イベントをオーバーライドし、プラグ アンド プレイ (PnP) ではないドライバー用の **DIFxDriverPackageInstall** 関数、または、PnP 用の **DIFxDriverPackagePreinstall** 関数を呼び出します。
5. インストールのビルドと実行を行います。

コンパイラ バージョンの確認

InstallShield では、_ISCRIPT_VER は 0x*VVM*、製品のメジャー バージョンは *VV*、メンテナンス パック リリース番号は *M* と定義付けられています。たとえば、0x900 は InstallShield DevStudio バージョン 9.0 のメンテナンスパック 0 (最初のリリース) と評価します。このプリプロセッサ定数は、InstallShield Professional 7 では 0x700、InstallShield Professional 6 では 0x600 と定義されていますが、InstallShield Professional 5.x では定義されておらず、ゼロと評価されます。

この定数は、スクリプトに以下のようなコードを含むことによって、InstallShield コンパイラが使用されているかどうかをテストするのに利用することができます。このコードは InstallShield DevStudio (バージョン 9.0、メンテナンスパック番号 0) またはそれ以降の最初のリリースが存在するかどうかをテストします。

```
#if _ISCRIPT_VER >= 0x900
   MessageBox("DevStudio 9.x とそれ以降",INFORMATION );
#elif _ISCRIPT_VER >= 0x700
   MessageBox ( "これは IS 7.x コンパイラです。", INFORMATION );
#else
   MessageBox ( "これは IS 6.x 以前のコンパイラです。", INFORMATION );
#endif
```

下位互換性のために保持されているプリプロセッサ定数

プロプロセッサ定数 _ISCRIPT_ISPRO と _ISCRIPT_ISDEV は下位互換性のために保持されています。_ISCRIPT_ISPRO は InstallScript プロジェクト (および InstallShield Professional プロジェクト) 用に定義されています。_ISCRIPT_ISDEV は InstallScript MSI プロジェクトと基本の MSI プロジェクト用に定義されています。

_ISCRIPT_ISDEV と _ISCRIPT_ISPRO を使用してオーサリング環境を確認する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

プリプロセッサ定数 _ISCRIPT_ISPRO は InstallScript プロジェクト (そして InstallShield Professional プロジェクト) では定義されていますが、InstallScript MSI プロジェクトや基本の MSI プロジェクトでは未定義となり、ゼロ評価します。プリプロセッサ定数 _ISCRIPT_ISDEV は InstallScript MSI や基本の MSI プロジェクトでは定義されていますが、InstallScript プロジェクト (そして InstallShield Professional プロジェクト) では未定義となり、ゼロ評価します。

_ISCRIPT_ISDEV と _ISCRIPT_ISPRO を使って、異なるプロジェクトの種類で別々の動作を行う単一のスクリプトを書くことができます。そのためには次のよなコードをスクリプトに含みます。

```
#ifdef _ISCRIPT_ISPRO
   //  InstallShield Professional と InstallScript プロジェクトに特有のコード
#else
   #ifdef _ISCRIPT_ISDEV
      // 基本の MSI プロジェクトと InstallScript MSI プロジェクトに特有のコード
   #endif
#endif
```

別の InstallScript インストールからインストールを起動する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

LaunchApplication 関数を呼び出して完全インストールを起動することができます。たとえば、子インストールのファイルを CD-ROM に配置した場合、次のように **LaunchApplication** を呼び出すことができます。

LaunchApplication (SRCDISK ^ "Launched Setup Folder¥¥Setup.exe", "", "", SW_HIDE,"", LAAW_OPTION_WAIT);

その他、プロジェクト内の子インストールをサポート ファイルとして含めることもできます。この方法を使うと、メイン インストールが子インストールをターゲット システムにコピーし、インストールを実行し、またその他のサポート ファイルと一緒にインストールを削除します。子インストールのパスで SUPPORTDIR 変数を使用します:

LaunchApplication (SUPPORTDIR ^ Setup.exe, "", "", SW_HIDE,"", LAAW_OPTION_WAIT);

また、子インストールへのリンクをファイル グループに挿入して、子インストールをターゲット システムにインストールしてから、ターゲット先からインストールを起動することもできます。(これらの方法を使った場合、親インストールがアンインストールされるまで子インストールがターゲット システムに残ります。)

InstallScript の言語サポート

次のテーブルは、InstallShield の Premier edition によってサポートされている言語です。以下は、各列の説明です。

- **InstallShield 言語** – この言語を参照するために InstallShield インターフェイスで使用される名前。
- **InstallScript 定数** – 言語固有のコンポーネントをフィルターするために InstallShield で提供されている言語定数。
- **Windows 日本語版での名前** – Windows (日本語版) でこの言語に使用される名前。

テーブル 4 ・ サポートされている言語

InstallShield 言語	InstallScript 定数	日本語版での名前
バスク語	ISLANG_BASQUE	バスク語
ブルガリア語	ISLANG_BULGARIAN	ブルガリア語
カタロニア語	ISLANG_CATALAN	カタロニア語
中国語 (簡体字)	ISLANG_CHINESE_SIMPLIFIED	中国語 (簡体字)
中国語 (繁体)	ISLANG_CHINESE_TRADITIONAL	中国語 (繁体)
クロアチア語	ISLANG_CROATIAN 下位互換性の目的で、この定数は (セルビア語との関連性から) より論理に適った 0x041a ではなく 0x001a のままです。ISLANG_CROATIAN_STANDARD ではなくこの定数を利用し続けてください。	クロアチア語
チェコ語	ISLANG_CZECH	チェコ語
デンマーク語	ISLANG_DANISH	デンマーク語
オランダ語	ISLANG_DUTCH	オランダ語 (標準)

テーブル 4 • サポートされている言語 (続き.)

InstallShield 言語	InstallScript 定数	日本語版での名前
英語	ISLANG_ENGLISH_UNITEDSTATES	英語 (U.S.)
フィンランド語	ISLANG_FINNISH	フィンランド語
フランス語(カナダ)	ISLANG_FRENCH_CANADIAN	フランス語(カナダ)
フランス語 (標準)	ISLANG_FRENCH_STANDARD	フランス語 (標準)
ドイツ語	ISLANG_GERMAN	ドイツ語 (標準)
ギリシャ語	ISLANG_GREEK	ギリシャ語
ハンガリー語	ISLANG_HUNGARIAN	ハンガリー語
インドネシア語	ISLANG_INDONESIAN	インドネシア語
イタリア語	ISLANG_ITALIAN	イタリア語 (標準)
日本語	ISLANG_JAPANESE	日本語
韓国語	ISLANG_KOREAN	韓国語
ノルウェー語	ISLANG_NORWEGIAN	ノルウェー語 (ブークモール)
ポーランド語	ISLANG_POLISH	ポーランド語
ポルトガル語(ブラジル)	ISLANG_PORTUGUESE_ BRAZILIAN	ポルトガル語(ブラジル)
ポルトガル語(標準)	ISLANG_PORTUGUESE_STANDARD	ポルトガル語(標準)
ルーマニア語	ISLANG_ROMANIAN	ルーマニア語
ロシア語	ISLANG_RUSSIAN	ロシア語
セルビア語	ISLANG_SERBIAN_CYRILLIC	セルビア語 (キリル)
スロバキア語	ISLANG_SLOVAK	スロバキア語
スロヴェニア語	ISLANG_SLOVENIAN	スロベニア語
スペイン語	ISLANG_SPANISH	スペイン語 (トラディショナル ソート)
スウェーデン語	ISLANG_SWEDISH	スウェーデン語
タイ語	ISLANG_THAI	タイ語

テーブル 4 • サポートされている言語 (続き.)

InstallShield 言語	InstallScript 定数	日本語版での名前
トルコ語	ISLANG_TURKISH	トルコ語

色化けの防止

InstallScript カスタム アクションで色化けの問題が生じている場合は、カラーパレットの転換によるものがほとんどです。このセクションでは、Windows システムに色が割り当てられる方法、およびインストールのエンドユーザー インターフェイスのひずみを最小限にするためのトラブルシューティングのヒントについて説明します。

ビデオドライバーによってサポートされている色の数にかかわらず、256 色モードで動作するシステムでは 256 色のカラーパレットで、16 色モードで動作するシステムでは 16 色のカラーパレットで、可能な色がすべて表示されます。High Color モードまたは True Color モードで動作するシステムでは、カラーパレットは使用されません。色は直接表示されるので、これらのシステムで色化けが起きる心配はありません。カラーパレットを使用するシステムでは、ビットマップなどのオブジェクトが表示される場合など必要に応じてエントリが割り当てられます。

現在のカラーパレットが割り当てられた後では、新しい色を表示するために Windows は既に割り当てられた色と類似した色を使おうとします。そのため、256 色システムでは、複数の異なる色からなる複数のオブジェクトが同時に表示されると色化けが生じる場合があります。

また、Windows はスタートアップ時に 16 の静的カラー (VGA カラー) に 4 つのグレーのシェードを加えて割り当てます。システムはこれらの色を使用して 16 色イメージを表示し、これらの色の割り当てが Windows によって解除されることはありません。カラーパレット操作の詳細については、Windows プラットフォーム SDK を参照してください。

以下のヒントを利用すると、カラーパレット転換に関する色化けを最小限に抑えることができます。カラーパレットのヒントは、インストール時に表示する背景色、.avi、メタファイル、ビットマップなどのすべてのイメージに適用できます。

メインインストールウィンドウの背景

- 1 つのパレットエントリしか必要としない単一色の背景を使用してみます。単一色背景定数を使用して **SetColor** 関数を呼び出し、背景色を作成します。
- 16 のパレットエントリのみを必要とする 16 色のグラデーション背景を使用してみます。**SetColor** 関数をグラデーション色の定数の 1 つと共に呼び出し、このタイプの背景を作成します。
- ビットマップ背景を並べて表示、または中央に表示して使用してみます。**PlaceBitmap** 関数を呼び出して、これらの背景を 1 つ作成します。
- 約 80 のパレットエントリを必要とする 256 色のグラデーションの背景は使用しないでください。

ビットマップとメタファイル

- **PlaceBitmap** 関数を REMOVE オプションと共に呼び出してビットマップを削除すると、そのビットマップが使用していたカラー パレット エントリのうち、現在表示されている他のイメージが使用していないすべてのエントリが、別のオブジェクトを表示するために他の色が必要になった時に、カラーパレットから解放されます。色化けが生じている場合は、ビットマップが表示されなくなったら、(別のビットマップで隠すのではなく) そのビットマップを必ず削除し、カラーパレットエントリが解放されるようにしてください。

- インストール中に表示されるビットマップやビルボードに対して、類似色 (常に割り当てられている 16 の静的カラーを含む) を使用して、必要なカラーパレットエントリの数を減らします。
- COLORS オプションを使用して **GetSystemInfo** 関数を呼び出し、ターゲット システムで使用できる最大の色数を決めることができます。戻り値に基づいて、異なる解像度のビットマップを表示することができます。
- 24 ビットのビットマップには、ビットマップファイルのカスタムカラーパレットが含まれません。256 色のシステムで 24 ビットのビットマップを表示している場合、使用できる追加のカラーパレットエントリがある場合でも、現在使用可能なパレットエントリだけが使用されます。インストールで 24 ビットのビットマップを表示し、256 色のシステムで実行する場合、256 色バージョンのビットマップも含めることをお勧めします。
- メタファイルは配置されるのではなく作成されるため、メタファイルが作成される際に、追加のカラーパレットエントリは割り当てられません。つまり、メタファイルの表示には、既存のカラーパレットエントリだけが使用されます。インストールを 256 色のシステムで実行する場合、標準の 16 の静的カラーは現在のカラーパレットのエントリにかかわらず使用できるため、これらの色だけをメタファイルで使用することをお勧めします。
- システムのビデオ ドライバーが 256 色以上の表示をサポートしていることを確認します。ビデオカードとモニターが 256 色表示をサポートしている場合でも、ビデオドライバーが 256 色表示をサポートしていない場合は、ビットマップは適切に表示されません。
- モニターで 256 色のビットマップが表示されない場合、16 の静的カラーだけが使用されます。このため、大幅な色化けが生じる可能性があります。インストールで 256 色のイメージを使用している場合は、256 色をサポートしているシステムでセットアップを実行するようエンドユーザーに要求することをお勧めします。

カスタム アクションを使用

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShieldでは、カスタム アクションを使った InstallScript、VBScript、または JavaScript コードの実行、DLL 関数の呼び出し、実行可能ファイルの実行、マネージ アセンブリ内にあるマネージ メソッドの呼び出し、プロパティやディレクトリの設定、エラーのトリガとインストールの中止、PowerShell スクリプトの実行、プロセスの終了、他のインストール パッケージの実行がサポートされています。

タスク ***InstallShield におけるカスタム アクションの作成および実行は、以下のステップで進めます。***

1. [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー)、またはカスタム アクション ウィザードを使用して、カスタム アクションを作成します。
2. カスタム アクションを実行するタイミングを決めます。

3. カスタム アクションを、シーケンスに挿入するかダイアログのコントロール イベントの結果として配置して実行します(基本の MSI プロジェクト)。

様々な機能をサポートするために InstallShield プロジェクトに自動的に追加される各 InstallShield カスタム アクションについての詳細は、「InstallShield カスタム アクション リファレンス」を参照してください。

プロジェクト: *マージ モジュール プロジェクトのみ、ダイレクト エディターで* ModuleInstallExecuteSequence *テーブルを変更して、マージ モジュール内のカスタム アクションの起動を制御することができます。インストール プロジェクトにマージ モジュールを追加する際、マージ モジュールの中に含められたすべてのカスタム アクションおよびダイアログは [カスタム アクションとシーケンス] ビューを通してインストールのシーケンスに挿入することができます。*

カスタム アクションウィザードを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクションウィザードを使用すると、カスタム アクションを作成したり既存のカスタム アクションを変更できます。

タスク ***カスタム アクションウィザードを起動するには、以下の操作を行います。***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。カスタム アクション ウィザードが起動します。
3. ウィザードパネルに従い、新しいカスタム アクションを作成します。

また、カスタム アクション ウィザードを使用せずに [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成することもできます。詳細については、「[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する」を参照してください。

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する

プロジェクト: *[カスタム アクションとシーケンス] ビューは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

このビューは、次のプロジェクトの種類では、[カスタム アクション] ビューという名前で呼ばれます。

- *DIM*
- *マージ モジュール*
- *MSM データベース*

カスタム アクションの作成は、カスタム アクション ウィザードの使用をお勧めします。ウィザードを使うと、必要な手順に従ってカスタム アクションを作成し、情報を入力できます。また、このウィザードを使用する代わりに、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) で設定を直接構成して、カスタム アクションを作成することもできます。

タスク

カスタム アクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、追加するカスタム アクションの種類をクリックします。
3. 新しいアクションの名前を変更するには、アクションを選択し、F2 キーを押して新しい名前を入力します。
4. カスタム アクションの設定を構成します。

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

カスタム アクションのクローン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

InstallShieldには、カスタム アクションをコピーまたは*クローン*する機能があります。クローンは、オリジナルカスタム アクションとすべて同じプロパティおよび値持つ、同じ種類の新しいカスタム アクションを作成します。クローンを利用して、各カスタム アクション属性を手動で設定する必要なく、類似する属性を持つ複数のカスタム アクションを作成することができます。

クローン処理は、カスタム アクションのクローンのみが行われます。[インストール] シーケンスへカスタム アクションを挿入することはありません。

タスク ***カスタム アクションをクローンするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーで、クローンするカスタム アクションを右クリックして、**[クローン]** をクリックします。

クローンされたカスタム アクションのコピーが、[カスタム アクション] エクスプローラーにコピーされます。カスタム アクションの名前は、クローンされたカスタム アクションと同じですが、コピーは、名前の終わりに 1 が付加されています。

カスタム アクションを別のプロジェクトにエクスポートする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

タスク ***カスタム アクションを別のプロジェクトへ保存するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI の場合) または **[カスタム アクション]** (DIM、マージ モジュール プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーで、カスタム アクションを右クリックして、**[エクスポート]** をクリックします。**[エクスポート情報]** ダイアログ ボックスが開きます。
3. 既存の InstallShield プロジェクト ファイル (.ism ファイル) の場所を参照します。保存場所は、インストール プロジェクトまたはマージ モジュール プロジェクトのどちらかを使用できますが、ファイルは InstallShield の別のインスタンスでは開きません。
4. **[開く]** をクリックします。

このカスタム アクションのコピーが特定の .ism ファイルに追加されます。対象の .ism ファイルが既に異なるプロパティで同じ名前のカスタム アクションを持つ場合は、[競合の解決] ダイアログ ボックスが表示され、競合を解決するオプションを選択できます。

カスタム アクションの設定を構成する

プロジェクト: *[カスタム アクションとシーケンス] ビューは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

このビューは、次のプロジェクトの種類では、[カスタム アクション] ビューという名前で呼ばれます。

- *DIM*
- *マージ モジュール*
- *MSM データベース*

カスタム アクションを作成後、その設定を構成する必要があります。また、その設定を後で変更しなくてはならない場合もあります。たとえば、カスタム アクションに割り当てたられた条件を編集する場合があります。

作成するカスタム アクションの種類によって、そのアクションの設定が異なる意味を持つ場合があります。たとえば、アクションが .msi パッケージ内に格納されている実行可能ファイルを呼び出す場合、呼び出す実行可能ファイルをポイントするテーブル内のエントリの名前が "ソース" 設定に存在する必要があります。"ターゲット" 設定は、この場合コマンドライン引数になります。

タスク ***カスタム アクションの設定を構成するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. [カスタム アクション] エクスプローラーで、その設定を構成するカスタム アクションをクリックします。
3. 右側のペインで、適切に設定を構成します。

各設定については、「カスタム アクションの設定」を参照してください。

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

Summary List of All Custom Action Types (Windows Installer ヘルプ ライブラリ)

カスタム アクションの種類についての概要

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクションの "MSI タイプ番号" 設定に表示される値は、カスタム アクション (標準 DLL、実行可能ファイル、または InstallScript など)、場所 (Binary テーブルに格納、製品と共にインストールなど)、および、いくつかのカスタム アクションの設定 ("戻り値の処理"、"スクリプト内処理"、"実行スケジュール" など) が組み合わされたものです。これは、Windows Installer CustomAction テーブルの Type 列で利用できるいくつかのビットを組み合わせるために OR 演算子を使って計算された十進数値です。また、InstallShield では、InstallScript カスタム アクションおよび標準 DLL ファイル関数などの拡張カスタム アクション タイプもサポートされています。

Windows Installer ヘルプライブラリでは、カスタム アクションは数値タイプで参照されています。[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でプロジェクトにカスタム アクションを追加すると、カスタム アクション タイプ番号は自動的に計算され、"MSI タイプ番号" 設定にこの値が設定されます。たとえば、完全修飾パスが Property テーブルに格納されている実行可能ファイルの起動は、カスタム アクション タイプ 50 として計算されます。基本のカスタム アクション タイプに使われる値ついての詳細は、Windows Installer ヘルプ ライブラリの「Summary List of All Custom Action Types」を参照してください。

InstallShield で使用できるカスタム アクションの各種類についての詳細は、「カスタム アクションの種類」を参照してください。

Custom Action Type 50 (Windows Installer ヘルプ ライブラリ)

カスタム アクションの戻り値

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクション ウィザードの [追加のオプション] パネルでの選択に応じて、Windows Installer はインストールを継続する前に正常な戻り値を待機します。カスタム アクションは、その状況に応じて以下の値のうちの 1 つを戻さなければなりません。

テーブル 5 ・ カスタム アクションの戻り値

戻り値	説明
ERROR_FUNCTION_NOT_CALLED	カスタム アクションは実行されませんでした。
ERROR_INSTALL_FAILURE	カスタム アクションが失敗しました。
ERROR_INSTALL_SUSPEND	カスタム アクションは保留されていますが、後で再実行されます。
ERROR_INSTALL_USEREXIT	エンドユーザーが実行を中止したため、カスタム アクションは正常に完了しませんでした。
ERROR_NO_MORE_ITEMS	カスタム アクションは正常に完了しましたが、残りのアクションをスキップする必要があります。
ERROR_SUCCESS	カスタム アクションは正常に終了しました。

InstallScript カスタム アクションを書いている場合、エントリポイント関数の戻り値として ERROR_SUCCESS または ERROR_INSTALL_FAILURE のいずれかを戻す可能性があります。これらの定数は、スクリプト中に Iswi.h を含めると自動的に定義されます。InstallScript エンジンはその他のすべての例外を Windows Installer に通知します。

スクリプト内実行

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクション作成時に構成されている必要がある設定は、カスタム アクションウィザードの [応答オプション] パネル にある "スクリプト内実行" 設定です。アクションは、シーケンスに表示される順に実行されます。ただし、シーケンスは通常のインストール中に何度も実行されます。"スクリプト内実行" 設定を使うと、アクションをトリガーするシーケンス反復を選択することができます。

(ターミナル サーバーで使用可能)

[ターミナル サーバーで使用可能] オプション (たとえば、即時実行 (ターミナル サーバーで使用可能) を選択した場合、ターミナルサーバー マシン上でマシンごとのインストールが実行されいる間、カスタム アクションはエンドユーザーに偽装します。

即時実行

名前のとおり、内部 Windows Installer インストール スクリプトがコンパイルされるとただちに起動します。.msi ファイルが起動すると、Windows Installer サービスはインストールデータベースのすべてのテーブルを内部スクリプトに変換します。(この内部スクリプトは、InstallScript コードとは違います。)このスクリプトは、インストールのすべてのアクションを表示順に循環することによってビルドされます。このスクリプトのビルドはすぐに実行されます。"スクリプト内実行" 設定を [すぐに実行] に設定しているアクションが実行されると、そのアクションが起動します。したがってこのアクションはファイル転送が起きる前に起動します。インストールのエンドユーザー インターフェイスが完全にロードする前に起動する可能性もあります。

一般に、「即時実行」にスケジューリングされているカスタム アクションはターゲット システムを変更せず、プロパティを設定してターゲット システムを照会 (たとえば、ターゲット システムが製品のに合っているか確認するなど) します。Windows Installer のプロパティを設定するカスタム アクションと、[ユーザー インターフェイス] シーケンスで発生するカスタム アクションはすぐに実行されるようにスケジューリングする必要があります。

このタイプのアクションはシステム変更が行われる前に起動するため、インストールにインストールされているファイルを使用できません。

遅延実行

[遅延実行] は、Windows Installer サービスが、[即時実行] の実行中に作成された内部スクリプトを使用する場合に実行されます。このスクリプトが完全に生成されると、Windows Installer サービスは新しくコンパイルされたスクリプトを実行します。スクリプトはシーケンス内のすべてのアクションを順番に起動します。ただし、アクションをすぐに実行するようスケジューリングした場合は、そのアクションは遅延実行時に再度起動しません。

遅延実行中に起動したアクションは、システムの一部としてインストールされているファイルにアクセスできます。したがって、インストールのこの部分で製品とともにインストールされた DLL ファイルから関数を呼び出すカスタム アクションを呼び出すことができます。ただし、遅延実行カスタム アクションを正しく実行するには、InstallInitialize と InstallFinalize の間で行う必要があります。

ターゲット システムにインストールされているファイルを使用するカスタム アクションや、既に実行されたその他のシステム変更を使用するカスタム アクションは、遅延実行を行うようスケジューリングする必要があります。

ロールバック実行

ロールバックは、インストールにエラーが発生した時や、エンドユーザーがインストール完了前にキャンセルした時に実行されます。ロールバック実行オプションを使用すると、ロールバック中にのみアクションが実行されるよう設定できます。したがって、ロールバック実行できるアクションは、遅延実行アクションのようにインストールスクリプトに書き込まれます。遅延実行アクションと異なり、ロールバック実行アクションはロールバック中にしか実行されません。(ロールバック カスタム アクションは、インストールが遅延実行中に失敗した場合にのみ実行されます。)

インストール中にターゲット システムに変更を加えるカスタム アクションは、ロールバック中にロールバック実行カスタム アクションでやり直されます。たとえば、ファイルを作成するカスタム アクションがある場合、ファイルを削除する 2 つ目のカスタム アクションを作成して、2 番目のアクションをロールバック実行にスケジューリングします。(ロールバックカスタム アクションは、元に戻すカスタム アクションの前にスケジューリングしてください。)

コミット実行

コミット実行は、InstallFinalize アクションが正常に完了するまで実行されません。つまり、インストールがファイル転送、COM サーバー登録、およびショートカットとレジストリ エントリの作成を完了するまで行われません。それから、コミット実行に設定されているアクションはシーケンスの表示順に実行されます。

たとえば、一時ファイルを作成するカスタム アクションがある場合、ファイルを削除する 2 つ目のカスタム アクションを作成して、そのアクションをコミット実行にスケジューリングします。

システム コンテキストの遅延実行

遅延実行アクション同様、これらのアクションも Windows Installer サービスによって生成されたスクリプトが実行されなければ実行されません。ただし、このタイプのアクションはユーザー偽装を実行しません。

カスタム アクションの動作をドキュメント化する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows ロゴ: *各カスタム アクションの意図された動作は、Windows ロゴ プログラムに基づいて文書化する必要があります。これは、システム管理者が製品を企業環境に配布する場合など、カスタム アクションの動作を把握する必要がある場面で特に有益です。インストール パッケージを検証した時、ヘルプ ファイル パス 設定の値が指定されていない場合、InstallShield は検証エラー ISICE10 を出力します。詳細については、「ISICE10」を参照してください。*

タスク ***プロジェクトに含まれるカスタム アクションの動作をドキュメント化するには、次の手順に従います。***

1. カスタム アクションの正常な動作を説明するファイルを作成します。.txt、.htm、.rtf ファイルなど、テキストベースのファイルを指定してください。カスタムアクション 1 個につき、1 つのファイルが必要です。
2. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
3. **[カスタム アクション]** エクスプローラーで、ドキュメントを作成するアクションをクリックします。
4. **[ヘルプ ファイル パス]** 設定については、省略記号ボタン (...) をクリックして、カスタム アクションの動作について説明するファイルを参照します。

ヒント: *ビルド時に InstallShield が、各カスタム アクション ヘルプ ファイルのコンテンツを .msi ファイルにストリームするかどうかを指定することができます。詳細については、[リリース] ビュー内にある製品構成の "カスタム アクションのヘルプを含める" 設定の説明を参照してください。*

Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows ロゴを適用する場合、アプリケーションは Microsoft Windows ロゴ プログラムの要件を満たさなくてはいけません。検証スイートを使って、作成中のインストール パッケージがカスタム アクション関連のロゴ要件のほとんどを満たしているかどうかを検証できます。ただし、一部の要件については検証スイートを使って検証することができません。

検証スイートまたは関連 ISICE で検証されないけれども、Windows ロゴ プログラムのロゴ コンプライアンスを達成するために満たす必要がある要件の一覧は次のとおりです。

- グローバル アセンブリ キャッシュ ツール (Gacutil.exe) をカスタム アクションから呼び出さない。このツールは、インストール中の使用のためには設計されていません。
- すべてのカスタム アクションの成功または失敗について、**MsiProcessMessage** を呼び出して Windows Installer ログに記録する。
- インストール中にシステム状態を変更するカスタム アクションは、アンインストール中に削除するか、元のシステム状態に復元しなくてはならない。また、システム状態を変更するカスタム アクションは、遅延およびロールバック カスタム アクションとの対で作成しなくてはならない。

  これは、カスタム アクション タイプ 19 (エラー、失敗、およびインストールの終了を表示する)、タイプ 35 (インストール ディレクトリを設定する)、または タイプ 51 (プロパティを設定する) には適用されません。
- インストールに含まれるすべてのカスタム アクションの正常な動作をドキュメント化しなくてはならない。これは、カスタム アクション タイプ 19 (エラー、失敗、およびインストールの終了を表示する)、タイプ 35 (インストール ディレクトリを設定する)、または タイプ 51 (プロパティを設定する) には適用されません。詳細については、「カスタム アクションの動作をドキュメント化する」を参照してください。

  様々な機能をサポートするために InstallShield プロジェクトに自動的に追加される各 InstallShield カスタム アクションについての情報は、「InstallShield カスタム アクション リファレンス」を参照してください。

カスタム アクションをリリース フラグに基づいて条件付きで起動する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールがビルドされると、は自動的に ISReleaseFlags というプロパティを Property テーブルに追加します。このプロパティには、ビルド内のすべてのリリースフラグが含まれています。これらのフラグは [リリース] ビューの "リリース フラグ" 設定と同じ状態で表示されます。複数のフラグはコンマで区切られています。

カスタム アクションを、リリース フラグに基づいて次の方法を使って条件付きで起動させることができます。

"条件" 設定を構成する

カスタム アクションを条件付きで起動する最も簡単な方法は、シーケンスにカスタム アクションを 挿入する時に、"条件" 設定を構成する方法です。この場合、"条件" 設定に **ISReleaseFlags><"MyReleaseFlag"** などの条件を入力することができます。この条件は、**"MyReleaseFlag"** という部分文字列が ISReleaseFlags プロパティの値に含まれている場合に成功します。

MsiGetProperty の呼び出し

リリース フラグに基づいてカスタム アクションの条件を設定するより効果的な方法は、カスタム アクション内で **MsiGetProperty** を呼び出すよう作成する方法です。アクション起動を決定するだけでなく、そのアクションによって実行されるコードも指定することができます。たとえば、すべての場合に対して .DLL から一つの関数を呼び出しながら、関数の条件付論理を使ってそれぞれの場合に別の機能を提供することができます。

この方法によるカスタム アクションの条件付き起動の欠点は、スクリプトまたは DLL カスタム アクションしか使用できないことです。さらに、カスタム アクションは見えないところでも確実に実行します。アクションを実行すると、アクションは、指定したリリースフラグがビルドに含まれているかどうか確認します。そのフラグがあれば、カスタム アクションは続行します。フラグがなければ、アクションは終了します。

.msi データベースにファイルを配置し、実行時に抽出する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

[サポート ファイル] ビュー (基本の MSI プロジェクトの場合) および [サポート ファイル/ビルボード] ビュー (InstallScript MSI プロジェクト) を使うと、インストール プログラムが使用するだけで、実際にインストールはされない一時ファイルを格納できます。これに該当するものには、ライセンスのテキストファイル (SdLicense で表示) や、インストーラーが呼び出す DLL ファイルなどがあります。実行時のサポート ファイルの抽出先は、Windows Installer プロパティ SUPPORTDIR に格納されています。

メモ: 基本の MSI インストールおよび InstallScript MSI インストール インストールの遅延/コミット/ロールバックのカスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、カスタム アクションで他のプロパティ (例、SUPPORTDIR) にアクセスする場合は、それらを CustomActionData に渡す必要があります。詳細については、「遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う」を参照してください。

プロジェクト: Windows Installer プロパティ SUPPORTDIR の値は、InstallScript システム変数 SUPPORTDIR の値とは同じではありませんので注意してください。

カスタム アクション スクリプトの例

InstallScript

インストール時に特定のサポート ファイルにアクセスするには、基本の MSI プロジェクトと InstallScript MSI プロジェクトでは、Windows Installer プロパティ SUPPORTDIR を参照する必要があります。ファイルの完全パスを取得するには、**MsiGetProperty** を使用してパスをクエリしてから、そのファイル名を SUPPORTDIR の値にファイル名を付加します。カスタム アクションの設定では、これは単に [SUPPORTDIR] です。

InstallScript カスタム アクションでは、次のようなコードを使用することができます。

```
export prototype STRING GetSupportFilePathMSI(HWND);

function STRING GetSupportFilePathMSI(hMSI)
    STRING szSupportDir[MAX_PATH + 1];
    STRING szMyFile[MAX_PATH + 1];
    NUMBER nLength;
begin
    // バッファー サイズの初期値を設定します
    nLength = MAX_PATH + 1;
    MsiGetProperty(hMSI, "SUPPORTDIR", szSupportDir, nLength);

    // バッファー サイズ変数をリセットします
    nLength = MAX_PATH + 1;
    MsiGetProperty(hMSI, "MYFILE", szMyFile, nLength);

    // 完全ファイル パスを返します
    return szSupportDir ^ szMyFile;
end;
```

この例では、使用されているサポート ファイルの名前が MYFILE プロパティに格納されます。

C++

C++ では、次のコードが利用できます:

```
UINT _stdcall ShowSupportdir(MSIHANDLE hInstall)
{
    TCHAR szSupportDir[MAX_PATH + 1] = {'¥0'};
    DWORD dwBuff = sizeof(szSupportDir);
    MsiGetProperty(hInstall,"SUPPORTDIR",szSupportDir,&dwBuff);
    MessageBox(NULL,szSupportDir,"SUPPORTDIR is ...",MB_OK);
```

```
}
```

VBScript

VBScript では、次のコードが利用できます:

```
dim strSupportDir
strSupportDir = Session.Property("SUPPORTDIR")
```

MsiGetProperty

遅延/コミット/ロールバック カスタム アクションで Windows Installer プロパティにアクセスする、またはその設定を行う

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

基本の MSI インストールおよび InstallScript MSI インストール インストールの遅延/コミット/ロールバックのカスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができます。遅延/コミット/ロールバックの実行中に、カスタム アクション で他のプロパティにアクセスする場合は、それらを CustomActionData に渡す必要があります。これを行うには、すぐにプロパティを設定するタイプのカスタム アクションをスケジュールして、カスタム アクションの名前に一致するプロパティを設定します。このプロパティの値が、遅延/コミット/ロールバックのカスタム アクション内の CustomActionData プロパティから取得できるようになります。

CustomActionData を使用して、プロパティにアクセスする

以下の例は、遅延 InstallScript カスタム アクションで、Windows Installer プロパティ SUPPORTDIR にアクセスする方法です。

タスク

遅延 InstallScript カスタム アクションで SUPPORTDIR にアクセスするには、以下の手順に従います:

1. **[カスタム アクションとシーケンス]** ビューで、**GetSUPPORTDIR** という名前のプロパティ設定型カスタム アクション (type 51) を作成します。カスタム アクションの **"プロパティ名"**、**"プロパティ値"**、および **"インストール実行シーケンス"** 設定を次のように構成し、他の設定は空白のままにしておきます。
   - プロパティ名: DisplaySupportDir
   - プロパティ値: [SUPPORTDIR]
   - インストール実行シーケンス: After InstallInitialize
2. **InstallScript** ビューで、**DisplaySupportDir** という名前で新しいカスタム アクションを作成します。
3. SUPPORTDIR の値を含むメッセージ ボックスを表示する次のコードを追加します。

```
function DisplaySupportDir(hMSI)
  STRING supportDirPath;
  NUMBER supportDirPathBuffer;
begin
```

```
    supportDirPathBuffer = MAX_PATH;
    if(MsiGetProperty(hMSI, "CustomActionData", supportDirPath, supportDirPathBuffer) == ERROR_SUCCESS) then
      SprintfBox(INFORMATION,"遅延実行","SUPPORTDIR の値は、%s",supportDirPath);
      SprintfBox(INFORMATION,"遅延実行","SUPPORTDIR の値は、%s",SUPPORTDIR);
    endif;
  end;
```

4. **[カスタム アクションとシーケンス]** ビューで、**DisplaySupportDir** という名前の InstallScript カスタム アクションを作成します。カスタム アクションの **"関数名"**、**"スクリプト内実行"**、および **"インストール実行シーケンス"** 設定を次のように構成し、他の設定は空白のままにしておきます。

   - 関数名: DisplaySupportDir
   - スクリプト内実行: システム コンテキストの遅延実行
   - インストール実行シーケンス: After GetSUPPORTDIR

***重要・**ステップ 1 で入力するプロパティ名は、ステップ 4 で作成するカスタム アクションの名前と一致しなくてはなりません。*

CustomActionData を使用して、複数のプロパティにアクセスする

遅延/コミット/ロールバック カスタム アクションで、複数の Windows Installer プロパティにアクセスするには、CustomActionData でそれらのプロパティを "パック" してから、CustomActionData の値を取得したあと、遅延/コミット/ロールバック カスタム アクションを使って、それらの値を "アンパック" します。

たとえば、[INSTALLDIR]、[SUPPORTDIR]、および [SetupType] の値を取得する場合、type 51 カスタム アクションの Property Value 設定を次のように設定します。

[INSTALLDIR];[SUPPORTDIR];[SetupType]

各プロパティは、セミコロンで区切ります。

CustomActionData プロパティからの値を "アンパック" するには、次の VBScript のようなコードを使用します。

```
Dim PropArray
PropArray = Split(Session.Property("CustomActionData"), ";")
INSTALLDIR = PropArray(0)
SUPPORTDIR = PropArray(1)
SetupType = PropArray(2)
'変数を使用することができます...
```

InstallScript カスタム アクション

***プロジェクト・**この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

Windows Installer インストール内の InstallScript 関数を使用するには、最初に、スクリプトを呼び出すカスタム アクションを作成する必要があります。カスタム アクション ウィザードを利用して、カスタム アクションを最も簡単に作成することができます。

このウィザードの主なパネルは次のとおりです。

- [アクションの種類] パネル
- [アクション パラメーター] パネル
- [応答オプション] パネル
- [シーケンスに挿入] パネル

ウィザードを完了すると、タイプ 65536 のカスタム アクションが作成されます。このカスタム アクションタイプは、Windows Installer ヘルプライブラリ で定義されていません。InstallScript は Windows Installer のビルトイン機能ではなく、アクションタイプ 65536 は InstallShield によって作成されています。

プロジェクト: *InstallScript カスタム アクションは、InstallScript MSI プロジェクトのユーザー インターフェイス シーケンスでは使用しないでください。InstallScript MSI プロジェクトにユーザーインターフェースを提供するには、イベント ドリブン型のスクリプトを使用します。*

標準 DLLI ファイルの関数を呼び出す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer では、カスタム アクション向けに作成された DLL ファイル内 の関数に渡すことができるパラメーターに制限がありますが、InstallShield には解決策が用意されており、任意の個数のパラメーターを関数に渡して戻り値を格納できます。関数は必ず _stdcall 呼び出し規則を使用しなくてはなりません。この規則が使用されていない場合、複数のパラメーターを持つ関数は適切に動作しません。

カスタム アクション ウィザードの [アクションの種類] パネルで **[標準ダイナミック リンク ライブラリ内の関数の呼び出し]** を [種類] オプションに選択して、標準 DLL を呼び出すように指定できます。次のパネル、[関数定義] パネルで、関数のパラメーターと戻り値を指定できます。

メモ: *製品と共にインストールされる標準 DLL ファイルの関数を呼び出し、アクションが ("スクリプト内実行" 設定で) "遅延" とスケジュールされている場合、呼び出す DLL ファイルはコンポーネントのキーファイルでなくてはなりません。*

Binary *テーブル内の標準 DLL ファイルの関数を呼び出すカスタム アクションは、"遅延実行" をスケジュールできません。*

Windows Installer プロパティを関数の戻り値に設定する場合、アクションを即時実行に構成する必要があります。遅延、ロールバック、またはコミット実行にスケジュールされているアクションの戻り値プロパティを指定しようとすると、実行時に戻り値プロパティは無視されます。

フォーマットされた文字列としてのパラメーターと戻り値について

標準 DLL ファイル アクションの関数シグネチャを指定するとき、そのアクションの "関数シグネチャ" 設定は、以下の形式で関数名、引数、戻り値タイプ、および戻り値プロパティを表示します。

DataType1=[PropertyName1] DllName::FuncName(in DataType2="値", Direction DataType3=[PropertyName2])

次のテーブルは、フォーマットのそれぞれの部分についての説明です。

テーブル 6 ・ 関数の引数のフォーマット

引数	説明
DataType1=[PropertyName1]	DataType1 のタイプは、void、STRING、BOOL、NUMBER、HWND、HANDLE、または POINTER のどれかです。PropertyName1 は、[プロパティ マネージャー] ビュー からのプロパティ名です。 戻り値が void の場合、=[PropertyName1] は省略されます。
DllName	ピリオドと拡張子を除く DLL ファイル名を指定します。
FuncName	呼び出す関数の名前を入力します。
in DataType2="Value"	これは、定数である引数の形式です。DataType1 のタイプは、STRING、BOOL、NUMBER、HWND、HANDLE、または POINTER のいずれかです。Value は、この引数に渡すデータです。定数のデータ型は、常に in から始まります。
Direction DataType3=[PropertyName2]	これは、Windows Installer のプロパティ値を参照する引数の形式です。Direction には、*in*、*inout*、または *out* のいずれかを指定できます。これらのプロパティ タイプについては、引数のソースの下にある [関数定義] パネルを参照してください。 DataType3 では、STRING、BOOL、NUMBER、HWND、HANDLE、または POINTER のタイプを使用できます。PropertyName2 は、Direction の値に応じて、値が関数に渡されるプロパティ、または関数によって値が設定されるプロパティの名前です。

例

インストールで MessageBoxA を呼び出す で作成されたカスタム アクションの場合、Target の値は次のようになります。

void User32::MessageBoxA(in HWND=0, in STRING=[MESSAGEPROP], in STRING=[CAPTIONPROP], in NUMBER=1) と表示されます。

ここ例で、データタイプは *void* で、*in HWND=0* は数値 0 を表します。また *MESSAGEPROP* は、User32.dll の関数 MessageBoxA に渡される文字列を含むプロパティです。

Windows Installer DLL ファイル内の関数を呼び出す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows Installer のカスタム アクション専用に作成されている DLL 関数では、インストーラー データベースへのハンドルだけをパラメーターとして使用できます。.msi データベースから情報を読み込み、カスタム アクションで使用する方法について、以下に説明します。

カスタム アクション ウィザードの [アクションの種類] パネルで **[Windows Installer ダイナミック リンク ライブラリ内の関数の呼び出し]** を選択して、カスタム アクションが Windows Installer DLL ファイル (カスタム アクション タイプ 1 および 17) にあることを指定します。

別の方法としては、[アクションの種類] パネルで **[標準ダイナミック リンク ライブラリ内の関数の呼び出し]** を選択することができます。この場合、InstallShield で関数のパラメーターを指定することができます。

タスク ***Windows Installer 用に作成された DLL の関数にパラメーターを渡す場合、主に 3 つのステップが必要です。***

1. DLL ファイルを作成します。
2. カスタム アクションを作成してシーケンスの 1 つに挿入します。
3. [プロパティ マネージャー] ビューを使用して、パラメーターを渡します。以下に、これらのステップを詳しく説明します。

DLL ファイルの作成

カスタム アクションの DLL ファイル関数にデータを渡すには、データを渡す関数で、Installer プロパティの値を取得する **MsiGetProperty** 関数を呼び出す必要があります。以下の例では、**MYPROPERTY** というパブリック プロパティから値を取り出します。

```
UINT _stdcall MyActionName(MSIHANDLE hInstall)
{
    TCHAR szValue[51] = {0};
    DWORD dwBuffer = 50;

    MsiGetProperty(hInstall, TEXT("MYPROPERTY"), szValue, &dwBuffer);

    MessageBox(GetForegroundWindow( ),
        szValue,
        TEXT("MYPROPERTY の値"), MB_OK | MB_ICONINFORMATION);

    return ERROR_SUCCESS;
}
```

さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「MsiGetProperty」を参照してください。

DLL ファイルのビルドに C++ コンパイラを使用している場合は、コンパイラが DLL からエクスポートされた関数名を変更していないよう確認してください。たとえば Microsoft Visual C++ では、DLL ファイルからエクスポートする関数名を指定する .def ファイルを作成することができます。通常の .def (MyActions.def という名前) ファイルは次のような形式です。

```
LIBRARY MyActions
EXPORTS
   MyActionName
```

関数名修飾についての詳細は、コンパイラのマニュアルを参照してください。

カスタム アクションを作成してシーケンスに挿入する

2 番目のステップでは、カスタム アクションを作成して、それを [インストール] シーケンスに挿入します。カスタム アクションを作成するには、カスタム アクション ウィザードを使用し、カスタム アクションを配置するには、[カスタム アクションとシーケンス] ビューを使用します。

Property Manager を使ってパラメーターを渡す

タスク ***プロパティ マネージャーを使用して新しいパラメーターを渡すには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンをクリックします。ビューの下に新しい行が追加されます。
3. **Name** 列で、**MsiGetProperty** (この例では **MYPROPERTY**) を使用して取得するプロパティの名前を入力します。
4. **Value** 列で、渡す値を入力します。たとえば Web サイトに URL を渡す場合、**http://www.mycompany.com** と入力します。

カスタム アクションで DLL ファイル関数にパラメーターを渡す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

カスタム アクション中の関数へパラメーターを渡すのには多くの理由があります。たとえば、Web 登録のために URL を渡したり、カスタムユーザーインターフェイスに UserName プロパティを渡したりする場合などです。ただし、Windows Installer は DLL ファイル関数へパラメーターを渡すことを直接にはサポートしません。

Windows Installer DLL ファイルのエントリポイント関数が持つことのできる唯一の引数はデータベースへのハンドルです。Windows Installer 用に書かれた DLL ファイルにパラメーターを渡す方法については、「Windows Installer DLL ファイル内の関数を呼び出す」を参照してください。

タスク ***InstallShield でこれを実現するには、以下の手順を実行します。***

1. カスタム ウィザードの **[アクションの種類]** パネルで、**[標準ダイナミックリンクライブラリの関数を呼び出す]** を選択します。
2. **[関数定義]** パネルで、InstallShield が関数に渡すパラメーターを指定します。

マネージ アセンブリのパブリック メソッドを呼び出す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Visual Basic .NET または C#のようなマネージ コードで書かれた .NET アセンブリ内のパブリック メソッドを呼び出すカスタム アクションの種類を使用します。このセクションには、次のトピックが含まれています:

- マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- .NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
- アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する

マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

マネージ コード タイプのカスタム アクションは、Visual Basic .NET または C# などのマネージ コードで書かれた .NET アセンブリ内にあるパブリック メソッドを呼び出します。

プロジェクトにマネージ コード カスタム アクションを含めると、.NET アセンブリに C++ Windows Installer ラッパー DLL が作成されます。ラッパー DLL には、アセンブリのほか、アセンブリの仲介、ロード、実行を行うために必要な情報が含まれています。

メモ: *マネージ アセンブリの場所を* Binary *テーブルに指定すると、ラッパー (アセンブリが含まれています)* Binary *テーブルに格納されます。*

マネージコード カスタム アクションの実行時要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

マネージ コード タイプのカスタム アクションは、ターゲット システムに Microsoft .NET Framework が必要です。

InstallShield には .NET Framework の再配布可能ファイルが含まれています。.NET Framework がターゲット システムにない可能性がある場合、適切な .NET Framework 再配布可能ファイルをプロジェクトに追加することができます。手順については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

IS_CLR_VERSION プロパティを利用すると、カスタム アクションがダウンロードしてマネージ コードを実行するセミコロンで区切られた .NET Framework のバージョン一覧を識別することができます。ほとんどのシナリオで、このプロパティはインストール パッケージで設定されていません。コマンドラインで設定されます。バージョン 1.1 が必要であることを指定する場合、次のコマンドライン パラメーターを使用します:

```
IS_CLR_VERSION=v1.1.4322
```

プロパティの値には、.NET Framework の完全バージョン番号を指定する必要があることに注意してください。複数のバージョンが使用可能な場合、バージョンの一覧をセミコロンで区切って指定できます。ロードが可能であるバージョンの中から最初のものが使用されます。次のコマンドライン パラメーターの例では、カスタム アクションがバージョン 2.0 をロードしようと試みます。このバージョンが存在しない場合、バージョン 1.1 のロードが試みられます。バージョン 1.1 が見つからなかった場合、カスタム アクションは失敗します。

```
IS_CLR_VERSION=v2.0.50727;v1.1.4322
```

指定したバージョンがインストールされなかったとき、カスタム アクションで、インストールされている .NET Framework の中から一番新しいバージョンをロードする試みを行うように指定する場合、下の例のようにセミコロンをプロパティ値の最後に付加します:

```
IS_CLR_VERSION=v2.0.50727;v1.1.4322;
```

また、プロパティ値の最後に付加されたセミコロンは、指定されたバージョンが存在しないが、あるバージョンの .NET Framework が既にロードされている場合、それが既にインストールされている最新バージョンではなくても、現在ロードされているバージョンを使用することを示します。

メモ: *マネージ コード カスタム アクションの実行が遅延モードに設定されている場合、Windows Installer プロパティ* CustomActionData *を使って* IS_CLR_VERSION *プロパティと値を渡す必要があります。詳細については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。*

ヒント: *.NET Framework のバージョンの不一致が原因でカスタム アクションで問題が発生した場合、インストールの実行時に、エンドユーザーにコマンドラインで* IS_CLR_VERSION *プロパティを設定するように指示することも考えられます。遅延カスタム アクションの場合、*CustomActionData *で* IS_CLR_VERSION *プロパティが既に渡されていることに注意してください。詳細については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。*

32 ビット マネージ コード カスタム アクションと 64 ビット マネージ コード カスタム アクションの違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクトでマネージコード カスタム アクションを含むリリースをビルドするとき、InstallShield はカスタム アクションと関連付けられたメインの .NET アセンブリのターゲット アーキテクチャ (32 ビットまたは 64 ビット) を判別しようとします。InstallShield は、アセンブリのポータブル実行形式ファイル (PE) を調査して、PE ファイルが 32 ビットであるか、64 ビットであるかを判別します。PE ファイルが Microsoft intermediate language (MSIL) コードを使用する場合、InstallShield はこれを 32 ビットとして取り扱います。InstallShield は、実行時にマネージ コードを実行するときに .NET Framework の適切なバージョン (32 ビットまたは 64 ビット) が使用されるようにリリースを構成します。

組み込まれたデフォルトの動作をオーバーライドするには、**[ダイレクト エディター]** ビューを使って、次のフィールドを含む新しいレコードを ISClrWrap テーブルに追加します。

テーブル 7 ・ マネージ コード カスタム アクションのデフォルト アーキテクチャをオーバーライドする

ISClrWrap テーブル列	説明
Action_	変更するマネージコード カスタム アクションの名前を入力します。
名前	次を入力します: **TargetPlatform**
値	適切なアーキテクチャを指定します。 • .NET Framework の 32 ビット バージョンを使用するには、次を入力します:**x86** • .NET Framework の 64 ビット バージョンを使用するには、次を入力します:**x64**

.NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

.NET Core タイプのカスタム アクションは、Visual Basic .NET または C#のようなマネージ コードで書かれた .NET アセンブリ内のパブリック メソッドを呼び出します。

.NET Core カスタム アクション タイプをプロジェクトに含んだ場合、InstallShield によって .NET アセンブリの C++ Windows Installer ラッパー DLL が作成されます。ラッパー DLL には、アセンブリのほか、アセンブリの仲介、ロード、実行を行うために必要な情報が含まれています。

メモ: *マネージ アセンブリの場所を* Binary *テーブルに指定すると、ラッパー (アセンブリが含まれています)* Binary *テーブルに格納されます。*

.NET Core カスタム アクションの実行時要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

.NET Core タイプのカスタム アクションを実行するには、次がターゲット システム上にインストールされている必要があります:

- Microsoft .NET Core 8.0
- Microsoft Visual C++ 2022 再配布可能ファイル

InstallShield には .NET Core の再配布可能ファイルが含まれています。ターゲット システムに .NET Core が搭載されていないことが予想される場合、プロジェクトに適切な .NET Core 再配布可能ファイルを追加することができます。手順については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

32 ビットと 64 ビット .NET Core カスタム アクションの違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクトに .NET Core カスタム アクションを含むリリースをビルドするとき、InstallShield はカスタム アクションと関連付けられているメインの .NET アセンブリのターゲット アーキテクチャ (32 ビットまたは 64 ビット) を判別しようとします。InstallShield は、アセンブリのポータブル実行形式ファイル (PE) を調査して、PE ファイルが 32 ビットであるか、64 ビットであるかを判別します。PE ファイルが Microsoft intermediate language (MSIL) コードを使用する場合、InstallShield はこれを 32 ビットとして取り扱います。InstallShield は、実行時にマネージ コードを実行するために .NET Core 32 ビットまたは 64 ビットの適切なバージョンを使用するリリースを構成します。

組み込まれた既定の動作をオーバーライドするには、**[ダイレクト エディター]** ビューを使って、次のフィールドを含む新しいレコードを ISClrWrap テーブルに追加します:

テーブル 8 - .NET Core カスタム アクションの既定のアーキテクチャをオーバーライドする

ISClrWrap テーブル列	説明
Action_	変更する .NET Core カスタム カスタム アクションの名前を入力します。
名前	次を入力します: **TargetPlatform**
値	適切なアーキテクチャを指定します。 • .NET Core の 32 ビット バージョンを使用するには、次を入力します :**x86** • .NET Core の 64 ビット バージョンを使用するには、次を入力します :**x64**

アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

カスタム アクション ウィザードの [マネージ メソッドの定義] パネルで、マネージ アセンブリ カスタム アクションが呼び出すメソッドの引数を指定できます。[メソッド シグネチャ] ダイアログ ボックスでも、この情報を指定することができます。

デフォルトのメソッド シグネチャと利用について

デフォルトのメソッド シグネチャを利用すると、カスタム アクションは最高パラメーターを 1 つまで使ってメソッドを処理します。メソッドでパラメーターが使用されると、メソッドは MsiHandle および任意の戻り値に渡されます。メソッドが整数を戻すと、この整数は直接 Windows Installer に渡されます。つまり、このシナリオでメソッドが整数 1603 (ERROR_INSTALL_FAILURE) を戻すと、インストールは中止されます。

詳しい情報は、Windows Installer ライブラリの「Custom Action Return Values」を参照してください。

即時カスタム アクションにカスタム メソッド シグネチャを使用する

カスタム メソッド シグネチャを利用すると、カスタム アクションは指定した引数と共にメソッドを呼び出します。パラメーターの値が Windows Installer プロパティで、カスタム アクションが即時実行に設定されている場合、メソッドはすべての参照 (ref) および出力 (out) パラメーターの値を渡されたプロパティに格納します。カスタム アクションは、また、戻り値を文字列に変換し、それを戻り値のプロパテに格納します。戻り値プロパティが指定されていない場合、戻り値の値は無視されます。

カスタム シグネチャを使用している場合、カスタム アクションの戻り値は Windows Installer に渡されません。ただし、マネージ コードが未処理の例外をスローすると、カスタム アクションは ERROR_INSTALL_FAILURE を Windows Installer に戻します。つまり、カスタム アクションで Windows Installer についてインストールの失敗を示すには、マネージ コードで未処理の例外をスローする必要があります。

遅延/コミット/ロールバック カスタム アクションにカスタム メソッド シグネチャを使用する

遅延/コミット/ロールバック カスタム アクションは、一部のビルトイン Windows Installer プロパティ (CustomActionData、ProductCode、および UserSID) にのみアクセスすることができますこのため、カスタム メソッド シグネチャを使用するとき、遅延/コミット/ロールバックの実行中に、マネージ アセンブリ カスタム アクションで他のプロパティを渡す場合は、それらを CustomActionData を通して渡す必要があります。遅延/コミット/ロールバック マネージ アセンブリ カスタム アクションについては、次のガイドラインに留意してください:

- プロパティを渡すカスタム シグネチャの値は、次の形式で CustomActionData として渡される必要があります:

  PROPERTYNAME="プロパティの値"

  複数のプロパティ名プロパティの値のペアは、次の例のようにスペースで区切ります:

  PROPERTYNAME1="プロパティ値 1" PROPERTYNAME2="プロパティ値 2"

- 遅延/ロールバック/コミット カスタム アクションのマネージ アセンブリが製品と共にインストールされる場合、CustomActionData を利用して、マネージ アセンブリの場所を識別し、次の形式を使用します:

  #filekey.dll="アセンブリの場所"

  "アセンブリの場所" の部分には、**[#filekey.dll]** という形式を使用します。

- マネージ アセンブリのパスが 1 つまたは複数のプロパティを参照する場合、CustomActionData を使って、各プロパティと対応する値を渡します。

- CustomActionData を使って、次のプロパティと値を渡すことを検討してください:

  IS_CLR_VERSION="[IS_CLR_VERSION]"

  メモ・この情報はマネージ コード カスタム アクション タイプにのみ適用します。

  カスタム アクションが正しくないバージョンの .NET Framework をロードしてマネージ コードを実行しようとしたときに、実行時に問題が発生する場合、インストールの実行時に、エンドユーザーがコマンドラインで IS_CLR_VERSION を設定するように構成することができます。遅延/コミット/ロールバック マネージ コード カスタム アクションの場合、エンドユーザーが実行時にこのオーバーライドを使えるように IS_CLR_VERSION を CustomActionData に渡す必要があります。IS_CLR_VERSION プロパティについては、「マネージコード カスタム アクションの実行時要件」を参照してください。

カスタム メソッド シグネチャのパラメーター値を指定する

カスタム メソッド シグネチャを利用すると、パラメーターに次の種類の値を入力できます:

- 角かっこで囲まれた単一プロパティ (例、**[ProductName]**)。即時モードのマネージ アセンブリ カスタム アクションは ref および out パラメーターを更新することができます。InstallShield では、パラメーターの値に、複数のプロパティを組み合わせたり、プロパティと文字列を混合させて使用することはできません。

- リテラル – たとえば、数字 **1** や **1603** などの文字列。

- 引用符で囲まれている明示的な文字列 (例、**「マイ プロジェクト名-1」**)。

- 変数 *MsiHandle* はインストールのハンドルです。

実行時に、プロパティが解決されたあと、または文字列が引用符内から分離されたあと、カスタム アクションはその文字列をそのメソッドの適切なパラメーターの種類のインスタンスに変換します。String 型のパラメーターは、そのままの形で渡され、すべて標準数値型は、文字列をその型に変えるパブリック スタティック Parse(string) メソッドを呼び出して処理されます。カスタム アクションは、すべての型のパブリック スタティック Parse(string) メソッドを呼び出し、その戻り値を渡されたパラメーターの値として使用します。IntPtr は、Parse メソッドがなくても処理されます。

プロセスの強制終了カスタム アクションの呼び出し

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *トランスフォーム*

Kill-Process タイプのカスタム アクションによって、実行時に、1 つまたは複数のプロセスが終了されます。以下は、プロジェクトで、このタイプのカスタム アクションの追加および構成方法の説明です。

タスク ***Kill-Process カスタム アクションをプロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[新しいプロセスの強制終了]** を選択します。プロセスの強制終了カスタム アクションがデフォルトの名前で追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このカスタム アクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 右側のペインで、アクションの設定を構成します。
   a. **"スクリプト内実行"** 設定で、アクションをトリガーするシーケンスの繰り返しを選択します。各オプションの詳細については、「スクリプト内実行」を参照してください。
   b. **[シーケンス]** 領域の設定を使って、プロセスを強制終了する位置にカスタム アクションをスケジュールします。代わりに、カスタム アクションを **[カスタム アクション]** エクスプローラーから、[シーケンス] エクスプローラーの下の適切なノードにドラッグします。
   c. **"関数名"** 設定で、適切な関数を選択します。
      - KillProcess— **"スクリプト内実行"** 設定で即時オプションのいずれかが選択されていて、特定の名前を持つプロセスを強制終了する場合、このオプションを選択します。

- KillProcessByID—**"スクリプト内実行"** 設定で即時オプションのいずれかが選択されていて、特定のプロセス ID (PID) を持つプロセスを強制終了する場合、このオプションを選択します。
- KillProcessDeferred—**"スクリプト内実行"** 設定で、遅延、コミット、またはロールバック オプションのいずれかが選択されていて、特定の名前を持つプロセスを強制終了する場合、このオプションを選択します。
- KillProcessByIDDeferred— **"スクリプト内実行"** 設定で、遅延、コミット、またはロールバック オプションのいずれかが選択されていて、特定の PID を持つプロセスを強制終了する場合、このオプションを選択します。

d. **"プロセス"** 設定で、強制終了するプロセスの実行可能ファイルの名前または PID を入力します。

関数名が次に設定されている場合:	次を使ってプロセスを指定します:
KillProcess または KillProcessDeferred	実行可能ファイル名
KillProcessByID または KillProcessByIDDeferred	PID

1 つ以上のプロセスを終了する場合、セミコロン (;) を使って、ファイル名または PID を区切ります。たとえば、**myfile1.exe** および **myfile2.exe** という名前のプロセスを終了する場合、次のように入力します:

myfile1.exe;myfile2.exe

ヒント: *"プロセス" 設定の値は、*ISTerminateProcesses *プロパティに書き込まれる場合があります。InstallShield 2015 以前から移行したプロジェクトなど、"プロセス" 設定に値が指定されていない追加の kill-process カスタム アクションがある場合、*ISTerminateProcesses *プロパティを共有して使用すると予期しない動作が発生する可能性があります。*

e. 必要に応じて、アクションの他の設定も構成します。

PowerShell カスタム アクションの呼び出し

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *トランスフォーム*

Windows PowerShell は、構成タスクのオートメーション化を可能にする .NET Framework ベースのコマンドライン シェルおよびスクリプト言語です。InstallShield では、この PowerShell スクリプト (.ps1) を実行するカスタム アクションがサポートされています。この種類のカスタム アクションは、インストールの実行時にシステムの構成タスクを実行するプロジェクトに追加することができます。PowerShell アクションには、ターゲット システム上に PowerShell 2.0 以降が必要です。

PowerShell カスタム アクションをプロジェクトに追加する

タスク ***PowerShell カスタム アクションをプロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックし、**[新しい PowerShell]** をポイントして、次のいずれかのコマンドをクリックします:
   - **Binary テーブルに保存**—コード ベースを Binary テーブルに格納する場合、このコマンドを選択します。この場所は、ターゲット システムにコード ファイルをインストールしない場合に便利です。
   - **製品と一緒にインストール**—ターゲット システムにインストールされるスクリプト ファイルからコードを呼び出す場合、このコマンドを選択します。

   PowerShell カスタム アクションがデフォルトの名前で追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このカスタム アクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. **"PowerShell スクリプト ファイル名"** 設定で、Binary テーブルに格納されているファイル リスト、またはプロジェクトに含まれているファイル リストから、PowerShell スクリプト ファイル (.ps1) を選択します。指定した場所が、Binary テーブルに格納されている場合、この設定で、省略記号ボタン (...) をクリックして、ファイルを参照します。
5. 必要に応じて、アクションの他の設定も構成します。

プロジェクトにカスタム アクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「アクションをシーケンスに挿入する」を参照してください。

PowerShell カスタム アクションの実行時要件

PowerShell は、デフォルトで、一部のオペレーティング システムにのみインストールされています。インストールで PowerShell カスタム アクションが起動されたときに PowerShell がインストールされていない場合、カスタム アクションは失敗します。

ターゲット システムに PowerShell がインストールされていることを確認する場合、プロジェクトに PowerShell の定義済みシステム検索を追加し、PowerShell カスタム アクションを構成して、システム検索で PowerShell がインストールされていることが判別された場合にのみ実行されるようにします。また、このシステム検索を使って、インストールする PowerShell のバージョンを判別したり、適切なターゲット システム上でのみ起動をトリガするカスタム アクションの条件を含めたりできます。デフォルトで、PowerShell の定義済みシステム検索は、POWERSHELLVERSION プロパティの値を PowerShell のバージョンに設定します。

Windows Installer プロパティ IS_CLR_VERSION を利用すると、カスタム アクションがダウンロードして PowerShell スクリプトを実行するセミコロンで区切られた .NET Framework のバージョン一覧を識別することができます。

POWERSHELLVERSION および IS_CLR_VERSION プロパティについての詳細は、「Windows Installer プロパティ リファレンス」を参照してください。

実行中の基本の MSI または InstallScript MSI インストールとの対話操作

実行中の基本の MSI または InstallScript MSI インストールとの対話操作を可能にする、いくつかの cmdlet があります:

テーブル 9 • 実行中の基本の MSI または InstallScript MSI インストールと対話操作を行う cmdlet

Cmdlet	説明
get-property -name [プロパティ名]	この cmdlet は Windows Installer プロパティの値を取得します。Name パラメーターは、位置指定パラメーターです。 次のサンプル コードは、MYPROPERTY という名前のプロパティの値を取得します。 $Value = get-property -name MYPROPERTY
set-property -name [プロパティ名] -value [値]	この cmdlet は Windows Installer プロパティの値を設定します。Name および Value パラメーターは位置指定パラメーターです。 次のサンプル コードは、MYPROPERTY プロパティの値を使って、NEWPROPERTY と呼ばれる 2 番目のプロパティの値を設定します。 $Value = get-property -name MYPROPERTY set-property -name NEWPROPERTY -value $Value
format-properties -Value [形式文字列]	この cmdlet は、形式化された式の値を展開します。Value パラメーターは、位置指定パラメーターです。 次のサンプル コードは、NEWPROPERTY と呼ばれる Windows Installer プロパティを含む文字列をフォーマットします。プロパティ名と、その周りの角かっこは、実行時にプロパティの値で置換されます。 $myFormat = format-properties -Value "これは、[NEWPROPERTY] が埋め込まれたテキスト文字列です。"
trace-info -LogMessage [長い文字列]	この cmdlet により、Windows Installer ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。LogMessage パラメーターは、位置指定パラメーターです。 次のサンプル コードは、文字列 **「これは PowerShell カスタム アクションです」**を Windows Installer ログに書き込みます。 trace-info -LogMessage “これは PowerShell カスタム アクションです”

スクリプトが成功したことを示すには、必ず 0 を戻します。たとえば、次の行でスクリプトを終了します:

exit(0)

実行可能ファイルを起動する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

サードパーティ ツールのインストール、Readme ファイルの表示、インストールする製品についての最新情報が記載された Web ページの表示など、インストールから実行可能ファイルを起動すると便利なことがあります。インストール内から実行可能ファイルを起動するには、[カスタム アクションとシーケンス] (基本の MSI、InstallScript MSI、MSI データベース、および トランスフォーム プロジェクトの場合) または [カスタム アクション] (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) にカスタム アクションを追加する必要があります。

カスタム アクション ウィザードを使用して、インストールから実行可能ファイルを起動するカスタム アクションを作成することができます。ウィザードの各パネルを下に示します。実行可能ファイルを起動するには各項目に値を入力する必要があります。

[基本情報] パネル

テーブル 10 • カスタム アクション ウィザードで [基本情報] パネル の設定を行う

パネル オプション	値
名前	カスタム アクションにわかりやすい名前を付けます。この名前は製品内部だけで、識別のために使用されます。
コメント	このカスタム アクションに関するコメントを入力します。

アクションの種類

テーブル 11 • カスタム アクション ウィザードの [アクションの種類] パネルにある設定

パネル オプション	値
Type	作成するカスタム アクションのタイプを選択します。この例では、[実行可能ファイルの起動] を選択します。
場所	選択は、インストール中に実行するターゲット実行可能ファイルの場所によって異なります。どのコンピューター上にも「メモ帳」はあるので、これがカスタム アクションのターゲットとなります。メモ帳は、ターゲット マシン上の Windows フォルダーにあることが保証されているため、Directory テーブルを使用してその位置をポイントすることができます。したがって、**[Directory テーブルに保存]** を選択します。

アクションのパラメーター

テーブル 12 • カスタム アクション ウィザードで [アクションのパラメーター] パネルの設定を行う

パネル オプション	値
ソース	Directory テーブルに保存されている実行可能ファイルを起動することを選択したため、現在 Directory テーブルにあるすべてのエントリを反映した選択リストが表示されます。これらのオプションの 1 つが **WindowsFolder** です。パスを直接入力するのではなく、このオプションを選択して「メモ帳」をポイントします。
ターゲット	このオプションでは、指定されたディレクトリ内にあるターゲットファイルを [ソース] オプションから入力する必要があります。**Notepad.exe** と入力して [次へ] ボタンをクリックし、続行します。

追加オプション

テーブル 13 • カスタム アクション ウィザードで [追加のオプション] パネルの設定を行う

パネル オプション	値
戻り値の処理	Windows Installer で、どのようにカスタム アクション スレッドの処理を制御するかを指定します。メインとカスタム アクションのスレッドが同期実行 (インストールがメイン インストール スレッドを再開する前にカスタム アクション スレッドが完了するのを待機する) か非同期 (インストールがメイン インストールの続行と同時にカスタム アクションを実行する) かを指定することができます。 この例では、**同期 (終了コードを確認)** オプションを選択します。

応答オプション

テーブル 14 • カスタム アクション ウィザードで [応答オプション] パネルの設定を行う

パネル オプション	値
スクリプト内実行	スクリプトの記述が表れたら、すぐにアクションを実行する場合には、[即時実行] を選択します。
実行スケジュール	カスタム アクションの記述が表れるたびにそれを実行する場合には、**[常に実行]** を選択します。

カスタム アクションのシーケンスへの挿入

カスタム アクションを作成後、[カスタム アクションとシーケンス] ビューでカスタム アクションをシーケンスに挿入する必要があります。

Msiexec.exe を使用して 2 番目の Windows Installer インストールを起動する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ネスト インストール カスタム アクション タイプを使用して 2 つ目の .msi パッケージを起動する代わりに、**Msiexec.exe** を起動するカスタム アクションを作成する方法があります。この方法で 2 つ目のインストールを起動すると、起動がその独自のプロセスで実行され、ターゲット システムに正しいエントリが作成されます。また、アンインストールプロセスは、この方法で 2 つ目のインストールを起動するとより効果的になります。レジストリ エントリは、正しくクリーンアップされ、参照カウントが正確に増加および減少されます。

カスタム アクションを作成する

まず、カスタム アクションを作成します。カスタム アクションを作成する最も簡単な方法は、カスタム アクション ウィザードを使用することです。

タスク ***ウィザードでカスタム アクションを作成するには、次の手順に従ってください。***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。**カスタム アクション ウィザード**が開きます。
3. **[基本情報]** パネルで、カスタム アクションの名前とコメントを指定、**[次へ]** をクリックして次のステップに進んでください。
4. **[アクション タイプ]** パネルの **[タイプ]** リストで、**[実行可能ファイルを起動]** を選択します。**[場所]** リストで、**[Directory テーブルに保存]** を選択します。Directory テーブルを使用すると、SystemFolder などの定義済みフォルダーから選択できます。ここに、Msiexec.exe が保存されます。
5. **[アクションのパラメーター]** パネルで、起動する実行可能ファイル (**Msiexec.exe**) を参照します。**[ターゲット]** ボックスを使用すると、起動する実行可能ファイルの名前、およびそのファイルを渡すコマンドライン パラメーターを指定できます。例:

   msiexec.exe /i "[SOURCEDIR]AnotherSetup¥SecondSetup.msi" /qb

   このエントリの最初の部分、**msiexec.exe** は、起動する実行可能ファイルの名前です。次のセクション、**/i "[SOURCEDIR]AnotherSetup¥SecondSetup.msi"** は、Msiexec.exe に実行するパッケージを知らせます。この場合、最初のインストールのソースフォルダーの下にある、フォルダーのファイルを指しています。最後に、**/qb** は、Windows Installer サービスに、最小限のユーザー インターフェイスによるインストールの実行を通知します。プログレスバーだけが表示されます。
6. **[追加のオプション]** パネルでデフォルトのオプションを選択し、ウィザードが完了するまで **[次へ]** をクリックします。

カスタム アクションをシーケンスへ挿入する

カスタム アクションが作成されたら、それをシーケンスに挿入する必要があります。

タスク ***カスタム アクションを配列するには、以下の手順を実行します。***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[ユーザー インターフェイス]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **CostFinalize** を右クリックし、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
4. 新しいカスタム アクションを選択します。
5. **[OK]** をクリックします。

次に、起動するインストールパッケージが [アクションパラメーター] パネルで指定したフォルダーにあることを確認し、インストールをビルドして実行します。

メモ: Msiexec.exe *を起動するカスタム アクションは [ユーザー インターフェイス] シーケンスに配置する必要があります。つまり、エンドユーザーがインストールをサイレント モードで実行すると、カスタム アクションは実行されません。*

ネスト インストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

重要: *Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

ネスト インストールは、実行中のインストーラー (*「親製品」*と呼ぶ) から別の .msi パッケージ (*「子製品」*と呼ぶこともある) をインストールまたは削除するカスタム アクションの一種です。

メモ: *ネストインストールのカスタム アクションを使用する前に、次の制約に注意してください。*

- *ネストインストールにはユーザー インターフェイスが表示されません。*

- *通常、子製品がエンド ユーザーのシステム上のコントロール パネルの中の [プログラムの追加と削除] に表示されることはなく、親製品がアンインストールされても、自動的にアンインストールされることはありません。*

ネスト インストール カスタム アクションを作成する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

***重要・** Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

タスク ***ウィザードでネストインストールカスタム アクションを作成するには、次の手順に従ってください。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。**カスタム アクション ウィザード**が開きます。
3. **[基本情報]** パネルで、カスタム アクションの名前とコメントを指定、**[次へ]** をクリックして次のステップに進んでください。
4. **[アクション タイプ]** パネルの **[タイプ]** リストで、**[別の .msi パッケージを起動]** を選択します。**[場所]** リストで、**[メインのセットアップに含める]** を選択します。
5. **[アクション パラメーター]** パネルで、起動する .msi ファイルの場所を参照します。(便宜上、子製品のすべてのファイルは .msi パッケージに圧縮されていると仮定します。)**[ターゲット]** ボックスを使って、子インストールに渡すコマンドライン スイッチを指定することができます。たとえば、デフォルト設定と共に子インストールのすべての機能をインストールし、また親インストールが利用したのと同じ ALLUSERS プロパティの値を使うときは次のように **[ターゲット]** ボックスに入力します。

   ARPSYSTEMCOMPONENT=1 ADDDEFAULT=ALL ALLUSERS=[ALLUSERS]

   同じ式を使って、親と同じように子でも INSTALLDIR と同じ値を使用することができます。
6. **[追加オプション]** パネルのデフォルトのオプションを選択し、ウィザードが完了するまで **[次へ]** をクリックします

ネストされたインストール カスタム アクションをシーケンスへ挿入する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

重要 *Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

カスタム アクションが作成されたら、それをシーケンスに挿入する必要があります。

タスク ***ネスト インストールをシーケンスに挿入するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[ユーザー インターフェイス]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **CostFinalize** を右クリックし、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
4. 新しいカスタム アクションを選択します。
5. **[条件]** ボックスで、**Not Installed** と入力します。これで、ネスト インストールは親製品がインストールされる初回インストール時にのみ実行されることを示します。
6. **[OK]** をクリックします。

子製品を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

重要・*Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

子製品は、親製品が削除されたときも自動的に削除されません。ただし、親製品を削除したときに一緒に自動的に削除されるような 2 番目のネストインストールカスタム アクションを作成することができます。

タスク ***カスタム アクションを作成するには以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。**カスタム アクション ウィザード**が開きます。
3. **[基本情報]** パネルで、 カスタム アクションの名前とコメントを指定、**[次へ]** をクリックして次のステップに進んでください。
4. **[アクション タイプ]** パネルの **[タイプ]** リストで、 **[別の .msi パッケージを起動]** を選択します。**[場所]** リストで、**[アドバタイズされるアプリケーション、または既にインストールされているアプリケーション]** を選択します。
5. **[アクションのパラメーター]**パネルで、削除する .msi ファイルの場所を参照します。**[ターゲット]** ボックスでは、デフォルトのプロパティをそのまま残します。

   ALLUSERS=[ALLUSERS] REMOVE=ALL
6. **[追加オプション]** パネルのデフォルトのオプションを選択し、ウィザードが完了するまで **[次へ]** をクリックします

タスク ***カスタム アクションを配列するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[実行]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **InstallValidate** を右クリックし、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
4. 新しいカスタム アクションを選択します。
5. **[条件]** ボックスで、次のように入力します。

   REMOVE="ALL"
6. **[OK]** をクリックします。

このカスタム アクションによって、親製品がアンインストールされたときに子製品がアンインストールされます。

メモ- [別の .msi パッケージの起動] するタイプのネスト インストール カスタム アクションを、[場所] の設定を [ソース メディアに保存] にして作成する場合、次の点に注意してください。

- *子インストーラーは、Setup.exe インストールランチャの中に圧縮できません。*
- *子インストールのファイルが子の .msi データベース内で圧縮されていない場合、子インストールのファイルを親インストールのリリースフォルダーの Disk1 フォルダーに手動でコピーする必要があります。*

インストールで MessageBoxA を呼び出す

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

タスク ***カスタム アクションで Windows API 関数 MessageBoxA() を呼び出してメッセージボックスを表示するには、以下の手順を実行します。***

パート A: カスタム アクション ウィザードを起動する

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。

パート B: カスタム アクションを開始する

1. **[基本情報]** パネルで、カスタム アクションの名前に**CA_Example** と入力します。**[次へ]** をクリックします。
2. **[アクションの種類]** パネルの [種類] リスト ボックスで、**[標準ダイナミック リンク ライブラリの関数を呼び出す]** を選択します。
3. **MessageBoxA** は、サポートされているすべての Windows プラットフォームに存在する User32.dll にエクスポートされるので、**[保存場所]** リストから **[インストール先のマシンの検索パス]** を選択します。続行するには **[次へ]** をクリックしてください。

パート C: 関数を定義する

MessageBoxA() の構文は次のとおりです。

```
int MessageBoxA (hWnd, lpText, lpCaption, uType);
```

この情報は、[関数定義] パネルで以下の手順で指定します。

1. **[名前]** ボックスで、**MessageBoxA** と入力します。
2. [**引数**] ボックスの最後の行をクリックして、最初のパラメーターを指定します。次のリストと同じ値になるよう、各パラメーターのフィールドを編集します。

   テーブル 15 - 引数ボックスのパラメーター

Type	ソース	値
HWND	定数	MsiWindowHandle
STRING	in プロパティ	MESSAGEPROP
STRING	in プロパティ	CAPTIONPROP
NUMBER	定数	1

   HWND タイプの **[値]** リストで、**MsiWindowHandle** を選択します。HWND タイプをこの値に設定すると、メッセージ ボックスが、インストール ウィンドウの手前に表示されます。MESSAGEPROP プロパティおよび CAPTIONPROP プロパティは、一覧からは使用できません。これらの名前は、**[値]** リストで入力すると、プロパティ マネージャーに追加されます。
3. **[戻り値の型]** リストで、**[void]** を選択します。(**MessageBoxA()** は数字を戻しますが、プロパティのその値のチェックについては、ここでは説明しません)。
4. **[次へ]** をクリックして **[アクションのパラメーター]** パネルに進みます。

パート D: MessageBoxA のソースを指定する

カスタム アクションウィザードで次に重要な設定は、[アクションのパラメーター] パネルの "ソース" フィールドです。MessageBoxA は User32.dll にあります。[参照] ボタンをクリックして、Windows システム フォルダーにある User32.dll を探します。

ウィザードの次のパネルでは、[次へ] をクリックしてデフォルト値をそのまま使用できます。続いて [概要] パネルで [完了] をクリックしてウィザードを閉じ、CA_Example をプロジェクトに追加できます。

パート E: プロパティの値を初期化する

MessageBoxA() の文字列パラメーターについては、プロパティは、上記手続きで説明説明されているように、提供されています。次に、これらのプロパティに値を入れます。

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックすると、製品のインストーラーのプロパティが表示されます。**カスタム アクション ウィザード**でこれらのプロパティを参照した結果として、MESSAGEPROP プロパティおよび CAPTIONPROP プロパティが作成されました。
2. **MESSAGEPROP** へ移動し、**Value** 列をクリックして値を入力します。値に「**このテキストが見えますか？** 」と入力します。
3. CAPTIONPROP プロパティを [**カスタム アクションの例**] に設定します。

パート F: カスタム アクションをシーケンスへ挿入する

カスタム アクションを実行するには、カスタム アクションをシーケンスに配置するか、またはカスタム アクションをダイアログのコントロール イベントの結果にする必要があります。

次の手順に従って、インストール プロジェクトの [インストール] シーケンスに CA_Example を挿入してください。

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、**[インストール]** シーケンスの下にある **[ユーザー インターフェイス]** アイテムを展開します。このシーケンスにスケジュールされているアクションとダイアログが一覧表示されます。
3. **MigrateFeatureStates** アクション (InstallWelcome ダイアログの直前のアクション) を右クリックして、**[挿入]** をクリックします。[**アクションの挿入**] ダイアログ ボックスが開きます。
4. ダイアログ ボックスの上にあるリストから、[**カスタム アクション**] (インストール プロジェクト) を選択します
5. カスタム アクションのリストから [**CA_Example**] を選択します。
6. **[OK]** をクリックします。

MigrateFeatureStates のすぐ後の [インストール] シーケンスに CA_Example が追加されます。

パート G: MessageBoxA の呼び出しをテストする

1. リリースをビルドします。(プロジェクトにコンポーネントやファイルを 1 つも追加していないと、出力ウィンドウでビルド エラーが表示されることがあります。)
2. **[ビルド]** メニューで **[実行]** をクリックします。

InstallShield がインストールを実行します。[ようこそ] ダイアログが表示されると、[カスタム アクションの例] というメッセージボックスが表示されます。

ターゲット システム上のファイルを検索する

[システム検索] ビュー

[システム検索] ビュー は、アプリケーションをインストールする前に特定のフォルダー、レジストリ、またはターゲット システム上の .ini ファイルを検索する機能を提供します。

ダイレクト エディター: 基本の MSI プロジェクト

Windows Installer は、ターゲット システム上のファイル検索の命令に、Signature、AppSearch および “locator” テーブルのレコードを使用します。Signature テーブルには、検出するファイルの情報を指定し、AppSearch テーブルには、検出された場合のファイル位置を完全パスで収めるプロパティを指定します。

たとえば、**FindMe.exe** というファイルを検索する場合は、ダイレクトエディターを使って Signature テーブルに次のレコードを追加します。

テーブル 16 • Signature テーブルのサンプル データ

テーブルの列	サンプル データ
Signature	findme_sig
FileName	FindMe.exe

Signature テーブルのその他のフィールドでは、オプションのバージョン、サイズ、データおよび言語情報を指定することができます。

AppSearch テーブルには、次のレコードを追加します。

テーブル 17 • AppSearch テーブルのサンプル データ

テーブルの列	サンプル データ	説明
プロパティ	LOCATION_OF_FINDME	パブリック プロパティにする必要があります。
Signature_	findme_sig	Signature テーブルで使用したのと同じ名前。

Windows Installer がファイルの検索を始める場所を指定することができる 4 つのロケータ テーブル、CompLocator、RegLocator、IniLocator、および DrLocator があります。特定のディレクトリ内のファイル検索には、DrLocator テーブルを使います。たとえば、ユーザーの Program Files ディレクトリ内にある **FindMe.exe** を検索する場合、DrLocator テーブルに次のレコードを追加します。

テーブル 18 • DrLocator テーブルのサンプル データ

テーブルの列	サンプル データ
Signature	findme_sig
Parent	
パス	[ProgramFilesFolder]
Depth	2

AppSearch アクションの実行後、**FindMe.exe** が見つかった場合は、パブリック プロパティ LOCATION_OF_FINDME にエンド ユーザーのシステム上でのこのファイルへの完全パスが収められ、ファイルが見つからなかった場合、このプロパティは未定義になります。

InstallScript: InstallScript MSI プロジェクト

FindFile および **FindAllFiles** 関数を使用すると、ターゲット システム上に存在するファイルを検索できます。たとえば、ユーザーの Program Files フォルダーから **FindMe.exe** というファイルを検索する場合、**OnAppSearch** イベント ハンドラー関数を次のように実装します。

```
function OnAppSearch()
    STRING svFoundFile;
begin
    FindAllFiles(PROGRAMFILES, "FindMe.exe", svFoundFile, RESET);
    MessageBox("FindMe.exe が検出されました: " + svFoundFile, INFORMATION);
end;
```

インストールが完了した後にアプリケーションを起動する

LaunchApplication 関数は、実行可能ファイルを起動します。たとえば、次のコードは、エンドユーザーがプログラムウインドウを閉じたとき、制御をスクリプトに戻して、エンドユーザーの INSTALLDIR フォルダーにある **Program.exe** のコピーを起動します。

```
LaunchApplication( INSTALLDIR ^ "Program.exe", "", "", SW_NORMAL,"",  LAAW_OPTION_WAIT );
```

プロジェクト: *InstallScript プロジェクトは INSTALLDIR ではなく TARGETDIR を利用します。*

InstallScript カスタム アクション内部からインストールを終了する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript カスタム アクションを含む基本の MSI*
- *InstallScript MSI*
- *InstallScript カスタム アクションを含むマージ モジュール*

ERROR_INSTALL_FAILURE という値を返す InstallScript エントリポイント関数によって、インストールが終了します。

スクリプトを通して ODBC プロパティを変更する

ODBC プロパティ (DBQ、 SystemDB など) をスクリプトを使用して変更する場合、これらのオプションを構成して、インストーラーのプロパティを使用します。そうすると、スクリプトからインストーラー プロパティを実行時に設定できます。たとえば、InstallShield インターフェイスで SystemDB を **[SYSTEM_DB_DIR]¥MyDatabase** に設定します。そうすると、スクリプトから **MsiSetProperty** を呼び出して、SYSTEM_DB_DIR の値を設定することができます。

ヒント: OnMoving イベントの前に、できれば OnFirstUIBefore でプロパティ値を設定してください。その後になると、インストーラーは既に内部スクリプト情報を作成済みで、これを変更することはできないため手遅れになります。

- *プロパティ名にはすべて大文字を利用します。*
- *Directory テーブルに SYSTEM_DB_DIR が存在する場合、ODBC 属性設定作成時に右かっこ (]) の後の円記号 (¥) を省略します。プロパティマネージャーで定義した場合、円記号が必ず必要です。*

VBScript カスタム アクションで INSTALLDIR プロパティを使用する

***プロジェクト:** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

VBScript カスタム アクションで INSTALLDIR プロパティを使用する必要がある場合、Session オブジェクトの Property プロパティを使用します。このオブジェクトはすべての VBScript カスタム アクションでアクセス可能です。

サンプルコードは次の通りです。

```
szInstallDir = Session.Property("INSTALLDIR")
```

詳細は、Windows Installer ヘルプライブラリの「Session オブジェクト」を参照してください。

アクション テキストを使う

***プロジェクト:** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンド ユーザーに進行状況を伝えるために、インストールは一般的に、実行中の処理を説明する進行状況ダイアログで説明テキストを表示します。通常、インストールのステータスを表示する進行状況バーも同時に表示されます。標準アクションおよびカスタム アクションが発生すると、そのアクションについてのメッセージが進行状況ダイアログに表示されます。この機能は、実行するのに時間がかかるアクションで、特に役立ちます。インストールのログ記録が作成される場合、同じアクション テキストが記録されます。

また、アクションは処理の必要があるアクション データを Windows Installer に送ります。データが処理された後、進行状況ダイアログにその表示が可能となります。

たとえば、InstallFiles アクションを実行中、デフォルトで進行状況ダイアログにはアクション テキスト、「新しいファイルをコピーしています」が表示されます。このアクションテキストは、インストールのログ ファイルにも記録されます。アクションが発生するときに、進行中の処理についての詳細情報を提供するには、進行状況ダイアログにアクションの詳細を表示するコントロールを追加して、そのコントロールが ActionData イベントをサブスクライブするようにします。InstallFiles アクションの場合、アクションの詳細として、ターゲット システム上にインストールされる各ファイルの名前、ディレクトリ、およびサイズが含まれます。

プロジェクト: *InstallScript MSI インストールでは、状態ダイアログにアクション データを表示できません。ただし、そのテンプレートはインストールのログ ファイルに記録されます。*

アクションの説明とアクション データのテンプレートを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield には、標準アクションとビルトイン InstallShield カスタム アクション用のデフォルトのアクションの説明が含まれています。また、適切な場合、これらのアクションの多くには、デフォルト テンプレートも用意されています。テンプレートは、アクション データ使用するフォーマットが示されています。

プロジェクトに含まれる標準アクションまたはビルトイン InstallShield カスタム アクションのデフォルトの説明またはテンプレートを変更しなくてはならない場合があります。また、独自のカスタム アクションのアクション テキストおよびアクション データを指定する必要がある場合があります。

タスク ***プロジェクトに含まれる説明およびアクション データ テンプレートを指定するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[アクション テキスト]** エクスプローラーを右クリックしてから、**[新規]** をクリックします。新しいアクション テキスト アイテムが追加されます。
3. アクション テキスト アイテムの名前として、アクションの名前を入力します。

重要: *[アクション テキスト] エクスプローラーの下にあるアクション テキスト項目の名前は、プロジェクトに含まれる標準およびカスタム アクションの名前と一致します。カスタム アクションの名前を変更する場合は、そのアクション テキスト項目の名前も変更しなくてはなりません。そうしないと、実行時にアクション テキストが表示されないか、インストールのログ ファイルに記録されません。ただし、プロジェクトに含まれるすべてのアクションに対してアクション テキストを作成する必要はありません。*

4. 右側のペインで、説明とテンプレートを指定します。詳細については、「アクション テキストの設定」を参照してください。

ヒント: *[アクション テキスト] エクスプローラーの下に既にリストされているアクションの説明またはテンプレートを編集するには、そのアクションを選択してから、右側のペインで適切な値を指定します。*

進行状況ダイアログにアクションの説明を表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallScript MSI インストールでは、進行状況ダイアログ (STATUSEX ダイアログ) が Enable (STATUSEX) の呼び出しで有効化されている場合、自動的にこのダイアログにアクション テキストが表示されます。このダイアログを編集することはできません。

デフォルトで、インストールに含まれるアクションに説明が指定されている場合、その説明は進行状況ダイアログの進行状況バーの上に表示されます。このため、プロジェクトに含まれるこのダイアログに何らかの変更を加えない限り、次のステップを行う必要はありません。

タスク ***進行状況ダイアログにアクションの説明を表示するには、以下の手順に従います:***

1. **SetupProgress** ダイアログで、アクションの説明を表示するコントロールを追加するには、以下の手順に従います:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、**SetupProgress** ダイアログを展開します。
   c. **SetupProgress** ダイアログの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
   d. **[コントロール]** ツールバーで、**[テキスト領域]** コントロールをクリックします。
   e. ダイアログ上でアクションの説明を表示する位置に長方形を描きます。
2. コントロールを選択した状態で、右のペインでその設定を構成します:
   a. **(Name)** 設定に、次のように入力します:

      ActionText

   b. **Text** 設定から、値を削除します。
3. ActionText イベントのサブスクリプションを新しいコントロールに追加します:

a. **SetupProgress** ダイアログの下にある **[ダイアログ]** エクスプローラーで、**[動作]** アイテムをクリックします。中央のペインにあるグリッドに SetupProgress ダイアログのすべてのコントロールが表示されます。

b. 作成した **ActionText** コントロールを選択してから、右下のペインにある **[サブスクリプション]** タブをクリックします。

c. 右上のペインで、**Event** フィールドに **ActionText** が選択されたレコードを追加します。**Attribute** フィールドには、**Text** と入力します。

実行時に、インストールが完全または簡易ユーザー インターフェイスで実行されたとき、進行状況ダイアログに、各アクションが発生するたびにその説明が表示されます。あるアクションに説明が定義されていない場合、そのアクションが起動されたときに進行状況ダイアログで説明は表示されません。

進行状況ダイアログにアクション データを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

InstallShield MSI インストールでは、進行状況ダイアログ (STATUSEX ダイアログ) で、アクション データを表示することはできません。ただし、InstallScript で Enable (INDVFILESTATUS) を呼び出すことで、同じような結果を得ることができます。INDVFILESTATUS 定数を **Enable** *関数を使って呼び出すと、***FeatureMoveData**、**CopyFile**、*または* **XCopyFile** *が呼び出されたときに、進行状況インジケーターが有効化されている場合、STATUSEX ダイアログに転送中の各ファイルの完全修飾ファイル名が表示されます。InstallScript MSI プロジェクトで、アクションにテンプレートを指定した場合、そのテンプレートがインストールのログ ファイルに書き込まれます。*

実行時にアクションが起動されるとき、進行中の処理についての詳細情報を提供するには、進行状況ダイアログにアクションの詳細を表示するコントロールを追加して、そのコントロールが ActionData イベントをサブスクライブするようにします。たとえば、InstallFiles アクションの実行中にアクションの詳細を表示するコントロールには、ターゲット システム上にインストールされている各ファイルの名前、ディレクトリ、およびサイズといった内容が含まれます。

タスク ***進行状況ダイアログにアクション データを表示するには、以下の手順に従います:***

1. **SetupProgress** ダイアログで、アクションの詳細を表示するコントロールを追加するには、以下の手順に従います:

   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、**SetupProgress** ダイアログを展開します。

   c. **SetupProgress** ダイアログの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。

   d. **[コントロール]** ツールバーで、**[テキスト領域]** コントロールをクリックします。

e. ダイアログ上でアクションの説明を表示する位置に長方形を描きます。

2. コントロールを選択した状態で、右のペインでその設定を構成します:

   a. **(Name)** 設定に、次のように入力します:

      ActionData

   b. **Text** 設定から、値を削除します。

3. ActionData イベントのサブスクリプションを新しいコントロールに追加します:

   a. **SetupProgress** ダイアログの下にある **[ダイアログ]** エクスプローラーで、**[動作]** アイテムをクリックします。中央のペインにあるグリッドに SetupProgress ダイアログのすべてのコントロールが表示されます。

   b. 作成した **ActionData** コントロールを選択してから、右下のペインにある **[サブスクリプション]** タブをクリックします。

   c. 右上のペインで、**Event** フィールドに **ActionData** が選択されたレコードを追加します。**Attribute** フィールドには、**Text** と入力します。

実行時に、インストールが完全ユーザー インターフェイスで実行されたとき、進行状況ダイアログに、各アクションが発生するたびにその詳細が表示されます。あるアクションにテンプレートが定義されていない場合、そのアクションが起動されたときに進行状況ダイアログで詳細が表示されません。

アクションの説明またはアクション データのテンプレートを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***プロジェクトに含まれる説明またはアクション データ テンプレートを削除するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[アクション テキスト]** エクスプローラーで、その説明またはテンプレートを削除するアクションを選択します。
3. **Description** 設定または **Template** 設定から、値を削除します。

メモ: *アクションの説明またはアクション テンプレートのどちらも使用しない場合、プロジェクトからアクション テキスト レコードを削除できます。その場合、[アクション テキスト] エクスプローラーで、削除するアクション テキスト アイテムを右クリックしてから、[削除] を選択します。*

以下のアクションのアクション テキストを削除することはできません:

- *GenerateScript*

- *Rollback*
- *RollbackCleanup*

さらに、これらの各アクションの "テンプレート" 設定の値は、常に [1] とします。

シーケンスを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストールのシーケンスを定義することは、インストール パッケージ開発の重要な部分です。シーケンスは、Windows Installer で、インストール プロセスを制御する標準およびカスタム アクションが起動される順番を指定します。

プロジェクト: *基本の MSIおよびトランスフォーム プロジェクトでは、ダイアログが表示される順番もシーケンスで指定されます。*

InstallScript MSI プロジェクトでは、ユーザー インターフェイス ダイアログは、InstallScript コードで定義され、インストールのユーザー インターフェイス部分は InstallScript エンジンによって制御されます。

InstallShield の [カスタム アクションとシーケンス] ビューで、プロジェクトのシーケンスを定義することができます。プロジェクトのアクションとダイアログは、このビューの [シーケンス] エクスプローラーで、該当のシーケンスで起動されるタイミングにしたがって時系列で表示されます。各アクションとダイアログは、シーケンス上の番号が付与され、Windows Installer は、シーケンスを小さい番号から順に実行されます。

プロジェクトに追加したすべてのカスタム アクションとダイアログに手動で数値を割り当てることなく、[カスタム アクションとシーケンス] ビューを使用して、アクションまたはダイアログをシーケンスに挿入するか、シーケンス タイムラインを編集することができます。シーケンスについての詳しい情報についてはヘルプの同セクションを参照してください。

プロジェクト: *カスタム アクションまたはカスタム ダイアログをマージ モジュールで作成する場合、まずモジュールをインストール プロジェクトにインポートし、[カスタム アクションとシーケンス] ビューを使用してそれをシーケンスに追加する必要があります。*

[インストール] シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

- *MSI データベース*
- *トランスフォーム*

[インストール] シーケンスは、エンドユーザーが新しい .msi ファイルをダブル クリックする操作などの、デフォルトのインストール モードでインストールを起動する場合に実行される一連のアクションです。実行されるアクションは、以下の 2 種類に分けられます。

- ユーザー インターフェイス
- 実行

プロジェクト: *以下に説明する [ユーザー インターフェイス] ダイアログは基本の MSI プロジェクト用のものです。InstallScript MSI プロジェクト については、InstallScript コードがインストールのユーザー インターフェイスを実行します。*

InstallScript MSI プロジェクトは、自動的に ISVerifyScriptingRuntime カスタム アクションを含みます。詳細については、「ISVerifyScriptingRuntime」を参照してください。

ユーザー インターフェイス

[ユーザー インターフェイス] シーケンスには、完全なユーザー インターフェイスをサポートするのに必要なアクションおよびダイアログがすべて含まれています。このシーケンスは、インストールをサイレント モードで実行する場合にはスキップされます。

標準のアクションに関する技術的な詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」を参照してください。

テーブル 19 - [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ

アクションまたはダイアログの名前	イベントの種類	説明
SetupCompleteError	ダイアログ	このダイアログは、インストールが致命的なエラーのために終了した場合に表示されます。
SetupInterrupted	ダイアログ	このダイアログは、ユーザーがインストールを終了した場合に表示されます。
SetupCompleteSuccess	ダイアログ	このダイアログは、インストールが正常に終了した場合に表示されます。
ISSetupFilesExtract	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューに追加した任意のファイルを抽出します。詳細については、「サポート ファイルを使用する」を参照してください。
ISSetAllUsers	カスタム アクション	ISSetAllUsers カスタム アクションは、Upgrade テーブル に 1 つ以上の記録があると、[ユーザー インターフェイス] シーケンスと [実行] インストール シーケンスの両方に挿入されます。

テーブル 19 • [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
AppSearch	標準アクション	このアクションは、ターゲット システム上のファイルを識別および検索するのに使用できます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。このアクションの使用についての詳細は、ナレッジベース記事 Q103147 を参照してください。
LaunchConditions	標準アクション	インストールを続行するために True を返す必要のある条件を評価します。条件のどれかが失敗した場合、エラーメッセージが表示され、インストールは終了します。起動条件は、[一般情報] ビューで編集できます。
SetupInitialization	ダイアログ	このダイアログは、セットアップの開始準備中に表示されます。このダイアログのデフォルトのテキストは、"インストールの準備をしています..." です。
FindRelatedProducts	標準アクション	このアクションを実行すると、インストーラーは各々のインストールした製品のアップグレード コードをパッケージの Upgrade テーブルにあるリストと比較します。コードが一致した場合、インストールしたパッケージの製品コードが Upgrade テーブル の ActionProperty 列に追加されます。
CCPSearch	標準アクション	CCPSearch アクションを使用すると、アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。
RMCCPSearch	標準アクション	RMCCPSearch アクションを使用すると、競合アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。
ValidateProductID	標準アクション	製品 ID の検証アクションは、ProductID プロパティを完全な製品識別子に設定する場合に使用します。この検証により、ユーザーに製品 ID の入力を要求し、違法な使用から製品を保護することができます。
CostInitialize	標準アクション	このアクションは、インストールの現在の設定に必要なディスク容量を判断する最初のステップです。コスティングを完了するには、CostInitialize アクションと CostFinalize アクションを組み合わせて使用する必要があります。

テーブル 19 • [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
FileCost	標準アクション	FileCost アクションは、インストールの現在の設定に必要なディスク容量を判断します。より新しいバージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。コスティングを完了するには、CostInitialize アクションと CostFinalize アクションを組み合わせて使用する必要があります。
IsolateComponents	標準アクション	このアクションは、コンポーネント (通常は DLL ファイル) を孤立した場所にインストールし、特定のアプリケーションだけがそのコンポーネントを使用するようにします。
setUserProfileNT	カスタム アクション	このカスタム アクションは USERPROFILE ディレクトリ識別子を初期化します。
SetAllUsersProfileNT	カスタム アクション	このカスタム アクションは ALLUSERSPROFILE ディレクトリ識別子を初期化します。
setAllUsersProfile2K	カスタム アクション	このカスタム アクションは ALLUSERSPROFILE ディレクトリ識別子を初期化します。
ResolveSource	標準アクション	ソースの位置を検出し、SourceDir プロパティを設定します。
CostFinalize	標準アクション	ディスク容量の計算アクションは、現在の構成のインストールで必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。ディスク容量の計算アクションの前に、ディスク容量計算の初期化アクションおよびダイナミックディスク容量計算の初期化アクションを呼び出す必要があります。呼び出さなかった場合、アクションは失敗します。
SetARPReadme	カスタム アクション	SetARPReadme カスタム アクションは、[一般情報] ビューの "Readme" 設定で使用されるディレクトリ識別子を解決します。このカスタム アクションが必要な理由は、ARPREADME が Windows Installer プロパティであるため、自動的にフォーマットされないためです。
MigrateFeatureStates	標準アクション	このアクションは、アプリケーションのアップグレード中に使用されます。元のインストールの機能の状態がターゲットマシンから読み込まれ、その後アップグレードした機能に適用されます。

テーブル 19 • [インストール] シーケンスのユーザー インターフェイス アクションとダイアログ (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
PatchWelcome	ダイアログ	パッチパッケージが完全ユーザー インターフェイスに適用されると、PatchWelcome ダイアログが表示されます。ここには、正しいオプションを使って REINSTALL および REINSTALLMODE を設定するためのコントロール イベントが含まれています。ただし、ユーザー インターフェイスが抑制されている場合、プロパティをコマンドラインで設定する必要があります。
InstallWelcome	ダイアログ	InstallWelcom ダイアログを使用すると、InstallScript MSI インストールが実行されたときに [ようこそ] パネルが表示されます。
SetupResume	ダイアログ	このダイアログは、それまでにキャンセルされたインストールが再開された場合に表示されます。
MaintenanceWelcome	ダイアログ	このダイアログは、エンド ユーザーがプログラムのインストール機能の変更、プログラムの削除、プログラムの再インストール、2 回目のインストールの実行、[プログラムの追加と削除] パネルでの現在の製品の選択を試みたときに表示されます。
SetupProgress	ダイアログ	このダイアログはインストールの進行状況を表示します。
ExecuteAction	標準アクション	このアクションは EXECUTEACTION プロパティを参照し、最初に呼び出すトップレベルのアクションを判断してそのアクションを呼び出します。トップレベルアクションには、INSTALL、ADVERTISE、ADMIN などのアクションがあります。通常、このアクションによって [インストール]–[実行] シーケンスが開始されます。
ISSetupFilesCleanup	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューでファイルを追加したとき表示されます。詳細については、「サポート ファイルを使用する」を参照してください。

実行

[実行] シーケンスには、マシンの状態を変更し、正常に機能するためにユーザー インターフェイスに依存しないすべてのアクションが含まれます。これらのアクションには、ファイル転送、コンポーネントおよび機能のパブリッシュ、COM サーバーの登録などがあります。[実行] シーケンスの多くは、シーケンスに挿入されたカスタム アクション以外、テストモードでインストールを実行したときにスキップされます。

テーブル 20 ・ [インストール] シーケンスでアクションとダイアログを実行する

アクションまたはダイアログの名前	イベントの種類	説明
ISSetupFilesExtract	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューに追加した任意のファイルを抽出します。詳細については、「サポート ファイルを使用する」を参照してください。
ISSetAllUsers	カスタム アクション	ISSetAllUsers カスタム アクションは、Upgrade テーブル に 1 つ以上の記録があると、[ユーザー インターフェイス] シーケンスと [実行] インストール シーケンスの両方に挿入されます。
AppSearch	標準アクション	このアクションは製品の以前のバージョンを識別、検索するのに使用できます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。このアクションの使用についての詳細は、ナレッジベース記事 Q103147 を参照してください。
LaunchConditions	標準アクション	インストールを続行するために True を返す必要のある条件を評価します。条件が満たされない場合、インストールはエラーメッセージを表示してインストールを終了します。起動条件は、[一般情報] ビューで編集できます。
FindRelatedProducts	標準アクション	このアクションを実行すると、インストーラーは各々のインストール済み製品のアップグレード コードをパッケージの Upgrade テーブル (ダイレクト エディター でサポートされている) にあるリストと比較します。コードが一致した場合、インストールしたパッケージの製品コードが Upgrade テーブル の ActionProperty 列に追加されます。
CCPSearch	標準アクション	CCPSearch アクションを使用すると、アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。
RMCCPSearch	標準アクション	RMCCPSearch アクションを使用すると、競合アップグレードできる製品がエンド ユーザーのシステムにあるかどうかを確認することができます。このアクションに必要な Signature テーブルは、ダイレクト エディターで使用可能です。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
ValidateProductID	標準アクション	ValidateProductID アクションは、ProductID プロパティを完全な製品識別子に設定する場合に使用します。検証により、ユーザーに製品 ID の入力を要求し、違法な使用から製品を保護することができます。
CostInitialize	標準アクション	このアクションは、現在のインストール構成に必要なディスク容量を判断する最初のステップです。
FileCost	標準アクション	現在のインストール構成に必要なディスク容量を判断します。新規バージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。
IsolateComponents	標準アクション	このアクションは、コンポーネント (通常は DLL ファイル) を孤立した場所にインストールし、特定のアプリケーションだけがそのコンポーネントを使用するようにします。
CostFinalize	標準アクション	CostFinalize アクションは、現在のインストール構成に必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。
SetARPINSTALLLOCATION	カスタム アクション	SetARPINSTALLLOCATION カスタム アクションは、ARPINSTALLLOCATION プロパティをアプリケーションのプライマリ フォルダーの完全修飾パスに設定します。
SetODBCFolders	標準アクション	SetODBCFolders アクションは、ターゲット システムに既存の ODBC ドライバーがないか確認し、既存のドライバーの場所に新しい各ドライバーのターゲットディレクトリを設定します。
MigrateFeatureStatus	標準アクション	このアクションは、アプリケーションのアップグレード中に使用されます。元のインストールの機能の状態がターゲットマシンから読み込まれ、その後アップグレードした機能に適用されます。
InstallValidate	標準アクション	InstallValidate アクションは、現在のインストール構成に必要なディスク容量があるかどうかを判断します。
RemoveExistingProducts	標準アクション	RemoveExistingProducts アクションによって、コードが OLDPRODUCTS に表示される製品のサイレントアンインストールが実行されます。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
InstallInitialize	標準アクション	InstallInitialize アクションは、エンド ユーザーのシステムに変更を加えるアクションの開始を知らせます。
AllocateRegistrySpace	標準アクション	インストーラーはこのアクションを実行する時に、システムに少なくとも AVAILABLEFREEREG プロパティで指定された使用可能なレジストリ空き領域があることを確認します。
ProcessComponents	標準アクション	ProcessComponents アクションは、コンポーネントの登録と登録解除を行います。このアクションはコンポーネントのキーパスおよびコンポーネントが持つその他のクライアントの登録、および登録解除も行います。
UnpublishComponents	標準アクション	アンインストール中に呼び出され、最初のインストールでパブリッシュされたコンポーネントを、他のアプリケーションでパブリッシュされたものであっても、パブリッシュ解除します。
MsiUnpublishAssemblies	標準アクション	MsiUnpublishAssemblies アクションは、オペレーティング システムのアセンブリのアンインストールを処理します。
UnpublishFeatures	標準アクション	UnpublishFeatures アクションは、インストールでパブリッシュされた機能への参照をすべて削除します。このようなリファレンスには、選択状態および機能コンポーネント マッピング情報を含むレジストリ エントリなどがあります。
StopServices	標準アクション	このアクションは、停止するように構成されている Windows サービスを停止します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。
DeleteServices	標準アクション	このアクションは、削除するように構成されている Windows サービスを削除します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。
UnregisterComPlus	標準アクション	このアクションは、COM+ アプリケーションを登録解除します。
SelfUnregModules	標準アクション	このアクションは、SelfReg テーブルのデータによって登録されたファイルを登録解除します。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
UnregisterTypeLibraries	標準アクション	アンインストール中、アンインストールするようマークされた TypeLib テーブル内のすべてのファイルを登録解除します。このテーブルは、[COM 登録] 詳細設定で新規 TypeLib を作成した際に埋められます。
RemoveODBC	標準アクション	アンインストール中、ODBCDataSourceテーブル、ODBCTranslator テーブル、および ODBCDriver テーブルを参照し、アンインストールでどの ODBC リソースを削除するかを判断します。削除するようマークされたリソースは、アンインストールされます。
UnregisterFonts	標準アクション	アンインストール用に設定されたすべてのフォントに関する情報を登録解除します。
RemoveRegistryValues	標準アクション	RemoveRegistryValues アクションは、次の条件がすべて満たされている場合、エンド ユーザーのレジストリから値を削除します。 • 値が Registry テーブルに入力されている場合。 • 値がアンインストールのためにマークされいる場合。 • レジストリ エントリが属するコンポーネントが、[ソースから実行]、または [ローカルにインストール] に設定されている場合。
UnregisterClassInfo	標準アクション	このアクションは、アンインストールされている機能に属する COM クラスのシステムレジストリ情報を削除します。
UnregisterExtensionInfo	標準アクション	アンインストール中にエンド ユーザーのシステムから、拡張子に関するすべての情報を登録解除します。
UnregisterProgIdInfo	標準アクション	アクションは [ファイルの種類] 詳細設定に作成された ProgIDs すべてを登録解除します。
UnregisterMIMEInfo	標準アクション	アンインストールされている現在の機能の MIME テーブルを照会し、サーバーの MIME 情報を登録解除します。この情報は、ファイルの種類 詳細設定で入力します。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
RemoveIniValues	標準アクション	このアクションは IniFile テーブルで (または [INI ファイルの変更] ビューを使って) コンポーネントに関連付けられた .ini 情報のみを削除します。このアクションは、IniFile テーブルに情報が存在することを確認した後、RemoveIniFile テーブルにリストされているすべての .ini ファイルの関連コンポーネントがアンインストール用にマークされ、かつローカルにインストールされた、またはソースから実行するように設定された場合に、これらの .ini ファイルを削除します。また、RemoveIniValue アクションでは、WriteIniValues アクションで書かれたすべての .ini ファイルも、関連コンポーネントがアンインストール用にマークされている場合、削除されます。
RemoveShortcuts	標準アクション	アンインストール用にマークされた機能の、アドバタイズされたショートカットをすべて削除します。また、アンインストールするようマークされたコンポーネントの、アドバタイズされていないショートカットもすべて削除されます。
RemoveEnvironmentStrings	標準アクション	コンポーネントが削除されると、このアクションは、WriteEnvironmentStrings アクションによってインストール中や再インストール中に環境変数に加えられた変更を元に戻します[環境変数] ビューを使って、環境変数の変更を指定できます。
RemoveDuplicateFiles	標準アクション	DuplicateFiles アクションで作成されたファイルを削除します。このアクションを正常に完了するには、複製されたファイルに関連したコンポーネントを、アンインストールするようマークしておく必要があります。
RemoveFiles	標準アクション	InstallFiles アクションでインストールされたファイルが、ソースから実行するかローカルにインストールされ、関連したコンポーネントがアンインストールするようマークされている場合に、これらのファイルを削除します。
RemoveFolders	標準アクション	Remove Folders アクションは、アンインストールするようマークされ、ソースから実行されたコンポーネントに関連している空のフォルダーを登録解除します。
CreateFolders	標準アクション	ローカルでインストールするよう設定されたコンポーネント用に、空のフォルダーを作成します。作成された新規フォルダーはその後、関連するコンポーネント GUID (コンポーネントの"コンポーネント コード" プロパティ 内) で登録されます。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
MoveFiles	標準アクション	このアクションにより、ターゲット システムに既に存在するファイルを移動またはコピーすることができます。このアクションで使用する MoveFiles テーブルは、ダイレクト エディターで使用可能です。
InstallFiles	標準アクション	InstallFiles アクションでは、選択されたコンポーネント機能がインストールするようマークされている場合、その機能のファイルすべてをターゲットマシンにコピーします。ローカルにインストールされるコンポーネントに関連したファイルのみが、ターゲットマシンにコピーされます。
PatchFiles	標準アクション	Patch テーブルを照会して、インストールされたファイルに使用できるパッチを判断します。これらのファイルに対してビットワイズパッチ処理が行われます。
DuplicateFiles	標準アクション	DuplicateFiles アクションは、InstallFiles アクションでインストールされた特定のファイルのコピーを作成します。これらのファイルは、元のファイルと同じディレクトリに別の名前でコピーするか、別のディレクトリに同じ名前でコピーすることができます。このアクションで使用する DuplicateFiles テーブルは、ダイレクト エディターで使用可能です。
BindImage	標準アクション	インポートした DLL ファイル関数の仮想アドレスを、BindImage テーブルの指定どおりにファイルのインポートアドレス テーブルに書き込みます。BindImage テーブルは、ダイレクト エディター で使用できます。
CreateShortcuts	標準アクション	このアクションは [ショートカット] ビューまたは [セットアップのデザイン] ビューのショートカット エクスプローラーで指定したショートカットを作成します。
RegisterClassInfo	標準アクション	このアクションは、[COM 登録] 詳細設定で指定した、またはコンポーネント ウィザードで抽出された、あるいはコンポーネントの "ビルド時に COM 抽出" 設定で指定されたすべての COM クラス情報を登録します。
RegisterExtensionInfo	標準アクション	RegisterExtensionInfo アクションは、[ファイルの種類] 詳細設定で指定したすべての拡張子を登録します。
RegisterProgIdInfo	標準アクション	このアクションは、詳細設定で指定し、インストールするようマークされたクラスサーバーまたは拡張サーバーにリンクした ProgID をすべて登録します。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
RegisterMIMEInfo	標準アクション	このアクションは、[ファイルの種類] 詳細設定で指定し、インストールするようマークされたクラスサーバーまたは拡張サーバーにリンクした MIME タイプをすべて登録します。
WriteRegistryValues	標準アクション	レジストリデータに関連したコンポーネントがインストールするようマークされ、ソースから実行するよう設定されているかローカルでインストールされている場合に、そのレジストリデータをターゲット システムに書き込みます。このレジストリ情報は、[レジストリ] ビューで作成したデータと同じです。
WriteIniValues	標準アクション	このアクションに関連するコンポーネントがローカルでインストールするかソースから実行するよう設定されている場合に、.ini ファイルに情報を書き込みます。このアクションおよびこれに対応するテーブルは、[INI ファイルの変更] ビューに表示されます。
WriteEnvironmentStrings	標準アクション	コンポーネントがインストールされると、このアクションは Environment テーブル、または [環境変数] ビューで指定されたシステムの環境変数を変更します。
RegisterFonts	標準アクション	インストールに含むフォントを登録します。多くの場合、これらのフォントは、コンポーネント ウィザードを使用してセットアップに含みます。
InstallODBC	標準アクション	[ODBC リソース] ビューで指定した ODBC リソースのドライバー、トランスレーター、およびデータソースすべてをインストールします。
RegisterTypeLibraries	標準アクション	このアクションは、[COM 登録] 詳細設定またはコンポーネント ウィザード を使用して、インストールに作成したライブラリを登録します。
SelfRegModules	標準アクション	このアクションは、SelfReg テーブルに表示された自己登録モジュールを登録します。このアクションは、デフォルトのユーザー権限で実行されます。
RegisterComPlus	標準アクション	このアクションは、COM+ アプリケーションを登録します。
InstallServices	標準アクション	このアクションは、インストールするように構成されている Windows サービスをインストールします。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
MsiConfigureServices	標準アクション	このアクションは、Windows サービスの拡張カスタマイズオプションを構成します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。 ***メモ:*** *このアクションは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
StartServices	標準アクション	このアクションは、開始するように構成されているすべてのサービスを開始します。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。
RegisterUser	標準アクション	プログラムのユーザーを識別するため、ユーザー情報を登録します。
RegisterProduct	標準アクション	製品をインストーラーで登録し、インストーラーデータベースをターゲットマシンに保存します。
PublishComponents	標準アクション	このアクションは、アドバタイズされた機能に関連するコンポーネントすべてをパブリッシュします。
PublishFeatures	標準アクション	各機能のインストール状態を登録します。状態には、なし、アドバタイズ済み、インストール済みがあります。機能がインストールされると、PublishFeatures アクションによって機能とコンポーネントの関係がレジストリに書き込まれます。
PublishProduct	標準アクション	製品がアドバタイズされている場合、その製品をパブリッシュします。
ScheduleReboot	標準アクション	ScheduleReboot アクションをアクション シーケンスに挿入すると、インストールの最後に、システム再起動を促すメッセージがエンドユーザーに表示されます。ScheduleReboot アクションは通常、シーケンスの終わりに配置されます。
InstallFinalize	標準アクション	このアクションは、最後に処理される手順です。
RemoveExistingProducts	標準アクション	このアクションは、Upgrade テーブルにリストされたすべての製品コードをループにして、これらの製品を削除します。

テーブル 20 • [インストール] シーケンスでアクションとダイアログを実行する (続き.)

アクションまたはダイアログの名前	イベントの種類	説明
ISSetupFilesCleanup	カスタム アクション	このカスタム アクションは、[サポート ファイル] ビューでファイルを追加したとき表示されます。詳細については、「サポート ファイルを使用する」を参照してください。

[アドバタイズ] シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アドバタイズ] シーケンスは、MsiExec.exe の /j コマンドライン オプションを使ってエンドユーザーがインストール プログラムを起動したときに実行されるアクションのリストです。このシーケンスにビルトインされているアクションの説明を、以下の表に示します。

ユーザー インターフェイス

入力規則 ICE78 では、[アドバタイズ]-[ユーザー インターフェイス] シーケンスが空になっている必要があるとしています。

実行

[実行] シーケンスには、正しく機能するためにユーザー インターフェイスを使用しないすべてのアクションが含まれます。これらのアクションには、ファイル転送、コンポーネントおよび機能のパブリッシュ、COM サーバーの登録などがあります。

標準のアクションに関する技術的な詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」を参照してください。

テーブル 21 • [アドバタイズ] シーケンスでアクションを実行する

アクション名	イベントの種類	説明
CostInitialize	標準アクション	インストールの現在の設定に必要なディスク容量を判断する最初のステップです。
CostFinalize	標準アクション	現在の設定でインストールに必要な合計ディスク容量を判断します。
InstallValidate	標準アクション	現在の構成に必要なディスク容量があるかどうかを判断します。

テーブル 21 • [アドバタイズ] シーケンスでアクションを実行する (続き.)

アクション名	イベントの種類	説明
InstallInitialize	標準アクション	エンドユーザーのシステムに変更を加えるアクションの最初にマークを付けます。
CreateShortcuts	標準アクション	[ショートカット] ビューまたは [セットアップのデザイン] ビューで指定したショートカットを作成します。
RegisterClassInfo	標準アクション	[COM 登録] 詳細設定で指定したすべての COM クラス情報を登録します。
RegisterExtensionInfo	標準アクション	[ファイルの種類] 詳細設定で指定したすべての拡張子を登録します。
RegisterProgIdInfo	標準アクション	コンポーネントがアドバタイズされている場合、COM 登録詳細設定で定義したすべての ProgID を登録します。
RegisterMIMEInfo	標準アクション	コンポーネントがアドバタイズされている場合、[ファイルの種類] 詳細設定で定義したすべての MIME タイプを登録します。
RegisterTypeLibraries	標準アクション	COM 登録詳細設定またはコンポーネントウィザードを使用して、インストールに作成したタイプライブラリを登録します。
PublishComponents	標準アクション	アドバタイズされた機能に関連するコンポーネントすべてをパブリッシュします。
MsiPublishAssemblies	標準アクション	共通言語ランタイムアセンブリと Win32 アセンブリのアドバタイズを管理します。このアクションは グローバルアセンブリキャッシュにアドバタイズまたはインストールされている機能を持つアセンブリを決定し、特定のアプリケーションから孤立した場所にアドバタイズまたはインストールされている親コンポーネントを持つアセンブリを判別するように MsiAssembly テーブルに問い合わせます。
PublishFeatures	標準アクション	各機能のインストール状態を登録します。状態には、なし、アドバタイズ済み、インストール済みがあります。機能がインストールされると、このアクションによって機能とコンポーネントの関係がレジストリに書き込まれます。
PublishProduct	標準アクション	製品がアドバタイズされている場合、その製品をパブリッシュします。
InstallFinalize	標準アクション	インストールまたはアンインストールの最終段階。

[管理] シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[管理] シーケンスは、/a コマンドライン オプションを使ってユーザーがセットアップ プログラムを起動したときに実行されるアクションのリストです。[管理] シーケンスに組み込まれているアクションおよびダイアログを、以下の表に示します。

ユーザー インターフェイス

[ユーザー インターフェイス] シーケンスには、完全なユーザー インターフェイスをサポートするのに必要なアクションおよびダイアログがすべて含まれています。このシーケンスは、インストールをサイレント モードで実行する場合にはスキップされます。

プロジェクト: *以下に説明するユーザー インターフェイスアクションは基本の MSI プロジェクト用のものです。InstallScript MSI プロジェクト については、InstallScript コードがインストールのユーザー インターフェイスを実行します。*

標準のアクションに関する技術的な詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」を参照してください。

テーブル 22 · [管理] シーケンスのユーザー インターフェイス アクション

アクション名	イベントの種類	説明
SetupCompleteError	ダイアログ	このダイアログは、セットアップが致命的なエラーのために終了した場合に表示されます。
SetupInterrupted	ダイアログ	このダイアログは、エンドユーザーによって終了されたインストールの終わりで表示されます。
SetupCompleteSuccess	ダイアログ	このダイアログは、インストールが正常に終了した場合に表示されます。
SetupInitialization	ダイアログ	このダイアログは、インストールの開始準備中に表示されます。このダイアログのデフォルトのテキストは、「インストールの準備をしています...」です。
CostInitialize	標準アクション	このアクションは、インストールの現在の設定に必要なディスク容量を判断する最初のステップです。

テーブル 22 • [管理] シーケンスのユーザー インターフェイス アクション (続き.)

アクション名	イベントの種類	説明
FileCost	標準アクション	インストールの現在の設定に必要なディスク容量を判断します。新規バージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。
CostFinalize	標準アクション	CostFinalize アクションは、現在の設定でインストールに必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。
AdminWelcome	ダイアログ	これは [管理] シーケンス中に最初に表示されるダイアログです。
SetupProgress	ダイアログ	このダイアログは、インストールの進行状況を表示します。
ExecuteAction	標準アクション	このアクションは [管理]–[実行] シーケンスを実行します。

実行

[実行] シーケンスには、正しく機能するためにユーザー インターフェイスを使用しないすべてのアクションが含まれます。これらのアクションには、ファイル転送、コンポーネントおよび機能のパブリッシュ、COM サーバーの登録などがあります。

テーブル 23 • [管理] シーケンスでアクションとダイアログを実行する

アクション名	イベントの種類	説明
CostInitialize	標準アクション	このアクションは、現在のインストール構成に必要なディスク容量を判断する最初のステップです。
FileCost	標準アクション	現在のインストール構成に必要なディスク容量を判断します。新規バージョンで上書きするファイルがあるかどうか、上書きがディスク容量にどのように影響するかを確認します。
CostFinalize	標準アクション	CostFinalize アクションは、現在の設定でインストールに必要な合計ディスク容量を判断します。また、すべてのターゲットディレクトリが書き込み可能であることを確認します。
InstallValidate	標準アクション	InstallValidate アクションは、現在のセットアップ設定に必要なディスク容量があるかどうかを判断します。
InstallInitialize	標準アクション	InstallInitialize アクションは、エンド ユーザーのシステムに変更を加えるアクションの開始を示します。

テーブル 23 • [管理] シーケンスでアクションとダイアログを実行する (続き.)

アクション名	イベントの種類	説明
InstallAdminPackage	標準アクション	このアクションは、インストール データベースを管理インストールの場所にコピーします。
InstallFiles	標準アクション	InstallFiles アクションでは、機能をインストールする場合、その機能に属するファイルすべてをターゲットマシンにコピーします。ローカルにインストールされるコンポーネントに関連したファイルのみが、ターゲットマシンにコピーされます。
InstallFinalize	標準アクション	インストールまたはアンインストールの最後のステップです。

[ユーザー インターフェイス] シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ユーザー インターフェイスには、デフォルトのユーザー インターフェイスに必要なすべてのアクションおよびダイアログが含まれています。これらのアクションやダイアログは、ターゲット システムを変更しません。一般的に、あとで実行するインストール作業のために、システム環境やエンドユーザーに関する情報を収集します。

デフォルトで、[管理] と [インストール] シーケンスは、それぞれ [ユーザー インターフェース] シーケンスを含みます。

プロジェクト: *InstallScript MSI プロジェクト用のユーザー インターフェイス シーケンスでは、InstallScript のカスタム アクションはサポートされていません。*

[実行] シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[実行] シーケンスは、インストールがサイレント モードで実行されていない限り、[ユーザー インターフェイス] シーケンスの後で実行されます。[実行] シーケンスには、ターゲット システムを変更するすべての標準アクションとカスタム アクションが含まれています。たとえば、ファイル転送は [実行] シーケンス中にが処理されます。通常、このシーケンスはターゲット コンピューターに変更が加えられた際に行われるインストール作業の一部として考えることができます。

[インストール]、[アドバタイズ]、および [管理] の各シーケンスのアクションは、[実行] シーケンス内でそれぞれ異なったアクションを起動します。3 種類のシーケンスの目的がそれぞれ異なるために、起動するアクションも異なります。たとえば、[インストール] シーケンスはターゲット コンピューターに製品をインストールします。一方、[アドバタイズ] シーケンスは、ターゲット システム上にある製品のアドバタイズのショートカット、ファイルの種類の情報および COM サーバー データなどをインストールしますが、エンド ユーザーがショートカットを選択し、登録された文書を開くか、アドバタイズされた COM サーバーを起動するまで製品のファイルを転送しません。シーケンス名から推測できるように、これは製品のアドバタイズのために使用します。

アクションをシーケンスに挿入する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

シーケンスはインストールと関連付けられたすべてのアクションが実行されるタイミングを指示します。アクションは、任意のシーケンスで追加 、削除 、または順序変更を行うことができます。たとえば、インストールの一部として Readme ファイルを表示するために、インストールに Readme ファイルを起動するカスタム アクションを追加することができます。このアクションはシーケンスに挿入する必要があります。

アクションのシーケンスへの挿入は、シーケンスのアクションがいつ実行されるかによって決まります。多くのアクションは、そのカスタム アクションが機能する前に実行されている他のアクションに依存しています。たとえば、インストールに組み込む実行可能ファイルを起動する場合、この実行可能ファイルに依存するアクションを実行できるのは、シーケンス処理におけるファイルのインストールの終了後になります。

メモ: *InstallShield では、シーケンスに対する検証はされません。アクションが正常に機能しないシーケンスにある場合でも、インストールが実行されるまでエラーは発生しません。*

アクションの挿入

タスク ***アクションをシーケンスに挿入するには以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、新しいアクションの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開き、シーケンスに追加することができるアクションおよびダイアログが一覧表示されます。
3. ダイアログ ボックスの上にあるリストから、挿入するアクションの種類を選択します。

4. アクションの一覧があるボックスで、挿入するアクションの種類を選択します。
5. **[OK]** をクリックします。

InstallShield では、[カスタム アクション] エクスプローラーからカスタム アクションを [シーケンス] エクスプローラーのシーケンスにドラッグ アンド ドロップすることもできます。

タスク ***ドラッグ アンド ドロップで、カスタム アクションをシーケンスに挿入するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[カスタム アクション]** エクスプローラーからカスタム アクションを、**[シーケンス]** エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。

メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が* CustomAction *テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに挿入することはできません。*

アクションの挿入場所の決定

インストールに含むことができるカスタム アクションには、標準または Windows Installer 対応の DLL 関数の呼び出し、実行可能ファイルの起動、スクリプト (Jscript、VBScript、または InstallScript) の実行、プロパティまたはディレクトリの設定、および 2 番目の .msi インストールの起動など、多くのカテゴリがあります。カスタム アクションの構成によって、これらのタイプをさまざまな場面で呼び出すことができます。詳細は、次を参照してください:

- DLL 内の関数を呼び出すカスタム アクションをシーケンスする
- .exe ファイルを起動するカスタム アクションをシーケンスする
- スクリプトを呼び出すカスタム アクションをシーケンスする
- プロパティまたはディレクトリのプロパティを設定するカスタム アクションのシーケンス
- 2 番目の .msi パッケージを起動するカスタム アクションをシーケンスする

インクルードしたマージ モジュールからのカスタム アクションの検討

ダイレクト エディターで ModuleInstallExecuteSequence テーブルを変更して、マージ モジュール内のカスタム アクションの起動を制御することができます。インストール プロジェクトにマージ モジュールを追加する際、マージ モジュールの中に含められたすべてのカスタム アクションおよびダイアログは [カスタム アクションとシーケンス] ビューを通してインストールのシーケンスに挿入することができます。

カスタム アクションを別のシーケンスへコピーする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

- *MSI データベース*
- *トランスフォーム*

InstallShield では、ドラッグアンドドロップ操作で、カスタム アクションをあるシーケンスから別のシーケンスへコピーすることができます。

タスク ***カスタム アクションをあるシーケンスから別のシーケンスへコピーするには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、コピーするアクションを検索します。
3. カスタム アクションをあるシーケンスから別のシーケンスへドラッグしている間、CTRL を押し続けます。そのカスタム アクションの直前にくるアクションまたはダイアログの上にドロップします。

メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が* CustomAction *テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに移動またはコピーすることはできません。*

ダイアログおよび標準アクションは、ドラッグアンドドロップ操作で、異なる種類のシーケンスへ移動することはできません。

シーケンスの順序を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[カスタム アクションとシーケンス] ビューの [シーケンス] エクスプローラー内のアクションおよびダイアログは、起動されるタイミングにしたがって時系列で編成されます。各アクションとアイテムにはまた、他のアイテムのシーケンス番号に関連してシーケンス中の位置を識別する番号があります。アイテムは、小さい番号から順番に起動されます。

(基本の MSI、MSI データベース、およびトランスフォーム プロジェクトに) ダイアログを、または、カスタム アクションをプロジェクトに追加するとき、それをシーケンスの適切な位置に追加して、起動されるタイミングを指定することができます。

タスク ***シーケンス内で実行するアクションとダイアログの順序を変更するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. 以下のいずれかを実行します。

- 新しいカスタム アクションをシーケンスするには、**[カスタム アクション]** エクスプローラーからそれを、**[シーケンス]** エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。
- アクションまたはダイアログを、シーケンス内の異なる位置へ(または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。

メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が* CustomAction *テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに移動またはコピーすることはできません。*

ダイアログおよび標準アクションは、ドラッグアンドドロップ操作で、異なる種類のシーケンスへ移動することはできません。

シーケンスの追加後にカスタム アクションを変更しても、それをシーケンスに割り当て直す必要はありません。シーケンス内に置かれたアクションは、実際のアクションに対するポインターで、[カスタム アクションとシーケンス] ビューでアクションに変更を加えると、動的に更新されます。

ヒント: *また、以下の任意の方法を用いて、**[シーケンス]** エクスプローラー内のアクションおよびダイアログを移動することができます。*

- *アクションまたはダイアログを右クリックして、**[上へ移動]** または **[下へ移動]** をクリックします。*
- *アクションまたはダイアログをクリックしたあと、CTRL+シフト+上矢印、または、CTRL+シフト+下矢印を押します。*
- *"シーケンス番号" 設定を編集して、アクションまたはダイアログの位置を変更します。"シーケンス番号" 設定は、アクションまたはダイアログをクリックしたときに、右のグリッドに表示されます。アクションまたはダイアログのシーケンス内における順序を先にする場合は、シーケンス番号を小さくします。反対に、シーケンス内における順序を後にする場合は、シーケンス番号を大きくします。*

アクションをシーケンスから削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***アクションをシーケンスから削除するには、以下の手順を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **シーケンス** エクスプローラーで、削除するアクションを含むシーケンスを展開します。
3. アクションを右クリックして、**削除**をクリックします。

アクションが削除されるのは指定したシーケンスからだけで、すべてのシーケンスまたはプロジェクトからは削除されません。インストール プロジェクトからのアクションの完全な削除は、[カスタム アクションとシーケンス] ビューの [カスタム アクション] エクスプローラーで行うことができます。

ロールバック カスタム アクションのシーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

ターゲット システムに直接変更を加えるインストール プロジェクトの中のカスタム アクションには、ロールバック同等のカスタム アクションが必要です。ロールバック カスタム アクションは、ロールバックが実行されたときにこれらの変更を元に戻すことができます。たとえば、ターゲット システムからファイルを削除するカスタム アクションがあるのに、削除したファイルを復元するロールバックカスタム アクションが存在しないと、ロールバックが終了しても、コンピューターが不安定な状態のままになることがあります。ロールバックカスタム アクションは、事前定義かカスタム アクションと同様に動作します。つまり、シーケンスで最初に発生したときには起動しません。ロールバックスクリプトに書き込まれ、ロールバックが起きたときにのみ起動します。

ロールバック カスタム アクションのシーケンスへの挿入時には、以下の事項に留意してください。

- ロールバックは、[実行] シーケンス中にのみ実行でき、[ユーザー インターフェイス] シーケンス中には実行できません。したがって、ロールバック アクションは、[実行] シーケンスで、InstallInitialize の後、および InstallFinalize の前に配置する必要があります。
- ロールバック カスタム アクションは、ロールバックするアクションの前にシーケンスする必要があります。その他のインスタンスでは、Windows Installer サービスはシーケンスを上から下へ実行するため、シーケンス番号が小さいほど、そのアクションが早く起動します。ロールバックスクリプトでは、サービスは反対方向に実行されます。したがって、必要なときにロールバック アクションを起動するには、ロールバックを行うアクションの前にそれをシーケンスします。

.exe ファイルを起動するカスタム アクションをシーケンスする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクションとして実行可能ファイルを起動することには、シーケンス中のどこにこのアクションが挿入されるかに関して、DLL ファイル関数の呼び出しよりもやや扱いにくい問題があります。インストールのどこで .exe ファイルを起動するのか、またどのシーケンス中でカスタム アクションを起動したいかなどについて考慮する必要があります。

.msi パッケージに格納されている実行可能ファイル

.msi パッケージに保存された .exe ファイルを起動する場合、カスタム アクションをシーケンスのどこにでも配置することができます。

テーブル 24 • .msi パッケージ内の .exe ファイルを起動するアクションの設定

アクションの種類	場所	スケジュール	Sequence
実行可能ファイルを起動する	.msi パッケージに保存	なし	シーケンス中の任意の場所

インストール済み実行可能ファイル

インストールの一部として .exe ファイルがインストールされる場合、選択したスケジューリングのプロパティ（遅延実行かまたは即時実行）に応じて以下の 2 組の規則のうちの 1 つに従う必要があります。すぐに実行の場合は、アクションを正常に動作させるために InstallFinalize アクションの後に置く必要があります。遅延実行の場合は、InstallFiles アクションと InstallFinalize アクションの間に挿入する必要があります。

テーブル 25 • インストールされた .exe ファイルを起動するアクションの設定

アクションの種類	場所	スケジュール	Sequence
実行可能ファイルを起動する	インストール済み	即時実行	InstallFinalize の後
実行可能ファイルを起動する	インストール済み	遅延実行	InstallFiles の後、InstallFinalize の前

ローカル実行可能ファイル

実行したい .exe ファイルが既にシステム上にある場合、アクションを CostFinalize アクションの呼び出しの後に挿入する必要があります。以上の事項を以下の表に整理して示します。

テーブル 26 • 既にインストールされた .exe ファイルを起動するアクションの設定

アクションの種類	場所	スケジュール	Sequence
実行可能ファイルを起動する	ローカル	なし	CostFinalize の後

DLL 内の関数を呼び出すカスタム アクションをシーケンスする

DLL ファイル内の関数を呼び出すカスタム アクションをシーケンスする場合、DLL ファイルの参照方法として、インストール時に DLL ファイルが配置されている場所に応じて 3 つの方法があります:

- DLL ファイルは .msi ファイルにストリーム入力できますが、インストールはされません。
- DLL ファイルはターゲット システムのパスに置くことができます。(標準の DLL ファイルに適用されますが、Windows Installer DLL ファイルLには適用されません)
- または、DLL ファイルを残りのインストール ファイルと一緒にインストールできます。

その他の考慮すべき事項として、スケジューリングがあります。2 つの主なスケジューリング方法は、即時実行と遅延実行です。即時実行は、Windows Installer が .msi ファイルを処理するときにカスタム アクションを実行する方法です。遅延実行は、アクションをキューに挿入し、スクリプトに書かれた順番に実行するようにインストーラーに指示する方法です。

.msi パッケージに格納されている DLL ファイル

.msi パッケージに保存された DLL ファイル から関数を呼び出す場合、カスタム アクションをシーケンスの任意の場所に配置することができます。

テーブル 27 ・ .msi パッケージに格納されている DLL ファイル内の関数を呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
DLL 関数	.msi パッケージに保存	なし	任意の場所

システムのパスにある DLL ファイル

ターゲット システムに存在する DLL ファイルから関数を呼び出す場合も、カスタム アクションを一連のアクションの任意の場所に置くことができます。

テーブル 28 ・ システムのパスで検索された DLL ファイル内の関数を呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
DLL 関数	ターゲット システム上	なし	任意の場所

インストール済み DLL ファイル

インストール中にターゲットコンピューターにインストールされるファイルから DLLI 関数を呼び出し、それを即時実行で実行するようにスケジューリングした場合は、そのアクションは InstallFinalize アクションの後にのみ配置できます。遅延実行中にアクションを起動したい場合は、そのアクションを InstallFiles アクションと InstallFinalize アクションの間に配置します。次の表はこの点を示したものです。

テーブル 29 ・ 既にインストールされた DLL ファイル内の関数を呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
DLL 関数	インストール済み	即時実行	InstallFinalize の後
DLL 関数	インストール済み	遅延実行	InstallFinalize の前、InstallFiles の後

スクリプトを呼び出すカスタム アクションをシーケンスする

プロジェクト: *VBScript と JScript の情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

カスタム アクションの中でスクリプトを使用すると、事実上どんなタスクを実行することも可能です。たとえば、外部アプリケーションを起動したり、レジストリ項目を作成したりできます。

VBScript と JScript コード

製品と一緒にインストール

インストール中にスクリプトがインストールされる場合、選択したスケジューリングのプロパティ (遅延実行かまたはすぐに実行) に応じて以下の 2 組の規則のうちの 1 つに従う必要があります。すぐに実行の場合は、アクションを正常に動作させるために InstallFinalize アクションの後に置く必要がありあます。遅延実行の場合は、InstallFiles アクションと InstallFinalize アクションの間に挿入する必要があります。

テーブル 30 ・ 製品と共にインストールされた VBScrilpt または JScript ファイルを呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
VBScript または JScript コードの実行	インストール済み	即時実行	InstallFinalize の後
VBScript または JScript コードの実行	インストール済み	遅延実行	InstallFiles の後、InstallFinalize の前

Binary テーブルに保存する

呼び出すスクリプト ファイルが Binary テーブルに保存されている場合、一連のアクションのどこにでもそのアクションを置くことができます。

テーブル 31 ・ Binary テーブルに格納されている VBScrilpt または JScript ファイルを呼び出すアクションの設定

アクションの種類	場所	スケジュール	Sequence
VBScript/JScript コードの実行	一時	なし	シーケンス中の任意の場所

ターゲット システム上に既存

呼び出すスクリプト ファイルがシステム上に既にある場合は、CostFinalize アクションの呼び出しの後にそのアクションを挿入する必要があります。

テーブル 32 ・

アクションの種類	場所	スケジュール	Sequence
VBScript/JScript コードの実行	ローカル	なし	CostFinalize の後

カスタム アクション中に保存されたディレクトリ

スクリプトをカスタム アクション中に直接置いた場合は、一連のアクション中のどの場所にもそのアクションを置くことができます。

InstallScript コード

InstallScript カスタム アクションは、InstallScript アクションのソースファイルが常に .msi パッケージにストリーム入力されるため、JScript または VBScript カスタム アクションと異なります。したがって、カスタム アクションはシーケンス番号として 2 と示されるアクションの後の任意の場所に挿入できます。この制限があるのは、スクリプト エンジンがシーケンス番号 2 の間に起動するためです。したがって、スクリプト エンジンが起動する前にスクリプトを呼び出した場合、システムはそれを処理できません。

カスタム アクションに、機能関数、またはセットアップの種類ダイアログ関数が含まれる場合、CostInitialize、FileCost、CostFinalize アクションの後に、カスタム アクションを挿入する必要があります。

プロパティまたはディレクトリのプロパティを設定するカスタム アクションのシーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロパティまたはディレクトリを設定するカスタム アクションをシーケンスに配置する場合、まず、そのディレクトリまたはプロパティが必要とされる前にカスタム アクションを起動する必要があることに注意してください。たとえば、ターゲットコンピューター上にある EXE を起動したい場合、インストールで最初にそれを検索する必要があります。

実行可能ファイルが見つかった後、そのパスを Directory テーブルに入力し、その後、実行可能ファイルを起動することができます。しかし、実行可能ファイルの起動後にカスタム アクションを呼び出した場合、実行可能ファイルは見つかりません。

その他のシーケンスおよびスケジュール上の制限は次のとおりです。

- 検証規則 ICE12 によると、すべての タイプ-35 カスタム アクション (カスタム アクションウィザードで「**ディレクトリの設定**」と呼ばれる) を、シーケンスの標準 CostFinalize アクションの後に並べる必要があります。
- 同様に、ICE12 によると、Directory テーブルのプロパティの値を設定するすべてのタイプ-51 カスタム アクション (カスタム アクションウィザードで **プロパティの設定** と呼ばれる) を、標準 CostFinalize アクションの前にスケジュールする必要があります。
- プロパティを設定するすべてのカスタム アクションは、すぐに実行するようスケジューリングする必要があります。

2 番目の .msi パッケージを起動するカスタム アクションをシーケンスする

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

***重要・** Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。さらに詳しい情報は、MSDN Web サイトの「Concurrent Installations」を参照してください。*

ネスト インストールのカスタム アクション (カスタム アクション ウィザードの **[別の .msi パッケージを起動]** という名前) をシーケンスする場合、標準の CostFinalize アクションの後に、[実行] シーケンスにアクションを配置する必要があります。

また、該当する場合、インストール内に含まれる任意ファイルがメイン インストールで必要になる前に、カスタム アクションが起動するか確認してください。たとえば、2 番目の .msi ファイルの一部としてインストールされた実行可能ファイルを起動したい場合は、その実行可能ファイルが起動される前に必ずパッケージがインストールされていなければなりません。

ISSetAllUsers カスタム アクション

ISSetAllUsers がインストール パッケージのシーケンスに表示される理由

([アップグレード] ビューを使用して) インストールの Upgrade テーブルに 1 つ以上の記録を入力すると、InstallShield は [ユーザー インターフェイス] と [実行] インストール シーケンスの部分の両方に、ISSetAllUsers という .DLL カスタム アクションを挿入します。ご使用の製品がアップグレードとしてインストールされている場合、ISSetAllUsers カスタム アクションはインストールされているバージョンの ALLUSERS プロパティの値をチェックします。

ALLUSERS プロパティは [カスタマー情報] ダイアログ (基本の MSI プロジェクトの場合) に示され、初回インストール中にエンドユーザーによって **SdCustomerInformation** または **SdCustomerInformationEx** ダイアログ (InstallScirpt MSI プロジェクトの場合) に示されます。ISSetAllUsers カスタム アクションは、インストールされたバージョンの値を新規バージョンの値と比較します。値が異なると、ISSetAllUsers は新しいバージョンの ALLUSERS プロパティを、インストールされているバージョンと一致するように設定します。

アップグレードの場合、ALLUSERS プロパティはカスタム アクションを使用しなければ設定できないことに注意してください。

新しいインストールの ALLUSERS プロパティは、FindRelatedProducts アクションで正常にアップグレード インストールを行うことができるように、インストールされているバージョンのプロパティと一致する必要があります。さらに、以前のバージョンが一人の特定のユーザー用にインストールされていて、すべてのユーザー用にアップグレードがインストールされていると、インストールが結果的に破損して、正しくアンインストールされない場合があります。ISSetAllUsers は ALLUSERS プロパティをリセットしてこの問題を解決します。

ISSetAllUsers の役割

次の例で、ISSetAllUsers カスタム アクションの実際の役割を説明します。

1. My Application 1.0 を、ALLUSERSプロパティを *1* に設定してインストールします (ユーザーはインストール中にエンド ユーザー ダイアログで [**すべてのユーザー用にインストール**] を選択)。
2. My Application 2.0 がバージョン 1.0 のアップグレードとして作成され、Upgrade テーブルにバージョン 1.0 のアップグレードエントリが挿入されます。
3. エンドユーザーはターゲット システムにバージョン 2.0 をアップグレードとしてインストールします。
4. インストールの間、ISSetAllUsers は次の方法でインストールされているバージョンの ALLUSERS プロパティを調べ、新しいバージョンのプロパティと比較します。
    a. ISSetAllUsers はバージョン 2.0 の Upgrade テーブルの各エントリを繰り返し調べます。
    b. Upgrade テーブルの各アップグレード コードについて、ISSetAllUsers はターゲット システム上の関連製品を探します。
    c. Upgrade テーブルのバージョン制限と言語制限が (ステップ 4b で見つけた) インストールされている製品と一致する場合、ISSetAllUsers はインストールされているバージョンの ALLUSERSプロパティを確認します。

d. インストールされているバージョンの ALLUSERS 値が新しいバージョンと異なる場合、ISSetAllUsers は新しくインストールされたバージョンの ALLUSERS プロパティをこの値に設定し直します。

メモ: *一致する製品がターゲット システムにインストールされていない場合、ISSetAllUsers は何も実行しません。*

サポート ファイルを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

サポート ファイルは、製品のインストール処理中にのみターゲット システムで使用できるファイルです。サポート ファイルはインストールが始まるとターゲット システムの一時ディレクトリにコピーされます。サポート ファイルはインストールが完了すると削除されます。サポート ディレクトリは、ダイナミック ファイルの場所を表し、ターゲット システムごとに異なります。また、同じシステムでもインストールするたびに異なります。

[サポート ファイル] ビュー (アドバンスト UI、基本の MSI、InstallScript オブジェクト、およびスイート/アドバンスト UI プロジェクトの場合)、または [サポート ファイル/ビルボード] ビュー (InstallScript プロジェクトおよび InstallScript MSI プロジェクトの場合) で、インストール中のみターゲット システム上で使用可能なファイルを追加および削除することができます。

サポート ファイルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

タスク ***サポート ファイルをインストール プロジェクトに追加するには、以下の手順に従います:***

1. 基本の MSI、InstallScript オブジェクト、およびスイート プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。

2. オプションで、ファイルに 1 つ以上のサブフォルダーを作成することもできます。その場合、**[サポート ファイル]** エクスプローラーで、[言語非依存] ノードまたは言語固有ノードのひとつを右クリックしてから、**[新しいフォルダー]** をクリックします。新しいフォルダーが追加されます。フォルダーに使用する名前を入力します。

3. **[サポート ファイル]** エクスプローラーで、追加するサポート ファイルが必要なアイテムをクリックします。

4. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

5. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。

6. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

ヒント: *Windows エクスプローラーからファイルをドラッグして、[ファイル] ペインにドロップすることもできます。*

ライセンス ファイルの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

[サポート ファイル/ビルボード] ビューで、使用許諾契約書を含むテキスト ファイルを追加できます。

タスク ***ライセンスファイルを追加するには、以下の手順を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。

2. オプションで、ライセンス ファイルに 1 つ以上のサブフォルダーを作成することもできます。その場合、**[サポート ファイル]** エクスプローラーで、[言語非依存] ノードまたは言語固有ノードのひとつを右クリックしてから、**[新しいフォルダー]** をクリックします。新しいフォルダーが追加されます。フォルダーに使用する名前を入力します。

3. **[サポート ファイル]** エクスプローラーで、追加するライセンス ファイルを含めるアイテムをクリックします。

4. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
6. スクリプトの **SdLicense*** 関数の 1 つにある szLicenseFile パラメーターでファイルを呼び出します。

サポート ファイルの並べ替え

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

[サポート ファイル] ビュー (アドバンスト UI、基本の MSI およびスイート/アドバンスト UI プロジェクトの場合) または [サポート ファイル/ビルボード] ビュー (InstallScript プロジェクトと InstallScript MSI プロジェクトの場合) の [ファイル] ペインで、ファイルをソートする列の見出しをクリックしてファイルをソートすることができます。任意の列でソートすることができます。

Disk1 フォルダーへファイルとフォルダーを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Disk1 ノードでは、インストールメディアの Disk1 に入れるファイルとフォルダーを指定することができます。これらのファイルとフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストール メディアにリンクさせることができます。

たとえば、アプリケーションと共に含める大きい再配布可能ファイルがあるとします。エンドユーザーが再配布可能ファイルにアクセスできるようにしたくても、アプリケーションのインストールには含めたくない場合があります。このような場合、このファイルを Disk1 フォルダーに入れることができます。

タスク ***Disk1 フォルダーにファイルまたはフォルダーを追加する場合、次の操作を実行します。***

1. 基本の MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。

2. **[サポート ファイル]** エクスプローラーで、**Disk1** をクリックします。
3. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
5. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

Disk1 フォルダーからファイルまたはフォルダーを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

タスク ***Disk1 フォルダーからファイルまたはフォルダーを削除する場合、以下の手順にしたがいます:***

1. 基本の MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: [動作とロジック] の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**Disk1** をクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

最後の Disk フォルダーにファイルとフォルダーを追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

最後の Disk ノードでは、インストールメディアの最後のディスクに入れるファイルとフォルダーを指定することができます。これらのファイルまたはフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストール メディアにリンクさせることができます。

たとえば、アプリケーションと共に含める大きい再配布可能ファイルがあるとします。エンドユーザーが再配布可能ファイルにアクセスできるようにしたくても、アプリケーションのインストールには含めたくない場合があります。このような場合、このファイルを 最後のディスク フォルダーに入れることができます。

タスク ***最後のディスクイメージフォルダーにファイルまたはフォルダーを追加する場合、次の操作を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**最後の Disk** をクリックします。
3. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
5. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

最後の Disk フォルダーからファイルとフォルダーを削除する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

タスク ***最後のディスクイメージフォルダーからファイルまたはフォルダーを削除する場合、次の操作を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**最後の Disk** をクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

その他の Disk フォルダーにファイルとフォルダーを追加する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

[その他] ノードを使うと、最初または最後のディスクの他にインストールメディアのディスクに入れるファイルとフォルダーを指定できます。これらのファイルまたはフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストールメディアにリンクさせることができます。

タスク ***ディスクイメージフォルダーにファイルまたはフォルダーを追加する場合、次の操作を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**[その他]** をクリックします。
3. **[ファイル]** ペインで右クリックして **[ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

4. 含めるファイルを参照します。複数のファイルを選択するには、ファイルをクリックしながら CTRL キーを押します。
5. **[OK]** をクリックします。

InstallShield は、ファイルを [ファイル] ペインに追加します。

ヒント: *ディスクを指定するには、リリース ウィザードを実行して [一般オプション] パネルで [その他のディスクファイル] ボタンをクリックします。*

他の Disk フォルダーからファイルとフォルダーを削除する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

タスク ***ディスクイメージフォルダーからファイルまたはフォルダーを削除する場合、次の操作を実行します。***

1. ビュー リストの [動作とロジック] の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、**[その他]** をクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

サポート ファイルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

タスク ***プロジェクトからサポート ファイルを削除するには、以下の手順に従います:***

1. 基本の MSI、InstallScript オブジェクト、およびスイート プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル]** をクリックします。

   InstallScript、および InstallScript MSI プロジェクトの場合: **[動作とロジック]** の下にある [ビューリスト] で、**[サポート ファイル/ビルボード]** をクリックします。
2. **[サポート ファイル]** エクスプローラーで、削除するサポート ファイルを含むアイテムをクリックします。
3. **[ファイル]** ペインでは、ファイルまたはフォルダーを右クリックして、**[削除]** をクリックします。

エンドユーザー インターフェイスを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ドキュメントのこのセクションでは、エンドユーザー インターフェイスの異なる要素を定義することができる InstallShield の異なる機能について説明します。「エンドユーザー インターフェイスの定義」セクションの一部では、インストールのエンドユーザー インターフェイスのダイアログの作成および使用方法についても説明しています。他では、インストールのローカライズを行うための文字列などのトピックを扱っています。

このセクションには、次のトピックが含まれています:

- ダイアログの使い方
- ウィザード インターフェイスを使って作業する
- エンドユーザー インターフェイスのローカライズ
- ビルボードを表示する
- InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する
- 実行時にリスト ボックスへデータを挿入する
- ファイル参照ダイアログを表示する
- InstallScript インストールでネットワーク参照ダイアログを表示する

ダイアログの使い方

このセクションでは、InstallShield の異なるプロジェクトの種類ごとに、エンドユーザー ダイアログの使い方について、基本的な情報からより高度な情報まで部分的に取り上げられています。

プロジェクト: *基本の MSI インストールでは通常、Windows Installer がランタイム ユーザー インターフェイスを制御します。InstallScript とInstallScript MSI インストールでは通常、InstallScript エンジンがランタイム エンド ユーザー インターフェイスを制御します。したがって、ドキュメントのこのセクションの一部は基本の MSI インストールに、一部は InstallScript とInstallScript MSI インストールに、また一部は 3 つのすべてのプロジェクト タイプに適用します。*

プロジェクトでダイアログを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ダイアログは、インストールのユーザー インターフェイスを提供します。エンドユーザーからの情報を要求し、インストール プロセスの進行状況についてフィードバックを提供します。

プロジェクトにおけるダイアログの作業方法は、作成するインストールの種類によって異なります。InstallScript または InstallScript MSI プロジェクトにダイアログを追加する手順は、基本の MSI プロジェクトにダイアログを追加する手順と異なります。

共通の操作

一部のダイアログ操作は、基本の MSI、InstallScript および InstallScript MSI プロジェクトで同一です。

- ダイアログ ウィザードを使用して新しいダイアログを作成する
- 他のプロジェクトで使用するダイアログを .isd ファイルへエクスポートする
- .isd ファイルからダイアログをインポートする
- すべてのダイアログを .rc ファイルへエクスポートする
- リソース .dll ファイルからダイアログをインポートする
- 他のプロジェクトへダイアログをエクスポートする

ダイアログ ウィザードを使用して新しいダイアログを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield は、ウィザード パネルに従って新しいダイアログをインストールに追加し構成できるダイアログ ウィザードを提供しています。

既存のユーザー アカウントを作成または設定する機能を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

多くのサーバー アプリケーションは、インストール中にユーザー アカウント情報を必要とします。サーバー アプリケーションが別のユーザーに制限されているリソースへアクセスできるようにするために、ユーザー アカウントを使えるようにしておくことが必要な場合が多々あります。InstallShield では、インストールに適切なランタイム ダイアログを追加して、インストール中に既存の Windows ユーザー アカウントの設定、または、新規アカウントの作成を行うことができます。

テーブル 1 ・ ユーザー アカウント サポートの追加 — プロジェクト別の説明

プロジェクトの種類	追加情報
基本の MSI	基本の MSI プロジェクトでは、LogonInformation ダイアログで入力したユーザー名、パスワード、およびグループ情報は、次のプロパティに格納されます。 • IS_NET_API_LOGON_USERNAME • IS_NET_API_LOGON_GROUP • IS_NET_API_LOGON_PASSWORD
InstallScript	InstallScript プロジェクトでは、LogonInformation ダイアログでエンドユーザーが入力したユーザー名、パスワード、およびグループ情報は、次のグローバル変数に格納されます。 • IFX_NETAPI_USER_ACCOUNT • IFX_NETAPI_PASSWORD • IFX_NETAPI_GROUP
InstallScript MSI	InstallScript MSI プロジェクトでは、指定されたユーザー アカウントおよびパスワードは、次のグローバル変数およびプロパティに格納されます。 • IFX_NETAPI_USER_ACCOUNT • IFX_NETAPI_GROUP • IFX_NETAPI_PASSWORD • IS_NET_API_LOGON_USERNAME • IS_NET_API_LOGON_GROUP • IS_NET_API_LOGON_PASSWORD

LogonInformation ダイアログのサポートを基本の MSI プロジェクトへ追加する

タスク

LogonInformation ダイアログを基本の MSI プロジェクトへ追加するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** を選択します。**新規ダイアログ ウィザード**が開きます。
3. ウィザードの各パネルを完成します。**[ダイアログ テンプレート]** パネルで、**[ログオン情報パネル、および関連する子ダイアログ]** を選択します。

新規ダイアログ ウィザードの各パネルを完了した後、インストール プロジェクトをビルドおよび実行します。LoginInformationDialog ダイアログが実行されたダイアログ シーケンスで、適切なダイアログが表示されます。

InstallScript プロジェクトまたは InstallScript MSI プロジェクトに LogonInformation ダイアログのサポートを追加する

タスク

InstallScript プロジェクトまたは InstallScript MSI プロジェクトに LogonInformation ダイアログのサポートを追加するには、次の手順を実行します。

1. LogonInformation ダイアログ セットを挿入する InstallScript コードの場所に移動します。通常、**OnFirstUIBefore** 内に追加します。例:

   ```
   Dlg_SdLogon:
   nResult = SdLogonUserInformation(szTitle, szMsg, szAccount, szPassword);
   if (nResult = BACK) goto Dlg_SdWelcome;
   ```

2. InstallScript MSI プロジェクトでは、次の関数を追加して Windows Installer のプロパティが InstallScript のグローバル変数と同じ値に設定されるようにします。**SdLogonUserInformation** を呼び出した後に追加することができます。

   ```
   OnLogonUserSetMsiProperties();
   ```

3. **[ビルド]** メニューで **[設定]** をクリックします。**[設定]** ダイアログ ボックスが開きます。
4. **[コンパイル/リンク]** タブをクリックします。
5. **[ライブラリ (.obl)]** ボックスに、新しいライブラリ ファイル (*.obl) の名前を入力します:

   ```
   NetApiRT.obl
   ```

メモ: *新しいライブラリ ファイル名は、*isrt.obl *ファイル名の前に追加します。*

InstallScript コードを追加した後、インストールをビルド並びに実行します。スクリプトが実行された時、適切なダイアログが表示されます。

基本の MSI、InstallScript および InstallScript MSI プロジェクトにおける LogonInformation ダイアログの制限

エンド ユーザーが LogonInformation ダイアログを使用しようとしたときに、ドメインの空白リストが表示されるか、「サーバーが見つかりません」エラーが表示されます。次の状況下で、この問題が発生します:

- ターゲットシステムがドメイン上に存在しない。
- ファイアーウォールでコンピュータ ブラウザ サービスが例外として構成されていない。
- ターゲット システム上で NetBIOS over TCP/IP が有効化されていない。
- ネットワーク上でブロードキャスト トラフィックが無効である。

他のプロジェクトで使用するダイアログを .isd ファイルへエクスポートする

エンドユーザーダイアログを別のプロジェクトで再利用するには、ダイアログを .isd ファイルとしてエクスポートしてから、必要に応じて別のプロジェクトにインポートすることができます。.isd ファイルは、ダイアログの動作とレイアウト情報を含みます。

タスク ***.isd ファイルへのダイアログのエクスポートは、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーでダイアログを右クリックしてから、**[ダイアログ ファイルへエクスポート]** を選択します。**[名前を付けて保存]** ダイアログ ボックスが開きます。
3. .isd ファイルを保存するフォルダーを参照します。必要であればファイル名を変更できます。
4. **[OK]** をクリックします。

これで .isd ファイルを使ってダイアログを別のインストール プロジェクトにインポートすることができます。

.isd ファイルからダイアログをインポートする

InstallShield では、ダイアログをインストール プロジェクトにインポートすることができます。ダイアログのファイル (.isd ファイル) はリポジトリに格納するか、その他の別の場所に格納することができます。

タスク ***.isd ファイルからダイアログをプロジェクトへインポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーでダイアログを右クリックしてから、**[ダイアログのインポート]** を選択します。[ダイアログのインポート] ダイアログ ボックスが開きます。
3. 以下のいずれかを実行します。
   - **[レポジトリ アイテム]**ボックスで、プロジェクトに追加するダイアログをクリックします。
   - インポートするダイアログ ボックス (.isd ファイル) のファイルがリポジトリに格納されていない場合、**[参照]** ボタンをクリックしてファイルを選択します。
4. **[OK]** をクリックします。

サポートされている各言語のコピーがユーザーのプロジェクトに追加されます。

メモ: ダイアログをインポートするときには、ダイアログによって参照される文字列識別子、プロパティ、およびアクションが新しいインストール プロジェクト内にあることを確認してください。

競合の解決

ダイアログをインポートするときには、Windows Installer によって要求されるとおりにダイアログの名前とそのコントロールが固有のものであるかどうかが確認されます。既存のダイアログと同じ名前のダイアログはインポートできません。これは InstallShield が同一のダイアログをインポートしていると認識するためです。

すでにインストールプロジェクトに既に存在するコントロールまたは文字列識別子の名前と同じ名前を使う .isd ファイルをインポートする場合は、InstallShield がこの競合の解決方法を問い合わせます。既存の値を上書きするか、または既存のものを残すためにインポートされた値をスキップするかのどちらかを選択できます。

インストール中にダイアログを表示する

プロジェクトにダイアログを追加しても、そのダイアログがインストールに表示されるわけではありません。1 つ以上のプロジェクトのシーケンスにダイアログを挿入する処理については、「基本の MSI インストール中にダイアログを表示する」を参照してください。

ダイアログ ファイル (.isd) をリポジトリにパブリッシュする

別のプロジェクトで再利用、または他のユーザーと共有したい既存のダイアログ (.isd) がある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***ダイアログをリポジトリにパブリッシュするには、次の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーでダイアログを右クリックしてから、**パブリッシュ ウィザード**を選択します。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへダイアログをインポートした後、現在のダイアログと既存のリポジトリ ダイアログとは関連性を持ちません。ダイアログを変更してリポジトリへ再パブリッシュしても、プロジェクト内のインポート済みダイアログには影響しません。但し、リポジトリからプロジェクトへダイアログを再インポートすることができます。

すべてのダイアログを .rc ファイルへエクスポートする

ダイアログのリソースを Visual Studio のようなリソースエディターを使用して編集する場合、ダイアログを .rc ファイルへエクスポートします。

タスク ***ダイアログをリソース スクリプト (.rc) ファイルにエクスポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[ダイアログをリソース スクリプトへエクスポートする]** を選択します。

プロジェクトの場所にある <プロジェクト名>.rc というファイルに、フォルトの言語のダイアログのリソースがすべてコピーされます。

各ダイアログのレイアウトをできるだけ多く保存するために可能な限りの試みがなされます。しかし、ダイアログ エディター内のいくつかの情報についてはリソース エディター内で対応するものがないため、InstallShield では、以下の例外を除くダイアログ エディター内のリソースがエクスポートされます。

テーブル 2 - .rc ファイルへエクスポートされないリソース

例外	説明
文字列エントリ	すべての文字列エントリは、デフォルトの言語の値へ解決されます。
パス	テキストファイルやビットマップなど、あらゆるリソースへのパス変数を含むパスが、後にダイアログをインポートするのを簡単にするために保存されています。
選択ツリー	Windows Installer の選択ツリーは、.rc ファイルの標準ツリー コントロールとなります。
テキスト スタイル	ダイアログ エディターが、フォント情報 (ベース テキスト スタイルやテキスト スタイル) および最大文字長について別々のプロパティを持っている間は、これらのプロパティは Windows Installer テーブル内にあるようにグループ化されています。
動作	ダイアログのレイアウトと動作の完全なコピーを含む.isd ファイルとは違い、.rc ファイルはリソースのみを保存します。

リソースエディターでのダイアログの編集

ダイアログのジオメトリは、Windows のダイアログと同じように、リソース エディター、またはメモ帳でも編集できます。

ダイアログをプロジェクトへインポートする前に、.rc ファイルを .res ファイルへコンパイルし、.dll へビルドする必要があります。

タスク ***Visual Studio を使用している場合は、以下のステップに従って MyProject.rc を MyProject.dll へビルドします。***

1. リソース コンパイラ (Rc.exe) を使用して、次のコマンドライン ステートメントで **MyProject.rc** を **MyProject** へコンパイルします。

   rc MyProject.rc

2. Incremental Linker (Link.exe) を実行し、以下のコマンドを使用して .dll をビルドします。

```
link /DLL /NOENTRY /NODEFAULTLIB /MACHINE:iX86 /OUT:MyProject.dll MyProject.res
```

リソース .dll ファイルからダイアログをインポートする

タスク ***.dll ファイルからすべてのダイアログリソースをインポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[ダイアログをリソース Dll からインポートする]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. インポートするダイアログリソースを含む .dll ファイルを参照します。
4. **[開く]** をクリックします。

各ダイアログがプロジェクトへ追加されます。名前付けの競合は、インポートしたダイアログに固有の番号を付加することによって解決されます。

インポートしたコントロールが、既に文字列エントリであるテキストを表示する場合、InstallShield では コントロールの Text プロパティに既存の文字列テーブルエントリが使用されます。

メモ: *ダイアログをインポートする際には、ダイアログによって参照されたプロパティが新規のインストール プロジェクト内にあるかどうかを確認してください。*

プロジェクトにダイアログを追加しても、そのダイアログが実行時にインストールで表示されるわけではありません。1 つ以上のインストールのシーケンスにダイアログを挿入する処理については、「基本の MSI インストール中にダイアログを表示する」を参照してください。

ダイアログは .rc ファイルへエクスポートされますが、InstallShield にそれらをインポートする前に、リソースを .dll ファイルへビルドする必要があります。詳細については、「すべてのダイアログを .rc ファイルへエクスポートする」を参照してください。

他のプロジェクトへダイアログをエクスポートする

InstallShield では、ダイアログを現在のプロジェクトからエクスポートして別の既存のプロジェクトへ挿入することができます。

タスク ***既存プロジェクトへダイアログをエクスポートするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ダイアログを右クリックしてから、**[プロジェクト ファイルへエクスポート]** を選択します。**[エクスポート情報]** ダイアログ ボックスが開きます。
3. 既存の .ism ファイルの保存場所を参照します。保存場所は、インストール プロジェクトまたはマージ モジュール プロジェクトのどちらかを使用できますが、ファイルは InstallShield の別のインスタンスでは開きません。
4. **[保存]** をクリックします。

このダイアログのコピーは、すべてのコントロール、すべてのダイアログの動作と共に特定の .ism ファイルに追加されます。ダイアログで使用されるすべての文字列テーブル、プロパティ、およびパス変数も新規プロジェクトにコピーされます。

対象の .ism ファイルに、異なるプロパティで同じ名前のダイアログがある場合は、[競合] ダイアログ ボックスが表示され、競合するダイアログを解決するオプションを選択できます。

リポジトリへダイアログをパブリッシュし、別のプロジェクトで再利用することもできます。詳細については、「リポジトリを使用してプロジェクトの要素を共有する」を参照してください。

プロジェクトからダイアログを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

基本の MSI プロジェクト

タスク

基本の MSI プロジェクトからダイアログを削除するには、以下の手順を実行します。

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ダイアログを右クリックし **[削除]** をクリックします。

メモ: *[カスタム アクションとシーケンス] ビューでダイアログを右クリックして [削除] を選択しても、プロジェクトからダイアログは削除されません。そのシーケンスから削除されるだけです。*

InstallScript および InstallScript MSI プロジェクト

タスク

InstallScript または InstallScript MSI プロジェクトからダイアログを削除するには、次の手順に従います。

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、ダイアログを右クリックし **[削除]** をクリックします。
3. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
4. インストールのエンドユーザー インターフェイスにダイアログを追加するスクリプトを削除します。

カスタム ダイアログの作成と構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

異なる種類のプロジェクトでカスタム ダイアログを作成および構成する際、次のヒントを考慮してください。

InstallScript および InstallScript MSI プロジェクト

[ダイアログ] ビューに新規ダイアログを追加するとき、ダイアログのエントリが Dialog テーブルに追加されます。このテーブルにあるデータは、ダイレクト エディターを使って直接変更することができます。

ダイレクト エディターでは、新しいダイアログ用の ISResourceId フィールドの値を指定することができます。ISResourceId 値は、このダイアログを識別するスクリプト内で利用されます。

ダイアログをプロジェクトに追加した後、このダイアログをメモリにロードし、それを実行時に表示するためには **EzDefineDialog** や **WaitOnDialog** などの関数を呼び出す必要があります。Dialog テーブルで ISResourceId 値に対応する値は、**EzDefineDialog** の 4 番目の引数に使用してカスタム ダイアログを参照します。また **EzDefineDialog** の 2 番目の引数で、ヌル文字列ではなく ISUSER を指定した方が便利です。**EzDefineDialog** 関数は次のようになります。

```
EzDefineDialog("MyCustomDialog", ISUSER, "", 12005)
```

この例では、12005 は Dialog テーブルで指定されたとおり、このカスタム ダイアログの ISResourceId です。

カスタム ダイアログ関数を呼び出し、必要に応じてカスタム ダイアログを操作します。これらの関数についての詳細は、InstallScript 言語リファレンスに記載されています。

基本の MSI プロジェクト

基本の MSI プロジェクトの [ダイアログ] ビューでカスタム ダイアログを作成する場合、Windows Installer タイプのダイアログが作成されます。InstallScript カスタム ダイアログ関数は、Windows Installer タイプのダイアログと一緒に使用することはできません。このダイアログをインストールに表示するには、[次へ] と [戻る] コントロールを使用して 2 つのダイアログの間で連続して配置する必要があります。

ダイアログ エディターで変更を元に戻す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ダイアログ エディターは、各ダイアログに対して 50 アクションまで記憶しており、最近のものから順に変更を元に戻すことができます。また、プロジェクトを保存した後でも変更を元に戻すことができます。ただし、プロジェクトを閉じて再び開くと、元に戻す操作の履歴は消去されます。

変更を元に戻す

タスク ***変更を元に戻すには、以下の手順に従います:***

- Ctrl + Z を押します。
- **[編集]** メニューで **[元に戻す]** をクリックします。
- ツールバーの **[元に戻す]** ボタンをクリックします。

元に戻した変更をやり直す

タスク ***元に戻した変更をやり直して有効にするには以下の手順に従います:***

- Ctrl+Y を押します。
- **[編集]** メニューで **[やり直す]** をクリックします。
- ツールバー上の **[やり直す]** ボタンをクリックします。

ダイアログ エディターの使用中は、[元に戻す] と [やり直す] ボタンおよびメニュー コマンドが有効です。アクションを元に戻すには、そのダイアログで実行したアクションの履歴から移動します。繰り返す場合、前に移動します。

たとえば、ボタンのサイズを変更して、Tab Index プロパティを変更した場合、[元に戻す] をクリックすると、Tab Index プロパティの直前の値が復元されます。また、[元に戻す] を 2 回クリックすると、ボタンが元のサイズに戻ります。次に、[やり直す] を 2 回クリックすると、ボタンのサイズおよびそのタブ索引の変更が再び反映されます。

メモ: *ダイアログ エディターで変更された文字列エントリを元に戻すことはできません。*

InstallScript および InstallScript MSI プロジェクトでダイアログを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield では、スクリプトで特定の関数を呼び出してエンドユーザー ダイアログを表示できます。[ダイアログ] ビューには、プロジェクトに追加するカスタム ダイアログおよび標準ダイアログがリストされています。

このセクションでは、InstallScript および InstallScript MSI プロジェクトで、ダイアログがある基本的な関数の作成および実行方法が説明されています。

InstallScript および InstallScript MSI インストールでダイアログを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript または InstallScript MSI インストールのユーザー インターフェイスのほとんどは、**OnFirstUIBefore** や **OnFirstUIAfter** などのイベント ハンドラーで定義されます。**OnFirstUIBefore** に含まれる次のサンプル InstallScript コード は、**SdWelcome** および **SdLicense2** 用のコードです:

```
Dlg_SdWelcome:
    szTitle = "";
    szMsg = "";
    nResult = SdWelcome( szTitle, szMsg );
    if (nResult = BACK) goto Dlg_Start;

Dlg_SdLicense2:
    szTitle = "";
    szOpt1 = "";
    szOpt2 = "";
    szLicenseFile = SUPPORTDIR ^ "License.rtf";
    nResult = SdLicense2Ex( szTitle, szOpt1, szOpt2, szLicenseFile, bLicenseAccepted, TRUE );
    if (nResult = BACK) then
        goto Dlg_SdWelcome;
    else
        bLicenseAccepted = TRUE;
    endif;
```

各ダイアログ関数は、エンド ユーザーがどのボタンをクリックしてダイアログを終了したかを示す整数を返します。エンド ユーザーがダイアログでクリックした [戻る] ボタンを処理するために、ダイアログの直後には、その戻り値を定数 BACK と比較する if ステートメントが、前のダイアログの直前にジャンプする goto ステートメントを使って続きます。前述のコードサンプルでは、エンド ユーザーが **SdLicense2** ダイアログで [戻る] ボタンをクリックした場合、goto ステートメントが Dlg_SdWelcome ラベルにジャンプして、エンド ユーザーには **SdWelcome** ダイアログが表示されます。このため、2 つのダイアログ (たとえば、**SdWelcome** と **SdLicense2**) の間にダイアログ関数を挿入する場合、if ステートメント、ラベル、および goto ステートメントを適切に調整する必要があります。

タスク ***インストールのユーザー インターフェイスの一部としてエンド ユーザーにダイアログを表示すには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
2. ダイアログ関数を、適切なスクリプト イベント ハンドラーに追加します。
3. 目的とするダイアログの動作に応じて、ダイアログ関数のパラメーターを変更します。

利用可能なパラメーターについては、ダイアログ関数に関するドキュメントを参照してください:

- ダイアログ関数
- ダイアログのカスタマイズ関数

4. たとえば、if および goto ステートメントを使って、ダイアログの流れを指定します。

InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるテキストを変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ほとんどのダイアログ関数は、ダイアログのタイトル領域に表示されるテキストを定義する szTitle と呼ばれる文字列引数を受け付けます。たとえば、**SdRegisterUser** の呼び出しは次のとおりです:

```
SdRegisterUser( szTitle, szMsg, szName, szCompany );
```

デフォルトで、パラメーター szTitle はヌル文字列 (“”) として定義され、ダイアログがデフォルトのタイトル テキストを使用することを示します。**SdRegisterUser** では、デフォルト タイトルが次のスクリーンショットのように表示されます。

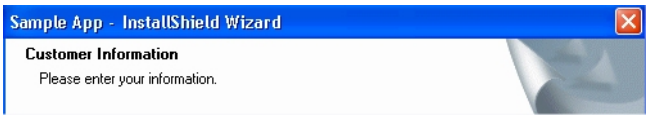


図 1: SdRegisterUser ダイアログのデフォルトのタイトル テキスト

タイトルを変更するには、szTitle パラメーターに表示する特定の文字列を入力します。タイトルは 2 つのセクションに分かれていて、タイトル領域の上部にある太字テキストと、メイン タイトルの下にある標準テキストがあります。

szTitle の値内で、この 2 つのセクションを指定するために、セクションの間に改行文字 ¥n を配置します。

```
SdRegisterUser (“新しいタイトル¥nこれは、新しいサブタイトルです。”,
    szMsg, szName, szCompany);
```

スクリプトを再コンパイルしてからインストールを実行すると、タイトルが次のサンプル スクリーンショットのように表示されます。

図 2: SdRegisterUser ダイアログの変更済みのタイトル テキスト

ダイアログに表示される他のテキストを変更する場合、表示されるテキストを含むパラメーター (たとえば、szMsg) が通常 1つ以上あります。ダイアログ ボックスのタイトルと同様に、メッセージ パラメーターのヌル文字列は、ダイアログが InstallShield で提供されているデフォルト メッセージ テキストを使用することを示します。

インストールに複数の UI 言語が含まれている場合、InstallScript でハードコード化された文字列の代わりに文字列 ID を使用できます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

InstallScript と InstallScript MSI プロジェクトのダイアログに含まれるデフォルト イメージについての背景情報

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

コントロール ID が 1200 であるスタティック ダイアログ ユーザー インターフェイス コントロールには、2 種類のイメージ (内部ダイアログのデフォルト バナー イメージ、および [ようこそ] ダイアログといった外部スキン ダイアログの左側に表示されるイメージ) が使用されます。デフォルト イメージは、透明色の使用が可能な .png イメージです。これらのコントロールのテキスト フィールドは、イメージを識別するのにデフォルトで次の構文が使用されます:

@@ResourceID;ScaleFactor

ResourceID は、.png イメージ (リソース タイプ PNG として格納される) またはビットマップ イメージ (リソース タイプ RT_BITMAP として格納される) のいずれかを検出するためのリソース識別番号を示します。*ScaleFactor* は、イメージが意図されている DPI スケール率を示します。

たとえば、スケール因子は 100% (96 DPI)、125% (120 DPI)、150% (144 DPI)、200% (192 DPI) となります。イメージに指定されているスケール因子が 200 の場合、200% DPI スケール以下で実行中のターゲット システム上で、イメージは縮小表示されます。200% ターゲットシステム上で、これは 1:1 の割合で表示されます。イメージに指定されているスケール因子が 100 の場合、200% DPI スケールで実行中のターゲット システム上で、イメージは拡大表示されます。

ビットマップ イメージ (.bmp) を識別する以前のフォーマットも使用できますが、スケーリングまたは .png イメージがサポートされていません:

@BitmapResourceID;TransparentFlag;3-DFlag;<unused>;TransparentColorKey

BitmapResourceID は、ビットマップ イメージのリソース ID 番号を示します。*TransparentFlag* は、1 (true) または 0 (false) のいずれかで示され、ビットマップを表示する際、*TransparentColorKey* フィールドで指定した色が透明かそうでないかを示します。*TransparentColorKey* は、ビットマップの透過色を示す RGB 値を指定します。

InstallScript および InstallScript MSI プロジェクトで新しいカスタム ダイアログを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

カスタム ダイアログを作成するには、次の一般的な手順を行う必要があります:

1. 新規ダイアログ ウィザードを使って、プロジェクトに新しいカスタム ダイアログを追加します。詳細については、「新規ダイアログ ウィザードを使って、InstallScript または InstallScript MSI プロジェクトに新しいカスタム ダイアログを追加する」を参照してください。
2. ダイアログにコントロールを追加します。詳細については、「InstallScript または InstallScript MSI プロジェクトのダイアログにコントロールを追加する」を参照してください。
3. ダイアログをメモリにロード、画面に表示、ダイアログのコントロールとのユーザーの対話を処理し、エンド ユーザーの操作が終了したときにダイアログを閉じるスクリプト関数を作成します。詳細については、「InstallScript を使用してカスタム ダイアログを実装する」を参照してください。

新規ダイアログ ウィザードを使って、InstallScript または InstallScript MSI プロジェクトに新しいカスタム ダイアログを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

タスク ***新しいカスタムダイアログをプロジェクトに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** を選択します。**新規ダイアログ ウィザード**が開きます。
3. ウィザードパネルに従い、新しいダイアログを作成します。**[ダイアログ テンプレート]** パネルで、**NewScriptBasedDialog** テンプレートまたは **NewSkinnableDialog** テンプレートを選択します。

   NewScriptBasedDialog および NewSkinnableDialog テンプレートには、コントロール ID 値 2 を持つ非表示コントロールが含まれています。エンド ユーザーが InstallScript ダイアログの右上にある閉じるボタンをクリックしてインストールをキャンセルできるようにするには、ダイアログにコントロール ID プロパティがこの値に設定されているボタン コントロールを含まなくてはなりません。ブランク ダイアログ テンプレートには、このコントロールは含まれていません。詳細については、「InstallScript を使用してカスタム ダイアログを実装する」を参照してください。

新しいカスタム ダイアログをプロジェクトに追加した後、ダイアログ コントロールを追加できます。

InstallScript または InstallScript MSI プロジェクトのダイアログ レイアウトを編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ エディターを使って、選択したダイアログのコントロールを追加、削除、または並べ替えることができます。またダイアログ エディターでは、ダイアログまたはそのコントロールのプロパティを変更することもできます。InstallShield ダイアログ エディターは、他のリソースエディターと同様に機能します。

ダイアログ エディターを使ってダイアログのレイアウトや外観を変更するとき、現在のプロジェクトに含まれるダイアログのコピーを変更しているだけで、その他のプロジェクトには変更が反映されません。

タスク ***ダイアログ エディターにアクセスするには、以下の手順を実行します。***

- **[ダイアログ]** ビューで、右側のダイアログ プレビュー ペインに表示されるダイアログをクリックし、編集するダイアログの言語リンクをクリックします。
- **[ダイアログ]** ビューで、ダイアログを右クリックし **[編集]** をクリックします。

中央ペインにダイアログ エディターが表示されます。

メモ: *エンド ユーザーがデスクトップ ディスプレイ テーマを使用している Windows プラットフォームでインストールを実行すると、そのテーマがインストールのエンドユーザー ダイアログを表示するときに使用されます。*

InstallScript または InstallScript MSI プロジェクトのダイアログにコントロールを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield の [コントロール] ツールバーを使用して、コントロールをダイアログに追加することができます。

タスク ***ダイアログにコントロールを追加するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、それから新しいダイアログ コントロールを追加するダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
4. **[コントロール]** ツールバーで、追加するコントロールをクリックします。
5. ダイアログ上のコントロールを表示する位置に長方形を描きます。
6. 右側のペインで、コントロールのプロパティを設定します。

[ダイアログ エディター] にコントロールを追加した後、ユーザーとコントロール間の対話を InstallScript を使って処理します。詳細については、「InstallScript を利用してダイアログ コントロールを処理する」を参照してください。

メモ: *Microsoft Visual Studio で作成されたプロジェクトの場合、ツールボックスを使用してダイアログ コントロールを追加できます。*

コントロールのプロパティを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ自体とそのすべてのコントロールには、ダイアログ エディター内にプロパティシートがあります。これらのプロパティでは、コントロールのサイズや位置、そのデフォルトのテキスト（文字列 ID ）またはダイアログがモーダルかどうかを判別します。

プロパティシートの 1 つを編集するには、ダイアログ上でコントロールを選択するか、プロパティシート上方のボックスからコントロールまたはダイアログの名前を選択します。プロパティはコントロールの種類に応じてさまざまです。

- ダイアログ
- チェック ボックス
- プッシュ ボタン
- 編集フィールド
- コンボ ボックス
- テキスト領域
- リスト ボックス
- ラジオ ボタン
- ビットマップ
- グループ ボックス
- 線
- ラジオ ボタン グループ
- 選択ツリー
- 進行状況バー
- リスト ビュー
- アイコン

ビットマップの上部にコントロールを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ビットマップその他のダイアログ コントロールのプロパティを設定するとき、コントロールがビットマップの上に配置されるように指定することができます。

タスク ***ビットマップ上にコントロールを表示するには、次の手順に従います:***

- ビットマップの **Tab Index** プロパティの値には、ビットマップの上に配置されるすべてのコントロールの **Tab Index** プロパティの値よりも小さい値を使用します。これを構成する最も簡単な方法は、ビットマップ コントロールの **Tab Index** プロパティを 0 に設定することです。
- すべてのコントロール (ビットマップを含む) では、**Tab Stop** のプロパティを **True** に設定しておく必要があります。

InstallScript を使用してカスタム ダイアログを実装する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

カスタム ダイアログ作成の次のステップは、ダイアログ コントロールとのエンドユーザーの対話を処理する InstallScript 関数の作成です。

タスク ***InstallScript を使用して、InstallScript プロジェクトでカスタム ダイアログを実装するには、次の手順を実行します。***

1. 最初に、カスタム ダイアログのコードを含む InstallScript ソース ファイルの新規作成を行います。ファイルに **CustomDialog.rul** という名前を付けます。

   ヒント: *他のプロジェクトでダイアログ スクリプトを使用する場合は、.rul ファイルを別の場所にコピーします。他のプロジェクトで、**InstallScript** ビューを開き、**InstallScript** エクスプローラーで **[ファイル]** を右クリックして、**[スクリプト ファイルの挿入]** をクリックします。*

2. CustomDialog.rul 内で、カスタム ダイアログ関数用のプロトタイプと実装を配置します。次のコードを CustomDialog.rul に追加します。

```
prototype NUMBER CustomDialog( );
function NUMBER CustomDialog( )
begin
end;
```

3. スクリプトをコンパイルしたときに、次の行を Setup.rul 上部付近に挿入して、メインのスクリプトの Setup.rul に CustomDialog 関数のコードが含まれていること示します:

```
#include "CustomDialog.rul"
```

4. CustomDialog.rul スクリプトで、ダイアログに追加したコントロールの数値 ID を格納する定数を定義します。標準ダイアログから [戻る]、[次へ] および [キャンセル] ボタンをコピーした場合、CustomDialog.rulの上部付近に次の行を追加することができます。

```
// コントロール識別子
#define BUTTON_NEXT    1
#define BUTTON_BACK    12
```

一般に、ダイアログのコントロールの数値 ID は、コントロールの Control Identifier プロパティにリストされている数値で、ダイアログ エディターでコントロールを選択するとプロパティ リストに表示されます。

***重要・**エンド ユーザーが InstallScript ダイアログの右上にある閉じるボタンをクリックしてインストールをキャンセルできるようにするには、 Control Identifier プロパティに値 2 を持つボタン コントロールをそのダイアログに含まなくてはなりません。このボタンはダイアログ内の任意の場所に配置でき、必要であればボタンを非表示にすることも可能です。これは、ダイアログにこの ID を持つボタンが含まれていない場合、インストールはダイアログの閉じるボタンがクリックされたときに生成されるメッセージを渡しません。*

たとえばチェック ボックス、編集フィールド、またはリストボックスといった、エンドユーザーが対話可能なその他の各コントロールに追加定数を定義する必要があります。

5. CustomDialog 関数は、**EzDefineDialog** 関数を使用して、カスタム ダイアログをメモリにロードします。

```
EzDefineDialog(
  "CustomDialog", // ダイアログのニックネーム
  ISUSER,         // ダイアログのリソースを含む DLL
  "CustomDialog", // [ダイアログ] ビューのダイアログの名前
  0);             // ダイアログの数値リソース ID (ここでは使用されない
```

EzDefineDialog と共に使う引数については、EzDefineDialog を参照してください。

***ヒント・**ダイアログ エディターで表示されるとおり、ダイアログのリソース ID はダイアログ名となります。数値リソース ID を指定する必要がある場合、カスタム ダイアログの Dialog テーブルにある **ISResourceId** 列に数値 ID を追加することができます。Dialog テーブルには、ダイレクト エディターを使ってアクセスできます。*

6. カスタム ダイアログのスクリプトでメッセージ ループを作成します。メッセージ ループは繰り返し **WaitOnDialog** 関数を呼び出します。この関数が呼び出されると、ユーザーがインタラクトするコントロールの数値コントロール ID が戻されます。通常のメッセージループは次のように表示されます。

```
// ユーザーが [次へ]、[戻る]、または [キャンセル] ボタンを使用してダイアログ ボックスを
// 終了するまで WaitOnDialog を繰り返し呼び出す
while (!bDone)

// ユーザーがコントロールと対話するのを待ち、
// コントロールの ID を戻す
nCtrl = WaitOnDialog("CustomDialog");

// switch ステートメントを使用して、異なるコントロールを処理する
```

```
switch (nCtrl)

    case CONTROL1:
        // ユーザーが CONTROL1 をクリックしたときにアクションを実行する

    case CONTROL2:
        // ユーザーが CONTROL2 をクリックしたときにアクションを実行する

    // その他のコントロール用の case ステートメント

endswitch;

endwhile;
```

たとえば、CustomDialog は現在 **[次へ]**、**[戻る]**、および **[キャンセル]** ボタンを含みます。そのメッセージループは次のように表示される可能性があります。

```
while (!bDone)

nControl = WaitOnDialog("CustomDialog");

switch (nControl)

    case DLG_INIT:
        // このダイアログの [戻る]、[次へ]、および [キャンセル] ボタンと有効/無効状態を
        // 初期化して、コントロール ID 700-724 および 202 上で %P、%VS、%VI を
        // それぞれ IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、および
        // IFX_INSTALLED_DISPLAY_VERSION で 置換します。
        hwndDlg = CmdGetHwndDlg("CustomDialog");
        SdGeneralInit("CustomDialog", hwndDlg, 0, "");

    case BUTTON_BACK:
        // ユーザーが [戻る] をクリック
        nReturn = BUTTON_BACK;
        bDone = TRUE;

    case BUTTON_NEXT:
        // ユーザーが [次へ] をクリック
        nReturn = BUTTON_NEXT;
        bDone = TRUE;


    デフォルト:
        // 標準のコントロール処理を確認します
        if (SdIsStdButton(nControl) && SdDoStdButton(nControl)) then
            bDone = TRUE;
        endif;

endswitch;

endwhile;
```

7. エンドユーザーが **[戻る]** または **[次へ]** をクリックすることによってダイアログを終了した場合、**EndDialog** と **ReleaseDialog** を使用して、画面とメモリからダイアログを削除します。

```
EndDialog("CustomDialog");
ReleaseDialog("CustomDialog");
```

メイン スクリプトでダイアログを使用するには、Setup.rul の **OnFirstUIBefore** イベント ハンドラーなど、CustomDialog 関数に呼び出しを追加します。

```
Dlg_SdWelcome:
    szTitle = "";
    szMsg   = "";
    nResult = SdWelcome(szTitle, szMsg);

Dlg_CustomDlg:
    nResult = CustomDialog( );
    if (nResult = BUTTON_BACK) goto Dlg_SdWelcome;

// 以下省略
```

エンドユーザーが **[戻る]** または **[次へ]** をクリックした場合、スクリプトは 1 つ前、または 1 つ後のダイアログを表示します。ユーザーが [**キャンセル**] をクリックした場合、標準の確認ダイアログが表示されます (**OnCanceling** イベント ハンドラーが処理)。

メモ: *ダイアログ コントロールの機能の実装に関する情報は、「InstallScript を使ったダイアログ コントロールの処理」を参照してください。*

特殊メッセージ

コントロール ID を返す以外にも、**WaitOnDialog** 関数はいくつかの特殊メッセージを返します。

- ダイアログが画面に表示される直前に、**WaitOnDialog** はメッセージ定数の DLG_INIT を返します。DLG_INIT case ステートメントでは、チェック ボックスやラジオ ボタンのデフォルトのステータスを設定したり、リストボックスやコンボボックス コントロールにデータを入力してデフォルトの選択を設定したり、編集フィールドの初期テキストを設定することができます。
- エラーが発生すると、**WaitOnDialog** は DLG_ERR 定数を返します。

InstallScript を利用してダイアログ コントロールを処理する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript および InstallScript MSI インストール プロジェクトでは、InstallScript を使用して、カスタム ダイアログに追加したコントロールを処理します。

チェック ボックスコントロールの使用

ボタンのクリックを処理することのほかに、カスタム ダイアログは通常、チェック ボックスなどのダイアログ コントロールにおけるエンド ユーザーの選択を取得できる機能も必要です 。

メモ: InstallShield には、AskOptions という標準ダイアログがあります。これは最大 9 つまでのチェック ボックスやラジオボタンを表示するので、エンドユーザーにチェック ボックスを表示するためだけにカスタム ダイアログを作成する必要はありません。

押しボタンと同様に、ダイアログ ボックスに追加する各チェック ボックス コントロールに対し、通常は数値コントロール ID に象徴的な名前を割り当てる #define ステートメントをスクリプトに追加します。たとえば、チェック ボックス コントロールをカスタム ダイアログ ボックスに追加すると、コントロールの数値 ID はコントロールの Control Identifier プロパティに表示されます。数値 ID が 1302 の場合は、次の行をスクリプトに追加します。

```
#define MYCHECKBOX1 1302
```

ほとんどのタイプのコントロールに対し、InstallScript は **CtrlGet** 関数と **CtrlSet** 関数を定義します。これらの関数は、コントロールの現在の状態またはデータを取得あるいは設定するために使用できます。たとえば、**CtrlGetState** を使用するとチェック ボックス コントロールの現在の状態を取得でき、**CtrlSetState** を使用するとチェック ボックス コントロールの現在の状態を設定できます。ダイアログのメッセージ ループの DLG_INIT case で、**CtrlSetState** を呼び出してチェック ボックスの初期選択状態を設定することができます。次のコードは、チェック ボックスが予め選択された状態で表示されるように設定します。

```
case DLG_INIT:
   // このダイアログの [戻る]、[次へ]、および [キャンセル] ボタンと有効/無効状態を初期化して、
   // ID 700-724 および 202 の %P、%VS、%VI を IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION、
   // および IFX_INSTALLED_DISPLAY_VERSION でそれぞれ置換します。
   hwndDlg = CmdGetHwndDlg("CustomDialog");
   SdGeneralInit("CustomDialog", hwndDlg, 0, "");
   CtrlSetState("CustomDialog", MYCHECKBOX1, BUTTON_CHECKED);
```

同様に、ダイアログのメッセージ ループの外に次のコードを追加すると、チェック ボックスの最終的な選択状態を検出することができます。

```
nState = CtrlGetState("CustomDialog", MYCHECKBOX1);
if (nState = BUTTON_CHECKED) then
   // チェックボックスが選択されている
else
   // チェックボックスが選択されていない
endif;
```

編集フィールドの使用

編集コントロールをカスタム ダイアログに追加することもできます。これにより、エンドユーザーは 1 行のテキストを入力できるようになります。InstallShield には、**SdShowDlgEdit1**、**SdShowDlgEdit2**、および **SdShowDlgEdit3** と呼ばれる標準ダイアログがあります。これは、エンドユーザーがテキストを入力できる 1、2、または 3 つの編集フィールドを表示します。

編集コントロールに格納されているテキストは **CtrlGetText** を使用して読み取ることができます。たとえば、リソース ID が 10000 の編集コントロールからテキストを読み取って svEdit という文字列変数に格納するには、次のコードを使用します。

```
CtrlGetText("CustomDialog", 10000, svEdit);
```

同様に、**CtrlSetText** を使用すると、編集コントロールに格納される (メッセージ ループの switch ステートメントの DLG_INIT ブロック内) 初期のテキストを設定することができます。

*ヒント: 編集コントロールの**パスワード**属性を利用して、編集フィールドにエンドユーザーが入力する文字を非表示に設定することができます。**番号**属性を利用して、編集フィールドにエンドユーザーが数値のみを入力できるように設定することができます。*

リストボックスとコンボボックスのコントロールの使用

リストボックスとコンボボックスのコントロールは LIST タイプの変数と関連付ける必要があります。リストボックスのコントロールを表示する前に、文字列リスト変数にデータを入力し、ダイアログの DLG_INIT ハンドラーで **CtrlSetList** を使用してリストボックスまたはコンボ ボックスのコントロールと関連付ける必要があります(コンボ ボックスのコントロールに対しては、通常 Drop-Down List プロパティを True に設定し、Height プロパティをコントロールのドロップダウンリスト部分の高さに設定します)。

リスト ボックスまたはコンボ ボックスのコントロールに初期選択を設定するには、ダイアログのDLG_INIT ハンドラーで **CtrlSetCurSel** を呼び出します。また、ユーザーの現在の選択を取得するには、**CtrlGetCurSel** を呼び出します。たとえば、次のコードは使用可能な各ドライブの文字をリスト変数に入力し (GetValidDrivesList を使用)、リストをコンボ ボックス コントロールに関連付け、ダイアログを終了する前にコンボ ボックスからエンドユーザーの選択を読み取ります。

```
function NUMBER CustomDialog( )
  NUMBER nReturn;
  NUMBER nControl;
BOOL   bDone;
      // コンボボックスリストと現在の選択のための変数
  LIST listDrives;
  STRING svDrive;
begin
    nReturn = EzDefineDialog("CustomDialog", ISUSER, "CustomDialog", 0);

    bDone = FALSE;

    // コンボボックス項目を含むリストの作成
    listDrives = ListCreate(STRINGLIST);
    // 使用可能な全ドライブ名をリストに入力
    GetValidDrivesList(listDrives, -1, -1);

    while (!bDone)

    nControl = WaitOnDialog("CustomDialog");

    switch (nControl)

        case DLG_INIT:
            // このダイアログの [戻る]、[次へ]、および [キャンセル] ボタンと有効/無効状態を初期化して、
            // コントロール ID 700-724 および 202 上で %P、%VS、%VI を
            // IFX_PRODUCT_DISPLAY_NAME、IFX_PRODUCT_DISPLAY_VERSION および IFX_INSTALLED_DISPLAY_VERSION で
            // それぞれ置換します。
            hwndDlg = CmdGetHwndDlg("CustomDialog");
            SdGeneralInit("CustomDialog", hwndDlg, 0, "");
            CtrlSetState("CustomDialog", MYCHECKBOX1, BUTTON_CHECKED);
            // コンボボックスとリストの関連付け
            CtrlSetList("CustomDialog", MYCOMBOBOX1, listDrives);
            // リストからの最初のドライブ名の取得...
            ListGetFirstString(listDrives, svDrive);
```

```
            // ...そしてそれを現在の選択に設定
            CtrlSetCurSel("CustomDialog", MYCOMBOBOX1, svDrive);
        // その他のコントロール用の case ステートメント

    endswitch;

    endwhile;

    // エンドユーザーの選択を取得し、メッセージボックスに表示
    CtrlGetCurSel("CustomDialog", MYCOMBOBOX1, svDrive);
    MessageBox("次のドライブを選択しました: " + svDrive, INFORMATION);

    EndDialog("CustomDialog");
    ReleaseDialog("CustomDialog");

    return nReturn;
end;
```

リスト ボックスとコンボ ボックス コントロールは Sorted プロパティがあります。これを [はい] に設定すると、リスト コントロールの内容をアイテムの表示名 別に並べ替えることができます。

InstallScript または InstallScript MSI プロジェクトでダイアログの動作を編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログの動作を変更するには、希望する動作に応じてダイアログ関数のパラメーターを変更します。利用可能なパラメーターについては、ダイアログ関数に関するドキュメントを参照してください:

- ダイアログ関数
- ダイアログのカスタマイズ関数

"Sd ダイアログ スタティック テキスト" フィールドに製品名、製品バージョン、または、インストール済みバージョンを含むフィールドを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield では、Sd ダイアログ スタティック テキスト フィールドに、%P、%VS、または %VI (場合により、追加スペースとして表示されます)というプレースホルダーを含む製品名、製品バージョン、または、インストールされたバージョンをグローバルに配置することができます。実行時に、システム変数 IFX_PRODUCT_DISPLAY_NAME, IFX_PRODUCT_DISPLAY_VERSION と IFX_INSTALLED_DISPLAY_VERSION の値がスタティック テキスト フィールドで %P、%VS、および %VI の代わりに表示されます。

プレース フォルダーを含むスタティック テキスト フィールドを追加する場合、701 から 799 の範囲内で (そのダイアログに対して) 一意の ID をスタティック テキスト フィールドに与えます。ID が 701 から 799 のとき、InstallShield はスタティック テキスト フィールドをスキャンしてプレースホルダーの存在を確認します。プレースホルダーが見つかると、InstallShield はそれを対応するシステム変数の値で置き換えます。

ダイアログで HTML コントロールを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield では、InstallScript と InstallScript MSI プロジェクトに含まれるダイアログで HTML コントロールをサポートします。HTML コントロール を使って、ダイアログ コントロールで HTML マークアップを使用できます。ダイアログで Web ページ、インストール済みの HTML ファイル、および HTML サポートファイルへのリンクを含めることができます。エンド ユーザーが実行時のダイアログでハイパーリンクをクリックしたときに、インターネット ブラウザーで HTML ページを開くか、InstallScript コードを使って定義した別の動作をトリガーできます。HTML コントロールを使って、外観を制御するためのスタイルを含む、有効な HTML マークアップを使用できます。

HTML コンテンツがインストール済みの HTML ファイルまたは HTML サポート ファイルである場合、HTML コントロールを使って、それを直接ダイアログに表示することも可能です。

HTML コントロールを作成するには、まずスタティック テキスト コントロールを作成します。ダイアログにスタティック テキストコントロールを配置してから、それを HTML コントロールに変換できます。

タスク ***ダイアログの任意のスタティック テキスト コントロールを HTML コントロールに変換するには、以下の手順の 1 つを実行します:***

- [ダイアログ] ビューで、コントロールの Text プロパティを次のとおりに設定します:

  [html] *HTML マークアップ*

- InstallScript を使って、コントロールのテキストを **[html]** に設定して、後に HTML マークアップを続けます。

実行時に、InstallScript エンジンがコントロールを HTML コントロールに変換します。

例

次の例では、スタティック テキスト コントロールを HTML コントロールに変換する方法をデモンストレーションします。サンプル HTML コードを使って、コントロールの色とフォントを設定して、ダイアログと一致させます。

タスク ***コントロールの Text プロパティを使って、スタティック テキスト コントロールを HTML コントロールに変換するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、HTML コントロールを含めるダイアログを右クリックして、**[編集]** をクリックします。ダイアログ エディターに、プロジェクトのデフォルト言語でダイアログが表示されます。

3. HTML コントロールに変換するスタティック テキスト コントロールをクリックします。InstallShield の右側に設定が表示されます。
4. **Text** プロパティで、省略記号ボタン (...) をクリックしますプロジェクトに含まれるすべての現在の文字列のリストが表示されます。
5. **[編集]** ボタンをクリックします。
6. 文字列 ID の値を次のように設定します:

   [html]<style type="text/css">html,body {padding:0; margin:0; background-color:ButtonFace} * {font-size: 8pt; font-family: "MS Sans Serif";} </style> Web サイトにアクセス

実行時、インストールはデフォルトの InstallScript ダイアログ フォント (8 ポイント MS Sans Serif) を使い、背景色を Windows ダイアログの色 (通常は灰色) に設定します。コンテンツが白色のダイアログ領域にある場合、*background-color:ButtonFace* スタイル部分を省略して、HTML コントロールの背景が白色になるようにします。

ヒント: *[ダイアログ] ビューを通してスタティック コントロールを変換する代わりに、InstallScript ビューでダイアログのスクリプト関数を編集することもできます。たとえば、ダイアログ メッセージ ループの DLG_INIT ケースに次のコードを追加します:*

*CtrlSetText(szDlg, HTML_CTRL_ID, "[html]<style type=¥"text/css¥">html,body {padding:0; margin:0; background-color:ButtonFace} * {font-size: 8pt; font-family: "MS Sans Serif";} </style>Web サイトにアクセス site");*

その他のスクリプト サンプルは、「 CtrlGetUrlForLinkClicked Example」を参照してください。

HTML コントロールにあるリンクのクリック イベントの処理方法

CtrlSetText 関数を使って HTML コントロール コンテンツを実行時に設定して、コントロールの作成に成功すると、**CtrlSetText** が 0 を返します。失敗すると、**CtrlSetText** が ISERR_GEN_FAILURE を返します。

HTML コントロールにあるリンク イベントを処理するため、**WaitOnDialog** 関数は、HTML コントロールのコントロール ID を返します (これは、元のスタティック テキスト コントロールのコントロール ID と同じです)。カスタム ダイアログ スクリプト関数に HTML コントロール ID を処理する case ステートメントを含めて、**CtrlGetUrlForLinkClicked** 関数を呼び出してクリックされたリンクの URL を取得できます。スクリプトはこのリンクを使って、たとえばインターネット ブラウザーを起動してリンクにナビゲートするなどの任意のアクションを行うことができます。

エンド ユーザーがアンカー タグを使った HTML コントロールのリンクをクリックすると、次の動作が発生します:

- アンカー タグにターゲット属性が含まれていない、またはターゲット値が新しいウインドウに表示されない場合、コントロール ID がダイアログ スクリプトに戻ります。これによって、スクリプトがリンク クリック自体を処理できます。
- アンカー タグにターゲット属性が含まれていて、その値が新しいウィンドウに表示される _blank またはそれに等しい値であるとき、メッセージはスクリプトに送られず、URL が自動的に起動されます。

HTML コントロールに関する特別考慮

ダイアログに HTML コントロールを追加する場合は、以下の詳細に注意してください。

HTML コントロールの実行時の要件

HTML コントロールは、完全に機能するためには Internet Explorer 5.5 以降を条件とします。ターゲット システムに以前のバージョンが存在するとき、HTML コントロールのコンテンツを設定する唯一の方法は、HTML ファイルを **CtrlSetText** に提供する方法です。ターゲット システム上で、Internet Explorer がデフォルトのブラウザーである必要はありません。

HTML ファイルの HTML コントロール サポート

HTML コントロールで HTML ファイルを使用する場合、ファイルへのフル パスを **CtrlSetText** 関数を使って指定しなくてはなりません。そうでない限り、InstallScript エンジンがファイルの場所を認識できません。サンプル InstallScript は次の通りです:

```
CtrlSetText(szDlg, HTML_CTRL_ID, "[html]file://" + SUPPORTDIR^"aboutpage.htm");
```

電子メール アドレスに HTML コントロールを使用する

電子メール アドレスをポイントするハイパーリンクを追加するには、次のようなコードを使用します:

```
CtrlSetText(szDlg, HTML_CTRL_ID, "[html]<a href=¥"mailto:support@mycompany.com?Subject=Example%20Line¥">サポートに電子メールを送信</a>");
```

エンド ユーザーがこのようなリンクをクリックしたとき、ターゲット システムの電子メール アプリケーションで新しい電子メール メッセージが自動的に開きます。mailto リンクでは、スクリプトにメッセージは送信されませんので、ご注意ください。

フォーマットされていない HTML テキスト コンテンツ

HTML コントロールのコンテンツ セットはそのままの形式で渡されます。コンテンツのフォーマットは行われません。コンテンツをフォーマットする必要がある場合、スクリプトでコンテンツ文字列を操作してからコンテンツ文字列を **CtrlSetText** に渡します。

スタティック テキスト コントロールのテキスト プロパティ文字数制限

スタティック テキスト コントロールのテキスト プロパティの文字数制限は 256 文字です。したがって、コントロールのテキスト コンテンツの始めに **[html]** を追加してスタティック テキスト コントロールを HTML コントロールに変換する場合、入力可能な最大文字数は 256 文字です。これにはハイパーリンク テキストおよび指定する HTML マークアップ文字のすべてが含まれます。コンテンツに 256 文字以上を入力するには、**CtrlSetText** 関数を使ってコントロールを作成します。

デフォルトのダイアログに戻す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログを編集してから、後で変更内容をすべて元に戻す場合、元のデフォルト ダイアログに復元することができます。

タスク ***デフォルトのダイアログに元に戻すには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. 編集したアクションを右クリックして、**[元に戻す]** をクリックします。

リソース コンパイラとリソース リンカ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript または InstallScript MSI プロジェクトでダイアログに変更を加えるには、リソース コンパイラとリソース リンカをシステムにインストールしておく必要があります。InstallShield には、これらのタイプのツールが両方搭載されています。

タスク ***リソース コンパイラとリソース リンカの位置の確認は、次の手順で行います。***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[リソース]** タブをクリックします。

[リソース] タブには、コンパイラとリンカの場所の一覧があります。

タスク ***リソースコンパイラまたはリソースリンカを既にシステム上にあるものと置き換えるには、以下の手順に従います:***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[リソース]** タブをクリックします。
3. **[リソースのコンパイラの場所]** ボックス、または、**[リソースのリンカの場所]** ボックスの横にある [参照] ボタンをクリックして、プログラム ファイルまで移動します。

ダイアログ サンプラー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ダイアログ サンプラーは、InstallScript と InstallScript MSI プロジェクトにおけるビルトイン ダイアログのサンプル、および、スクリプト ダイアログ (sd) 関数によって呼ばれるダイアログを表示する InstallShield ウィザードです。異なる 2 つの ダイアログ サンプラーを起動することができます。

タスク ***ダイアログ ウィザードを起動するには、以下の手順を実行します。***

InstallShield の **[ツール]** メニューで **InstallScript** をポイントして、**[標準ダイアログ サンプラー]** または **[スキン ダイアログ サンプラー]** をクリックします。

[プログラム] メニューにある InstallShield プログラムフォルダーのショートカットを使用して、これらのサンプラーを起動することもできます。

ダイアログ スキン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

概要

ダイアログ スキンを使うと、エンドユーザー ダイアログを異なる外見とカラー スキームを使って表示できます。スキンされたダイアログは標準ダイアログよりやや大きめで、コントロールの場所も異なります。スキン付きダイアログを適切に表示するには、ターゲットマシンのデスクトップ領域が 800 x 600 ピクセル以上 (大きなフォントが利用されている場合は 1024 x 768 ピクセル) で、システムが 16-bit 色以上を表示可能でなくてはなりません。

制限

プロジェクトでダイアログのスキンを使用する場合、いくつか制約があります。

- 標準ダイアログを [ダイアログ エディタ] 内で一度編集すると、そのダイアログにダイアログ スキンを適用または削除することはできません。したがって、編集するダイアログのスキンを指定するときは、まずスキンを指定してから、編集してください。
- 標準ナビゲーション ボタン ([次へ]、[キャンセル]、[戻る]、および [終了]) の場所はすべてのスキンで共通です。.skin ファイルはその位置を制御します。ダイアログ エディターで移動しても、ランタイムの位置に影響はありません。また、いくつかのダイアログでは [参照] ボタン (例 **SdAskDestPath**) を再配置できない場合があります。
- スキン ダイアログではウィンドウ ビルボードを使用できません。ウィンドウ スタイルのビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。
- プロジェクトにカスタム ダイアログを追加する場合、新しいダイアログは、他のエンドユーザー ダイアログと同じサイズである必要があります。カスタム ダイアログのサイズが異なる場合、正しく表示されないことがあります。たとえば上記の位置関係の問題により、ナビゲーション ボタンが実行時にエンドユーザーに表示されないことがあります。

メモ: *カスタム ダイアログにダイアログ スキンを指定する場合、スキンが適切に表示されるためにダイアログ エディターでカスタム ダイアログを作成する前にスキンの設定を行う必要があります。*

ダイアログ スキンの指定

__プロジェクト__: この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*

ダイアログ スキンを使用して、インストールのエンドユーザー ダイアログの外観を変更することができます。1 つのプロジェクトにつき、1 つのスキンを指定することができます。プロジェクト内のすべてのダイアログは、選択したスキンを利用して表示されます。

__メモ__: 標準ダイアログを [ダイアログ エディタ] 内で一度編集すると、そのダイアログにダイアログ スキンを適用または削除することはできません。したがって、編集するダイアログのスキンを指定するときは、まずスキンを指定してから、編集してください。

カスタム ダイアログにダイアログ スキンを指定する場合、スキンが適切に表示されるためにダイアログ エディターでカスタム ダイアログを作成する前にスキンの設定を行う必要があります。

ダイアログ スキンの選択

タスク

ダイアログ スキンを選択するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[スキン]** を開いて利用可能なダイアログ スキンオプションを表示します。
3. スキンをクリックして、右側のパネルにプレビュー ダイアログを表示します。
4. 表示されたスキンを選択するには、プレビュー ペインの **[選択]** をクリックします。[**ダイアログ**] エクスプローラーで、選択したスキンのアイコンに赤いチェック記号が表示されます。

Blue スキン以外のすべてのダイアログ スキンは、フェード グラフィックに .gif ファイルを利用します。Blue スキンは .bmp ファイルを利用するのでファイルサイズが大きくなります。.gif ファイルを利用したとき、16 ビット カラーシステム上での Blue スキンの外観はその他のカラーに比べると鮮明ではありません。16-bit カラーシステムをサポートする Blue スキンを利用する場合で、ファイルサイズの大きさが関係ない場合は、Blue オプションを選択して下さい。より小さいサイズには、グラフィックに .gif ファイルを利用する BlueTC (True Color) オプションを選択してください。

ダイアログスキンの選択をクリアする

タスク

ダイアログ スキンの選択をクリアするには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[スキン]** アイテムを開いて利用可能なダイアログ スキンオプションを表示してから、**<なし>** アイテムを選択します。
3. プレビュー パネルの **[選択]** をクリックします。

基本の MSI プロジェクトでダイアログを使用する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

このセクションでは、基本の MSI プロジェクトで、ダイアログがある他の基本的な関数の作成および実行方法が説明されています。

タスク ***新しいダイアログを作成して、それをインストールで表示するプロセスは、次のように複数のタスクに分割できます:***

1. ダイアログをプロジェクトに追加する。
2. ダイアログのレイアウトを編集する。
3. コントロールの動作を定義する (表示する条件、インタラクションがトリガーするイベント、およびサブスクライブするイベント)。
4. ダイアログを表示する (別のダイアログのコントロール内の NewDialog または SpawnDialog イベントを使用するか、またはダイアログをプロジェクトのシーケンスに挿入します)

基本の MSI インストール中にダイアログを表示する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

次に説明するいずれかの方法で、エンドユーザー ダイアログを表示できます。

- [ユーザー インターフェイス] シーケンスにダイアログを挿入する。
- 別のダイアログの動作を介してダイアログを起動する。

シーケンスへダイアログを挿入する

プロジェクト: *マージ モジュール内でダイアログを作成する場合、ダイアログをシーケンスに挿入するにはモジュールをインストール プロジェクトと関連付ける必要があります。*

タスク ***ダイアログをシーケンスにスケジュールするには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
2. **[シーケンス]** エクスプローラーで、ダイアログが通常表示される 3 つの上位シーケンス (**[インストール]**、**[アドバタイズ]**、または、**[管理]**) から 1 つを展開して、その **[ユーザー インターフェイス]** シーケンスを表示します。
3. **[ユーザー インターフェイス]** シーケンスを展開して、既存のアクション、ダイアログ、およびカスタム アクションの順序付けを確認します。

4. シーケンス内で作業中のダイアログの前にくるアクションまたはダイアログを右クリックして、**[挿入]** をクリックします。[アクションの挿入] ダイアログ ボックスが開きます。
5. リストで、**ダイアログ**を選択します。
6. ダイアログのリストから、新しいダイアログを選択します。
7. **[OK]** をクリックします。

インストールが完全ユーザー インターフェイス (/q コマンドライン パラメーターで定義) で実行され、このダイアログに対するすべての条件が満たされると、ダイアログが選択したシーケンス内に表示されます。

InstallShield でダイアログを [実行] シーケンスに挿入できても、挿入すると ICE13 違反になります。

ダイアログを別のダイアログの動作を介して起動する

ほとんどのダイアログは、NewDialog (ウィザードで別のダイアログを表示する) または SpawnDialog (モデルダイアログを表示する) コントロールイベントの結果として表示されます。ダイアログの動作 (通常は [次へ] ボタン) でこのイベントを選択することによって、イベントを正常に実行するダイアログを指定できます。コントロールイベントに条件を追加することによって、直前のダイアログでの選択の結果に依存して別々のダイアログを表示できます。

タスク ***たとえば、InstallWelcome ダイアログの後に WelcomeBitmap ダイアログを表示するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**InstallWelcome** ダイアログを拡張してから、そのダイアログの **[動作]** サブノードをクリックします。
3. コントロール リストおよび対応するコントロールの種類を含むペインで、**[次へ]** ボタンを選択します。
4. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[NewDialog]** をクリックします。**"イベント"** 設定の下に、いくつかの **"NewDialog"** サブ設定が追加されます。
5. **"NewDialog"** 設定に、次のように入力します: **1**

   1 の条件は、ダイアログがすべての条件の下に作成されなくてはならないことを示します。
6. **"ダイアログ名"** 設定で、次を入力します: **WelcomeBitmap**

メモ: *InstallWelcome ダイアログが [次へ] ボタンから別のダイアログを起動した場合は、上記の手順を繰り返して WelcomeBitmap の [次へ] ボタンで次のダイアログを表示するようにします。*

[カスタム アクションとシーケンス] ビューで新しいダイアログを表示する詳しい方法については、「[カスタム アクションとシーケンス] ビューでエンドユーザー ダイアログの順序の表示する (基本の MSI プロジェクト)」を参照してください。

基本の MSI プロジェクトで新しいダイアログを作成する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***新規で基本の MSI プロジェクトを作成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** をクリックします。ダイアログ ウィザード が起動し、新規ダイアログ作成手順を案内します。

次のステップでは、ダイアログのレイアウトおよびコントロールの動作を編集します。

ヒント: *ダイアログを別のプロジェクトからインポートして追加することもできます。*

エンドユーザー インターフェイスでダイアログを表示する

プロジェクトにダイアログを追加しても、そのダイアログがインストールに表示されるわけではありません。1 つ以上のプロジェクトのシーケンスにダイアログを挿入する処理については、「基本の MSI インストール中にダイアログを表示する」を参照してください。

基本の MSI プロジェクトでダイアログ レイアウトを編集する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ダイアログ エディターを使用して、ダイアログの修正、コントロールの編集、およびダイアログとコントロールの表示プロパティの設定をすべてビジュアル環境で行うことができます。

メモ: *エンド ユーザーがデスクトップ ディスプレイ テーマを使用している Windows プラットフォームでインストールを実行すると、そのテーマがインストールのエンドユーザー ダイアログを表示するときに使用されます。*

[ダイアログ] ビューは、各プロジェクトのサポート言語のダイアログのバージョンを管理します。レイアウトを編集するには、言語固有のバージョンを選択しなければなりません。コントロールのサイズに加える変更を除き、これらのバージョンは変わりません。詳細については、「各言語ごとのダイアログを変更する」を参照してください。

ダイアログ エディターを開く

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***ダイアログ エディターでダイアログを開くには、次のいずれかを実行します。***

- **[ダイアログ]** ビューの中のダイアログの名前の下にある、ダイアログの言語固有のバージョンを選択します。
- **[カスタム アクションとシーケンス]** ビューで、ダイアログを右クリックして、**[レイアウトの編集]** をクリックします。

ダイアログ エディターは、[ダイアログ] ビューの右ペインで開きます。

基本の MSI プロジェクトでダイアログにコントロールを追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield の [コントロール] ツールバーを使用して、コントロールをダイアログに追加することができます。

タスク ***ダイアログにコントロールを追加するには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、それから新しいダイアログ コントロールを追加するダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
4. **[コントロール]** ツールバーで、追加するコントロールをクリックします。
5. ダイアログ上のコントロールを表示する位置に長方形を描きます。
6. 右側のペインで、コントロールのプロパティを設定します。

ヒント: *Microsoft Visual Studio 内から InstallShield プロジェクトを編集している場合、ツールボックスを使用してコントロールを追加できます。*

コントロールの設定を構成する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ダイアログ自体とそのすべてのコントロールには、ダイアログ エディター内に設定のグリッドが表示されます。これらの設定では、コントロールのサイズや位置、そのデフォルトのテキスト (文字列 ID など) またはダイアログがモーダルかどうかなどの特徴を判別します。

コントロールの設定を編集するには、ダイアログ上でコントロールを選択します。設定は、コントロールの種類に応じてさまざまです。

- ビットマップ
- ダイアログ
- チェック ボックス
- プッシュ ボタン
- 編集フィールド

- コンボ ボックス
- テキスト領域
- リスト ボックス
- ラジオ ボタン
- ビットマップ
- グループ ボックス
- ビルボード
- 線
- ラジオ ボタン グループ
- 選択ツリー
- 進行状況バー
- リスト ビュー
- スクロール可能テキスト
- アイコン
- ディレクトリ リスト
- ディレクトリ コンボ
- ボリューム コスト リスト
- ボリューム選択コンボ
- マスク編集
- パス編集

ビットマップ上にコントロールを表示する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***ビットマップ上にコントロールを表示するには、次の手順に従います:***

- ビットマップの **Tab Index** プロパティの値には、ビットマップの上に配置されるすべてのコントロールの **Tab Index** プロパティの値よりも小さい値を使用します。これを構成する最も簡単な方法は、ビットマップ コントロールの **Tab Index** プロパティを 0 に設定することです。
- すべてのコントロール (ビットマップを含む) では、**Tab Stop** のプロパティを **True** に設定しておく必要があります。

基本の MSI プロジェクトでダイアログ レイアウトを編集する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

[ダイアログ] ビューで、実行時のコントロールの動作を指定する設定を構成できます。コントロールの動作関連の設定では、次の種類の機能を構成できます:

- コントロールで対話するエンドユーザーが起動するイベント
- コントロールがサブスクライブする必要がある Windows Installer のイベント
- コントロールを表示する条件

タスク ***コントロールの動作設定にアクセスするには、以下の手順に従います:***

1. 以下のいずれかを実行します。
   - **[ダイアログ]** ビューで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
   - **[カスタム アクションとシーケンス]** ビューで、ダイアログを右クリックして、**[動作の編集]** をクリックします。
2. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、構成するコントロールをクリックします。設定が、右のペインに表示されます。
3. 必要に応じて **"イベント"**、**"サブスクリプション"**、および **"条件"** 設定を構成します。

基本の MSI ダイアログでコントロール イベントを使用する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

コントロール イベントを使うと、ダイアログの各コントロールに対してカスタム機能を提供することができます。コントロール イベントでは、カスタム アクションの起動や新しいダイアログの生成、また進行状況バーの表示などを行うことができます。多くのコントロール イベントは、パブリッシュされた情報を使用しています。また、情報を使用するには、コントロール イベント情報を サブスクライブしなければならないコントロールもあります。

イベントとコントロールを関連付けるには、[ダイアログ] ビューでダイアログの [動作] 領域を使用します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。コントロールのサブスクリプションに関する詳細は、「基本の MSI ダイアログでサブスクリプションを使用する」を参照してください。

このイベントの条件を指定するには、"イベント" 設定で省略記号ボタン (...) をクリックします。

次は使用可能なコントロール イベント一覧です:

テーブル 3 ・ コントロールのイベントの種類

コントロール イベント	説明
AddLocal	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) をローカルにインストールすることを設定します。 機能を指定するには、この設定のサブ設定を構成してください。
AddSource	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) をソースからインストールすることを設定します。 機能を指定するには、この設定のサブ設定を構成してください。
CheckExistingTargetPath	指定のパスが書き込み可能であることを検証し、また指定のパスが存在しない限り、追加コントロールイベントがトリガされないように防ぎます。 パスを含むプロパティを指定するには、この設定のサブ設定を構成します。
CheckTargetPath	指定された値が有効なパスでない場合、追加コントロール イベントがトリガされないように防ぎます。 パスを含むプロパティを指定するには、この設定のサブ設定を構成します。
DirectoryListNew	ディレクトリ リスト コントロールに、新規フォルダーの作成が必要であることを通知します。インストーラーは新しいフォルダーを作成すると、ユーザーはフォルダーに新しい名前を付けることができます。
DirectoryListOpen	ディレクトリ リスト コントロールに、選択されたディレクトリを開くことを通知します。ディレクトリが選択されていない場合、追加のコントロール イベントがトリガされないように防ぎます。
DirectoryListUp	ディレクトリ リスト コントロールに、1 階層上へ移動して現在のディレクトリの親を選択することを通知します。
DoAction	選択されたコントロールがアクティブになると、カスタム アクションをトリガします。 カスタム アクションを指定するには、この設定のサブ設定を構成してください。
EnableRollback	インストールのロールバック機能を有効または無効にします。 イベントがロールバックを有効にするかどうかを指定するには、この設定のサブ設定を構成してください。True の値はロールバックを有効にし、False はロールバックを無効にします。

テーブル 3 • コントロールのイベントの種類 (続き.)

コントロール イベント	説明
EndDialog	モーダル ダイアログを閉じます。このイベントは、プッシュ ボタン コントロールまたは選択ツリー コントロールに使用できます。 このイベントで発生させる動作を指定するには、この設定のサブ設定を構成してください。 標準ダイアログで使用可能なオプション: • **終了** – ダイアログ シーケンスが閉じると、コントロールによって UserExit の値がインストーラーに返されます。このオプションは、子ウィンドウがある場合は使用できません。 • **再試行** – ダイアログ シーケンスが閉じると、コントロールによって Suspend の値がインストーラーに返されます。このオプションは、子ウィンドウがある場合は使用できません。 • **無視** – ダイアログ シーケンスが閉じると、コントロールによって Finished の値がインストーラーに返されます。このオプションは、子ウィンドウがある場合は使用できません。 • **戻す**–ダイアログが終了して、コントロールが親ウィンドウに戻ります。親ウィンドウが存在しない場合、コントロールによって Success の値がインストーラーに返されます。 エラー ダイアログで使用可能なオプション: • ErrorOk • ErrorCancel • ErrorAbort • ErrorRetry • ErrorIgnore • ErrorYes • ErrorNo

テーブル 3 ・ コントロールのイベントの種類 (続き.)

コントロール イベント	説明
MsiLaunchApp	 ***メモ:*** *Windows Installer 5 で、このイベントがサポートされています。以前のバージョンの Windows Installer はこのイベントを無視します。したがって、Windows Installer 4.5 以前が搭載されているシステム上でインストールを実行するときに、このイベントを使用する場合、ダイアログ コントロールに条件を追加することで、Windows Installer 4.5 以前が搭載されたシステム上では、これが表示されないようにします。* 指定されたファイルを実行します。 通常このイベントは、SetupCompleteSuccess ダイアログ上のチェック ボックス コントロールに使用します。エンド ユーザーはチェック ボックスを使って、インストールの終わりにアプリケーションを実行するかどうかを選択できます。チェック ボックス コントロールには、アンインストール中にコントロールが表示されるのを防ぐ条件を含みます。 ファイルを起動するコマンドラインを指定するには、この設定のサブ設定を構成します。
MsiPrint	 ***メモ:*** *Windows Installer 5 で、このイベントがサポートされています。以前のバージョンの Windows Installer はこのイベントを無視します。したがって、Windows Installer 4.5 以前が搭載されているシステム上でインストールを実行するときに、このイベントを使用する場合、ダイアログ コントロールに条件を追加することで、Windows Installer 4.5 以前が搭載されたシステム上では、これが表示されないようにします。* エンド ユーザーがコントロールの内容を印刷できるようにします。 このイベントを、スクロール可能なテキスト コントロールを含むダイアログにあるプッシュ ボタン コントロールに追加できます。エンド ユーザーがプッシュ ボタン コントロールをクリックすると、スクロール可能なテキスト コントロールの内容が印刷されます。
NewDialog	別のダイアログに進みます。 新しいダイアログを指定するには、この設定のサブ設定を構成してください。
Reinstall	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) を再インストールします。 機能を指定するには、この設定のサブ設定を構成してください。
Remove	指定の機能 (または、**[すべて]** が指定された場合にはすべての機能) の削除が選択されていることを示します。 機能を指定するには、この設定のサブ設定を構成してください。

テーブル 3 • コントロールのイベントの種類 (続き.)

コントロール イベント	説明
Reset	ダイアログのすべてのコントロールが行なったプロパティの変更を元の値にリセットします。
RMShutdownAndRestart	再起動マネージャーが使用中のファイルを含むすべてのアプリケーションをシャットダウンし、インストールの終わりにそれらを再起動させることを示します。
SelectionBrowse	エンド ユーザーが強調表示されたアイテムのパスを変更できるようにする、指定された [参照] ダイアログを起動します。 ダイアログを指定するには、この設定のサブ設定を構成してください。
SetInstallLevel	INSTALLLEVEL プロパティを指定の値に設定します。 適切なインストール レベルを指定するには、この設定のサブ設定を構成してください。
SetProperty	プロパティの値を設定します。 プロパティの値を指定するには、この設定のサブ設定を構成してください。
SetTargetPath	選択されたパスを確認および設定します。 パスを含むプロパティを指定するには、この設定のサブ設定を構成します。
SpawnDialog	現在のダイアログを閉じずに、指定されたモーダルダイアログを起動します。
SpawnWaitDialog	条件式が True 評価されるまで待機するダイアログを起動してから、ダイアログを閉じます。
ValidateProductID	ProductID プロパティを設定します。

基本の MSI ダイアログでサブスクリプションを使用する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Windows Installer サービスは、実行時にインストールに多くの情報を提供します。インストール中の最も便利な情報収集方法はサブスクリプションです。サブスクライブとは、一般的に、パブリッシュまたはサービスに自分の名前を入力することと定義づけられます。Windows Installer では、アクションが通常情報をパブリッシュし、ダイアログ コントロールがその情報をサブスクライブします。

サブスクリプションの一般的な使用例

この関係を表す一番いい例は、進行状況バーです。InstallFiles アクションは、移動したファイルの割合と、残りのファイルの割合に関する情報をパブリッシュします。進行状況バーがこの情報をサブスクライブすると、インストールのファイル転送プロセス状況を正確に表示することができます。

Windows Installer は、「*ティック*」と呼ばれる進行状況を追跡します。進行状況バーがこの情報をサブスクライブしている場合、**ticksSoFar** および **totalTicks** の値を渡します。進行状況バーはこの情報を使用してインストールの全進行状況を表示します。

サブスクリプションは進行状況バー以外にも使用されます。シリアル番号をインストールの一部として検証するカスタム アクションがあることがあります。このアクションは検証の成否をパブリッシュできます。ダイアログで [次へ] ボタンをクリックするとこの情報をサブスクライブできます。アクションが検証結果として失敗をパブリッシュすると、ボタンは有効になりません。アクションが成功をパブリッシュすると、ボタンは有効になります。

サブスクリプションの種類

イベントとコントロールのサブスクリプションを設定するには、[ダイアログ] ビューでダイアログの [動作] 領域を使用します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

次はコントロールがサブスクライブ可能なイベントの一覧です:

テーブル 4 - コントロールがサブスクライブ可能なイベントの種類

イベント	説明
ActionData	最終アクションについての情報を表示します。 テキスト コントロールはこのイベントをサブスクライブすることにより、インストール中にコピーされたファイル名などの情報を表示できます。
ActionProgress	監視中のアクションについての進行状況を表示します。 進行状況バー コントロールが、このイベントをサブスクライブできます。 アクションを指定するには、この設定のサブ設定を構成してください。
ActionText	現在のアクションの名前を表示します。 テキストコントロールはこのイベントをサブスクライブすることにより、インストール中に実行された現在のアクションの説明を表示できます。
IgnoreChange	現在のディレクトリ内のフォルダを開かずに強調表示します。
ScriptInProgress	Windows Installer がインストールの実行スクリプトをコンパイル中に、情報テキストを表示します。
SelectionAction	強調表示されたアイテムについて説明する UI 文字列を表示します。 テキスト コントロールで、このイベントをサブスクライブして UI 文字列を表示できます。
SelectionDescription	テキスト コントロール内の UI 文字列を表示します。UI 文字列は、選択ツリー コントロールで選択された機能について説明します。
SelectionNoItems	説明テキストを削除するか、不要なボタンを無効にします。

テーブル 4 ・ コントロールがサブスクライブ可能なイベントの種類 (続き.)

イベント	説明
SelectionPath	選択ツリー コントロールで選択されたアイテムのパスをパブリッシュします。 テキスト コントロールで、このイベントをサブスクライブしてパスを表示できます。
SelectionPathOn	選択パスが現在選択されている機能と関連付けられているかどうかを示すブール値をパブリッシュします。
SelectionSize	強調表示されたアイテムのサイズをパブリッシュします。
SetProgress	インストールの進行状況についての情報をパブリッシュします。
TimeRemaining	現在処理中のシーケンスの残りの秒数を概算でパブリッシュします。 テキスト コントロールで、このイベントをサブスクライブして残りの時間を表示できます。

基本の MSI ダイアログでコントロール イベントをトリガーする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

イベントは、エンド ユーザーがコントロールを使用した時に発生する定義済みの動作です。イベントは、[ダイアログ] ビューの [動作] 領域 で定義されます。使用可能な各イベントに関する詳細は、「基本の MSI ダイアログでコントロール イベントを使用する」を参照してください。

コントロールのイベントを表示するには、コントロール名のリストからイベントを選択して、右のペインにある″イベント″ 設定を構成します。

ダイアログ コントロールに新しいイベントを追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク

ダイアログコントロールに新しいイベントを追加するには、以下の手順を実行します。

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、構成するコントロールをクリックします。設定が、右のペインに表示されます。

4. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから、追加するイベントの種類を選択します。**"イベント"** 設定の下に新しい一連の行が追加されます。
5. イベントの条件を指定するには、"イベント" 設定で省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開き、ここでアクションに対して満たさなくてはならない条件を指定できます。

   任意の条件下でイベントをトリガする場合、次の条件を入力します:1

   1
6. イベント設定にサブ設定がある場合、必要に応じてサブ設定を構成します。

ダイアログ コントロールのイベントを並べ替える

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ユーザー インターフェイス コントロールに構成されたイベントは、実行時にコントロールの "イベント" 設定で表示された順番で起動されます。

タスク ***イベントを移動するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、並べ替えるイベントを含むコントロールをクリックします。設定が、右のペインに表示されます。
4. 設定のグリッドの **[イベント]** 領域で、移動させたいイベントのイベント設定を見つけます。
5. 移動するイベントのイベント設定で、**[イベントの移動]** ボタンをクリックして、**[上に移動]** または **[下に移動]** をクリックします。

イベントをダイアログ コントロールから削除する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***コントロールからイベントを削除するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。

3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、削除するイベントを含むコントロールをクリックします。設定が、右のペインに表示されます。
4. 設定のグリッドの **[イベント]** 領域で、削除したいイベントのイベント設定を見つけます。
5. ”イベント” 設定で、**[イベントの削除]** ボタンをクリックします。

ダイアログ ボタンからカスタム アクションを起動する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ダイログ上のボタンからカスタム アクションを起動するには、コントロールの DoAction イベントを使用します。DoAction イベントをセットアップするには、カスタム アクションを作成する必要があります。

カスタム アクションを作成したら、このアクションを起動する DoAction イベントを設定できます。

タスク ***プッシュボタン コントロールからカスタム アクションを起動するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、変更するダイアログのダイアログ ノードを拡張してから、**[動作]** サブノードをクリックします。
3. 選択されたダイアログのすべてのコントロールの一覧が表示される中央ペインで、カスタム アクションをトリガするプッシュボタン コントロールをクリックします。設定が、右のペインに表示されます。
4. **”イベント”** 設定で、**[新しいイベント]** ボタンをクリックしてから **[DoAction]** をクリックします。InstallShield によって、**”イベント”** 設定の下に新しい行のセットが追加されます。
5. イベントの条件を指定する場合: **”DoAction”** 設定で省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開き、ここでアクションに対して満たさなくてはならない条件を指定できます。
6. **”アクション”** サブ設定で、コントロールと関連付けるアクションの名前を指定します。

インストールが期待通りに動作するのを確認するために、インストールのビルドとテストを行います。

[カスタム アクションとシーケンス] ビューでエンドユーザー ダイアログの順序の表示する (基本の MSI プロジェクト)

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。InstallScript または InstallScript MSI プロジェクトのエンドユーザー ダイアログの順序を表示するには、InstallScript ビュー を使用してスクリプトを調べます。*

[カスタム アクションとシーケンス] ビューで、インストールの実行時にエンドユーザーに表示されるプライマリおよび次のダイアログの順序を表示できます。プライマリ ダイアログは、最上位のダイアログで、[インストール] シーケンスに挿入されています。[カスタム アクションとシーケンス] ビューでプライマリ ダイアログの順序を変更できます。

次のダイアログは、直前のダイアログの [次へ] ボタンのコントロールイベントのアクションにより起動されるダイアログです。次のダイアログは [カスタム アクションとシーケンス] ビューで表示できますが、[インストール] シーケンスの一部ではないので、[カスタム アクションとシーケンス] ビューで順序を変更することはできません。次のダイアログの順番を変更するには、[ダイアログ] ビューでダイアログの下にある [動作] ノードを使用しなくてはなりません。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

初期の [シーケンス] エクスプローラー

[カスタム アクションとシーケンス] ビューを初めて開いたとき、[シーケンス] エクスプローラーで、すべてのプライマリ ダイアログおよびアクションのリストが、インストール中に実行される順序で示されます。多数の [次へ] ダイアログには、プラス記号 (+) が横に付いています。

拡張 [シーケンス] エクスプローラー

インストール シーケンスに挿入されているプライマリダイアログの順序を表示するほか、[シーケンス] エクスプローラーで次のダイアログの順序を表示することもできます。

プライマリ ダイアログの隣りのプラス記号 (+) をクリックすると、リストが展開され、実行時に表示されるプライマリ ダイアログの次のダイアログ (ある場合) の順序が表示されます。特定のダイアログに複数の [次へ] ボタンのコントロールイベントがある場合は、可能性のあるすべての次のダイアログがプライマリダイアログの下に表示されます。

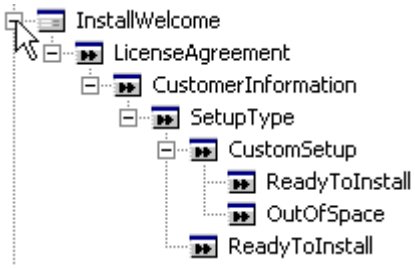


図 3: シーケンス内のダイアログの順番を表示する

次のダイアログの情報パネル

次のダイアログをクリックすると、右側のペインにダイアログの情報が表示されます。右のペインにはまた、ヘルプ情報のほか、動作エディターおよびレイアウト エディターへのハイパーリンクが表示されます。選択されたダイアログのより詳しい情報は、[ダイアログ] ビューで見ることができます。

コンテキストメニューのコマンド

プライマリ ダイアログを右クリックすると、コンテキスト メニューによりコマンドが表示され、シーケンスの編集、ダイアログ シーケンス ビューの更新、ダイアログのレイアウトや動作の編集ができます。次のダイアログを右クリックすると、コンテキスト メニューによりコマンドが表示され、ダイアログのレイアウトや動作を編集できます。

動作の編集

タスク

[カスタム アクションとシーケンス] ビューの [シーケンス] エクスプローラー内から [ダイアログの動作エディター] に移動するには、以下のいずれかの方法を実行します。

- 編集する次のダイアログを右クリックして、コンテキストメニューから**動作の編集** をクリックします。
- 編集するダイアログ名前をクリックします。次に、右のペインの **[アクション項目]** セクションで **[ダイアログの動作を編集する]** をクリックします。

詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

レイアウトの編集

タスク

[カスタム アクションとシーケンス] ビューの [シーケンス] エクスプローラー内から [ダイアログのレイアウトエディター] に移動するには、以下のいずれかの方法を実行します。

- 編集する次のダイアログを右クリックして、コンテキスト メニューから **[レイアウトの編集]** をクリックします。
- 編集するダイアログ名前をクリックします。次に、右のペインの **[アクション項目]** セクションで **[ダイアログのレイアウトを編集する]** をクリックします。インストールが複数の言語で作成されている場合は、リストで適切な言語を選択する必要があります。

詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。

CustomSetup ダイアログのオプション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*
- *トランスフォーム*

CustomSetup ダイアログには、ターゲット システム、インストール中の機能、および Windows Installer のインストール オプションについての情報と緊密に統合された、洗練されたエンドユーザー インターフェイスがあります。これによって、エンドユーザーがインストールを最大限に制御することができます。

このダイアログが提供する多くのオプションおよび情報は、下記で説明されているようにセットアップのデザインで設定された機能のプロパティによって決定されます。

[アドバタイズ] オプション

アドバタイズ機能によって、インストールが最初に実行された後に、必要に応じてファイルをインストールすることができます。CustomSetup ダイアログで、エンドユーザーが機能をクリックすると、**[必要な場合にインストール]** オプションを選択して、その機能を後からインストールするように指定することができます。

ただし、このデフォルトのオプションが表示されるのは、機能の "アドバタイズ" 設定に [アドバタイズを許可する] または [アドバタイズを優先する] を選択した場合のみです。Windows Installer が機能をアドバタイズするオプションを表示しないようにする場合、この設定で [アドバタイズを許可しない] を選択します。

機能を非表示にする

機能の "表示" 設定 を [非表示] に設定すると、CustomSetup ダイアログに機能またはサブ機能は表示されず、エンド ユーザーはインストール オプションを変更できなくなります。

すべてのサブ機能を表示する

機能の "表示" 設定も、以下のオプションを持つダイアログが最初に表示されたときに、サブ機能が展開されるかどうかを制御します。

テーブル 5 ・ 機能の "表示"設定のオプション

オプション	説明
閉じて表示する	デフォルトでサブ機能が閉じた状態で、機能が CustomSetup ダイアログで表示されます。
展開して表示する	デフォルトでサブ機能が展開された状態で、機能が CustomSetup ダイアログで表示されます。

機能のデフォルトのインストール先フォルダーを設定する

CustomSetup ダイアログで機能を選択し、[詳細] ボタンをクリックすることによって、ユーザーは機能がインストールされるフォルダーを表示して編集できます。機能の "インストール先" 設定は、表示されるデフォルト値であり、実際には [機能の詳細] ダイアログにユーザーが新しい値を入力するか新しいパスを参照すると、その新しい値に再び割り当てられます。

機能のインストール場所に影響する要因についての総合的な説明は、「機能のインストール先を設定する」を参照してください。

機能に名前を付ける

"表示名" を設定して、CustomSetup ダイアログに表示される別の名前を指定します。

機能の説明を表示する

機能を選択したときに CustomSetup ダイアログの下部に表示される説明は、機能の"説明" 設定から取得されます。

機能の順序を変更する

CustomSetup ダイアログで機能がエンドユーザーに対して表示される順序を変更することができます。機能が表示される順序は、[セットアップのデザイン] ビューでの順序から取得されます。詳細については、「機能の並べ替え」を参照してください。

機能のインストールを必須にする

機能の"必須" 設定を [はい] に設定すると、エンドユーザーに対して **[使用不能にする]** オプションが表示されず、機能をインストールしなければならなくなります。

ダイアログのテーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

ダイアログ テーマは、エンドユーザー ダイアログに統一感のとれた個性的な印象を与えることができる、あらかじめ定義されている 1 セットのイメージです。

ボタンをクリックするだけで、プロジェクトに提供されているテーマから 1 つ選んで、Setup.exe 初期化ダイアログを含む、プロジェクトで使用されているすべての内部および外部ダイアログに適用することができます。それぞれのダイアログは [ダイアログ] ビュー内から簡単にプレビューすることができ、選択したテーマがどのように見えるかを実際に確認することができます。

メモ: *InstallShield では現在、独自のダイアログ テーマを作成することはできませんが、多くのテーマが用意されています。詳細については、「提供されているテーマと対応するダイアログのサイズ」を参照してください。*

幅の広いダイアログ テーマにおける実行時の要件

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

幅の広いダイアログ テーマを使用する場合、ターゲット システムの画面解像度は最低 640 x 480 ピクセル (大きいフォントが使用されている場合、1024 x 768) でなければなりません。解像度がそれより低い場合、水平スクロールバーが実行時に、幅の広いテーマで表示される各ダイアログの底辺に追加されます。

ダイアログ テーマのプレビュー

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

プロジェクトに異なるテーマを選択すると、そのテーマは、[ダイアログ] ビューにあるすべての内部および外部ダイアログに適用されます。ダイアログの表示に、インストールをビルドして実行する必要はありません。

タスク ***プロジェクトでテーマを選択する前にプレビューするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**InstallWelcome** テーブルをクリックします。
3. **[テーマ]** フォルダーの下に表示されているテーマから 1 つクリックします。

右のペインに、選択されたテーマでサンプル外部ダイアログのスクリーン ショットが表示されます。

タスク ***プロジェクトのダイアログが選択したテーマでどのように見えるかをプレビューするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、それから新しいダイアログ コントロールを追加するダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。

中央のペインに、選択されたテーマが適用された状態でダイアログのプレビューが表示されます。

ダイアログ テーマの選択または変更

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

ダイアログ テーマを使用して、インストールのエンドユーザー ダイアログの外観を変更することができます。1 つのプロジェクトにつき、1 つのテーマを選択することができます。

ヒント: *幅の広いテーマから標準サイズのテーマに変更すると、ダイアログの左端または右端に配置されたコントロールの位置が変わる可能性があります。*

プロジェクトに選択したテーマを変更して、カスタム外部ダイアログも追加する場合、まずテーマを変更してから、カスタム外部ダイアログを追加します。これを逆にすると、ダイアログが適切に表示されない場合があります。

タスク ***プロジェクトに使用されているダイアログ テーマを変更するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**InstallWelcome** テーブルをクリックします。
3. 使用するテーマを右クリックして、**[選択]** をクリックします。

選択されたテーマが、プロジェクトのダイアログに適用されます。さらに [ダイアログ] エクスプローラーでは、選択されたテーマのアイコンに赤いチェック マークが表示されます。

ヒント: *また、[ダイアログ] エクスプローラーで使用するテーマを選択して、ボタン 1 つでテーマを変更することもできます。右のペインで [選択] ボタンをクリックします。*

InstallShield でテーマがどのように適用されるかについての背景事情

ダイアログ テーマを別のテーマへ切り替えると、InstallShield はそのテーマを、特定の幅と高さに一致するプロジェクト内のすべてのダイアログに適用します。したがって、プロジェクトでダイアログの幅または高さを変更したあとで、テーマを変更すると、そのテーマはカスタマイズされた幅または高さを持つダイアログには適用されません。

たとえば、標準幅のテーマを他のテーマに変更した場合、そのテーマは標準の幅と高さ (374 x 266) を持つすべてのダイアログに適用されます。幅の広いテーマを他のテーマに変更した場合、そのテーマは 584 x 274 のすべてのダイアログに適用されます。

外部ダイアログ スタイルを使用するダイアログに関する背景情報

デフォルトでは、すべての外部ダイアログに、ダイアログの左側にイメージが表示される同じスタイルのダイアログが使用されます。InstallWelcome ダイアログと SetupCompleteSuccess ダイアログは、外部ダイアログの例です。

テーマを選択したあと、カスタム外部ダイアログをプロジェクトに追加すると、選択されたテーマが新しいカスタム外部ダイアログに使用されます。ただし、プロジェクトのテーマをあとで変更すると、そのテーマはカスタム外部ダイアログで適切に表示されない場合があります。このタイプの問題を解決するには、「テーマをカスタム外部ダイアログに適用する」をご覧ください。

テーマをカスタム外部ダイアログに適用する

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

プロジェクトに選択したテーマを変更して、カスタム外部ダイアログも追加する場合、まずテーマを変更してから、カスタム外部ダイアログを追加する方法がより簡単です。そうしなかった場合、ダイアログが適切に表示されない場合があります。新しく選択されたテーマの内部ダイアログイメージが部分的にカスタム外部ダイアログに追加されます。

カスタム外部ダイアログをプロジェクトに追加したあと、テーマを変更する場合、カスタム外部ダイアログの名前を、InstallShield Program Files フォルダーにインストールされている適切な .theme ファイルに追加する必要があります。そのあと、テーマをプロジェクトに適用できます。

タスク ***カスタム外部ダイアログをプロジェクトに追加したあとでテーマを変更するには、以下の手順に従います:***

1. InstallShield を閉じます。
2. InstallShield Program Files フォルダーで、ダイアログで使用するテーマの .theme ファイルを見つけます。デフォルト保存先は次の場所です:

   C:¥Program Files¥InstallShield¥2022¥Support¥Themes

3. メモ帳などのテキスト エディターで .theme ファイルを開きます。たとえば、Global テーマを使用する場合、Global.theme ファイルを開きます。
4. ファイルの **<Include>** または **<Exclude>** セクションで、次の行を追加します:

   <Name>*NameOfDialog***</Name>**

   NameOfDialog には、ダイアログの名前が入ります。
5. InstallShield でプロジェクトを開きます。
6. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
7. **[ダイアログ]** エクスプローラーで、**InstallWelcome** テーブルをクリックします。
8. 使用するテーマを右クリックして、**[選択]** をクリックします。

ロゴまたは他のイメージを外部ダイアログを追加する

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

外部ダイアログは、インストールの始めと終わりに表示されるダイアログで、通常 Welcome ダイアログと Completion ダイアログがこれにあたります。一部のテーマの外部ダイアログには、会社または製品のロゴを配置することができます。たとえば、Monitor テーマと Circles テーマの外部ダイアログには、ダイアログの左側に空白部分があります。これらのダイアログにビットマップ コントロールを追加して、エンドユーザー インターフェイスをカスタマイズすることができます。

タスク ***外部ダイアログにイメージを追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、イメージを追加するダイアログのダイアログ アイテムを展開します。
3. 展開したダイアログ アイテムの下にある言語をクリックします。中央のペインにあるダイアログ エディターに、選択した言語のダイアログが表示されます。
4. **[コントロール]** ツールバー上で、**[ビットマップ]** ボタンをクリックします。
5. ダイアログ エディターで、ビットマップ コントロールが開始する箇所をクリックし、左マウス ボタンを押したまま、コントロールが終わる箇所までマウスをドラッグします。左マウス ボタンを解放します。
6. 右のペインでビットマップ コントロールのプロパティを構成します。

   "ファイル名" プロパティをクリックして、省略記号ボタン (...) をクリックして、使用するビットマップ ファイルを参照します。

必要に応じて、この手続きをプロジェクトのそれぞれの言語および外部ダイアログについて繰り返します。以下は、すべての新しい基本の MSI プロジェクトでデフォルトで提供されている外部ダイアログです:

- AdminWelcome
- InstallWelcome

- MaintenanceWelcome
- PatchWelcome
- SetupCompleteError
- SetupCompleteSuccess
- SetupInitialization
- SetupInterrupted
- SetupResume
- SplashBitmap

提供されているテーマと対応するダイアログのサイズ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *一部のテーマは、InstallShield の Premier Edition と Professional Edition の両方で提供されていますが、Premier Edition のみで提供されているものもあります。*

一部のテーマのダイアログのサイズは、標準の幅と高さ (374 x 266) に設定されています。幅の広いテーマのダイアログ サイズは 584 x 274 に設定されています。次のテーブルは、各ダイアログのサイズとそのテーマを含む InstallShield エディションを示します。

テーブル 6 · InstallShield のダイアログ テーマ

名前	InstallShield エディション	ダイアログの幅	ダイアログの高さ
Circles テーマ (大)	Premier および Professional	584	274
Classic テーマ	Premier および Professional	374	266
Cooperation テーマ (大)	Premier	584	274
Filmstripテーマ (大)	Premier	584	274
Global テーマ	Premier および Professional	374	266
InstallShield Blue テーマ	Premier および Professional	374	266

テーブル 6 • InstallShield のダイアログ テーマ (続き.)

名前	InstallShield エディション	ダイアログの幅	ダイアログの高さ
InstallShield Blue テーマ (大)	Premier および Professional	584	274
InstallShield Silver テーマ	Premier	374	266
Monitor テーマ	Premier	374	266
Pastel Wheat テーマ	Premier	374	266
Theater テーマ (大)	Premier	584	274

Circles テーマ (大)

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、Circle テーマ (大) のサンプル外部ダイアログと内部ダイアログです。

図 4: Circle テーマ (大) のサンプル外部ダイアログ

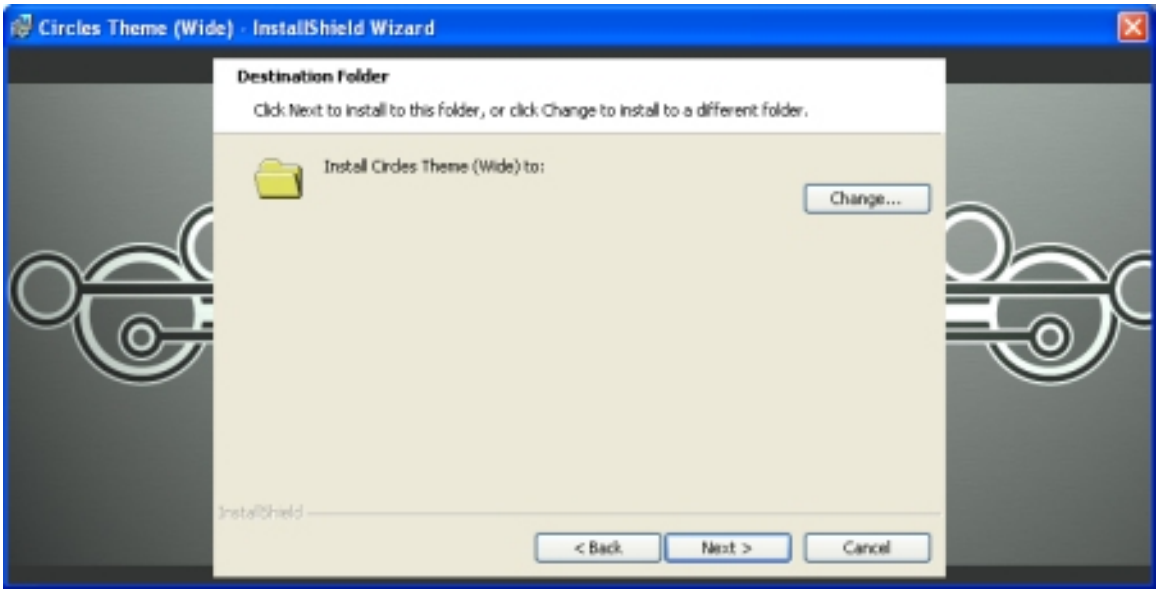


図 5: Circle テーマ (大) のサンプル内部ダイアログ

外部ダイアログに会社または製品ロゴを追加する方法については、「ロゴまたは他のイメージを外部ダイアログを追加する」をご覧ください。

Classic テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、Classic テーマのサンプル外部ダイアログと内部ダイアログです。

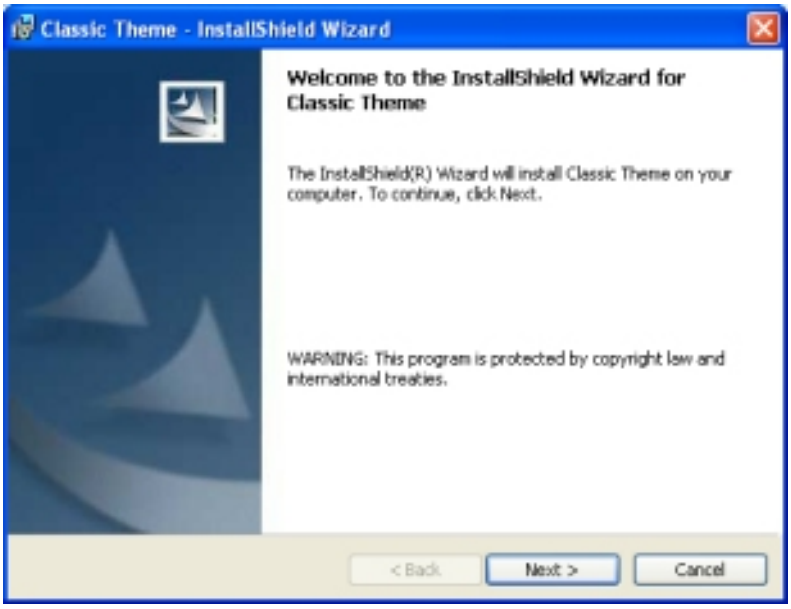


図 6: Classic テーマのサンプル外部ダイアログ

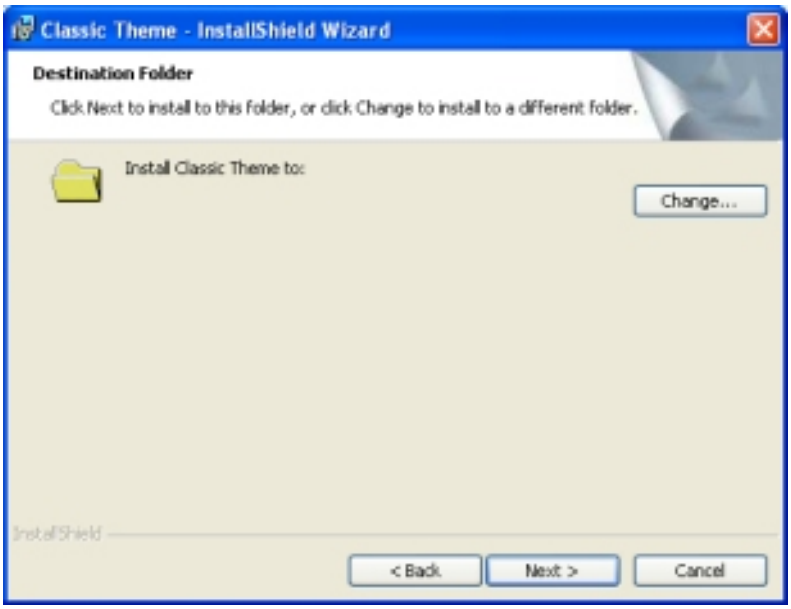


図 7: Classic テーマのサンプル内部ダイアログ

Cooperation テーマ (大)

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Cooperation テーマ (大) のサンプル外部ダイアログと内部ダイアログです。

図 8: Cooperation テーマ (大) のサンプル外部ダイアログ

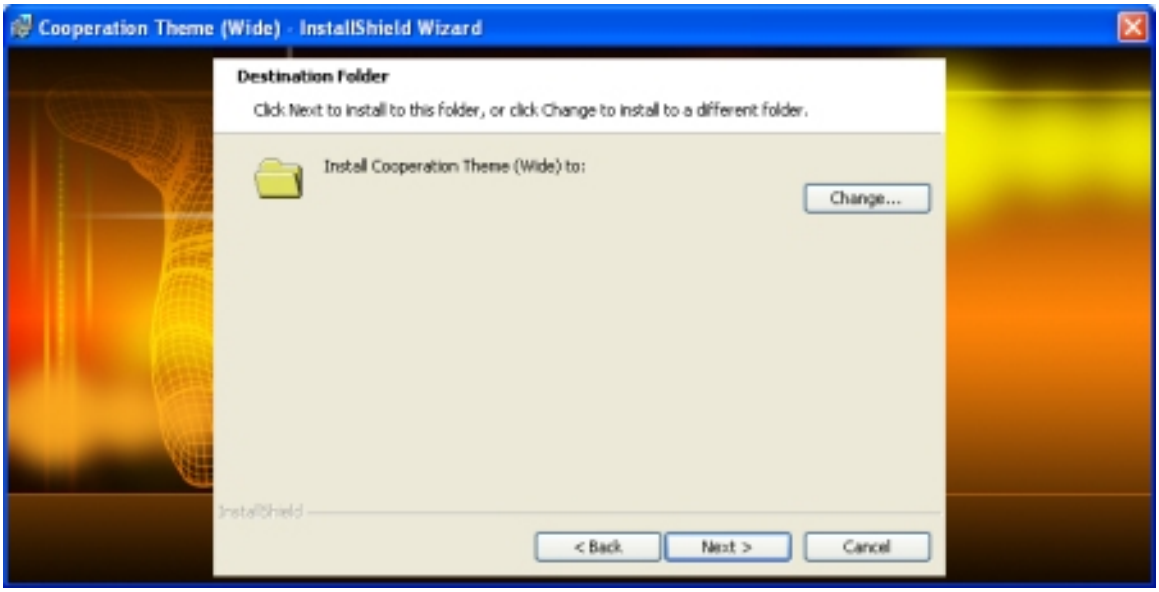


図 9: Cooperation テーマ (大) のサンプル内部ダイアログ

Filmstripテーマ (大)

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Filmstrip テーマ (大) のサンプル外部ダイアログと内部ダイアログです。

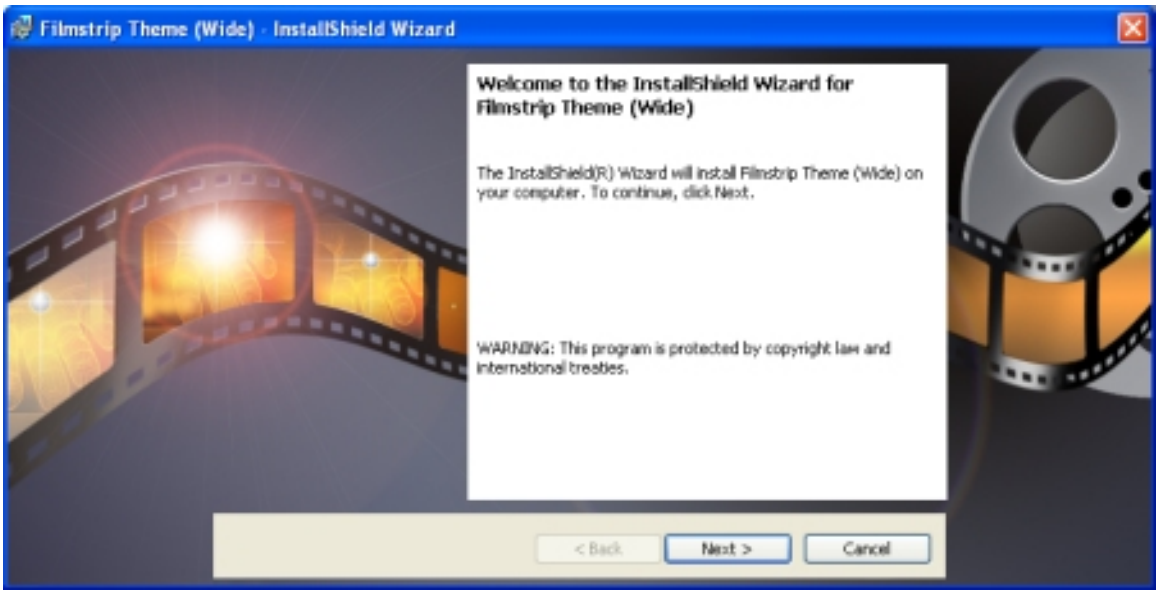


図 10: Filmstrip テーマ (大) のサンプル外部ダイアログ

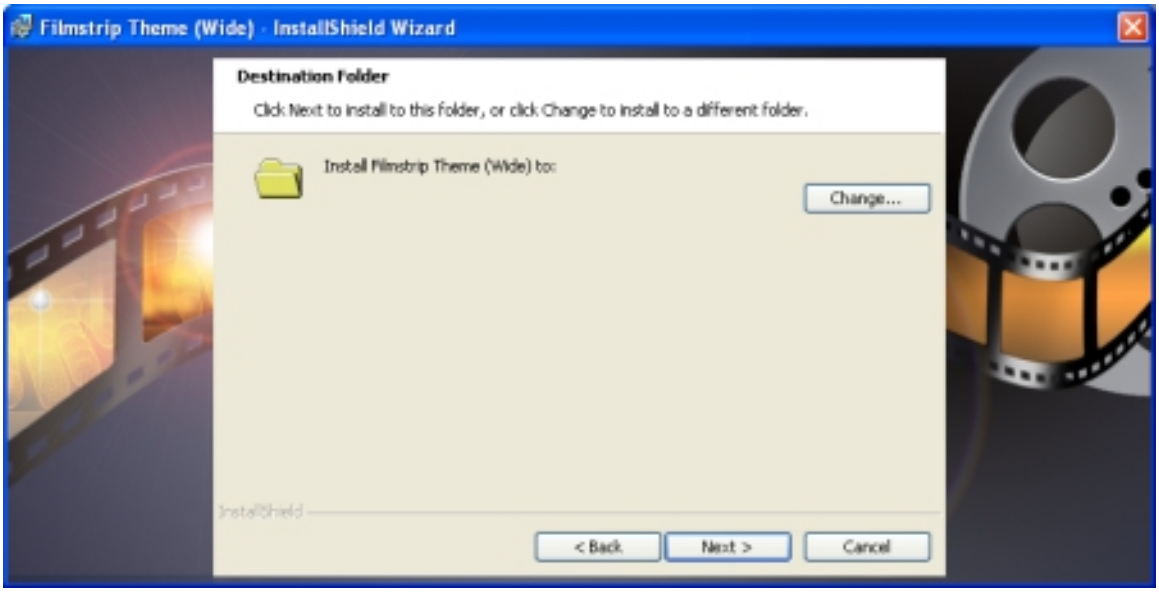


図 11: Filmstrip テーマ (大) のサンプル内部ダイアログ

Global テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、Global テーマのサンプル外部ダイアログと内部ダイアログです。

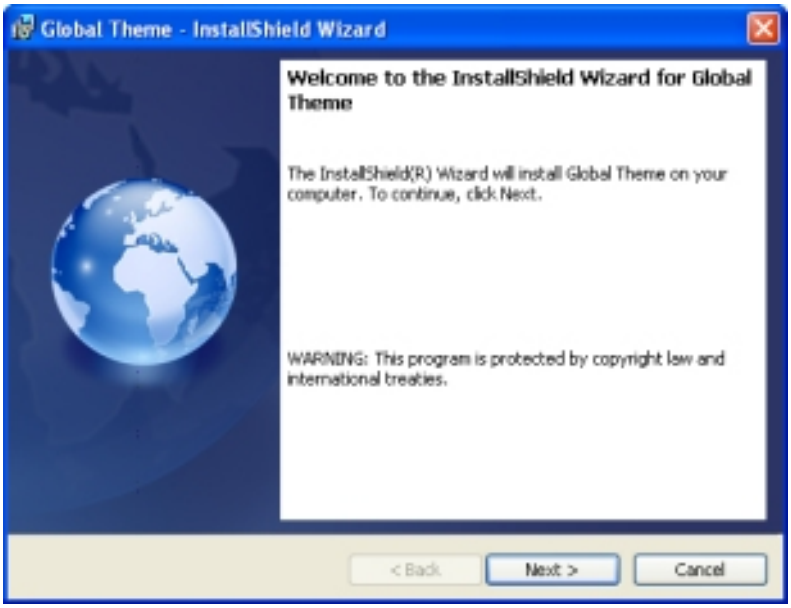


図 12: Global テーマのサンプル外部ダイアログ

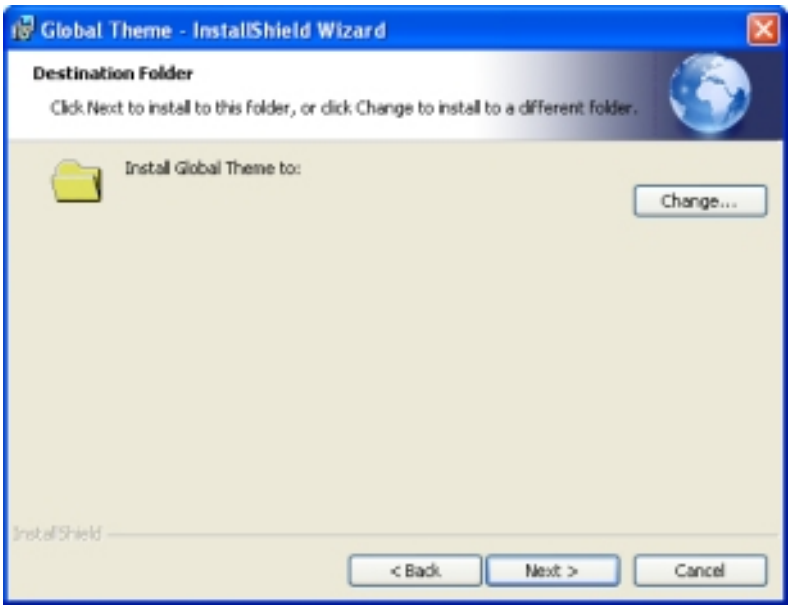


図 13: Global テーマのサンプル内部ダイアログ

InstallShield Blue テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、InstallShield Blue テーマのサンプル外部ダイアログと内部ダイアログです。

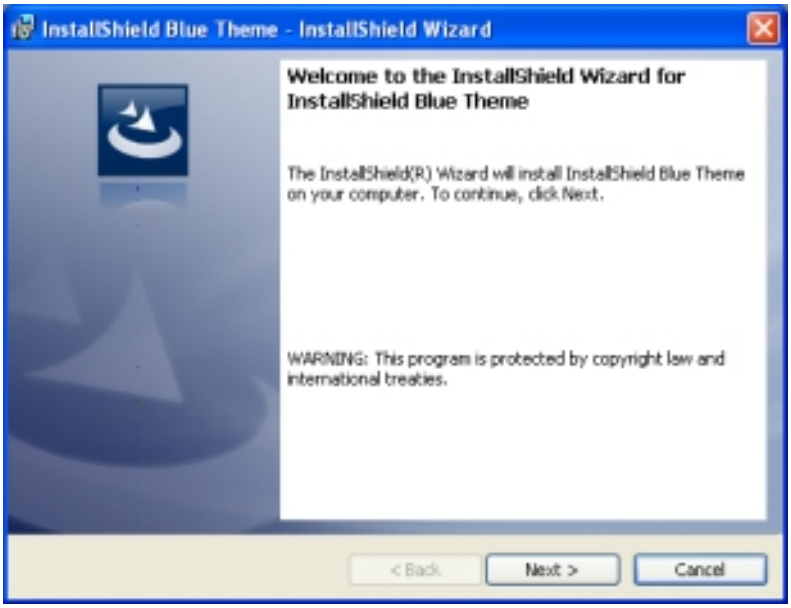


図 14: InstallShield Blue テーマのサンプル外部ダイアログ

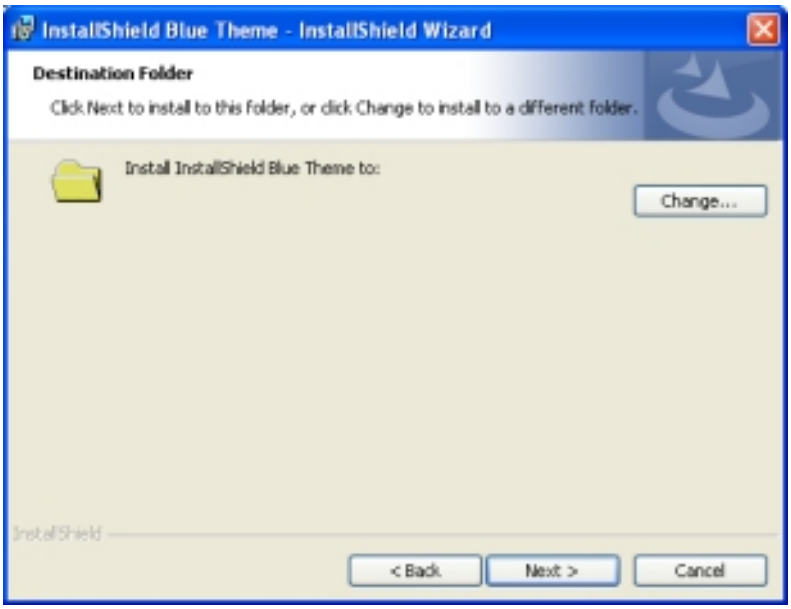


図 15: InstallShield Blue テーマのサンプル内部ダイアログ

InstallShield Blue テーマ (大)

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

以下は、InstallShield Blue テーマのサンプル外部ダイアログと内部ダイアログです。

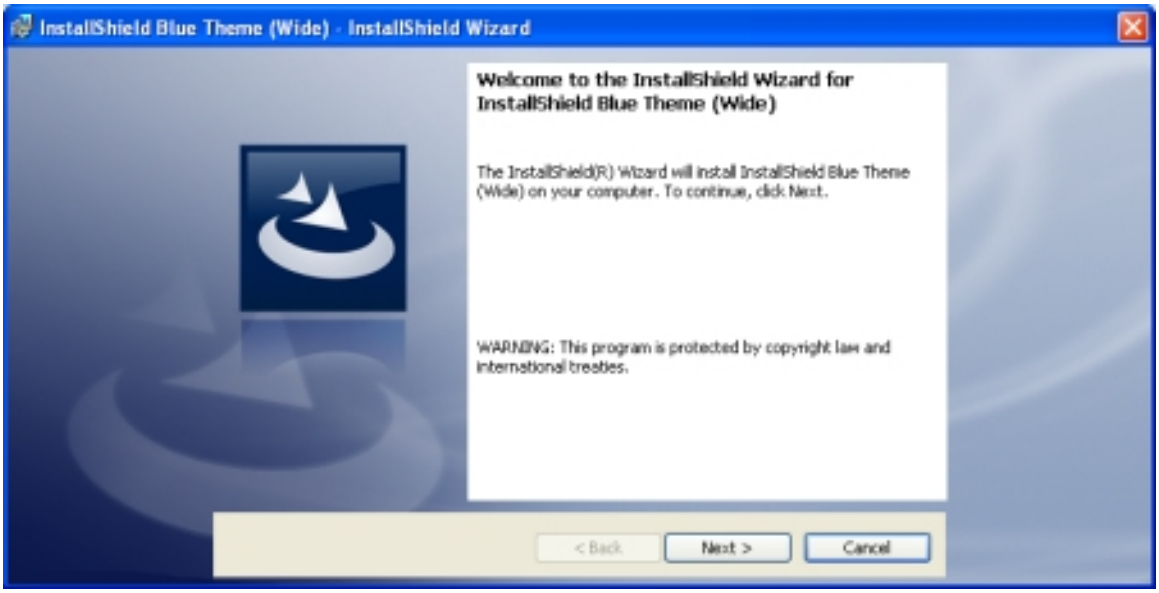


図 16: InstallShield Blue テーマ (大) のサンプル外部ダイアログ

図 17: InstallShield Blue テーマ (大) のサンプル内部ダイアログ

InstallShield Silver テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、InstallShield Silver テーマのサンプル外部ダイアログと内部ダイアログです。

図 18: InstallShield Silver テーマのサンプル外部ダイアログ

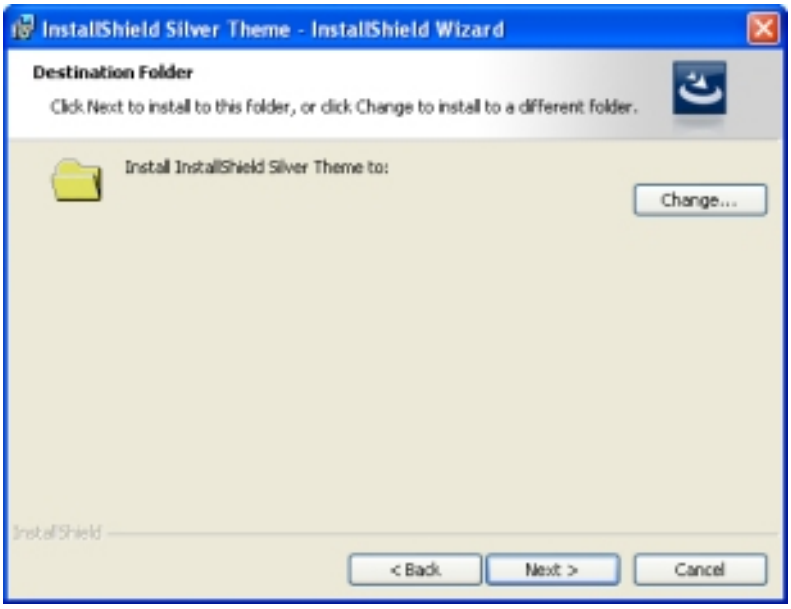


図 19: InstallShield Silver テーマのサンプル内部ダイアログ

Monitor テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Monitor テーマのサンプル外部ダイアログと内部ダイアログです。

図 20: Monitor テーマのサンプル外部ダイアログ

図 21: Monitor テーマのサンプル内部ダイアログ

外部ダイアログに会社または製品ロゴを追加する方法については、「ロゴまたは他のイメージを外部ダイアログを追加する」をご覧ください。

Pastel Wheat テーマ

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Pastel Wheat テーマのサンプル外部ダイアログと内部ダイアログです。

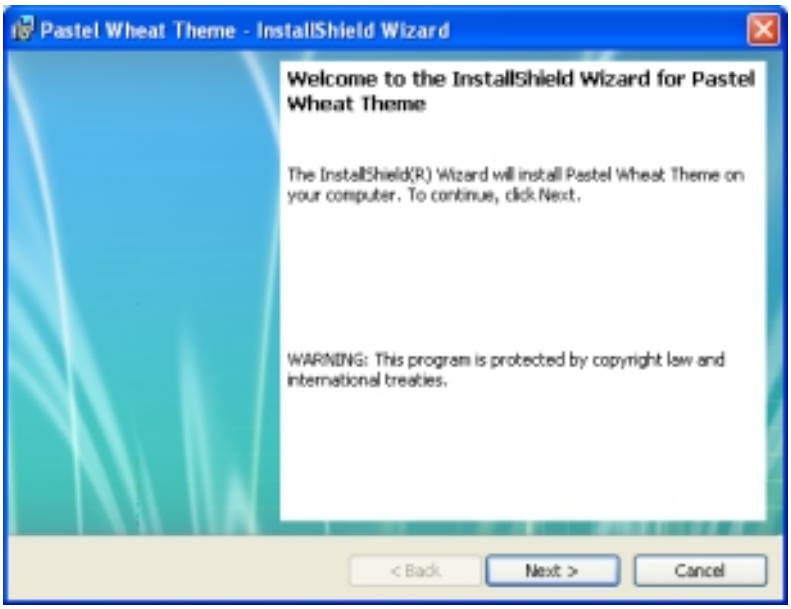


図 22: Pastel Wheat テーマのサンプル外部ダイアログ

図 23: Pastel Wheat テーマのサンプル内部ダイアログ

Theater テーマ（大）

プロジェクト: *ダイアログ テーマは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

エディション: *このテーマは、InstallShield の Premier Edition で提供されています。*

以下は、Theater テーマ（大）のサンプル外部ダイアログと内部ダイアログです。

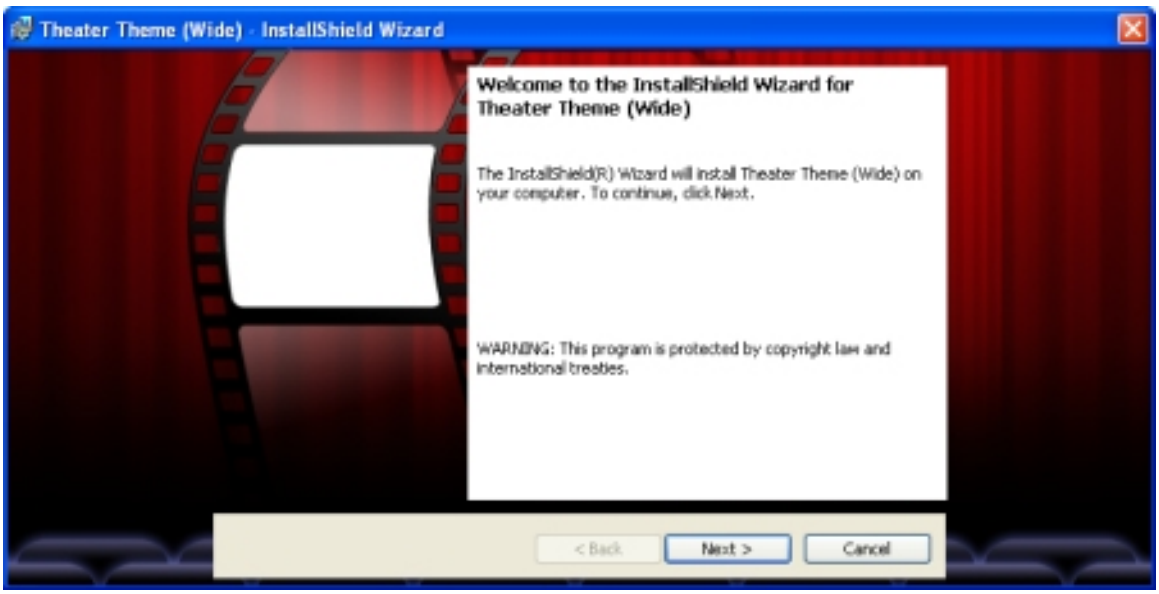


図 24: Theater テーマ（大）のサンプル外部ダイアログ

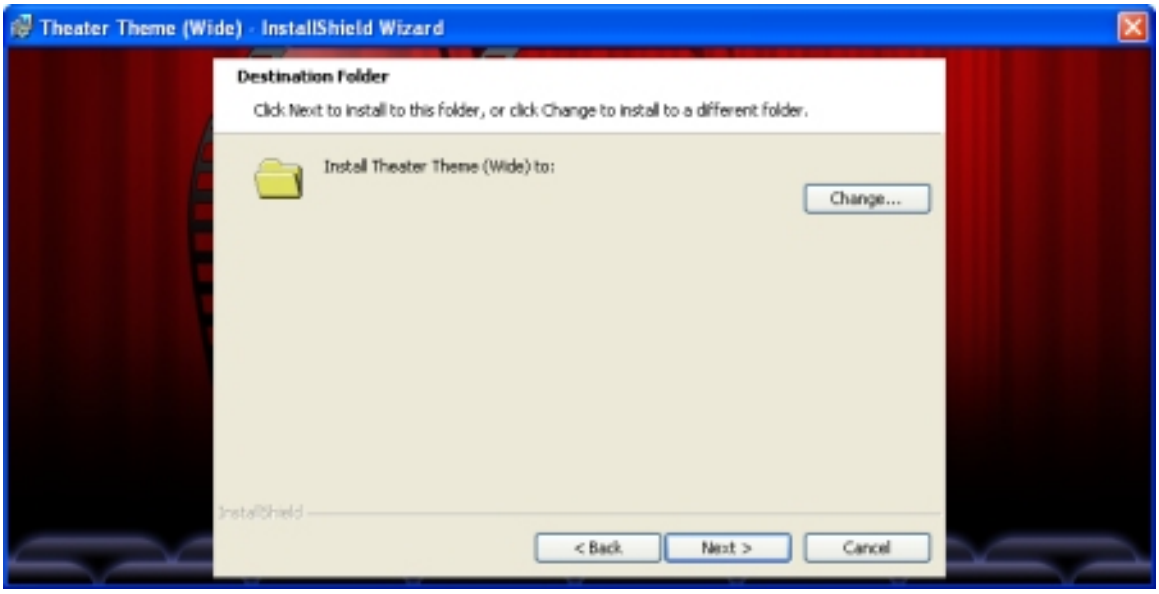


図 25: Theater テーマ (大) のサンプル内部ダイアログ

外部ダイアログに会社または製品ロゴを追加する方法については、「ロゴまたは他のイメージを外部ダイアログを追加する」をご覧ください。

右から左に記述される言語のダイアログ サポート

エディション: *InstallShield Premier Edition には、右から左に読まれるアラビア語とヘブライ語のダイアログ サポートが含まれます。*

プロジェクト: *次のプロジェクト タイプに、右から左に記述される言語のダイアログ サポートが含まれます:*

- *基本の MSI*
- *マージ モジュール*

InstallShield には、右から左に記述される言語であるアラビア語 (サウジアラビア) とヘブライ語のサポートが含まれています。デフォルトのエンド ユーザー ダイアログ文字列のすべてが、これらの言語で利用できます。

これらの言語は右から左に読まれるため、InstallShield にはアラビア語とヘブライ語ダイアログのミラー サポートが含まれます。これによって、アラビア語とヘブライ語のダイアログには、右から左方向へのレイアウトが使用されます。たとえば、英語やその他の左から右に読まれる言語のダイアログで右側にあるボタンは、右から左に読まれる言語のダイアログでは左側に移動されます。これは InstallShield の [ダイアログ] ビューの [ダイアログ エディター] ペインで処理され、実行時にも同様に処理されます。

適切な場合、ビルトイン ダイアログ テーマで表示されるダイアログ イメージの逆向きバージョンが表示されます。逆向きバージョンは、イメージ ファイル名の .bmp または .jpg 部分の直前に *_mirror* が付いています。たとえば、左から右に読み書きされる言語のイメージ名が **banner.jpg** の場合、右から左に読み書きされる言語の対応するイメージの名前は **banner_mirror.jpg** となります。これらの 2 つのファイルは同じフォルダーに配置され、ダイアロ

グの右から左に読まれる言語バージョンには、自動的に banner_mirror.jpg が使用されます。イメージを逆向きにすべきではない場合、そのイメージの *_mirror.jpg または *_mirror.bmp バージョンは含まれず、ダイアログの右から左方向バージョンは、イメージのミラー イメージを表示しません。

ヒント: *ランタイム ダイアログにカスタム イメージを使用する場合で、プロジェクトが右から左に記述される言語のサポートを含むときは、これらの右から左に記述される言語用にミラー イメージ バージョンを作成する必要がある可能性があります。[ダイアログ] ビューで、ダイアログの右から左方向レイアウトをプレビューして、カスタム イメージがどのように表示されるかを確認できます。適切な場合、カスタム イメージの左から右方向バージョンが含まれるフォルダーに、イメージの *_mirror.jpg または *_mirror.bmp バージョンを追加します。*

[ファイルを開く] ダイアログを起動する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield は、基本の MSI インストールのダイアログの 1 つから [ファイルを開く] ダイアログを起動するサポートを含みます。エンド ユーザーがダイアログの 1 つから [参照] ボタンをクリックすると、[ファイルを開く] ダイアログが起動します。[ファイルを開く] ダイアログを使って、エンド ユーザーはファイルを参照できます。エンド ユーザーがファイルを選択して [開く] ボタンをクリックすると、[ファイルを開く] ダイアログが閉じて、インストールがダイアログの編集フィールドにその完全パスとファイル名を書き込みます。インストールはまた、IS_BROWSE_FILEBROWSED プロパティの値を、エンド ユーザーが選択したファイルのパスとファイル名に設定します。

インストールに [ファイルを開く] ダイアログのサポートを組み込む場合、次の機能を指定するいくつかのプロパティを定義できます。

- このダイアログで表示されるデフォルト パスを指定することができます。
- [ファイルを開く] ダイアログの **[ファイルの種類]** ドロップダウン リストに表示する文字列を指定できます。
- エンド ユーザーがファイルを選択するためにフォルダーを参照するときに表示される、ファイルの拡張子を指定できます。その他のファイル拡張子を持つファイルは表示されず、またそれらを開くこともできません。
- [ファイルを開く] ダイアログが使用するデフォルトのファイル拡張子を指定できます。エンド ユーザーが拡張子を入力しなかった場合、[ファイルを開く] ダイアログがファイル名にこのデフォルト拡張子を追加します。

以下は、ダイアログの要件です:

- [ファイルを開く] ダイアログで、エンド ユーザーが新しいファイルを作成することはできません。つまり、ファイルが既存しないとき、エンド ユーザーが [ファイルを開く] ダイアログで新しいファイル名を手作業で入力することはできません。
- [ファイルを開く] ダイアログが起動するダイアログに複数の編集フィールド コントロールが含まれている場合、ファイルへの完全パスを含む編集フィールド コントロールの Tab Stop プロパティは最小値を持たなくてはなりません。

タスク　　エンド ユーザーダイアログに [ファイルを開く] ダイアログ機能を追加するには、以下の手順を実行します。

1. **FileBrowse** という名前の新しい .dll カスタム アクションをプロジェクトに追加します:
   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
   b. **[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが、*NewCustomActionN* という名前で追加されます (ここで *N* は連続番号です)。
   c. カスタム アクションの名前を **FileBrowse** に変更します。
   d. 右のペインで、このカスタム アクションについて次の設定を構成します:
      - DLL Filename: <ISRedistPlatformDependentFolder>¥FileBrowse.dll
      - 関数名: FileBrowse
      - 戻り値の処理: 同期 (終了コードを無視)
      - スクリプト内実行: 即時実行
      - 実行スケジュール: Always execute

      他のすべての設定については、デフォルト値をそのまま使用します。"MSI タイプ番号" 設定の値は **65** にします。
2. [ファイルを開く] ダイアログを起動するダイアログを作成または編集して、その動作とレイアウトを構成します:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** アイテムを展開します。
   c. 既存のダイアログを選択するか、新しいダイアログを作成します。

      既存のダイアログそ選択して、そのダイアログに複数の編集フィールド コントロールが含まれている場合、ファイルへの完全パスを含む編集フィールド コントロールの **Tab Stop** プロパティは最小値を持たなくてならない点にご注意ください。
   d. このダイアログの下にある、レイアウトを構成する言語をクリックします。
   e. エンド ユーザーが実行時に選択する、完全パスとファイル名を含む編集フィールド コントロールを追加します。コントロールに関連付けられたプロパティ名がプロンプトされたとき、**IS_BROWSE_FILEBROWSED** を入力します。
   f. 編集フィールド コントロールのとなりに、プッシュボタン コントロールを追加します。このボタンが [ファイルを開く] ダイアログを起動します。
   g. プッシュボタン コントロールを選択してから、右側のグリッドで必要に応じてそのプロパティを編集します。たとえば、ボタンで表示するテキストを指定するには、**Text** プロパティの値を追加します。
   h. **[ダイアログ]** エクスプローラーで、構成するダイアログの下にある **[動作]** アイテムをクリックします。
   i. ダイアログ コントロールがリストされる中央のペインで、作成したプッシュボタン コントロールをクリックします。設定が、右のペインに表示されます。

j. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[DoAction]** をクリックします。
InstallShield によって、**"イベント"** 設定の下に新しい行のセットが追加されます。

k. **"DoAction"** 設定に、次のように入力します:

1

l. **"アクション"** サブ設定で、次のアクション名を指定します:

FileBrowse

m. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[SetProperty]** をクリックします。
InstallShield によって、**"イベント"** 設定の下に新しい行が追加されます。

n. **"SetProperty"** 設定に、次のように入力します:

1

o. **"プロパティ"** サブ設定で、次のアクション名を指定します:

IS_BROWSE_FILEBROWSED

p. **"値"** サブ設定で、次のアクション名を指定します:

[IS_BROWSE_FILEBROWSED]

***メモ・** SetProperty イベントは、ダイアログの "編集" フィールド コントロールに表示される MSI プロパティ値を更新するため、ファイルの参照ウィンドウでファイルが選択された後に表示される MSI プロパティ値が更新されます。*

3. [ファイルを開く] ダイアログの動作を指定するいくつかのプロパティと、[ファイルを開く] ダイアログを起動するダイアログを構成します:

a. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。

b. IS_BROWSE_FILEBROWSED プロパティを探します。デフォルト値は 0 です。以下のいずれかを実行します。

- エンド ユーザーが最初に表示したときに、ダイアログの編集フィールド コントロールを空白のままにするには、IS_BROWSE_FILEBROWSED プロパティが含まれる行を右クリックしてから、**[プロパティの削除** を選択します。
- 編集フィールド コントロールにデフォルトのパスとファイル名を表示するには、IS_BROWSE_FILEBROWSED プロパティの値をパスとファイル名に変更します。

***メモ・** このプロパティの値を手動で変更しなかったとき、または削除した場合、[ファイルを開く] ダイアログを起動するダイアログの編集フィールド コントロールのデフォルト値は 0 に設定されます。*

c. オプションとして、IS_BROWSE_FILEEXT と呼ばれるプロパティを追加し、エンド ユーザーがファイルを選択するためにフォルダーを参照する際に表示されるファイルの拡張子を識別するフィルター文字列に、その値を設定することもできます。その他のファイル拡張子を持つファイルは表示されず、またそれらを開くこともできません。

フィルター文字列には、有効なファイル名文字と、ワイルドカードとしてアスタリスク (*) の組み合わせが可能です。

複数のファイル拡張子を指定する場合は、セミコロンで区切ります。スペースは使用できません。たとえば、エンド ユーザーが .exe と .dll ファイルを選択できるようにするには、IS_BROWSE_FILEEXT プロパティの値として、以下の文字列を入力します:

***.exe;*.dll**

この例では、[ファイルを開く] ダイアログでエンドユーザーは .exe と .dll ファイルを選択できます。その他のファイル タイプは表示されません。

このプロパティを設定しなかった場合、[ファイルを開く] ダイアログでエンド ユーザーは任意のファイル タイプを選択できます。

d. オプションとして、IS_BROWSE_FILETYPE と呼ばれるプロパティを追加し、その値を [ファイルを開く] ダイアログの **[ファイルの種類]** ドロップダウン リストに表示する文字列に設定します。ドロップダウン リストに表示可能なオプションは 1 つだけです。

たとえば、エンド ユーザーが .tt または .doc ファイルを選択できるようにするには、IS_BROWSE_FILETYPE プロパティの値として次の文字列を入力します:

テキスト文書 (*.txt); Word 文書 (*.doc)

このプロパティを設定しなかった場合、[ファイルを開く] ダイアログの **[ファイルの種類]** ドロップダウン リストは空白です。

e. オプションとして、IS_BROWSE_DEFAULTEXTENSION と呼ばれるプロパティを追加して、その値を [ファイルを開く] ダイアログが使用するデフォルトのファイル拡張子に設定します。エンド ユーザーが拡張子を入力しなかった場合、[ファイルを開く] ダイアログがファイル名にこのデフォルト拡張子を追加します。たとえば、デフォルトのファイルカクチョウシとして .exe を使用するには、IS_BROWSE_DEFAULTEXTENSION プロパティの値として次の文字列を入力します:

exe

f. オプションとして、IS_BROWSE_INITIALDIR と呼ばれるプロパティを追加して、その値を [ファイルを開く] ダイアログが使用するデフォルト パスに設定します。たとえば、**C:¥Program Files¥My Product** を使用するには、IS_BROWSE_INITIALDIR プロパティの値として次の文字列を入力します:

C:¥Program Files¥My Product

実行時、エンド ユーザーが新しいプッシュボタン コントロールをクリックしたとき、[ファイルを開く] ダイアログが開きます。エンド ユーザーはファイルを参照して選択できます。インストールは IS_BROWSE_FILEBROWSED プロパティの値を、エンド ユーザーが選択したファイルのパスとファイル名に設定し、[ファイルを開く] ダイアログを起動したダイアログの編集フィールド コントロールにそのパスとファイル名を表示します。

LicenseAgreement ダイアログでエンドユーザーが EULA を最初から最後までスクロールするのを必須にする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield では、エンドユーザーが次のいずれかの方法でスクロール可能な EULA コントロール内にあるエンド ユーザー使用許諾契約 (EULA) テキストの終わりに到達するまで、**LicenseAgreement** ダイアログ上の **[次へ]** ボタンを無効にすることができます。

- スクロール バーを使用する。
- スクロール可能な EULA コントロールにフォーカスがある状態でマウスホイールを使用する。

- スクロール可能な EULA コントロールにフォーカスがあるとき、PAGE DOWN を押す。
- スクロール可能な EULA コントロールにフォーカスがあるとき、CTRL+PAGE DOWN を押す。
- スクロール可能な EULA コントロールにフォーカスがあるとき、DOWN ARROW を押す。
- スクロールを右クリックして、[最下部] をクリックする。

ユーザーはまた、**[次へ]** ボタンを有効にするためには、**[ソフトウェア使用許諾契約に同意します]** オプションを選択する必要があります。

エンドユーザー が EURA テキストの最後まで読むことも必須にする場合、次のタスクを実行します。

タスク ***エンドユーザーがスクロール可能な EULA コントロール内の EURA テキストの最後まで到達することを必須にする場合、以下の手順に従います:***

1. **WatchScroll** という名前の新しい .dll カスタム アクションをプロジェクトに追加します:
   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。
   b. **[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが、*NewCustomActionN* という名前で追加されます (ここで *N* は連続番号です)。
   c. カスタム アクションの名前を **WatchScroll** に変更します。
   d. 右のペインで、このカスタム アクションについて次の設定を構成します:
      - DLL Filename: <ISRedistPlatformDependentFolder>¥EulaScrollWatcher.dll
      - 関数名: WatchScroll
      - 戻り値の処理: 非同期 (終了コードを待機)
      - スクリプト内実行: 即時実行
      - 実行スケジュール: Always execute

      他のすべての設定については、デフォルト値をそのまま使用します。"MSI タイプ番号" 設定の値は **129** にします。
2. 適切なカスタム アクションが起動されるように LicenseAgreement ダイアログを編集します:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
   b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、**LicenseAgreement** アイテムを展開し、**[動作]** をクリックします。
   c. LicenseAgreement コントロールの一覧がある中央のペインで、**Memo** という名前の ScrollableText コントロールをクリックします。このコントロールが EULA のテキストを含むコントロールです。設定が、右のペインに表示されます。
   d. **"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[DoAction]** をクリックします。InstallShield によって、**"イベント"** 設定の下に新しい行のセットが追加されます。
   e. **"DoAction"** 設定に、次のように入力します:

      Not LicenseViewed AND Not ISLicenseWatching
   f. **"アクション"** サブ設定で、次のアクション名を指定します:

WatchScroll

g. LicenseAgreement コントロールの一覧がある中央のペインで、**Next** という名前の PushButton コントロールをクリックします。

h. **"条件"** 設定で、**[新しい条件]** ボタンをクリックしてから **[Disable]** をクリックします。**"Disable"** 設定が追加されます。

i. **"Disable"** 設定に、次のように入力します:

AgreeToLicense ◇ "はい" OR Not LicenseViewed

j. **"条件"** 設定で、**[新しい条件]** ボタンをクリックしてから **[Enable]** をクリックします。**"Enable"** 設定が追加されます。

k. **"Enable"** 設定に、次のように入力します:

AgreeToLicense = "はい" AND LicenseViewed

3. これは任意です。エンドユーザーが EULA テキストの終わりに到達した後に、**[使用許諾契約オプションの条件に同意する]** オプションを自動的に選択する場合は、上記の手順に加えて次の手順を実行します (ステップ 2)。

a. 前述のステップ 2 に引き続き、**"イベント"** 設定で、**[新しいイベント]** ボタンをクリックしてから **[SetProperty]** をクリックします。InstallShield によって、**"イベント"** 設定の下に新しい行のセットが追加されます。

b. **"SetProperty"** 設定に、次のように入力します:

LicenseViewed

c. **"Property"** サブ設定に、次のように入力します:

AgreeToLicense

d. **"Value"** サブ設定に、次のように入力します:

Yes

実行時に、EULA コントロールがインストールによって非同期カスタム アクションで監視されます。このカスタム アクションの実行中、ISLicenseWatching プロパティが設定されます。カスタム アクションの完了すると、ISLicenseWatching プロパティは削除されます。これにより、エンドユーザーが EULA のテキストをスクロールするときに砂時計フリッカーが表示されるのを防ぐことができます。エンドユーザーが EULA テキストの最後まで到達すると、

- インストールによって LicenseViewed プロパティが設定されます。
- ステップ 3 で説明されているオプションの構成が適用されている場合、AgreeToLicense プロパティは [Yes] に自動的に設定され、**[使用許諾契約オプションの条件に同意する]** オプションが選択されます。
- [**次へ**] ボタンは、**[次へ]** ボタンの条件に応じて有効になります。

[次へ] ボタンは Control テーブルに格納されているテキストで検索されます。したがって、コントロールの名前が **Next** であるかぎり、前述の手順をすべての言語トランスフォームで採用することができます。

[印刷] ボタンをダイアログに追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield X 以降で作成された基本の MSI プロジェクトでは、LicenseAgreement ダイアログに [印刷] ボタンが含まれます。このボタンを利用して、エンドユーザーがダイアログのスクロール テキスト コントロールの内容を印刷することを可能にします。このボタンの イベント はカスタム アクション ISPrint を実行します。このアクションは基本の MSI プロジェクトに含まれます。次は [印刷] ボタンを別のダイアログに追加する、および InstallShield DevStudio 9.0 以前で作成した 既存プロジェクトに追加する 手順です。

[印刷] ボタンを別のダイアログに追加する

カスタム アクション IS Print が LicenseAgreement 以外のダイアログで適切に作動させるためには、ユーザー定義の Windows Installer プロパティ IS_PRINT_DIALOG の値を ダイアログの名前に設定しなくてはなりません。(IS_PRINT_DIALOG が既存するプロパティではない場合、ISPrint は LicenseAgreement ダイアログ ボックスのスクロール テキスト コントロールの内容を印刷します。)

タスク ***[印刷] ボタンを別のダイアログへ追加するには、以下の手順を実行します。***

1. ダイアログ ボックスでボタンコントロールを作成し、オプションでその **"テキスト"** プロパティを**[印刷(&P)]** に設定します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。
2. [印刷] ボタンに DoAction イベントを追加し、イベントの **"アクション"** サブ設定で ISPrint を選択します。詳細については、「基本の MSI ダイアログでコントロール イベントをトリガーする」を参照してください。
3. ダイアログ ボックスおよびその前後のダイアログの [戻る] および [次へ] ボタンのイベントから IS_PRINT_DIALOG の値を変更します。
   a. どのダイアログが [印刷] ボタンを追加するダイアログの前に表示されるのかを判別します。これは、NewDialog イベントの引数で確認して、ダイアログの [戻る] ボタンを探すか、[カスタム アクションとシーケンス] ビューを開いて、ダイアログの順序を表示することで確認することができます。
   b. SetProperty イベント を追加します。**SetProperty** 設定で、イベント条件として次を指定します:

      1

   c. **[プロパティ名]** ボックスで、次のように入力します。

      IS_PRINT_DIALOG

   d. "値" 設定で、[印刷] ボタンを追加するダイアログの名前を指定します。
   e. [印刷] ボタンを追加するダイアログのあとにも [印刷] ボタンを含むダイアログがある場合、次を実行します。
      i. どのダイアログが [印刷] ボタンを追加するダイアログの後に表示されるのかを確認します。これは、NewDialog イベントの引数を確認して、ダイアログの [次へ] ボタンを探すか、[カスタム アクションとシーケンス] ビューを開いてダイアログの順序を表示することで確認することができます。
      ii. SetProperty イベントを次のダイアログの [戻る] ボタンに追加して、その **"プロパティ名"** 設定を **IS_PRINT_DIALOG**、**"値"** 設定を [印刷] ボタンを追加するダイアログの名前に設定します。

次のダイアログが [印刷] ボタンを含まない場合、または次のダイアログ ボックスが LicenseAgreement ダイアログの場合は、[印刷] ボタンを追加するダイアログの [次へ] ボタンに SetProperty イベントを追加して、その **"プロパティ名"** 設定を **IS_PRINT_DIALOG**、"値" 設定を **LicenseAgreement** に設定します。

f. [印刷] ボタンを追加するダイアログの前に表示される任意のダイアログに [印刷] ボタンが含まれているとき、また直前のダイアログに [印刷] ボタンがない、あるいはそれが LicenseAgreement ダイアログである場合、[印刷] ボタンを追加するダイログの [戻る] ボタンに SetProperty イベントを追加し、イベントの **"プロパティ名"** 設定を **IS_PRINT_DIALOG**、**"値"** 設定を **LicenseAgreement** に設定します。

既存プロジェクトへ [印刷] ボタンを追加する

タスク ***InstallShield DevStudio 9.0 以前を使って作成した既存プロジェクトへ [印刷] ボタンを追加するには、以下の手順を実行します。***

1. ISPrint カスタム アクションを作成するには、以下手順を実行します。
   a. カスタム アクション ウィザードを開始します。
   b. **[基本情報]** パネルの **[名前]** ボックスに **ISPrint** を入力します。
   c. **[アクションの種類]** パネルの **[種類]** ボックスで、**[Windows Installer ダイナミック リンク ライブラリの関数を呼び出す]** を選択します。
   d. **[アクション パラメーター]** パネルで、**[参照]** ボタンをクリックし、フォルダーの Redist¥Language Independent¥i386 サブフォルダーにある SetAllUsers.dll ファイルを参照します。**[開く]** をクリックします。

      プロジェクトでパス変数の使用が構成されている場合、InstallShield はパスの一部に定義済みパス変数 <ISRedistPlatformDependentFolder> を使用します。
   e. **[アクションのパラメーター]** パネルの **[ターゲット]** ボックスで、**PrintScrollableText** を入力します。
   f. 以降すべてをデフォルトの設定にしてウィザードを完成します。
2. ダイアログ ボックスでボタンコントロールを作成し、オプションでその **"テキスト"** プロパティを**[印刷(&P)]** に設定します。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」を参照してください。
3. [印刷] ボタンに DoAction イベントを追加し、イベントの **"アクション"** サブ設定で **ISPrint** を選択します。詳細については、「基本の MSI ダイアログでコントロール イベントをトリガーする」を参照してください。
4. Windows Installer プロパティ IS_PRINT_DIALOG を作成して、その値をダイアログ ボックスの名前に設定します。詳細については、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

メモ: *ISPrint カスタム アクションがコントロールイベントによって実行される場合、Windows Installer 制限のため、カスタム アクションのログ記録は通常通りに インストールログ記録 へは記録されません。情報は ISPrintLogmNoten 形式のプロパティの値にログ記録されます。*

Windows Vista 以降のシステムの再起動を最小限にする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Windows ロゴ *・インストール終了後のシステム再起動は、エンドユーザーにとって不都合なものです。Windows ロゴ プログラムの要件の 1 つに、エンドユーザーがインストール完了後自動的にアプリケーションを閉じて再起動を行うことができるオプションを含まなくてはならないという項目があります。*

この要件をサポートするため、すべての基本の MSI プロジェクトには、デフォルトで [MSI 再起動マネージャー – 使用中のファイル] ダイアログが含まれています。インストール中に更新が必要なファイル (複数可) が他のアプリケーションによって使用中の場合、Windows Vista 以降のシステム上では [使用中のファイル (再起動マネージャー)] ダイアログが表示されます。ダイアログには、エンドユーザーが選択できる次の 2 つのオプションが含まれます。

- エンドユーザーは選択で、インストールの完了後、自動的にファイルを使用中のアプリケーションを閉じて、再起動することができる。
- エンドユーザーは、アプリケーションを閉じない選択ができる。インストールの終わりで再起動が必要。

エンド ユーザー エクスペリエンスを最適化するため、アプリケーションには再起動マネージャー API の利用が推奨されます。再起動マネージャーは、エンド ユーザーがアプリケーションを停止した時点から正確に、また効果的にこれを再開します。詳しい情報は、「About Restart Manager (再起動マネージャー)」および MSDN Web サイトで再起動マネージャーに関するその他の文書を参照してください。

Using Windows Installer with Restart Manager (Windows Installer ヘルプ ライブラリ)

ダイアログのコントロール

基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、またはマージ モジュール プロジェクトのいずれの場合でも、定義済みダイアログおよびカスタム ダイアログのレイアウトと動作は、さまざまな共通のコントロールを使って変更することができます。

テーブル 7 • ダイアログのコントロール

コントロールの種類	プロジェクトの種類	説明
ビルボード	基本の MSI、マージ モジュール	ビルボード コントロールは、コントロールイベントに応答して更新可能なデータを表示するために使用されます。たとえば、ビルボードを使用して、遅延しているカスタム アクションの進行を表示できます。 ***プロジェクト:*** *インストールに* Setup.exe *ランチャーが含まれている場合、起動ツールがファイル転送処理中にビルボードを表示するように設定できます。これは、Windows Installer コントロールであるビルボード コントロールの代替となります。この* Setup.exe *ビルボード サポートは、基本の MSI、InstallScript、および InstallScript MSI インストールで使用できます。この種類のビルボードについての詳しい情報は、「ビルボードを表示する」を参照してください。*
ビットマップ	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビットマップ コントロールはイメージを表示します。
チェック ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	チェック ボックス コントロールは、エンド ユーザーが選択またはクリアできるチェック ボックスを表示します。
コンボ ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コンボ ボックス コントロールは、定義済みの値を表示するドロップ ダウン リストを含みます。このボックスはまた、エンド ユーザーが値を入力できる編集フィールドです。

テーブル 7 - ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
ダイアログ	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログ コントロールは、構成された一連の設定と関連付けられています。
ディレクトリ コンボ	基本の MSI、マージ モジュール	ディレクトリ コンボは、ディレクトリ リストやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ コンボは、使用中のシステムにマッピングしたドライブのリストを表示します。
ディレクトリ リスト	基本の MSI、マージ モジュール	ディレクトリ リストは、ディレクトリ コンボやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ リストにはディレクトリ リスト コントロールで選択されたドライブ下にあるフォルダーが表示され、パス編集コントロールの値が入力されます。
編集フィールド	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	編集フィールド コントロールは、エンド ユーザーが文字列または整数を入力できるテキスト ボックスです。
グループ ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	グループ ボックスは、あるエリア内の複数のコントロールを囲むのに使用します。また、グループ ボックスには、そこに含まれるコントロール間の関係を表現するのに使用できるラベルがあります。
ハイパーリンク	基本の MSI、マージ モジュール	ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。 ***重要*** *- Windows Installer 5 以降は、この新しいハイパーリンク コントロールをサポートします。* ***プロジェクト*** *- InstallScript プロジェクト、または InstallScript MSI プロジェクトで HTML コントロールをダイアログに追加する方法については、「ダイアログで HTML コントロールを使用する」を参照してください。*

テーブル 7 • ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
アイコン	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	アイコン コントロールは、アイコンの画像を表示します。
線	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	行コントロールは長さと太さが調節可能な行を作成します。行を使ってダイアログの領域を区分したり、グラフィカルな要素を追加します。
リスト ボックス	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リスト ボックス コントロールは、エンド ユーザーが定義済みオプション リストから単一のオプションを選択できる、標準のリスト ボックスです。
リスト ビュー	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リスト ビュー コントロールは、オプションの単一の列を表示します。各オプションの横にアイコンが表示されます。
マスク編集	基本の MSI、マージ モジュール	マスク編集コントロールは、基本的にエンド ユーザーが指定されたフォーマットで情報を入力できる編集フィールド コントロールです。
パス編集	基本の MSI、マージ モジュール	パス編集コントロールは、ディレクトリ コンボ コントロールやディレクトリ リスト コントロールと一緒に使用して、参照ダイアログを作成します。パス編集には、ディレクトリ コンボおよびディレクトリ リストでエンド ユーザーによる選択によって構成された完全なパスが表示されます。これは、エンドユーザーによる編集に沿ったものである場合も、そうでない場合もあります。
進行状況バー	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	進行状況バーは、コントロール イベントに応じて満たされる動的なグラフィック バーです。

テーブル 7 - ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
プッシュ ボタン	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	プッシュ ボタン コントロールは、エンド ユーザーがクリックしたときにコマンドが実行されるボタンです。
ラジオ ボタン	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ラジオ ボタン コントロールは、エンド ユーザーがその中から 1 つだけを選択できる、2 つまたはそれ以上のオプションの 1 つです。ラジオ ボタンは、ラジオ ボタン グループに挿入しなくてはなりません。これは、ラジオ ボタン グループ コントロールの一部として機能します。
ラジオ ボタン グループ	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ラジオ ボタン グループ コントロールは、ラジオ ボタン コントロールを収容するコンテナです。ラジオ ボタン コントロールとそのラジオ ボタンは、単一のコントロールとして動作します。
スクロール可能テキスト	基本の MSI、マージ モジュール	スクロール可能テキストは、ダイアログに収まりきらないテキストの長い文字列を表示します。このコントロールに含まれているスクロールバーを使って、エンド ユーザーがテキストをスクロールできます。LicenseAgreement ダイアログは、スクロール可能テキスト コントロールを一般的に含むダイアログの例です。
選択ツリー	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	選択ツリーは、CustomSetup ダイアログで見られるような、ユーザーが機能の選択状態を変更できる特殊なコントロールです。
テキスト領域	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	テキスト領域コントロールは、テキストを表示します。
ボリューム コスト リスト	基本の MSI、マージ モジュール	ボリューム コスト リストコントロールには、各ボリュームまたはドライブに関連付けられたディスク容量の要件が表示されます。

テーブル 7 • ダイアログのコントロール (続き.)

コントロールの種類	プロジェクトの種類	説明
ボリューム選択コンボ	基本の MSI、マージ モジュール	ボリューム選択コンボ コントロールによって、エンドユーザーがボリューム、またはドライブをアルファベット順のリストから選択できます。

ビルボード コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

インストールに Setup.exe *ランチャーが含まれている場合、起動ツールがファイル転送処理中にビルボードを表示するように設定できます。これは、Windows Installer コントロールであるビルボード コントロールの代替となります。この* Setup.exe *ビルボード サポートは、基本の MSI、InstallScript、および InstallScript MSI インストールで使用できます。この種類のビルボードについての詳しい情報は、「ビルボードを表示する」を参照してください。*

ビルボード コントロールは、コントロールイベントに応答して更新可能なデータを表示するために使用されます。ビルボードはこの情報を表示する別のコントロールを含むことができますが、テキスト、ビットマップ、およびアイコンを含むそれらのコントロールは、Windows Installer にリンクされていない、スタティックコントロールである必要があります。たとえば、ビルボードを使用して、遅延しているカスタム アクションの進行を表示できます。

ビルボードは、ダイアログ エディター内で完全にはサポートされていません。ビルボードが Windows Installer アクションとインタラクトしてその他のコントロールを表示する場合、プロジェクトの Billboard および BBControl テーブルを変更しなくてはなりません。これらのテーブルの変更には、[ダイレクト エディター] ビューを使用します。

[ダイアログ] ビューにあるダイアログのビルボード コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 8 - ビルボード コントロールの設定

設定	説明
Name	このビルボードの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 8 • ビルボード コントロールの設定 (続き.)

設定	説明
Text	この設定には、ビルボードのコントロールの初期値 (BBControl テーブルを参照) に使用されるテキストが含まれます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ビルボードのコントロールにビットマップまたはアイコンが含まれている場合、この値はコントロールに最初に表示されるファイルに対する Binary テーブルに入る外部キーです。
Text Style	ビルボードのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

Billboard Table (Windows Installer ヘルプ ライブラリ)
BBControl Table (Windows Installer ヘルプ ライブラリ)

ビットマップ コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

ビットマップ コントロールはイメージを表示します。

[ダイアログ] ビューにあるダイアログのビットマップ コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 9 - ビットマップ コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このビットマップの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンドユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
File Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに使用するイメージ ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。

テーブル 9 • ビットマップ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。

テーブル 9 • ビットマップ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

メモ: *InstallScript プロジェクトと基本の MSI プロジェクトで、[ダイアログ] エクスプローラの* ***SdWelcome*** *および* ***InstallWelcome*** *ダイアログでビットマップ コントロールを使用する場合、ダイアログ ボックス上のダイアログ画像の位置が変更されます:*

- *InstallScript プロジェクトの場合、ダイアログ画像はダイアログ ボックスの左側にのみ表示されます。*
- *基本 MSI プロジェクトの場合、ダイアログ画像はダイアログ ボックス全体に表示されます。*

[チェック ボックス] ボタン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

チェック ボックス コントロールは、エンド ユーザーが選択またはクリアできるチェック ボックスを表示します。

[ダイアログ] ビューにあるダイアログのチェック ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。

プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこのタイプのコントロールを作成したとき、InstallShield によって、Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。*

テーブル 10 ・ チェック ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このチェック ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 10 • チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Control Style	基本の MSI、マージ モジュール	このコントロールをテキスト ラベル、アイコン、またはビットマップのいずれを使用してマークするかを指定します。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
File Name	基本の MSI、マージ モジュール	この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。 このコントロールに使用するイメージ ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。

テーブル 10 • チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Icon Size	基本の MSI、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのチェック ボックスを選択したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それを Value 設定に割り当てます。

テーブル 10 • チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。
Push Button	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このチェック ボックスをプッシュ ボタン コントロールに変更するには、True を選択し、そうでない場合は、False を選択します。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。

テーブル 10 • チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、コントロールのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 10 ・ チェック ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Value	基本の MSI、マージ モジュール	チェック ボックスが選択されると、Property 設定で指定されたプロパティがこの値に設定されます。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

コンボ ボックス コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

コンボ ボックス コントロールは、定義済みの値を表示するドロップ ダウン リストを含みます。このボックスはまた、エンド ユーザーが値を入力できる編集フィールドです。

[ダイアログ] ビューにあるダイアログのコンボ ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。

プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこのタイプのコントロールを作成したとき、InstallShield によって、このコンボボックスに属するすべての項目を判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはス*

イート/アドバンスト UI プロパティの使い方」を参照してください。(Windows Installer コンボ ボックスでは、エンド ユーザーは 1 つの項目しか選択できないことに注意してください。)

テーブル 11 • コンボ ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコンボ ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Code Page	基本の MSI、マージ モジュール	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Drop-Down List	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールをドロップダウン リストにするには、True を選択します。コントロールを編集可能なコンボ ボックスにするには、False を選択します。

テーブル 11 • コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。 ***ヒント:*** *コンボ ボックス コントロールのドロップダウンリスト部分の高さを指定する、"Height" 設定に十分な大きさの数字 を指定するようにしてください。*
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェックボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Items	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールにリストするオプションを指定するには、この設定で省略記号ボタン (...) をクリックして、[リスト アイテム] ダイアログ ボックスを開きます。 新しいアイテムをリストに追加するには、[リスト アイテム] ダイアログ ボックスにある [追加] ボタンをクリックします。各アイテムについて、コンボ ボックスに表示されているテキストおよび値 (このアイテムが選択されたときにプロパティに割り当てられる値) を入力する必要があります。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 11 • コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Max. Length	基本の MSI、マージ モジュール	エンドユーザーがコンボ ボックスに入力できる文字数を指定します。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコンボボックスに値を入力した場合、またはコンボボックスから値を選択した場合に設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値 (Items 設定 で指定します) に割り当てます。
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。 Item 設定のすべての値が、Property Is Integer 設定で選択した値に基づいて整数または文字列のどちらかに設定されていることを確認してください。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。

テーブル 11 • コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Sorted	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択すると、コンボボックスの項目はアルファベット順に並べ替えられます。False の値を選択すると、各項目に対して [List Items] ダイアログ ボックスで指定した順序が保持されます。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 11 • コンボ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

ダイアログ コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

[ダイアログ] ビューの [ダイアログ エディター] にあるダイアログを選択すると、右側のペインに以下の設定が表示されます。

コントロールと異なり、ダイアログ エディターからダイアログの名前を変更することはできません。名前を変更するには、[ダイアログ] ビューでダイアログを右クリックして [名前の変更] をクリックします。

テーブル 12 • ダイアログ コントロールの設定

設定	プロジェクトの種類	説明
Caption	基本の MSI、マージ モジュール	ダイアログのタイトルとして使用する名前を指定します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 12 • ダイアログ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
コメント	基本の MSI、マージ モジュール	ダイアログのコメントを入力します。入力したコメントは、参考用にプロジェクト ファイルに保存され、インストールでは使用されません。
Custom Palette	基本の MSI、マージ モジュール	カスタム パレットが必要になるのは、ダイアログに Windows Installer が作成するデフォルトのカラーパレットとは異なるカラーパレットを使用したイメージが含まれる場合のみです。ダイアログに異なるカラー パレットを使用するイメージが含まれていない場合、この値を False のままに残します。
Error Dialog	基本の MSI、マージ モジュール	ダイアログがエラー ダイアログとして機能する場合は、True を選択します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	インストーラー ユニットのダイアログの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Keep Modeless	基本の MSI、マージ モジュール	この値が True で、ダイアログが DoAction イベントにより起動された場合、他のすべてのダイアログはそのままです。この場合に Keep Modeless 設定が False のとき、その他のダイアログは表示されません。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログを配置する位置が画面左端からのパーセンテージで指定します。値が 50 の場合、ダイアログは横方向の中央に表示されます。 ダイアログがインストール ウィザードの一部であり、前のダイアログが異なる位置にあったか、またはエンドユーザーが移動した場合には、この値は無視されます。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択して、ダイアログの左側にスクロールバー (存在する場合) を設定します。デフォルト値が False の場合は、スクロールバーが右側に残ります。 縦長のスクロールバーがある場合に、これをダイアログの左側に表示するには、True を選択します。デフォルト値 False の場合は、スクロールバーが右側に配置されます。
Minimize	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True は、エンドユーザーがこのダイアログを最小化できることを意味します。False はオプションがタイトル バーにないことを意味します。

テーブル 12 • ダイアログ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Modal	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログがインストール ウィザードの一部であり、このダイアログの上に他のダイアログを配置しないようにする場合、True に設定します。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Resource Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この読み取り専用の設定は、ダイアログのリソース ID を表示します。これはダイアログに固有な数値 ID です。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合キャプションはコントロールの左に寄せて配列されます。True に設定するとキャプションは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Text Style	InstallScript、InstallScript MSI	ダイアログに使用するフォントを指定します。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログを配置する位置を、画面上端からのパーセンテージで指定します。値が 50 の場合、ダイアログは縦方向の中央に表示されます。 ダイアログがインストール ウィザードの一部であり、前のダイアログが異なる位置にあったか、またはエンドユーザーが移動した場合には、この値は無視されます。
Track Disk Space	基本の MSI、マージ モジュール	何らかのアクションを実行する前、エンドユーザーに対してドライブの容量が足りない、または OutOfDiskSpace プロパティの値を確認する警告を表示するコントロールがこのダイアログにある場合は True を選択します。

テーブル 12 • ダイアログ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合ダイアログは表示され、False の場合は非表示になります。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	インストーラー ユニットのダイアログの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[ディレクトリ コンボ] コントロール

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *マージ モジュール*

ディレクトリ コンボは、ディレクトリ リストやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ コンボは、使用中のシステムにマッピングしたドライブのリストを表示します。

ディレクトリ コンボをダイアログに最初に作成する場合、InstallShield はWindows Installer のプロパティを指定するプロンプトを表示します。このプロパティは、それに伴うディレクトリ リストとパス編集では同じにする必要があります。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

[ダイアログ] ビューにあるダイアログのディレクトリ コンボ コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 13 • ディレクトリ コンボ コントロールの設定

設定	説明
Name	このディレクトリ コンボの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのテキストに使用されます。

テーブル 13 • ディレクトリ コンボ コントロールの設定 (続き.)

設定	説明
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。

テーブル 13 • ディレクトリ コンボ コントロールの設定 (続き.)

設定	説明
Property	エンド ユーザーがこのディレクトリ コンボの値を選択する場合に設定されるプロパティの名前を入力します。このコントロールはディレクトリ リストに表示されるパスの最初の部分を入力するので、ディレクトリ コンボ、ディレクトリ リスト、およびパス編集を参照ダイアログで一緒に使用する場合は同じプロパティを使用しなければなりません。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Show CD-ROM	ディレクトリコンボに CD-ROM のボリュームを含めるには、True を選択します。
Show Fixed	ディレクトリコンボにハード ドライブを含めるには、True を選択します。
Show Floppy	ディレクトリコンボにフロッピー ドライブを含めるには、True を選択します。
Show RAMDisk	ディレクトリコンボに RAM のボリュームを含めるには、True を選択します。
Show Remote	ディレクトリコンボにマップされたネットワーク ドライブを含めるには、True を選択します。
Show Removable	ディレクトリコンボにリムーバブル ドライブを含めるには、True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 13 • ディレクトリ コンボ コントロールの設定 (続き.)

設定	説明
Text	この設定は、コントロールのテキストに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[ディレクトリ リスト] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ディレクトリ リストは、ディレクトリ コンボやパス編集 コントロールと一緒に使用して、参照ダイアログを作成します。ディレクトリ リストにはディレクトリ リスト コントロールで選択されたドライブ下にあるフォルダーが表示され、パス編集コントロールの値が入力されます。

ディレクトリ リストをダイアログに最初に作成する場合、InstallShield はWindows Installer のプロパティを指定するプロンプトを表示します。このプロパティは、それに伴うディレクトリ コンボとパス編集では同じにする必要があります。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

[ダイアログ] ビューにあるダイアログのディレクトリ リスト コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 14 ・ ディレクトリ リスト コントロールの設定

設定	説明
Name	このディレクトリ リストの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。

テーブル 14 • ディレクトリ リスト コントロールの設定 (続き.)

設定	説明
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Property	エンド ユーザーがこのディレクトリ リストから値を選択する場合に設定されるプロパティの名前を入力します。このコントロールはパス編集に表示されるパスを入力し、ディレクトリ コンボでの選択に応じて変更されます。したがって、参照ダイアログでこの 3 つのコントロールを使用する場合には、すべてのコントロールに対して同じプロパティを使用する必要があります。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールのテキストに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 14 • ディレクトリ リスト コントロールの設定 (続き.)

設定	説明
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[編集フィールド] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

編集フィールド コントロールは、エンド ユーザーが文字列または整数を入力できるテキスト ボックスです。

[ダイアログ] ビューにあるダイアログの編集フィールド コントロールを選択すると、右側のペインに以下の設定が表示されます。

プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこのタイプのコントロールを作成したとき、InstallShield によって、Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows*

Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

テーブル 15 • 編集フィールド コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	この編集フィールドの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージモジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォントスタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージモジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Max. Length	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールに入力できる最大文字数を指定します。
MultiLine	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールを複数行のコントロールとするには、True を選択します。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Password	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールをパスワード コントロールのように動作させ、各文字の代わりにアスタリスク (*) だけを表示するには、True を選択します。このコントロールを通常の編集フィールドコントロールとして使用するには、False を選択します。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールにテキストを入力したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Property Is Integer	基本の MSI、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージ モジュール	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 15 • 編集フィールド コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、マージ モジュール	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[グループ ボックス] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

グループ ボックスは、あるエリア内の複数のコントロールを囲むのに使用します。グループ ボックスを囲む個別のコントロールと一緒にグループ ボックスを移動および調整するには、グループ ボックスと各コントロールを選択する時に CTRL キーを押します。また、グループ ボックスには、そこに含まれるコントロール間の関係を表現するのに使用できるラベルがあります。

[ダイアログ] ビューにあるダイアログのグループ ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 16 • グループ ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このグループ ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 16 • グループ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 16 • グループ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定はグループ ボックスのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 16 • グループ ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

ハイパーリンク コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

InstallScript プロジェクト、または InstallScript MSI プロジェクトで HTML コントロールをダイアログに追加する方法については、「ダイアログで HTML コントロールを使用する」を参照してください。

プロジェクト内のダイアログにハイパーリンクを追加するには、ハイパーリンク コントロールを使用します。ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。

重要: *Windows Installer 5 以降は、この新しいハイパーリンク コントロールをサポートします。このコントロールを Windows Installer の以前のバージョンを持つシステムで表示されるダイアログに使用すると、ランタイム エラー 2885 が発生して、インストールが中止されます。このため、ダイアログ上でハイパーリンク コントロールを使用するとき、インストールが Windows Installer 4.5 以前をターゲットとする場合、プロジェクトにハイパーリンク コントロールを含むバージョンと、含まないバージョンの 2 種類のダイアログを含めることをお勧めします。ダイアログに条件を追加して、ターゲット マシンの Windows Installer バージョンによってダイアログを表示または非表示とします。*

ハイパーリンク コントロールを含むダイアログに使用できるサンプル条件は以下の通りです:

VersionMsi >= "5.00"

ハイパーリンク コントロールを一切含まない代替のダイアログに使用できるサンプル条件は以下の通りです:

VersionMsi < "5.00"

[ダイアログ] ビューにあるダイアログのハイパーリンク コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 17 ・ ハイパーリンク コントロールの設定

設定	説明
Name	このハイパーリンク コントロールの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Code Page	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。

テーブル 17 • ハイパーリンク コントロールの設定 (続き.)

設定	説明
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Format as Bytes	インストールが "テキスト" 設定に入力された値をバイトで表示する場合は、True を選択します。True を選択した場合、"テキスト" 設定には、数値のみを単位を付けずに指定する必要があります。そうすると、実行時に実行時に数値が 512 で除算され、サイズに応じて KB、MB、GB の正しい数字で表示されます。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
No Prefix	"テキスト" 設定にアンパサンドが含まれている場合に、アンパサンド文字 (&) として表示するには、True を選択します。 "テキスト" 設定でアンパサンド文字を使って、後続の文字を下線付きにして表示したい場合 (たとえば、「**クリック(&C)**」は「クリック(C)」と表示されます) には False を選択します。
No Text Wrap	このコントロールでテキストの折り返しを許可するかどうかを示します。True を選択した場合にテキストがテキスト領域の幅を越えると、文字列の末端が切り詰められて省略記号ボタン (...) に置き換わります。False に設定するとテキストが折り返され、テキスト領域の高さが十分ではない場合にはテキスト領域が拡張されます。
Property	エンド ユーザーがこのハイパーリンクをクリックしたときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。

テーブル 17 • ハイパーリンク コントロールの設定 (続き.)

設定	説明
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールで使用するテキストを含みます。この設定でハイパーリンクに HTML マークアップを使用します。この設定のサンプル エントリは、以下のとおりです: **ABC 社の最新情報を見るには、ABC Company Web site を参照してください。** たとえば、**ABC 社の Web サイト** テキストは、**http://www.abccompany.com** Web サイトをポイントするハイパーリンクとして表示されます。 ハイパーリンクに唯一サポートされているプロトコールは HTML です。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Transparent	このコントロールを透過して背景を表示するには、True を選択します。 色付きのビットマップ上にコントロールを配置する場合、コントロールを透明にすることは避けたほうがよい場合があります。エンド ユーザーがディスプレイの配色を変更した場合、テキストが見えなくなる場合があります。たとえば、エンド ユーザーがアクセシビリティの理由でハイコントラスト パラメーターを設定した場合、このコントロールのテキストが見えなくなる場合があります。

テーブル 17 • ハイパーリンク コントロールの設定 (続き.)

設定	説明
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

アイコン コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

アイコン コントロールは、アイコンの画像を表示します。

[ダイアログ] ビューにあるダイアログのアイコン コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 18 • アイコン コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このアイコンの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。

テーブル 18 • アイコン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
File Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに使用するアイコン ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Icon Size	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 18 • アイコン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようアイコンのサイズを変更するには、True を選択します。 アイコンがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 18 • アイコン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

行コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

行コントロールは長さと太さが調節可能な行を作成します。行を使ってダイアログの領域を区分したり、グラフィカルな要素を追加します。

[ダイアログ] ビューにあるダイアログの行コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 19 • 行コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この行の名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。

テーブル 19 • 行コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 19 • 行コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

リスト ボックス コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

リスト ボックス コントロールは、エンド ユーザーが定義済みオプション リストから単一のオプションを選択できる、標準のリスト ボックスです。

[ダイアログ] ビューにあるダイアログのリスト ボックス コントロールを選択すると、右側のペインに以下の設定が表示されます。

プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこの種類のコントロールを作成したとき、InstallShield によって、このリスト ボックスに表示されるすべての項目を判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。*

テーブル 20 ・ リスト ボックス コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このリスト ボックスの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Code Page	基本の MSI、マージ モジュール	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Items	基本の MSI、マージ モジュール	このコントロールにリストするオプションを指定するには、この設定で省略記号ボタン (...) をクリックして、[リスト アイテム] ダイアログ ボックスを開きます。 新しいアイテムをリストに追加するには、[リスト アイテム] ダイアログ ボックスにある [追加] ボタンをクリックします。各アイテムについて、リスト ボックスに表示されているテキストおよび値 (このアイテムが選択されたときにプロパティに割り当てられる値) を入力する必要があります。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールからオプションを選択したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択のテキストに割り当てます。
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sorted	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択すると、リスト ボックスの項目はアルファベット順に並べ替えられます。False の値を選択すると、各項目に対して [List Items] ダイアログ ボックスで指定した順序が保持されます。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 20 • リスト ボックス コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、マージ モジュール	この設定は、スクリーン リーダーで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

リスト ビュー コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

リスト ビュー コントロールは、オプションの単一の列を表示します。各オプションの横にアイコンが表示されます。

[ダイアログ] ビューにあるダイアログのリスト ビュー コントロールを選択すると、右側のペインに以下の設定が表示されます。

プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこの種類のコントロールを作成したとき、InstallShield によって、このリスト ビューに表示されるすべての項目を判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいてこのプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。*

テーブル 21 ・ リスト ビュー コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このリスト ビューの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 21 • リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Icon Size	基本の MSI、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェックボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。

テーブル 21 • リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Items	基本の MSI、マージ モジュール	このコントロールにリストするオプションを指定するには、この設定で省略記号ボタン (...) をクリックして、[リスト アイテム] ダイアログ ボックスを開きます。 新しいアイテムをリストに追加するには、[リスト アイテム] ダイアログ ボックスにある [追加] ボタンをクリックします。各アイテムについて、リスト ビュー コントロールに表示されているテキストおよび値 (このアイテムが選択されたときにプロパティに割り当てられる値) を入力する必要があります。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのコントロールから値を選択したときに設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Property Is Integer	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールのプロパティ (Property 設定で指定します) に整数値が含まれる場合は、True を選択します。プロパティの値が文字列の場合、False を選択します。

テーブル 21 • リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sorted	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True を選択すると、コンボボックスの項目はアルファベット順に並べ替えられます。False の値を選択すると、各項目に対して [List Items] ダイアログ ボックスで指定した順序が保持されます。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。

テーブル 21 • リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージ モジュール	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 21・リスト ビュー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

マスク編集コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

マスク編集コントロールは、基本的にエンド ユーザーが指定されたフォーマットで情報を入力できる編集フィールド コントロールです。

[ダイアログ] ビューにあるダイアログのマスクされた編集コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 22・マスクされた編集コントロールの設定

設定	説明
Name	このマスクされた編集フィールドの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 22 • マスクされた編集コントロールの設定 (続き.)

設定	説明
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Mask	マスクの形式を入力します。詳細については、Windows Installer ヘルプ ライブラリの「MaskedEdit Control」を参照してください。
Max. Length	エンド ユーザーがこのコントロールに入力できる最大文字数を指定します。
Property	エンド ユーザーがマスクされた編集コントロールに入力する値で設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 22 • マスクされた編集コントロールの設定 (続き.)

設定	説明
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

パス編集コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

パス編集コントロールは、ディレクトリ コンボ コントロールやディレクトリ リスト コントロールと一緒に使用して、参照ダイアログを作成します。パス編集には、ディレクトリ コンボおよびディレクトリ リストでエンド ユーザーによる選択によって構成された完全なパスが表示されます。これは、エンドユーザーによる編集に沿ったものである場合も、そうでない場合もあります。

[ダイアログ] ビューにあるダイアログのパス編集コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 23 • パス編集コントロールの設定

設定	説明
Name	このパス編集フィールドの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。

テーブル 23 • パス編集コントロールの設定 (続き.)

設定	説明
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	このプロパティは将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。

テーブル 23・パス編集コントロールの設定 (続き.)

設定	説明
Property	エンドユーザーがこのパス編集に入力する場合や、ディレクトリリストから選択する場合に設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。

テーブル 23 • パス編集コントロールの設定 (続き.)

設定	説明
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[進行状況バー] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

進行状況バーは、コントロール イベントに応じて満たされる動的なグラフィック バーです。

[ダイアログ] ビューにあるダイアログの進行状況バーコントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 24 • 進行状況バー コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この進行状況バーの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。

テーブル 24 • 進行状況バー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Appearance	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	適切なオプションを選択します: • **継続**—進行状況バーは、継続するバーで構成されます。 • **セグメント化**—進行状況バーは、一連の長方形で構成されます。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、コントロールにこのフォント スタイルが使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。

テーブル 24 • 進行状況バー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージ モジュール	この設定は、コントロールで表示されるテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。

テーブル 24 • 進行状況バー コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[プッシュ ボタン] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プッシュ ボタン コントロールは、エンド ユーザーがクリックしたときにコマンドが実行されるボタンです。

[ダイアログ] ビューにあるダイアログのプッシュ ボタン コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 25 • プッシュ ボタン コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このプッシュ ボタンの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Control Style	基本の MSI、マージ モジュール	このコントロールをテキスト ラベル、アイコン、またはビットマップのいずれを使用してマークするかを指定します。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 25 • プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
File Name	基本の MSI、マージ モジュール	このコントロールに使用するイメージ ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Icon Size	基本の MSI、マージ モジュール	アイコン ファイルが複数のリソースを持つことを想定して、コントロールに使用するイメージのサイズを指定します。**[最初のイメージを使用]** オプションを選択すると、ファイル内の最初のイメージが表示され、Stretch to Fit 設定での指定に関わらず、コントロールの大きさに合うよう引き伸ばされます。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 25 • プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
UAC シールド アイコンの表示	基本の MSI、マージ モジュール	エンド ユーザーが Windows Vista 以降のシステムでインストールを実行しているとき、コントロールに [ユーザー アカウント制御 (UAC)] シールド アイコンを含める場合は、True を選択します。シールド アイコンによって、昇格された権限が必要かもしれないことがエンドユーザーに通知されます。 UAC シールド アイコンをコントロールに含めない場合は、False を選択します。
Stretch to Fit	基本の MSI、マージ モジュール	コントロールの領域に収まるようイメージのサイズを変更するには、True を選択します。 イメージがコントロールのサイズよりも小さい場合に中央に配置するか、コントロールのサイズよりも大きい場合にそれをコントロールのサイズにトリミングする場合は、False を選択します。 この設定は、"コントロール スタイル" 設定に [ビットマップ] または [アイコン] が選択されている場合に有効です。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 25 • プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 25 • プッシュ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[ラジオ ボタン] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ラジオ ボタン コントロールは、エンド ユーザーがその中から 1 つだけを選択できる、2 つまたはそれ以上のオプションの 1 つです。ラジオ ボタンは、ラジオ ボタン グループに挿入しなくてはなりません。これは、ラジオ ボタン グループ コントロールの一部として機能します。

ラジオ ボタン コントロールとそのラジオ ボタンは、単一のコントロールとして動作します。ラジオ ボタン グループ内にある個別のボタンを表示または非表示にすることは不可能です。1 つのグループ内のボタンのスタイルはすべて同じです。たとえば、すべてのボタンにテキストを使うか、すべてのボタンにビットマップを使います。

ラジオ ボタン グループを削除すると、グループのすべてのラジオ ボタンも削除されるので注意してください。また、ラジオ ボタン グループ コントロールの名前を変更したとき、そののラジオ ボタンはすべて削除されます。

[ダイアログ] ビューにあるダイアログのラジオ ボタン コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 26 • ラジオ ボタン コントロールの設定

設定	プロジェクトの種類	説明
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。

テーブル 26 • ラジオ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。 グループの中の最初のラジオ ボタンのコントロール ID は、ラジオ ボタン グループのコントロール ID です。2 番目以降のラジオ ボタンのコントロールは、ラジオ ボタン グループの識別子を 1 ずつ加算したものになります。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Order	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ラジオ ボタンをラジオ ボタン グループで表示する順番を示す番号を指定します。この設定は一意の正の整数が設定されなければなりませんが、このグループ内の他のラジオ ボタンの Order 設定は連続的である必要はありません。

テーブル 26 • ラジオ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、ラジオ ボタンのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Value	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ユーザーがラジオ ボタンを選択したときに、それと関連付ける値を入力します。 ***プロジェクト:*** *基本んお MSI プロジェクトの場合: デフォルトでボタンの 1 つを選択にするには、ラジオ ボタン グループの Windows Installer プロパティを [プロパティ マネージャー] ビュー に入力します。スコープ内でプロパティをパブリックにする場合は、すべて大文字で入力する必要があります。その後、デフォルトで表示するボタンの値を指定します。たとえば、ラジオ ボタンのひとつが 104 という値を持ち、ラジオ ボタン グループがプロパティ* GRP_PROPERTY1 *を使う場合、次の情報をプロパティ マネージャーで入力して、このラジオ ボタンをデフォルト選択にします。* • *Name: GRP_PROPERTY1* • *値: 104*

テーブル 26 • ラジオ ボタン コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[ラジオ ボタン グループ] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

基本の MSI とマージ モジュール プロジェクトの場合、ラジオ ボタン グループ内にある個別のボタンを表示または非表示にすることは不可能です。1 つのグループ内のボタンのスタイルはすべて同じです。たとえば、すべてのボタンにテキストを使うか、すべてのボタンにビットマップを使います。

ラジオ ボタン グループ コントロールは、ラジオ ボタン コントロールを収容するコンテナです。ラジオ ボタン コントロールとそのラジオ ボタンは、単一のコントロールとして動作します。ラジオ ボタン グループを削除すると、グループのすべてのラジオ ボタンも削除されるので注意してください。また、ラジオ ボタン グループ コントロールの名前を変更したとき、そののラジオ ボタンはすべて削除されます。

[ダイアログ] ビューにあるダイアログのラジオ ボタン グループ コントロールを選択すると、右側のペインに以下の設定が表示されます。

プロジェクト: *(基本の MSI およびマージ モジュール プロジェクト) ダイアログ上に初めてこの種類のコントロールを作成したとき、InstallShield によって、このラジオ ボタン グループに表示されるすべてのボタンを判別するための Windows Installer プロパティの名前を付けるようプロンプトが表示されます。InstallShield は、この名前をコントロールの Property 設定の値として使用します。実行時に、インストールはエンド ユーザーの選択に基づいて*

このプロパティの値を設定します。詳細については、「Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方」を参照してください。

テーブル 27 • ラジオ ボタン グループ コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このラジオ ボタンの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。 この設定は、ラベルにイメージを含むラジオ ボタン グループには影響しません。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。 グループの中の最初のラジオ ボタンのコントロール ID は、ラジオ ボタン グループのコントロール ID です。2 番目以降のラジオ ボタンのコントロールは、ラジオ ボタン グループの識別子を 1 ずつ加算したものになります。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 27 • ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Has Border	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ラジオ ボタン グループの境界を表示するには、True を選択します。境界を省略するには、False を選択します。 Sunken 設定を使って、境界の種類をより詳細に変更できます。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェックボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。

テーブル 27 • ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Property	基本の MSI、マージ モジュール	エンド ユーザーがこのグループ内のラジオ ボタンの 1 つを選択すると設定されるプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 この設定を使ってデフォルト選択をラジオ ボタン グループに設定する方法については、ラジオ ボタンの Value 設定を参照してください。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。

テーブル 27 • ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text	基本の MSI、マージ モジュール	この設定は、ラジオ ボタン グループのラベルに使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は、ラベルにイメージを含むラジオ ボタン グループには影響しません。
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、ラベルにイメージを含むラジオ ボタン グループには影響しません。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。

テーブル 27 - ラジオ ボタン グループ コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[スクロール可能テキスト] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

スクロール可能テキストは、ダイアログに収まりきらないテキストの長い文字列を表示します。このコントロールに含まれているスクロールバーを使って、エンド ユーザーがテキストをスクロールできます。LicenseAgreement ダイアログは、スクロール可能テキスト コントロールを一般的に含むダイアログの例です。

[ダイアログ] ビューにあるダイアログのスクロール可能テキスト コントロールを選択すると、右側のペインに以下の設定が表示されます。

メモ: *多くのコントロールのテキスト値とは異なり、Windows Installer ではスクロール可能テキスト コントロール内のプロパティ名またはその他の値を解決しません。そのため、ファイルのテキストは書かれたとおりに表示されます。*

テーブル 28 - スクロール可能テキスト コントロールの設定

設定	説明
Name	このスクロール可能テキスト コントロールの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。

テーブル 28・スクロール可能テキスト コントロールの設定 (続き.)

設定	説明
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
File Name	このコントロールに使用する .rtf ファイルのパスと名前を入力するか、この設定の省略記号ボタン (...) をクリックして、そのファイルを参照します。ビルド時にそのファイルがリリースに追加されます。ファイルはインストールでバイナリ リソースとして格納されなくてはなりません。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。

テーブル 28・スクロール可能テキスト コントロールの設定 (続き.)

設定	説明
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[選択ツリー] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

選択ツリーは、CustomSetup ダイアログで見られるような、ユーザーが機能の選択状態を変更できる特殊なコントロールです。

[ダイアログ] ビューにあるダイアログのツリー コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 29 • Selection Tree Control の設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この選択ツリーの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのラベルに使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。

テーブル 29 • Selection Tree Control の設定 (続き.)

設定	プロジェクトの種類	説明
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Indirect Property	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) ボタンをクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
Property	基本の MSI、マージ モジュール	選択ツリーでのプロパティの名前を入力します。エンド ユーザーが参照ダイアログでプロパティを設定することができます。(このダイアログでは、パス編集、ディレクトリ コンボ、およびディレクトリ リスト、コントロールを使用しています。)

テーブル 29 • Selection Tree Control の設定 (続き.)

設定	プロジェクトの種類	説明
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、マージモジュール	この設定は、スクリーン リーダーで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 29 • Selection Tree Control の設定 (続き.)

設定	プロジェクトの種類	説明
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[テキスト領域] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

テキスト領域コントロールは、テキストを表示します。

[ダイアログ] ビューにあるダイアログのテキスト領域コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 30 • テキスト エリア コントロールの設定

設定	プロジェクトの種類	説明
Name	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このテキスト エリアの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	基本の MSI、マージ モジュール	"テキスト スタイル" 設定に何も指定しない場合、テキストにこのフォント スタイルが使用されます。
Cancel	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Code Page	基本の MSI、マージ モジュール	コントロールでインストーラーのパッケージで定義されたコード ページからのフォントを使用する場合は、[データベース] を選択します。コントロールでターゲット システムのデフォルト コード ページのフォントを使用するには、[ユーザーのシステム] を選択します。
Context Help	基本の MSI、マージ モジュール	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Control Identifier	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定には、コントロールの ID として、ダイアログ内で他と重複しない数値を入力します。この ID は、Visual C++ のリソース ID と同じです。ダイアログにデフォルトで用意されているコントロールについては、コントロール ID を変更しないでください。
Default	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。

テーブル 30 • テキスト エリア コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Enabled	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Format as Bytes	基本の MSI、マージ モジュール	インストールが "テキスト" 設定に入力された値をバイトで表示する場合は、True を選択します。True を選択した場合、"テキスト" 設定には、数値のみを単位を付けずに指定する必要があります。そうすると、実行時に実行時に数値が 512 で除算され、サイズに応じて KB、MB、GB の正しい数字で表示されます。
Height	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの高さを指定します。
Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Other Window Styles	InstallScript、InstallScript MSI、InstallScript オブジェクト	省略記号 (...) をクリックすると、[その他のウィンドウ スタイル] ダイアログ ボックスが表示されます。
No Prefix	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	"テキスト" 設定にアンパサンドが含まれている場合に、アンパサンド文字 (&) として表示するには、True を選択します。 "テキスト" 設定でアンパサンド文字を使って、後続の文字を下線付きにして表示したい場合 (たとえば、「**クリック(&C)**」は「クリック (C)」と表示されます) には False を選択します。
No Text Wrap	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールでテキストの折り返しを許可するかどうかを示します。True を選択した場合にテキストがテキスト領域の幅を越えると、文字列の末端が切り詰められて省略記号ボタン (...) に置き換わります。False に設定するとテキストが折り返され、テキスト領域の高さが十分ではない場合にはテキスト領域が拡張されます。

テーブル 30 • テキスト エリア コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
プロパティ	基本の MSI、マージ モジュール	テキストの初期値を提供するプロパティの名前を入力します。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。
Right-Aligned	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Sunken	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この設定は、コントロールで内部に表示されるテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 30 • テキスト エリア コントロールの設定 (続き.)

設定	プロジェクトの種類	説明
Text Style	基本の MSI、マージ モジュール	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	基本の MSI、マージ モジュール	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Transparent	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコントロールを透過して背景を表示するには、True を選択します。 色付きのビットマップ上にコントロールを配置する場合、コントロールを透明にすることは避けたほうがよい場合があります。エンド ユーザーがディスプレイの配色を変更した場合、テキストが見えなくなる場合があります。たとえば、エンド ユーザーがアクセシビリティの理由でハイコントラスト パラメーターを設定した場合、このコントロールのテキストが見えなくなる場合があります。
Visible	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI および マージ プロジェクトの場合: Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。 InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクトの場合: ダイアログ ユニットでコントロールの幅を指定します。

[ボリューム コスト リスト] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ボリューム コスト リストコントロールには、各ボリュームまたはドライブに関連付けられたディスク容量の要件が表示されます。

リストには、5 つの列があります。[ダイレクト エディター] ビューでアクセス可能な UIText テーブルの各列のヘッダーは構成可能です。UIText テーブルは、文字列エントリの値を識別し、適切な場合に翻訳された文字列が実行時に表示されるようにします。各列における UIText テーブルのキーと英語のデフォルト値を以下に示します:

- VolumeCostVolume—Volume
- VolumeCostSize—Disk Size
- VolumeCostAvailable—Available
- VolumeCostRequired—Required
- VolumeCostDifference—Differences

[ダイアログ] ビューにあるダイアログのボリューム リスト コスト コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 31 ・ Volume Cost List コントロールの設定

設定	説明
Name	このボリューム コスト リストの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Available Col Width	Available 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルをボリューム コスト リストのコンテンツに使用します。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。

テーブル 31 • Volume Cost List コントロールの設定 (続き.)

設定	説明
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Differences Col Width	Differences 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Disk Size Col Width	Differences 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Required Col Width	Required 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Show CD-ROM	コントロールに CD-ROM のボリュームを含めるには、True を選択します。
Show Fixed	コントロールにハード ドライブを含めるには、True を選択します。
Show Floppy	コントロールにフロッピー ドライブを含めるには、True を選択します。
Show RAMDisk	コントロールに RAM のボリュームを含めるには、True を選択します。
Show Remote	コントロールにマップされたネットワーク ドライブを含めるには、True を選択します。

テーブル 31・Volume Cost List コントロールの設定 (続き.)

設定	説明
Show Removable	コントロールにリムーバブル ドライブを含めるには、True を選択します。
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールのラベルに使用するテキストを含みます。ラベルは表示されませんが、スクリーン リーダによって使用されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Volume Col Width	Volume 列におけるデフォルトの幅をインストーラー単位で指定します (システム フォントの高さの 1/12 に定義されています)。この列をゼロに設定するか空白のままにすると、列は非表示となります。

テーブル 31 • Volume Cost List コントロールの設定 (続き.)

設定	説明
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

[ボリューム選択コンボ] コントロール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *マージ モジュール*

ボリューム選択コンボ コントロールによって、エンドユーザーがボリューム、またはドライブをアルファベット順のリストから選択できます。

[ダイアログ] ビューにあるダイアログのボリューム選択コンボ コントロールを選択すると、右側のペインに以下の設定が表示されます。

テーブル 32 • ボリューム選択コンボ コントロールの設定

設定	説明
Name	このボリューム選択コンボの名前を入力します。名前は、プロジェクトのすべてのコントロールに対して一意でなければなりません。
Base Text Style	"テキスト スタイル" 設定に何も指定しない場合、このフォント スタイルがコントロールのテキストに使用されます。
Cancel	これがダイアログを終了する唯一のコントロールである場合、True を選択します。キャンセル コントロールをクリックする操作は、ESC キーを押す操作、またはタイトルバーの [閉じる] ボタンをクリックする操作と同じです。[キャンセル] ボタンまたは [完了] ボタンが、通常のキャンセル コントロールです。 このコントロールを含むダイアログの ErrorDialog 設定に True が選択されている場合、この値は無視されます。
Context Help	この設定は将来利用するために予約されています。Windows Installer では、現在コンテキスト ヘルプ項目をインストールから起動することはできません。
Default	これがダイアログ内でデフォルトのコントロールにする唯一のコントロールである場合、True を選択します。エンド ユーザーが Enter キーを押すと、このコントロールがアクティブになります。[次へ] ボタンまたは [OK] ボタンが、通常のデフォルトのコントロールです。
Enabled	コントロールが有効であるかどうかを示します。True は、コントロールが使用可能である (エンドユーザーがコントロールとインタラクトできる) ことを意味します。False は、使用不可能である (灰色表示される) ことを意味します。

テーブル 32 • ボリューム選択コンボ コントロールの設定 (続き.)

設定	説明
Height	Windows Installer ユーザー インターフェイス ユニットでコントロールの高さを指定します (システム フォントの高さの 1/12 に定義されています)。
Indirect Property	このコントロールに関連付けられたプロパティが間接的に参照される場合は True を、そうでない場合は False を選択します。 Indirect Property に True が選択された場合、Windows Installer は参照したプロパティを実行時に解決します。たとえば、このチェック ボックスで _BROWSE というプロパティを使用し、その値が INSTALLDIR であったとします。Indirect Property 設定に True を選択すると、_BROWSE の値は INSTALLDIR プロパティの現在の値となります。Indirect Property 設定に False を選択すると、_BROWSE の値は、文字列 INSTALLDIR を含みます。
Left	ダイアログの左端からコントロールの開始位置までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Left Scrollbar	コントロールの左側に縦長のスクロールバーを表示するには、True を選択します。このオプションは、Right-to-Left 設定で True が選択されている場合のみ選択できます。
Property	エンド ユーザーがこのコントロールから値を選択したときに設定されるプロパティの名前を入力します。このコントロールはディレクトリ リストに表示されるパスの最初の部分を入力するので、ボリューム選択コンボ、ディレクトリ リスト、およびパス編集を参照ダイアログで一緒に使用する場合は同じプロパティを使用しなければなりません。このプロパティはこのコントロールに固有にすることができ、[プロパティ マネージャー] ビューに入力する必要はありません。 このコントロールのデフォルト値を設定するには、その名前に大文字のみを使用してプロパティをパブリック プロパティとし、[プロパティ マネージャー] ビューを使ってパブリック プロパティを追加してから、それをデフォルト選択の値に割り当てます。
Right-Aligned	デフォルト値は False で、この場合テキストはコントロールの左に寄せて配列されます。True に設定するとテキストは右に寄せて配列されます。
Right-to-Left	英語および左から右に書かれるすべての言語には False を選択します。ヘブライ語および右から左に書かれるすべての言語には True を選択します。
Show CD-ROM	コントロールに CD-ROM のボリュームを含めるには、True を選択します。
Show Fixed	コントロールにハード ドライブを含めるには、True を選択します。
Show Floppy	コントロールにフロッピー ドライブを含めるには、True を選択します。
Show RAMDisk	コントロールに RAM のボリュームを含めるには、True を選択します。
Show Remote	コントロールにマップされたネットワーク ドライブを含めるには、True を選択します。
Show Removable	コントロールにリムーバブル ドライブを含めるには、True を選択します。

テーブル 32・ボリューム選択コンボ コントロールの設定 (続き.)

設定	説明
Sunken	コントロールの端をくぼませて、3 次元表示にするには、True を選択します。コントロールにデフォルトのビジュアル スタイルを使うには、False を選択します。
Tab Index	整数を割り当てて、このダイアログにあるスタティック テキストなどのコントロールを除いたコントロールの中で、エンド ユーザーが [タブ] キーを押したときに強調されるダイアログ上の各コントロールの順番を指定します。使用可能な最も低いタブ インデックス番号は 0 です。
Tab Stop	このコントロールがタブ順序で強調されるかどうかを示します。True の場合コントロールは強調され、False の場合、コントロールは強調されません。
Text	この設定は、コントロールで使用するテキストを含みます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Text Style	コントロールのラベルで表示するフォント スタイル、サイズおよび色 (使用可能な場合) を選択します。値を <デフォルト> のままにしておくと、DefaultUIFont プロパティのフォントが表示されます。 この設定は、"コントロール スタイル" 設定に [テキスト] が選択されている場合に有効です。
ツールヒント	エンド ユーザーがコントロールの上にマウス ポインターを置いたときに表示されるテキストを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
Top	ダイアログの上部からコントロールの上部までの距離をインストーラー単位で指定します。(システム フォントの高さの 1/12 に定義されています)。
Visible	True の場合コントロールは表示され、False の場合は非表示になります。ダイアログの [動作] 領域で条件を編集することによって、コントロールを可視にすることもできます。
Width	Windows Installer ユーザー インターフェイス ユニットでコントロールの幅を指定します (システム フォントの高さの 1/12 に定義されています)。

コントロールのコピーと貼り付け

ダイアログ エディターでは、標準的な Windows キーボード ショートカットやコンテキスト (右クリック) メニューを使用してコントロールをコピーし、同じダイアログ内で、またはあるダイアログから別のダイアログに貼り付けることができます。

タスク ***あるダイアログから別のダイアログにプッシュ ボタンをコピーするには以下の手順に従います:***

1. コピーするボタンが含まれるダイアログのレイアウトを開きます。
2. プッシュボタンを選択して CTRL+C を押します。
3. ダイアログ エディターで 2 番目のダイアログを開き、CTRL+V を押してコントロールをダイアログに貼り付けます。

プッシュボタンは、元のボタンと同じ相対サイズと位置を使って貼り付けられ、他のプロパティも、元のプロジェクトと同一です。プッシュ ボタンのコピーには、コントロールの種類に基づいて一意の名前が付けられます。ただし、新規コントロールの動作の情報は (基本の MSI プロジェクトに) コピーされないので 、編集の必要があります。

メモ: *ラジオ ボタン グループは特殊なケースです。このグループをコピーすると、そのラジオ ボタンのすべてが自動的にコピーされます。特定のラジオ ボタンだけをコピーすることはできません。*

コントロールの切り取りと貼り付け

ダイアログ エディターでは、標準的な Windows キーボード ショートカットやコンテキスト (右クリック) メニューを使用してコントロールをあるダイアログから別のダイアログに移動することができます。

タスク ***コントロールをあるダイアログから切り取って、別のダイアログに貼り付けるには、以下の手順を実行します。***

1. コントロールを右クリックしてから **[切り取る]** をクリックするか、CTRL+X を押します。**[コントロールの削除]** ダイアログ ボックスが表示されて、コントロールを削除すると、関連した条件やアクションも同時に削除されることを通知するメッセージが表示されます。
2. コントロールを引き続き現在のダイアログから切り取る場合は **[はい]** をクリックします。
3. ダイアログ エディターで 2 番目のダイアログを開き、CTRL+V を押してコントロールをダイアログに貼り付けます。

コントロールを新しい場所に貼り付けたら、コントロールのプロパティと動作を指定します。

ウィザード インターフェイスを使って作業する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ドキュメントのこのセクションでは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトのウィザード インターフェイスを使った作業について説明します。ウィザード インターフェイス要素は、ウィザード ページおよび 2 番目のウィンドウ (ポップアップ ウィンドウとも呼ばれます) で構成されます。InstallShieldでは、様々なスタイル、ブラシ、およびコントロール テーマを定義ならびにカスタマイズして、ウィザード インターフェイスを効率よくフォーマットできます。また、InstallShield を使ってウィザード ページおよび 2 番目のウィンドウを追加、変更または削除し、各要素が表示されるタイミングをスケジュールできます。

ウィザード インターフェイスの要素

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

次の 2 つのスクリーンショットで、ウィザード インターフェイスの基本要素を示します。

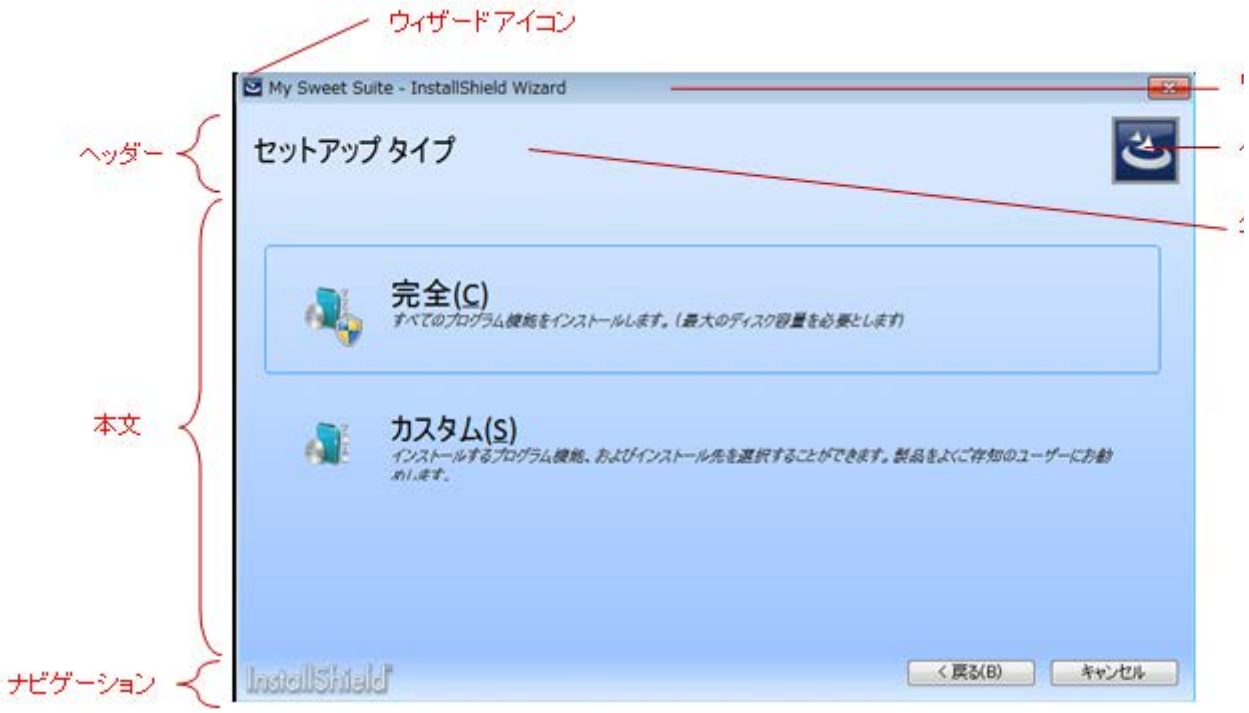


図 26: アドバンスト UI またはスイート/アドバンスト UI インストールのサンプル ウィザード ページ

図 27: アドバンスト UI およびスイート/アドバンスト UI インストールのサンプル 2 番目のウィンドウ

InstallShield では、独自のウィザード アイコン、ヘッダー イメージ、およびウィザード キャプションを使用できます。また、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのユーザー インターフェイスのスタイル、ブラシ、その他をヘッダー、本文、およびナビゲーション領域で定義およびカスタマイズすることも可能です。

スタイル、ブラシ、コントロール テーマ、および最小ウィザード ページを使ってウィザード インターフェイスをカスタマイズする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、様々なユーザー インターフェイス関連のスタイル、ブラシ、およびコントロールのテーマを定義およびカスタマイズして、個別のユーザー インターフェイス要素を手動で編集することなく、ウィザード インターフェイスの外観を簡単に変更することができます。スタイル、ブラシ、およびコントロールのテーマを使用して、インストール全体を通してウィザード インターフェイスのフォーマットを統一することができます。

スタイルウィザード インターフェイスのスタイル

InstallShield には、ウィザード UI 上のテキスト属性 (たとえば、テキストの色、サイズ、フォント名) を定義する多くのビルトイン テキスト スタイルが含まれています。また、InstallShield には 1 つのビルトイン フォント セットが含まれています。フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。フォント セットは、テキスト スタイルと組み合わせて使用します。テキスト スタイルはフォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。任意のビルトイン テキスト スタイルとフォント セットを変更して、必要に応じて独自のテキスト スタイルとフォント セットを作成できます。詳しくは、次を参照してください:

- カスタム スタイルをウィザード インターフェイスに定義する
- テキスト スタイルをウィザード インターフェイスのテキストに適用する

ブラシウィザード インターフェイスのブラシ

InstallShield には、いくつかのビルトン ブラシが用意されています。ブラシを使って、ウィザード インターフェイスの様々な要素、たとえばウィザード ページやコントロール背景に純色、グラデーション、またはイメージを指定します。任意のビルトイン ブラシを変更して、必要に応じて独自のテキスト スタイルとフォント セットを作成できます。詳しくは、次を参照してください:

- カスタム ブラシをウィザード インターフェイスに定義する
- ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する
- ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする

ウィザード インターフェイスのコントロールのテーマ

InstallShield では、ウィザード インターフェイス コントロールのテーマを定義することができます。コントロールのテーマは、コントロールの様々な部分の色 (テキストの色、背景の色、および境界線の色) や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトで、ユーザー インターフェイスのコントロールに使用するコントロールのテーマを指定できます。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

次の種類のコントロールで、コントロールのテーマを使用できます:

- ボタン (ナビゲーション ボタンを含む)
- チェック ボックス
- ラジオ ボタン
- コマンド リンク

詳しくは、次を参照してください:

- カスタム コントロール テーマをウィザード インターフェイスに定義する
- テーマをウィザード インターフェイスのコントロールに適用する

ウィザード インターフェイスのルック アンド フィールをカスタマイズする

[ウィザード インターフェイス] ビューの [ウィザード ページ] ノードを使って、デフォルトのプロジェクト全体を通して使用されるスタイル、ブラシ、およびコントロールのテーマを選択することができます。このビューにある [ウィザード ページ] ノードを使って、ウィザード キャプション、ウィザード アイコン、ヘッダー イメージ、およびナビゲーション バーの高さのデフォルト値をオーバーライドし、その他のプロジェクト全体を通して使用する UI 関連のデフォルト値を指定することもできます。

スタイルの特定の使用方法をオーバーライドしたい場合は、特定のウィザード ページ、2 番目のウィンドウ、およびウィザード コントロールで使用可能な "本文の背景" 設定などのページ固有、およびコントロール固有の設定を使用します。たとえば、InstallationWelcome ウィザード ページの "本文の背景" 設定を使って、プロジェクトのデフォルト背景を InstallationWelcome ウィザード ページでのみオーバーライドすることができます。

カスタム スタイルをウィザード インターフェイスに定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、カスタム フォント セットとテキスト スタイルをプロジェクトに追加してから、これらのフォント セットとテキスト スタイルを様々なウィザード インターフェイス要素に適用することができます。フォント セット とテキスト スタイルは、フォーマットの一貫性を保ち、ウィザード インターフェイス全体にテキスト フォーマットの変更を適応するのに役立ちます。

タスク ***カスタム スタイルを定義するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで **[スタイル]** を右クリックしてから、**[フォント セットの追加]** または **[テキスト スタイルの追加]** をクリックします。

   フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。

   テキスト スタイルはフォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。

   InstallShield によって **[スタイル]** の下に新しいスタイルが追加されます。必要に応じてスタイルの名前を変更します。
3. 新しいスタイルを選択してから、右のペインで設定を構成します。

フォント セットを追加した後、[ウィザード インターフェイス] ビューで定義された任意のテキスト スタイルの "フォント一覧" 設定で、これを選択できます。後で、そのフォント セットに基づいたテキスト スタイルに使用するフォントの一覧を変更したい場合、フォント セットを選択して、その設定を編集します。

テキスト スタイルを追加すると、[ウィザード インターフェイス] ビューで様々なスタイル関連の設定にそれらを選択できます。後でそのスタイルを使用するウィザード インターフェイス要素の外観を変更したい場合は、スタイルを選択してその設定の値を編集できます。

スタイルに加えた変更は、そのスタイルを使用するすべてのウィザード インターフェイス要素に反映されます。

テキスト スタイルをウィザード インターフェイスのテキストに適用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、様々なウィザード インターフェイス要素にテキスト スタイルを適用することができます。テキスト スタイルが使用されているウィザード インターフェイス要素の外観を後で変更したい場合は、そのテキスト スタイルが使用されている各要素を個別に編集する代わりに、テキスト スタイルを簡単に変更することができます。

テキスト スタイルをテキストに適用するとき、ウィザード インターフェイス全体に適用するか、各ウィザード ページのコントロール レベルで適用するかを選択できます。

タスク ***テキスト スタイルをテキスト全体に適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。InstallShield の右側のペインに、グローバル ウィザード インターフェイス設定が表示されます。
3. 次の設定で、適切なテキスト スタイルを選択します:
   - ヘッダー テキスト スタイル
   - ナビゲーション テキスト スタイル

InstallShield によって、プロジェクト全体で該当する UI 要素のテキスト スタイルが変更されます。

タスク ***テキスト スタイルを特定のテキストに適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** ノードまたは **[2 番目のウィンドウ]** ノードを展開して、スタイルを適用するテキストが含まれているウィザード ページまたはウィンドウを見つけます。
3. スタイルを適用するテキストが含まれているコントロールを選択します。
4. 右側のペインにある **"テキスト スタイル"** 設定を拡張してから、そのサブ設定を使って適切なテキスト スタイルを指定します。詳細については、これらの設定を選択したときに右下のペインに表示されるインライン ヘルプを参照してください。

選択されたコントロールのテキスト スタイルが変更されます。

カスタム ブラシをウィザード インターフェイスに定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ブラシを使って、ウィザード インターフェイスの様々な要素、たとえばウィザード ページやコントロール背景に純色、グラデーション、またはイメージを指定します。InstallShield では、カスタム ブラシをプロジェクトに追加してから、これらのブラシを様々なウィザード インターフェイス要素に適用することができます。

タスク ***カスタム スタイルを定義するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ブラシ]** を右クリックしてから、**[ブラシの追加]** をクリックします。InstallShield によって **[ブラシ]** の下に新しいブラシが追加されます。必要に応じてブラシの名前を変更します。
3. 新しいブラシを選択してから、右のペインで設定を構成します。

ブラシを追加すると、[ウィザード インターフェイス] ビューで様々な背景関連の設定にそれを選択できます。たとえば、ウィザード ページのナビゲーション領域の背景にデフォルトで使用するブラシを指定するには、"ナビゲーション背景" 設定を使用します。

ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

デフォルトで、InstallShield ではウィザード ページのヘッダー、本文、およびナビゲーション領域に異なるブラシを使用できます。場合によっては、これらの 3 つの領域にわたって、単一のブラシ (たとえば、単一の背景イメージ、または垂直方向のグラデーション) のみを使用したいときがあります。次の手順は、3 つの異なるブラシまたは単一のブラシを使ってカスタマイズを行う方法を説明します。

ヒント: *ブラシの定義についての情報は、「カスタム ブラシをウィザード インターフェイスに定義する」を参照してください。*

ヘッダー、本文、およびナビゲーション領域に 3 つの異なるブラシを使用する

タスク ***ヘッダー、本文、およびナビゲーション領域に 3 つの異なるブラシを使用するには、以下の手順い従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。"グローバル ウィザード インターフェイス" 設定が、右のペインに表示されます。
3. **"デフォルトの本文背景"** 設定で、ウィザード インターフェイスの本文領域で背景として使用するブラシを選択します。
4. **"ヘッダー背景"** 設定で、ウィザード インターフェイスのヘッダー領域で背景として使用するブラシを選択します。
5. **"ナビゲーション背景"** 設定で、ウィザード インターフェイスのナビゲーション領域で背景として使用するブラシを選択します。

重要 *前述の背景の設定でブラシを選択するとき、"完全なウィザード背景" 設定のすべての値を削除してください。ウィザード インターフェイスでは完全なウィザード背景に単一のブラシを使用するか、ヘッダー、本文、ナビゲーション領域に個別のブラシを使用することができます。これらの 4 つの領域に同時にブラシを指定することはできません。*

ヘッダー、本文、およびナビゲーション領域にわたって単一のブラシを使用する

タスク ***ヘッダー、本文、およびナビゲーション領域に単一のブラシを使用するには、以下の手順い従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。"グローバル ウィザード インターフェイス" 設定が、右のペインに表示されます。
3. **"完全なウィザード背景"** 設定で、ウィザード インターフェイスのナビゲーション背景、本文およびナビゲーション領域に使用するブラシを選択します。

重要 *前述の設定でブラシ を選択するとき、その他の背景設定 ("デフォルトの本文背景"、"ヘッダー背景"、および "ナビゲーション背景") のすべての値を削除してください。ウィザード インターフェイスでは完全なウィザード背景に単一のブラシを使用するか、ヘッダー、本文、ナビゲーション領域に個別のブラシを使用することができます。これらの 4 つの領域に同時にブラシを指定することはできません。*

ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ほとんどの場合、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのユーザー インターフェイス内の各ウィザード ページと 2 番目のウィンドウには、同じ背景が使用されます。一部の場合において、プロジェクト内の特定のウィザード ページまたは 2 番目のウィンドウに設定されているデフォルトの背景をオーバーライドすることもあります。その場合、該当するウィザード ページまたはウィンドウの背景に異なるブラシを選択することができます。

タスク ***特定のウィザード ページまたは 2 番目のウィンドウにブラシを適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** ノードまたは **[2 番目のウィンドウ]** ノードを展開して、ブラシを適用するウィザード ページまたはウィンドウをクリックします。InstallShield の右側のペインに選択されたウィザード ページまたはウィンドウの設定が表示されます。
3. **"本文背景"** 設定で、選択されたウィザード ページまたは 2 番目のウィンドウに使用するブラシを選択します。

選択されたウィザード ページまたは 2 番目のウィンドウの背景が変更されます。

カスタム コントロール テーマをウィザード インターフェイスに定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ウィザード インターフェイス コントロールのテーマを定義することができます。コントロールのテーマは、コントロールの様々な部分の色 (テキストの色、背景の色、および境界線の色) や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトでユーザー インターフェイスのコントロールに使用するコントロール テーマを指定することができます。このテーマは、ユーザー インターフェイス上のすべてのナビゲーション ボタン (たとえば、[戻る] および [次へ] ボタン) のテーマとなります。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

次の種類のコントロールで、コントロールのテーマを使用できます:

- ボタン (ナビゲーション ボタンを含む)
- チェック ボックス
- ラジオ ボタン
- コマンド リンク

タスク ***コントロール テーマを定義するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[コントロール テーマ]**を右クリックしてから、**[コントロール テーマの追加]** をクリックします。InstallShield によって **[コントロール テーマ]** の下に新しいコントロール テーマが追加されます。必要に応じて、コントロール テーマの名前を変更します。
3. コントロール テーマを選択してから、右のペインで設定を構成します。

コントロール テーマを追加すると、[ウィザード インターフェイス] ビューで様々なテーマ関連の設定にそれを選択できます。そのコントロール テーマに基づくコントロールの外観を変更するには、そのコントロール テーマを選択してから、設定を編集します。

コントロール テーマに加えた変更は、そのコントロール テーマを使用するすべてのウィザード インターフェイス要素に反映されます。

テーマをウィザード インターフェイスのコントロールに適用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ボタン、チェックボックス、ラジオボタン、およびコマンド リンクにコントロール テーマを適用することができます。コントロール テーマが使用されているウィザード インターフェイス要素の外観を後で変更したい場合は、そのコントロール テーマが使用されている各要素を個別に編集する代わりに、コントロール テーマを簡単に変更することができます。

いくつかの個別のコントロールにテーマを適用して、その他のコントロールをそのままにしておくか、プロジェクト全体にテーマを適用して、いくつかの個別のコントロールのテーマをオーバーライドすることができます。プロジェクト全体のテーマを変更しても、コントロールごとのテーマが指定されている場合、それが変更されることはありません。

コントロール テーマをプロジェクト全体のナビゲーション ボタンに適用できますが、個別のコントロールに適用することはできませんので、ご注意ください。

タスク ***コントロール テーマをウィザード インターフェイス全体に適用するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。InstallShield の右側のペインに、グローバル ウィザード インターフェイス設定が表示されます。
3. ***"デフォルト コントロール テーマ"*** 設定で、ボタン (ナビゲーション ボタンを含む)、チェックボックス、ラジオボタン、およびコマンド リンクなどのコントロールにデフォルトで使用するコントロールの適切なテーマを選択します。

コントロールごとのテーマが選択されていないすべてのコントロールのテーマが変更されます。

ヒント: *インストールで特定のテーマで定義されている色ではなく、ユーザー定義のシステム色を使用するには、"デフォルト コントロール テーマ" 設定で* ***(テーマなし)*** *オプションを選択します。*

タスク ***デフォルト テーマの選択をオーバーライドしてコントロールにテーマを適用するには、次の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** ノードまたは **[2 番目のウィンドウ]** ノードを展開して、テーマを適用するコントロール (ボタン、チェックボックス、ラジオボタン、コマンド リンク) が含まれているウィザード ページまたはウィンドウを見つけます。
3. テーマを適用するコントロールを選択します。
4. 右側のペインで、**"コントロール テーマ"** 設定を見つけて、選択されたコントロールに使用するテーマを選択します。

選択されたコントロールのテーマが変更されます。

ヒント: *選択されたコントロールのデフォルトのコントロール テーマがオーバーライドされないようにするは、"コントロール テーマ" 設定で [テーマなし] オプションを選択するか、この設定を空白のままに残します。*

最小ウィザード ページのカスタマイズ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield を使って、最小ウィザード ページのテーマをカスタマイズすることができます。カスタマイズされたテーマは、インストール中に表示される 2 番目のダイアログにも適用されます。

- ダーク
- ライト

[最小ウィザード ページ] 領域では、ウィザード ページのキャプション、背景色および [閉じる] および [最小化] ボタンのアイコンをカスタマイズすることもできます。

個別のウィザードページでは、次に記述される追加のカスタマイズも可能です:

- 進行状況バーの色をカスタマイズする
- ライセンス ファイルまたは URL を指定する
- インストール ディレクトリを変更するオプション

テキスト スタイル最小ウィザード インターページのテキストに適用する

[ウィザード インターフェイス] ビューの [スタイル] 領域では、ウィザードの 1 ヶ所以上に適用するフォーマット情報のまとまりを構成して、素早く外観を変更することができます。次は、ダークおよびライト テーマでサポートされている既定のテキストのスタイルです:

- ダーク テーマ – HeaderDark、CaptionDark、および BodyDark
- ライト テーマ – HeaderLight、CaptionLight、および BodyLight

ここに記述されているスタイルは、最小ウィザード ページのテキストに適用される既定のフォント セットおよびテキスト スタイルを指定します。これらのスタイルは、インストール作成者の必要に応じて、さらにカスタマイズを行うことができます。現時点で、最小ウィザード ページでサポートされているカスタム スタイルはありません。

最小ウィザード ページ セットアップのコントロール テーマを選択する

選択されたテーマに基づいて、最小ウィザード ページにはデフォルトで 2 つのコントロール テーマがサポートされています。これらを使って、最小ウィザード ページ セットアップに表示されるボタンの外観を制御することができます。プロパティ

- MinimalPanelLight
- MinimalPanelDark

最小ウィザード ページでサポートされているカスタム コントロール テーマはありません。

最小ウィザード ページのアイコンを選択

[最小ウィザード ページ] 領域では、製品ロゴ、[閉じる] および [最小化] ボタンのアイコンをカスタマイズすることができます。**[サポート ファイル]** 領域には、最小ウィザード ページに表示されるすべてのデフォルト テーマ固有のアイコン ファイルが 100、150 および 200 倍率で含まれています。

制限

スイート プロジェクトにおける最小ウィザード ページのサポートに関する制限事項:

- 最小ウィザード ページのレイアウトのカスタマイズはサポートされていません。ページおよびコントロールの追加/削除を行うことはできません。
- 最小ウィザード ページ上のボタン クリックのアクションを指定することはできません。
- インストールのパスワード保護は、現在サポートされていません。
- セットアップのインストール中、タスクバーに表示される進行状況は、サポートされていません。
- Windows 7 上の InstallShield IDE で最小ウィザード ページをプレビューすることはできません。

ウィザード インターフェイスのフォーマットを選択する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield ではインストールのウィザード インターフェイスに、いくつかの異なるウィザード フォーマットを選択できます:

- **Glass**—インストールはユーザー選択のシステム色を使って、ウィザード インターフェイスのキャプション バー、ヘッダー、およびナビゲーション領域を表示します。また、ウィザード インターフェイスの同じ領域には、Windows Vista で使用できるガラス効果 (透明感) を使うこともできます。このフォーマットを使用すると、ウィザード インターフェイスのキャプション バー、ヘッダー、およびナビゲーション領域に定義されているすべてのブラシは、インストールによって無視されます。

  一部のオペレーティング システムは、このサポートを完全にサポートしません。したがって、インストールが Glass フォーマットの代わりに Wizard 97 フォーマットを使用する場合もあります。

  デフォルトでは、これが設定されています。

- Aero—このフォーマットは、一般的に Glass フォーマットと同じ外観を持ちます。

  一部のオペレーティング システムは、このサポートを完全にサポートしません。したがって、インストールが Aero フォーマットの代わりに Wizard 97 フォーマットを使用する場合もあります。

- Wizard 97—このフォーマットは、従来型のウィザード フォーマットです。インストールはウィザード インターフェイスの様々な領域に定義されているブラシを使います。

- Wizard Lite—このフォーマットは Wizard 97 フォーマットに類似していますが、ヘッダー (タイトルが表示される領域、およびウィザード ページ上部の角に表示されるイメージ) を含みません。インストールはウィザード インターフェイスの別の領域に定義されているブラシを使います。

一部の状況下で、インストールは Glass または Aero フォーマットの代わりに Wizard 97 フォーマットを使用します:

- Windows 8 および Windows Server 2012 システムは、透明感をサポートしません。これらのシステムでは、Glass フォーマットを持つウィザード インターフェイスは Wizard 97 フォーマットを使用します。ただし、Aero フォーマットを持つウィザード インターフェイスは Aero を使用しますが、ガラス効果はありません。

- Windows 7、Windows Vista、Windows Server 2008 R2、および Windows Server 2008 では、エンド ユーザーが有効化/無効化できる透明感サポートが使用できます。これらのシステムでは、透明感が無効化されている場合、Glass フォーマットを使ったウィザード インターフェイスと Aero フォーマットを使ったウィザード インターフェイスでは、Wizard 97 フォーマットが使用されます。また、これらのウィザード インターフェイスでは、ターゲット システム上のデスクトップ コンポジション機能が無効化されている場合、Wizard 97 フォーマットが使用されます。

- Windows XP および Windows Server 2003 上では、Glass フォーマットのウィザード インターフェイスと Aero フォーマットのウィザード インターフェイスは Wizard 97 フォーマットを使用します。

InstallShield の [ウィザード インターフェイス] ビューのウィザード インターフェイス エディターでは、インターフェイスの編集可能なプレビューを表示するときに Glass フォーマットまたは Aero フォーマットを使用しません。アドバンスト UI またはスイート/アドバンスト UI プロジェクトでこれらのフォーマットのどちらかを選択した場合、ウィザード インターフェイス エディターでは、編集可能なプレビューを表示するときに Wizard 97 フォーマットを使用します。

タスク ***ウィザード インターフェイスのウィザード ページに使用するフォーマットを選択するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を選択します。
3. 右側のページに表示される、**"ウィザード フォーマット"** 設定で、適切なオプションを選択します。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白のウィザード ページを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、プロジェクトに新しいブランク ウィザード ページを追加できます。

タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白のウィザード ページを作成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[ブランク ページを追加]** をクリックします。

[ウィザード ページ] の下に新しいウィザード ページが追加されます。必要に応じて、ウィザード ページの名前を変更します。右側のペインで設定を構成して、中央ペインでページを参照およびレイアウトを変更します。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、定義済みのウィザード ページを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに様々な定義済みウィザード ページを追加できます。以下に例を示します:

- エンド ユーザーが .msi パッケージのインストール ディレクトリを選択できるページ
- エンド ユーザーがカスタマー情報およびシリアル番号を入力できるページ
- 機能ツリーおよび機能の説明とサイズを表示してエンド ユーザーがインストールする機能を選択できるページ
- データベース ログイン ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージがターゲットとするデータベース サーバーへの接続を設立することができます。

タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、定義済みのウィザード ページを追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。**ユーザー インターフェイス ウィザード**が開きます。.
3. ウィザードのパネルを必要に応じて完成します。

新しい定義済みのページががプロジェクトに追加されます。右側のペインで設定を構成して、中央ペインでページを参照およびレイアウトを変更します。

メモ: *プロジェクトに [インストール フォルダーを参照] と同様のウィザード ページ (BrowseFolder と呼ばれる) を追加すると、InstallShield によってプレースホルダー イメージ コントロールが追加され、ページにロック イメージを表示することができます。コントロールの "リソース" 設定を構成してこのコントロールと共に表示するファイルを指定するか、イメージを除外する場合は、イメージ コントロールを削除します。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい 2 番目のウィンドウを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、プロジェクトに新しいブランクの 2 番目のウィンドウを追加できます。

タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しい空白の 2 番目のウィンドウを作成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[2 番目のウィンドウ]** を右クリックしてから、**[ブランク ウィンドウを追加]** をクリックします。

2 番目のウィンドウ リストの最後に新しいウィンドウが追加されます。必要に応じてウィンドウの名前を変更します。右側のペインで設定を構成して、中央ペインでページを参照およびレイアウトを変更します。

シーケンス ウィザード ページ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューを使って、アドバンスト UI または スイート/アドバンスト UI インストールで実行時にウィザード ページを表示する順番を指定します。ウィザード ページは、このビューの [ウィザード ページ] ノードの下に、実行時にインストールが表示する順番でリストされます。

タスク

プロジェクト内でウィザード ページをシーケンスするには、次の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード ページ]** ノードの下にある **[ウィザード インターフェイス]** エクスプローラーで、順番を変更するウィザード ページを右クリックしてから、**[上へ移動]** または **[下へ移動]** をクリックします。

リスト内で、ウィザード ページが上または下に移動します。

ウィザード ページまたは 2 番目のウィンドウのレイアウトおよび動作を編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューを使って、インストールに含まれるウィザード ページおよび 2 番目のウィンドウのレイアウトならびに動作を編集します。

ウィザード ページまたは 2 番目のウィンドウにコントロールを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューでウィザード ページまたは二次ウィンドウを選択すると、ウィザード インターフェイスのプレビュー ペインのすぐ上に表示されるツールバーに、新しいコントロールを追加できるボタンやコントロールが追加表示されます。

タスク ***ウィザード インターフェイスのコントロールを追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するウィザード ページまたは 2 番目のウィンドウをクリックします。中央ペインのウィザード エディターに、ページまたは 2 番目のウィンドウが表示されます。
3. ウィザードのインターフェイスのプレビュー ペインの上にあるツールバーで、適切な [新しいコントロール] ボタン ([ラベル]、[テキスト ボックス]、[ボタン]、または [**コンボボックス**] ボタン) をクリックして、コントロール ボタンの横にあるドロップダウン リストで表示されているコントロールの種類から 1 つ選択します。利用可能なそれぞれのコントロールの種類についての詳細は、「[ウィザード インターフェイス] ビューのツールバー」を参照してください。

   ページまたは 2 番目のウィンドウの本文領域の左上付近にコントロールが追加されます。また、選択されたウィザード ページまたはウィンドウの下にあるコントロールにサブノードが追加されます。
4. コントロールの中央にカーソルを配置してから、それを適切な場所にドラッグして移動させます。
5. 右側のペインで、コントロールの設定を構成します。

ヒント: *ウィザード ページのノードの下にリストされているコントロールの順番は、タブの順番に一致します。*

1 つのページで 1 つ以上のラジオボタン コントロール グループを使用している場合、各グループの最初のラジオボタンの "スタイル" 設定に WS_GROUP が含まれていることを確認してください。

ウィザード ページまたは 2 番目のウィンドウに含まれるコントロールのタブの順番を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

多くの場合、エンド ユーザーは TAB キーを使って、ウィザード ページまたはウィンドウ上にあるコントロール間のフォーカスを移動させることができます。デフォルトで、タブの順番は、ウィザード ページまたはウィンドウに追加されたコントロールの順番となります。

タスク

ウィザード ページまたはウィンドウに含まれるコントロールのタブの順番を変更するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、タブの順番を変更するウィザード ページまたは 2 番目のウィンドウのノードを拡張します。

   ページまたはウィンドウ上にある各コントロールには、それぞれサブノードがあります。コントロールはタブの順番に従ってリストされています。つまり、ページ ノードの下にある最初のコントロールが、そのページでフォーカスされる最初のコントロールです。
3. タブの順番にシーケンスするコントロールを右クリックして、**[上に移動]** または **[下に移動]** をクリックします。
4. ページの下にリストされているコントロールが適切なタブの順番に並ぶまで、コントロールを移動させます。

一部の状況において、コントロールがタブの順番に含まれていない場合があります。たとえば、無効なコントロールについては、エンド ユーザーがタブをクリックしてもコントロール間でフォーカスが移動しません。

ウィザード インターフェイスにおけるキーボード ショートカットの指定とアンパサンド (&) の使用

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ウィザード インターフェイス文字列で、アンパサンド (&) はキーボード ショートカットを指定する特殊な文字の場合がありますが、リテラルなアンパサンドの表示を意味することもあります。ボタン、チェックボックス、およびラジオ ボタン コントロールでは、単一のアンパサンドは常に、コントロールをクリックまたは選択するのに使用されるキーボード ショートカットを指定します。また、2 重のアンパサンド (&&) は、単一のアンパサンドが表示されることを意味します。ラベル コントロールの場合、アンパサンドの解釈は、それが使用されているラベル コントロールのスタイルによって異なります。

テーブル 33 ・ ウィザード インターフェイスのラベル コントロールでアンパサンドを使用する

ラベル コントロールのサンプル文字列エントリ	コントロールの "スタイル" 設定における SS_NOPREFIX の状態	ウィザード インターフェイスで表示される文字列	注
新しい && 強化された製品のシリアル番号(&S) を入力してください	SS_NOPREFIX = False	新しい & 強化された製品のシリアル番号(S) を入力してください	エンド ユーザーが ALT キーを押すと、文字列の「S」に下線が表示されます。キーボード ショートカット ALT + S は、タブの順序でこのラベルの直後に表示されるコントロールにフォーカスを移動させます。 「新しい」と「強化された」の間にある 2 重のアンパサンドは、単一のアンパサンドが表示されることを意味します。
新しい & 強化された製品のシリアル番号を入力してください	SS_NOPREFIX = True	新しい & 強化された製品のシリアル番号を入力してください	SS_NOPREFIX 定数の True 値は、コントロールの文字列内のアンパサンドがキーボード ショートカットとして使用されないように防ぎます。

デフォルトでは、ウィザード インターフェイスに新しいラベル コントロールを追加すると、ラベル コントロールの "スタイル" 設定の下にある SS_NOPREFIX サブ設定が False に設定されています。必要に応じて、この値を変更できます。

ウィザード インターフェイスにキーボード ショートカットを持たないラベル コントロールを含む場合、そのコントロールの SS_NOPREFIX サブ設定を True に選択する方法を考慮してください。これによって、コントロールの文字列エントリに含まれる任意のアンパサンドが、誤ってキーボード ショートカットとして解釈されることが無い様にできます。

製品にアンパサンドが含まれている場合、ウィザード インターフェイスに含まれる 2 つの製品名プロパティを使用することができます (1 つめは、単一のアンパサンドを使用するプロパティ、もう 1 つはエスケープされたアンパサンド、つまり二重アンパサンドを使用する値を持つプロパティ)。製品名を持つ各コントロールの種類とスタイルに基づいて、必要に応じて適切な製品名プロパティを使用します。例:

テーブル 34 - アンパサンドを含む文字列の 2 つの異なるプロパティの使用方法サンプル

プロパティ	プロパティ値	このプロパティの使用例
[ProductName]	新しい & 強化された製品	• SS_NOPREFIX = True であるラベル コントロール • [ウィザード ページ] ノードの "ウィザード キャプション" 設定 – この値は、ウィザード ページのキャプション バー内のキャプションとして使用されます。この設定のデフォルト値: [ProductName] – InstallShield Wizard • ウィザード ページの "タイトル" 設定 – タイトル テキストは、ウィザード ページのヘッダー領域に表示されます。
[EscapedProductName]	新しい && 強化された製品	• SS_NOPREFIX = False であるラベル コントロール • SS_NOPREFIX スタイル設定をサポートせず、単一のアンパサンドを常にキーボード ショートカットの指定と解釈するチェックボックス、コマンド リンク、およびラジオボタンなどのコントロール

ウィザード ページでナビゲーション ボタンを使う

プロジェクト- *この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション- *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのウィザード ページには、以下のビルトイン ナビゲーション ボタンを 1 つ以上含めることができます:

テーブル 35 • ウィザード ページのビルトイン ナビゲーション ボタン

ボタン	デフォルト動作の説明
Next	[表示] 条件が False 評価されるページをすべてスキップして、インターフェイスの次のウィザード ページに移動します。 現在のウィザード ページがインターフェイスで最後のページの場合、このボタンをクリックしても何も動作しません。 [次へ] ナビゲーション ボタンは、アクティブ ページの設定アクションを使うコントロールの動作とよく似ていて、特定のページへ移動する動作をトリガします。
戻る	[表示] 条件が False 評価されるページをすべてスキップして、インターフェイスの前のウィザード ページに移動します。 現在のウィザード ページがインターフェイスで最初のページの場合、このボタンをクリックしても何も動作しません。 [戻る] ナビゲーション ボタンは、[アクティブ ページの設定] アクションを使うコントロールの動作とよく似ていて、特定のページへ移動する動作をトリガします。
Cancel	質問 IDS_SUITE_CONFIRMCANCEL を含むメッセージ ボックスを表示します。[はい] の場合、ウィザードをキャンセルして最後のウィザード ページに移動します。 ほとんどの場合、この時点でインストールをキャンセルしても何ら効果がないため、最後のウィザード ページには [キャンセル] ボタンを表示しません。[キャンセル] ボタンが表示されている場合、それをクリックするとウィザードが閉じます。
Install	インストールを開始して次のウィザード ページに移動します。 [インストール] ナビゲーション ボタンは、[インストール] アクションを使うコントロールと似たような動作をし、インストールを開始してから特定のウィザード ページに移動します。
Finish	インストールまたはメンテナンスが正しく完了した後に再起動が必要な場合は、再起動をするかどうか尋ねてから (IDS_SUITE_ASKREBOOT)、適切な応答をトリガします。どちらの場合も (すぐにまたは遅延の再起動が必要、または不要な場合)ウィザードが閉じます。

ウィザード ページにナビゲーション ボタンを含める、またはナビゲーション ボタンを除外する

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューでウィザード ページを選択すると、ビューの右側のグリッドに選択されたページの設定が表示されます。"ナビゲーション領域" の設定を使って、そのページのナビゲーション ボタンを構成できます。

タスク ***ナビゲーション ボタンを含める、または除外するには、以下の手順を実行します:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するボタンを含むウィザード ページをクリックします。
3. ビューの右側にある **[ナビゲーション]** 領域を使って、以下の処理を行います:
   - ウィザード ページに特定のボタンを含めるには、そのボタンの設定で **[はい]** を選択してから、そのボタンの設定の下にあるサブ設定を必要に応じて構成します。
   - ウィザード ページから特定のボタンを除外するには、そのボタンの設定で **[いいえ]** を選択してから、そのボタンの設定の下にあるサブ設定を必要に応じて構成します。

ナビゲーション ボタンのデフォルトの動作を変更する処理については「ウィザード ページのナビゲーション ボタンのデフォルト動作をオーバーライドする」を参照してください。

ウィザード ページのナビゲーション ボタンのデフォルト動作をオーバーライドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ナビゲーション ボタンの "クリック" イベント は、"アクティブ ページの設定" アクションまたは [インストール] アクションを含み、そのアクションをウィザード インターフェイス内で移動させると、デフォルトの動作が阻止されます。デフォルトの動作は、アクションの条件が False のために、スケジュールされたアクションがスキップされた場合でも阻止されます。

たとえば、[インストール] ボタンのデフォルトの動作をオーバーライドしたいとき、インターフェイス内で次のウィザード ページに移動する代わりに、インストールを特定のウィザード ページに進めたい場合があります。このシナリオでは、[インストール] ボタンの "クリック" 設定に [インストール] アクションを追加して、次に続く特定のページを指定しなくてはなりません。その後、アクションに条件を追加すると、条件が [インストール] アクションの実行を阻止して、[インストール] ボタンのデフォルトの動作が抑制されます。つまり、エンド ユーザーが [インストール] ボタンをクリックしても、インストールは開始しません。(この場合、適切な場合は [ナビゲーション] ボタンの "有効" 設定に条件を定義して、[インストール] ボタンを確実に無効にしてください。効果のない [インストール] ボタンをクリックできる状態にしておくと、エンド ユーザーが混乱する可能性があります。)

ISuiteUIExtension インターフェイス上の対応するメソッドを実際に呼び出さない限り、拡張 DLL アクションが、デフォルト アクションを阻止することはできません。アドバンスト UI またはスイート/アドバンスト UI エンジンが、拡張アクションがメソッドの呼び出しを選択したかどうかを判別することはできません。拡張アクション

を使ってデフォルト動作をオーバーライドする必要がある場合で、特定の条件下でのみナビゲートまたはインストールするときは、False 条件 (空白の None グループなど) を持つ [アクティブ ページの設定] アクションまたは [インストール] アクションを追加します。

ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ウィザード インターフェイスに定義済みの選択可能なオプションを含む 1 つ以上のコンボ ボックス コントロールおよびリスト ボックス コントロールを追加できます。その手順には、コントロールの設定を構成して選択可能なオプションのコンテンツを定義する処理が含まれます。また、2 つのプロパティ (コントロールで表示するオプションのリストを定義するプロパティおよび実行時にエンド ユーザーが選択するオプションを格納するためにインストールが使用するプロパティ) も使用します。

タスク

定義済みオプションを含むコンボ ボックス コントロールまたはリスト ボックス コントロールを設定するには、以下の手順に従います:

1. 2 つのプロパティ (コントロールで表示するオプションのリストを定義するプロパティおよび実行時にエンド ユーザーが選択するオプションを格納するためにインストールが使用するプロパティ) をコントロールに関連付けます。

   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。

   b. **[ウィザード インターフェイス]** エクスプローラーで、変更を行うウィザード ページまたは 2 番目のウィンドウを拡張してから、構成したいコンボ ボックス コントロールまたはリスト ボックス コントロールを選択します。

   c. **"プロパティ"** 設定で、コントロールと関連付けるアドバンスト UI またはスイート/アドバンスト UI のプロパティの名前 (たとえば、SELECTIONPROPERTY) を入力します。実行時に、インストールはこのプロパティを使って、エンド ユーザーが選択する値を格納します。

   d. **"コンテンツ プロパティ"** 設定で、このコントロールでリストするオプションを識別するのに使用するアドバンスト UI またはスイート/アドバンスト UI プロパティの名前を入力します。(例、CONTROLOPTIONS)

2. コンボ ボックス コントロールまたはリスト ボックス コントロールに表示するオプション リストを定義します:

a. **"コンテンツ プロパティ"** 設定で、省略記号ボタン (...) をクリックします。**[オプション リストの編集]** ダイアログ ボックスが開きます。

b. **[テキスト]** 列に、コントロールで表示するオプションの 1 つを入力します。

この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

c. **[値]** 列に、手順 2b で入力したリスト オプション テキストと関連付ける値を入力します。エンド ユーザーが **[テキスト]** 列に入力されているリスト オプションを選択したときに、このプロパティ値 (手順 1c で入力) が、SELECTIONPROPERTY に格納されます。

d. 追加でテキスト値のペアを追加する場合、**[新規]** ボタンをクリックすると InstallShield によって追加される行のフィールドに必要な値を入力します。

たとえば、次のようなデータを入力します:

テキスト	値
ID_Option1	Value1
ID_Option2	Value2
ID_Option3	Value3

e. 適切な場合、リスト オプションの順番を並べ替えます。

インストールでは、このダイアログ ボックスでリストされている順番で、リストオプションが表示されます。リスト オプションの順番を並べ替えるには、移動させたい行を選択してから、必要に応じて [上] または [下] ボタンをクリックします。連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。連続しない複数の行を選択するには、選択する行の 1 つをクリックしてから CTRL を押しながら追加する各行をクリックします。

3. **[OK]** をクリックします。

[コンボ ボックス コントロール] または [リスト ボックス] コントロールのデフォルト オプションを指定する

コンボ ボックス コントロールまたはリスト ボックス コントロールのデフォルト オプションを指定するには、[プロパティ マネージャー] ビューを使ってアドバンスト UI またはスイート/アドバンスト UI プロパティの値をデフォルトの選択オプションに対応するプロパティ値と等しく設定します。つまり、デフォルトで *ID_Option3* を選択するには、[プロパティ マネージャー] ビューに SELECTIONPROPERTY プロパティを追加してから、その値を *Value3* に設定します。コントロールのデフォルトを空白にするには、[プロパティ マネージャー] ビューでプロパティを定義付けないままとします。

コンボ ボックス コントロールまたはリスト ボックス コントロールからのエンド ユーザー データを使った作業

実行時に、エンド ユーザーはコンボ ボックスまたはリスト ボックスから単一の選択を行うことができます。これによって、コントロールのプロパティ（前述の手順では SELECTIONPROPERTY）が選択されたオプションに関連付けられた値、たとえば *Value1*、*Value2*、または *Value3* に設定されます。SELECTIONPROPERTY プロパティの値に基づいて、様々な種類の動作をトリガすることができます。例:

- ウィザード ページまたは 2 番目のウィンドウでコントロールを有効または無効できます。そのためには、コントロールの"有効" 設定を使って条件のプロパティの種類を定義し、コントロールの有効状態をプロパティの特定の値と関連付けます。
- ウィザード ページまたは 2 番目のウィンドウで特定のコントロールを表示または非表示できます。そのためには、コントロールの"表示" 設定を使って条件のプロパティの種類を定義し、コントロールの有効状態をプロパティの特定の値と関連付けます。
- 特定のウィザード ページまたは 2 番目のウィンドウを表示またはスキップすることができます。そのためには、ウィザード ページの 2 番目のウィンドウの "表示" 設定を使って条件のプロパティの種類を定義し、ウィザード インターフェイスの有効状態をプロパティの特定の値と関連付けます。
- アドバンスト UI またはスイート/アドバンスト UI プロパティ セットの値を、アドバンスト UI またはスイート/アドバンスト UI インストール内のパッケージで使用されている特定のプロパティに使用することができます。そのためには、[パッケージ] ビューのグリッドにある [操作] 領域で使用できるサブ設定で、コマンドラインおよびサイレント コマンドライン設定を構成します。詳細については、「アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する」を参照してください。

ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、一部の様々なウィザードのインターフェイス コントロールに対して、検証を実装することができます。例:

- エンドユーザーが、テキスト ボックスでシリアル番号を指定されたフォーマットで入力したことを確認できます。
- テキスト ボックスの検証を構成して、エンドユーザーがテキスト ボックスで指定したフォルダーが正しいかどうかを確認できます。
- エンドユーザーが特定の矛盾する組み合わせで、チェックボックスを選択するのを防ぐことができます。

コントロールの検証によって、そのコントロールに関連付けられているプロパティがチェックされます。たとえば、エンドユーザーがチェックボックスをクリックしたとき、それに関連付けられているプロパティは、潜在的に、2 つの新しい値の 1 つに変更されます。検証では、この新しい値が適切であるかどうかを確認する方法が提供されます。

様々な種類のコントロールによって、検証が異なるイベントでトリガされます:

テーブル 36・異なる種類の検証のトリガ

コントロールの種類	検証をトリガするイベント
テキスト ボックス **パスワード ボックス**	コントロールのエントリに変更が発生した時、検証がトリガされます。
ボタン、 **コマンド リンク、** **チェック ボックス、** **ラジオ ボタン、** **イメージ ボタン**	エンドユーザーがコントロールをクリックした時、検証がトリガされます。
コンボ ボックス、 **リストボックス、** **チェック済みリストボックス、** **機能選択ツリー**	コントロールの選択に変更が発生した時、検証がトリガされます。

Text Style 設定のサブ設定を使って、コントロールに異なるテキスト スタイルを指定することにより、エンドユーザーに、異なるコントロールのステータス（デフォルト、有効、または無効）を視覚的に示すことができます。

また、[イベント] 領域のサブ設定を使って、エンドユーザーがコントロールをクリックまたは使用したときに、アクションがトリガされるようにもできます。コントロールに構成されたアクションと検証がある場合、検証はアクションの前に実行されます。アクションに関する詳細は、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

タスク　***コントロールの検証を構成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するウィザード ページまたは 2 番目のウィンドウを選択します。
3. ウィザード エディターで、検証に必要なコントロールを選択します。
4. **"検証"** 設定で、**[検証]** ボタンをクリックしてから発生させる検証の種類をポイントします。

   あるタイプの検証では、作成した DLL ファイル内の関数を呼び出すことができます。プロジェクトの [サポート ファイル] ビューに DLL ファイルが存在する場合、それらのファイルがこのリストで表示されますので、検証関数の名前を、呼び出す DLL 内の関数で上書きすることができます。

次のテーブルは、使用可能な検証の種類について説明します。同じ種類の検証は、特定の種類のコントロールに適用されます。たとえば、値の長さの検証は、テキスト ボックスおよびパスワード ボックス コントロールにのみ適用されます。

テーブル 37・ウィザード インターフェイスで使用されるコントロールのための検証の種類

検証の種類	説明
既存ファイル	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、存在するファイルの完全修飾パスと名前が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでパスとファイル名を入力したときに設定されます。
既存フォルダー	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、存在するフォルダーの完全修飾パスと名前が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでパスとフォルダー名を入力したときに設定されます。
既存フォルダーの新しいファイル	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、存在しないファイルの完全修飾パスと名前が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでパスとファイル名を入力したときに設定されます。 この種類の検証には、ファイルのパスが存在している必要ありますので注意してください。
値の長さ	このタイプの検証は、このコントロールに関連付けられているプロパティの値に、特定の数の文字が含まれているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールでテキストを入力したときに設定されます。 この種類の検証を構成するには、この設定の下のサブ設定値に最小値と最大値を指定します。
値のマスク	このタイプの検証は、このコントロールに関連付けられているプロパティの値が、特定のフォーマットに一致しているかどうかを確認します。通常、プロパティの値は、エンドユーザーがこのコントロールで情報（シリアル番号など）を入力したときに設定されます。 この種類の検証を構成するには、この設定の下の ″パターン″ 設定に必要なフォーマットを指定します。数字の代わりに、次のいずれかの文字を入力します: **# % @** 文字の代わりに、次のいずれかの文字を入力します: **& ^ ? '** その他のすべての入力された文字は正確に一致している必要があります。

テーブル 37 • ウィザード インターフェイスで使用されるコントロールのための検証の種類 (続き.)

検証の種類	説明
DLL アクションを参照	このタイプの検証は、[サポート ファイル] ビューで作成し、プロジェクトに追加した DLL ファイル内の関数を呼び出します。たとえば、DLL ファイルを使用して、プロパティの値の有効性 (エンドユーザーがコントロールで入力した内容) をチェックできます。

ウィザード インターフェイス内の要素のアクションを構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、ウィザード ページまたは 2 番目のウィンドウが開いているまたは閉じているときに起動されるアクションを定義することができます。この機能によって、コンボボックスの値を動的に取得するなど、インストールで任意の初期設定処理が可能となります。また、実行時にウィザード インターフェイスが表示される前に実行されるアクションをスケジュールすることも可能です。

また、InstallShield では、エンド ユーザーが様々なウィザード インターフェイス コントロールを使用したときにトリガされるアクションを定義できます。たとえば、LicenseAgreement ページに [印刷] ボタンを追加して、そのボタン コントロールの印刷アクションを定義できます。つまり、エンド ユーザーが [印刷] ボタンをクリックすると、[印刷] ダイアログ ボックスが開いて、エンド ユーザーが使用許諾契約を印刷できるようにします。

コントロールに構成されたアクションと検証がある場合、検証はアクションの前に実行されます。アクションは、検証が成功したときも、失敗したときも実行されます。これらのアクションはまた、スプラッシュ ページが抑制されたとき、イメージ ボタンがフォーカスされたとき、または解除されたとき、および、エンドユーザーが、リッチ テキスト ボックスの下端までスクロールしたときも実行されます。検証に関する詳細は、「ウィザード ページまたはウィンドウに含まれるコントロールの検証を構成する」を参照してください。

タスク

ウィザード ページ、2 番目のウィンドウ、またはコントロールのアクションを構成するには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、変更するウィザード ページまたは 2 番目のウィンドウを選択します。
3. 次のいずれかの手順に従って、コントロールのアクションを構成します:

- ナビゲーション コントロールの 1 つ ([次へ]、[戻る]、[キャンセル]、[インストール]、または [完了] ボタン) にアクションを構成するには、右側のペインでアクションを変更したいコントロールの設定を拡張します。
- その他のコントロールのアクションを構成するには、ウィザード エディターで、アクションを変更したいコントロールを選択します。

4. 以下のいずれかを実行します。
   - 選択されたウィザード ページまたはウィンドウが開いたときに実行するアクションをスケジュールする場合: **"開始ページ"** 設定で **[新しいアクション]** ボタンをポイントしてから構成するアクションの種類を選択します。
   - 選択されたウィザード ページまたはウィンドウが閉じたときに実行するアクションをスケジュールする場合: **"終了ページ"** 設定で **[新しいアクション]** ボタンをポイントしてから構成するアクションの種類を選択します。
   - コントロールのアクションをスケジュールする場合: **"クリック"** 設定、または **"イベント"** 設定の下にあるアクション関連の設定で **[新しいアクション]** ボタンをポイントしてから、構成するアクションの種類をクリックします。
5. アクションの下のサブ設定を必要に応じて構成します。

複数のアクション ステートメントを入力した場合、アクションは、"アクション" 設定で表示されている順番で実行されます。

あるタイプのアクションでは、作成した DLL ファイル内の関数を呼び出すことができます。プロジェクトの [サポート ファイル] ビューに DLL ファイルが存在する場合、それらのファイルがこのリストで表示されますので、アクション関数の名前を、呼び出す DLL 内の関数で上書きすることができます。

次のテーブルは、使用可能なアクションの種類について説明します。

テーブル 38 ・ コントロールのアクション

アクションの種類	説明
プロパティの設定	この種のアクションは、特定のプロパティを特定の値に設定します。
インストール	この種のアクションは、インストールを開始して特定のウィザード ページに移動します。
開く	この種類アクションは、ファイルまたは Web ページを開きます。
印刷	この種のアクションは、[印刷] ダイアログ ボックスを起動して、エンド ユーザーが指定するファイルを印刷できるようにします。

テーブル 38 • コントロールのアクション（続き.）

アクションの種類	説明
フォルダーの参照	この種のアクションは、エンド ユーザーがコントロールをクリックしたときに [フォルダーを参照] ダイアログ ボックスを起動します。 エンド ユーザーがこのダイアログ ボックスでフォルダーを選択してから [OK] ボタンを クシックすると、ダイアログ ボックスが閉じて、インストールは特定のプロパティの値をエンド ユーザーが選択したフォルダーの完全パスに設定します。 この種のアクションを構成するには、このアクションの下にあるサブ設定を構成します。
ファイルの参照	この種のアクションは、エンド ユーザーがコントロールをクリックしたときに [ファイルの参照] ダイアログ ボックスを起動します。 エンド ユーザーがこのダイアログ ボックスでファイルを選択してから [開く] ボタンを クリックすると、ダイアログ ボックスが閉じて、インストールは特定のプロパティの値をエンド ユーザーが選択したファイルの完全パスとファイル名に設定します。 この種のアクションを構成するには、このアクションの下にあるサブ設定を構成します。
新しいファイルの参照	この種のアクションは、エンド ユーザーがコントロールをクリックしたときに [新しいファイルの参照] ダイアログ ボックスを起動します。ダイアログ ボックスを使って、エンド ユーザーは新しいファイルを保存する場所を参照できます。 エンド ユーザーがこのダイアログ ボックスでファイルを指定してから [保存] ボタンを クリックすると、ダイアログ ボックスが閉じて、インストールは ″プロパティ″ サブ設定で指定されたプロパティの値をエンド ユーザーが指定するファイルの完全パスとファイル名に設定します。 この種のアクションを構成するには、このアクションの下にあるサブ設定を構成します。
アクティブ ページの設定	この種のアクションは、特定のウィザード ページに移動します。
ウィンドウを表示	この種のアクションは、2 番目のウィンドウをモーダル ウィンドウとして表示します。

テーブル 38・コントロールのアクション（続き.）

アクションの種類	説明
ウィンドウを閉じる	この種類のアクションは、メインのウィザード ページまたは 2 番目のウィンドウを閉じるか、場合によっては 2 番目のウィンドウを条件付きで閉じます。 [ウィンドウを閉じる] アクションには、次の定義済みリターン コード ID に対応するパラメータを使用できます：IDOK、IDCANCEL、IDABORT、IDRETRY、IDIGNORE、IDYES、IDNO、および IDCLOSE [ウィンドウを閉じる] アクションの動作は、ウィザード ページおよび 2 番目のウィンドウで次のように多少異なります： ・ ウィザード ページの場合、[ウィンドウを閉じる] アクションはそのリターン コード パラメータが IDCANCEL に設定されている場合、エンド ユーザーがウィザードをキャンセルできるプロンプトを表示（また、エンド ユーザーが [はい] を指定した場合にウィザードを中断します）。その他のリターン コード ID の場合、ウィザードがすぐに閉じます。 ・ 2 番目のウィンドウの場合、[ウィンドウを閉じる] アクションは 2 番目のウィンドウを閉じます。また特殊な場合、たとえば 2 番目のウィンドウが ISRMFilesInUse および ISRMFileInUse の場合は指定されたリターン コード値が戻されます。 InstallShield では現在、指定されたリターン コード ID によって異なるカスタム動作を含む、次の 2 番目のウィンドウが提供されています： ・ ISDownloadProgress ・ ISPromptForSourceMedia ・ ISFilesInUse ・ ISRMFilesInUse ・ ISUpgradeParcel ・ ISSQLBrowse
イベントの停止	この種類のアクションは、後に続くアクションの処理を条件付きで停止します。たとえば、このアクションを使ってボタンのデフォルト動作を抑制することができます。
スイート アクションの実行	 ***プロジェクト：** この機能は、スイート/アドバンスト UI プロジェクトで使用できます。* この種類のアクションは、インストールでアクションを起動します。スイート/アドバンスト UI プロジェクトの [イベント] ビューで定義されている任意のアクションを選択できます。詳細については、「スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する」を参照してください。

テーブル 38 • コントロールのアクション (続き.)

アクションの種類	説明
Web 配置パブリッシュ プロファイルのロード	 ***プロジェクト•*** *この機能は、スイート/アドバンスト UI プロジェクトで使用できます。* この種類のアクションは、エンド ユーザーが Web 配置パッケージに使用する構成設定を含むパブリッシャー プロファイル ファイル (.publishsettings) を参照およびロードすることができる 2 番目のウィンドウを起動します。詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

テーブル 38 • コントロールのアクション（続き.）

アクションの種類	説明
データベース	 ***プロジェクト:*** *この機能は、スイート/アドバンスト UI プロジェクトで使用できます。* 次のデータベース関連のオプションが使用できます: • **データベース メタデータの構成** – データベース メディア（接続文字列、ODBC ドライバー、タイムアウト値等）を初期化し、アクティブな構成として別のアクションが参照できるよう、拡張 DLL に構成の名前を保存します。 • **SQL ログインのプロパティをオーバーライド** – 指定されたデータベース サーバーへの接続を確立するとき、SQL ログインの設定を識別するために使用されるスイートのプロパティを定義します。たとえば、**[データベース ログインの検証]** アクションが呼び出されると、スイート SQL ランタイムは **[SQL ログインのプロパティをオーバーライド]** アクションによって指定されたスイートのプロパティの値を読み込んで、[スイート UI] ウィザード ページに入力された SQL ログインの設定を取得してから、指定されたデータベース サーバーへの接続を設立しようとします。 • **データベース カタログの参照**–使用可能なデータベース カタログ リストを取得して、その結果をスイート プロパティ（**ISSQLBrowseList**）に保存します。このアクションは、**[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** アクションによって識別されたメタデータおよびプロパティを使用します。 • **データベース サーバーの参照**–使用可能なデータベース サーバー リストを取得して、その結果をスイート プロパティ（**ISSQLBrowseList**）に保存します。このアクションは、**[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** アクションによって識別されたメタデータおよびプロパティを使用します。 • **登録済みデータベース サーバーの参照**–ODBC DSN に登録されているデータベース サーバー リストを取得して、そのリストをスイート プロパティ（**ISSQLBrowseLocalList**）に保存します。このアクションは、**[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** アクションによって識別されたメタデータおよびプロパティを使用します。 • **データベース ログインの検証** – **[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** によって識別されたデータベース メタデータおよびプロパティを使って、データベース ログイン資格情報を検証する [検証] アクションを追加します。 • **SQL 文字列の実行** – **[データベース メタデータの構成]** および **[SQL ログインのプロパティをオーバーライド]** によって識別されたデータベース メタデータおよびプロパティを使って、SQL ステートメントを実行します。

テーブル 38 • コントロールのアクション (続き.)

アクションの種類	説明
DLL アクションを参照	このタイプのアクションは、[サポート ファイル] ビューで作成し、プロジェクトに追加した DLL ファイル内の関数を呼び出します。たとえば、DLL ファイルを使って、カスタム動作をトリガすることができます。

ウィザード インターフェイスにおけるイメージ、テキスト ファイル、その他のオブジェクトの使用

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのウィザード インターフェイスで使用するイメージ、テキスト ファイル、その他のファイルはすべて、プロジェクトにサポートファイルとして含める必要があります。サポート ファイルとは、インストール処理中のみの使用目的でターゲット システムにインストールされるファイルです。これらのファイルは、インストールの完了時に自動的にターゲット システムから削除されます。

プロジェクトが複数言語をサポートする場合、言語固有のサポート ファイルを [サポート ファイル] ビューの適切な言語固有の領域にに追加できます。実行時、アドバンスト UI またはスイート/アドバンスト UI インストールは、適切な言語固有のサポートファイルを表示します。

詳細については、「サポート ファイルを使用する」を参照してください。

ウィザード インターフェイスにおける DPI 対応

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける DPI 対応には、2 つの基本事項があります:

- エンジンが使用するアイテムは、システムの DPI 設定にしたがって調整されます。つまり、ターゲット システムで DPI 200% が設定されている場合、チェックボックス コントロールも同様に調整されます。
- エンジンは、ウィザード インターフェイスに含まれたイメージその他のリソースを表示するとき、スケール因子と言語を考慮します。

たとえば、エンジンがスケール因子を 150 と判断したとき、適切な言語が英語である場合は、ファイル名またはパス名に *scale-150* の文字列を含むリソースを探します。ターゲット システム上で image.png を検索するとき、次にリストされている順番でパスを検索します:

1. [SUPPORTDIR]0409¥**scale-150**¥image.png
2. [SUPPORTDIR]0409¥image.**scale-150**.png
3. [SUPPORTDIR]0409¥image.png
4. [SUPPORTDIR]**scale-150**¥image.png
5. [SUPPORTDIR]image.**scale-150**.png
6. [SUPPORTDIR]¥image.png

正しい DPI よりも一致する言語が優先します。

InstallShield はアドバンスト UI およびスイート/アドバンスト UI プロジェクトに含まれるビルトイン デフォルト イメージに、scale-150 および scale-200 イメージを含みます。プロジェクトにカスタム リソースを含める場合、[サポート ファイル] ビューを使って適切なスケールのイメージを追加します。

エンドユーザー インターフェイスのローカライズ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield には、エンド ユーザーに表示するテキスト文字列を指定するための多くの設定があります。たとえば、機能の "表示名" と "説明" 設定に値を入力すると、実行時に CustomSetup ダイアログで表示される機能名と説明を構成することになります。1 つまたは複数の言語にローカライズが必要な設定に値を入力するたびに、プロジェクトがサポートする各言語用に InstallShield が自動的に文字列エントリを作成します。各文字列エントリは、言語非依存の識別子と対応する言語固有の値で構成されます。実行時に、インストールは適切な翻訳済み文字列値を表示します。ハードコード化されたテキストの代わりに文字列エントリを使用することで、インストール中に表示される全ての言語固有文字列から、インストールのコードを完全に切り離すことができます。

プロジェクトのローカライズ処理を簡素化するために、インストール処理中、実行時に表示されるすべてのテキスト文字列は、統合された 1 つのビュー、[文字列エディター] ビューに表示されます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。またこのビューでは、各言語の文字列エントリをファイルにエクスポートして、ファイルにリストされている値を翻訳してから、翻訳済みファイルをプロジェクトにインポートすることができます。

複数言語をサポートするプロジェクトで文字列エントリを処理する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

エディション: *InstallShield Premier Edition では、複数言語インストールの作成をサポートします。*

メモ: *プロジェクトに言語を追加するには、[一般情報] ビューの "セットアップ言語" 設定を使用します。*

プロジェクトは、ローカライズ可能な各テキスト文字列およびプロジェクトでサポートする各言語用の文字列エントリを含みます。[文字列エディター] ビューは、プロジェクト内の言語非依存文字列 ID の一覧と対応する言語固有の値が表示されます。

複数言語サポートを含めるプロジェクトでは、文字列エントリに関する次の詳細にご注意ください:

- InstallShield ですべてのローカライズ可能な設定の値 (たとえば、機能の "表示名" と "説明" 設定) は、プロジェクトのデフォルト言語を使用します。プロジェクトのデフォルト言語は、[文字列エディター] ビューで指定できます。これらのローカライズ可能な設定の値を編集すると、InstallShield が [文字列エディター] ビューでデフォルト言語の文字列エントリの値を同時に更新します。

  同様に、[文字列エディター] ビューを使ってプロジェクトのデフォルト言語の文字列値を変更すると、InstallShield が、その文字列エントリの値を表示するその他の InstallShield ビューの設定を同時に更新します。

- プロジェクトでサポートする各言語は、同じ一連の文字列 ID を使用します。

  したがって、プロジェクトに文字列エントリを追加すると、InstallShield は各言語に対して同じ文字列 ID を使用します。文字列 ID の名前を変更すると、すべての言語の文字列 ID も変更されます。文字列エントリを削除すると、各言語の文字列エントリが削除されます。

- 新しい言語をプロジェクトに追加すると、プロジェクト内の他の言語で利用可能な文字列 ID と同じ文字列 ID がその言語にも追加されます。デフォルトの文字列エントリ (つまり、組み込みダイアログやその他のユーザー インターフェイス要素で使用できる文字列エントリ) には、既に翻訳済みの文字列値が追加されます。カスタム文字列エントリには、新しい言語用のデフォルト言語からの文字列が使用されます。必要に応じて、新しい言語用の文字列値をどれでも変更することができます。

プロジェクト作成の終了後、テキスト ファイルに各言語の文字列エントリをエクスポートして、翻訳することができます。テキスト ファイルの文字列エントリを翻訳した後、それをプロジェクトにインポートできます。

InstallShield で文字列エントリを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

メモ: *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージ (.appx) をビルドします。*

プロジェクトを通して文字列をハードコード化する代わりに、InstallShield でローカライズ済みテキストを受け入れる領域では、文字列エントリを使用できます。文字列エントリを選択する方法は、それらを使用する場所によって異なります。

様々なビューでの設定

InstallShield のビュー内にある設定で値を入力するときに、その値がエンド ユーザーに表示されるテキスト文字列である場合、自動的にその設定の文字列 ID が使用されます。InstallShield は、文字列値の前に中括弧で囲まれた文字列 ID を配置します。機能の "表示名" 設定の値の例を以下に示します:

{ID_STRING24}My New Feature

この例では、**ID_STRING24** が、"表示名" 設定で使用される文字列 ID です。**My New Feature** は、プロジェクトのデフォルト言語の文字列値です。

タスク ***ローカライズ可能な文字列エントリを受け付ける設定での処理には、以下のいずれかを行います:***

- プロジェクト内の何処にも存在しない新しいテキスト文字列を入力するには、使いたいテキスト文字列を入力します。入力した値には、自動的に新しい文字列 ID が割り当てられます。

  プロジェクトが複数の言語をサポートする場合、プロジェクトの言語すべてに新しい文字列 ID が追加され、入力した文字列値がすべての言語の値として使用されます。

- プロジェクトで使用されている既存する文字列エントリの一覧を参照するには、設定の値をクリックしたときに右側に表示される省略記号ボタン (...) をクリックします。既存の文字列エントリを選択、または新しい文字列エントリを作成できる [文字列の選択] ダイアログ ボックスが開きます。

注意・ *既存の文字列値を編集するのと、InstallShield 設定でテキストを入力して新しい文字列 ID を入力するのは異なる点に注意してください。設定に対して文字列 ID を選択した後に設定のローカライズ可能なテキストを削除しても、単に現在値が削除されるだけです。これでは、現在の文字列 ID は新しい ID に置き換えられません。*

ダイアログ エディター

ダイアログ エディターには、エンドユーザーに表示される文字列を使用できるたくさんのコントロールがあります。たとえば、テキストプロパティやツールチッププロパティは常にローカライズ可能テキストが必要になります。

ダイアログ エディターで行う文字列エントリの処理は、InstallShield のその他の場所で文字列 ID を選択するのと似ています。

タスク ***ダイアログ エディターとローカライズ可能テキストを使用するコントロールのプロパティを使用するには、以下のいずれかを行います:***

- コントロールのプロパティ シートをクリックして、文字列を編集します。すると、現在の言語の文字列エントリが変更されます。最初に文字列を選択せずにテキスト値を入力すると、InstallShield がプロジェクトの新しい文字列 ID を作成します。

  プロジェクトが複数の言語をサポートする場合、プロジェクトの言語すべてに新しい文字列 ID が追加され、入力した文字列値がすべての言語の値として使用されます。

- ローカライズ可能なプロパティの値を編集するためにクリックすると、省略記号ボタン (...) がプロパティ シート内に表示されます。省略記号ボタンをクリックして、現在の言語の文字列エントリを表示します。

ダイアログ コントロールのプロパティと、InstallShield 内のその他のビューにある設定との最も大きな違いは、その他のビューでは常にデフォルト言語の文字列が表示される点です。ただし、ダイアログのレイアウトを編集するとき、最初にダイアログの言語を選択しなくてはなりません。すると、ダイアログとプロパティシートの文字列はすべて編集しているダイアログで使用中の言語で表示されます。

InstallScript スクリプト エディター ペイン

InstallScript ビューの InstallScript スクリプト エディター ペインでは、[文字列の選択] ダイアログ ボックスを使って、文字列エントリを選択および編集することができます。

タスク ***InstallScript で文字列 ID を使用するには、以下の手順を実行します。***

1. スクリプト内で文字列 ID を挿入する場所にカーソルを置きます。
2. **[編集]** メニューで、**[挿入]** をポイントして **[文字列エントリ]** を選択します。**[文字列の選択]** ダイアログ ボックスが開きます。
3. 以下のいずれかを実行します。
   - 既存の文字列エントリを使って、スクリプトで使用する文字列エントリを含む行をクリックします。
   - 新しい文字列エントリを作成するには、**[新規作成]** ボタンをクリックします。**[文字列エントリ]** ダイアログ ボックスが開き、ここで新しい文字列 ID と値を作成できます。
4. [OK] をクリックします。

InstallShield は、スクリプト中に (@) 記号に続けて文字列 ID を配置します。

詳細は、「文字列定数演算子(@)」を参照してください。

ヒント: *ハードコード化されたユーザー インターフェイス文字列を直接 InstallScript コードに含めることもできますが、この方法は回避することをお勧めします。InstallScript コード内にハードコード化された文字列は、Unicode として格納されません。したがって、正しいコード ページが設定されたシステム上でインストールが実行された場合にのみ、言語が正しく表示されます。[文字列エディタ] ビューを使ってプロジェクトに文字列を追加して、InstallScript コードからその文字列に関連付けられた文字列 ID を参照することで、この問題を回避できます。*

ビルド時と実行時に文字列エントリに起きる処理

InstallShield は、文字列エントリを InstallShield プロジェクト ファイル (.ism) の ISString テーブルに格納します。ほとんどの場合、リリースを作成するとき、ビルド時に文字列 ID ではなく文字列値が使用されます。つまり、文字列 ID はリリースに組み込まれず、文字列 ID は実行時に使用されません。

InstallShield がビルド時に文字列 ID を使用する唯一の例外は、プロジェクトに InstallScript コードが含まれている場合です。この場合、InstallShield がビルドするリリースには、すべての文字列エントリを含む文字列テーブルが含まれます。これらの文字列テーブルによって、ハードコード化されたテキスト文字列の代わりに、InstallScript コード内に文字列 ID を使用できるようになります。実行時、インストールは必要に応じて、文字列 ID を文字列値と置き換えます。

文字列エントリを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] ビューでは、後で InstallShield 内の設定や、プロジェクトに含まれるダイアログ、および InstallScript コードで参照することが可能な新しい文字列エントリを作成できます。

タスク ***新しい文字列エントリをプロジェクトに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 以下のいずれかを実行します。
   - **[新しい文字列エントリ]** ボタンをクリックします。
   - INSERT キーを押す。

   [文字列エントリ] ダイアログ ボックスが開きます。

3. **[文字列 ID]** ボックスで、文字列エントリに使用する言語非依存の文字列 ID を指定するか、自動的にボックスに挿入される新しいデフォルト文字列 ID のままに残します。
4. **[値]** ボックスに、文字列エントリに使用するローカライズ可能なテキスト文字列を指定します。
5. **[コメント]** ボックスに、オプションとして、文字列エントリについて内部で使用するメモを指定します。コメントは実行時には表示されません。
6. [OK] をクリックします。

[文字列エディター] ビューで、新しい文字列エントリ用の新しい行が追加されます。

ヒント: *プロジェクトが複数の言語をサポートする場合、プロジェクトの言語すべてに文字列エントリが追加され、入力した文字列値がすべての言語の値として使用されます。必要な場合は、言語の文字列値を編集できます。*

メモ: *InstallShield では、ローカライズ可能なテキストを受け入れる設定には必ず文字列 ID を使用しなくてはならないため、既存の文字列 ID を選択せずに設定に値を入力すると、プロジェクトに新しい文字列エントリが自動的に追加されます。*

文字列エントリを編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] ビューには、プロジェクト内の文字列エントリを変更できる、スプレッドシート形式のテーブルが表示されます。

タスク ***[文字列エディター] ビューで文字列エントリを編集するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 変更する文字列エントリを見つけます。
3. 以下のいずれかを実行します。
   - テーブル セル内のテキストをすべて上書きするには、テーブル セルをクリックしてから、新しい ID、値、またはコメントを入力します。
   - テーブル セル内の特定の位置にカーソルを配置するには、その場所をダブルクリックします。次に、変更を入力します。

文字列 ID の名前を変更すると、プロジェクト内の全ての言語に対する文字列 ID も変更されますので、ご注意ください。

文字列エントリを編集すると、変更を行った日付と時刻が [更新日時] 列に表示されます。[文字列エディター] ビュー内の文字列エントリは、更新日時順に並べ替えることができます。これは、文字列が最後に翻訳された以降に変更された文字列を識別するのに、便利です。

ヒント: *InstallShield の他のビュー内にあるローカライズ可能な設定を編集中に、その設定にある省略記号ボタン (...) をクリックできます。[文字列の選択] ダイアログ ボックスが開き、ここで選択した設定に使用する文字列エントリを指定できます。*

プロジェクト: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、一部の文字列値は角括弧でかこまれた Windows Installer プロパティを含みます (例、***Install [ProductName]***)。実行時に、このプロパティと括弧はプロパティ値で置き換えられます。*

*文字列値の中には、中括弧でかこまれたフォント情報を含むものもあります (例、***{&MSSansBold8}OK***)。フォント情報は、実行時に文字列を表示するのに使用するスタイルの詳細を示します。*

文字列 ID のインスタンスを検索する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

文字列 ID は InstallShield プロジェクト全体で使用されます。特定の文字列 ID がプロジェクト内の何処で使用されているかを確認したい場合、[文字列エディター] ビューで検索を行うことができます。

タスク ***プロジェクト内の特定の文字列 ID をすべて見つけるには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 検索を行う文字列 ID を含む行を選択します。

   連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。連続しない複数の行を選択するには、最初の行をクリックしてから CTRL を押しながら追加するそれぞれの行をクリックします。
3. **[選択した文字列をプロジェクトで検索]** ボタンをクリックします。

InstallShield が、検索結果を [出力] ウィンドウの [結果] タブに表示します。[結果] タブは、文字列 ID が使用されている回数を表示します。また、[ダイレクト エディター] ビュー内のどの場所に文字列 ID が使用されているかも示します。

文字列 ID がプロジェクト内で使用されていない場合、[結果] タブには 0 個見つかったと表示されます。

メモ: *InstallShield は、文字列 ID のインスタンスをプロジェクトの InstallScript ファイル (.rul) では検索しません。*

文字列エントリの検索と置換

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] は、標準の検索/置換機能をサポートします。この機能は、プロジェクト内で特定の文字列を検索して、改訂済みの文字列と置換するときに便利です。

文字列エディターで文字列を検索する

タスク ***文字列エディターで文字列を検索するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[文字列の検索]** ボタンをクリックします。**[検索]** ダイアログ ボックスが開きます。
3. **[検索する文字列]** ボックスで、検索する文字列を入力します。
4. 必要に応じて、その他のオプションを選択します。
5. **[次を検索]** をクリックします。

InstallShield が、指定された文字列の最初のインスタンスを検出します。

文字列エディターで文字列を検索/置換する

タスク ***文字列エディターで文字列を検索/置換するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[検索/置換]** ボタンをクリックします。**[置換]** ダイアログ ボックスが開きます。

3. **[検索する文字列]** ボックスで、置換するテキストを入力します。
4. **[置換後の文字列]** ボックスで、新しい文字列を入力します。
5. 必要に応じて、その他のオプションを選択します。
6. **[次を検索]**、**[置換]**、または **[すべて置換]** をクリックします。

InstallShield が必要に応じて文字列を置換します。

文字列エントリを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクト内の特定の文字列エントリが不要になった場合は、それを削除できます。

文字列エントリを削除すると、プロジェクトに含まれる各言語からその文字列エントリが削除されます。

タスク ***文字列エントリをプロジェクトから削除するには、次の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. 削除する文字列エントリを見つけます。
3. **[選択した文字列を削除]** ボタンをクリックするか、または DELETE キーを押します。

ローカライズ可能な設定から文字列 ID を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield のビュー内にある設定で値を入力するときに、その値がエンド ユーザーに表示されるテキスト文字列である場合、自動的にその設定の文字列 ID が使用されます。InstallShield は、文字列値の前に中括弧で囲まれた文字列 ID を配置します。

設定の値を削除すると、実際は現在の言語用の文字列値が削除されます。そのような場合、設定には空の値を持つ文字列 ID が使用されます。

つまり、ローカライズ可能な設定から文字列 ID を削除する場合、それを異なる文字列 ID に置き換えるか、新しい文字列 ID を追加しなくてはなりません。

文字列 ID の置換についての詳細は、「文字列エントリを編集する」を参照してください。新しい文字列 ID の作成方法については、「文字列エントリを追加する」を参照してください。

ビルボードを表示する

プロジェクト: *ビルボードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

プロジェクト: *ビルボードのサポートは、使用しているプロジェクトの種類によって異なります。*

このセクションには、次のトピックが含まれています:

- 基本の MSI インストールでビルボードを表示する
- InstallScript および InstallScript MSI インストールでビルボードを表示する

基本の MSI インストールでビルボードを表示する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

基本の MSI プロジェクトにおけるビルボード ファイルの種類

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield では、ビルボード用に様々な種類のファイルがサポートされています:

- Adobe Flash アプリケーション ファイル (.swf)
- イメージ (.bmp、.gif、.jpg、および .jpeg)

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

メモ: *プロジェクトに複数のイメージ ビルボードを追加することができますが、Adobe Flash アプリケーション ファイル ビルボードの場合は 1 つだけしか追加できません。*

基本の MSI プロジェクトにおけるビルボードの種類

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield は、異なる種類のビルボードをサポートします。たとえば、インストールが全画面背景を使用し、ビルボードを前景に、また小さい進行状況ボックスを画面の右下に表示するスタイルがあります。別のスタイルでは、インストールがビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。

各ビルボード タイプの説明とサンプル スクリーンショットは、次のとおりです。

全画面表示、右下に小さい進行状況ボックスを表示する

全画面表示、右下に小さい進行状況ボックスを表示するタイプのビルボードは、インストールが標準エンドユーザー ダイアログを表示するときに、全画面の背景も表示します。ファイルの転送中、インストールが全画面背景を使用し、ビルボードを前画面に、また小さい進行状況ボックスを画面の右下に表示します。

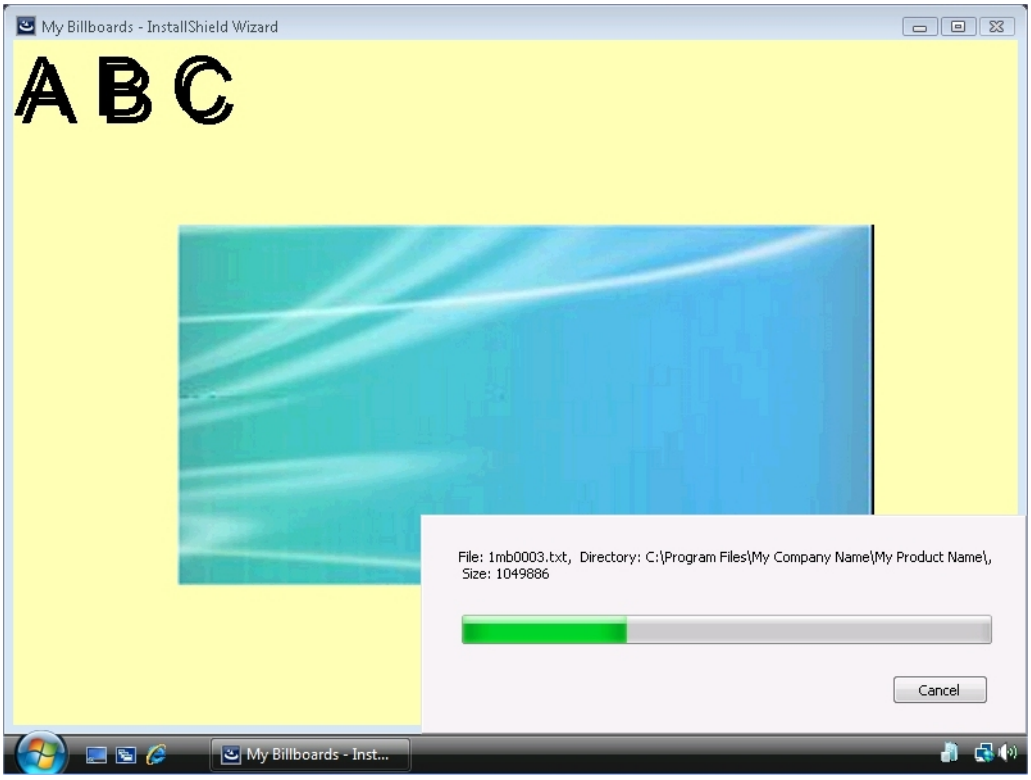


図 28: 全画面表示、右下に小さい進行状況ボックスを表示する

サンプル スクリーンショットでは、ビルボードは中央にある青緑色の長方形です。構成可能なビルボード設定の一部は次のように設定されています:

- 原点— 中央揃え
- タイトル — A B C
- フォント — 48 pt. Arial
- 背景色 — 黄色

ウィンドウ表示、標準の進行状況を表示する

ウィンドウ表示、標準の進行状況を表示するタイプののビルボードでは、 ファイルの転送中、インストールはビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。このスタイルの場合、インストールは背景を表示しません。

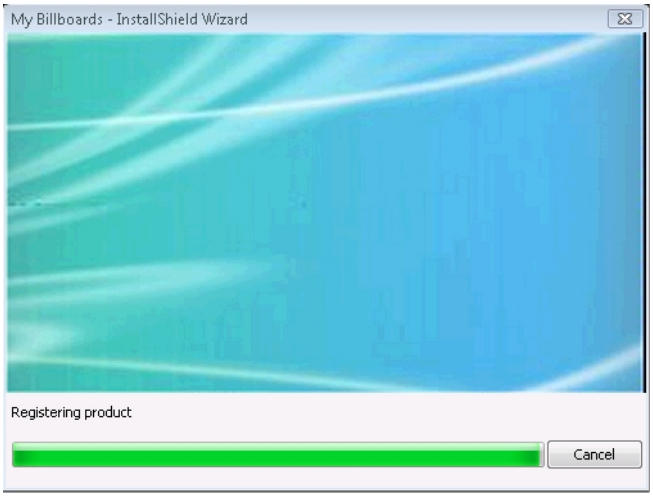


図 29: ウィンドウ表示、標準の進行状況を表示する

サンプル スクリーンショットでは、ビルボードは青緑色の長方形です。そのサイズは、幅が 544 ピクセルで、高さが 281 ピクセルです。

ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)

ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)タイプのビルボードでは、ファイル転送中にインストールは小さい進行状況ボックスを画面の右下に表示します。ビルボードまたは背景は一切表示しません。

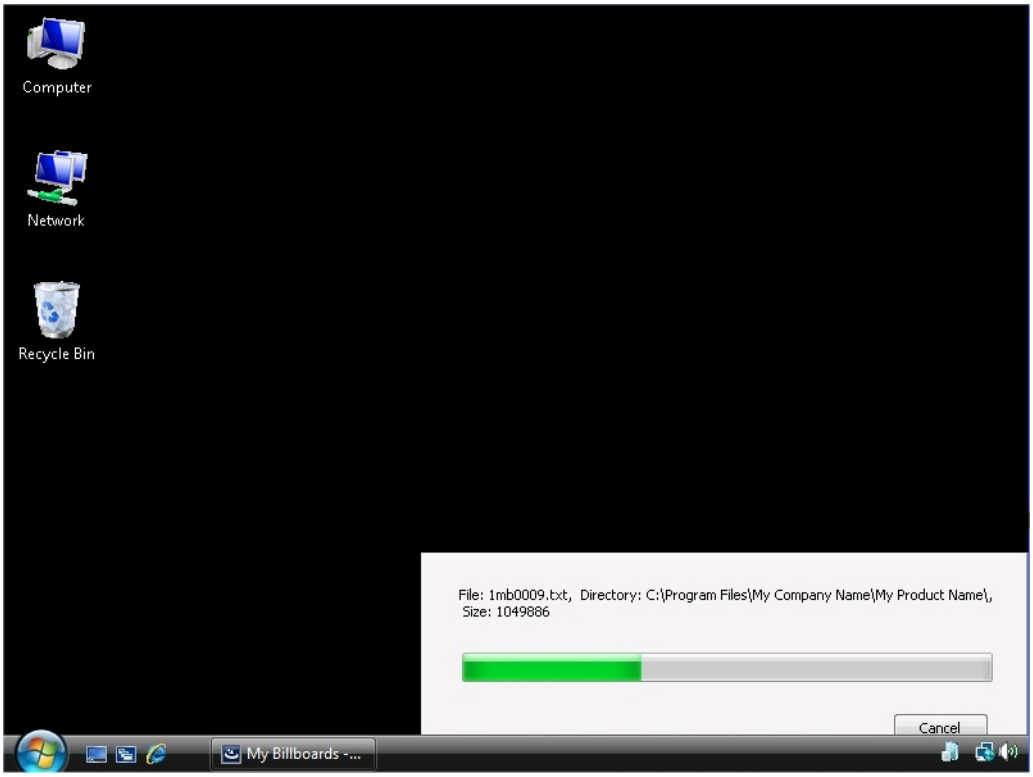


図 30: ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)

サンプル スクリーンショットに見られるように、進行状況バーが表示されますが、ビルボードは表示されません。黒色の背景は、エンド ユーザーのデスクトップです。

基本の MSI プロジェクトで使用するビルボードの種類を指定する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield は、異なる種類のビルボードをサポートします。

タスク ***インストールで使用するビルボードの種類を指定するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. 中央のペインで、**[ビルボード]** エクスプローラーをクリックします。右側のペインに **"ビルボードの種類"** 設定が表示されます。
3. **"ビルボードの種類"** 設定で、適切なビルボードの種類を選択します。

各ビルボード タイプのサンプルは、「基本の MSI プロジェクトにおけるビルボードの種類」を参照してください。

基本の MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield を使って、ファイル転送処理中に Flash アプリケーション ファイル ビルボードを表示できます。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。Flash ビデオ ファイル (.flv) や MP3 オーディオ ファイルは .swf ファイルに埋め込んで、ファイル転送中にターゲット システム上のローカルで使用できるようにすることが推奨されます。.swf ファイルは Web サイト上に配置された外部ファイルを参照することが可能ですが、この外部実装ではエンド ユーザーがインターネットに接続されていることが必須となります。

タスク ***Adobe Flash アプリケーション ファイル ビルボードをインストール プロジェクトに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、**[Adobe Flash アプリケーション ファイル(.swf)]** を右クリックしてから、**[新しいビルボード]** を選択します。新しいビルボード が **NewBillboard1** という名前で作成されます。
3. ビルボードの名前を入力します。この名前はインストールを作成するときにアイテムを識別するために使用されます。この名前はインストール時には表示されません。
4. 右側のペインで、ビルボードの設定を構成します。

メモ: *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

基本の MSI プロジェクトにイメージ ビルボードを追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ファイル転送処理中に 1 つのイメージ ビルボードのみを表示したり、一連のイメージ ビルボードで、各ビルボードが特定の時間表示されるよう設計したりすることができます。InstallShield は、.bmp、.gif、.jpg、および .jpeg イメージ ファイルをサポートします。

メモ: *動画 .gif ファイルはサポートされていません。ビルボードで動画を使用したい場合は、Adobe Flashアプリケーション ファイル ビルボードの使用をご検討ください。*

タスク ***イメージ ビルボードをインストールに追加するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、**[イメージ]** を右クリックしてから、**[新しいビルボード]** をクリックします。新しいビルボード が **NewBillboard1** という名前で作成されます。
3. ビルボードの名前を入力します。この名前はインストールを作成するときにアイテムを識別するために使用されます。この名前はインストール時には表示されません。
4. 右側のペインで、ビルボードの設定を構成します。

基本の MSI プロジェクトでビルボードの設定を構成する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

Adobe Flash アプリケーション ファイル ビルボード、またはイメージ ビルボードをプロジェクトに追加するとき、その設定を構成する必要があります。

タスク ***ビルボードの設定を構成するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. 中央ペインの **[ビルボード]** エクスプローラーで構成するビルボードを選択します。右側のペインにビルボードが表示されます。
3. 必要に応じて設定を構成します。

ビルボードの各設定についての詳細は、「Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定」を参照してください。

リリースをビルドまたは起動せずにビルボードをプレビューする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield では、リリースをビルドおよび実行せずに、実行時にビルボードがどのように表示されるのかをプレビューできます。

ビルボードをプレビューすると、そのビルボードに現在構成されている背景色、位置、および関連設定を使ったビルボードの外観を確認できます。

タスク ***ビルボードをプレビューするには、以下の手順を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. 中央ペインの **[ビルボード]** エクスプローラーでプレビューするビルボードを右クリックしてから、**[ビルボードのプレビュー]** を選択します。

InstallShield が、実行時に表示されるビルボードのプレビューを表示します。

プレビューを止めるには、[プレビュー] ウィンドウにある [キャンセル] ボタンをクリックします。

ヒント: *ビルボードのプレビューを使うと、Flash またはイメージ ビルボードが、選択された異なるビルボード タイプではどのように表示されるのかを確認するのに特に便利です。ビルボードをプレビューし、ビルボード タイプの変更してから、再度ビルボードをプレビューすることができます。*

基本の MSI プロジェクトでビルボードの順序を設定する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

イメージ ビルボードは、[ビルボード] ビューで表示されているのと同じ順序で、上から下に順番に表示されます。

タスク ***イメージ ビルボードが実行時に表示される順番を変更するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、移動するビルボードを 1 つ右クリックし、**[上に移動]** または **[下に移動]** をクリックします。

すべてのビルボードが正しく並べ替えられるまで最後のステップを繰り返します。

ビルボードを含む基本の MSI インストールにおける実行時の動作

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

重要: *インストールにビルボードが含まれている場合、インストールには* Setup.exe *セットアップランチャーを含めなくてはなりません。セットアップランチャーが実行時にビルボードを表示するため、これが必須となります。[リリース] ビューにあるリリースについての Setup.exe タブでは、セットアップランチャーを使用するかどうかなどの情報を指定することができます。詳細については、「リリースの [Setup.exe] タブ」を参照してください。*

インストールに Flash ビルボードと 1 つ以上のイメージ ビルボードが含まれている場合、実行時のファイル転送処理中に Flash ビルボードまたはイメージ ビルボードのうち 1 つのビルボード タイプのみが表示されます。

- Flash Player がターゲット システムに存在する場合、インストールは Flash ビルボードを表示します。
- Flash Player が存在しない場合、インストールはイメージ ビルボードを表示します。

実行時の動作は、インストールが Flash ビルボードかイメージ ビルボードのどちらを表示するかによって、多少異なります:

- **インストールが Flash ビルボードを表示する場合** – ファイル転送が完了すると、インストールは Flash ビルボードに割り当てられた時間が経過するまで、それを表示し続けます。割り当てられた時間が経過すると、インストールはビルボードの表示を終了して、適切な SetupComplete ダイアログを表示します。

  ファイル転送が Flash ビルボードに割り当てられた時間よりも長くかかった場合、インストールはファイル転送が終了するまで、Flash ビルボードを表示し続けます。

- **インストールがイメージ ビルボードを表示する場合** – ファイル転送が完了すると、その他のビルボードがスケジュールされていても、また現在のビルボードに割り当てられた時間が経過していなくても、インストールはイメージ ビルボードの表示を終了します。次に、インストールは適切な SetupComplete ダイアログを表示します。

  ファイルの転送時間が、ビルドボードに割り当てられた時間を超える場合、インストールはファイル転送が終了するまでビルボードを表示し続けます。[ビルボード] ビューの "ビルボードのループ" 設定に [いいえ] が選択されている場合、インストールがファイルの転送を終了する前に最後のビルボードに到達したとき、インストールはファイルの転送が終了するまで最後のイメージ ビルボードを表示し続けます。次に、インストールは適切な SetupComplete ダイアログを表示します。この設定に [はい] が選択されている場合、インストールがファイルの転送を終了する前に最後のビルボードに到達したとき、インストールは最初のビルボードから再び表示を開始します。必要な場合、ファイルの転送が終了して SetupComplete ダイアログが表示されるまでループが継続します。

メモ: *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*

基本の MSI プロジェクトからビルボードを削除する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

タスク ***インストールからビルボードを削除するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ビルボード]** をクリックします。
2. **[ビルボード]** エクスプローラーで、削除するビルボードを右クリックして **[削除]** を選択します。

InstallScript および InstallScript MSI インストールでビルボードを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield は、2 種類のビルボードをサポートします。1 つめは、全画面の背景ウィンドウを含み、2 つめはこれを含みません。両方のスタイルで進行状況バーが表示されます。

以下に、両方の種類のビルボードの説明とサンプル、および適切なサポート対象ファイルの種類を示します。

ウィンドウ表示のビルボードで進行状況を表示する

このスタイルのビルボードでは、インストールが標準サイズのダイアログを表示します。ファイルの転送中、ダイアログ上部にビルボードが表示され、ダイアログの下部に進行状況バーが表示されます。このスタイルのビルボードの場合、インストールは背景ウィンドウを表示しません。

このスタイルのビルボードでは、以下の種類のファイルを使用できます:

- Adobe Flash アプリケーション ファイル (.swf)
- イメージ (.bmp、.gif、.jpg、および .jpeg)

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

メモ: *進行状況を表示するウィンドウ表示のビルボードでは、プロジェクトに複数のイメージ ビルボードを追加することができますが、Adobe Flash アプリケーション ファイル ビルボードの場合は 1 つだけしか追加できません。*

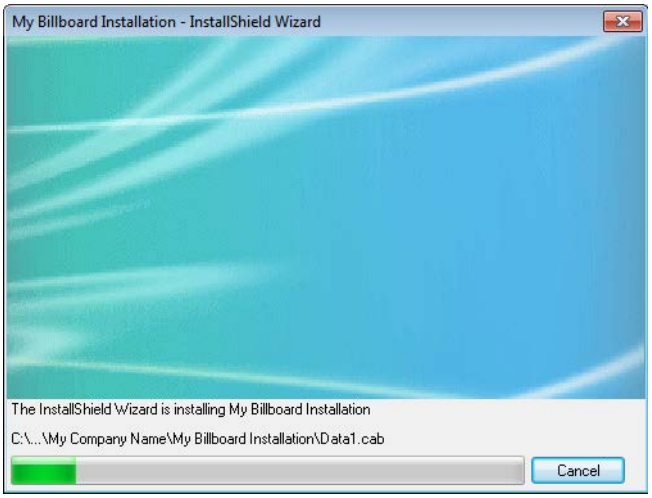


図 31: ウィンドウ表示のビルボードで進行状況を表示する

サンプル スクリーンショットでは、ビルボードは青緑色の長方形です。そのサイズは、幅が 544 ピクセルで、高さが 281 ピクセルです。

メモ: *スキン ダイアログではウィンドウ表示のビルボードを使用できません。スキンについての詳細は、「ダイアログ スキン」を参照してください。*

全画面背景ウィンドウ表示ビルボード

このスタイルのビルボードでは、インストールが標準のエンド ユーザー ダイアログを表示するとき、全画面の背景ウィンドウも表示します。ファイルの転送中、全画面の背景ウィンドウの手前にビルボードが表示され、標準サイズの進行状況ダイアログがビルボードの手前に表示されます。このビルボード スタイルには、背景ウィンドウは不要です。

このスタイルのビルボードでは、イメージ ファイル (.bmp、.gif、.jpg、および .jpeg) を使用できます。

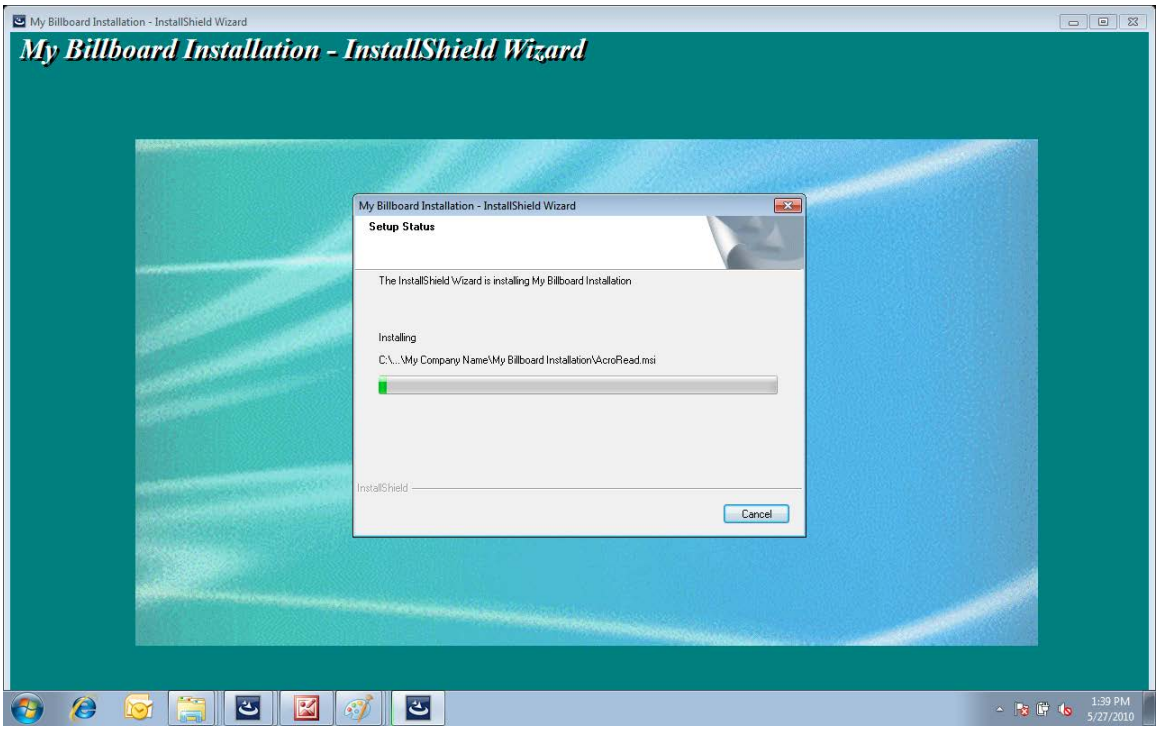


図 32: 全画面背景ウィンドウ表示ビルボード

サンプル スクリーンショットでは、ビルボードは進行状況ダイアログの後ろにある青緑色の長方形イメージで、背景ウィンドウの手前にあります。

InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *ここで説明されているAdobe Flash アプリケーション ファイル サポートは、使用するビルボード スタイルがウィンドウ表示スタイルで進行状況を含む場合に適用します。全画面背景ウィンドウ表示ビルボード スタイルには適用しません。Flash ファイル サポートを全画面表示で背景ウィンドウを含むスタイルと共に使用するには、* **PlayMMedia** *関数を使用します。このサポートを使うと、以下の命名規則は Adobe Flash ファイルに適用しません。詳細については、「PlayMMedia」を参照してください。この実装で背景ウィンドウを表示する方法については、「InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する」をご参照してください。*

プロジェクトに複数のイメージ ビルボードを追加することができますが、Adobe Flash アプリケーション ファイル ビルボードの場合は 1 つだけしか追加できません。

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

ビルボードをプロジェクトに追加する最初の手順は、適切な場合は Adobe Flash アプリケーション ファイルと、インストールのビルボードとして使用するイメージ ファイル (.bmp、.gif、.jpg、または .jpeg) の作成です。ファイルに正しい名前を付けたら、ビルボードをプロジェクトに追加できます。Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

ビルボードファイルには、指定の命名規則を使用する必要があります。各ファイル名は **bbrd** で始まり、ビルボード番号 (1 から 99) が続きます。最後に、サポート対象のファイル拡張子 (.swf、.bmp、.gif、.jpg、または .jpeg) で終わります。プロジェクトには、最多で 1 つの Adobe Flash ファイル (.swf) を含むことができます。.swf ファイルを含める場合、その名前にはプロジェクトに含まれるビルボード ファイルのリストで 1 番低い番号を含めなくてはなりません (たとえば、bbrd1.swf)。

ビルボードは、ファイル名の順番に表示されます。たとえば、bbrd2.bmp といい名前のビルボード ファイルは、bbrd4.gif の前に表示されます。

ファイル名番号が連続番号である必要はありません。つまり、プロジェクトに bbrd1.jpg、bbrd3.jpg、および bbrd5.jpg を追加すると、実行時に各イメージが順番通りに表示されます。

ヒント: *各イメージ ビルボードを表示する時間は、インストール プロジェクトに含まれているイメージ ビルボードの数によって異なります。表示時間の割合は、おおよそ 1 をビルボードの数で割ったものです。たとえば、インストールに 4 つのビルボードがある場合、各ビルボードが表示される時間は、ファイル転送時間の 25% の長さになります。*

InstallScript または InstallScript MSI プロジェクトに Adobe Flash アプリケーション ファイル ビルボードを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *ここで説明されているAdobe Flash アプリケーション ファイル サポートは、使用するビルボード スタイルがウィンドウ表示スタイルで進行状況を含む場合に適用します。全画面背景ウィンドウ表示ビルボード スタイルには適用しません。Adobe ファイル サポートを全画面の背景ウィンドウと共に使用する場合、**PlayMMedia** 関数を使用します。詳細については、「PlayMMedia」を参照してください。この実装で背景ウィンドウを表示する方法については、「InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する」をご参照してください。*

InstallShield を使って、ファイル転送処理中に Flash アプリケーション ファイル ビルボードを表示できます。Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。Flash ビデオ ファイル (.flv) や MP3 オーディ

オ ファイルは .swf ファイルに埋め込んで、ファイル転送中にターゲット システム上のローカルで使用できるようにすることが推奨されます。.swf ファイルは Web サイト上に配置された外部ファイルを参照することが可能ですが、この外部実装ではエンド ユーザーがインターネットに接続されていることが必須となります。

タスク ***Adobe Flash アプリケーション ファイル ビルボードをインストール プロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポートファイル]** エクスプローラーで、ビルボード (**言語非依存**、または言語固有のどちらでも可) を含む **[ビルボード]** 項目をクリックします。右側に [**ファイル**] ペインが表示されます。
3. **[ファイル]** ペインの任意の場所を右クリックして、**[ファイルの挿入]** をクリックするか、**[ファイル]** ペインにカーソルを置いて Insert キーを押します。**[開く]** ダイアログが開きます。
4. **bbrd1.swf** と名づけられた Adobe Flash アプリケーション ファイル ビルボード ファイルを選択してから、**OK** をクリックします。

ファイルがプロジェクトへ追加されます。

メモ: *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*

Flash アプリケーション ファイルを表示するために必要な Adobe Flash Player がターゲット システムに存在しない場合、インストールはそれを検知して Flash ビルボードの代わりにイメージ ビルボードを表示します。このため、Flash ビルボードをプロジェクトに含める場合は、1 つ以上のイメージ ビルボードもプロジェクトに含めることが推奨されます。

InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ファイル転送処理中に 1 つのイメージ ビルボードのみを表示したり、一連のイメージ ビルボードで、各ビルボードが特定の時間表示されるよう設計したりすることができます。InstallShield は、.bmp、.gif、.jpg、および .jpeg イメージ ファイルをサポートします。

メモ: *動画 .gif ファイルはサポートされていません。ビルボードで動画を使用したい場合は、Adobe Flashアプリケーション ファイル ビルボードの使用をご検討ください。*

タスク ***イメージ ビルボードをプロジェクトに追加するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポートファイル]** エクスプローラーで、ビルボード (**言語非依存**、または言語固有のどちらでも可) を含む **[ビルボード]** 項目をクリックします。右側に [**ファイル**] ペインが表示されます。
3. **[ファイル]** ペインの任意の場所を右クリックして、**[ファイルの挿入]** をクリックするか、**[ファイル]** ペインにカーソルを置いて Insert キーを押します。**[開く]** ダイアログが開きます。
4. ビルボード ファイルを選択して [**OK**] をクリックします。

ファイルがプロジェクトへ追加されます。

ヒント: *各イメージ ビルボードを表示する時間は、インストール プロジェクトに含まれているイメージ ビルボードの数によって異なります。表示時間の割合は、おおよそ 1 をビルボードの数で割ったものです。たとえば、インストールに 4 つのビルボードがある場合、各ビルボードが表示される時間は、ファイル転送時間の 25% の長さになります。*

InstallScript または InstallScript MSI プロジェクトでコードを追加または変更してビルボードを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

実行時にビルボードを表示するために必要な InstallScript コードは、使用するビルボードのスタイルによって異なります。

- **ウィンドウ表示のビルボードで進行状況を表示する**—実行時に、進行状況を表示するウィンドウ表示のビルボードを表示するには、STATUSBBRD 定数を**Enable** 関数と一緒に使います。
- **全画面背景ウィンドウ表示ビルボード**—全画面背景ウィンドウと共に表示されるビルボードを表示するには、FULLWINDOWMODE および BACKGROUND 定数を **Enable** 関数と一緒に使います。詳しくは、「InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する」をご覧ください。

  Adobe Flash ビルボードをこのスタイルのビルボードと共に使用する場合、**PlayMMedia** 関数も使用する必要があります。

これらの異なるスタイルのビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。

InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

イメージ ビルボードは、ファイル名の数字の順番に表示されます。bbrd2.bmp という名前のビルボードファイルは、bbrd3.bmp より先に、また bbrd1.bmp というファイルの後に表示されます。命名規則についてのガイドラインは、「InstallScript または InstallScript MSI プロジェクトにおけるビルボード ファイルの名前」を参照してください。

タスク ***ビルボードが表示される順番を変更するには、以下の手順を実行します。***

1. 順番を変更するビルボード ファイルを削除します。(詳細については、「InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する」を参照してください。
2. Windows エクスプローラーを使用して、適切な順番に来るようにビルボード ファイルの名前を変更します。
3. 「InstallScript または InstallScript MSI プロジェクトにイメージ ビルボードを追加する」の説明にしたがって、名前を変更したファイルを追加します。

ファイル名番号が連続番号である必要はありません。つまり、プロジェクトに bbrd1.jpg、bbrd3.jpg、および bbrd5.jpg を追加すると、実行時に各イメージが順番通りに表示されます。

InstallScript または InstallScript MSI インストール中に特殊効果を持つビルボードを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

SetDisplayEffect 関数を使うと、ビルボードが最初にメインのインストールウィンドウに表示されたときに特殊効果を使うことができます。ファイル転送中に **PlaceBitmap** を使ってビットマップを表示するか、ビルボードを表示する前に、この関数オプションの 1 つを選択します。

InstallScript または InstallScript MSI プロジェクトでビルボードを画面上の異なる場所に移動させる

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

メモ: *ここで説明されているAdobe Flash アプリケーション ファイル サポートは、使用するビルボード スタイルが全画面背景ウィンドウ表示ビルボードの場合 (***PlayMMedia** *関数を使用します) に適用します。ウィンドウ表示スタイルで進行状況を含む場合 (STATUSBBRD 定数を* **Enable** *関数と共に使用します) には適用しません。これらの 2 つのスタイルのビルボードについての詳細は、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。*

ビルボードはデフォルトでメイン インストール ウィンドウの中央に表示されます。別の場所を指定するには、BILLBOARD オプションと一緒に **PlaceWindow** 関数を呼び出します。たとえば画面左上隅から 10 ピクセルの所にビルボードを配置するには、次のように呼び出します。

```
PlaceWindow (BILLBOARD, 10, 10, UPPER_LEFT);
```

ビルボードはファイル転送中にのみ表示されるため、ファイル転送を開始する前に **PlaceWindow** を呼び出すようにしてください。

InstallScript または InstallScript MSI プロジェクトでビルボードの順序を設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

タスク ***インストール プロジェクトからビルボードを削除するには次の操作を実行します。***

1. ビュー リストの **[動作とロジック]** の下にある **[サポート ファイル/ビルボード]** をクリックします。
2. **[サポートファイル]** エクスプローラーで、削除するビルボード (**言語非依存**、または言語固有のどちらでも可) を含む **[ビルボード]** 項目をクリックします。右側に [**ファイル**] ペインが表示されます。
3. ビルボード ファイルを右クリックして、**[削除]** をクリックします。

InstallScript および InstallScript MSI インストールで背景ウィンドウを表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

デフォルトで、インストールは背景ウィンドウを表示しません。**PlayMMedia** 関数を使って Adobe Flash アプリケーション ファイル (.swf) または AVI ファイルを表示するためには、インストールが背景ウィンドウを表示する必要があります。また、1 つの種類のビルボードに対してのみ背景ウィンドウの表示が必要な場合もあります。

ヒント: *Flash ファイル ビルボードおよびイメージ ビルボードは、背景ウィンドウなしで表示できます。詳細については、「InstallScript および InstallScript MSI プロジェクトにおけるビルボード スタイルおよびファイルの種類」を参照してください。*

背景ウィンドウを表示する方法は、使用するプロジェクトの種類によって異なります。

InstallScript プロジェクト

InstallScript プロジェクトで背景ウィンドウを表示するには、次のコード行の初めについている 2 つのスラッシュ (//) を削除して、スクリプトの **OnShowUI** イベント ハンドラー関数を修正します:

```
//if ( LoadStringFromStringTable( "TITLE_MAIN", szTitle ) < ISERR_SUCCESS ) then // タイトル文字列のロード
//    szTitle = IFX_SETUP_TITLE;
//endif;
//SetTitle( szTitle, 24, WHITE );
//Enable( FULLWINDOWMODE );
// Enable( BACKGROUND );
//SetColor( BACKGROUND, RGB( 0, 128, 128 ) );
```

InstallScript MSI プロジェクト

InstallScript プロジェクトで背景ウィンドウを表示するには、次のコード行の初めについている 2 つのスラッシュ (//) を削除して、適切なスクリプトの UI イベント ハンドラー (たとえば、**OnFirstUIBefore** または **OnMaintUIBefore**) を修正します:

```
// SetTitle( @PRODUCT_NAME, 24, WHITE );
// SetTitle( @PRODUCT_NAME, 0, BACKGROUNDCAPTION );
// Enable( FULLWINDOWMODE );
// Enable( BACKGROUND );
// SetColor(BACKGROUND,RGB (0, 128, 128));
```

実行時にリスト ボックスへデータを挿入する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

基本の MSI プロジェクト

実行時にリストボックスのアイテムを設定するには、SQL クエリを使って .msi データベース用の一時レコードを作成する必要があります。具体例については、Revenera コミュニティの「Populating A List Box Dynamically At Run Time (実行時にリストボックスを動的に入力する)」を参照してください。

詳細は、Revenera コミュニティの「Accessing the MSI Database at Run Time (実行時にMSIデータベースにアクセスする)」を参照してください。

InstallScript MSI プロジェクト

CtrlSetList 関数は、ダイアログ上の ListBox コントロールに、文字列リスト変数を関連付けます。例:

```
function FillListBox( )
    LIST listDays;
begin
    listDays = ListCreate(STRINGLIST);
    ListAddString(listDays, "Monday", AFTER);
    ListAddString(listDays, "Wednesday", AFTER);
    ListAddString(listDays, "Friday", AFTER);
```

```
    CtrlSetList("DialogName", nListBoxId, listDays);
end;
```

ファイル参照ダイアログを表示する

GetOpenFileName API を呼び出して、ファイル参照ダイアログを表示する例については、ナレッジベース記事 Q104325 を参照してください。

InstallScript インストールでネットワーク参照ダイアログを表示する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

SelectDirEx 関数を利用して、ユーザーがネットワークコンピューターを参照できるダイアログを表示することができます。

```
prototype NetBrowse( );

function NetBrowse( )
    STRING svSelectedDir[MAX_PATH + 1];
    NUMBER nReturn;
begin
    svSelectedDir = PROGRAMFILES;

    nReturn = SelectDirEx("", "Select a directory:", "", "",
        BIF_EDITBOX | BIF_RETURNONLYFSDIRS, svSelectedDir);

    if (nReturn = OK) then
        MessageBox("選択したディレクトリは: " + svSelectedDir, INFORMATION);
    elseif (nReturn = CANCEL) then
        MessageBox("ユーザーがキャンセルをクリック", INFORMATION);
    else
        MessageBox("ダイアログ表示エラー", WARNING);
    endif;
end;
```

サーバーの構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インストールを作成しているとき、ターゲット システムにインストールされるテクノロジに対してサーバー側のサポートを提供する必要があることに気がつくことがあります。InstallShield では、サーバー側のインストールまたは COM+ サーバー アプリケーションのアプリケーション プロキシ管理を簡単に構成することができます。InstallShield では、インターネット インフォメーション サービス、SQL、およびコンポーネント サービスがサポートされています。

このセクションには、次のトピックが含まれています:

- SQL サポートの構成
- スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する
- COM+ アプリケーションとコンポーネントの管理
- インターネット インフォメーション サービス

SQL サポートの構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

メモ: *スイート/アドバンスト UI プロジェクトにおける SQL サポートの構成についての詳細は、「スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する」を参照してください。*

InstallShield では、Microsoft SQL Server、Microsoft Windows Azure、MySQL、および Oracle のサポートが提供されています。[SQL スクリプト] ビューは、ユーザー インターフェイス内ですべての SQL スクリプトをサーバー接続および設定によって管理並びに編成するための拠点です。InstallShield 内の SQL サポートにより、次の処理が可能です。

- SQL サーバーへ接続する。
- カタログスキーマおよび/またはデータをインポートする。

- SQL スクリプトとコンポーネントを関連付ける。
- 必要な SQL サーバー/スクリプトプロパティ（サーバー名、データベース名、認証メソッドなど）を設定する。
- インストールまたはアンインストール中に実行するための SQL スクリプトの設定する。
- SQL スクリプトを編集する。
- SQL Server、MySQL、または Oracle の特定バージョンを要求またはターゲットする、もしくはその両方。
- SQL スクリプトのテキスト置換を定義する。
- Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer でスクリプトを開く。

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。Oracle ユーザーの方は、Oracle データベース ユーティリティの Oracle Web ページで、InstallShield と共に利用可能なユーティリティについての情報をご覧いただけます。*

使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:

1. *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
2. *Microsoft SQL Server 2005 Management Studio (*SqlWb.exe*)*
3. *Microsoft SQL Server 2005 Management Studio Express (*ssmsee.exe*)*
4. *Microsoft SQL Server 2000 Query Analyzer (*isqlw.exe*)*

SQL ログインの設定に Windows Installer プロパティを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

[SQL スクリプト] ビューを使って新しい SQL データベース接続をプロジェクトに追加すると、InstallShield はデフォルトで次の Windows Installer プロパティをプロジェクトに追加します。

テーブル 1 • デフォルト SQL ログインのプロパティ

プロパティ	説明
IS_SQLSERVER_AUTHENTICATION	このプロパティは、指定されたカタログの接続に使用する認証の種類を識別します。デフォルトのプロパティは IS_SQLSERVER_AUTHENTICATION です。プロパティ値には、次の数値が有効です: • 0 – 現在のユーザーの Windows 認証情報 • 1 – サーバー認証 このプロパティは、選択された SQL 接続の [詳細] タブにある "認証タイプのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_DATABASE	このプロパティは、インストール中にそのプログラムへの接続を作成する SQL カタログの名前を識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "ターゲット カタログのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_PASSWORD	このプロパティは、サーバー認証に使用するパスワードを識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "サーバー認証パスワードのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_SERVER	このプロパティは、(Microsoft SQL Server および MySQL の場合) ターゲット サーバーのインスタンス名、または (Oracle の場合) URL 接続文字列またはローカル ネット サービス名を識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "ターゲット サーバーのプロパティ名" 設定のデフォルト値です。
IS_SQLSERVER_USERNAME	このプロパティは、サーバー認証に使用するログイン ID を識別します。このプロパティは、選択された SQL 接続の [詳細] タブにある "サーバー認証ログイン ID のプロパティ名" 設定のデフォルト値です。

既存の接続でこれらの Windows Installer プロパティの 1 つをオーバーライドする場合、プロパティ マネージャーで新しいプロパティを追加します。そのあと、[SQL スクリプト] ビューで、その接続を選択します。[詳細] タブで、該当の一覧から新しいプロパティの名前を選択します。

製品が後で使用できるように、ターゲットシステム上にこれらの任意のプロパティの値を保存したい場合、それが可能です。以下は、これらのプロパティの使用例です:

- [レジストリ] ビューで、データが **[IS_SQLSERVER_SERVER]** のレジストリ値を作成します。実行時に、インストールがレジストリ値を作成するとき、そのレジストリ値のデータは、SQL カタログの名前に設定されます。
- [テキスト ファイルの変更] ビューを使って、実行時に処理を行うテキスト文字列の置換を構成します。このビューで、製品と共にインストールされるファイルを説明するテキスト ファイル リファレンスを追加してから、検索/置換の基準を指定します。検索/置換の基準には、SQL Server マシンの名前の場所に **[IS_SQLSERVER_DATABASE]** と入力します。実行時、インストールがテキスト ファイルを編集するとき、SQL Server マシンの名前がテキストファイルに書き込まれます。

ヒント: *デフォルトの SQL ランタイム動作をオーバーライドするためにプロジェクトで定義づけることが可能な追加 Windows Installer プロパティのリストは、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。*

プロジェクト: *基本の MSI インストールでは、エンド ユーザーはビルトイン SQLLogin ダイアログを使って、前述のプロパティを構成できます。基本の MSI プロジェクトで SQL 接続の [詳細] タブにある SQL プロパティのいずれかを変更した場合に、[ダイアログ] ビューの SQLLogin ダイアログ内にある対応するプロパティが自動的に更新されることはありません。従って、ダイアログのプロパティを手動で変更して、SQL 接続の [詳細] タブで選択したプロパティと一致させる必要があります。*

新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

InstallShield では、デフォルトで、プロジェクトに追加した新しい SQL データベース接続が、同じ Windows Installer のプロパティを共有するかどうかを指定することができます。

たとえば、すべての接続で同じ Windows Installer のデフォルト プロパティを共有させたいとき、[SQL スクリプト] ビューで接続の 1 つをプロジェクトに追加し、**MyConnection** のカタログ名を指定することができます。2 つ目の接続をプロジェクトに追加したとき、その同じ **MyConnection** のカタログ名が 2 つ目の接続に使用されます。いずれか一方のカタログ名を変更すると、両方とも同じ Windows Installer プロパティに基づいているため、もう一方の接続のカタログ名も自動的に更新されます。

タスク

すべての SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定するには、以下の手順に従います。

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[SQL スクリプト]** タブをクリックします。
3. **[新しい接続に一意の Windows Installer プロパティを生成する]** チェック ボックスを目的に従って選択またはクリアします:
   - 追加したすべての新しい接続で Windows Installer プロパティを共有する場合、このチェック ボックスをクリアします。
   - 追加した新しい接続で異なる Windows Installer プロパティを使用する場合、このチェック ボックスを選択します。

チェック ボックスが選択されている状態で 2 つ目の接続をプロジェクトに追加すると、その 2 つ目の接続に新しい Windows Installer プロパティ セットが作成されます。

チェック ボックスがクリアされている状態で 2 つ目の接続を追加すると、その 2 つ目の接続にデフォルトの Windows Installer プロパティが使用されます。Windows Installer のデフォルトのプロパティは、プロジェクトに追加した 1 つ目の接続用に追加されたプロパティです。

ヒント: *既存の接続の Windows Installer プロパティをオーバーライドする場合、プロパティ マネージャーで新しいプロパティを追加します。そのあと、[SQL スクリプト] ビューで、その接続を選択します。[詳細] タブで、該当の一覧から新しいプロパティの名前を選択します。*

プロジェクト: *基本の MSI プロジェクトの ビルトイン SQLLogin ダイアログは、プロジェクトに追加された 1 つ目の SQL 接続用の Windows Installer のデフォルト プロパティと共に動作するようにデザインされています。**[新しい接続に一意の Windows Installer プロパティを生成する]** オプションを選択した場合、新しい Windows Installer のプロパティと動作するように、各接続ごとにダイアログを複製する必要があります。*

InstallScript と InstallScript MSI プロジェクトで SQL ランタイム関数を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallScript 言語には、*SQLRT* というプレフィックスで始まる多数のビルトイン SQL ランタイム (SQLRT) 関数が含まれています。一部の関数は InstallScript プロジェクトでのみ、または InstallScript MSI プロジェクトでのみ提供されています。また、両方のプロジェクトで提供されている関数もあります。

SQLRT 関数は、SQL 情報を [SQL スクリプト] ビューで構成する必要があります。この構成情報は、SQL ランタイム関数が適切に動作するように、SQLRT.ini ファイルに書き込まれます。InstallScript プロジェクトの場合、SQLRT 関数は SQLRT.obl ファイルに存在し、SQLRT.dll ファイルを呼び出します。InstallScript MSI プロジェクトの場合、SQLRT 関数は SQLCONV.obl ファイルに存在し、ISSQLSRV.dll ファイルを呼び出します。これらのサポート ファイルは、[SQL スクリプト] ビューを使用したとき、自動的にプロジェクトに追加されます。

SQLRTInitialize2 関数は SQL サーバー ランタイムを初期化します。InstallScript プロジェクトの場合、**SQLRTInitialize2** 関数は、OnSQLServerInitialize イベント ハンドラーの実行中、自動的に呼び出されます。InstallScript MSI プロジェクトの場合、**SQLRTInitialize2** 関数は、OnSQLLogin イベント ハンドラーの実行中、自動的に呼び出されます。SQLRT 関数の 1 つを OnSQLServerInitialize または OnSQLLogin イベントの前に呼び出す必要がある場合、**SQLRTInitialize2** 関数をまず呼び出す必要があります。

メモ: *InstallShield の以前のバージョンでは、**SQLRTInitialize** 関数は InstallScript プロジェクトで、SQL サーバー ランタイムを初期化するために使用されていました。この関数は、**SQLRTInitialize2** 関数によって置き換えられました。*

新しい SQL 接続の追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

SQL スクリプト ビューでは、スクリプトは接続によって編成されます。これは接続が確立されるまでスクリプトがサーバー上で実行することができないためです。したがって、プロジェクトに SQL Script を追加する前に、まず SQL 接続を作成する必要があります。

タスク

新しい SQL 接続を作成するには、以下の手順を実行します:

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。

エクスプローラーに、新しい接続が追加されます。右のペインにあるタブを利用して、この接続に関連付けられている設定を構成します。

メモ: *SQLLogin と SQLBrowse ダイアログを通して、エンド ユーザーはエイリアス名を使って SQL Server データベースに接続および参照できます。*

Microsoft から Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) と呼ばれる新しい OLE DEB プロバイダーがリリースされました。今後、新しいプロバイダーは最新のサーバー機能で更新されます。このプロバイダーは、TLS 1.2 のみの環境もサポートします。

InstallShield 2019 R3 以降は、すべての新しい SQL Server 接続にデフォルトとして MSOLEDBSQL を使用します。これは、[要件] タブで [データベース サーバーのドライバーを選択します] ドロップダウンから異なる SQL ドライバーを選択して変更することができます。

重要: *Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) は、InstallShield 2019 R3 以降に前提条件として含まれています。*

詳細については、[要件] タブ の「データベース サーバーの SQL ドライバーを選択する」を参照してください。

SQL Server データベースのデフォルトの TCP/IP ネットワーク ライブラリを別のプロトコルで上書きする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*

デフォルトで、InstallShield インストールは SQL Server データベースへの接続時に、TCP/IP ネットワーク ライブラリを使用します。別のプロトコルを使用する場合、必要に応じてこのデフォルト動作をオーバーライドすることができます。

タスク ***SQL Server データベースへの接続時に、デフォルトの TCP/IP ネットワーク ライブラリを上書きするには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。エクスプローラーに、新しい接続が追加されます。右のペインにあるタブを利用して、この接続に関連付けられている設定を構成します。
3. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
4. **[テーブル]** エクスプローラーで、Property テーブルをクリックします。
5. IS_SQLSERVER_NETLIB_MS フィールドを見つけます。デフォルトの値は次のとおりです。

   Network Library=DBMSSOCN

   DBMSSOCN は、TCP/IP ネットワーク ライブラリのモジュール名 (拡張子を除く) を示します。
6. 必要に応じて、デフォルトの DBMSSOCN 名を置き換えます。例:
   - Named Pipes ネットワーク ライブラリを使用するには、**DBNMPNTW** を指定します。
   - SPX/IPX を使用するには、**DBMSSPXN** を指定します。
   - Banyan Vines を使用するには、**DBMSVINN** を指定します。
   - Multi-Protocol (Windows RPC) を使用するには、**DBMSRPCN** を指定します。

実行時に、SQL Server データベースに接続する際、指定されたプロトコルが使用されます。

SQL Server Express LocalDB のインスタンスに接続するときの要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

インストールで、SQL Server Express LocalDB のインスタンスへの接続をサポートする場合、ターゲット システム上に、SQL Server Native Client 11 ODBC ドライバーが存在している必要があります。当該のドライバーが存在していることを確認するために、インストールに Microsoft SQL Server 2012 Native Client 前提条件を含めることができます。このドライバーを使用するには、プロジェクトの構成も必要になります。

タスク ***SQL Server Express LocalDB のインスタンスへの接続をサポートするために、プロジェクトに適切な SQL Server Native Client 前提条件を含めるには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. 再配布可能ファイルのリストで、[**Microsoft SQL Server 2012 Native Client**] チェック ボックスを選択します。

InstallShield 前提条件ファイル (.prq) で定義された条件が満たされた場合のみ、InstallShield 前提条件が起動します。条件の一覧を参照するには、[再配布可能ファイル] ビューまたは [前提条件] ビューにある SQL Server Native Client 前提条件をクリックしてから、右側の詳細ペインに表示される説明を確認します。これらのビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。

タスク ***InstallShield インストール プロジェクトを構成して、インストールで、SQL Server Native Client 11 ODBC ドライバーが使用されるようにするには、以下の手順に従います::***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、ISSQLDBMetaData テーブルをクリックします。
3. **[ISSQLDBMetaData]** 列で、まず **[MSSQLServer]** 行を見つけます。
4. **[AdoDriverName]** 列の値を、次の値で置き換えます:

   sqlncli11

実行時、SQL Server Express LocalDB データベースへに接続する際、SQL Server Native Client 11 ODBC ドライバーが使用されます。

新しい SQL スクリプトの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

タスク ***新しい SQL 接続の作成後、新しい SQL スクリプトを追加するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. 新しい接続を右クリックしてから、**[新しいスクリプト]** をクリックします。

新しいスクリプトをプロジェクトに追加すると、新しいコンポーネントがそのスクリプトに追加されます。このスクリプトを機能に関連付けます。機能が存在しない場合、SQL スクリプトを追加したときに [新しい機能の作成] ダイアログ ボックスが表示され、新しい機能を作成するように要求してきます。スクリプトと機能間の関連付けは、[SQL スクリプト] ビューにある [全般] タブの **[SQL スクリプトが属する機能の選択]** 領域を利用して後で変更することができます。

ヒント: *また、それらをインポートまたは挿入することで、プロジェクトにスクリプトを追加することもできます。詳細については、「SQL スクリプトの挿入およびインポート」を参照してください。*

メモ: *使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:*

- *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
- *Microsoft SQL Server 2005 Management Studio (*SqlWb.exe*)*
- *Microsoft SQL Server 2005 Management Studio Express (*ssmsee.exe*)*
- *Microsoft SQL Server 2000 Query Analyzer (*isqlw.exe*)*

SQL スクリプトの挿入およびインポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、複数のプロジェクトで SQL スクリプト ファイル (.sql) を再利用することができます。[SQL スクリプト] ビューで、プロジェクトにスクリプト ファイルを挿入したりインポートしたりすることができます。

- スクリプト ファイルを挿入すると、ファイルの現在の保存場所へのリンクが作成されます。
- スクリプト ファイルをインポートすると、プロジェクトのスクリプト ファイルを格納しているフォルダーにそのファイルがコピーされます。インポートしたスクリプト ファイルは、システム上またはリポジトリに保存することができます。

SQL スクリプト ファイルの挿入

タスク ***SQL スクリプト ファイルを挿入するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. SQL 接続をまだ追加していない場合、**[SQL スクリプト]** エクスプローラーで、SQL 接続の追加を行います。
3. SQL 接続を右クリックしてから、**[スクリプト ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 挿入する SQL スクリプト ファイル (.sql) を選択します。
5. **[開く]** をクリックします。

SQL スクリプト ファイルのインポート

タスク ***SQL スクリプト ファイルをインポートするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. SQL 接続をまだ追加していない場合、**[SQL スクリプト]** エクスプローラーで、SQL 接続の追加を行います。
3. SQL 接続を右クリックしてから、**[スクリプト ファイルのインポート]** をクリックします。[SQL スクリプト ファイルのインポート] ダイアログ ボックス が開きます。
4. 以下のいずれかを実行します。
   - **[レポジトリ アイテム]** ボックスで、プロジェクトに追加する SQL スクリプト ファイル (.sql) をクリックします。
   - インポートするスクリプト ファイルがリポジトリに格納されていない場合、[参照] ボタンをクリックしてファイルを選択します。
5. **[OK]** をクリックします。

新しいスクリプトをプロジェクトに挿入またはインポートすると、新しいコンポーネントがそのスクリプトに追加されます。このスクリプトを機能に関連付けます。機能が存在しない場合、SQL スクリプトを追加したときに [新しい機能の作成] ダイアログ ボックスが表示され、新しい機能を作成するように要求してきます。スクリプトと機能間の関連付けは、[SQL スクリプト] ビューにある [全般] タブの **[SQL スクリプトが属する機能の選択]** 領域を利用して後で変更することができます。

SQL Server データベースのインポートと SQL スクリプト ファイルの生成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*

タスク ***既存の Microsoft SQL Server データベースをインポートして、それからスクリプト ファイルを生成するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックしてから、**[データベース インポート ウィザード]** をクリックします。**データベース インポート ウィザード**が開きます。
3. データベース インポート ウィザードのパネルを完成します。

データベース インポート ウィザードが起動してデータベース設定のインポート手順を案内し、これらの設定および選択されたオプションに基づいて SQL スクリプト ファイルを生成します。

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。データベース インポート ウィザードで生成されたスクリプトは、他の種類のデータベースとは互換性を持ちません。*

SQL スクリプト ファイルの編集

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

スクリプト ファイルは、作成から挿入、インポートまですべて終えた時点で、InstallShield 内で編集することができます。

プロジェクト: *InstallScript プロジェクトで、InstallShield にアップグレードする前に、OnFirstUIBefore をオーバーライドしていたスクリプトを使って作業する必要があり、そのスクリプトが OnSQLServerInitialize を呼び出さない場合、そのコードをスクリプト ファイルに追加する必要があります。*

タスク ***InstallShield でスクリプト ファイルを編集するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、編集するスクリプト ファイルをクリックします。
3. **[スクリプト]** タブをクリックすると、スクリプト ファイルのコンテンツが表示されます。
4. 必要に応じてファイルを編集します。

プロジェクトをビルドするたびにスクリプトを再作成する場合、[SQL スクリプト] ビュにあるスクリプトのための [データベースのインポート] タブで [ビルド時に SQL スクリプトを再作成する] オプションを選択できます。既存のスクリプト ファイルのバックアップは必ず最初にとるようにしてください。

メモ: *使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:*

- *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
- *Microsoft SQL Server 2005 Management Studio (*SqlWb.exe*)*
- *Microsoft SQL Server 2005 Management Studio Express (*ssmsee.exe*)*
- *Microsoft SQL Server 2000 Query Analyzer (*isqlw.exe*)*

SQL スクリプト ファイルのバージョン番号を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[SQL スクリプト] ビューでスクリプトの作成またはインポートを行ったあと、SQL スクリプト ファイルへスキーマ バージョン番号を割り当てることができます。スキーマ バージョン情報を指定すると、適切なときにだけ SQL スクリプトの起動をトリガーするのに便利です。

タスク ***SQL スクリプト ファイルのスキーマ バージョンを指定するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、構成する SQL スクリプトをクリックします。
3. **[全般]** タブをクリックします。
4. **[スキーマ バージョン (xxxxx.xxxxx.xxxxx.xxxxx)** ボックスで、SQL スクリプトのスキーマ バージョン番号を指定します。

インストールは、ターゲット データベースにある現在のスキーマ バージョンを確認します。スキーマ バージョンは、InstallShield というカスタム テーブルの ISSchema 列に格納されます。SQL スクリプトへスキーマ バージョン番号を指定すると、インストールは、スクリプト スキーマ バージョン番号が現在のスキーマ バージョン番号より大きいときのみ、スクリプトを実行します。一旦スクリプトが実行されると、インストールはターゲット データベースにある現在のスキーマ バージョンを更新して、新しいスキーマ バージョン番号を反映します。

SQL スクリプトのスキーマ バージョン (xxxx.xxxxx.xxxx) を指定しなかった場合、スクリプトは常に起動されます。

たとえば、インストールで、新しい接続が **TestDB** という名前のデータベースに対して作成されてから **TestScript** と呼ばれるスクリプトが作成され、そのスクリプトが **12345.54321.12345.54321** というスキーマ バージョン番号を持つとします。**TestDB** データベースに、カスタム InstallShield テーブルが作成され、そのテーブルの InstallShield 列にスキーマ バージョンが格納されます。同じシステムで別のインストールが実行され、かつこのインストールにも **TestScript** という名前の SQL スクリプトがある場合、この SQL スクリプトは、もう 1 つのスクリプトのスキーマ バージョン番号がターゲット データベースの InstallShieldテーブルにある ISSchema 列に格納されているバージョン番号よりも大きいときのみ実行されます。スクリプトが実行されると、インストールは ISSchema 列を新しいスキーマ バージョン番号で更新します。

ヒント: *"スキーマ バージョン" 設定で番号を指定して、InstallShield によって、ターゲット データベースにそのスキーマ バージョン番号を格納するための* InstallShield *テーブルが追加されると、データは、アンインストールが実行されたときも自動的に削除されません。従って、インストールが変更内容をロールバックできるようにするためには、アンインストール時にテーブルを削除するカスタム スクリプトを作成して、*InstallShield *テーブルを削除する必要があります。*

接続に関連付けられている複数 SQL スクリプトの実行順序を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

プロジェクトで 1 つ以上の SQL スクリプトを SQL 接続に追加する場合、インストールがターゲット マシンでこれらの SQL スクリプトを実行する順序を指定できます。順序を指定するための手順は、Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトで異なります。

基本の MSI プロジェクト、DIM、InstallScript MSI プロジェクトで SQL スクリプトの順序を指定する

基本の MSI、DIM、および InstallScript MSI プロジェクトの [SQL スクリプト] ビューで、接続の SQL スクリプトが一覧表示される順序は、SQL スクリプトが実行される順序です。たとえば、[SQL スクリプト] ビューで接続を作成し、その接続に 1 つ以上の SQL スクリプト追加した場合、SQL 接続のすぐ下に表示されているスクリプトが、インストールで一番最初に実行されます。

タスク ***接続の SQL スクリプトが実行される順番を変更するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、移動する SQL スクリプト右クリックし、**[上に移動]** または **[下に移動]** をクリックします。

順序は、次回インストールをビルドして実行したとき、更新されます。

InstallScript プロジェクトで SQL スクリプトの順序を指定する

InstallScript インストールでは、デフォルトで、SQL スクリプトのバッチ モードが無効になっています。したがって、接続の SQL スクリプトが実行される順序は、インストールが関連付けられているコンポーネントを処理する順番に対応します。

このデフォルト動作をオーバーライドして、SQL スクリプトを実行する順序を再指定できるようにするには、[SQL スクリプト] ビューでバッチ モードを有効にします。

タスク ***バッチ モードを有効または無効にするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[バッチ モード]** をクリックします。

[バッチ モード] コマンドにチェック マークが付いている場合、バッチ モードは有効になっています。チェック マークが付いてない場合、バッチ モードは無効です。

SQLRTGetBatchList 関数は、バッチ モードが有効になっているとき実行する必要がある SQL スクリプトに関連付けられているコンポーネントの一覧を返します。詳細については、「SQLRTGetBatchList」を参照してください。

デフォルトの SQL ランタイム動作をオーバーライドする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

次の Windows Installer のプロパティを定義して、デフォルトのランタイム動作をオーバーライドすることができます:

テーブル 2 ・ SQL の Windows Installer プロパティ

プロパティ	説明
IS_SQLSERVER_CONNECTIONS_TO_VALIDATE	SQLLogin ダイアログの [次へ] ボタンをクリックしたときテストされる接続をオーバーライドします。複数の接続は、セミコロンで区切って指定します。デフォルトで、ISSQLConnection テーブルにあるすべての接続が検証されます。
IS_SQLSERVER_CXNS_ABSENT_FROM_INSTALL	インストールまたはアンインストール中にスキップする 1 つまたは複数の SQL 接続を指定します。1 つ以上の SQL 接続を指定するには、各接続をセミコロン (;) で区切ります。すべての SQL 接続をスキップするには、このプロパティの値を **ALL** に設定します。このプロパティは、SQL スクリプト作成エラーが原因で製品をアンインストールできないとき便利です。

テーブル 2 • SQL の Windows Installer プロパティ (続き.)

プロパティ	説明
IS_SQLSERVER_DO_NOT_USE_REG	レジストリに格納されている SQL Server ログイン情報を使用しないように指定します。SQLLogin ダイアログは、メンテナンスおよびアンインストール モードでは表示されないので、インストールはインストール中に指定されたログイン情報を格納します。この動作が必要ない場合は、IS_SQLSERVER_DO_NOT_USE_REG プロパティを設定してください。
IS_SQLSERVER_LOCAL_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールでローカル SQL Server のみ表示するよう指定します。 ***メモ:*** *ローカル SQL Server とリモート SQL Server または SQL Server エイリアスのみを表示するには、IS_SQLSERVER_LOCAL_ONLY プロパティと、IS_SQLSERVER_REMOTE_ONLY プロパティまたは IS_SQLSERVER_ALIAS_ONLY プロパティのどちらかを設定します。*
IS_SQLSERVER_REMOTE_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールでリモート SQL Server のみ表示するよう指定します。 ***メモ:*** *リモート SQL Server とローカル SQL Server または SQL Server エイリアスのみを表示するには、IS_SQLSERVER_REMOTE_ONLY プロパティと、IS_SQLSERVER_LOCAL_ONLY プロパティまたは IS_SQLSERVER_ALIAS_ONLY プロパティのどちらかを設定します。*
IS_SQLSERVER_ALIAS_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールで SQL Server エイリアスのみ表示するよう指定します。 ***メモ:*** *リモート SQL Server エイリアスとローカル SQL Server または リモート SQL Servers のみを表示するには、IS_SQLSERVER_ALIAS_ONLY プロパティと、IS_SQLSERVER_LOCAL_ONLY プロパティまたは IS_SQLSERVER_REMOTE_ONLY プロパティのどちらかを設定します。*

プロジェクト: *基本の MSI プロジェクトの場合、すべての接続は SQLLogin ダイアログにリンクされています。複数の SQLLogin ダイアログを表示するには、[ダイアログ] ビューから SQL ダイアログをコピーし、デフォルトの SQL ダイアログに似せてその動作とイベントを変更します。必ず、新しいプロパティを作成して、[SQL スクリプト] ビューにある接続の [詳細] タブでそれらを設定するようにしてください。作成した SQL ダイアログのコピーを変更する場合、それらの新しいプロパティを使用できます。*

SQL ランタイム エラーの処理

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

タスク ***インターフェイスで SQL スクリプトのエラー処理プロパティを設定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、エラー処理を追加する SQL スクリプト ファイルをクリックします。
3. **[ランタイム]** タブをクリックします。
4. **[スクリプト エラー処理]** 領域で、一覧からオプションを 1 つ選択するか、**[カスタム]** ボタンをクリックして、スクリプトのデフォルト エラー処理をオーバーライドします。選択可能なオプションは以下のとおりです:
   - エラー時に、次のスクリプトへ移動する
   - エラー時に、次のステートメントへ移動する
   - エラー時に、インストールを中止する

SQL スクリプトにデータベース サーバー タイプの条件を設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、Microsoft SQL Server および MySQL 両方をターゲットとし、かつ異なるデータベース サーバー テクノロジごとに複数の SQL スクリプトをもつ単一接続を作成することができます。ただし、接続は最初に検出されたデータベース サーバーのタイプに基づいて作成されます。したがって、スクリプトは、検出されたデータ

ベース サーバー タイプに対してのみ実行され、検出されなかったデータベース サーバー タイプ固有のスクリプトは失敗します。たとえば、ランタイムが Microsoft SQL Server を検出すると, Microsoft SQL Server に関連付けられたスクリプトが実行され、MySQL 固有のスクリプトは失敗します。

この動作の回避策として、インストールが検出されなかったデータベース サーバー タイプのスクリプトを実行したときに SQL スクリプト エラーが起こるように SQL に条件を設定することをお勧めします。条件を設定することで、スクリプト作成エラーにカスタム エラー処理を設定し、接続の次のスクリプトへ移動することができます。以下は、スクリプトにデータベース サーバー タイプの条件を設定するための手順です。

タスク

Microsoft SQL Server をターゲットにするスクリプトにデータベース サーバー タイプの条件を設定するには、以下の手順に従います。

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、条件を作成するスクリプト ファイルをクリックします。
3. **[スクリプト]** タブをクリックします。
4. スクリプトの始めに、次のステートメントを追加します。

   SELECT @@ROWCOUNT
5. **[ランタイム]** タブをクリックします。
6. **[スクリプト エラーの処理]** 領域で、**[カスタム]** ボタンをクリックします。**[カスタム エラー処理]** ダイアログ ボックスが開きます。
7. **[ここをクリックして新規の項目を追加します]** 行をクリックします。
8. **[エラー番号]** 列で、**1193** と入力します。これは、MySQL がスクリプトの始めで追加された指定のシステム変数を持っていなかったときに、MySQL より返されるエラー番号です。
9. **[動作]** 列で、**[エラー時に、次のスクリプトへ移動する]** をクリックします。
10. **[プロジェクト全体]** 列で、**[いいえ]** を選択します。
11. **[OK]** をクリックします。

タスク

MySQL をターゲットにするスクリプトにデータベース サーバー タイプの条件を設定するには、以下の手順に従います。

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、条件を作成するスクリプト ファイルをクリックします。
3. **[スクリプト]** タブをクリックします。
4. スクリプトの始めに、次のステートメントを追加します。

   SELECT @@table_cache
5. **[ランタイム]** タブをクリックします。
6. **[スクリプト エラーの処理]** 領域で、**[カスタム]** ボタンをクリックします。**[カスタム エラー処理]** ダイアログ ボックスが開きます。
7. **[ここをクリックして新規の項目を追加します]** 行をクリックします。

8. **[エラー番号]** 列で、**137** と入力します。これは、Microsoft SQL Server がスクリプトの始めで追加された指定のシステム変数を持っていなかったときに、Microsoft SQL Server より返されるエラー番号です。
9. **[動作]** 列で、**[エラー時に、次のスクリプトへ移動する]** をクリックします。
10. **[プロジェクト全体]** 列で、**[いいえ]** を選択します。
11. **[OK]** をクリックします。

Windows Installer ベースのインストールで SQL スクリプトを条件付きで起動する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

インストールで、SQL スクリプトを、ターゲット システムである条件が満たされた場合のみ起動するようにした方がよい場合があります。たとえば、作成した SQL スクリプトで、エンドユーザーが管理者権限を所持している必要があるとします。エンドユーザーが管理者権限を所持していない場合、この SQL スクリプトは起動されません。

タスク ***SQL スクリプトの条件を作成するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、条件を作成するスクリプト ファイルをクリックします。
3. **[ランタイム]** タブをクリックします。
4. **[スクリプト条件]** 領域で、**[条件ステートメントを指定する]** チェック ボックスを選択します。
5. 省略記号ボタン (...) をクリックします。**[条件ビルダー]** ダイアログ ボックスが開きます。
6. 1 つまたは複数の条件を作成します。

SQL スクリプトはコンポーネントの状態に関連付けられています。[SQL スクリプト] ビューで設定された条件は、SQL スクリプトのコンポーネントの条件です。コンポーネントの条件がターゲット システムで満たされた場合、インストールで SQL スクリプトがインストールされます。インストールされる予定ではない SQL スクリプトがターゲット システムにインストールされる場合、インストールのログ ファイルを生成すると、SQL スクリプトが間違ってインストールされる理由を判別しやすくなります。

ヒント: *条件および条件の構文の詳細については、「条件ステートメントのビルド」および「条件ステートメントの構文」を参照してください。*

InstallScript インストールで SQL スクリプトを条件付きで起動する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

インストールで、SQL スクリプトを、ターゲット システムである条件が満たされた場合のみ起動するようにした方がよい場合があります。

InstallShield は、1 セットのデフォルトのグローバル イベント ハンドラーを生成します。各イベント ハンドラーは InstallScript 言語でスクリプトされた関数です。次の SQL 関連のイベントは、InstallShield フレームワークによって自動的に呼び出されます。

- OnSQLServerInitialize
- OnSQLComponentInstalled

OnSQLServerInitialize は, OnFirstUIBefore より呼び出され, OnSQLComponentInstalled は、ファイル転送中に、インストールされた各コンポーネントに対して呼び出されます。

メモ: *InstallShield にアップグレードする前に OnFirstUIBefore をオーバーライドしていたスクリプトを使って作業する必要があり、そのスクリプトが OnSQLServerInitialize を呼び出さない場合、そのコードをスクリプト ファイルに追加する必要があります。*

スクリプトで、OnSQLServerInitialize と OnSQLComponentInstalled を変更し、異なる事柄についてチェックを行うことができます。たとえば、下のサンプル コードではユーザーが管理者特権を持つかどうかを確認をすることができます。

```
function OnSQLComponentInstalled(szComponent)
string sMessage;
string sData;
number nResult;
begin

    if( Is( USER_ADMINISTRATOR, sData ) ) then

        nResult = SQLRTComponentInstall( szComponent );

        if( nResult = SQL_ERROR_ABORT ) then

            sMessage = SdLoadString( IDS_IFX_SQL_ERROR_RUN_FAILED );
            MessageBox( sMessage, MB_OK );
            abort;

        endif;
    else
        //管理者権限がないユーザーです。スクリプトを実行しません。
    endif;
end;
```

メモ: *InstallShield インターフェイスで、スクリプトが失敗したときの動作を構成するには、[SQL スクリプト] エクスプローラーで [SQL スクリプト] をクリックして、[ランタイム] タブに移動します。[スクリプト エラー処理] セクションを使用して、次のオプションの中の一つを選択することができます。*

- *エラー時に、次のスクリプトへ移動する*
- *エラー時に、次のステートメントへ移動する*
- *エラー時に、インストールを中止する*

SQL スクリプトが完全 SQL サーバーに対してのみ実行されるように要求する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

SQL スクリプト サポートを含むインストールでは、エンドユーザーはデフォルトで、Microsoft SQL Server Desktop Engine (MSDE) および SQL Server Express Edition で SQL スクリプトを実行することができます。エンドユーザーが完全 SQL サーバーでのみ SQL スクリプトを実行できるようにする場合、[SQL スクリプト] ビューを使用して、インストールの任意のデータベース接続対してそれを構成することができます。

タスク

SQL スクリプト ファイルが MSDE または SQL Server Express Edition を持つターゲット システム上で実行されるのを防ぐには、次の手順を実行します。

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーで、構成する SQL 接続をクリックします。
3. **[要件]** タブで、**[Microsoft SQL Server Desktop Engine / SQL Server Express へのインストールを許可する]** チェック ボックスをクリアします。

Windows Installer プロパティを使って SQL スクリプトの文字列を動的に置換する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

基本の MSI、DIM、InstallScript MSI プロジェクトでは、Windows Installer のプロパティを使用して、実行時の SQL スクリプトのテキストの置換を指定することができます。これによって、エンド ユーザーがターゲット システム上で起動される SQL スクリプトで使用される情報を指定することができるように設定できます。Windows Installer は MsiFormatRecord を使って、実行時に SQL スクリプトのプロパティを解決します。

例

次の手順は、ユーザーがビルトイン SQL ログイン ダイアログ (基本の MSI インストールの SQLLogin ダイアログ、または InstallScript MSI インストールの SQLServerSelectLogin2 ダイアログ) で入力する情報を含むカスタム SQL スクリプトを使って、実行時にデータベースを作成する方法を説明します。Windows Installer プロパティが、データベース名とそのターゲット場所に使用されます。

タスク

エンド ユーザーが実行時に指定する情報を含む SQL スクリプトを使ってデータベースを作成するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。InstallShield 新しい SQL 接続が追加されます。
3. SQl 接続をクリックしてから、**[全般]** タブをクリックします。
4. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。
5. **[SQL スクリプト]** エクスプローラーで、新しい接続を右クリックして、**[新しいスクリプト]** をクリックします。新しい SQL スクリプトが SQL 接続に追加されます。
6. SQL スクリプトをクリックして、**[スクリプト]** タブをクリックします。
7. [スクリプト] ペインで、次のように入力します:

```
if not exists(select name from master.dbo.sysdatabases where name = '%DBNAME%')
begin
CREATE DATABASE %DBNAME%
ON
( NAME = %DBNAME%_dat,
    FILENAME = '%DBPATH%%DBNAME%dat.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = %DBNAME%_log,
    FILENAME = '%DBPATH%%DBNAME%log.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
end
GO
```

8. SQL スクリプトの 2 つのテキスト置換を構成します:
   a. **[テキスト置換]** タブをクリックしてから、**[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。
   b. **[検索文字列]** ボックスに次を入力します:

      %DBNAME%

   c. **[置換文字列]** ボックスに次を入力します:

      []S_SQLSERVER_DATABASE]

   d. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

e. **[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。

f. **[検索文字列]** ボックスに次を入力します:

%DBPATH%

g. **[置換文字列]** ボックスに次を入力します:

[INSTALLDIR]

h. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

9. **[ランタイム]** タブをクリックします。

10. **[インストール中にスクリプトを実行]** チェック ボックスをクリアして、**[ログイン中にスクリプトを実行]** チェック ボックスを選択します。

11. リリースをビルドします。

MsiFormatRecord (Windows Installer ヘルプ ライブラリ)

InstallScript テキスト置換を使って SQL スクリプトの文字列を動的に置換する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、テキスト置換文字列変数を使用して、実行時に、SQL スクリプトで使用されているテキストの置き換えを指定することができます。これによって、エンド ユーザーがターゲット システム上で起動される SQL スクリプトで使用される情報を指定することができるように設定できます。InstallScript ランタイム コードは **TextSubSubstitute** 関数を使用して、文字列変数を SQL スクリプト内の適切な値で置き換えます。

例

次の手順は、エンド ユーザーが **SQLServerSelectLogin2** ダイアログに入力する情報を含むカスタム SQL スクリプトを使って、実行時にデータベースを作成する方法を説明します。InstallScript テキスト置換が、データベース名とそのターゲット場所に使用されます。

タスク

エンド ユーザーが実行時に指定する情報を含む SQL スクリプトを使ってデータベースを作成するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。

2. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。InstallShield 新しい SQL 接続が追加されます。

3. SQl 接続をクリックしてから、**[全般]** タブをクリックします。

4. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。

5. **[SQL スクリプト]** エクスプローラーで、新しい接続を右クリックして、**[新しいスクリプト]** をクリックします。新しい SQL スクリプトが SQL 接続に追加されます。

6. SQL スクリプトをクリックして、**[スクリプト]** タブをクリックします。

7. ［スクリプト］ペインで、次のように入力します:

```
if not exists(select name from master.dbo.sysdatabases where name = '%DBNAME%')
begin
CREATE DATABASE %DBNAME%
ON
( NAME = %DBNAME%_dat,
    FILENAME = '%DBPATH%¥%DBNAME%dat.mdf',
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 5 )
LOG ON
( NAME = %DBNAME%_log,
    FILENAME = '%DBPATH%¥%DBNAME%log.ldf',
    SIZE = 5MB,
    MAXSIZE = 25MB,
    FILEGROWTH = 5MB ) ;
end
GO
```

8. SQL スクリプトの 2 つのテキスト置換を構成します:

   a. **[テキスト置換]** タブをクリックしてから、**[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。

   b. **[検索文字列]** ボックスに次を入力します:

      %DBNAME%

   c. **[置換文字列]** ボックスに次を入力します:

      <MYDATABASENAME>

   d. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

   e. **[追加]** ボタンをクリックします。**[検索/置換]** ダイアログ ボックスが開きます。

   f. **[検索文字列]** ボックスに次を入力します:

      %DBPATH%

   g. **[置換文字列]** ボックスに次を入力します:

      <TARGETDIR>

   h. **[OK]** をクリックします。**[検索/置換]** ダイアログ ボックスが閉じます。

9. **[ランタイム]** タブをクリックします。

10. **[インストール中にスクリプトを実行]** チェック ボックスをクリアして、**[ログイン中にスクリプトを実行]** チェック ボックスを選択します。

11. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

12. Database Name コントロールを含むダイアログの OnSQLServerInitialize イベントでダイアログ コードを見つけ、InstallScript 関数 **TextSubSetValue** の呼び出しを追加します。たとえば、ユーザー名をエンド ユーザーが SQLServerSelectLogin2 ダイアログで指定する名前にする場合、次のコードに示されている **TextSubSetValue** の呼び出しを追加します:

```
// ダイアログ ボックスを表示する (接続名なし)
// ダイアログを入れ替える場合、コメントアウト解除
```

```
// nResult = SQLServerSelectLogin2( szConnection, szServer, szUser, szPassword, bWinLogin, szDB, FALSE, TRUE );

// ダイアログ ボックスを表示する (接続名を含む)
// ダイアログを入れ替える場合、コメントアウト
nResult = SQLServerSelectLogin2( szConnection, szServer, szUser, szPassword, bWinLogin, szDB, TRUE, TRUE );
TextSubSetValue ("<MYDATABASENAME>", szDB, FALSE);
```

13. リリースをビルドします。

Windows Installer ベースのインストールで SQL サーバー側インストールを要求する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

インストールが SQL Server マシン上でのみ実行されるように構成する方法として、レジストリ情報のシステム検索を実行し、その結果をプロパティに格納したあと、そのプロパティを設定する条件に使用するという方法があります。以下は、その手順です。

タスク ***Windows Installer ベースのインストールが SQL サーバー マシン上でのみ実行されるように構成するには、以下の手順に従います。***

1. **[動作とロジック]** の下のビュー リストで、**[システム検索]** をクリックします。
2. グリッドを右クリックして、**[追加]** をクリックします。**システム検索ウィザード**が開きます。
3. **[検索内容]** パネルで、**[レジストリ エントリ]** をクリックし、**[次へ]** をクリックします。
4. **[検索方法を指定してください]** パネルで、以下を行います。
   a. **[レジストリ ルート]** 一覧で、**HKEY_LOCAL_MACHINE** をクリックします。
   b. **[レジストリ キー]** ボックスで、次の文字列を入力します。

      Software¥Microsoft¥Microsoft SQL Server

   c. **[レジストリ値]** ボックスで、次の文字列を入力します。

      InstalledInstances

   d. **[次へ]** をクリックします。
5. **[値の処理方法]** パネルで、以下を行います。
   a. **[値を保存するプロパティ]** ボックスで、次を入力します。

      SQLSERVERFOUND

   b. **[追加のオプション]** 領域で、**[プロパティに値を保存し、インストール条件でこのプロパティを使用]** オプションを選択します。
   c. **[完了]** をクリックします。**条件ビルダー**が開きます。

6. 条件を確認し、レジストリ エントリがシステム上で見つからなかったとき、エンドユーザーに表示するメッセージを入力します。たとえば、次のようなメッセージを入力します。

   このマシン上に Microsoft SQL Server が見つかりませんでした。このインストールはサーバー マシン上でのみ実行されるよう設計されています。

7. [OK] をクリックします。

エントリが [システム検索] グリッドに追加されます。

InstallScript プロジェクトのサーバー側インストールを要求する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトで、サーバーサイド インストールを強制する一つの方法として、インストール プロジェクトが特定のレジストリ キーと値を検索し、値が検索されたときのみそれ自身をインストールするという形でプロジェクトをインストールするという方法があります。InstallScript プロジェクトで、これをどのように行うかの例は、以下のサンプル コードを参照してください。

```
function OnBegin()
    string sKey, sValue, sData;
    string sMsg;
    number nType, nSize, nResult;
begin

    RegDBSetDefaultRoot ( HKEY_LOCAL_MACHINE );
    sKey = "Software¥¥Microsoft¥¥Microsoft SQL Server";
    sValue = "InstalledInstances";
    nResult = RegDBGetKeyValueEx( sKey, sValue, nType, sData, nSize );

    if( nResult < 0 ) then

        //SQL Server レジストリキーがありません
        sMsg = "このマシン上に Microsoft SQL Server が見つかりませんでした。¥n" +
               "このインストールはサーバー マシン上でのみ実行されるよう設計されています。";
        MessageBox( sMsg, SEVERE );
        abort;

    endif;

end;
```

SQL スクリプト ファイルをリポジトリにパブリッシュする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

別のプロジェクトで再利用、または他のユーザーと共有したい既存の SQL スクリプト ファイルがある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***SQL スクリプト ファイルをリポジトリにパブリッシュするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. **[SQL スクリプト]** エクスプローラーでパブリッシュするスクリプト ファイルを右クリックしてから、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへスクリプト ファイルをインポートした後、現在のスクリプト ファイルと既存のリポジトリ スクリプト ファイルとは関連性を持ちません。スクリプト ファイルを変更してリポジトリへ再パブリッシュしても、プロジェクト内のインポート済みスクリプト ファイルには影響しません。ただし、リポジトリからプロジェクトへスクリプト ファイルを再インポートすることができます。

MySQL ODBC ドライバーをインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールが実行される前に、MySQL ODBC ドライバーのインストールおよび起動を行う場合、MySQL Connector 3.51 用の InstallShield 前提条件をプロジェクトに含めます。

タスク ***インストールが実行される前に、MySQL ODBC ドライバーのインストールおよび起動を行うには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。
2. 再配布可能ファイルのリストで、[**MySQL Connector ODBC 3.51**] チェック ボックスを選択します。

ヒント: *MySQL Connector ODBC 3.51 セットアップ前提条件は、そのインストーラーをダウンロードして、InstallShield 前提条件をシステムに追加した場合のみ使用可能です。詳しい手順については、「MySQL Connector ODBC 前提条件を含める」を参照してください。*

アンインストール中に、SQL データベースを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

メモ: *現在接続中のデータベースは削除することができません。*

SQL データベースをアンインストール中に削除する必要がある場合、SQL スクリプトを使ってこれを実現できます。以下は、プロジェクトと SQL スクリプトを構成して、Microsoft SQL Server データベースを削除する方法です。

タスク ***アンインストール中に、Microsoft SQL Serve データベースを削除するには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
2. SQL 接続を追加します。
3. **[全般]** タブにある **[カタログ名]** ボックスで、**Master** と入力します。

   Master は、すべての Microsoft SQL Server システムに存在するシステム データベースの名前です。現在接続されているデータベースは削除することができないため、Master データベースに接続して、それから接続されていない別のデータベースを削除することができます。
4. サーバーの認証情報を入力します。
5. 新しい Script ファイルを追加します。
6. 次を SQL スクリプト ファイルに追加します。

   DROP DATABASE *DatabaseName*
   Go

   DatabaseName は削除するデータベースの名前です。

ヒント: *上記に代わる別の手順として、SQL スクリプトでも同じ操作を実行することができます。これを実現するには、次のコードを SQL スクリプトに入力します。*

USE Master
DROP DATABASE DatabaseName
Go

DatabaseName は削除するデータベースの名前です。

Microsoft SQL Azure Database サーバーへの接続と SQL スクリプトの実行

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ターゲット システムにおける Microsoft Windows Azure SQL Database Server に接続するための要件

Microsoft Windows SQL Azure SQL データベース サーバーへの接続には、インストールを実行するターゲット システム上で SQL Server Native Client 10.0 ODBC ドライバーが必要です。このドライバーがターゲット システムっ上に存在していることを確認するには、プロジェクトに Microsoft SQL Server 2012 Native Client 10.00.2531 の InstallShield 前提条件を含めます。

タスク ***Microsoft SQL Server 2008 Native Client 10.00.2531 前提条件をプロジェクトに含めるには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. 再配布可能ファイル リストで、ターゲット システムのアーキテクチャが 32 ビットか 64 ビットかに応じて、適切な **[Microsoft SQL Server 2008 Native Client 10.00.2531]** チェック ボックス (複数可) を選択します。

InstallShield 前提条件ファイル (.prq) で定義された条件が満たされた場合のみ、InstallShield 前提条件が起動します。条件の一覧を参照するには、[再配布可能ファイル] ビューまたは [前提条件] ビューにある SQL Server Native Client 前提条件をクリックしてから、右側の詳細ペインに表示される説明を確認します。これらのビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。

実行時の Windows Azure SQL データベースの接続に使用するユーザー名を指定する

インストールで SQLLogin ダイアログを使用し、かつ SWindows Azure SQL データベースをターゲットする場合、エンドユーザーが SQLLogin ダイアログの [ログイン ID] ボックスで文字列を入力するとき、次のフォーマットの使用が必要になります:

DatabaseUserName@ServerName

参考例:

MyUserName@wbzdh64drd

Oracle のインスタンスに接続して SQL スクリプトを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

メモ: *Oracle 11g InstaInt Client がターゲット システム上で必要な時にインストールがこれをダウンロードする場合、Oracle に問い合わせて、それが可能かどうかを確認してください。可能な場合、独自の Web サイトでダウンロードをホストして、InstallShield 前提条件にそのダウンロード パスを追加しなくてはなりません。Revenera では、このインストールのダウンロードを提供していません。*

Oracle のインスタンスに接続する場合、インストールを実行するエンドユーザーのマシン上に次の要素が必要です:

- Microsoft ODBC for Oracle
- x86 プロジェクトには、Oracle Instant Client ソフトウェアの 32 ビット バージョンが必要です (ターゲット システムが 64 ビットの場合も含む)
- x64 プロジェクトには、64 ビットのターゲット システム上で Oracle Instant Client ソフトウェアの 64 ビット バージョンが必要です。

最新版の MDAC (Microsoft Data Access Components) は、Microsoft ODBC for Oracle ドライバーをサポートします。ただし、Microsoft ODBC for Oracle ドライバーには、Oracle データベースサーバーと応答するために Oracle Instant Client が必要となります。このため、Oracle 11g Instant Client 前提条件を含めることで、インストール時にターゲット マシン上で Oracle Instant Client の構成を可能にできます。Oracle はファイルのインストールを提供しません。InstallShield Program Files フォルダーにある Oracle Instant Client インストール プロジェクトを使って、簡単にファイルをインストールすることができます。手順については、「Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する」を参照してください。

SQL スクリプトに Unicode 文字が含まれている場合、Unicode 文字を含む SQL スクリプトの実行をサポートする Oracle ODBC Instant Client を使用する必要があります。Microsoft ODBC for Oracle は、これをサポートしません。ODBC Instant Client インストールを Oracle 11g Instant Client 前提条件と共に含めて、ターゲット マシン上で Oracle Instant Client を構成し、また ODBC Instant Client のセットアップも行います。詳細については、「Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する」を参照してください。Microsoft ODBC for Oracle の代わりに ODBC Instant Client を使用する場合は、プロジェクトを適切に構成するようにしてください。詳細については、「Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する」を参照してください。

Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Oracle 11g Instant Client サポートを基本の MSI、InstallScript、または InstallScript MSI プロジェクトに追加する前に、Oracle Instant Client を Oracle の Web サイトからダウンロードしてください。Oracle では、ファイルのインストールを提供していません。したがって、.msi パッケージは自分で作成する必要があります。.msi パッケージは、InstallShield Program Files フォルダーにある Oracle Instant Client インストール プロジェクトを利用することで、簡単に作成することができます。

.msi パッケージを作成した後、それを使用中のマシンの適切な場所に追加してから、この Oracle Instant Client .msi パッケージを使用する InstallShield 前提条件に追加できます。

***メモ・** SQL スクリプトに Unicode 文字が含まれている場合、Unicode 文字を含む SQL スクリプトの実行をサポートする Oracle ODBC Instant Client を使用する必要があります。Microsoft ODBC for Oracle は、これをサポートしません。ODBC Instant Client インストールを Oracle 11g Instant Client 前提条件と共に含めて、ターゲット マシン上で Oracle Instant Client を構成し、また ODBC Instant Client のセットアップも行います。詳細については、「Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する」を参照してください。*

タスク ***Oracle Instant Client をダウンロードして、.msi パッケージを作成するには、以下の手順に従います。***

1. http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html から、Oracle Instant Client バージョン 11.1.0.x (32 ビット Windows プラットフォーム用の基本パッケージ) をダウンロードします。ダウンロード ファイルは、instantclient-basic-win32-11.1.0.7.0.zip という名前です。

   ***重要・** InstallShield における Oracle サポートには、ターゲット システムが 32 ビットまたは 64 ビット システムのいずれの場合においても、Oracle Instant Client の 32 ビット バージョンのインストールが必要です。*

2. ファイルを C ドライブのルートに展開します。ファイルを展開すると、すべてのファイルが次の場所に追加されます:

   C:¥instantclient_11_1

3. InstallShield で Oracle Instant Client インストール プロジェクトを開きます。プロジェクトのパスは、次のとおりです。

   InstallShield Program Files フォルダー¥Support¥Oracle Instant Client¥instantclient-win32-11_1.ism

   このプロジェクトは Oracle Instant Client のバージョン 11.1.0.7 のために Reveneraが作成したものですが、このプロジェクトを、より新しいバージョンの Oracle 11g Instant Client (例、11.2.01) に使用することも可能です。

4. Oracle 11g Instant Client のインストール中、**Oracle11g Instant Client 11.1.0.7** という名前が実行時に表示されます。この名前は、.msi ファイルにも使用されます。この名前を変更して、異なるバージョン番号 (例、11.2.0.1、11.2.0.x など) を反映させるには、以下の手順に従います:

   a. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。

   b. **"製品名"** 設定で、既存のテキストを必要に応じて編集します。

5. 11.1.0.7 以後にリリースされたバージョンを使用していて、依存関係が必要な場合、そのファイルまたはマージ モジュールをプロジェクトに追加します。

6. ツールバー上で、**[ビルド]** ボタンをクリックします。

Oracle 11g Instant Client を作成中の InstallShield プロジェクトに含めることができるように、Oracle 11g Instant Client .msi ファイルがビルドされ、次のディレクトリに追加されます。

InstallShield Program Files フォルダー¥SetupPrerequisites¥oracle

ヒント: *11.1.0.7 以後にリリースされたバージョンを使用している場合、InstallShield 前提条件エディターで Oracle 11g Instant Client 11.1.0.7 前提条件を開きます。前提条件の名前を変更して、適切なバージョン番号を反映させます。また、条件も更新して、適切なバージョン番号を反映させます。*

Oracle 11g Instant Client をインストールに追加する方法については、「Oracle 11g Instant Client をインストールする」を参照してください。

Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

Oracle 11g Instant Client および Oracle ODBC Instant Client のサポートを基本の MSI、InstallScript、または InstallScript MSI プロジェクトに追加する前に、Oracle の Web サイトから Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードする必要があります。Oracle では、ファイルのインストールを提供していません。したがって、.msi パッケージは自分で作成する必要があります。.msi パッケージは、InstallShield Program Files フォルダーにある Oracle Instant Client インストール プロジェクトを利用することで、簡単に作成することができます。

.msi パッケージを作成した後、それを使用中のマシンの適切な場所に追加してから、この Oracle .msi パッケージを使用する InstallShield 前提条件に追加できます。

タスク ***Basic Instant Client および ODCBC Instant Client をダウンロードして .msi パッケージを作成するには、以下の手順に従います:***

1. http://www.oracle.com/technetwork/database/features/instant-client/index-097480.html から、Oracle Instant Client バージョン 11.1.0.x (32 ビット Windows プラットフォーム用の基本パッケージ) および Instant Client Package for ODBC バージョン 11.1.0.x をダウンロードします。ダウンロード ファイルの名前は instantclient-basic-win32-11.1.0.7.0.zip および instantclient-odbc-win32-11.1.0.7.0.zip です。

   重要: *InstallShield における Oracle サポートには、ターゲット システムが 32 ビットまたは 64 ビット システムのいずれの場合においても、ターゲット システムに Basic Instant Client および ODBC Instant Client の 32 ビット バージョンが必要です。*

2. ファイルを C ドライブのルートに展開します。ファイルを展開すると、すべてのファイルが次の場所に追加されます:

   C:¥instantclient_11_1

ファイルが C:¥instantclient11_1¥instantclient11_1 に展開されないことを確認してください。

3. Oracle Instant Client インストール プロジェクトのコピーを作成します。プロジェクトのパスは、次のとおりです。

   InstallShield Program Files フォルダー¥Support¥Oracle Instant Client¥instantclient-win32-11_1.ism

   instantclient-win32-odbc11_1.ism のコピーを作成して、元の .ism ファイルと同じディレクトリに配置します。この新しいファイルのパスは次のとおりです:

   InstallShield Program Files フォルダー¥Support¥Oracle Instant Client¥instantclient-win32-odbc11_1.ism

   instantclient-win32-11_1.ism ロジェクトは Oracle Instant Client のバージョン 11.1.0.7 のために Revenera が作成したものですが、このプロジェクトを、より新しいバージョンの Oracle 11g Instant Client および ODBC Instant Client (例、11.2.01) に使用することも可能です。

4. InstallShield で作成した新しいプロジェクトを開きます。

5. 製品コード、アップグレード コード、および製品名を更新します:

   a. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。

   b. **"製品コード"** 設定で、**[新しい GUID の生成]** ボタン (**{..}**) をクリックします。

   c. **"アップグレード コード"** 設定で、**[新しい GUID の生成]** ボタン (**{..}**) をクリックします。

   d. **"製品名"** 設定で、既存のテキストを必要に応じて変更します。

      Oracle11g Instant Client – ODBC 11.1.0.7

      入力した名前は、実行時、Oracle のインストール中に表示されます。この名前を変更して、異なるバージョン番号 (例、11.2.0.1 または 11.2.0.x など) を反映させることができます。

6. 11.1.0.7 以後にリリースされたバージョンを使用していて、依存関係が必要な場合、そのファイルまたはマージ モジュールをプロジェクトに追加します。

7. ODBC Instant Client インストールを起動するカスタム アクションを追加します:

   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

   b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい EXE]** をポイントしてから、**[ディレクトリを参照するパス]** をクリックします。新しいカスタム アクションが追加されます。

   c. 次のように右側のペインでカスタム アクションの設定を構成します:

      - 作業ディレクトリ: INSTALLDIR
      - ファイル名とコマンド ライン: [INSTALLDIR]odbc_install.exe
      - スクリプト内実行: システム コンテキストの遅延実行
      - インストール実行シーケンス: 次の後: InstallODBC
      - インストール実行条件: Not Installed

8. エンド ユーザーが Instant Client をアンインストールするときに ODBC Instant Client をアンインストールするカスタム アクションを追加します:

   a. ビュー リストの **[動作とロジック]** の下にある **[カスタム アクションとシーケンス]** をクリックします。

b. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい EXE]** をポイントしてから、**[ディレクトリを参照するパス]** をクリックします。新しいカスタム アクションが追加されます。

c. 次のように右側のペインでカスタム アクションの設定を構成します:

- 作業ディレクトリ: INSTALLDIR
- ファイル名とコマンド ライン: [INSTALLDIR]odbc_uninstall.exe
- スクリプト内実行: システム コンテキストの遅延実行
- インストール実行シーケンス: 次の後: RemoveODBC
- インストール実行条件: REMOVE="ALL"

9. .msi パッケージ ファイルの名前を指定します:

a. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。

b. **[リリース]** エクスプローラーの中央ペインで、**[製品構成 1]** と名づけられた製品構成をクリックします。

c. **"MSI パッケージ ファイル名"** 設定に、次の名前を入力します:

Oracle11g Instant Client – ODBC 11.1

InstallShield は、作成する Basic Instant Client および ODBC Instant Client に .msi パッケージ用に入力された名前を使用します。このファイルが、作成する InstallShield 前提条件によって起動されます。

10. ツールバー上で、**[ビルド]** ボタンをクリックします。

Oracle11g Instant Client – ODBC 11.1.msi を作成中の InstallShield プロジェクトに含めることができるように、Oracle .msi ファイルがビルドされ、次のディレクトリに追加されます:

InstallShield Program Files フォルダー¥SetupPrerequisites¥oracle

タスク ***前述の手順に従って作成した .msi パッケージを起動する InstallShield 前提条件を作成するには、以下の手順に従います:***

1. Oracle11g Instant Client の 既存の InstallShield 前提条件 (.prq) のコピーを作成します。前提条件のパスは、次のとおりです:

InstallShield Program Files フォルダー¥SetupPrerequisites¥Oracle11g Instant Client 11.1.0.7.prq

コピーを Oracle11g Instant Client – ODBC 11.1.0.7.prq と名づけて、元の .prq ファイルと同じディレクトリに配置します。この新しいファイルのパスは次のとおりです:

InstallShield Program Files フォルダー¥SetupPrerequisites¥Oracle11g Instant Client – ODBC 11.1.0.7.prq

2. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター** が開きます。

3. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

4. 新しい Oracle11g Instant Client – ODBC 11.1.0.7.prq ファイルを参照してから、**[開く]** ボタンをクリックします。

5. **[プロパティ]** タブにある **"InstallShield 前提条件の固有の ID"** 設定に次を入力します:

Oracle11g Instant Client – ODBC 11.1.0.7

異なるバージョンを使用する場合は、適切な ID に更新してください。

6. InstallShield 前提条件の条件を構成します:
   a. **[条件]** タブをクリックします。
   b. 各既存の条件をクリックしてから、**[削除]** ボタンをクリックします。
   c. **[追加]** ボタンをクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
   d. **[レジストリキーの存在の有無]** オプションを選択します。
   e. **"確認するレジストリ キー名を指定します"** 設定に次を入力します:

      HKEY_LOCAL_MACHINE¥SOFTWARE¥ODBC¥ODBCINST.INI¥Oracle in instantclient11_1

      異なるバージョンを使用する場合は、適切な条件を構成してください。
   f. **[32 ビット]** オプションを選択します。
   g. **[指定されたレジストリ キーが "存在しない" とき]** オプションを選択してから、**[OK]** をクリックします。
7. InstallShield 前提条件に含める適切な .msi ファイルを指定します:
   a. **[含めるファイル]** タブをクリックします。
   b. 既存の **Oracle11g Instant Client 11.1.msi** を選択してから、**[更新]** ボタンをクリックします。**[新規ファイル]** ダイアログ ボックスが開きます。
   c. 前述の手順に従ってビルドした Oracle11g Instant Client – ODBC 11.1.msi ファイルを選択してから、**[OK]** をクリックします。
8. InstallShield 前提条件が起動するファイルを指定します:
   a. **[実行するアプリケーション]** タブをクリックします。
   b. **"起動するアプリケーションを指定する"** 設定で、.msi ファイルを選択します。
9. **[ファイル]** メニューで、**[保存]** をクリックします。

Oracle 11g Instant Client および ODBC Instant Client をインストールに追加する方法については、「Oracle 11g Instant Client をインストールする」を参照してください。

Microsoft ODBC for Oracle の代わりに Oracle ODBC Instant Client を使用する場合は、プロジェクトを適切に構成するようにしてください。詳細については、「Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する」を参照してください。

Oracle 11g Instant Client をインストールする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールが実行される時に Oracle 11g Instant Client をインストールする場合、Oracle 11g Instant Client 前提条件をプロジェクトに含めます。

InstallShield 前提条件を含める前に、Oracle Basic Instant Client をダウンロードして、.msi パッケージを作成する必要があります。Oracle ODBC Instant Client のインストールも必要なとき、たとえば Oracle で Unicode サポートを使用する場合は、Basic Instant Client および ODBC Instant Client をダウンロードし、.msi パッケージをビルドし、InstallShield 前提条件を作成してから (Oracle 11g Instant Client をテンプレートとして使用できます)、必要に応じて設定を更新します。詳細については、次を参照してください。

- Oracle Instant Client をダウンロードして、そのための .msi パッケージを作成する
- Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージおよび InstallShield 前提条件を作成する

重要: *InstallShield における Oracle サポートには、ターゲット システムが 32 ビットまたは 64 ビット システムのいずれの場合においても、Oracle Instant Client の 32 ビット バージョンのインストールが必要です。*

タスク ***Oracle 11g Instant Client 前提条件をプロジェクトに含めるには、以下の手順に従います:***

1. 基本の MSI および InstallScript MSI プロジェクトの場合、**[アプリケーション データ]** の下にあるビュー リストで、**[再配布可能ファイル]** をクリックします。

   InstallScript プロジェクトの場合: **[アプリケーション データ]** の下にあるビュー リストで、**[前提条件]** をクリックします。

2. 再配布可能ファイルのリストから [**Oracle11g Instant Client 11.1.0.7**] チェック ボックス (または、任意の Oracle 前提条件のチェック ボックス) を選択します。

InstallShield 前提条件ファイル (.prq) で定義された条件が満たされた場合のみ、InstallShield 前提条件が起動します。条件の一覧を参照するには、[再配布可能ファイル] ビューまたは [前提条件] ビューにある Oracle 前提条件をクリックしてから、右側の詳細ペインに表示される説明を確認します。これらのビューで [詳細を表示] ボタンをクリックして、詳細の表示/非表示を切り替えることができます。

メモ: *Oracle 11g InstaInt Client がターゲット システム上で必要な時にインストールがこれをダウンロードする場合、Oracle に問い合わせて、それが可能かどうかを確認してください。可能な場合、独自の Web サイトでダウンロードをホストして、InstallShield 前提条件にそのダウンロード パスを追加しなくてはなりません。Revenera では、このインストールのダウンロードを提供していません。*

Oracle Database Server 上で Unicode 文字を含む SQL スクリプトを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

InstallShield の [SQL スクリプト] ビューで使用できるデフォルトの SQL スクリプト サポートを使用するインストールの場合、Microsoft ODBC for Oracle を使用して、ターゲット システム上の Oracle データベースとの接続が行われます。一部の状況において、このデフォルトの動作をオーバーライドして、Microsoft ODBC for Oracle の代わりに、Oracle ODBC Instant Client を使用した方がよい場合もあります。

たとえば、SQL スクリプトに Unicode 文字が含まれている場合、Unicode 文字を含む SQL スクリプトの実行をサポートする Oracle ODBC Instant Client を使用する必要があります。Microsoft ODBC for Oracle は、これをサポートしません。

タスク ***InstallShield プロジェクトで、インストールが Microsoft ODBC for Oracle ではなく Oracle ODBC Instant Client を使用するように構成するには、以下の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、**ISSQLDBMetaData** テーブルをクリックします。
3. **Oracle** レコードについて、以下の手順を行います:
   a. AdoDriverName フィールドを以下の値に変更します:

      {Oracle in instantclient11_1}

   b. AdoCxnServer フィールドを以下の値に変更します:

      DBQ=

実行時にインストールは Oracle ODBC Instant Client を使って Oracle データベースに接続し、SQL スクリプトを実行します。

ヒント: *Oracle ODBC Instant Client は、Oracle Instant Client ソフトウェアの ODBC パッケージがインストールされている場合にターゲット システム上で使用可能です。インストールに追加する方法については、「Oracle Basic Instant Client および Oracle ODBC Instant Client をダウンロードして、両方の .msi パッケージ および InstallShield 前提条件を作成する」を参照してください。*

カスタマイズした SQL スクリプトを実行して SQL Server データベースを作成するサンプル プロジェクトを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

以下は、カスタマイズした SQL スクリプトを使用して SQL Server データベースを作成するインストールを作成する方法です。

タスク ***カスタマイズした SQL スクリプトを実行して、ターゲット マシン上で SQL Server データベースを作成するインストールを作成するには、以下の手順に従います。***

1. 基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを新規作成します。
2. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
3. 次の名前を持つ新しいプロパティを作成します。

   IS_SQLSERVER_DATABASE2
4. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
5. 新しい SQL 接続を追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection1** というデフォルト名で作成されます。
   b. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** をクリックし、**[全般]** タブをクリックします。
   c. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、**TESTSQLSERVER** と入力します。
   d. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。
   e. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。
   f. **[ログイン ID]** ボックスで、**sa** と入力します。**[パスワード]** ボックスは空白にできません。
   g. **[要件]** タブをクリックします。
   h. **Microsoft SQL Server** チェック ボックスが選択されていて、**MySQL** と **Oracle** チェック ボックスは選択されていないことを確認してください。
6. NewConnection1 の新しい SQL スクリプトを追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** を右クリックして、**[新しいスクリプト]** をクリックします。
   b. スクリプトの名前を **NewScript1** に変更します。
   c. **[SQL スクリプト]** エクスプローラーで、**NewScript1** をクリックし、**[スクリプト]** タブをクリックします。
   d. スクリプト エディター ペインで、次のスクリプトを追加します。

      CREATE DATABASE [TestDB] ON (NAME = N' TestDB', FILENAME = N'C:¥Program Files¥Microsoft SQL Server¥MSSQL¥data¥testdb.mdf' , SIZE = 3, FILEGROWTH = 10%) LOG ON (NAME = N' TestDB_log', FILENAME = N'C:¥Program Files¥Microsoft SQL Server¥MSSQL¥data¥testdb.ldf' , SIZE = 1, FILEGROWTH = 10%) COLLATE SQL_Latin1_General_CP1_CI_AS
   e. **[ランタイム]** タブをクリックします。
   f. **[スクリプトの実行]** 領域で、**[ログイン中にスクリプトを実行する]** チェック ボックスを選択し、**[インストール中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。
7. ２ つ目の新しい SQL 接続を追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection2** というデフォルト名で作成されます。

b. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** をクリックし、**[詳細]** タブをクリックします。

c. **[ターゲット カタログ プロパティ名]** リストから、**IS_SQLSERVER_DATABASE2** を選択します。

d. **[全般]** タブをクリックします。

e. **[カタログ名]** ボックスで、**TestDB** と入力します。

f. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。

g. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、**TESTSQLSERVER** と入力します。

h. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。

i. **[ログイン ID]** ボックスで、**sa** と入力します。**[パスワード]** ボックスは空白にできません。

j. **[要件]** タブをクリックします。

k. **Microsoft SQL Server** チェック ボックスが選択されていて、**MySQL** と **Oracle** チェック ボックスは選択されていないことを確認してください。

8. NewConnection2 の新しい SQL スクリプトを追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** を右クリックして、**[新しいスクリプト]** をクリックします。

b. スクリプトの名前を **NewScript2** に変更します。

c. **[SQL スクリプト]** エクスプローラーで、**NewScript2** をクリックし、**[スクリプト]** タブをクリックします。

d. スクリプト エディター ペインで、次のスクリプトを追加します。

```
CREATE TABLE TestTable (TestColumn1 CHAR NOT NULL PRIMARY KEY)
```

e. **[ランタイム]** タブをクリックします。

f. **[スクリプトの実行]** 領域で、**[インストール中にスクリプトを実行する]** チェック ボックスを選択し、**[ログイン中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。

インストールを実行すると、**TestDB** データベースが作成され、**TestTable** という名前のテーブルがデータベースに追加されます。

プロジェクト: *DIM プロジェクトを使用して、その DIM をインストール プロジェクトに追加しても、上記の手順と同様の結果を得ることができます。*

カスタマイズした SQL スクリプトを実行して Oracle スキーマを作成するサンプル インストールを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

以下は、カスタマイズした SQL スクリプトを使用して、Oracle スキーマを作成するインストールを作成する方法です。

タスク

カスタマイズした SQL スクリプトを実行して、ターゲット マシン上で Oracle スキーマを作成するインストールを作成するには、以下の手順に従います。

1. 基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを新規作成します。
2. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
3. 次の名前を持つ 3 つの新しいプロパティを作成します。

   IS_SQLSERVER_DATABASE2

   IS_SQLSERVER_USERNAME2

   IS_SQLSERVER_PASSWORD2
4. **[サーバー構成]** の下のビュー リストにある **[SQL スクリプト]** をクリックします。
5. 新しい SQL 接続を追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection1** というデフォルト名で作成されます。
   b. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** をクリックし、**[全般]** タブをクリックします。
   c. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、次のように入力します。

   //sch01jsmith.installshield.com:1521/orcl

   d. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。
   e. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。
   f. **[ログイン ID]** ボックスで、**scott** と入力します。
   g. **[パスワード]** ボックスで、**scott** と入力します。
   h. **[要件]** タブをクリックします。
   i. **Oracle** チェック ボックスが選択されていて、**Microsoft SQL Server** と **MySQL** チェック ボックスがクリアされていることを確認してください。
6. NewConnection1 の新しい SQL スクリプトを追加し、構成します。
   a. **[SQL スクリプト]** エクスプローラーで、**NewConnection1** を右クリックして、**[新しいスクリプト]** をクリックします。

b. スクリプトの名前を **NewScript1** に変更します。

c. **[SQL スクリプト]** エクスプローラーで、**NewScript1** をクリックし、**[スクリプト]** タブをクリックします。

d. スクリプト エディター ペインで、次のスクリプトを追加します。

```
CREATE TABLESPACE TEST_TS LOGGING DATAFILE 'test01.dbf' SIZE 1M AUTOEXTEND ON NEXT 2M MAXSIZE
UNLIMITED
Go
CREATE USER TEST_USER IDENTIFIED BY MYPSWD DEFAULT TABLESPACE TEST_TS QUOTA UNLIMITED on TEST_TS
Go
GRANT CONNECT TO TEST_USER
Go
GRANT DBA TO TEST_USER
Go
ALTER USER TEST_USER DEFAULT ROLE ALL
Go
```

e. **[ランタイム]** タブをクリックします。

f. **[スクリプトの実行]** 領域で、**[ログイン中にスクリプトを実行する]** チェック ボックスを選択し、**[インストール中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。

7. ２ つ目の新しい SQL 接続を追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーを右クリックして、**[新しい SQL 接続]** をクリックします。新しい接続が **NewConnection2** というデフォルト名で作成されます。

b. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** をクリックし、**[詳細]** タブをクリックします。

c. **[ターゲット カタログ プロパティ名]** リストから、**IS_SQLSERVER_DATABASE2** を選択します。

d. **[サーバー認証ログイン ID プロパティ名]** リストから、**IS_SQLSERVER_USERNAME2** を選択します。

e. **[サーバー認証パスワード プロパティ名]** リストから、**IS_SQLSERVER_PASSWORD2** を選択します。

f. **[全般]** タブをクリックします。

g. **[カタログ名]** ボックスで、**TEST_USER** と入力します。

h. **[存在しない時、カタログを作成する]** チェック ボックスをクリアします。

i. **[デフォルト ターゲット サーバー名 (オプション)]** ボックスで、次のように入力します。

//sch01jsmith.installshield.com:1521/orcl

j. **[接続方法]** 領域で、**[次のログイン ID およびパスワードを使用したサーバー認証]** オプションを選択します。

k. **[ログイン ID]** ボックスで、**TEST_USER** と入力します。

l. **[パスワード]** ボックスで、**MYPSWD** と入力します。

m. **[要件]** タブをクリックします。

n. **Oracle** チェック ボックスが選択されていて、**Microsoft SQL Server** と **MySQL** チェック ボックスがクリアされていることを確認してください。

8. NewConnection2 の新しい SQL スクリプトを追加し、構成します。

a. **[SQL スクリプト]** エクスプローラーで、**NewConnection2** を右クリックして、**[新しいスクリプト]** をクリックします。

b. スクリプトの名前を **NewScript2** に変更します。

c. **[SQL スクリプト]** エクスプローラーで、**NewScript2** をクリックし、**[スクリプト]** タブをクリックします。

d. スクリプト エディター ペインで、次のスクリプトを追加します。

```
CREATE TABLE TestTable (TestColumn1 CHAR NOT NULL PRIMARY KEY)
```

e. **[ランタイム]** タブをクリックします。

f. **[スクリプトの実行]** 領域で、**[インストール中にスクリプトを実行する]** チェック ボックスを選択し、**[ログイン中にスクリプトを実行する]** および **[アンインストール中にスクリプトを実行]** チェック ボックスが選択されていないことを確認してください。

インストールを実行すると、**TEST_USER** スキーマで **TEST_USER** ユーザーと **TestTable** テーブルが作成されます。

プロジェクト: *DIM プロジェクトを使用して、その DIM をインストール プロジェクトに追加しても、上記の手順と同様の結果を得ることができます。*

スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

SQL サーバーは、特に InstallShield スイート インストールで提供される複数パッケージ サポートを活用する多くのアプリケーションで不可欠です。以前、InstallShield SQL サポートが使用できるのは基本の MSI、InstallScript、および InstallScript MSI プロジェクトのみでした。今回より、SQL サポートがスイート/アドバンスト UI プロジェクトに追加されており、以下の機能を使用することができます:

- 新しい SQLLogin 定義済みウィザード ページを追加
- SQL ステートメントを直接スイートから実行

SQLLogin ウィザード ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージがターゲットとするデータベース サーバーへの接続を設立することができます。

SQLLogin ウィザード ページをプロジェクトに追加すると、以下が可能となります:

- スイート プロジェクト内の SQL ログインの設定を識別するプロパティを指定してから、これらのプロパティを受け取る .msi パッケージを選択する

- .msi パッケージ内で SQL ログインの設定を識別するプロパティを指定する
- データベース テクノロジ (Microsoft SQL Server、Microsoft Windows Azure、MySQL、または Oracle) を選択して、ターゲットにする ODBC ドライバーを選択する
- OnBegin でアクションを使って、更新済みの SQL ドライバーが確実にスケジュールされるようにする

さらに、スイート/アドバンスト UI プロジェクトでは今回より、ユーザー インターフェイスから SQL データベース サーバー上の SQL ステートメントを直接実行することができます。これによって、インストール続行前に SQL データベース サーバーを調査することができます。これによって、スイート プロパティで SQL クエリの結果にアクセスが可能となります。

次は、SQLLogin 定義済みウィザード ページをプロジェクトに追加する方法を示します。別の方法として、スイート プロジェクト内の SQL サポートを手動で構成して、インストールのニーズに従って UI をカスタマイズすることもできます。

タスク ***スイート/アドバンスト UI プロジェクトで、SQLLogin 定義済みウィザード ページを追加して、オプションで SQL ステートメントを直接実行するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。**ユーザー インターフェイス ウィザード**が開きます。.
3. **[タスク ページ]** リストから、**[データベース サーバーのログイン情報を入力]** を選択します。
4. **[後続ページ]** リストを使って、ウィザード ページ シーケンスでデータベース サーバー ウィザード ページをスケジュールする場所を指定します。実行時に、このリストで選択したページが、新しいデータベース サーバー ウィザード ページの直後に表示されます。
5. ユーザー インターフェイス ウィザードの [タスク構成] パネルには、次のオプションがあります:
   a. スイート プロジェクト内の "SQL ログイン" の設定を識別するプロパティを指定する

      ***メモ・** 1 つのスイート プロジェクトには、異なる SQL サーバーを必要とする、または共有する必要のある複数パッケージが含まれている場合があります。*

   b. プロパティを受け取る .msi パッケージを選択する。
   c. データベース テクノロジおよびドライバーを選択する。

   InstallShield は、指定済みの "タスク構成" プロパティに対応する "開始ページ" 設定を追加します。
6. オプションで、実行時にスイート/アドバンスト UI プロジェクトから 直接実行する SQL ステートメントを追加する場合、プロジェクトに **[SQL 文字列の実行]** アクションを追加してから、必要に応じてそのサブ設定を指定します。SQL クエリ結果には、**[データ プロパティ]** サブ設定で指定するスイート プロパティからアクセスが可能です。

COM+ アプリケーションとコンポーネントの管理

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[コンポーネントサービス] ビューでは、インストールパッケージ用の COM+ アプリケーションとコンポーネントを管理することができます。COM+ サーバー アプリケーションとアプリケーション プロキシの両方を管理することができます。COM+ アプリケーション プロキシはサーバー アプリケーション属性のサブセットで構成され、これはクライアン ト コンピューターからアプリケーションが存在するマシンへのリモート アクセスを可能にします。

InstallShield のコンポーネント サービスに関する次の情報をお読みください。

- COM+ システム アプリケーション以外のみプロジェクトに追加することができます。したがって、InstallShield は [コンポーネント サービス] ビュー の [COM+ アプリケーション] エクスプローラーの下に、COM+ システム アプリケーション以外のみを表示します。
- ローカル マシンにインストールされている COM+ アプリケーションのみが、[コンポーネント サービス] ビューに表示され、プロジェクトに追加することが可能です。
- アプリケーションプロキシは COM+ サーバーアプリケーションでのみ利用可能で、ライブラリアプリケーションでは利用できません。

[コンポーネント サービス] ビューの外観は、[コントロール パネル] の [コンポーネント サービス] 管理ツールに似ています。

COM+ サーバー アプリケーションの管理

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

タスク ***COM+ サーバー アプリケーションのインストール設定を構成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[コンポーネント サービス]** をクリックします。
2. **[コンポーネント サービス]** エクスプローラーで、構成する COM+ アプリケーションをまだ選択していない場合は、それを選択します。
3. **[インストール]** タブで、**[サーバー]** チェック ボックスを選択します。

メモ・[プロキシ]チェック ボックス、または [サーバー] チェック ボックス、もしくはその両方を選択してください。これらのチェック ボックスの 1 つをクリアすると、もう 1 つのチェック ボックスは選択された状態が保たれますが、変更することはできません。これは誤って両方のチェック ボックスをクリアしないよう防ぐためです。

4. **[ビルド時にクライアントマシンから COM+ 設定をリフレッシュする]** チェック ボックスを必要に応じて選択します。
5. インストールが COM+ アプリケーション プロキシ サポートを含まない場合、[**プロキシ**] チェック ボックスをクリアします。

COM+ サーバー アプリケーションをプロジェクトへ追加すると、対応するコンポーネントが、プロキシ コンポーネントと関連付けられている各機能に対して作成されます。このコンポーネントは COM+ アプリケーションのサーバー .dll ファイルを含みます。

メモ・選択された COM+ アプリケーションがサーバーとプロキシ インストールの両方を含む場合、インストール条件を追加して適切なアプリケーションがターゲット マシンへインストールされるように設定することができます。詳細については、「サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める」を参照してください。

COM+ アプリケーション プロキシの管理

***プロジェクト・**この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

COM+ アプリケーション プロキシ サポートは、クライアント マシンからサーバー アプリケーションが存在するマシンへのリモート アクセスを可能にするシステム設定を構成しますクライアント アプリケーションをインストールに追加しなくてはならない場合もあります。

タスク ***COM+ アプリケーション プロキシのインストール設定を構成するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[コンポーネント サービス]** をクリックします。
2. **[コンポーネント サービス]** エクスプローラーで、構成する COM+ アプリケーションをまだ選択していない場合は、それを選択します。
3. **[インストール]** タブで、**[プロキシ]** チェック ボックスを選択します。
4. インストールが COM+ サーバー アプリケーションを含まない場合は、**[サーバー]** チェック ボックスをクリアします。

***メモ・[プロキシ]**チェック ボックス、または **[サーバー]** チェック ボックス、もしくはその両方を選択してください。これらのチェック ボックスの 1 つをクリアすると、もう 1 つのチェック ボックスは選択された状態が保たれますが、変更することはできません。これは誤って両方のチェック ボックスをクリアしないよう防ぐためです。*

5. **[リモート サーバー名]** ボックスには、アプリケーションが常駐するリモート サーバー コンピューターの名前を指定します。正しい名前を入力するか、またはデフォルトの [REMOTESERVERNAME] プロパティを利用することが可能です。このデフォルト値はインストールで COM+ アプリケーションの **[プロキシ]** チェック ボックスを選択したときに自動的に作成されます。

   ***メモ・**REMOTESERVERNAME プロパティのデフォルト値は、InstallShield で COM+ アプリケーションを インストール プロジェクトへ追加するために使用されるマシンの名前です。*[REMOTESERVERNAME] *プロパティの値を変更するには、[プロパティ マネージャー] ビューを使用します。*

   *エンドユーザーがリモート サーバーを指定できるようにしたい場合は、**[ダイアログ]** ビューのエンドユーザー ダイアログにリモート サーバー編集フィールド コントロールを追加します。このコントロールの **[プロパティ]** 値に* **REMOTESERVERNAME***を設定します。*

6. **[分散 COM をクライアント マシンで有効にする]** チェック ボックスを必要に応じて選択します。DCOM (分散コンポーネント オブジェクト モデル) がすべてのクライアント マシン上で既に有効であることが分かっていて、クライアント マシン上での管理者権限を持たないことが分かっている場合、このチェック ボックスをクリアします。

   このチェック ボックスを選択した場合、インストール時に Y が HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Ole レジストリキーの EnableDCOM エントリに書き込まれて、DCOM が有効となります。

   ***メモ・**エンドユーザーは [コントロール パネル] にある [コンポーネント サービス] 管理ツールを利用して、マシン上で DCOM を有効または無効にすることができます。ただし、そのマシンの DCOM が無効になっている場合、アプリケーション プロキシはクライアント マシン上では動作しません。そのため、[クライアント マシン上で 分散 COM を有効にする] チェック ボックスの選択をお勧めします。*

   エンドユーザーがアプリケーション プロキシ サポートをアンインストールする場合、EnableDCOM レジストリ エントリは、インストール プロセスでこのレジストリ エントリがターゲット マシンで Y に変わったとしても、変更はされません。

COM+ アプリケーション プロキシをプロジェクトへ追加すると、対応するコンポーネントが、サーバー コンポーネントに関連付けられている各機能に対して作成されます。このコンポーネントは COM+ アプリケーションのサーバー .dll ファイルを含みます。これらのサーバー ファイルはクライアント マシン上のタイプ ライブラリにインストールされます。

***メモ・**選択された COM+ アプリケーションがサーバーとプロキシ インストールの両方を含む場合、インストール条件を追加して適切なアプリケーションがターゲット マシンへインストールされるように設定することができます。詳細については、「サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める」を参照してください。*

サーバーとクライアント マシンをターゲットにした COM+ アプリケーションを含める

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

適切なコンポーネントがターゲット マシンにインストールされるようにインストール条件を作成することで、インストール時に、サーバー アプリケーションをサーバーにインストールし、アプリケーション プロキシ サポートをクライアント マシンにインストールすることができます。実際の手順を以下に例を挙げて説明します。

タスク

サーバーとクライアント マシン両方をターゲットにした COM+ アプリケーションをインストールに含めるには、以下の手順を実行してください。

1. **[プロパティ マネージャー]** ビューで、COMPLUSTYPE という名前で新しいプロパティを作成します。
2. **[ダイアログ]** ビューで、新規または既存のエンドユーザー ダイアログにラジオ ボタン グループ コントロールを追加し、このコントロールの **Property** 値に **COMPLUSTYPE** を設定します。このラジオ ボタン グループを利用して、エンドユーザーはサーバーまたはプロキシのどちらのインストールを実行するのかを指定できます。
3. ラジオ ボタン グループに **[サーバー]** オプション用のラジオ ボタンを追加し、このコントロールの **Value** プロパティを [Server] に設定します。
4. ラジオ ボタン グループに **[プロキシ]** オプション用のラジオ ボタンを追加し、このコントロールの **Value** プロパティを [Proxy] に設定します。
5. **[コンポーネント サービス]** ビューで、構成する COM+ アプリケーションを選択します。
6. **[インストール]** タブで、[サーバー インストール] および [アプリケーション プロキシ] のパラメーターを設定します。
7. **[サーバー]** チェック ボックスの下の **[条件]** ボックスに、次を入力します:

   COMPLUSTYPE="サーバー"
8. **[プロキシ]** チェック ボックスの下の **[条件]** ボックスに、次を入力します:

   COMPLUSTYPE="プロキシ"

インターネット インフォメーション サービス

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インターネットインフォメーションサービス (IIS) は Microsoft が開発した Web サーバーです。Web ベースのアプリケーションをビルドおよび配布、Web サイトの管理、およびインターネットまたはイントラネットへ情報をパブリッシュするための安定したプラットフォームを提供します。

InstallShield の [IIS 構成] ビューでは、新しい IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を作成および管理することができます。

InstallShield における IIS サポートのバージョン固有情報

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

以下は、特定のバージョンの IIS に関する情報です。

- IIS は Windows 2000 Server 以降および Windows XP 以降のシステムに含まれています。IIS 6 は Windows Server 2003 システムでのみ利用できます。IIS 7 は Windows Vista と Windows Server 2008 システムで提供されています。IIS 7.5 は Windows 7 と Windows Server 2008 R2 システムで提供されています。IIS は自動的にインストールされません (デフォルト)。
- [IIS 構成] ビューにある一部の Web サイトと仮想ディレクトリの設定は、特定のバージョンの IIS に適用します。これらの設定についてのバージョン固有の情報は、InstallShield のインライン ヘルプ ペインに表示されます。バージョン固有のプロパティが構成されていて、ターゲット システムに対応するバージョンの IIS がないとき、IIS はバージョン固有のプロパティを無視します。

  たとえば、IIS 7 と IIS 6 は、アプリケーションまたは仮想ディレクトリの "アプリケーション保護" プロパティをサポートしません。[IIS 構成] ビューで、このプロパティは、アプリケーションまたは仮想ディレクトリの [アプリケーションの設定] 領域にある設定を使って構成されます。[IIS 構成] ビューでこの設定を選択すると、右下に表示されるヘルプ ペインに、この設定が IIS 6 以降には適用しないことが示されます。この設定が選択されているときに、エンドユーザーが製品を IIS 6 以降があるターゲット システムにインストールすると、"アプリケーション保護" 設定は無視されます。
- Web サービス拡張、アプリケーション プール、およびすべての関連プロパティは、IIS 6 があるマシン上でのみ提供されています。

  IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。この機能がインストールされていない場合、インストールのログ ファイルに、この機能が必須である可能性があることを通知するエラー2147467259 が表示される場合があります。

  Windows Vista または Windows Server 2008 システムに、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされているかどうかを判別する方法については、「ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する」を参照してください。

- Windows Vista 以降と Windows Server 2008 以降のシステムでは、インストール プロジェクトで [IIS 構成] ビューの "コンテンツのソース パス (ローカルまたは UNC)" 設定で指定された物理パスにある構成ファイルに個別の Web サイト、アプリケーション、および仮想ディレクトリの設定が保管されます。したがって、各 Web サイト、アプリケーション、または仮想ディレクトリは、一意の物理パスを持ちます。一意の物理パスを持たなかった場合、同じ物理パスを持つ 2 つの異なる仮想ディレクトリがあるとき、たとえばテスト環境などで予期しない動作に遭遇する可能性があります。

  たとえば、同じ物理パスがある 2 つの仮想ディレクトリがあり、ディレクトリの参照が片方のみで有効にされているとき、作成された 2 つ目の仮想ディレクトリの "ディレクトリの参照" 設定は、1 つ目の仮想ディレクトリの設定をオーバーライドします。

- Windows Server 2003 を持つシステムでは、IIS 6 がインストールされていない場合、他の IIS ディレクトリおよびサイトは上記にかかわらず作成されます。IIS 6 固有の設定はスキップされます。
- IIS 5.1 for Windows XP Professional は、一回につき 1 つの Web サイトのみサービス可能です。これは IIS 5.1 の制限によるものです。
- InstallShield はバージョン 5 およびそれ以降の IIS をサポートします。

インストールする IIS のバージョンを判別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューで Web サイトを追加すると、ターゲット システムにインストールされる IIS バージョンを検索するためのエントリが [システム検索] ビューに追加されます。IIS_VERSION プロパティは (ダイレクト エディターにある) RegLocator と AppSearchテーブルのエントリによって設定され、インストールされた IIS のバージョンを判別します。

IIS_VERSION プロパティには、IIS のバージョン番号が含まれています。インストールされている IIS バージョンを判別する必要がある場合、このプロパティの値を確認します。

IIS サポートの実行時要件

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield インストールの IIS サポートは、ターゲット マシンに IIS がインストールされていて、エンド ユーザーが管理者権限を持つ場合のみ動作します。

IIS の設定を含むパッケージのインストール実行中、InstallShield インストールはターゲットマシンに IIS があるかどうかを確認します。IIS がインストールされていない場合、インストールはエンドユーザーにダイアログを表示して IIS がインストールされていないことを通知します。ダイアログで、エンドユーザーは [中止]、[再試行]、[無視] のいずれかを選択することができます。

- エンドユーザーが [中止] を選択すると、インストールが終了します。
- エンドユーザーが IIS をインストールしてから、[再試行] を選択したとき、インストールは IIS の存在を再確認し、インストールを続行します。エンドユーザーが IIS をインストールしていないにもかかわらず [再試行] を選択したとき、インストールは IIS の存在を再確認し、ダイアログを再度表示します。
- エンドユーザーが [無視] を選択すると、インストールは続行しますが、IIS Web サイト、アプリケーション、仮想ディレクトリ、および他の IIS 要素は構成されません。

メモ: *InstallShield では、インストールを実行している ターゲット マシン以外のターゲット マシンでの Web サイトの作成はサポートされていません 。*

プロジェクト: *InstallScript プロジェクトでアプリケーション プールを作成する場合で、アプリケーション プールに仮想ディレクトリ、アプリケーション、または Web サイトに関連付けるには、同じコンポーネント内でアプリケーション プールと仮想ディレクトリ、アプリケーション、または Web サイトを作成しなくてはなりません。これを行うことで、仮想ディレクトリ、アプリケーション、または Web サイトが作成される前にアプリケーション プールが作成されるようになります。これは、IIS 6 以降の要件で、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの制限事項ではありません。*

Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

サーバー側インクルード (SSI) ディレクティブは、コンテンツを Web ページに挿入するようにWeb サーバーに指示します。#exec タイプのディレクティブによって、Web サーバーは Web ページにシェル コマンドの出力を含めることができます。

IIS Web サーバーを構成して、#exec ディレクティブの CMD コマンドがシェル コマンドの実行に使用されるのを防いだり、CMD コマンドがこのタイプのコマンドの実行に使用されることを許可することができます。HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters レジストリ キーの SSIEnableCmdDirective レジストリ値によって、CMD コマンドが許可されているかどうかが判別されます。

InstallShield では、インストール時に、ターゲット システム上で SSIEnableCmdDirective レジストリ値をどう構成するかを指定することができます。インストールで SSIEnableCmdDirective レジストリ値を変更しない場合、そのように指定することもできます。

セキュリティに関する懸念により、デフォルトの SSIEnableCmdDirective 値は FALSE (0) になっています。FALSE (0) 値により、エンドユーザーによって承認されていないサーバー側での実行可能ファイルの実行を防ぐことができます。

タスク ***Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. 中央のペインで、**[Web サイト]** エクスプローラーをクリックします。右側のペインにショートカットの設定が表示されます。
3. **"SSIEnableCmdDirective レジストリ値"** 設定で、適切なオプションを選択します。
   - **無視する** – ターゲット システム上の SSIEnableCmdDirective レジストリ値を変更しません。デフォルトでは、これが設定されています。
   - **FALSE (0)**–ターゲット システム上の SSIEnableCmdDirective レジストリ値を 0 に設定します。これにより、サーバー側インクルードの #exec CMD ディレクティブがシェル コマンドの実行に使用されるを防ぐことができます。この値を選択すると、IIS Web サーバーに #exec CMD ディレクティブに依存するアプリケーションが存在した場合、インストール プロジェクトの Web サイトおよび仮想ディレクトリがインストールされたあと、これらのアプリケーションが誤作動を起こす可能性があります。
   - **TRUE (1)**–ターゲット システム上の SSIEnableCmdDirective レジストリ値を 1 に設定します。これにより、サーバー側インクルードの #exec CMD ディレクティブがシェル コマンドの実行で使用できるようになります。

FALSE または TRUE オプションを選択すると、値 (FALSE の場合 0、TRUE の場合 1) がINSTALLSHIELD_SSI_PROP プロパティに格納されます。

インストールにある 1 つまたは複数の Web サイト、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張がターゲット システムにインストールされて、**"SSIEnableCmdDirective レジストリ値"** 設定で FALSE または TRUE オプションが選択されている場合、SSIEnableCmdDirective レジストリ値がターゲット システムで更新されます。

メモ: *製品のインストール中に SSIEnableCmdDirective レジストリ値が変更された場合でも、ターゲット システムから製品がアンインストールされるときに、SSIEnableCmdDirective レジストリ値が変更されることはありません。*

Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield の [IIS 構成] ビューで、プロジェクトに IIS Web サイトを追加します。このビューではまた、Web サイトにアプリケーションおよび仮想ディレクトリを追加することもできます。Web サイトをコンポーネントに関連付けると、アプリケーションまたは仮想ディレクトリを持たない Web サイトを追加することもできます。

タスク ***実行時にターゲット システム上で Web サイトを作成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトの追加]** をクリックします。InstallShield が新しい Web サイトを追加します。
3. 設定を構成する Web サイトを選択します。

ヒント: *InstallShieldでは、プロジェクトで Web サイトの TCP ポートとサイト番号を指定できます。これらの設定は、実行時に新しい Web サイトが作成されるか、既存の Web サイトが更新されるのかを判断するのに役立ちます。詳細については、「TCP ポート番号とサイト番号の構成」を参照してください。*

タスク ***実行時にターゲット システム上でアプリケーションを作成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、アプリケーションを含める Web サイトを右クリックして、**[新しいアプリケーション]** をクリックします。新しいリリースが追加されます。
3. 設定を構成するアプリケーションを選択します。

タスク ***実行時にターゲット システム上で仮想ディレクトリを作成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ア仮想ディレクトリを含める Web サイトを右クリックして、**[新しい仮想ディレクトリ]** をクリックします。新しい仮想ディレクトリが追加されます。
3. 設定を構成する仮想ディレクトリを選択します。

メモ: *Web サイト、アプリケーション、および仮想ディレクトリをコンポーネントと機能に関連付ける方法については、「IIS サポートの機能とコンポーネントの関連付け」を参照してください。*

IIS Web サイトのスキャンをして、その設定を InstallShield プロジェクトにインポートする

エディション: *IIS データを InstallShield プロジェクトにインポートする機能は、InstallShield Premier Edition のみで使用できます。*

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield には、既存の IIS Web サイトをスキャンし、その Web サイトについての IIS データを記録して、それらの設定を InstallShield プロジェクトの [IIS 構成] ビューにインポートするサポートが含まれています。データをプロジェクトにインポートした後、必要に応じて IIS 設定を変更して、IIS Web サイトを作成または管理するインストールをビルドすることができます。

IIS スキャナー (IIScan.exe) は、IIS Web サイトをスキャンして、InstallShield の [IIS 構成] ビューで構成可能な設定の値を記録するコマンドライン ツールです。IIScan.exe は、すべての値を含む XML ファイルを作成します。この XML ファイルを使って、[IIS 構成] ビューに値をインポートできます。

IIScan.exe は、以下の場所にインストールされています:

InstallShield Program Files フォルダー¥System¥iisscan.exe

タスク ***既存の Web サイトをスキャンして、[IIS 構成] ビューで構成可能なすべての IIS データを記録するには、以下の手順に従います:***

1. IIScan.exe ツールを、スキャンする IIS Web サイトを持つサーバー上に配置します。
2. コマンドライン プロンプトで、特定の Web サイトをスキャンする IIS スキャナーを起動するコマンドライン ステートメントを入力します。次は、サンプル ステートメントです:

   iisscan.exe –website "Site Name" –outfile "C:¥PathToFile¥FileName.xml"

   IIScan.exe に関する詳細については、「IIScan.exe」を参照してください。

IIS スキャナーは、指定の Web サイトに関する IIS 情報を含む XML ファイルを作成します。

タスク ***XML ファイルからのデータを InstallShield プロジェクトにインポートするには、以下の手順に従います:***

1. IIS スキャナーが作成した XML ファイルを取得して、それを InstallShield を持つマシンからアクセス可能な場所に配置します。
2. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
3. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトのインポート]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. IIS スキャナーが作成した XML ファイルを選択してから、**[開く]** をクリックします。

InstallShield が Web サイトとその設定を [IIS 構成] ビューにインポートします。インポート中に Web サイトの設定、仮想ディレクトリ、およびアプリケーションすべてがこのビューで構成されます。

ヒント: *InstallShield では、特定の IIS データ (たとえば Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、またはそれらの設定) が [IIS 構成] ビューにインポートされないように防止するためのフィルターを構成できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする

エディション: *IIS データを InstallShield プロジェクトにインポートする機能は、InstallShield Premier Edition のみで使用できます。*

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

IIS スキャナーが生成した XML ファイルを使って IIS Web サイトとその設定を [IIS 構成] ビューにインポートするとき、インストールで構成する必要がない IIS 設定もインポートされる場合があります。XML ファイルを使って IIS 設定をインポートするたびにこれらの設定が構成されないようにするためには、Filters.xml を編集します。このファイルを使って、インポート時に無視する任意の IIS 設定を指定できます。

Filters.xml は、以下の場所にあります:

InstallShield Program Files フォルダー¥Support

このファイルは編集後も同じ場所になくてはなりません。そうでない場合、IIS インポートが正しく動作しません。

ヒント: Filters.xml *ファイルを使って、COM 抽出中にどのレジストリ項目を除外するのかを制御することもできます。詳細については、「COM 抽出のレジストリ変更をフィルターする」を参照してください。*

Filters.xml *ファイルを使って、依存関係のスキャン中にどのファイルを除外するまたは含めるのかを制御することもできます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。*

IIS データの除外

Filters.xml ファイルの <Exclude> 要素では、インポートしない各 IIS アイテムのサブ要素 (Web サイト、アプリケーション、仮想ディレクトリ、およびアプリケーション プール) を追加します。また、インポートしない各 IIS 設定のサブ要素を追加することもできます。ここでリストされたアイテムと設定は、プロジェクトの [IIS 構成] ビューには追加されません。

<Exclude> 要素における IIS データ指定の例

TestAppPool と名付けられたアプリケーション プールが [IIS 構成] ビューにインポートされないように防ぐには、<Exclude> 内に以下の行を追加します。

```
<IisScanItem name="TestAppPool" type="4"/>
```

type は必須です。次のテーブルは、使用可能な IIS アイテムの種類の一覧です:

テーブル 3 ・ <IisScanItem> サブ要素の type 属性のとして使用可能な属性

属性値	説明
1	Web サイト
2	アプリケーション
3	仮想ディレクトリ
4	アプリケーション プール

指定の値を持つ IIS 設定が [IIS 構成] ビューにインポートされないように防ぐには、<Exclude> 内に以下の行を追加します。

```
<IisScanProperty name="PropertyName" value="PropertyValue"/>
```

<Exclude> 要素内にこの行があると、InstallShield は **PropertyValue** と等しい、またはこれを含む値を持つ **PropertyName** と名付けられた設定をブロックします。

メモ・ *プロパティ名と値は Windows Installer プロパティまたはその値ではありません。これらは、InstallShield の [IIS 構成] ビューで構成可能な設定の名前と値の省略形です。*

重要・ *その場合、XML コードが適切に書かれていることを再確認してください。不適切な場合には、すべてのフィルターが失敗します。多くの場合、インターネット エクスプローラーで* Filters.xml *ファイルを開いて、不適切に書かれた XML コードを確認することができます。<Filters>、<Include>、および <Exclude> 要素は縮小および展開が可能です。これらが不可能な場合、コードにエラーが無いか確認してください。*

<Exclude> または <Include> 要素にサブ要素を追加した場合、それらが誤ってコメントアウトされたセクションに配置されていないことを確認してください。InstallShield は Filters.xml *ファイルのコメントアウト部分を無視します。*

次のサンプル XML コードで、Filters.xml ファイルの形式を説明します。

```
<Filters>
 <Include>
  <!—この要素にファイルを追加する方法
  —>
  <File name="mfc42.dll" We="needthis"/>
 <Include>
 <Exclude>
  <!—この要素にファイルを追加する方法
  —>
```

```
 <Registry key="HKEY_CLASSES_ROOT¥Interface¥{00020404-0000-0000-C000-000000000046}"/>
 <File name="12520437.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 <File name="12520850.cpx" path="[SystemFolder]" wrp="4.0-10.0" />
 <IisScanProperty name="AppMappings" value="*;;*;0" />
 <IisScanProperty name="CustomErrors" value="C:¥WINDOWS¥help¥iisHelp¥common" />
 </Exclude>
</Filters>
```

ネスト仮想ディレクトリの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

既存の仮想ディレクトリの下に、仮想サブディレクトリを作成することができます。

また、インストールの一部としてインストールされる仮想ディレクトリの下にも仮想サブディレクトリを作成することができます。親仮想ディレクトリは、仮想サブディレクトリの前にインストールされる必要があります。

プロジェクト: *InstallScript の場合、親仮想ディレクトリが必ず子仮想ディレクトリの前にインストールされるようにするには、親仮想ディレクトリを含むコンポーネントが、子仮想ディレクトリを含むコンポーネントの前にインストールされる必要があります。*

タスク ***既存の仮想ディレクトリの下に仮想ディレクトリを作成するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ネスト仮想ディレクトリを含める Web サイトを選択します。
3. 新しい Web サイトを右クリックして **[新しい仮想ディレクトリ]** を選択します。新しい仮想ディレクトリが追加されます。
4. **[全般]** タブをクリックします。
5. **"名前"** 設定で、既存のディレクトリ名と、作成するネスト仮想サブディレクトリの名前を指定します。2 つの名前はスラッシュで区切ります。

   たとえば、**VirtualDirectory** という名前の既存の仮想ディレクトリの下に **MySubdirectory** という名前の仮想ディレクトリを作成する場合、次のように入力します:

   VirtualDirectory/MySubdirectory

メモ: *親ディレクトリがターゲット システムに既に存在しない場合、エンドユーザーが IIS マネージャーでそのディレクトリを開いたとき、ターゲット システムでエラーが表示されます。*

TCP ポート番号とサイト番号の構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShieldでは、プロジェクトで Web サイトの TCP ポートとサイト番号を指定できます。これらの設定は、実行時に新しい Web サイトが作成されるか、既存の Web サイトが更新されるのかを判断するのに役立ちます。これらはまた、[インターネット インフォーメーション サービス] ビューで構成された Web サイトの設定が、ターゲット システムの Web サイトに適用されるかどうかにも影響します。

タスク

Web サイトに TCP ポート番号とサイト番号を指定するには、以下の手順に従います:

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、構成する Web サイトを選択します。
3. **[Web サイト]** タブをクリックします。
4. **[TCP ポート番号]** ボックスと **[サイト番号]** ボックスに、適切な番号を入力します。

実行時の動作

実行時に、インストールが次の規則に基づいて Web サイト、アプリケーション、および仮想ディレクトリを作成します:

テーブル 4 · 各種サンプル "TCP ポート番号" および "サイト番号" 設定値の実行時の結果

InstallShield の "TCP ポート番号" 設定	InstallShield の "サイト番号" 設定	実行時の結果
0	ゼロ以外の値	"TCP ポート番号" 設定が 0 で、"サイト番号" 設定が 0 以外のとき、インストールは Web サイトのアプリケーションおよび仮想ディレクトリをシステムの最初のサイト番号にインストールします。指定されたサイト番号は無視されます。 たとえば、ターゲット システム上の最初のサイト番号が 1 のとき、"サイト番号" 設定に異なるゼロ以外の番号 (たとえば、3) が指定された場合でも、インストールは Web サイトのアプリケーションと仮想ディレクトリをサイト番号 1 にインストールします。 インストールは、[IIS 構成] ビューで構成された Web サイト設定を、ターゲット システム上の Web サイトに一切適用しません。

テーブル 4 - 各種サンプル "TCP ポート番号" および "サイト番号" 設定値の実行時の結果 (続き.)

InstallShield の "TCP ポート番号" 設定	InstallShield の "サイト番号" 設定	実行時の結果
80 (ゼロ以外の値)	0 (デフォルト値)	指定された TCP ポートがターゲット システムに存在するとき、インストールは TCP ポートで実行中の Web サイト (この例では、ポート 80) にアプリケーションと仮想ディレクトリをインストールします。インストールは、[IIS 構成] ビューで構成された Web サイト設定を、ターゲット システム上の Web サイトに一切適用しません。 TCP ポートがターゲット システムに存在しない場合、プロジェクトで構成された Web サイトの設定を使って新しい Web サイトが作成されます。またインストールは、Web サイトのアプリケーションと仮想ディレクトリをインストールします。
81 (ゼロ以外の値)	3 (ゼロ以外の値)	指定された TCP ポートおよびサイト番号がターゲット システムに存在するとき、インストールは TCP ポートで実行中の Web サイト (この例では、ポート 3) にアプリケーションと仮想ディレクトリをインストールします。インストールは、[IIS 構成] ビューで構成された Web サイト設定を、ターゲット システム上の Web サイトに一切適用しません。 TCP ポートがターゲット システムに存在するが、サイト番号が存在しない場合、インストールは新しい Web サイトおよびそのアプリケーションと仮想ディレクトリを、既存のポートに新しいサイト番号を使ってインストールします。またインストールは、[IIS 構成] ビューで設定された Web サイトのプロパティを構成します。 TCP ポートがターゲット システムに存在しないが、サイト番号が存在する場合、インストールは新しい Web サイトおよびそのアプリケーションと仮想ディレクトリをこの TCP ポートにインストールします。またインストールは、[IIS 構成] ビューで設定された Web サイトのプロパティを構成します。

IIS Web サイトとそのアプリケーションおよび仮想ディレクトリを、次に利用できる新しいサイト番号にインストールする

次の手順は、IIS Web サイトとそのアプリケーション、および仮想ディレクトリを、次の利用できる新しいサイト番号にインストールする方法を説明します。手続きのは、プロジェクトの種類によって異なりますので注意してください。

タスク ***基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトで、次に利用できる新しいサイト番号にインストールするには、以下の手順に従います:***

1. 新しい Windows Installer プロパティを作成して、その値を **INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER** に設定します:
   a. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
   b. **[新しいプロパティ]** ボタンをクリックします。ビューの下に新しい行が追加されます。
   c. **[名前]** 列に、新しいプロパティを入力します。例:

      MYPROPERTY

   d. **[値]** 列で、次の値を入力します:

      INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER
2. Web サイトのサイト番号のプロパティを指定します:
   a. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
   b. **[Web サイト]** エクスプローラーで、構成する Web サイトを選択します。
   c. **"サイト番号"** 設定に、ステップ 1c で作成したプロパティを入力します。プロパティはすべて大文字で、角括弧で囲みます。例:

      [MYPROPERTY]

実行時、インストールは Web サイトとそのアプリケーション、および仮想ディレクトリを次に利用できる新しいサイト番号にインストールします。

タスク ***InstallScript インストール中、次に使用できる新しいサイト番号にインストールするには、以下の手順に従います:***

1. 新しい文字列エントリを作成して、その値を INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER に設定します:
   a. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
   b. **[新しい文字列エントリ]** ボタンをクリックします。**[文字列エントリ]** ダイアログ ボックスが開きます。
   c. **[文字列識別子]** ボックスに、新しい変数名を入力します。変数名はすべて大文字でなくてはなりません。例:

      MYPROPERTY

   d. **[値]** ボックスに、次の値を入力します:

      INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER
2. Web サイトのサイト番号の変数を指定します:
   a. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
   b. **[Web サイト]** エクスプローラーで、構成する Web サイトを選択します。
   c. **"サイト番号"** 設定に、ステップ 1c で作成した変数を入力します。すべて大文字でなくてはなりません。

実行時、インストールは Web サイトとそのアプリケーション、および仮想ディレクトリを次に利用できる新しいサイト番号にインストールします。

Web サイトの IIS ホスト ヘッダー名を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、ホスト ヘッダー名を指定して、インストール中にインストールされる IIS Web サイトを識別することができます。ホスト ヘッダー（ドメイン名とも呼ばれます）を利用して、複数の Web サイトを Web サーバー上の IP アドレスに割り当てることができます。

タスク ***Web サイトのホスト ヘッダー名を指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ホスト ヘッダー名を指定する Web サイトを選択します。
3. **"ホスト ヘッダー名"** 設定で、使用するホスト ヘッダー名を入力します。例:

   www.mycompany.com

Web サイトの SSL 証明書を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

サーバー証明書を利用して、ユーザーは Web サーバーの認証および Web コンテンツの有効性の確認を行うことができると共に、セキュリティで保護された接続を確立することができます。では、実行時にインストールできるように、インストールに Web サイトのサーバー証明書を含めることができます。

タスク ***Web サイトにインストールする SSL 証明書を指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、SSL 証明書を指定する Web サイトを選択します。
3. **"SSL 証明書"** 設定で、省略記号ボタン (...) をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. インストールするセキュリティ証明書ファイル (.cer または .pfx) を選択して、**[開く]** をクリックします。
5. 証明書にパスワードが必要な場合、**"SSL 証明書パスワード"** 設定でパスワードを指定します。

.cer ファイルが Binary テーブルに格納されます。実行時、インストールで Web サイトと仮想ディレクトリがインストールされるとき、SSL 証明書もインストールされます。

実行時に IIS 証明書ファイルを設定する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

エンド ユーザーが SSL 証明書に提供する IIS 証明書ファイルを参照し、インストール実行時にパスワードを入力できる、**[IIS の SSL を構成する]** ダイアログ (IISBrowseSSLCertificate) をインストーラーに含めるかどうかを選択できます。

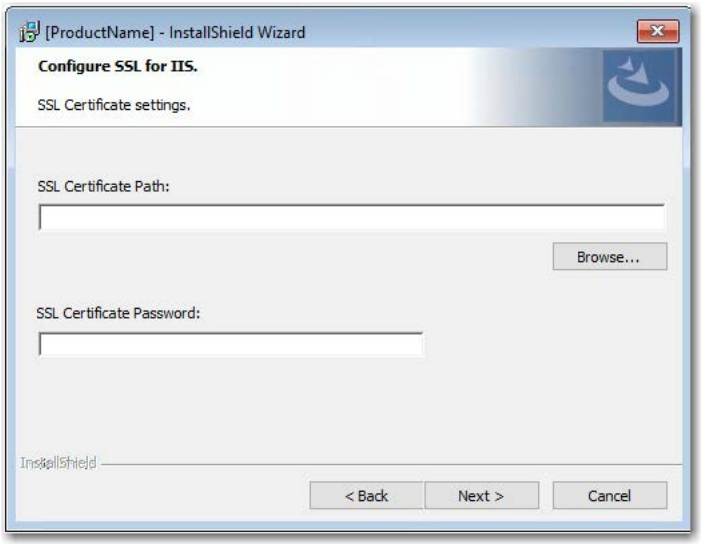


図 1: IIS ダイアログの SSL を構成する

インストーラーに **[IIS の SSL を構成する]** ダイアログ を追加するには、次の手順に従います:

タスク ***インストーラーに [IIS の SSL を構成] ダイアログ を追加するには:***

1. **[サーバー構成]** の下の **ビュー リスト** にある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーを右クリックして、**[Web サイトの追加]** をクリックします。InstallShield が新しい Web サイトを追加します。
3. 新しい Web サイトを選択して **[セキュリティ > セキュリティで保護された通信]** の下にある **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティに移動します。
4. **"SSL 証明書"** および **"SSL 証明書パスワード"** プロパティを次の値に設定します:

プロパティ	値
SSL 証明書	[IS_IIS_WEBCERTPATH]
SSL 証明書のパスワード	[IS_IIS_WEBCERTPASSWORD]

5. **[ユーザー インターフェイス > ダイアログ]** ビューを開いて、ダイアログ シーケンスに **IISBrowseSSLCertificate** ダイアログを追加します。

   [編集] ボックス (**IISWebCertPassword** と **IISWebCertPath**) およびプッシュボタン (**BrowseCertificate**) イベントの **IISBrowseSSLCertificate** ダイアログで、ユーザーが構成した SSL 証明書およびパスワードのプロパティ名がプロパティ名を更新する必要があります。

ファイルを IIS 仮想ディレクトリに追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

タスク ***ファイルを IIS 仮想ディレクトリに追加する:***

1. IIS Web サイトをプロジェクトに追加していない場合は、それを行います。InstallShield が自動的に定義済みのパス **[IISROOTFOLDER]** を **[ファイルとフォルダー]** ビューへ追加します
2. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
3. **[機能]** リストで、ファイルに関連付ける機能を選択します。
4. **[インストール先コンピューターのフォルダー]** ペインで、ファイルを **[IISROOTFOLDER]** フォルダー、または **[IISROOTFOLDER]** フォルダーのサブフォルダーに追加します。
5. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。

6. 新しい仮想ディレクトリを作成します。
7. **[Web サイト]** エクスプローラーで、作成した仮想ディレクトリをクリックします。
8. **"コンテンツ ソース パス (ローカルまたは UNC)"** 設定で省略記号ボタン (...) をクリックします。**[ディレクトリの参照]** ダイアログ ボックスが開きます。基本の MSI または InstallScript MSI プロジェクトでは、このダイアログ ボックスで Windows Installer プロパティ (たとえば、[IISROOTFOLDER]) を選択するか、または新しいプロパティを作成できます。InstallScript プロジェクトでは、このダイアログ ボックスで InstallScript 変数 (たとえば、<IISROOTFOLDER>) を選択するか、または新しい変数を作成できます。デフォルトでは、これらのファイルは IISROOTFOLDER に格納されています。
9. **[ファイルとフォルダー]** ビューで追加した新しいファイルを含むディレクトリと同じターゲット ディレクトリを入力します。
10. **[OK]** をクリックします。

インストール時、ファイルはターゲット ディレクトリ フォルダーへコピーされます。さらに IIS が存在する場合、ターゲット システム上のフォルダーに仮想ディレクトリが構成されます。

[IIS 構成] ビューからアプリケーションと仮想ディレクトリを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

タスク ***アプリケーションまたは仮想ディレクトリをインストールから削除するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、アプリケーションまたは仮想ディレクトリを右クリックしてから **[削除]** をクリックします。

アプリケーション プールの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

アプリケーション プールは、1 つ以上の Web アプリケーションを 1 つ以上のワーカー プロセスにルートするときの方法を指定するひとまとまりの構成設定です。InstallShield では、アプリケーション プールを作成して、Web サイトおよび仮想ディレクトリをその作成したアプリケーション プールに関連付けることができます。

プロジェクト: *InstallScript プロジェクトでアプリケーション プールを作成する場合で、アプリケーション プールに仮想ディレクトリ、アプリケーション、または Web サイトに関連付けるには、同じコンポーネント内でアプリケーション プールと仮想ディレクトリ、アプリケーション、または Web サイトを作成しなくてはなりません。これを行うことで、仮想ディレクトリ、アプリケーション、または Web サイトが作成される前にアプリケーション プールが作成されるようになります。これは、IIS 6 以降の要件で、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの制限事項ではありません。*

メモ: *アプリケーション プールは、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

タスク ***[IIS 構成] ビューで新しいアプリケーション プールを追加するには、以下の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[アプリケーション プール]** エクスプローラーを右クリックして、**[アプリケーション プールの追加]** をクリックします。InstallShield が新しいアプリケーション プール項目を追加します。
3. 新しいアプリケーション プール アイテムの名前を変更して、その設定を構成します。

[IIS 構成] ビューからアプリケーション プールを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

タスク ***アプリケーション プールをインストールから削除するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[アプリケーション プール]** エクスプローラーで、アプリケーション プールを右クリックし **[削除]** をクリックします。

Web サービス拡張の追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

Web サービス拡張は、基本の IIS 機能をさらに拡張して、静的コンテンツの提供以上のことをできるようにする IIS 機能です。Web サービス拡張の例として、アクティブ サーバー ページ (.asp)、ASP.NET、およびサーバー側インクルード (SSI) などがあります。

InstallShield では、Web サービス拡張をインストールに追加することができます。

メモ: *サービス拡張は、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。詳細については、「InstallShield における IIS サポートのバージョン固有情報」を参照してください。

タスク ***Web サービス拡張をインストールに追加するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サービス拡張]** エクスプローラーを右クリックして、**[Web サービス拡張の追加]** をクリックします。InstallShield が Web サービス拡張を追加します。
3. 新しいWeb サービス拡張アイテムの名前を変更して、その設定を構成します。

[IIS 構成] ビューから Web サービス拡張を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

タスク ***Web サービス拡張をインストールから削除するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サービス拡張]** エクスプローラーで、Web サービス拡張を右クリックして、**[削除]** をクリックします。

IIS サポートの機能とコンポーネントの関連付け

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

コンポーネントと IIS データ (Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張) は、一対多の関係を持ちます。したがって、1 つのコンポーネントに複数の IIS 要素を関連付けることが可能です。コンポーネントがインストールされている間、コンポーネントに関連付けられているすべての Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プールおよび Web サービス拡張が実行時に作成されます。

プロジェクト: *InstallScript プロジェクトで Web サイトがコンポーネントに関連付けられている場合、その Web サイトに追加されるアプリケーションまたは仮想ディレクトリも同じコンポーネントに関連付けなくてはなりません。したがって、1 つまたは複数のアプリケーション、または仮想ディレクトリを含む Web サイトのコンポーネントを変更しようとすると、InstallShield はメッセージ ボックスを表示して、Web サイトのアプリケーションおよび仮想ディレクトリのすべてに対して、同じコンポーネントの変更が行われることを通知します。このメッセージ ボックスは、Web サイトに含まれる任意のアプリケーションまたは仮想ディレクトリのコンポーネントを変更しようとした場合にも表示されます。どちらの場合も、メッセージ ボックスはコンポーネントの変更を続行するか、キャンセルするか選択肢を提供します。*

必須ではありませんが、基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、Web サイトのすべてのアプリケーションおよび仮想ディレクトリと同様に、Web サイトを同じコンポーネントに保持することが推奨されます。

InstallScript プロジェクトでアプリケーション プールを作成する場合で、アプリケーション プールに仮想ディレクトリ、アプリケーション、または Web サイトに関連付けるには、同じコンポーネント内でアプリケーション プールと仮想ディレクトリ、アプリケーション、または Web サイトを作成しなくてはなりません。これを行うことで、仮想ディレクトリ、アプリケーション、または Web サイトが作成される前にアプリケーションプールが作成されるようになります。これは、IIS 6 以降の要件で、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの制限事項ではありません。

Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プールまたは Web サービス拡張を含めるコンポーネントは、必要に応じていつでも [IIS 構成] ビューで変更することができます。

タスク ***Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張をプロジェクト内の異なるコンポーネントに関連付けるには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. エクスプローラーで、コンポーネントに関連付ける Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張をクリックします。
3. **[コンポーネント]** 設定で、選択した IIS データを含める既存のコンポーネントの名前を選択します。省略記号ボタン (...) ボタンをクリックして、既存のコンポーネントを選択することもできますし、新しいコンポーネントをこの IIS データに作成することもできます。

ヒント・ *プロジェクトからコンポーネントを削除した場合、コンポーネントに関連付けられたすべての Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張もプロジェクトから削除されます。*

Web サイト、アプリケーション、および仮想ディレクトリのアンインストール

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インストールによって作成された Web サイトは、次の条件の両方が True とならない限り、削除されません:

- Web サイトにアプリケーションまたは仮想ディレクトリが含まれていない。
- [IIS 構成] ビューにある Web サイトの ***"アンインストール時に削除する"*** 設定の値が [はい] である。

メモ・ *Web サイトがコンポーネントと関連付けられている場合、Web サイトの* ***"アンインストール時に削除する"*** *設定は、そのコンポーネントの "パーマネント" 設定 (基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトの場合) またはそのコンポーネントの "アンインストール" 設定 (InstallScript プロジェクトの場合) に対応します。つまり、Web サイトの* ***"アンインストール時に 削除する"*** *設定を選択またはクリアすると、自動的にコンポーネントの "パーマネント" 設定または "アンインストール" 設定が適切に更新されます。*

コンポーネントがアンインストールされると、そのすべての Web サイト、アプリケーション、および仮想ディレクトリもアンインストールされます。パーマネントと構成されている場合、コンポーネントはアンインストールされません。詳細については、「アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する」を参照してください。

Web サービス拡張とアプリケーション プールのアンインストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

メモ: *サービス拡張、アプリケーション プール、および関連プロパティは、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。詳細については、「InstallShield における IIS サポートのバージョン固有情報」を参照してください。

インストール内の Web サービス拡張およびアプリケーション プールと同じ名前を持つ、IIS Manager にインストールされている Web サービス拡張およびアプリケーション プールはどれも、それらのコンポーネントがパーマネントとマークされていない限り、関連付けられている機能がアンインストールされるときに必ずアンインストールされます。Web サービス拡張またはアプリケーション プールがインストール以外によって作成された場合でも、(Web サービス拡張およびアプリケーション プールの) 名前がインストールに含まれるものと一致する場合は、アンインストール時にターゲット システムから削除されます。唯一の例外は DefaultAppPool というアプリケーション プールで、これはアンインストール時に削除されることはありません。

タスク ***アンインストール中に Web サービス拡張をアンインストールするように構成するには、次の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サービス拡張]** エクスプローラーで、Web サービス拡張を選択します。
3. **"コンポーネントをパーマネントとしてマークする"** 設定で、**[いいえ]** を選択します。

タスク ***アンインストール中にアプリケーション プールをアンインストールするように構成するには、次の手順を実行します。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[アプリケーション プール]** エクスプローラーで、アプリケーション プールを選択します。
3. **"コンポーネントをパーマネントとしてマークする"** 設定で、**[いいえ]** を選択します。

"コンポーネントをパーマネントとしてマークする" 設定で [はい] が選択されていと、Web サービス拡張またはアプリケーション プールはアンインストールの実行後もターゲット マシンに残ります。

ヒント: *"コンポーネントをパーマネントとしてマークする" 設定の値を変更すると、Web サービス拡張またはアプリケーション プールを含むコンポーネントの "パーマネント" 設定 (基本の MSI、DIM、InstallScript MSI、およ*

びマージ モジュール プロジェクトの場合)、または "アンインストール" 設定 (InstallScript プロジェクトの場合) の値に反映します。したがって、"コンポーネントをパーマネントとしてマークする" 設定で [はい] を選択して、かつコンポーネントにファイル、ショートカット、レジストリ エントリなどの他のデータが含まれている場合、IIS データおよびコンポーネントの他のデータはアンインストール時にアンインストールされません。

Web サイトまたはアプリケーションの ASP.NET バージョンを設定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、インストールにある Web サイトまたはアプリケーションの ASP.NET バージョンを設定することができます。ASP.NET バージョンを指定すると、Web サイトまたはアプリケーションが作成された後、ASP.NET IIS の Registration Tool (Aspnet_regiis.exe) がインストールで実行され、Web サイトまたはアプリケーションを指定したバージョンにマッピングします。

Web サイトに ASP.NET バージョンを指定すると、IIS はその値を実行時に作成された Web サイトおよびすべてのそのアプリケーションに使用します。

重要: *マイクロソフト社は、Aspnet_regiis.exe ツールの機能に制限があるため、Windows Vista または Windows Server 2008 システムでの使用を推奨していません。結果として、場合により、アプリケーションのマッピングを [IIS 構成] ビューで手動で定義する必要があります。詳細については、「Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する」を参照してください。*

ASP.NET 3.0 には Aspnet_regiis.exe ツールは含まれていません。したがって、ASP.NET バージョンを バージョン 3 の ASP.NET に設定できません。

タスク ***プロジェクトで Web サイトまたはアプリケーションの ASP.NET バージョンを指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ASP.NET バージョンを指定する Web サイトまたはアプリケーションを選択します。Web サイトまたはアプリケーションの設定が右側に表示されます。
3. **"ASP .NET バージョン"** 設定で、アプリケーションに必要な .NET Framework のバージョン番号を完全な形で入力するか、一覧から選択します。

   たとえば、バージョン 2 の ASP.NET を指定する場合、**2.0.50727** と入力します。バージョン 1.1 の ASP.NET を指定するには、**1.1.4322** と入力します。

ヒント: *.NET Framework を持つ Windows の 64 ビット バージョンでインストールが実行可能な場合、Web サイトまたはアプリケーションの "ASP.NET プラットフォーム" 設定を使って、どの ASP.NET プラットフォーム (32 ビットまたは 64 ビット) を使って ASP.NET バージョンに Web サイトまたはアプリケーションをマップするのかを指定できます。システム上で Enable32BitAppOnWin64 プロパティが True に設定されていない限り、64 ビット システム上で 32 ビット ASP.NET を登録することはできません。*

繰り返しまたは 1 回限りの ASP.NET 登録を行うかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューの **"アプリケーション"** 設定にある **[ASP.NET 登録]** 設定を使って、繰り返しまたは 1 回限りの ASP .NET 登録を行うかどうかを指定できます。この機能を使って、同じ Web サイトに ASP.NET アプリケーションおよび ASP.NET コア アプリケーションの両方をインストールすることができます。

タスク ***プロジェクトでアプリケーションの ASP.NET 登録を指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、ASP.NET 登録を指定する アプリケーションを選択します。アプリケーションの設定が右側に表示されます。
3. **"ASP .NET 登録"** 設定で次のオプションから 1 つを選択します:
   - **再帰的** – 指定されたアプリケーションおよびすべてのサブアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。
   - **非再帰的** – 特定のアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。サブアプリケーションは変更されません。

64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

IIS インストールが .NET Framework を持つ Windows の 64 ビット バージョンで実行可能な場合、Web サイトまたはアプリケーションに [IIS 構成] ビューの "ASP.NET プラットフォーム" 設定を使用できます。この設定を使って、どの ASP.NET プラットフォームを使って Web サイトまたはアプリケーションを ASP.NET バージョンにマップするのかを指定できます。デフォルトで、この設定は構成されていません。この設定の値を変更しなかった場合、インストールは 32 ビット サポートが不要であると判断します。

"ASP.NET プラットフォーム" と "ASP.NET バージョン" 設定で構成した値によって、デフォルトで実行時に以下の動作が発生します:

- ターゲット システムが Windows 32 ビット バージョンを持つ場合、指定された ASP.NET の 32 ビットバージョンが使用されます。
- ターゲット システムが Windows 64 ビット バージョンを持つ場合で、ASP.NET プラットフォームに 64 ビットが指定されるとき、ターゲット システム上の Enable32bitAppOnWin64 プロパティは False でなくてはなりません。そうでない場合、32 ビット バージョンをサポートするように構成されているシステム上で ASP.NET の 64 ビット バージョンを使用すると、ASP.NET が破損する原因となります。このため、ASP.NET サービスの破損を避けるためにインストールは中止されます。
- ターゲット システムが Windows 64 ビット バージョンを持つ場合で、ASP.NET プラットフォームに 32 ビットが指定されるとき、ターゲット システム上の Enable32bitAppOnWin64 プロパティは True でなくてはなりません。そうでない場合、64 ビット バージョンをサポートするように構成されているシステム上で ASP.NET の 32 ビット バージョンを使用すると、ASP.NET が破損する原因となります。このため、ASP.NET サービスの破損を避けるためにインストールは中止されます。

[IIS 構成] ビューでアプリケーション プールの "32 ビット アプリケーションを有効にする" 設定を構成して、IIS 7 が搭載されたターゲット システム上の Enable32bitAppOnWin64 プロパティを設定できます。実行時、インストールはインストールするアプリケーション プールに適切な Enable32bitAppOnWin64 プロパティを設定します。

IIS 6 が搭載されたシステム上では、Enable32bitAppOnWin64 プロパティは IIS サーバー上のすべての Web サイトとアプリケーションに反映するグローバル プロパティです。インストール実行時にプロパティの値を変更すると、そのサーバー上にインストール済みの Web サイトとアプリケーションが失敗する原因となるため、InstallShield で作成されたインストールは、ターゲット システムに IIS 6 が搭載されている場合、実行時に Enable32bitAppOnWin64 プロパティを変更しません。

実行時、IIS 6 を持つシステム上でインストールが Enable32bitAppOnWin64 プロパティをチェックするように設定できます。そのとき、製品の要件およびチェック結果に基づいて、たとえば 32 ビット IIS 6 アプリケーションまたは 64 ビット IIS アプリケーションを含む特定のコンポーネントのインストールをスキップして、残りのファイル転送を続行できます。

InstallShield には、Enable32bitAppOnWin64 プロパティがどのように設定されているかを検出できるサンプル Windows Installer DLL ファイルが含まれています。DLL ファイルのリリースおよびデバッグ バージョン、およびファイルの作成に使用された Visual Studio 2008 C++ プロジェクトは以下の場所にインストールされています:

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility

プロジェクトで Windows Installer DLL ファイルを使用するためには、まず DetectIIS6Compat.dll ファイルで **DetectIIS6AppPool32BitSupport** 関数を呼び出すカスタム アクションを作成する必要があります。

タスク

IIS 6 が搭載されているシステム上の 32 ビット アプリケーションのサポートをチェックするカスタム アクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]**（基本の MSI、InstallScript MSI の場合）または **[カスタム アクション]**（DIM、マージ モジュール プロジェクトの場合）をクリックします。
2. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが追加されます。
3. アクションの名前を、たとえば **DetectIIS6AppPool32BitSupport** に変更します。
4. 右側のペインでカスタム アクションの設定を構成します:
   - **"DLL ファイル名"** 設定で、DetectIIS6Compat.dll ファイルへのパスを指定します。通常のパスは以下のとおりです:

     <ISProductFolder>¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility¥DetectIIS6Compat.dll
   - **"関数名"** 設定で、次の名前を入力します:

     DetectIIS6AppPool32BitSupport
   - **"スクリプト内実行"** 設定で、**[即時実行]** を選択します。
   - 1 つまたは複数のシーケンス設定で適切なオプションを選択して、必要に応じて、カスタム アクションをスケジュールします。
   - "条件" 設定に次の条件を入力します:

     VersionNT<600

     Windows Server 2008 以降または Windows Vista 以降は IIS 7 以降が搭載されているため、この条件は、カスタム アクションがこれらのオペレーティング システムで実行されないように保証します。

実行時、特定の条件下でカスタム アクションは Windows Installer プロパティ ISIIS6APPPOOLSUPPORTS32BIT を設定します。

テーブル 5 ・ IIS 6 システム上で 32 ビット アプリケーションのサポートをチェックする Windows Installer プロパティ

ターゲット システム環境	結果
32 ビット Windows、IIS 6 （Enable32bitAppOnWin64 の値は関係ありません。）	ISIIS6APPPOOLSUPPORTS32BIT は 1 に設定されます。
64 ビット Windows、IIS 6、Enable32bitAppOnWin64=true	ISIIS6APPPOOLSUPPORTS32BIT は 1 に設定されます。
64 ビット Windows, IIS 6、Enable32bitAppOnWin64=false	ISIIS6APPPOOLSUPPORTS32BIT は設定されません。

条件ステートメントで ISIIS6APPPOOLSUPPORTS32BIT プロパティを使用できます。たとえば、64 ビット アプリケーションをサポートしない IIS 6 サーバー上で製品の使用が不可能な場合、[一般情報] ビューの "インストール条件" 設定に次の条件ステートメントを入力します:

Not ISIIS6APPPOOLSUPPORTS32BIT

別の方法として、エラー カスタム アクションを作成することもできます。実行時、そのカスタム アクションの条件が True の場合、エラーが表示されます。したがって、64 ビット アプリケーションをサポートしない IIS 6 サーバーで製品の使用が不可能な場合、エラー カスタム アクションに次の条件を使用します:

ISIIS6APPPOOLSUPPORTS32BIT=1

32 ビット アプリケーション サポートが有効化されている IIS 6 サーバー上にのみコンポーネントをインストールする場合、[コンポーネント] ビューの "条件" 設定に次の条件ステートメントを入力します:

ISIIS6APPPOOLSUPPORTS32BIT=1

Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、ファイル名拡張子とこれらのファイルを処理するアプリケーション間のマッピングを定義することができます。

タスク ***アプリケーションのマッピングの追加/編集/削除を行うには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、構成する Web サイト、アプリケーション、または仮想ディレクトリを選択します。
3. **"アプリケーションのマッピング"** 設定で、省略記号ボタン (...) をクリックします。**[アプリケーションのマッピング]** ダイアログ ボックスが開きます。
4. 以下のいずれかを実行します。
   - 新しいマッピングを追加するには、**[追加]** ボタンをクリックします。**[アプリケーション拡張子マッピング]** ダイアログ ボックスが開きます。詳細については、「[アプリケーション拡張子マッピング] ダイアログ ボックス」を参照してください。
   - 既存のマッピングを変更するには、編集するマッピングを選択して、**[編集]** ボタンをクリックします。
   - 既存のマッピングを削除するには、それを選択して、**[削除]** ボタンをクリックします。

Web サイト、アプリケーション、または仮想ディレクトリのアプリケーションのマッピングを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

InstallShieldWeb サイト、アプリケーション、または仮想ディレクトリのタイムアウトのパラメーターを指定する

タスク ***タイムアウトのパラメーターを指定するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、構成する Web サイト、アプリケーション、または仮想ディレクトリを選択します。
3. **"アプリケーションの設定"** 領域で、**[構成]** ボタンをクリックします。**[アプリケーションのマッピング]** ダイアログ ボックスが開きます。
4. **"セッション タイムアウト(分)"** と **"ASP スクリプト タイムアウト(秒)"** 設定で、適切なタイムアウト値を指定します。

ターゲット システムに IIS 6 以前が搭載されているか、または IIS 6 メタベースの互換性機能があるかどうかを判別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

実行時に、インストールによって、ターゲット システムに IIS メタベースおよび IIS 6 構成との互換性機能がインストールされているかどうか、または IIS 以前がインストールされているかを検出できます。製品の要件と検出結果に基づいて、インストールを終了およびエラー メッセージを表示できます。

たとえば、Web サービス拡張は IIS 6 を持つシステムにインストールできます。IIS 7 を持つシステムでは、Web サービス拡張は IIS メタベースおよび IIS 6 構成との互換性機能がインストールされている場合のみインストール可能です。このため、IIS 6 が存在するか、IIS 6 との互換性機能がインストールされていることをインストールが検証するように構成することが必要な場合があります。これらのどちらの条件も False の場合、インストールが終了してエラー メッセージが表示されます。

InstallShield には、以下のどちらかが True であることを検出するためのサンプル Windows Installer DLL ファイルが含まれています:

- ターゲット システムには IIS 7 が搭載されていて、IIS 6 メタベースおよび IIS 6 構成の互換性機能がインストールされている。
- ターゲット システムに IIS 6 以前が搭載されている。

DLL ファイルのリリースおよびデバッグ バージョン、および ファイルの作成に使用された Visual Studio 2008 C++ プロジェクトは以下の場所にインストールされています:

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility

プロジェクトで Windows Installer DLL ファイルを使用するためには、まず DetectIIS6Compat.dll ファイルで **DetectIIS6Interfaces** 関数を呼び出すカスタム アクションを作成する必要があります。

タスク ***IIS メタベースおよび IIS 6 構成との互換性機能と IIS 6 以前をチェックするカスタム アクションを追加するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI の場合) または **[カスタム アクション]** (DIM、マージ モジュール プロジェクトの場合) をクリックします。
2. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが追加されます。
3. アクションの名前を、たとえば **DetectIIS6Interfaces** に変更します。
4. 右側のペインでカスタム アクションの設定を構成します:
   - **"DLL ファイル名"** 設定で、DetectIIS6Compat.dll ファイルへのパスを指定します。通常のパスは以下のとおりです:

     <ISProductFolder>¥Samples¥WindowsInstaller¥Detect IIS6 Compatibility¥DetectIIS6Compat.dll
   - "**関数名**" 設定で、次の名前を入力します:

     DetectIIS6Interfaces
   - **"スクリプト内実行"** 設定で、**[即時実行]** を選択します。
   - 1 つまたは複数のシーケンス設定で適切なオプションを選択して、必要に応じて、カスタム アクションをスケジュールします。

IIS 6 メタベースおよび IIS 6 構成の互換性機能がインストールされているか、ターゲットシステムに IIS 6 以前が搭載されている場合、Windows Installer プロパティ ISIISMETABASECOMPATPRESENT の値は 1 に設定されます。ターゲット システムに IIS 7 以降が搭載されていて、互換性機能がインストールされていない場合、プロパティは設定されません。

条件ステートメントで ISIISMETABASECOMPATPRESENT を使用できます。たとえば、IIS 6 メタベースおよび IIS 6 構成の互換性機能または IIS 6 以前が搭載されていないシステム上で製品の使用を不可能にする場合、[一般情報] ビューの "インストール条件" 設定に次の条件ステートメントを入力します:

ISIISMETABASECOMPATPRESENT

別の方法として、エラー カスタム アクションを作成することもできます。実行時、そのカスタム アクションの条件が True の場合、エラーが表示されます。したがって、IIS 6 メタベースおよび IIS 6 構成の互換性機能または IIS 6 以前が搭載されていないシステム上で製品の使用を不可能にする場合、エラー カスタム アクションに以下の条件を使用します:

Not ISIISMETABASECOMPATPRESENT

IIS 6 メタベースおよび IIS 6 構成の互換性機能または IIS 以前が搭載されているシステム上にのみコンポーネントをインストールする場合、[コンポーネント] ビューの "条件" 設定に次の条件ステートメントを入力します:

ISIISMETABASECOMPATPRESENT=1

IIS の詳細設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

メモ: *詳細 IIS 設定は IIS 6 以前に適用します。IIS 7 は、これらの設定を無視します。*

[IIS 構成] ビュー内の [Web サイト] エクスプローラーで Web サイト、アプリケーション、または仮想ディレクトリを選択したときに表示される右側のペインには、IIS の最も一般的な設定が含まれています。右側ペインの下にある [詳細] 領域では、その他の領域では表示されない IIS 設定を構成できます。

タスク ***[IIS 構成] ビューに表示されていない設定を構成するには、以下の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、詳細設定を構成する Web サイト、アプリケーション、または仮想ディレクトリを選択します。
3. **"その他の IIS プロパティ"** 設定で、省略記号ボタン (...) をクリックします。**[その他の IIS プロパティ]** ダイアログ ボックスが開きます。
4. プロパティー覧で、値を変更するプロパティを選択して、**[値の変更]** ボタンをクリックします。**[IIS メタデータ値の編集]** ダイアログ ボックスが開きます。
5. **[値]** ボックスで、新しい値を指定します。

詳細プロパティのデフォルト値を変更すると、そのプロパティと、指定した値、および追加の必須情報が ISIISProperty テーブルに追加されます。詳細については、「ISIISProperty テーブル」を参照してください。

Web サイト、アプリケーション、または仮想ディレクトリのカスタム エラー メッセージを定義する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

エンドユーザーが Web サイトに接続しようとして、HTTP (Hypertext Transfer Protocol) エラーが発生した場合、エンド ユーザーのブラウザーはエラーを説明するデフォルトのメッセージを表示します。HTTP エラー コードをファイルまたは URL (Uniform Resource Locator) へマッピングすることで、インストールが IIS を構成して、デフォルトのエラー メッセージの代わりにカスタム エラー メッセージを表示するよう設定することができます。

タスク

Web サイト、アプリケーション、または仮想ディレクトリのカスタムエラーメッセージを構成するには、以下の手順を実行します。

1. カスタムエラーメッセージを含むファイルを作成し、インストールに追加します。
2. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
3. HTTP エラーメッセージをカスタマイズする Web サイト、アプリケーション、または仮想ディレクトリを選択します。Web サイト、アプリケーション、または仮想ディレクトリの設定が、右側に表示されます。
4. **"カスタム エラー"** 設定で、省略記号ボタン (...) をクリックします。**[カスタム エラー]** ダイアログ ボックスが開きます。
5. 変更する HTTP エラーコードを選択し、**[プロパティの編集]** ボタンをクリックします。**[新しいプロパティの追加]** ダイアログ ボックスが開きます。
6. ファイルへエラーコードをマップするには、以下の手順を実行します。
   a. **[メッセージの種類]** リストから **[ファイル]** を選択します。
   b. [**ファイル**] ボックスで、インストール内のカスタム エラー メッセージをポイントするパスおよびファイル名を入力するか、[参照] ボタンを押してファイルを指定します。

   URL へエラー コードをマップするには、以下の手順を実行します。
   a. [**メッセージの種類**] 一覧から URL を選択します。
   b. **URL** ボックスで、カスタム エラー メッセージへの URL を入力します。

Windows Installer のプロパティを使用して、IIS 設定を動的に変更する

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*

- *InstallScript MSI*

基本の MSI および InstallScript MSI プロジェクトでは、Windows Installer のプロパティを使用して、実行時に IIS 設定を動的に構成することができます。これにより、エンドユーザーがターゲット マシンにインストールしている仮想ディレクトリの名前、TCP ポート、サイト番号、または Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張の他の IIS 設定を指定できるようにすることができます。

Windows Installer は MsiFormatRecord を使って、実行時にこのプロパティを解決します。このプロパティとその値は、アンインストールと修復の時に使用できるように、インストールによって次のレジストリ キーに書き込まれます:

HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Uninstall¥InstallShield Uninstall Information¥*[ProductCode]*

したがって、Web サイトがエンド ユーザーによって指定されたサイト番号にインストールされた場合、Web サイトとそのアプリケーション、およびその仮想ディレクトリは、このサイト番号から正常にアンインストールされるようになります。

ヒント: *IIS データをレジストリに格納しないほうが良い場合、プロジェクトで* IS_IIS_DO_NOT_USE_REG *プロパティを設定します。*

[IIS 構成] ビューのパスワードにプロパティを使用すると、パスワードは暗号化されてレジストリに格納されます。

例

以下は、エンドユーザーがインストール中に、Web サイトに匿名アクセスするときに使用されるユーザー名を指定できるようにする手順です。文字列の入力が許可されている [IIS 構成] ビューあるすべての IIS 設定では、ハードコード化されている値にプロパティを代入することができます。このビューでプロパティを指定するとき、プロパティは角かっこで囲み、プロパティ名はすべて大文字にする必要があります (例、**[MYPROPERTY]**)。

手順のステップ 5 はプロジェクトの種類によって若干異なります。これは、基本の MSI インストールのユーザー インターフェイスは Windows Installer が制御し、InstallScript MSI インストールのユーザー インターフェイスは InstallScript エンジンが制御するためです。

タスク ***エンドユーザーがユーザー名を指定できるようにするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、エンドユーザーが匿名アクセスにユーザー名を指定できる Web サイトをクリックします。
3. **"匿名アクセスを有効にする"** 設定で、**[はい]** を選択します。
4. **"匿名ユーザー名"** 設定で、次のように入力します:

   [MYPROPERTY]
5. ダイアログでプロパティを使用します。この部分の手続きは、使用しているプロジェクトの種類によって異なります。
   - 基本の MSI プロジェクトの場合:
     a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。

b. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** フォルダーを展開して、User Name コントロールを含めるダイアログの下にある言語をクリックします。代わりに、新しいダイアログを追加することもできます。

c. **Edit Field** コントロールをダイアログに追加し、**Property** プロパティを次にように設定します:

MYPROPERTY

- InstallScript MSI プロジェクトの場合:

a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。

b. User Name コントロールを含めるダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、Windows Installer API 関数 **MsiSetProperty** の呼び出しを追加します。たとえば、ユーザー名をエンド ユーザーが CustomerInformation ダイアログで指定する名前にする場合、次のコードに示されている **MsiSetProperty** の呼び出しを追加します:

```
Dlg_SdCustomerInformation:
    nResult = SdCustomerInformation(szTitle, svName, svCompany, nUser);
    MsiSetProperty(ISMSI_HANDLE, "MYPROPERTY", svName);
    if (nResult = BACK) goto Dlg_SdWelcome;
```

6. リリースをビルドします。

MsiFormatRecord (Windows Installer ヘルプ ライブラリ)

InstallScript テキスト置換を使用して、IIS 設定を動的に変更する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、テキスト置換文字列変数を使用して、実行時に IIS 設定を動的に構成することができます。これにより、エンドユーザーがターゲット マシンにインストールしている仮想ディレクトリの名前、TCP ポート、サイト番号、または Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張の他の IIS 設定を指定できるようにすることができます。InstallScript ランタイム コードは **TextSubSubstitute** 関数を使用して、文字列変数を適切な値で置き換えます。

例

以下は、エンドユーザーがインストール中に、Web サイトに匿名アクセスするときに使用されるユーザー名を指定できるようにする手順です。文字列の入力が許可されている [IIS 構成] ビューあるすべての IIS 設定では、ハードコード化されている値にテキスト置換文字列変数を代入することができます。このビューで指定する文字列変数は山かっこで囲む必要があります; 例、**<MYPROPERTY>**。指定する文字列変数は大文字と小文字を区別します。

タスク ***エンドユーザーがユーザー名を指定できるようにするには、以下の手順に従います。***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、エンドユーザーが匿名アクセスにユーザー名を指定できる Web サイトをクリックします。
3. **"匿名アクセスを有効にする"** 設定で、**[はい]** を選択します。
4. **"匿名ユーザー名"** 設定で、次のように入力します:

<MYPROPERTY>

5. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
6. User Name コントロールを含むダイアログの OnFirstUIBefore イベントでダイアログ コードを見つけ、InstallScript 関数 **TextSubSetValue** の呼び出しを追加します。たとえば、ユーザー名をエンド ユーザーが CustomerInformation ダイアログで指定する名前にする場合、次のコードに示されている **TextSubSetValue** の呼び出しを追加します:

```
Dlg_SdRegisterUser:
    szMsg = "";
    szTitle = "";
    //{{IS_SCRIPT_TAG(Dlg_SdRegisterUser)
    nResult = SdRegisterUser( szTitle, szMsg, szName, szCompany );
    TextSubSetValue("<MYPROPERTY>", szName, TRUE);
    //}}IS_SCRIPT_TAG(Dlg_SdRegisterUser)
    if (nResult = BACK) goto Dlg_SdLicense2;
```

7. リリースをビルドします。

ターゲット マシン上で Web サービスを配布する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ターゲット システムへ Web サービスを配布するには、Web サービス専用のファイルを特定の場所にコピーし、そのフォルダーに仮想ディレクトリ名を割り当てる必要があります。これにより、HTTP を通して Web サービスにアクセスできるようになります。

ヒント: *仮想ディレクトリをプロジェクトに追加する方法については、「Web サイトの作成とアプリケーションまたは仮想ディレクトリの追加」をご覧ください。*

タスク ***ターゲット マシンへ Web サービスを配布するには、以下の手順に従います。***

1. **[アプリケーション データ]** の下にあるビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. ターゲット システムにファイルをインストールするためのフォルダー (ターゲット ディレクトリ) を選択します。そのフォルダーにファイルを追加します。
3. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
4. **[Web サイト]** エクスプローラーで、Web サイトに関連付けられている仮想ディレクトリを選択します。

5. **"コンテンツ ソース パス (ローカルまたは UNC)"** 設定で省略記号ボタン (...) をクリックします。**[ディレクトリの参照]** ダイアログ ボックスが開きます。**[ファイルとフォルダー]** ビューで追加した新しいファイルを含むディレクトリと同じターゲット ディレクトリを入力します。

インストール時、ファイルはターゲット ディレクトリ フォルダーへコピーされます。さらに IIS が存在する場合、ターゲット システム上のフォルダーに仮想ディレクトリが構成されます。

Web アプリケーションのフォーム認証を有効化する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Web サイトの **[IIS 構成]** ビューで **[セキュリティ > 認証済みアクセス]** セクションの下にある **"フォーム認証"** 設定を使って、Web アプリケーションのフォーム認証を設定します。**[フォーム認証]** オプションを **[はい]** に設定してフォーム認証を有効化します。

ASP.NET フォーム ベースの認証は、多くの要求を受け取るパブリック Web サーバー上にあるサイトまたはアプリケーションに適しています。この認証モードでは、オペレーティング システムが提供している認証方法に頼ることなく、アプリケーション レベルでのクライアント登録および認証を管理することができます。

重要: *フォーム認証は、Web サーバーにユーザー名とパスワードをテキスト形式で送信します。ホームページ以外のアプリケーション内のすべてのページ、およびログオン ページには、Secure Sockets Layer (SSL) 暗号化を使用してください。*

タスク ***Web サイトのフォーム認証を有効化するには、次の手順に従います:***

1. **[サーバー構成]** の下のビュー リストにある **[IIS 構成]** をクリックします。
2. **[Web サイト]** エクスプローラーで、Web サイトを選択します。
3. **[セキュリティ > 認証アクセス]** の下にある **"フォーム認証"** 設定を **[はい]** に設定します。

IISROOTFOLDER サポートの追加

IISROOTFOLDER は、ターゲット システム上にある Web サーバーのルート ディレクトリを判断するのに利用される InstallShield ディレクトリ変数です。インストール プロジェクトで IIS 機能を使用していて、Web サイトが既に追加されている場合 IISROOTFOLDER は自動的に追加されます。

メモ: *[ファイルとフォルダー] ビューで IISROOTFOLDER ディレクトリに追加されたファイルはすべて、ターゲット マシン上の Web サーバーのルート ディレクトリにインストールされます。IIS がインストールされていない場合、ファイルはルート フォルダーにコピーされます。*

IIS_WEBSITE_NAME プロパティ

IIS_WEBSITE_NAME プロパティは現在使用されていません。これらのプロパティが以前のプロジェクト バージョンから存在している場合、アップグレーダはこれらを自動的に処理します。アップグレーダは Web サイトを作成して、サイト番号フィールドを [IIS_WEBSITE_NAME] に設定します。新しい Web サイトには、任意のプロパティまたはハードコード化された番号を使用できます。

IIS_PORT_NUMBER プロパティ

IIS_PORT_NUMBER プロパティは現在使用されていません。これらのプロパティが以前のプロジェクト バージョンから存在している場合、アップグレーダはこれらを自動的に処理します。アップグレーダは Web サイトを作成して、ポート番号フィールドを [IIS_PORT_NUMBER] に設定します。新しい Web サイトを作成すると、任意のプロパティまたはハードコード化された番号を使用できます。

メンテナンスおよびアンインストールのための インストールを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、エンドユーザーがインストールを再実行して、プログラム機能の変更したり、アプリケーションの再インストールまたは削除をすることができます。エンドユーザーがコントロール パネルの [プログラムの追加と削除] でアプリケーションを選択すると、インストールによって以下を実行できるダイアログ ボックスが表示されます。

- 以前にインストールされなかった機能の個別インストールおよびアンインストール。
- 最初のインストールで選択された設定でアプリケーションを再インストール
- アプリケーションのアンインストール

アプリケーションの変更、修正、アンインストールを行う場合、オペレーティング システムにがアプリケーションの存在を認知している必要があります。このため、インストールはオペレーティング システムにアプリケーションを登録し、従って簡単にメンテナンスまたはアンインストールができます。[一般情報] ビューで必要な情報を入力します。

すべての InstallShield インストールでは、メンテナンス (変更および修正) はデフォルトで自動的に処理されます。

Windows Installer ベースのインストールの場合、アンインストールは自動的に処理されますが、唯一の例外は、アンインストール中それ自身で効果を取り消すか、または、アンインストール中のみ実行される別のカスタム アクションによってその効果を取り消させる必要があるカスタム アクションがあるということです。

InstallScript または InstallScript MSI インストールでは、数多くのスクリプト関数のアクションは自動アンインストールによってログされますが、ログされないスクリプト関数アクションは、アンインストールのスクリプトで対応する必要があります。

このセクションには、次のトピックが含まれています:

- アンインストールのための InstallScript インストールを作成
- アンインストール時またはインストール時のみにスクリプト コードを実行する
- アンインストール用にログされた InstallScript 関数
- メンテナンス/アンインストール機能
- 製品が作成したファイルの削除
- 製品が作成したレジストリデータの削除

アンインストールのための InstallScript インストールを作成

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*

アンインストール機能の構成は、その結果生じる操作の複雑性を考えると、比較的単純です。**CreateInstallationInfo** (または **InstallationInfo**) および **MaintenanceStart** (または **DeinstallStart**) の関数を呼び出します。イベント指向のスクリプトでは、**CreateInstallationInfo** および **MaintenanceStart** がデフォルト **OnMoveData** イベント ハンドラーコードで呼び出されます。

他にも、アンインストール機能の構成に重要な役割を果たす関数があります。たとえば、**RegDBCreateKeyEx** 関数を使って作成されたすべてのキーは、ロギングが無効ではない限り、アンインストール用に記録されます。また、**Disable** 関数を呼んでロギングを無効にする場合、必ず **Enable** を呼んで、アンインストール用に記録が必要なその後に続くアクションのために再度ロギングを有効化するようにしてください。

MaintenanceStart (または **DeinstallStart**) 必要なレジストリキーおよびそのための必要な値を作成します。このキーの下に追加のレジストリキーを作成する場合、 RegDBSetItem を呼ぶか、または、カスタム値の場合、次のようなコードを使用してください。

```
RegDBSetDefaultRoot( HKEY_USER_SELECTABLE );
RegDBSetKeyValueEx( "Software¥¥Microsoft¥¥Windows¥¥Current Version¥¥Uninstall¥¥" + INSTANCE_GUID, "カスタム値", nType, szValue, nSize);
```

*メモ: アンインストールは、初回インストールが **FeatureTransferData** または **FeatureMoveData** を呼び時のみ実行可能です。(イベント指向のスクリプトでは、**FeatureTransferData** 関数が自動的に呼び出されます)。インストールが他の手法 (例、**XCopyFile** 等) によってファイルをインストールするときに、エンドユーザーにアンインストールを実行できるようにする場合、すべてのコンポーネントを選択解除し、**FeatureTransferData** または **FeatureMoveData** を呼び出します。この呼び出しが実行されている間、**SdShowMsg** を呼んでメッセージを表示することもできます。*

アンインストール時に実行するスクリプト コードを含める

*メモ: **DeinstallStart** を呼んでアンインストールを有効にすると、インストール スクリプトはアンインストール中に実行されません。アンインストール時にスクリプト コードを実行するには、代わりに **MaintenanceStart** を呼びます。(イベントベースのインストールでは、**MaintenanceStart** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。)*

イベントベースのスクリプト

イベント指向のスクリプトでは、次のイベント ハンドラー関数がアンインストール中に自動的に呼び出されます。

- **OnBegin**
- **OnMaintUIBefore**

- **OnMoving**
- <機能名>**Uninstalling**（各インストール済みの機能に対して）
- <機能名>**Uninstalled**（各インストール済みの機能に対して）
- **OnMoved**
- **OnMaintUIAfter**
- **OnEnd**

オーバーライド可能なデフォルトコードを持つこれらのイベント ハンドラー関数のコードはアンインストール中に実行されます。アンインストール、または、インストール中のみ特定のイベント ハンドラーを実行する手順については、アンインストール時またはインストール時のみにスクリプト コードを実行するを参照してください。

手続き型スクリプト

手続き型のスクリプト(プログラム ブロックを使用したスクリプト)で、**DeinstallStart** を使わずに **MaintenanceStart** を使ってアンインストールを有効にすると、スクリプトはアンインストール中に実行されます。アンインストール、または、インストール中のみ特定のスクリプトを実行する手順については、アンインストール時またはインストール時のみにスクリプト コードを実行するを参照してください。

アンインストール時またはインストール時のみにスクリプト コードを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

DeinstallStart ではなく **MaintenanceStart** を呼び出してアンインストールを可能にすると、アンインストールおよびメンテナンス インストール中にインストールスクリプトが実行されます。(イベントベースのスクリプトでは、**MaintenanceStart** がデフォルトの **OnMoveData** イベント ハンドラーコードで呼び出されます。)コードをアンインストール中、またはメンテナンスインストール中のみ実行するには、コードを次のような if-then ステートメントで囲みます。

```
if MAINTENANCE then
    /* このコードは、
    アンインストール または メンテナンス インストール */
endif;
```

コードを最初のインストール中のみ実行するには、コードを次のような if-then ステートメントで囲みます。

```
if !MAINTENANCE then
    /* このコードは、
    初回のインストール */
endif;
```

コードを上のどちらの if-then ステートメントで囲まない場合、アンインストール、メンテナンスインストール、および初回のインストール中に実行されます。ただし、イベント指向のスクリプトでは、以下のような例外があります。

- 次のようなイベント ハンドラーのコードは、アンインストール中およびメンテナンスインストール中のみ実行されます。

  OnMaintUIBefore

  OnMaintUIAfter

- 次のようなイベント ハンドラーのコードは、最初のインストール中のみ実行されます。

  OnCCPSearch

  OnAppSearch

  OnFirstUIBefore

  OnFirstUIAfter

アンインストール用にログされた InstallScript 関数

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI (InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合)*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

一般的に、ログ記録が有効な状態で作成または置換されたすべてのファイル、フォルダー、レジストリ エントリ、.ini ファイル エントリ、ショートカット フォルダー、およびサービスは、アンインストール用にログ記録されます。また、InstallScript インストールおよび InstallScript MSI インストール中にターゲット システムで変更を行う InstallScript 関数もアンインストール用にログ記録されます。デフォルトでインストールがアンインストール用にログ記録を行う一般的な関数の例は以下の通りです:

- AddFolderIcon
- AddProfString
- CopyFile
- CreateInstallationInfo
- CreateProgramFolder
- FeatureMoveData
- FeatureTransferData
- RegDBSetAppInfo
- RegDBSetItem
- ReplaceProfString
- ServiceAddService

- ServiceStartService
- VerSearchAndUpdateFile
- VerUpdateFile
- WriteProfString
- XCopyFile

ログ記録は、InstallScript コード内でDisable (LOGGING); および Enable (LOGGING); 関数を呼び出して有英語効または無効にできます。詳細については、「Disable」および「 Enable」を参照してください。

メンテナンス/アンインストール機能

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

メンテナンス/アンインストール機能はメンテナンス設定とアンインストールに必要なファイルをインストールします。ファイルのターゲット先はシステム変数 *DISK1TARGET* の値によって指定されます。この機能は、 .cab に自動的に配置され、InstallShield インターフェイスでは表示されません。

実行時にメンテナンス/インストール機能は、全セットアップの種類用に自動選択されます。**FeatureUpdate** を呼び出すと、この機能は選択解除されます。メンテナンス/インストール機能は、**FeatureSelectItem** を呼び出すことで選択/選択解除され、システム変数 *DISK1COMPONENT* を関数の第 2 引数として渡します。たとえば、次のコードは選択解除する必要があります。

```
FeatureSelectItem( MEDIA, DISK1COMPONENT, FALSE );
```

製品が作成したファイルの削除

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

デフォルトでは、インストール後の製品によって作成されたファイルは、ユーザーのデータであると見なされるので、アプリケーションをアンインストールする際にも削除されません。

基本の MSI プロジェクト

RemoveFile テーブルにレコードを作成して (ダイレクト エディターを使用)、削除するファイルを指定することができます。たとえば、アプリケーションが *INSTALLDIR* の指定先に Product.ini というファイルを作り、メインの実行可能プログラムを含むコンポーネントの削除時に、このファイルも同時に削除させる場合は、RemoveFile テーブルに次のレコードを追加します。

テーブル 1 ・ RemoveFile テーブルに追加する追加

FileKey	remove_file
Component_	ProgramFiles
FileName	Product.ini
DirProperty	INSTALLDIR
InstallMode	2

InstallScript および InstallScript MSI プロジェクト

スクリプトで、アンインストールのイベント ハンドラー関数の中から DeleteFile および DeleteDir 関数を呼び出すことで、指定したファイルやディレクトリを削除できます。

製品が作成したレジストリデータの削除

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

デフォルトでは、製品のアンインストーラーは、インストール プログラムが作成したデータのみを削除します。

基本の MSI プロジェクト

レジストリエクスプローラーは特別なアンインストールフラグを使って、アンインストール時に削除するレジストリデータを管理します。特に、レジストリキーで “-” フラグ (キー全体をアンインストール) を使用すると、キーおよびそのすべての値とサブキーがアンインストール時に削除されます。

***メモ・** RemoveRegistry テーブル (ダイレクト エディターで表示) は、製品のインストール時にのみ削除するデータを指定します。詳細については、Windows Installer ヘルプライブラリを参照してください。*

InstallScript MSI プロジェクト

スクリプトで、アンインストールのイベント ハンドラー関数の中から RegDBDeleteValue および RegDBDeleteKey 関数を呼び出すことで、レジストリ値やキーを削除できます。

トランザクション処理を使って複数パッケージ インストールを構成する

プロジェクト: *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

ドキュメントのこのセクションでは、InstallShield で連鎖されたパッケージとしてインストールに含める .msi パッケージを指定する方法について説明します。また、連鎖するパッケージに渡すプロパティなどの設定の構成方法についても説明します。このセクションには、次のトピックが含まれています:

- トランザクション処理を使った複数パッケージ インストールについての概要
- 新しい連鎖 .msi パッケージをプロジェクトに追加する
- リリース フラグを連鎖 .msi パッケージに割り当てる
- プロジェクトから連鎖 .msi パッケージを削除する

メモ: *Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。以前のバージョンは、連鎖 .msi パッケージのいずれも起動しません。*

インストールで Windows Installer 4.5 をインストールする方法は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

トランザクション処理を使った複数パッケージ インストールについての概要

プロジェクト: *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*

インストールに 1 つまたは複数の .msi パッケージが連鎖するパッケージとして追加されていて、エンド ユーザーが Windows Installer 4.5 を搭載したシステム上でこのインストールを実行すると、Windows Installer はすべてのパッケージを単一のトランザクションでインストールすることができます。トランザクションに含まれる 1 つまたは

複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

トランザクション処理を行うとき、Windows Installer は .msi パッケージから様々なスクリプトの状態をバッチ処理します。たとえば、Windows Installer が [実行] シーケンス中に InstallInitialize アクションを開始するとき、その時点で Windows Installer は全てのパッケージの即時アクションを実行し、また全ての遅延アクションをインストール スクリプトに配置します。次に、Windows Installer は全てのパッケージのインストール スクリプトに含まれる遅延アクション (コミットまたはロールバック アクションを除く) を実行します。さらに、Windows Installer はロールバック アクションをロールバック スクリプトにコピーします。この時点まで各パッケージの製品が正しくインストールされると、Windows Installer は各パッケージのすべてのコミットアクションを実行します。そうでない場合、Windows Installer はロールバック スクリプトを実行して、ターゲット システムを元の状態に復元します。

この機能についての詳細は、Windows Installer ヘルプ ライブラリの「Multiple-Package Installations」を参照してください。

メモ: *Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。以前のバージョンは、連鎖 .msi パッケージのいずれも起動しません。*

インストールで Windows Installer 4.5 をインストールする方法は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

新しい連鎖 .msi パッケージをプロジェクトに追加する

プロジェクト: *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield では、ビルド済みの Windows Installer パッケージ (.msi ファイル) を連鎖 .msi パッケージとしてプロジェクトに追加することができます。Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

[リリース] ビューの [連鎖 .msi パッケージ] 領域で、メイン インストールに連鎖させる 1 つまたは複数の .msi パッケージをプロジェクトに追加できます。

連鎖 .msi パッケージは、インストール中にマナー アップグレードをサポートします。

メモ: *Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。以前のバージョンは、連鎖 .msi パッケージのいずれも起動しません。*

インストールで Windows Installer 4.5 をインストールする方法は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

タスク ***.msi パッケージを連鎖するパッケージとして追加するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[連鎖 .msi パッケージ]** エクスプローラーを右クリックしてから、**[新しい連鎖パッケージ]** をクリックします。InstallShield は、デフォルト名を持つ新しい置換アイテムを追加します。
3. 新しい名前を入力するか、または名前を後で右クリックしてから **[名前の変更]** を選択して新しい名前を付けます。

   名前は実行時には表示されません。これはメイン インストールに含まれる異なる連鎖 .msi パッケージ間でパッケージを区別するのに内部で使用される名前です。
4. 新しい連鎖されたパッケージ アイテムを選択します。InstallShield の右側に設定が表示されます。
5. **"インストール (実行時のパス)"** 設定で、**[参照]** ボタンをクリックします。**[ファイルの参照]** ダイアログ ボックスが開きます。
6. 追加する Windows Installer パッケージ (.msi ファイル) を選択してから、**[開く]** をクリックします。選択するパッケージは、**MsiInstallProduct** を使ったインストールが可能な .msi パッケージでなくてはなりません。InstallScript MSI パッケージは選択できませんので、ご注意ください。

InstallShield は、.msi パッケージ (および、.msi パッケージが圧縮されていない場合は非圧縮ファイル) を製品のメイン .msi パッケージにストリームするかどうかを問い合わせるプロンプトを表示します。

- ファイルのストリームを指定した場合、InstallShield は .msi パッケージの名前を **"インストール (実行時のパス)"** 設定に追加します。また、自動的にファイルの名前 (圧縮された .msi パッケージの場合) またはエントリ *.*(非圧縮の .msi パッケージの場合) が **[ストリームされたファイル]** ボックスに追加されます。ワイルドカード エントリ *.* は、.msi パッケージと同じフォルダーにある全てのファイルとともに .msi パッケージにストリームされます。
- ファイルのストリームを行わないことを指定した場合、InstallShield は次のパスを **"インストール (実行時のパス)"** 設定に追加します:

  [SourceDir]FileName.msi

  このパスは、必要に応じて変更が可能です。たとえば次のようなパスを使用し、また **[インストール後にストリームされたファイルを削除する]** チェック ボックスをクリアしたい場合があります:

  [LocalAppDataFolder]{ProductCodeGUID}¥FileName.msi

  この例では、.msi パッケージはローカル システムにキャッシュされ、それはメンテナンスで使用することができます。

  ヒント・*Windows Installer では、.msi パッケージのファイル サイズに制限があるため、ファイルを .msi パッケージにストリームしなくない場合があります。*

7. 必要に応じて、連鎖 .msi パッケージの設定を構成します。詳細については、「連鎖 .msi パッケージの設定」を参照してください。

メイン インストールにいくつかの .msi パッケージが含まれている場合、InstallShield はメイン インストールの ISChainPackage テーブルの Order 列を使って、実行時に連鎖するパッケージを起動する順番を決定します。アンインストールは Order 番号が高い方から低い方へ逆の順番で、また全てのインストールは Order 番号が低い方から高い方へ順番に処理されます。

メモ: *[リリース] ビューの [連鎖 .msi パッケージ] 領域で .msi パッケージを指定した場合、メイン インストールのリリースをビルドするときに InstallShield は連鎖パッケージをビルドしません。*

リリース フラグを連鎖 .msi パッケージに割り当てる

プロジェクト: *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

特定のビルドから除外する連鎖 .msi パッケージに、1 つまたは複数のリリース フラグを割り当てることができます。たとえば、連鎖 .msi パッケージを必要とする特別なアドオンを含んだ製品の特別なエディションにのみ含める連鎖 .msi パッケージがある場合、その連鎖パッケージにフラグを付けることで、必要な場合にのみそれがビルドに含まれるようにします。

タスク

インストール プロジェクトに追加した連鎖 .msi パッケージにリリース フラグを追加するには、次の手順に従います。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[連鎖 .msi パッケージ]** エクスプローラーで、リリース フラグを含めるパッケージを選択します。
3. **"リリース フラグ"** 設定で、文字列を入力します。文字列には、文字または数字を使用できます。1 つの連鎖パッケージに複数のフラグを付けるには、コンマを使ってフラグを区切ります。

リリース フラグに基づいて連鎖 .msi パッケージをフィルターする方法についての詳細は「リリース フラグ」を参照してください。

プロジェクトから連鎖 .msi パッケージを削除する

プロジェクト: *次のプロジェクト タイプで、トランザクション処理を使った複数パッケージのインストールがサポートされています:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

タスク　　プロジェクトから連鎖 .msi パッケージを削除するには、以下の手順に従います。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[連鎖 .msi パッケージ]** エクスプローラーで、プロジェクトから削除する連鎖 .msi パッケージを右クリックしてから、**[削除]** を選択します。

InstallShield がプロジェクトから連鎖 .msi パッケージを削除します。InstallShield は、.msi パッケージ、またはそれに属するファイルをファイル システムからは削除しませんので、ご注意ください。

インストールのビルド、テスト、および配布

インストール プロジェクトの機能、コンポーネント、ファイル、ショートカット、レジストリ エントリ、エンド ユーザー ダイアログ、およびその他の要素を構成が完了すると、インストールのリリースを作成しビルドできます。InstallShield では、1 つのプロジェクトに対して複数リリースの作成が可能で、異なるメディア タイプおよび異なる要件の組み合わせにしたがって各リリースを構成することができます。リリースをビルドすると、インストールのコンテンツがパッケージされ、配布メディアにコピーして、必要に応じて配布または配置することができるディスク イメージが作成されます。

テストは安定したインストールを作成するために最も重要です。InstallShield では、リリースのエンドユーザー インターフェイスのみを部分的にテスト実行することが可能ですInstallShield のボタンをクリックするだけで、インストールを実行することもできます。この方法でインストールを実行した場合、エンド ユーザーのマシン上とまったく同様にインストールが実行します。ファイルはすべて転送され、ショートカットおよびレジストリ エントリが作成され、ユーザー インターフェイスが表示されます。

製品をリリースする前に、インストール プロジェクトを検証した方がよい場合があります。Windows Installer プロジェクトを検証するには、一連の内部整合性評価プログラム (ICE) のルールをインストール プロジェクトに適用します。これらの ICE はマイクロソフトが作成したもので、アクションが正常に実行するために必要な有効データベースがインストール パッケージに含まれていることを確認するのに利用します。

InstallShield のデバッグ ツールを利用して、InstallScript スクリプトまたは Windows Installer リリースをデバッグして問題の原因を追究することができます。スクリプトをデバッグする際、スクリプトをステートメント毎に実行し、実行ポイントがスクリプトの中を移動するのを確認しながらコントロールの流れを追います。また、スクリプト実行中どの時点でも、スクリプト中の変数の値を監視することもできます。Windows Installer リリースをデバッグするとき、InstallShield は実行を一時停止するブレーク ポイントに到達するまで各アクションおよびダイアログ ボックスを実行します。この時点で、プロパティを表示して設定することができます。

このセクションには、次のトピックが含まれています:

- リリースの作成とビルド
- インストールのテストと実行
- プロジェクトの検証
- インストールまたはアンインストールがターゲット システムを再開するタイミングについて
- Windows Installer ベースのインストールのデバッグ
- インストールを複数のディスクまたは CD に分割する
- セットアップランチャーの作成
- インストールを配布する
- 64 ビット セットアップ ランチャーのビルド
- Windows アプリ パッケージの配布

リリースの作成とビルド

InstallShield でプロジェクトのデザインが完了すると、構成およびビルドしてエンドユーザーへ配布できるリリースを作成できます。リリースは、リリース ウィザード、または [リリース] ビューに表示される設定で指定されたオプションに基づいてビルドされます。

製品構成の使用

プロジェクト: *以下は、製品構成をサポートするプロジェクトの種類です:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

ビルドした各リリースは、製品構成のメンバーです。製品構成により、製品名、製品コード、パッケージコードなど、類似したプロパティを共有するリリースを一緒にグループ化することができます。

製品の構成により、InstallShield の外部でフォルダー構造を設定しなくてもビルドの履歴を保持することができます。たとえば開発中の製品が、同じバージョンでも複数のリリースになるような場合がありえます。製品を再ビルドするごとに各リリースを上書きすることを、避けたい場合もあります。製品構成により、過去のビルド履歴の保持と新しいリリースの組み込みを同時に実現できます。

以前は、このような履歴を InstallShield の外部に保管しなければなりませんでした。その場合、ドライブ上にディレクトリ構造を作成して各フォルダーに手作業でラベルを付け、そのディレクトリにリリースをコピーしていました。製品構成により、これらの作業は InstallShield で自動的に行われます。

タスク

製品構成を作成するには、以下の手順に従います。

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーを右クリックし、**[新しい製品構成]** をクリックします。

InstallShield が [リリース] エクスプローラーに新しい製品構成を追加します。[全般] タブで製品構成の設定が一覧表示されます。

各製品構成の設定については、製品構成の [全般] タブ を参照してください。

ヒント: *リリース ウィザード を使って製品構成を作成することもできます。(-a パラメーターを使って) コマンドラインから既存の製品構成を指定することができますが、コマンドラインを通してリリースの設定を定義することはできません。*

ビルドに適切なアーキテクチャ検証の種類を選択する

プロジェクト: *以下は、製品構成をサポートするプロジェクトの種類です:*

- *基本の MSI*
- *マージ モジュール*

WOW64 (32-bit Windows-on-Windows) サポートを含まない x64 Windows ベースのシステムがより一般的なので、InstallShield でリリースをビルドするときにアーキテクチャの検証を実行することが推奨されます。アーキテクチャの検証を使って、インストールがターゲット システムのアーキテクチャと一致しない製品ファイルをインストールしようとしたり、ランタイム バイナリを使用したりする問題を引き起こす可能性のある状況を検出することができます。

たとえば、エンド ユーザーが WOW64 サポートが搭載されていない 64 ビット Windows Server Core システムでインストールを実行することが可能な場合、アーキテクチャの検証を使って、インストールに含まれるすべての x86 ビット の製品ファイル、または x86 カスタム アクション ファイルを識別できます。x86 バイナリを WOW64 サポートを搭載しない x64 ターゲット システムにロードすることはできません。

また、単一のプロジェクトに x64 と x86 製品ファイル (または x64 と x86 カスタム アクション) が混在し、個別の x86 .msi パッケージと x64 .msi パッケージを生成する場合、アーキテクチャの検証を使って、x86 リリースに含まれる任意の x64 製品またはカスタム アクション ファイルを識別することができます。これらのファイルを x86 ターゲット システムでロードすることはできません。

アーキテクチャの検証を使って、ビルトイン InstallShield カスタム アクション DLL の x86 バージョンのみを含む x86 専用 .msi パッケージ、またはビルトイン InstallShield カスタム アクション DLL の x64 バージョンのみを含む x64 専用 .msi パッケージを生成することができます。この機能を有効化するため、InstallShield では 2 つの定義済みパス変数 (ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder) が使用されます。デフォルトで、これらのパス変数の値は、InstallShield カスタム アクション DLL の x86 バージョンを含むサブフォルダーです。サブフォルダーは、*InstallShield Program Files Folder*¥Redist にあります。InstallShield は、必要な場合、ビルド時にこれらのパス変数の値を変更して、InstallShield カスタム アクション DLLの (x86 バージョンの代わりに) x64 バージョンをリリースに含みます。これらのファイルは *InstallShield Program Files フォルダー*¥Redist の異なるサブフォルダーにあります。

アーキテクチャの検証の概要

InstallShield は、次の異なるレベルでビルド時のアーキテクチャ検証をサポートします:

- **なし** – ビルド時のアーキテクチャ検証を行いません。
- **厳密な検証をしない**—この種類の検証を使って、x86 および x64 .msi パッケージ (Template Summary プロパティで指定) をビルドでき、両方の種類のパッケージで x86 と x64 製品ファイルとカスタム アクション ファイル を混合することができます。

  ビルド時に、ISRedistPlatformDependentFolder と ISRedistPlatformDependentExpressFolder パス変数は、x86 InstallShield カスタム アクション DLL を含む x86 の場所をポイントします。InstallShield は、プロジェクトにこれらのカスタム アクションを必要とするサポートが含まれている場合、これらの x86 カスタム アクション DLL をビルドに含みます。

  [厳密な検証をしない] オプションは、Template Summary プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガしません。

  [厳密な検証をしない] オプションは、Template Summary プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合に、ビルド警告をトリガしません。

  [厳密な検証をしない] オプションがデフォルトの選択です。

- **厳密な検証をする**—この種類の検証を行うと、Template Summary プロパティで Intel (x86 用) または x64 のどちらが指定されているかによって、InstallShieldは x86 専用または x64 専用 .msi パケージをビルドしようとします。

  Template Summary プロパティで Intel が指定されている場合、ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder パス変数は x86 InstallShield カスタム アクション DLL を含む x86 の場所をポイントします。ただし、Template Summary プロパティで x64 が指定されていて、厳密な検証を行う場合、これらのパス変数は x64 InstallShield カスタム アクション DLL を含む x64 の場所をポイントします。プロジェクトにこれらのカスタム アクションのいずれかを必要とするサポートが含まれている場合、InstallShield は適切な x86 または x64 バージョンの DLL をビルドに追加します。

厳密な検証を行うオプションは、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガする場合があります。したがって、x86 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x64 カスタム アクションに対してビルド エラーを生成します。x64 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x86 カスタム アクションに対してビルド エラーを生成します。

[厳密な検証を行う] オプションは、Template Summary プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合に、ビルド警告をトリガする場合があります。したがって、x86 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x64 製品ファイルに対してビルド警告を生成します。x64 パッケージをビルド時に検証中、InstallShield はパッケージに含まれる各 x86 製品ファイルに対してビルド警告を生成します。

***ヒント・** Template Summary プロパティに関する情報は、「"テンプレート概要" プロパティを使用する」を参照してください。*

64 ビット オペレーティング システムをターゲットとする処理についてのヒントは、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。

アーキテクチャ検証の種類を選択する

InstallShield では、製品の [リリース] ビューで定義されている各製品構成に使用するアーキテクチャ検証の種類を指定できます。

タスク ***ビルド時に実行するアーキテクチャ検証の種類を選択するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成する製品構成を選択します。
3. 次いで、**[全般]** タブの **"アーキテクチャの検証"** 設定で、適切なオプションを選択します。選択可能なオプションは以下のとおりです:
   - なし
   - 厳密な検証をしない
   - 厳密な検証をする

厳密なアーキテクチャの検証におけるビルド エラーと警告のトラブルシューティング

厳密なアーキテクチャの検証を使用して、Template Summary プロパティに x64 を指定する .msi パッケージをビルドする場合、ビルド時に InstallShield が次のようなエラーおよび警告を生成する可能性があります:

エラー –7319: 32 ビットの標準 DLL カスタム アクション MyCustomAction を 64 ビット限定のパッケージに含めることはできません。

警告 –7326: 32 ビット PE ファイル C:¥SourceFiles¥x86¥Myx86File.exe を64 ビット限定のパッケージに含んでいます。

多くの場合、x64 ターゲット システム上で x86 ファイルを実行することができます。ただし、x64 システムが WOW64 をサポートしない場合、x86 ファイルを実行することができないこともあります。

要件によって、ビルド エラーまたは警告で参照されている x86 ファイルまたはカスタム アクションを x64 ファイルまたはカスタム アクションで置換することが推奨されます。その他の状況下では、ビルド エラーまたは警告を無視することができます。たとえば、コンポーネントが x64 システムにインストールされないように防ぐ条件がファイルのコンポーネントに含まれている場合、対応するビルド警告を無視することができます。一部の状況において、要件を再検討し、WOW64 をサポートしない x64 ターゲット システムをサポートしない結論もあります。

同様に、x86 .msi パッケージをビルドするときに厳密なアーキテクチャの検証を使用する場合、次のようなビルド エラーと警告が発生する可能性があります:

エラー –7320: 64 ビットの標準 DLL カスタム アクション MyCustomAction を 32 ビット限定のパッケージに含めることはできません。

警告 –7327: 64 ビット PE ファイル C:¥SourceFiles¥x64¥Myx64File.exe を32 ビット限定のパッケージに含んでいます。

x86 システムで x64 ファイルまたはカスタム アクションを実行することはできません。要件によっては、プロジェクトを変更するか、ビルド エラーと警告を無視することができます。一部の場合、要件を再検討するかもしれません。

リリースに関する作業

プロジェクト: *インストール プロジェクト: インストール プロジェクトをビルドする前に、機能、コンポーネント、ファイル、ショートカット、レジストリ エントリ、エンドユーザー インターフェイスなどの、プロジェクトの各要素に対して設定のデザインと構成が完了していることを確認します。*

マージ モジュール プロジェクト: パッケージをビルドする前に、コンポーネント、ファイル、ショートカット、レジストリ エントリなど、マージ モジュールのすべての要素について設定のデザインと構成が完了していることを確認します。

InstallShield でプロジェクトのデザインが完了すると、構成およびビルドしてエンドユーザーへ配布できるリリースを作成できます。1 つのプロジェクトに対して複数リリースの作成したり、異なるメディア タイプおよび異なる要件の組み合わせにしたがって各リリースを構成したりすることができます。InstallShield では、次のような方法を利用して、リリースの作成またはビルドを行うことができます。

- リリース ウィザードを使用する。(このメソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは使用できません。)
- [リリース] ビューを使用する。
- ISCmdBld.exe を使用してコマンドライン ビルドからビルドする。
- オートメーション インターフェイスでリリースをビルドする。(このメソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは使用できません。)

Visual Studio 内部から InstallShield リリースをビルドする方法については、「Microsoft Visual Studio でリリースをビルドする」を参照してください。

ヒント: *リリースをビルドする際、Windows エクスプローラーが* Disk1 *フォルダーまたはサブフォルダーをポイントしていないことを確認してください。Windows エクスプローラーが* Disk1 *フォルダーをポイントしていると、ビルド プロセスが失敗してエラーが生成されます。*

リリース ウィザードを使用して、リリースの作成およびビルドを行う

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

リリース ウィザードでは、製品のリリースのビルド、および、リリースの設定の指定を簡単に行うことができます。

タスク ***リリース ウィザードを起動して、インストール パッケージをビルドするには、以下のいずれかを実行します。***

- ツールバーの **[リリース ウィザード]** ボタンをクリックします。
- **[ビルド]** メニューで **[リリース ウィザード]** をクリックします。
- **[リリース]** ビューで、リリース名を右クリックしてから **[リリース ウィザード]** をクリックします。

リリース ウィザードのパネルの指示に従います。ウィザードから返されるエラーの一覧については、「ビルド エラーと警告」を参照してください。

[リリース] ビューを使用して、リリースの作成およびビルドを行う

プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[リリース] ビューでは、リリースの設定の指定やリリースのビルドを行うことができます。リリースの作成とビルドの手順は、使用しているプロジェクトの種類によって異なります。

基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトにおけるリリースの作成とビルド

タスク ***リリースの作成とビルドを行うには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで新しいリリースを含める製品構成を右クリックして、**[新しいリリース]** をクリックします。新しい製品構成の作成方法については、「製品構成の使用」を参照してください。InstallShield によって、製品構成の下に新しい行が追加されます。設定がタブ上に表示されます。
3. リリースの設定を適宜構成します。
4. リリースを右クリックして、**[ビルド]** をクリックします。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

タスク ***リリースの作成とビルドを行うには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーを右クリックしてから、**[新しいリリース]** をクリックします。新しいリリースが追加されます。設定がタブ上に表示されます。
3. リリースの設定を適宜構成します。
4. リリースを右クリックして、**[ビルド]** をクリックします。

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにおけるリリースの作成とビルド

タスク ***リリースの作成とビルドを行うには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーを右クリックして、**[新しいリリース]** をポイントし、作成するメディアの種類をクリックします。エクスプローラーに、新しいリリースが追加されます。設定がタブ上に表示されます。
3. リリースの設定を適宜構成します。
4. リリースを右クリックして、**[ビルド]** をクリックします。

リリースの場所の設定

プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールのディスク イメージ フォルダーは、リリースの場所にビルドされます。圧縮されていないアプリケーション ファイルの保存に必要な、すべての追加のファイルやフォルダーは、ディスク イメージ フォルダーのサブフォルダーに配置されます。リリースの場所は、プロジェクトの場所のサブフォルダーです。

ビルドしたリリースを表示するには、製品名の下にある [ディスク イメージ] をクリックして、[リリース] ビューでそのリリース名をクリックします。リリースの場所を変更していない場合、ビルドされたインストール パッケージは、次のフォルダーに配置されます。

InstallShieldプロジェクト フォルダー¥*プロジェクト名*¥*製品構成*¥*リリース名*¥DiskImages¥Disk1

リリースの場所は、リリース ウィザードの [詳細設定] パネル、または [リリース] ビューの "リリースの場所" プロパティのいずれかで設定できます。

コマンドラインからのリリースのビルド

プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

Windows Installer ベースのプロジェクト、InstallScript プロジェクト、およびアドバンスト UI またはスイート/アドバンスト UI プロジェクトでは、ISCmdBld.exe を使って、コマンドラインからリリースをビルドできます。また、プロジェクトに InstallScript が含まれている場合、リリースがビルドされる前に、ISCmdBld.exe によってコンパイルされます。

ISCmdBld.exe を使ったインストールのビルドは、バッチ ファイルからビルドをするときに便利です。また ISCmdBld.exe を使用してコマンドラインからリリースをビルドすると、リリースをビルドするのに加えて .ipr ファイルと .ipo ファイル (InstallShield Professional で作成されたオブジェクト プロジェクト ファイル) が .ism ファイルに変換されます。

ヒント: *バイナリまたは XML フォーマットの .ism ファイルは、どちらもコマンドライン ビルドで利用できます。コマンドライン ビルドは、アドバンスト UI およびスイート/アドバンスト UI プロジェクト ファイル (.issuite) でも使用できます。*

Standalone Build で ISCmdBld.exe *を使って、コマンドラインからリリースをビルドできます。詳細については、「Standalone Build」を参照してください。*

ISCmdBld.exe を使用して、コマンドラインからリリースをビルドする

プロジェクト: *この情報は、次のプロジェクトに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

ISCmdBld.exe は、デフォルトで、InstallShield フォルダーの System サブフォルダーに配置されます。ISCmdBld.exe は、インストールされた場所から移動させないでください。

ISCmdBld.exe の構文および利用可能なコマンドライン パラメーターについては、「ISCmdBld.exe」を参照してください。

.ini ファイルでコマンドライン ビルド パラメーターを渡す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

コマンドラインを使ったビルド中に多くのパラメーターを渡す場合や、同じパラメーターを継続的に渡す場合、.ini ファイルを使うと便利な場合があります。以下は、IsCmdBld.exe を実行して、**MySetup.ini** ファイルで指定したパラメーターと共に リリースをビルドするステートメントの例を示します。

```
ISCmdBld.exe -i "C:¥InstallShield 2022 Projects¥MySetup.ini"
```

コマンドラインでパラメーターを渡すときと同じ情報を .ini ファイルに含める必要があります。このファイルには 4 つのセクションがあります:

- [Project]—このセクションには、製品構成の名前およびプロジェクト ファイル (.ism) へのパスの エントリを含めます。パッチをビルドしている場合、ビルド中のパッチ構成の名前のエントリを含めます。
- [Release]—このセクションには、圧縮の種類 (圧縮または非圧縮)、ビルド フラグ、Setup.exe の設定、およびリリース名などのリリース構成情報のエントリを含めます。
- [Mode]—このセクションには、Silent=yes (リリースのビルド中に、ビルド エラーまたは警告メッセージを抑制する場合) のような、使用可能なオプション エントリを含めます。このセクションでは、ログ ファイルを作成するかどうかを指定することもできます。
- [BuildLocation]—このセクションでは、リリースの出力場所をオプションで指定することができます。

一部のセクションは必須ではありません。コマンドラインから直接パラメーターを渡すことに関しては、サイレント ビルドやビルドの場所などの要件のためのパラメーターはオプションです。下の .ini ファイルの例では、これらのパラメーターは [Mode] セクションと [BuildLocation] セクションにあります。デフォルトを使用する場合は、.ini ファイルからこれらのエントリを省略してください。デフォルトでは、ログ ファイルは作成されず、インストールもサイレント モードで実行されません。また、リリースは [オプション] ダイアログ ボックスの [ファイルの場所] タブで指定されたプロジェクトの場所に作成されます。

サンプル .ini ファイル

次のテーブルは、サンプル .ini ファイルに含まれている 4 つのセクションからのサンプル エントリです。サンプル エントリは、各テーブルの最初の列に表示されています。その他の列は、対応するコマンドライン パラメーターと説明です。

[Project] セクションのエントリ

テーブル 1 ・ サンプル エントリ – [Project] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
Name="C:¥InstallShield 2022 Projects¥Othello.ism"	–p	.ism ファイルへのパス。
BuildLabel=Label1	–a	ビルドするリリースの製品構成の名前。パラメーターが存在しない場合は、作成されます。
Product=Othello		プロジェクトの名前。このエントリを使って、[一般情報] ビューで指定された値をオーバーライドできます。
PatchConfigName="Version 1.2"	–patch_config	ビルドする [パッチのデザイン] ビューのパッチ構成の名前。
CompileScript=no	–n	Setup.rul がビルドプロセスの一部としてコンパイルされないようにします。 ***メモ:*** *CompileOnly を CompileScript と共に使用することはできません。*

[Release] セクションのエントリ

テーブル 2 ・ サンプル エントリ – [Release] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
BuildFlags=flags	–f	カンマで区切られたインストールに含まれるリリース フラグ。
Configuration=COMP	–c	圧縮、非圧縮の違い。
Name=Othello Beta	–r	リリース名。
SetupEXE=yes	–e	Setup.exe ファイルを作成します。

[Mode] セクションのエントリ

テーブル 3 ・ サンプル エントリ – [Mode] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
CubFile="C:¥Program Files¥InstallShield¥2022¥Support¥Validation¥MyValidation.cub"	–m	インストールまたはマージ モジュール パッケージのビルドが完了したあと、指定された .cub ファイルを使って検証を実行します。完全パスと名前を .cub ファイルに含めます。 複数の .cub ファイルを使用するには、各パスをセミコロン (;) で区切ります。長いファイル名は引用符で囲んでください。 たとえば、次のエントリは、検証に InstallShield 検証スイート – Windows 8 (ISWin8.cub) および完全 MSI 検証スイート (darice.cub) が必要であることを示します: CubFile="C:¥Program Files¥InstallShield¥2022¥Support¥Validation¥ISWin8.cub;C:¥Program Files¥InstallShield¥2022¥Support¥Validation¥darice.cub"
Silent=yes	–s	サイレント モードで実行します。
StopOnFirstError=yes	–x	最初にエラーが発生した時点でビルドを中止します。
CompileOnly=no	–q3	Setup.rul のみをコンパイルして、以前にビルドされていれば、Setup.inx を .msi ファイル の**Binary** テーブルにストリーミングします。 ***メモ:*** *CompileOnly、BuildTablesRefreshFiles、および BuildTablesOnly エントリは、相互に排他的です。1 つだけ [はい] に設定することができます。* *CompileOnly を CompileScript と共に使用することはできません。*
BuildTablesRefreshFiles=no	–q2	Windows Installer テーブルをビルドしてファイルをリフレッシュします。 ***メモ:*** *CompileOnly、BuildTablesRefreshFiles、および BuildTablesOnly エントリは、相互に排他的です。1 つだけ [はい] に設定することができます。*

テーブル 3 • サンプル エントリ – [Mode] セクション (.ini ファイル) (続き.)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
BuildTablesOnly=yes	–q1	リリースの Windows Installer テーブルのみをビルドします。 ***メモ•*** *CompileOnly、BuildTablesRefreshFiles、および BuildTablesOnly エントリは、相互に排他的です。1 つだけ [はい] に設定することができます。*
UpgradeOnly=no	–u	リリースをビルドではなく、アップグレードすることができます。
Verbose=yes	–v	エンジン ログ ファイルを生成。
WarningAsError=yes	–w	ビルド警告すべてをビルドエラーとして扱う。
MSIVersion=2.0	–g	ターゲット システムで必要な Windows Installer のバージョン。 ***メモ•*** *このエントリは Standalone Build のみに適用します。*
DotNetVersion=	–j	ターゲット システムに必要な Microsoft .NET Framework の最小バージョン。このパラメーターはオプションで、デフォルトでは InstallShield がサポートする .NET Framework の最新バージョンに設定されています。 ***メモ•*** *このエントリは Standalone Build のみに適用します。*
DotNetPath="C:¥..."	–t	ビルド マシン上の Microsoft .NET Framework へのパス。 ***メモ•*** *このエントリは Standalone Build のみに適用します。*

テーブル 3 • サンプル エントリ – [Mode] セクション (.ini ファイル) (続き.)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
SkipValidators=yes	–h	通常ビルドの終わりで実行されるアップグレード検証ツールをオフにすることができます。 *メモ• このエントリは Standalone Build のみに適用します。*
MergeModulePath="C:¥..."	–o	マージ モジュール (.msm ファイル) のパスを検索します。 詳細については、「マージ モジュールを含むディレクトリを指定する」を参照してください。 *メモ• このエントリは Standalone Build のみに適用します。*
PrerequisitePath="C:¥..."	–prqpath	InstallShield 前提条件ファイル (.prq) の検索パス。 詳細については、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。 *メモ• このエントリは Standalone Build のみに適用します。*
LicCheckTimeOut="40"	–licCheckTimeOut <秒>	このパラメーターを使って、ライセンス サーバーからライセンスを使用可能かどうかチェックする時間帯を設定することができます。

[BuildLocation] セクションのエントリ

テーブル 4 • サンプル エントリ – [BuildLocation] セクション (.ini ファイル)

エントリ	対応する ISCmdBld.exe コマンドライン パラメーター	説明
Path="C:¥InstallShield 2022 Projects¥Othello"	–b	リリースの出力場所。

.ini ファイルのエントリについての詳細は、「コマンドラインからのリリースのビルド」を参照してください。

ISBuild.exe を使用してコマンドラインからのビルドする

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

重要: *ISBuild.exe が使用できなくなったため、ISBuild.exe の代わりに ISCmdBld.exe を使ってビルドをリリースしてください。*

ISBuild.exe (レガシー InstallScript プロジェクトに提供されているコマンドライン ツール) は、ISCmdBld.exe に呼び出されるシェルです。ISCmdBld.exe は、.ipr ファイルと .ipo ファイル (InstallShield Professional を使って作成されたオブジェクト プロジェクト) を .ism ファイルに変換処理して、リリースをビルドします。

ISBuild.exe で使用できるコマンドライン パラメーターの一覧については、「ISBuild.exe」を参照してください。

コマンドラインからの自己展開型実行可能ファイルをビルドする

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ReleasePackager.exe を使用して、コマンドラインから自己展開型実行可能ファイルをビルドすることができます。バッチ ファイルからビルドを行うときに便利です。

ReleasePackager.exe は、デフォルトで InstallShield フォルダーの System サブフォルダーに配置されます。ReleasePackager.exe を、インストールされた場所から移動しないでください。

ReleasePackager.exe の構文と利用可能な引数およびオプションについては、「ReleasePackager.exe」を参照してください。

リリースの再ビルド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

リリース ウィザードで設定したデフォルトのエントリを使用して、いつでもリリースを再ビルドできます。再ビルドされるのは、現在強調表示されているリリースです。

タスク ***リリースを再ビルドするには、以下のいずれかを行ってください。***

- **[ビルド]** ボタンをクリックします。

- **[ビルド]** メニューで **[ビルド]** をクリックします。
- **[リリース]** ビューで、リリース名を右クリックしてから **[ビルド]** をクリックします。

また、リリース ウィザードを再度実行して、リリースを再ビルドすることもできます。ウィザードでは InstallShield プロジェクト ファイルのリリース設定が保持されるので、リリース自体を削除した場合でも、最後にリリースをビルドした際に設定したオプションがすべてウィザードに表示されます。

新しいインストール パッケージを表示するには、[リリース] ビューで適切なリリースのディスク イメージをクリックします。デフォルトのリリースの場所を変更していない場合、ビルドされたインストール パッケージは、次のフォルダーに配置されます。

C:¥InstallShield 2022 プロジェクト¥*プロジェクト名*¥*プロジェクト構成*¥*リリース名*¥DiskImages¥Disk1

メモ: *リリースをビルドする際、Windows エクスプローラーが* Disk1 *フォルダーまたはサブフォルダーをポイントしていないことを確認してください。*Disk1 *フォルダーがポイントされていると、ビルド処理は終了しません。Windows Explorer がサブフォルダーにアクセスしていると、ビルドでエラーが発生します。*

クイック ビルドを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールをテストするとき、ファイルに変更がなければインストールのすべてを頻繁にビルドする必要はありません。InstallShield では、[テーブルのビルドのみ] と [テーブルのビルドとファイルのリフレッシュ] の 2 つのクイック ビルド オプションが提供されています。

テーブルのみをビルドする

このオプションの名前が示すように、Windows Installer のテーブルだけがビルドされます。このセットアップをまだビルドしていない場合、新しい .msi ファイルが作成されますが、インストールにはファイルが追加されません。インストールを既にビルドしている場合、.msi ファイルは、すべてのテーブルがビルドされたとき更新されますが、ファイルは転送されません。このオプションを使用して、インストールのユーザー インターフェイスをテストすることができます。

タスク ***インストールの Windows Installer テーブルのみをビルドするには、以下のいずれかを実行します。***

- **[ビルド]** メニューで **[テーブルのみをビルド]** をクリックします。

- **[リリース]** ビューで、ビルドするリリースを右クリックしてから **[テーブルのみをビルド]** をクリックします。

テーブルのビルドとファイルのリフレッシュ

すべてのインストール ファイルを新しい場所に転送せずに、ほぼ完全にビルドを行う場合、.msi ファイルを再ビルドして、Files テーブルを更新するビルドを実行できます。これにより、新しいファイルや変更されたファイルがインストールに含まれます。変更されたファイルは、サイズまたはタイムスタンプが、ビルドにすでに含まれるコピーと異なる場合にのみ更新されます。削除されたファイルへの参照は、インストールから削除されますが、ファイルは、ビルドの場所に残ります。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワークイメージである場合だけ作動します。

タスク ***Windows Installer テーブルのみをビルドして、インストールのファイルをリフレッシュするには、以下のいずれかを実行します。***

- **[ビルド]** メニューで **[テーブルのビルドとファイルのリフレッシュ]** をクリックします。
- **[リリース]** ビューで、ビルドするリリースを右クリックしてから **[テーブルのビルドとファイルのリフレッシュ]** をクリックします。

バッチ ビルドを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

構成の異なる複数のリリースを同時にビルドする必要がある場合があります。この機能をバッチビルドと呼びます。

タスク ***InstallShield でバッチ ビルドを実行するには、次の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. 基本の MSI、InstallScript MSI、マージ モジュール プロジェクトの場合、[製品構成] を右クリックして **[バッチ ビルド]** を選択します。

   InstallScript または InstallScript オブジェクト プロジェクトの場合、**[リリース]** エクスプローラーを右クリックして **[バッチ ビルド]** を選択します。

[バッチ ビルド] ダイアログが開き、ビルドする製品構成とリリースを選択するようメッセージが表示されます。

ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する

エディション: *この機能は、InstallShield の Premier Edition で提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクトによって異なる情報がある場合は、その内容が記載されています。

InstallShield では、ビルド プロセスの様々な段階で実行するコマンドを指定できます。たとえば、基本の MSI プロジェクトを作成中の場合、ビルド時に InstallShield が作成する .msi パッケージを、デジタル署名が行われて Setup.exe ファイルにストリームされる前に変更したい場合があります。ビルド中、適切な時点で .msi パッケージに必要な変更を行うためのスクリプトを実行するコマンドを指定できます。

コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンド用に定義された以下の変数を使用することもできます:

テーブル 5 ・ ビルド イベント変数

変数	プロジェクトの種類	説明
<ISReleaseName>	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	この変数は、InstallShield がビルド中のリリースの名前を示します。
<ISProductConfigName>	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	この変数は、InstallShield がビルド中のリリースを含む製品構成の名前を示します。(InstallScript プロジェクトの場合、この名前は常に *Media* です。)
<ISReleasePath>	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	この変数は、InstallShield がビルド中のリリースがビルドされる場所を示します。これは、[リリース] ビューにあるリリースの [ビルド] タブにある "リリースの場所" 設定で行います。

テーブル 5 - ビルド イベント変数 (続き.)

変数	プロジェクトの種類	説明
<ISReleaseUsesShallowFolderPaths>	基本の MSI、InstallScript MSI	この変数は True または False に設定され、InstallShield がリリースをビルドする際にシャロー フォルダー ディレクトリ構造を使用するかどうかを示します。

ビルド時に、InstallShield は <ISReleaseName>、<ISProductConfigName>、<ISReleasePath>、および <ISReleaseUsesShallowFolderPaths> の一時環境変数を設定します。InstallShield はまた、プロジェクトに含まれるすべての定義済みパス変数の一時環境変数も設定します。バッチ ファイル内のこれらの環境変数の値の 1 つを、その変数名をパーセント記号 (%) で囲んで参照することができます。例:

set PATH = %<ISReleaseName>%;%PATH%

ビルドが終了すると、設定された一時環境変数は削除されます。

タスク ***ビルド前、ビルド中、またはビルド後に実行する 1 つ以上のコマンドを指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。
3. **[イベント]** タブをクリックします。
4. 以下の 1 つ以上の設定で、InstallShield が実行するコマンドを入力します:
   - ビルド前のイベント
   - ファイルの準備イベント
   - 圧縮前のイベント (このイベントは、スイート/アドバンスト UI プロジェクトでは使用できません。)
   - ビルド後のイベント

   コマンド プロンプトから実行されるのと同じ要領で各コマンドを入力します。つまり、たとえばスペースを含むパスを指定する場合には、そのパスを引用符で囲みます。

   各設定の詳細については、「リリースの [イベント] タブ」を参照してください。

   これらの任意の設定に 1 つ以上のコマンドを指定するには、この設定で省略記号ボタン (...) をクリックします。1 つ以上のコマンドを入力することができるダイアログ ボックスが開きます。1 行につき 1 つのコマンドを入力します。

1 つのイベント設定に対して 1 つ以上のコマンドを入力すると、InstallShield はビルド時に、その設定でリストされている順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。

ヒント: *詳細ビルド ログには、ビルド時に実行される各ビルド イベントが表示されます。ビルド イベントのトラブルシューティングには、詳細ログ ファイルを生成して、"Launching prebuild イベント"、"Launching precompression build イベント"、および "Launching postbuild イベント" などの文字列を検索します。ログ ファイルのこれらの領域には、ビルド時に実行されたコマンドについての詳細がリスト表示されます。*

詳細ビルド ログを生成するには、コマンドラインから IsCmdBld.exe *を起動するときに、–v パラメーターを渡します。*

ファイルの準備イベントを使って、署名されていない InstallShield ファイルに署名を行う

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[メディア] の **[リリース]** ビューで、リリースの **[イベント]** タブにある **"ファイルの準備イベント"** 設定を使って、ユーザーは InstallShield ビルド処理中に生成されたすべての内部ファイルに署名を行う独自のカスタム スクリプトを呼び出すことができます。これは、独自のスクリプトまたは実行可能ファイルを実行して InstallShield のサインアウトを構成したい場合に便利です。

ファイルの準備イベント コマンドは、プロジェクトのビルド中に、InstallShield の内部サポートファイルが準備され、それらのファイルが MSI または Setup.exe file にストリームされる前に呼び出されます。

プロジェクトのビルド中、InstallScript MSI プロジェクトではファイルの準備イベント コマンドが 2 回呼び出されますが、その他のプロジェクトでは一回だけ呼び出されます。次の InstallShield プロジェクトをビルド中に署名されていない InstallShield ファイルの場所:

テーブル 6 ・ 署名されていないファイルの場所

プロジェクト	署名されていないファイルの場所
基本の MSI	• _XXXXXXXXXX.clr — プロジェクトに管理対象カスタム アクションまたは PowerShell カスタム アクションが含まれている場合、内部フォルダーで署名なしの cir ファイルを使用できます。このフォルダーは、プロジェクトのビルド中に製品構成フォルダー内に作成されます。 • XXXX.tmp — プロジェクトでアプリケーションまたはショートカットのアイコンを設定するように構成されている場合、内部フォルダーに署名されていない一時ファイルが作成されます。 • ISSetup.dll — プロジェクトに InstallShield カスタム アクションが含まれている場合、内部フォルダーに署名されていない dll ファイルが作成されます。
InstallScript	• _isuser_0xNNNN.dll — InstallScript プロジェクト ダイアログが変更された場合、Script File フォルダーの隣にあるプロジェクト フォルダーに署名されていない dll ファイルが作成されます。 • ISSetup.dll — 署名されていない dll が Disk1 フォルダーに作成されます。

テーブル 6 • 署名されていないファイルの場所

プロジェクト	署名されていないファイルの場所
InstallScript MSI	• _isuser_0xNNNN.dll — InstallScript MSI プロジェクト ダイアログが変更された場合、Script File フォルダーの隣にあるプロジェクト フォルダーに署名されていない dll ファイルが作成されます。 • _XXXXXXXXXX.clr — プロジェクトに管理対象カスタム アクションまたは PowerShell カスタム アクションが含まれている場合、内部フォルダーで署名なしの cir ファイルを使用できます。このフォルダーは、プロジェクトのビルド中に製品構成フォルダー内に作成されます。 • XXXX.tmp — プロジェクトでアプリケーションまたはショートカットのアイコンを設定するように構成されている場合、内部フォルダーに署名されていない一時ファイルが作成されます。 • ISSetup.dll — 署名されていない dll ファイルが内部フォルダーに作成されます。この DLL は InstallScript MSI プロジェクトに必要です。 ***メモ•*** *ファイルの準備イベント コマンドは、InstallScript MSI プロジェクトで 2 回呼び出されます。*
スイート/アドバンスト UI	• ISSetup.dll — プロジェクトに InstallScript イベントが含まれている場合、内部フォルダーに署名されていない dll ファイルが作成されます。このフォルダーは、プロジェクトのビルド中にリリース フォルダー内に作成されます。 • Setup64.exe — プロジェクトに x64 パッケージまたは条件が含まれている場合、内部フォルダーに署名されていない exe ファイルが作成されます。

ビルドエラーと警告を解決する

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

リリースをビルドすると、InstallShield ユーザー インターフェイスの下の部分に出力ウィンドウが開きます。このウィンドウの [タスク] タブには、ビルド プロセス中に発生したすべてのビルドエラーおよび警告が一覧で表示されます。

タスク ***発生したビルドエラーおよび警告を解決する方法を調べるには、次のどちらかを行なってください。***

- **[出力]** ウィンドウの **[タスク]** タブでエラーコードをクリックすると、インターネット ブラウザーを起動し、選択されたビルド エラーまたは警告に関するナレッジ ベース記事およびドキュメントのトピックを表示することができます。
- ビルド エラーと警告の解決方法に関するヒントを見ることができる「ビルド エラーと警告」をご覧ください。

コマンドラインからプロジェクトの設定を変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield オートメーション インターフェイスでは、VBScript スクリプトや Visual Basic アプリケーションなどを使って、無人のビルド プロセスから多数のプロジェクト設定をクエリおよび変更できます。

オートメーション インターフェイスを使ってプロジェクトにアクセスするスクリプトは、次のような枠組となります (このスクリプトを Framework.vbs という名前のテキストファイルにコピーしてから、ファイルのアイコンをダブルクリックすることができます)。

```
Set oProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
oProject.OpenProject "C:¥MySetups¥MyProject.ism"

' ここでクエリと変更を行う

oProject.SaveProject ' プロジェクトを変更する場合のみ必要
oProject.CloseProject
```

ビルドのキャンセル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*

- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

タスク ***既に起動したビルドをキャンセルするには、以下の手順に従います:***

ツールバー上の **[ビルドの中止]** ボタンをクリックします。

Standalone Build

InstallShield には、Standalone Build モジュールがあり、InstallShield のインストールをコンパイルする部分 (および含める再配布可能ファイル) のみインストールすることができます。Standalone Build と再配布可能ファイルのビルド マシンへのインストール方法については、「Standalone Build をビルド マシンにインストールする」を参照してください。

InstallShield 2025 Standalone Build は、同じマシン上で別のメジャー バージョンの Standalone Build と共存することができます。ただし、異なるフォルダーに格納されている必要があります。

プロジェクトのアップグレード

Standalone Build は、InstallShield Developer 7.0 またはそれ以降を使って作成されたプロジェクトを自動的にアップグレードします。ただし、InstallShield for Windows Installer を使って作成されたプロジェクトをアップグレードすることはできません。

コマンドライン ビルドの使用

コマンドラインから Standalone Build を実行することができます。詳細については、「スタンドアロン コマンドライン ビルド」を参照してください。

スタンドアロン オートメーション インターフェイスの利用

Standalone Build は、オートメーション インターフェイスのスタンドアロン バージョン (スタンドアロン オートメーション インターフェイス) を含みます。

Standalone Build をビルド マシンにインストールする

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

- *マージ モジュール*
- *スイート/アドバンスト UI*

Standalone Build のインストール

Standalone Build のインストールは、単一の圧縮済み実行可能ファイルで構成されています。複数言語インストールのビルドに必要なすべてのファイルが含まれています。

タスク ***Standalone Build をビルド マシンにインストールするには、次の手順を実行します。***

1. Standalone Build インストールの実行可能ファイルを見つけるには、次の手順を実行します。
   - InstallShield DVD をお持ちの場合、ファイルは DVD に含まれているので DVD ブラウザーを使って見つけることができます。
   - InstallShield をダウンロードされた場合、Standalone Build インストール ファイルは InstallShield のダウンロードおよびライセンス ガイド (http://www.installshield.com/instructions/sab.asp) の手順にに従ってダウンロードできます。
2. 実行可能ファイルをダブルクリックして、インストールを実行します。Standalone Build のライセンシング構成に関する詳細は、InstallShield ダウンロード & ライセンス ガイド (http://www.installshield.com/instructions/sab.asp) を参照してください。

インストール中、[カスタム セットアップ] ダイアログが表示され、デフォルトで無効化されている Standalone Build の 2 つの機能が表示されます。これらの機能を使用するには、有効化します。

- **オートメーション インターフェイス**—Standalone Build でオートメーション インターフェイスを使用するのに必要なファイルをインストールおよび登録します。
- **InstallScript オブジェクト サポート**—InstallScript オブジェクトを含む InstallScript プロジェクトのビルドを可能にする InstallScript オブジェクトのサポート ファイルをインストールおよび登録します。

Standalone Build のインストール中に登録されるファイル

Standalone Build をインストールすると、以下のマージ モジュールで既存しないものがインストールおよび登録されます。

- ATL 3.0 (ATL.msm)
- Microsoft C ランタイム ライブラリ (MSVCRT.msm)
- Microsoft C Runtime Library 7.1 (VC_User_CRT71_RTL_X86_---.msm)
- MSXML3 (msxml3_wim32.msm)
- MSXML4 (msxml4sxs32.msm または msxml4sys32.msm)
- Visual C++ 8.0 ATL (x86) WinSXS MSM (Microsoft_VC80_ATL_x86.msm)
- Visual C++ 8.0 ATL.Policy (x86) WinSXS MSM (policy_8_0_Microsoft_VC80_ATL_x86.msm)
- Visual C++ 8.0 CRT (x86) WinSXS MSM (Microsoft_VC80_CRT_x86.msm)
- Visual C++ 8.0 CRT.Policy (x86) WinSXS MSM (policy_8_0_Microsoft_VC80_CRT_x86.msm)

Standalone Build インストールはまた、Microsoft Scripting Runtime Library (scrrun.dll) が既存しない場合に、これをインストールおよび登録します。これは、[SystemFolder] にインストールされます。

Standalone Build をインストールする代わりに、1 台目のマシンから別のマシンに Standalone Build ファイルをコピーした場合で、コピー先のマシンにこれらのマージ モジュールに含まれているテクノロジーと scrrun.dll ファイルが既存しないときは、手動でインストールしなくてはなりません。このマージ モジュール テクノロジー用のインストール パッケージは、Microsoft Web サイト (http://www.microsoft.com) から取得できます。

オプションのオートメーション サポート機能のインストール中に登録されるファイル

Standalone Build インストールのオートメーション インターフェイス機能は、Standalone Build でオートメーション インターフェイスを使用するときに必要なファイルをインストールおよび登録します。登録する必要があるファイルの一覧は次のとおりです:

- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥IsmAuto.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥Ismmupdater.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥ISWIBuild.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥IsAppServices.tlb
- *Program Files フォルダー*¥Common Files¥InstallShield¥Shared¥IsUpgrade.tlb
- *Standalone Build Program Files フォルダー*¥System¥ISWiAutomationAutomation Interface Version.dll

Standalone Build をインストールする代りに、1 台目のマシンから別のマシンに Standalone Build ファイルをコピーした場合、コピー先のマシンの Standalone Build でオートメーション インターフェイスを使用するには、これらのファイルを手動で登録しなくてはなりませんん。

オプションの InstallScript オブジェクト サポート機能のインストール中に登録されるファイル

InstallScript オブジェクト サポート機能は、InstallScript オブジェクトを含む InstallScript プロジェクトのビルドを可能にする InstallScript オブジェクトのサポート ファイルをインストールおよび登録します。登録する必要があるファイルの一覧は次のとおりです:

- *Standalone Build Program Files フォルダー*¥System¥DObjAutomation Interface Version.dll
- *Standalone Build Program Files フォルダー*¥System¥ISBE.dll
- *Standalone Build Program Files フォルダー*¥System¥ISMKAutomation Interface Version.dll
- *Standalone Build Program Files フォルダー*¥System¥StockWiz.dll

Standalone Build をインストールする代りに、1 台目のマシンから別のマシンに Standalone Build ファイルをコピーした場合、コピー先のマシンで InstallScript オブジェクトを含む InstallScript プロジェクトをビルドするには、これらのファイルを手動で登録しなくてはなりませんん。

Standalone Build のファイルを手動で登録する

前述の Standalone Build 用の DLL ファイルの 1 つを手動で登録するには、System32 フォルダーにインストールされている Regsvr32.exe を使用します。

.tlb ファイルの 1 つを手動で登録するには、以下のファイルを使用します:

Standalone Build Program Files フォルダー¥System¥RegTypLib.exe

Standalone Build 用に再配布可能ファイルをビルド マシンへ追加する

マージ モジュール、オブジェクト、および InstallShield 前提条件を含み、再配布可能ファイルのほとんどは非常に多くのハード ディスク容量を使用するため、Standalone Build と共に自動的にはインストールされません。これによって、作成中の InstallShield プロジェクトに必要な再配布可能ファイルのみをビルド マシンにインストールすることができます。唯一の例外は .NET Framework 2.0 再配布可能ファイルで、これは自動的にインストールされます。

ヒント: *Standalone Build は、InstallShield がプログラム ファイルに使用するディレクトリ構造と同じ基本のディレクトリ構造を使用します。これにより、InstallShield があるマシンから再配布可能ファイルや他のファイルを Standalone Build があるマシンにコピーするとき、同じ相対パスを使用できるようになりました。たとえば、ファイルが InstallShield があるマシンの* InstallShield Program Files フォルダー¥Modules¥i386 *ディレクトリにある場合、そのファイルを Standalone Build があるマシンの S*tandalone Build Program Files フォルダー¥Modules¥i386 *ディレクトリにコピーします。*

タスク ***.NET 再配布可能ファイルをビルド マシンに追加するには、次の手順を実行します。***

1. InstallShield の完全バージョンを持つマシン上で、必要な再配布可能ファイルを参照します。通常ファイルは、InstallShield Program Files フォルダー¥Redist または Redist フォルダーのサブフォルダーに配置されています。InstallShield 再配布可能ファイルがマシン上で利用できる状態では無い場合、まずこれをダウンロードします。詳細については、「再配布可能ファイル ダウンローダー ウィザード」を参照してください。
2. InstallShield を持つマシンから、ビルドマシン上にある Standalone Build Program Files フォルダー へ再配布可能ファイルをコピーします。

タスク ***基本の MSI または InstallScript MSI プロジェクト用の再配布可能ファイルをビルド マシンに追加するには、次の手順を実行します。***

1. InstallShield の完全バージョンを持つマシン上で、必要な再配布可能ファイルを参照します。InstallShield 再配布可能ファイルがマシン上で利用できる状態では無い場合、まずこれをダウンロードします。詳細については、「再配布可能ファイルをコンピューターにダウンロードする」を参照してください。
   - InstallShield マージ モジュール の 2 つのデフォルトの場所は、次の通りです。

     InstallShield Program Files フォルダー¥Modules¥i386

     Common Files¥Merge Modules
   - InstallShield オブジェクトのデフォルトの場所は、次の通りです。

     InstallShield Program Files フォルダー¥Objects
   - すべての InstallShield セットアップ前提条件ファイルは、次の場所に格納されています:

     InstallShield Program Files フォルダー¥SetupPrerequisites
   - 独自のマージ モジュールを InstallShield でビルドした場合、[オプション] ダイアログ ボックスの [マージ モジュール] タブにリストされている場所のどこかに格納されています。
2. 再配布可能ファイルを InstallShield を持つマシンからビルド マシンの適切な場所へコピーします。
   - マージ モジュールを次のフォルダーに追加します。

Standalone Build Program Files Folder¥Modules¥i386

- オブジェクトを次のフォルダーに追加します。

  Standalone Build Program Files Folder¥Objects

- InstallShield セットアップ前提条件を次のフォルダーに追加します:

  Standalone Build Program Files Folder¥SetupPrerequisites

タスク ***InstallScript プロジェクト用の InstallShield 前提条件をビルド マシンに追加するには、以下の手順に従います:***

1. InstallShield の完全バージョンがインストールされているマシン上で、必要な InstallShield 前提条件を参照します。InstallShield 再配布可能ファイルがマシン上で利用できる状態では無い場合、まずこれをダウンロードします。詳細については、「再配布可能ファイルをコンピューターにダウンロードする」を参照してください。

   すべての InstallShield セットアップ前提条件ファイルは、次の場所に格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites

2. InstallShield 前提条件を InstallShield がインストールされているマシンからビルド マシンの適切な場所へコピーします。

   Standalone Build Program Files Folder¥SetupPrerequisites

タスク ***ビルド マシンに InstallScript プロジェクト用の InstallShield 前提条件の 1 つを追加するには、以下の手順を実行します:***

1. 再配布可能ファイルのインストールをダウンロードします。インストールは FlexNet Connect を通して提供されます。詳細については、「InstallShield のアップデートを取得する」を参照してください。

2. 再配布可能ファイル インストールをビルド マシン上で実行します。InstallShield ウィザードが [インストール先の選択] ダイアログを表示したら、次の場所を参照します。

   Standalone Build Program Files Folder¥ObjectsPro

   ObjectsPro フォルダーがまだ作成されていない場合、このフォルダーを追加する必要があります。

InstallScript オブジェクト インストールは、必要なオブジェクト ファイルをインストールおよび登録します。

タスク ***ビルドマシンに InstallScript 再配布可能ファイルまたはサードパーティ再配布可能ファイルの 1 つを追加するには、次の手順を実行します。***

1. 再配布可能ファイルを参照します。

2. これをビルド マシン上の次の場所に配置します。

   Standalone Build Program Files Folder¥ObjectsPro

   ObjectsPro フォルダーがまだ作成されていない場合、このフォルダーを追加する必要があります。

スタンドアロン コマンドライン ビルド

Standalone Build では、ISCmdBld.exe を使って、コマンドラインからリリースをビルドすることができます。手順については、「コマンドラインからのリリースのビルド」を参照してください。

ISCmdBld.exe がサポートするコマンドライン パラメーターの一覧については、「ISCmdBld.exe」を参照してください。

スタンドアロン オートメーション インターフェイス

Standalone Build は、オートメーション インターフェイスのスタンドアロン バージョン (InstallShield スタンドアロン オートメーション インターフェイス) を含みます。このインターフェイスは、InstallShield と共にインストールされる ISWiAutomationAutomation Interface Version.dll と同じファイルを使用しますが、インストールされる場所は異なります。

*メモ: Standalone Build を InstallShield と同じマシンにインストールする場合、最初にインストールされた ISWiAutomation*Automation Interface Version*.dll ファイルが登録されます。詳細については、「Standalone Build と InstallShield を同一のマシン上にインストールする」を参照してください。*

Standalone Build の将来のバージョンには、新しい ProgID および GUID が与えられます。これにより、複数のバージョンのスタンドアロン オートメーション インターフェイスを同じビルド マシンにサイドバイサイドでインストールすることができます。

ISWiAutomationAutomation Interface Version.dll の登録

スタンドアロン オートメーション インターフェイスは、ISWiAutomationAutomation Interface Version.dll というファイルに含まれています。Standalone Build を Standalone Build のインストールを起動してインストールした場合、このファイルは既に登録されています。インストールを実行する代りに、Standalone Build のファイルをコピーした場合、System32 フォルダーにインストールされている Regsvr32.exe を使って、このファイルを手動で登録する必要があります。ISWiAutomationAutomation Interface Version.dllファイルは、Standalone Build Program Files フォルダーの System サブフォルダーにインストールされます。

Standalone Build と InstallShield を同一のマシン上にインストールする

ほとんどの場合、InstallShield がインストールされているマシン上に Standalone Build がインストールされることはありません。同じマシン上に両方をインストールする場合は、スタンドアロン オートメーション インターフェイスは InstallShield が使用するのと同じ ISWiAutomationAutomation Interface Version.dll ファイル (ただし、異なる場所にインストールされている) を使用する点にご注意ください。また、スタンドアロン オートメーション インターフェイスと InstallShield は同じ ProgID (IswiAutoAutomation Interface Version.ISWiProject) を使用します。このため、適切な ProgID を反映させるためにコードを変更する必要なく、スタンドアロン オートメーション インターフェイスと InstallShield の両方で同じオートメーション スクリプトを使用できます。

InstallShield または Standalone Build のどちらかを最初にインストールしたとき、インストールが ISWiAutomationAutomation Interface Version.dll を登録します。次にもう 1 つの方のツールをインストールすると、2 番目のインストールは ISWiAutomationAutomation Interface Version.dll が既に登録済みであることを検出して、再登録を行いません。ISWiAutomationAutomation Interface Version.dll の 1 つのインスタンスが登録されているかぎり、Standalone Build と InstallShield の両方でオートメーション インターフェイスを使用できます。

最初にインストールしたインスタンスをアンインストールすると、インストールは ISWiAutomationAutomation Interface Version.dll を登録解除します。このため、残された製品でオートメーション インターフェイスを使用するためには ISWiAutomationAutomation Interface Version.dll を手動で登録しなくてはなりません。たとえば、InstallShield をインストールしてから Standalone Build をインストールした後、InstallShield をアンインストールした場合、Standalone Build 用に ISWiAutomationAutomation Interface Version.dll ファイルを手動で登録しなくてはなりません。これを怠った場合、スタンドアロン オートメーション インターフェイスを使用することはできません。

ISWiAutomationAutomation Interface Version.dll を手動で登録するには、System32 フォルダーにインストールされている Regsvr32.exe を使用します。ISWiAutomationAutomation Interface Version.dll ファイルは、InstallShield および Standalone Build Program Files フォルダーの System サブフォルダーにインストールされます。

Microsoft ビルド エンジン (MSBuild)

InstallShield は、.NET Framework に含まれている Microsoft ビルド エンジン (MSBuild) をサポートします。MSBuild サポートを利用して、Visual Studio がインストールされていないビルド ラボ環境で InstallShield プロジェクトと共に Visual Studio ソリューションをビルドすることができます。

概要

MSBuild は、Visual Studio に依存するビルドを削除するように設計された拡張ビルド フレームワークです。.NET Framework は InstallShield Standalone Build に似た役割で機能し、コマンドラインまたは MSBuild の他のホストからプロジェクトまたはソリューションをビルドすることができます。MSBuild についての詳細は、MSDN ライブラリを参照してください。

MSBuild のタスク

MSBuild の柔軟性および拡張性は、タスクと呼ばれる細かくグループ分けされた内部ビルド手順を通して制御されます。MSBuild と共に発送されるタスクの 1 つは Csc と呼ばれ、Visual C# プロジェクトからのコードをコンパイルすることができます。InstallShield は、InstallShield という名前の MSBuild タスクをインストールします。このタスクは、InstallShield プロジェクトと、プロジェクトのデフォルトのビルド ステップを提供するターゲット ファイルをビルドします。このカスタマイズされたタスクとターゲット ファイルによって、MSBuild は InstallShield プロジェクトを Visual Studio ソリューションの一部としてビルドするのに必要なすべてのアクションを実行することができます。

以下のテーブルは、InstallShield タスクのパラメーターの説明です。[プロジェクトの種類] 列には、該当のプロジェクト タイプが表示されます。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
InstallShieldPath	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、InstallShield アプリケーションを含むフォルダーへのパスを指定します。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
プロジェクト	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、プロジェクト ファイル (.ism) の場所を指定します。
ProductConfiguration	文字列	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、リリースの製品構成を指定します。
ReleaseConfiguration	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、リリース名を指定します。 ***メモ:*** *ReleaseConfiguration または PatchConfiguration を指定します。これらのいずれかを指定しなかった場合、または両方を指定した場合、ビルド エラーが発生します。*
PatchConfiguration	文字列	基本の MSI、InstallScript MSI	このパラメーターは、ビルド中の [パッチのデザイン] ビューでのパッチ構成の名前を指定します。 ***メモ:*** *ReleaseConfiguration または PatchConfiguration を指定します。これらのいずれかを指定しなかった場合、または両方を指定した場合、ビルド エラーが発生します。*
OutDir	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、出力先フォルダーとファイルの保存先フォルダーへの完全修飾パスを指定します。
MergeModulePath	String[ ]	基本の MSI、InstallScript、InstallScript MSI	このパラメーターは、プロジェクトで参照するマージ モジュール (.msm ファイル) を含むフォルダーを指定します(複数指定可)。 InstallShield では、マージ モジュールを含むフォルダーを指定するその他の方法も提供されています。詳細については、「マージ モジュールを含むディレクトリを指定する」を参照してください。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
PrerequisitePath	String[ ]	基本の MSI、InstallScript、InstallScript MSI	このパラメーターは、プロジェクトで参照される InstallShield 前提条件ファイル (.prq ファイル) を含むフォルダーをセミコロンで区切って指定します(複数指定可)。 InstallShield では、InstallShield 前提条件ファイル ファイルを含むフォルダーを指定するその他の方法も提供されています。詳細については、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。
ReleaseFlags	String[ ]	基本の MSI、InstallScript MSI	このパラメーターを使用して、リリースに含めるリリース フラグを指定できます。複数のフラグはカンマで区切ります。
PathVariables	ITaskItem[ ]	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターを使用して、プロジェクトのパス変数をオーバーライドすることができます。このパラメーターを使用して、パス変数へのパスを指定したり、PathVariable サブ要素を使用して、パス変数の名前の値を定義することができます。 同じ Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルを参照するには、VSSolutionFolder と呼ばれる定義済みのパス変数を使用します。詳細については、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。 MSBuild ITaskItem[ ] プロパティについての詳細は、MSDN ライブラリを参照してください。
PreprocessorDefines	ITaskItem[ ]	基本の MSI、InstallScript、InstallScript MSI	このパラメーターを使用して、プロジェクトのプリプロセッサ定義を追加またはオーバーライドすることができます。このパラメーターを使用して、プリプロセッサ定義の定義を指定したり、Token サブ要素を使用して、プリプロセッサ定義の名前の値を定義することができます。 MSBuild ITaskItem[ ] プロパティについての詳細は、MSDN ライブラリを参照してください。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
OutputGroups	ITaskItem[ ]	基本の MSI、InstallScript、InstallScript MSI、マージモジュール	このパラメーターは、Visual Studio からのプロジェクト出力グループを指定します。このパラメーターを使用して、プロジェクト出力グループのソースの場所へのパスを指定したり、次のリストにあるサブ要素を使用して、これらの追加の値を定義することができます。 • **Name**—プロジェクトの名前 • **OutputGroup**—プロジェクト出力グループの名前 • **TargetPath**—プロジェクト出力グループのターゲット パス (ソースの場所とは異なります) MSBuild ITaskItem[ ] プロパティについての詳細は、MSDN ライブラリを参照してください。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
ビルド	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	このパラメーターは、実行するビルドのタイプを指定します。次のビルドのタイプから選択することができます。 • 完全—このオプションを選択すると、全リリースがビルドされます。 • コンパイル—このオプションを選択すると、Setup.rul スクリプト ファイルがコンパイルされます。また、基本の MSI およびマージ モジュール プロジェクトの場合、このオプションは ISSetup.dll を .msi パッケージの Binary テーブルにストリームします。 • テーブル—データベース用の Windows Installer テーブルのみをビルドします。このセットアップをまだビルドしていない場合、新しい .msi ファイルが作成されますが、インストールにはファイルが追加されません。インストールをすでにビルドしている場合、すべてのテーブルがビルドされたときに .msi ファイルが更新されますが、ファイルは転送されません。理想的には、このオプションは、インストールのユーザー インタフェースをテストするときに使用します。このオプションは、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで適用します。 • TablesAndFiles—基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合、Windows Installer テーブルをビルドしてファイルを更新します。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワークイメージである場合だけ作動します。 InstallScript プロジェクトの場合、前回のビルドの実行以後に変更になった部分のみを再ビルドします。このパラメーターが使用されない場合、ファイル メディア ライブラリ全体が再ビルドされます。 • UpgradeOnly—このオプションによって、InstallShield の以前のバージョンで作成された基本の MSI プロジェクトをアップグレードすることができるようになります。ビルドはできません。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
BuildCompressed	ブール値	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、リリースを単一ファイルに圧縮するか、または圧縮せずに複数ファイルとして残すかを指定します。
BuildSetupExe	ブール値	基本の MSI、InstallScript MSI、マージ モジュール	基本の MSI および InstallScript MSI プロジェクトの場合、このパラメーターは、インストールと同時に Setup.exe ファイルを作成するかどうかを指定します。 マージ モジュール プロジェクトの場合、このオプションを使って、マージ モジュールをビルドするかどうか、ビルドする場合、それをマージ モジュール フォルダーにコピーするかどうか、または、コピーせずに単にマージ モジュールをビルドするかどうかを指定できます。
ProductVersion	文字列	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、製品バージョンを指定します。これは、製品バージョンのビルド バージョン (3 番目のフィールド) を増加するときに、特に便利です。 有効な製品バージョン番号については、「製品バージョンを指定する」を参照してください。
PropertyOverrides	ITaskItem[ ]	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターを利用すると、Windows Installer プロパティの値をオーバーライドしたり、プロパティが存在しないとき、それを新規作成したりできます。このパラメーターを利用するには、値に新しいプロパティ値があり、かつメタデータ Property がそのプロパティの名前であるアイテムのプロパティー覧を含めます。 このパラメーターは、InstallShield タスクの PropertyOverrides プロパティに InstallShieldPropertyOverrides ItemGroup パススルーとして露出されます。
RunMsiValidator	ITaskItem[ ]	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターを利用すると、ビルドが完了したあと、インストール パッケージおよびマージ モジュールを検証に使用する .cub ファイルを指定できます (複数可)。 検証の詳しい情報については、「プロジェクトの検証」を参照してください。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
RunUpgradeValidation	ブール値	基本の MSI、InstallScript MSI	このパラメーターは、アップグレードの検証を実行するかしないかを指定します。
StopOnFirstError	ブール値	基本の MSI、InstallScript MSI、マージモジュール	このパラメーターは、一番最初のエラーが発生した時点で、ビルドを停止するかどうかを指定します。
MsiVersion	文字列	基本の MSI、InstallScript MSI、マージモジュール	このパラメーターは、ターゲット マシン上でインストールが許可する Windows Installer の最小バージョンを指定します。
DotNetFrameworkVersion	文字列	基本の MSI、InstallScript MSI	このパラメーターは、ターゲット マシン上でインストールが許可する .NET Framework の最小バージョンを指定します。
DotNetUtilPath	文字列	基本の MSI、InstallScript MSI	Regasm.exe と InstallUtilLib.dll は .NET Framework の各バージョンに含まれるユーティリティです。このパラメーターは、.NET インストーラー クラスおよび COM Interop を含むリリースのビルド時に使用する、これらのファイルの 32 ビット バージョンを含むディレクトリのパスを指定します。これは、.NET Framework 再配布可能ファイルへのパスではありません。 ***メモ:*** *InstallShield を 64 ビット システムで使用する場合、*ISCmdBld.exe *はこのパラメーターに指定されたパスに基づいて対応する .NET Framework の 64 ビット バージョンの場所を判別し、適切な場合、*ISCmdBld.exe *は* Regasm.exe *および* InstallUtilLib.dll *の 64 ビットの場所を使用します。*
Disk1Folder	出力文字列	基本の MSI、InstallScript、InstallScript MSI、マージモジュール	このパラメーターは、出力ファイルの場所を指定します。

テーブル 7 • MSBuild InstallShield タスク

パラメーター	種類	プロジェクトの種類	説明
SummaryInfoComments	文字列	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	ビルド番号を含めるなど、ビルド時に **概要情報ストリーム** コメントを設定するには、このパラメーターを使用します。 このプロパティを使って追加されたコメントは、ビルド済みのインストーラーの **[プロパティ]** ダイアログ ボックスで参照することができます。
MSIPackageFileName	文字列	基本の MSI、InstallScript MSI	このパラメーターを使ってビルド時に MSI パッケージ ファイル名を指定します。.msi ファイルに使用するファイル名を指定します (ピリオドまたは拡張子は除きます)。
InstallShieldDigitalCertificatePassword	文字列	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	このパラメーターはオプションで、デジタル証明書パスワードを指定します。指定されなかった場合、ファイルの署名にはプロジェクトで構成されたパスワードが使用されます。

MSBuild スクリプト

MBuild は、1 つのファイル形式 (XML ビルド スクリプト) を理解します。XML ビルド スクリプトは、Visual C# および Visual Basic .NET プロジェクト (.csproj および .vbproj) のプロジェクト ファイル形式として使用されます。MSBuild はまた、ソリューション ファイルおよび Visual C++ プロジェクト ファイル フォーマット (.sln と .vcproj) を処理するための内部フックも備えています。

InstallShield の Visual Studio との統合では、MSBuild と互換性のある XML フォーマット プロジェクト ファイル (.isproj) が使用されています。これにより、MSBuild は、InstallShield プロジェクトを含む Visual Studio ソリューションをシームレスにビルドすることができます。スタンドアロン環境でソリューションをビルドするには、ビルド マシンに InstallShield Standalone Build をインストールします。

.isproj ファイルのカスタマイズ

InstallShield プロジェクトの変更を .isproj ファイルに反映させるには、.isproj ファイルの上部に PropertyGroup 要素または ItemGroup 要素 を追加するか、または既存の PropertyGroup または ItemGroup 要素を更新します。そのあと、InstallShield 関連のパラメーターを必要に応じて追加します。

.isproj ファイルからの、以下のサンプルコードは、次の処理を行う方法を説明します:

- 製品バージョンを設定する。

- 製品名を設定する。
- カスタム パブリック プロパティ MY_PROPERTY を *My Value* に設定する。
- **MyPathVariableName** と呼ばれるパス変数を *C:¥MyPath* の新しい値でオーバーライドします。
- InstallShield 前提条件に以下の検索パス、<ISProductFolder>¥SetupPrerequisites および <ISProjectFolder>¥MyCustomPrerequisites を指定する。

```
<PropertyGroup>
  <InstallShieldProductVersion>1.2.3</InstallShieldProductVersion>
</PropertyGroup>
<ItemGroup>
  <InstallShieldPropertyOverrides Include="新しい製品">
    <Property>ProductName</Property>
  </InstallShieldPropertyOverrides>
  <InstallShieldPropertyOverrides Include="値">
    <Property>MY_PROPERTY</Property>
  </InstallShieldPropertyOverrides>
  <InstallShieldPathVariableOverrides Include="C:¥MyPath">
    <PathVariable>MyPathVariableName</PathVariable>
  </InstallShieldPathVariableOverrides>
  <InstallShieldPrerequisitePath Include="&lt;ISProductFolder&gt;¥SetupPrerequisites"/>
  <InstallShieldPrerequisitePath Include="&lt;ISProjectFolder&gt;¥MyCustomPrerequisites"/>
</ItemGroup>
```

MSBuild を使用して、コマンドラインからリリースをビルドする

メモ: *MSBuild を使って、InstallShield プロジェクトを含む Visual Studio ソリューションをビルドするには、.NET Framework 3.5 以降が必要です。*

MSBuild により、Visual Studio がインストールされていないマシン上で、簡単にコマンドラインからリリースをビルドすることができます。唯一マシンにインストールされている必要のあるコンポーネントは、.NET Framework と InstallShield Standalone Buildです。マシン上に Visual Studio ソリューションのコピーを作成して、MSBuild を実行します。

タスク ***コマンドラインから MSBuild を使用するには、以下の手順に従います。***

1. [コマンドライン プロンプト] ウィンドウを開きます。
2. MSBuild.exe を含むディレクトリへ変更します:

   C:¥Windows¥Microsoft.NET¥Framework¥*Version Folder*¥
3. Visual Studio 統合プロジェクトのリリース ビルドを行うコマンドライン ステートメントを入力します。例:

   MSBuild.exe C:¥*My Visual Studio Solution を含むフォルダー*¥*My Solution*.sln /property:Configuration=Release

リリースの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースを作成すると、デフォルトが設定されます。[リリース] ビューで、すべてのリリース設定を編集することができます。リリース ウィザードでも、リリースの設定を編集することができます。

タスク ***リリース設定を編集するには、次の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を構成するリリースを選択します。設定が、右のペインのタブ上に表示されます。
3. タブの 1 つから設定を選択して、値を変更します。設定についての情報が、[ヘルプ] パネルに、または**[リリース]** ビュー内で F1 を押すと、表示されます。

配布する単一実行可能ファイルの作成

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ビルドを単一の実行可能ファイルとしてパッケージする場合、リリース ウィザードの [一般オプション] パネルにある **[単一の実行可能ファイルを作成する]** オプション、または、**[リリース]** ビューの **"単一 .exeファイル名"** 設定を、次の手順にしたがって使用します:

メモ: *作成した単一実行可能ファイルでは、Setup.exe で使用できるすべてのコマンドライン パラメーターを使用できます。*

タスク ***リリース ウィザードを使用してビルドを単一実行可能ファイルにパッケージするには、以下の手順に従います。***

1. リリース ウィザードを起動します。
2. **[全般オプション]** パネルに移動します。
3. **[単一実行可能ファイルの作成]** チェック ボックスを選択します。

4. デフォルトで、ファイル名 **<プロジェクト名>.exe** が **[ファイル名]** ボックスに入ります。自己展開型ファイルに別の名前を付ける場合は、コンボ ボックスに新しいファイル名を入力するか、ファイル名を定義するパス変数を選択します。

5. [**Icon**] ボックスには、InstallShield がビルド時に Setup.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタンをクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

6. **[概要]** パネルで、**[リリースのビルド]** チェック ボックスを選択し、**[完了]** ボタンをクリックして、**リリース ウィザード**を完成します。

タスク ***[リリース] ビューの設定を使って、ビルドを単一実行可能ファイルとしてパッケージするには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。

2. **[リリース]** エクスプローラーで、単一実行可能ファイルとしてパッケージするリリースを選択します。

3. **Setup.exe** タブを作成します。

4. **"単一 .exe ファイル名"** 設定で、ファイル名または値がファイル名を定義するパス変数 (山かっこで囲みます。例、**<MY EXE FILENAME>**) を入力します。

5. [**Setup.exe アイコン ファイル**] 設定には、InstallShield がビルド時に Setup.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタン (...) をクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

6. リリースをビルドします。

実行可能ファイルにデジタル署名を付加する方法についての詳細は、「デジタル署名とセキュリティ」を参照してください。自己展開型実行可能ファイルを起動するためにエンドユーザーにパスワードの入力を要求する手続きの詳細については、「インストールのパスワード保護」を参照してください。

ディスク イメージにファイルを非圧縮のまま残す

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

タスク ***機能の一部または全部を非圧縮のままディスク イメージに残すには、リリース ウィザードの [メディア レイアウト] パネルを使用します:***

1. **リリース ウィザード**を起動し、**[メディア レイアウト]** パネルに移動します。
2. 以下のいずれかを選択します :
   - すべての機能ファイルを圧縮せずにディスクイメージに配置するには、**[CDROM フォルダー]** オプションを選択します。機能のすべてのファイルが機能の **CD-ROM フォルダー**のプロパティで指定したフォルダーのディスク イメージに配置されます。フォルダーが機能に指定されていない場合、機能のファイルはディスク イメージのルートに配置されます。
   - 一部の機能のファイルを圧縮せずにディスク イメージに配置するには、以下の手順を実行します。
     a. **[カスタム]** を選択して、**[次へ]** をクリックします。**[カスタム メディア レイアウト]** パネルが開きます。
     b. **[キャビネットの機能]** ボックスで、機能の横にあるチェック ボックスを選択または選択解除します。機能のファイルがキャビネット ファイルに格納されている場合、チェック ボックスを選択解除します。機能のファイルが機能の "**CD-ROM フォルダー**" プロパティで指定したフォルダーのディスク イメージに格納されている場合は、チェック ボックスを選択します。フォルダーが指定されていない場合、機能のファイルはディスクイメージのルートに配置されます。

ヒント・ *機能のほとんどをキャビネット ファイルに格納する場合は、**[すべてクリア]** をクリックしてから、ディスク イメージに非圧縮のまま残す機能を選択します。反対に、機能のほとんどを非圧縮でディスク イメージに残す場合は、**[すべて選択]** ボタンをクリックしてから、キャビネット ファイルに格納する機能を選択解除します。*

3. **リリース ウィザード**を完了します。

メモ・ *機能のコンポーネントの "圧縮" 設定で [はい]、[いいえ] のどちらを選択しても、機能に非圧縮 (CD-ROM フォルダー) オプションを選択すると、機能のファイルは配布メディアで圧縮されません。*

Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield では、Windows Vista 以降のプラットフォーム上でインストール (セットアップ ランチャー、InstallShield 前提条件、.msi ファイル、および任意のアドバンスト UI またはスイート/アドバンスト UI パッケージ) を実行するために、インストールの Setup.exe ファイルが必要とする最低実行レベルを指定することができます。プロジェクトの個別のリリースに対して設定することが可能です。

タスク ***リリースの必要実行レベルを指定するには、次の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"必要実行レベル"** 設定に、適切なオプションを選択します。

選択可能なオプションは、以下のとおりです。

- **管理者** – Setup.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。非管理者は、管理者としての認証が必要になります。
- **最高権限** – Setup.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を同意(コンセント)する必要があります。非管理者は、管理者権限を持たずに Setup.exe を実行します。これは、InstallScript プロジェクトと InstallScript MSI プロジェクトのデフォルト オプションです。
- **起動者** – Setup.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Setup.exe を実行することができます。Setup.exe は、資格情報または同意 (コンセント) を求める UAC メッセージを表示しません。これは、アドバンスト UI、基本の MSI、およびスイート/アドバンスト UI プロジェクトのデフォルト オプションです。

アドバンスト UI、InstallScript、InstallScript MSI、スイート/アドバンスト UI プロジェクト、および"セットアップ ランチャー" 設定が [はい] に設定されている基本の MSI プロジェクトでは、InstallShield は Windows アプリケーション マニフェストをリソースとして Setup.exe ランチャーに埋め込みます。このマニフェストは選択された実行レベルを指定します。Windows Vista よりも古いバージョンのオペレーティング システムでは、必要実行レベルは適用されません。実行レベルは、マニフェストで次のように定義されています:

```
<requestedExecutionLevel
  level="asInvoker"
  uiAccess="false"/>
```

レベル属性の他の有効な値は、**highestAvailable** と **requireAdministrator** です。

基本の MSI プロジェクトで、"セットアップ ランチャー" 設定が [いいえ] に設定されている場合、InstallShield は Windows アプリケーション マニフェストを Setup.exe ランチャーに埋め込みません。

基本の MSI プロジェクトで必要実行レベルを昇格することの利点は、Setup.exe を実行するための権限の昇格が、必要な場合 1 度で済むという点です。またこれらの権限はインストールに含まれるすべての InstallShield 前提条件および .msi パッケージの [実行] シーケンスにも適用できるため、承認を得るためのプロンプトを複数回にわたって行う必要がありません。たとえば、すべての InstallShield 前提条件のうち 2 つが管理者権限を必要とする場合、この設定を [管理者] に変更します。そうすることにより、インストール中、Windows Installer が Setup.exe ファイルを実行する前、プロンプトはエンドユーザーに対して一度のみ表示されます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにも似たような利点があり、Setup.exe を実行するための権限の昇格が、必要な場合 1 度で済みます。またこれらの権限はアドバンスト UI またはスイート/アドバンスト UI パッケージのすべてにも適用できるため、承認を得るためのプロンプトを複数回にわたって行う必要がありません。

ただし、権限を昇格して、インストールの終わりでもアプリケーションを起動する場合、この昇格された権限はアプリケーションに適用されますので注意してください。ほとんどの場合、Windows Vista 以降では、昇格された権限を使用したアプリケーションの実行は推奨されていません。

重要: *InstallShield は昇格された権限で実行されます。InstallShield 内からインストールを起動すると、昇格された権限がインストールに引き継がれるため、インストールは自動的に昇格された権限を持ちます。Windows Vista 以降を使用するエンド ユーザーに対して表示される動作を反映しない可能性があります。したがって、開発システムに Windows Vista 以降を使用する場合、InstallShield 内からではなく、リリース フォルダーを開いてインストールを直接起動することを考慮してください。*

リリース ファルダに素早くアクセスしてリリースを直接起動するには、標準ツールバーまたは [ツール] メニューの [リリースフォルダーを開く] をクリックします。

エンド ユーザーのインストール エクスペリエンスは、インストールが必要とする権限によってのみ実行された場合、安全性がより一層向上します。アプリケーションは、システム管理者のみによる実行が必須の場合を除き、最も低い権限で実行されることが理想的です。

InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

構成方法によって、InstallShield 前提条件を含むインストールがインストール中のいくつかの時点で、Windows Vista 以降のシステム上で昇格された権限のユーザー アカウント制御 (UAC) プロンプトを表示することができます:

1. エンド ユーザーが Setup.exe ファイルを起動するとき
2. Setup.exe ファイルが、昇格された権限を必要とするセットアップ前提条件を起動するとき
3. Setup.exe ファイルが、昇格された権限を必要とする機能前提条件を起動するとき
4. Windows Installer が .msi パッケージの [実行] シーケンスを開始するとき

[リリース] ビューでインストールの "必要実行レベル" 設定で [管理者] を選択すると、通常 Windows Vista 以降は UAC プロンプトを一度だけ表示します。これは、エンド ユーザーが Setup.exe ファイルを起動したときに表示されます。ただし、この設定で [起動者] を選択した場合、Windows Vista 以降はインストール中に複数回にわたって UAC プロンプトを表示する可能性があります。たとえば、Windows Vista 以降は昇格された権限を必要とするセットアップ前提条件に UAC プロンプトを表示してから、.msi パッケージの [実行] シーケンスで別の UAC プロンプトを表示することがあります。作成中のインストールがこのシナリオに該当する場合、インストールをアドバタイズしてから .msi パッケージを実行し、エンドユーザーが .msi パッケージの UAC プロンプトを避けることができるようにした方がよい場合があります。作成中のインストールがこのシナリオに該当しない場合、.msi パッケージのアドバタイズを避けることが推奨されます。アドバタイズを行うと、UAC プロンプトが回避されなくなるためです。

ヒント: *[一般情報] ビューの "管理者権限が必要" 設定で、.msi パッケージが管理者権限を必要とするかどうかを指定します。InstallShield 前提条件エディターの [動作] タブでは、前提条件が管理者権限を必要とするかどうかを指定します。Windows Vista における UAC プロンプトの表示に関連する他の InstallShield 設定については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。*

タスク ***.msi パッケージをアドバタイズするかどうかを指定するには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"前提条件が昇格必要時のアドバタイズ"** 設定で、適切なオプションを選択します。

選択可能なオプションは、以下のとおりです。

- **アドバタイズ: サイレント** – インストールのセットアップ前提条件が、昇格された権限で正常にインストールされたとき、.msi パッケージがアドバタイズされ、サイレント (ユーザー インターフェイスがない状態) で実行されます。これにより、資格情報または同意を求める追加の UAC プロンプトが、メイン インストールの [実行] シーケンスで不要となります。
- **アドバタイズ: 完全 UI** – インストールのセットアップ前提条件が、昇格された権限で正常にインストールされた場合、.msi パッケージがアドバタイズされ、完全ユーザー インターフェイスと共に実行されます。これにより、メイン インストールの [実行] シーケンスで追加の UAC プロンプトが不要となります。
- **アドバタイズなし**—.msi パッケージはアドバタイズされません。エンド ユーザーがインストールを実行したとき、1 つまたは複数の UAC プロンプトを表示して、セットアップ前提条件をインストールできます。.msi パッケージの [実行] シーケンスにも昇格が必要な場合は、Windows Installer が [実行] シーケンスを開始する前に追加 UAC プロンプトを表示できます。

重要: *これらのアドバタイズ オプションが動作するためには、パッケージでアドバタイズがサポートされている必要があります。アドバタイズは瞬間的ではないので、インストールに多少の遅延が生じます。また、アドバタイズの後、インストールの主要部分が完了する前にエンドユーザーが [キャンセル] をクリックすると、予期しない動作が発生することがあります。たとえば、製品のアドバタイズ ショートカットが、メインのインストールが始まる前にデスクトップに表示されてしまうことがあります。また、この時、混乱したユーザーがメインのインストールをキャンセルすると、パッケージがアドバタイズされたまま、完全にインストールされないということも場合によって起きます。したがって、状況によっては、この設定を [アドバタイズなし] のままにすることで、追加の UAC プロンプトを許可し、製品のアドバタイズを避けるた方が安全な場合もあります。*

一般的に目標とされているのは、インストールが UAC プロンプトを一回のみ表示することです。"前提条件が昇格必要時のアドバタイズ" 設定のアドバタイズ オプションは、この目的を容易に実現できるように設けられていますが、すべての場合においてこの目的の実現を保証するものではありません。たとえば、インストールにより再起動が実行された場合、インストールのプロセスは、再起動後、常に制限された権限に戻ります。昇格がセットアップ前提条件、機能前提条件、または .msi パッケージのいずれに必要であるかにかかわらず、その後の権限の昇格により、追加の UAC プロンプトが表示されることもあります。再起動が最後の前提条件と .msi ファイルの間に来た場合、.msi パッケージはアドバタイズされません。次は、この設定でいずれかのアドバタイズ オプションを選択した際に発生する可能性がある異なる結果の例です。

- 前提条件は昇格を必要とし、ターゲット マシンで正常にインストールされる。製品はアドバタイズされ、インストールされる。製品インストールは、UAC プロンプトを表示しない。
- インストールの前提条件の 1 つが再起動を必要とする可能性がある。前提条件の再起動が実行されたあと、追加の前提条件はインストールされず、インストールされた他の前提条件も昇格を必要としない。このシナリオでインストールされた最後の前提条件は、昇格された前提条件の単純な成功例とは異なるため、.msi パッケージのアドバタイズは実行されない。
- インストールの前提条件は、すべてターゲット マシンに存在するためインストールをする必要がない。したがって、.msi パッケージのアドバタイズは実行されない。
- インストールの前提条件は、すべて昇格を必要としない。したがって、.msi パッケージのアドバタイズは実行されない。
- 昇格された前提条件は正常にインストールされる。ただし、この場合、パッケージは既にインストールされている製品のマイナー アップグレード。つまり、パッケージの製品コードが、既にターゲット マシンに存在する製品の製品コードと一致している。.msi パッケージのアドバタイズは実行されない。UAC プロンプトは、適切なデジタル証明書が以前のインストールおよびパッチに含まれているかどうかによって、表示される可能性もあるし、表示されない可能性もある。

インストールのパスワード保護

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

セキュリティをさらに向上させるため、インストール パッケージをパスワードで保護することができます。インストールをパスワードで保護すると、パッケージをインストールするエンドユーザーは、インストールを開くために大文字小文字の区別があるパスワードを入力する必要があります。

リリース ウィザードの [パスワードと著作権] パネルでパスワード保護をアクティベートできます。

リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield では、インストールに含まれている InstallShield 前提条件のランタイムの場所を指定できます。

タスク ***リリースの InstallShield 前提条件の配置場所を指定するには、以下の手順に従います。***

- **リリース ウィザード**の**InstallShield [前提条件]** パネル を使用します。
- **[リリース]** ビューでリリースを選択します。次いで、**[Setup.exe]** タブの **"InstallShield 前提条件の場所"** 設定で、適切なオプションを選択します。

タスク ***各 InstallShield 前提条件に対して異なる場所を指定するには、次の手順に従います:***

1. **[再配布可能ファイル]** ビュー (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合)、または **[前提条件]** ビュー (InstallScript プロジェクトの場合) で、各 InstallShield 前提条件の適切な場所を指定します。詳細については、「特定の InstallShield 前提条件の実行時の場所を指定する」を参照してください。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. ビルドするリリースを選択します。
4. **Setup.exe** タブをクリックします。
5. **InstallShield"前提条件の場所"** 設定で、**[個々の選択に従う]** を選択します。

ヒント: *または、リリース ウィザードの [InstallShield 前提条件] パネルで、[個々の選択に従う] オプションを選択することもできます。*

InstallShield 前提条件が別の前提条件の依存ファイルとしてプロジェクトに追加される場合、前提条件依存ファイルの場所は、それを必要とする前提条件の場所設定に従います。

InstallShield 前提条件を含むリリースのビルドで次の両方が当てはまるとき、1 つまたは複数のビルド エラーが発生する可能性があります。

- 前提条件が、Setup.exe から抽出されるとき、または (エンド ユーザーのコンピューターに Web からダウンロードされる代わりに) ソース メディアからコピーされるときの InstallShield 前提条件の場所を指定してください。
- 前提条件ファイルはコンピューターにはありません。

ビルドエラーを除去するには、プロジェクトから InstallShield 前提条件を削除するか、インターネットからコンピューターに InstallShield 前提条件をダウンロードするか、またはリリースの InstallShield 前提条件の場所をダウンロードオプションに変更してから、リリースを再ビルドします。

リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれているパッケージのランタイムの場所を指定できます。

ヒント: *アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート セットアップ ランチャーの構成を行う場合、パッケージのランタイムの場所は、セットアップ ランチャーから抽出するか、Web からダウンロードします。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。*

詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

タスク ***リリースのパッケージの配置場所を指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. ビルドするリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"パッケージの場所"** 設定で、適切なオプションを選択します。

タスク ***各パッケージに対して異なる場所を指定するには、次の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. それぞれの InstallShield 前提条件に対して、適切な場所を指定します。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する」を参照してください。
3. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
4. ビルドするリリースを選択します。
5. **Setup.exe** タブをクリックします。
6. **"パッケージの場所"** 設定で、**[個々の選択に従う]** を選択します。

パッケージが別のパッケージの依存関係としてプロジェクトに追加される場合、パッケージの依存関係ファイルの場所は、それを必要とするパッケージの場所設定に従います。

パッケージを含むリリースのビルドで次の両方が当てはまるとき、1 つまたは複数のビルド エラーが発生する可能性があります。

- パッケージの場所で、パッケージが Setup.exe から抽出されるか、または (エンド ユーザーのコンピューターに Web からダウンロードされる代わりに) ソース メディアからコピーされるかを指定します。

- パッケージ ファイルは、使用中のコンピューター上にはありません。

ビルドエラーをなくすには、プロジェクトからパッケージを削除してコンピューターに追加するか、またはリリースのパッケージの場所をダウンロード オプションに変更してから、リリースを再ビルドします。

スイート でパッケージのアンインストール順を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[リリース] ビューの **[Setup.exe]** タブにある **"アンインストール順"** プロパティを使って、スイート プロジェクトに含まれるパッケージのアンインストール順を指定することができます。

タスク ***スイート プロジェクトでパッケージのアンインストール順を指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. ビルドするリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. **"アンインストール順"** 設定で次のオプションから 1 つを選択します:
   - **パッケージ順と同じ** – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
   - **パッケージ順と逆** – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

オートメーション インターフェイスにおける UninstallOrder メソッド

オートメーション インターフェイスで UninstallOrder メソッドを使って **"アンインストール順"** プロパティを設定することもできます。次の値のうち 1 つを指定します。

- euoForward(0) – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。
- euoReverse(1) – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

デジタル署名とセキュリティ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールとアプリケーション ファイルにデジタル署名をすることで、インストールやアプリケーション内のコードが、発行時以来、改ざんまたは変更されていないことをエンドユーザーに保証することができます。

[署名] タブを使って、InstallShield がファイルに署名するときに使用するデジタル署名に関する情報 (証明機関より付与されたデジタル証明書ファイルを含む) を指定します。

[署名] タブでまた、ビルド時にデジタル署名をするインストール内のファイルを指定することもできます。InstallShield は、作業中のプロジェクトの種類に応じて、リリースに含まれる次の任意およびすべてのファイルに署名することができます。

- 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの Windows Installer パッケージ (.msi ファイル)
- マージ モジュール プロジェクトのマージ モジュール パッケージ (.msm ファイル)
- アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI プロジェクトの Setup.exe ファイル
- InstallScript プロジェクトのメディア ヘッダー ファイル
- リリースの任意のファイル (アプリケーション ファイルを含む)

Windows ロゴ: *インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

[署名] タブにある設定に関する詳しい情報は、「リリースの [署名] タブ」を参照してください。

証明機関

証明機関とは、デジタル証明書 (デジタル ID とも呼ばれます) を発行、管理する VeriSign のような組織です。証明機関は、指定された基準に基づいて要求側の ID を検証し、デジタル証明書を発行します。デジタル証明書を取得するには、証明機関に会社と製品に関する特定の情報を提供する必要があります。

証明機関の一覧については、MSDN Web サイトの「Microsoft Root Certificate Program Members」を参照してください。

SHA-1 と SHA-256 証明書の違い

InstallShield では、インストールおよびファイルをビルド時に署名する際、HA-256 または SHA-1 ハッシュ アルゴリズムを使ったデジタル証明書を使用できます。

SHA-1 はセキュリティの脆弱性があるため、SHA-256 の使用が推奨されます。Microsoft は、Windows では 2016 年 1 月以降に SHA-1 証明書を使って署名およびタイムスタンプが追加されているアイテムを信頼しないことを発表しました。さらに、証明書を発行する組織である証明機関では、SHA-1 証明書が段階的に廃止されます。したがって、InstallShield プロジェクトに含まれる任意の SHA-1 証明書は、SHA-256 証明書と差し替えることが推奨されます。最新情報および特定の詳細については、証明機関にお問い合わせください。

プロジェクトで、SHA-256 証明書を使った署名が構成されている場合、InstallShield はビルド時に署名を行うファイルの署名に SHA-256 ハッシュを使用します。プロジェクトで SHA-1 証明書を使った署名が構成されている場合、InstallShield は SHA-1 ハッシュを使用します。SHA-1 証明書を使用すると、SHA-1 の使用についてアラートするビルド警告 -7346 が発生します。

InstallShield 2016 から、Windows Installer および InstallScript プロジェクトの SHA-256 デジタル証明書サポートが次のように強化されています:

- ダイジェスト タイプを指定できる機能
- Settings.xml で SHA-256 タイムスタンプ サーバーを指定することができます。
- 証明書ストアにある類似した名前の証明書も処理します

重要: *2016 年 1 月以降に作成またはタイムスタンプが付けられたすべての新しい署名は、SHA-256 に基づく必要があります。SHA-1 証明書を使って署名されているすべてのファイルを継続してサポートするためには、2016 年 1 月以前の日時を使ったタイムスタンプを含める必要があります。これらのファイルは、すべての現在のバージョンの Windowsですべての SHA-1 サポートが停止される 2020 年 1 月 14 日まで、MOTW (Mark of the web) システムを使って引き続き使用することができます。*

証明書ファイルまたは証明書ストアにある証明書を使ってデジタル署名を生成する

ファイルおよびインストールの署名に使用するデジタル署名情報を指定するとき、InstallShield では次のオプションから選択できます:

- 使用中のマシンにある .pfx 証明書ファイルを指定できます。
- 証明書を含む証明書ストアを参照できます。

オプション 1 ー.pfx ファイル

.pfx (Personal Information Exchange) ファイルを使って、インストールおよびアプリケーションにデジタル署名を行います。次のツールを利用して、.pvk ファイルと .spc ファイルから .pfx ファイルを作成することができます:

- PVK2PFX.exe—Windows Platform SDK の一部で、Microsoft Visual Studio 2005 にも含まれています。
- pvkimprt.exe—この PVK Digital Certificate Files Importer ツールは、Microsoft Web サイト (http://www.microsoft.com/downloads/details.aspx?FamilyID=F9992C94-B129-46BC-B240-414BDFF679A7&displaylang=EN) からダウンロードすることができます。

.pfx ファイルは通常、パスワードに関連付けられています。

オプション 2 ー 証明書ストアにある証明書

証明書ストアにデジタル証明書を格納する場合、使用する証明書を含む証明書ストアをプロジェクトで参照することができます。この方法で署名を行う場合、ストア名 (Personal、Trusted Root Certification Authorities、Enterprise Trust、Intermediate Certification Authorities)、ストアの場所 (ユーザー、マシン)、および特定の証明書を識別するためのサブジェクトといった情報を指定する必要があります。

ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。

デジタル署名にタイムスタンプを追加する

リリースのデジタル署名情報を指定すると、InstallShield はデフォルトで、ビルド時にデジタル署名にタイムスタンプを追加します。デジタル証明書のタイムスタンプは、ファイルが署名された日時を記録します。これによって、署名が行われた時点で証明書の期限が切れていないことを証明することができます。信頼されたタイムスタンプ サーバーからのタイムスタンプは、一般的に証明書の有効期限よりも長く、デジタル署名を有効な状態に保ちます。

InstallShield が使用するデフォルトのタイムスタンプ サーバーを変更する方法、またはタイムスタンプ機能を無効にする方法については、「デジタル署名のタイムスタンプ サーバーを変更する」を参照してください。

ビルド時にリリースとそのファイルにデジタル署名を行う

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

では、リリースに構成できるデジタル署名の設定があります。ビルド時に、InstallShield は構成した設定を使用して、インストール パッケージ、Setup.exe ファイル、および、リリースにある定義した基準を満たす他のすべてのファイルに署名します。

タスク ***リリースとそのファイルのデジタル署名を構成するには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、署名するリリースをクリックします。
3. **[署名]** タブをクリックします。
4. 次の設定を適切に構成します。
   - **署名の種類** – ビルドが生成したファイルにデジタル署名を行う方法を指定します。**[カスタム]** オプションを選択すると、**"パス"** および **"引数"** 設定が有効になります。
   - **パス** – 使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。
   - **引数** – 署名ツール構成のコマンドライン引数を指定します。

- **証明書 URL** – 完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。
- **デジタル証明書情報** – この設定の省略記号ボタン (...) をクリックします。[証明書の選択] ダイアログボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。
- **証明書パスワード**–ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。
- **出力ファイルに署名する** — 署名を行うファイルを指定します (Setup.exe、.msi パッケージ、その両方、またはどちらにも署名しない)。**パッケージ内のファイルに署名する**
- **64 ビット署名を使用** – 64 ビット署名フレームワークを使用してパッケージに署名を行うかどうかを指定します。
- **署名の説明** – "署名出力ファイル" 設定で指定されたファイルに使用する署名の説明を指定します。
- **パッケージのファイルに署名** – **[サポートファイル]** ビューで構成された InstallShield サポートファイル (英語、言語非依存、および詳細ファイル) を含む、インストール プロセス中にのみ使用されるすべてのファイルにデジタル署名するかどうかを指定します。

  [はい] オプションを選択する場合、***"パッケージ内のファイルに署名する"*** 設定の下にある他の設定を使って、署名を行うファイルとファイル パターンおよび署名を行わない項目を指定します。

  ***メモ・** 署名から除外するように指定されたファイルとファイル パターンは、署名するように設定されたものよりも優先されます。たとえば、**"含める"** 設定および **"除外する"** 設定に ***.exe** を指定すると、InstallShield は .exe ファイルに署名を行いません。*

***ヒント・** [署名] タブにある設定に関する詳しい情報は、「リリースの [署名] タブ」を参照してください。*

ビルド時に、InstallShield は [署名] タブで指定されたファイルに署名を行います。リリースが、マージ モジュールを含むインストール用の場合、ファイルは、マージ モジュールがマージされる前に署名されます。

コマンドラインからビルドされたリリースにデジタル署名を行う

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

ビルド時に InstallShield でアプリケーションに署名を行う代わりに、iSign アプリケーション (iSign.exe) を使用して、InstallScript プロジェクトのリリースをビルドした後、コマンドラインからそのリリースにデジタル署名を行うことができます。

iSign.exe は、次のディレクトリに保存されています:

InstallShield Program Files フォルダー¥System

iSign.exe の構文および利用可能なコマンドライン パラメーターについては、「iSign.exe」を参照してください。

リリース フラグ

プロジェクト: *次のプロジェクト タイプは、リリース フラグのサポートを含みます:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリースをビルドする際、リリースのタイプによって機能、InstallShield 前提条件、および連鎖 .msi パッケージを含めることも除外することもできます。たとえば、製品のトライアル版を作成していて、ビルドにすべての機能を含めない場合、機能にフラグを付けて、これらの機能をリリースの 製品構成 の下でこれらを指定することができます。

タスク ***リリース フラグに基づいて機能、InstallShield 前提条件、および連鎖 .msi パッケージをフィルターするには、以下の手順に従います。***

1. リリース フラグを機能、InstallShield 前提条件、および連鎖 .msi パッケージに割り当てます。

   この方法についての詳細は、「リリース フラグを機能に割り当てる」、「リリース フラグを InstallShield 前提条件に割り当てる」、または「リリース フラグを連鎖 .msi パッケージに割り当てる」を参照してください。

2. 適切な場合にリリースに含めるフラグを指定します。

トライアル版と完全バージョンのシナリオを比較すると、リリース フラグを使用する理由がよく分かります。次のテーブルでは、4 つの異なる機能の例を示します。どの機能が製品で使用できるかは、エンドユーザーが入手するバージョンによって決まります。

この例で、完全バージョンにはアプリケーションの実行可能ファイル、ヘルプ ファイル、スペルチェッカー、およびアドオン パックが連鎖 .msi パッケージとして含まれています。しかし、トライアル版にはアドオン パックは含まれませんが、アップグレード パックが含まれます。実質的に 1 つの製品であるものに対し、異なるインストール プロジェクトを 2 つ作成する必要はなく、特定のバージョンに固有の機能にフラグを付けることができます。たとえば、アドオン機能は **[完全]** としてフラグされ、アップグレード機能は **[トライアル]** としてフラグされます。

テーブル 8 ・ サンプル リリース フラグ

機能、InstallShield 前提条件、または連鎖 .msi パッケージ	バージョン	リリース フラグ
Windows Installer 4.5 再配布可能ファイル (InstallShield 前提条件)	すべてのバージョン	なし
Program_Executable (機能)	すべてのバージョン	なし
Help_Files (機能)	すべてのバージョン	なし

テーブル 8 • サンプル リリース フラグ (続き.)

機能、InstallShield 前提条件、または連鎖 .msi パッケージ	バージョン	リリース フラグ
Spellchecker (機能)	完全バージョン	完全
Add_Ons (連鎖 .msi パッケージ)	完全バージョン	完全
Upgrade_Pack (機能)	トライアル バージョン	Trial

リリースのビルドにリリース ウィザードを使用した場合、リリースにフラグの付いた機能、InstallShield 前提条件、および連鎖 .msi パッケージを含めるかどうかを選択できるオプションがあります。デフォルトでは、リリースにはすべての機能、InstallShield 前提条件、および連鎖 .msi パッケージが含まれます。リリース ウィザードの [フィルターの設定] パネル内でリリース フラグに **Full** を指定した場合、フラグがつかないすべての機能および InstallShield 前提条件と一緒に **Spellchecker** 機能と **Add_Ons** 連鎖 .msi パッケージのみがリリース リリースにビルドされます。アップグレード パックは、インストールに含まれません。

ヒント: *デフォルトでは、すべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースに含まれます。[リリース] ビューまたはリリース ウィザードのどちらかでリリース フラグを指定すると、フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージ、並びに指定されたフラグが付いた機能、InstallShield 前提条件、および連鎖 .msi パッケージがインストールに含まれます。[リリース] ビューまたはリリース ウィザードで指定されていないリリース フラグがついた機能、InstallShield 前提条件、または連鎖 .msi パッケージはリリースに含まれません。*

製品構成フラグとリリース フラグの違い

プロジェクト: *以下のプロジェクト タイプには、製品構成フラグとリリースフラグのサポートが含まれています:*

- *基本の MSI*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールに含めるフラグをリリース (*「リリース フラグ」*) と製品構成 (*「製品構成フラグ」*) の両方について指定できます。リリース フラグは、リリースの設定とリリース ウィザードの [フィルターの設定] パネルの両方で表示されますが、製品構成フラグは [リリース] ビューの製品構成の [全般] タブでのみ編集が可能です。

製品構成フラグは、リリース フラグを補足します。つまり、製品に対して指定するすべての機能フラグは、ビルド中の各リリースのフラグと組み合わせられ、指定されたフラグが付いたすべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースにビルドされます。製品構成レベルでは、どのフラグが含まれているかが確認できないので、このウィザードでフラグを指定するときは注意してください。

製品構成とリリースフラグとの関係を分かりやすく説明するため、以下の機能、InstallShield 前提条件、および連鎖 .msi パッケージを含むプロジェクトを想定します。

テーブル 9 - サンプル機能、InstallShield 前提条件、および連鎖 .msi パッケージ

機能、InstallShield 前提条件、または連鎖 .msi パッケージ	バージョン	リリース フラグ
Windows Installer 4.5 再配布可能ファイル (InstallShield 前提条件)	すべてのバージョン	なし
Program_Executable (機能)	すべてのバージョン	なし
Help_Files (機能)	すべてのバージョン	なし
Spellchecker (機能)	完全バージョン	完全
Add_Ons (連鎖 .msi パッケージ)	完全バージョン	完全
Upgrade_Pack (機能)	トライアル バージョン	Trial

次の表は、製品構成フラグとリリース フラグの組み合わせによって、どの機能、InstallShield 前提条件、および連鎖 .msi パッケージがインストールに含まれるかを説明します。

テーブル 10 - 製品構成フラグとリリース フラグを組み合わせる

製品構成フラグ	リリース フラグ	インストールに含まれる機能、InstallShield 前提条件、および連鎖 .msi パッケージ
<なし>	<なし>	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Spellchecker、Add_Ons、Upgrade_Pack
<なし>	完全	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Spellchecker、Add_Ons
Trial	<なし>	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Upgrade_Pack
完全	Trial	Windows Installer 4.5 再配布可能ファイル、Program_Executable、Help_Files、Spellchecker、Add_Ons、Upgrade_Pack

リリース フラグに基づくフィルター

プロジェクト: *以下のプロジェクト タイプには、製品構成フラグとリリースフラグのサポートが含まれています:*

- *基本の MSI*

- *InstallScript MSI*
- *スイート/アドバンスト UI*

リリース フラグを機能および InstallShield 前提条件に割り当てると、割り当てたフラグに基づいてこれらを含めるリリースを作成することができます。デフォルトでは、すべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースに含まれます。[リリース] ビューまたはリリース ウィザードのどちらかでフラグを指定すると、フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージ、並びに指定されたフラグが付いた機能、InstallShield 前提条件、および連鎖 .msi パッケージがインストールに含まれます。フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージのみを含めるには、存在しないフラグを指定します。たとえば、**NoFlags** の様に指定します。こうすると、フラグが付いていない機能、InstallShield 前提条件、および連鎖 .msi パッケージだけがリリースに組み込まれます。

サブ機能を含めることはできますが、この親機能を含めることはできません。このような場合、サブ機能はリリースに最上位の機能として組み込まれ、この親はリリースからは除外されます。

リリース ウィザードまたは [リリース] ビューで、機能、InstallShield 前提条件、および連鎖 .msi パッケージをフィルターすることができます。この 2 つの方法を、以下に説明します。

リリース ウィザードでのフィルター

リリースの最も簡単な作成方法は、リリース ウィザードを使用することです。ウィザードの [フィルターの設定] パネルで、リリースに含めるフラグを指定できます。

[リリース] ビューでのフィルター

[リリース] ビューを使用して、含めるフラグを指定することもできます。フラグを製品構成レベルとリリースの両方で、フラグを含めることができます。詳細については、「製品構成フラグとリリース フラグの違い」を参照してください。

タスク ***製品構成フラグを指定するには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、変更する製品構成をクリックします。設定が、右のペインにある **[全般]** タブで表示されます。
3. ***"製品設定フラグ"*** 設定に、含めるフラグを入力します。複数のフラグを含める場合は、それぞれカンマで区切ってください。

タスク ***リリース フラグを指定するには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、変更するリリースをクリックします。設定が、右のペインのタブ上に表示されます。
3. "リリース フラグ" 設定に、このリリースに含めるフラグを入力します。複数のフラグを含める場合は、それぞれカンマで区切ってください。

1 つのプロジェクトから複数の製品構成のインストールを作成する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

評価版リリースと完全リリースなど、2 つ以上の構成の製品をリリースしたい場合があります。単一プロジェクト内で、異なるプロジェクト機能のサブセットを持つリリースをビルドできます。プリプロセッサ命令を使用した単一のインストール スクリプトがあると、異なるランタイム動作をするリリースをビルドすることができます。

タスク ***ビルド時にリリースの含める機能を判別するには、以下の手順に従います。***

1. **リリース ウィザード**を起動し、**[機能]** パネルに移動します。
2. **[以下に含める機能を指定]** オプションを選選択します。
3. **[機能]** エクスプローラーで、製品構成に含める機能のチェック ボックスを選択し、含めない機能のチェック ボックスを選択解除します。
4. **リリース ウィザード**の他のパネルを完成します。

タスク ***プリプロセッサ命令を使ってリリースのランタイム動作を決めるには、以下の手順に従います。***

1. スクリプトで、プリプロセッサ命令を使用して異なる製品構成に異なるコードを実行します。例 :

```
#ifdef EVAL_RELEASE
   /*評価固有コード */
#elif FULL_RELEASE
   /*完全リリース固有コード */
#endif
```

2. [**コンパイラ プリプロセッサ定義**] ボックス (**リリース ウィザード**の [**一般オプション**] パネル) または、"**コンパイラ プリプロセッサ定義**" 設定 ([**リリース**] ビューの [**ビルド**] タブ) は、リリースに実行させるコードに対応するプリプロセッサ定数を指定します。
3. リリースをビルドします。

標準 Web インストールと One-Click Install の違い

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

アプリケーションをインターネットからダウンロードする際、多くのエンドユーザーはダウンロード方法やアプリケーションを正しく実行する方法を説明した、長く複雑な説明書に直面します。標準的なダウンロードでは 10 以上の手順が必要となります。この作業は時間がかかる上に、必ずしも成功するとは限りません。アプリケーションのダウンロードに失敗したエンドユーザーは製品サポートのヘルプに頼らなければならないか、完全にあきらめて二度とアプリケーションにアクセスしなくなるでしょう。これは顧客にとっても、開発チームにとっても時間を無駄にする結果となります。

そこで、One-Click Install セットアップを使用すると、エンドユーザーはアプリケーションを必要最小限の操作でダウンロードすることができ、すぐにアプリケーションを開始することが可能です。以下に示す標準セットアップと One-Click Install セットアップのインストール手順を比べてみてください。

標準インターネット インストールの手順

標準的なインターネット インストールは以下のように行われます。

1. エンドユーザーが Web ページにアクセスし、アプリケーションをダウンロードするためのリンクをクリックします。
2. ダイアログが開きます。ソフトウェアの使用許諾契約に同意することを求めるダイアログが表示されます。
3. ここでエンド ユーザーは、インストール実行可能ファイルをディスクに保存することができます。エンド ユーザーは、インストールの実行ファイルに後でアクセスするため、この場所を覚えておく必要があります。
4. ファイル転送が開始されます。
5. エンドユーザーは、インストール実行可能ファイルを検索して、それを起動します。
6. インストール実行可能ファイルが展開、および実行されます。
7. アプリケーションを適切に実行するために他のリソースが必要な場合、エンドユーザーはこのときにダウンロードする必要があります。
8. インストールが実行され、ユーザーは案内に沿ってインストールを開始します。
9. エンドユーザーは、アプリケーション実行可能ファイルのディスクへの場所など、いくつか選択を行います。
10. アプリケーション ファイルの転送が開始されます。
11. エンドユーザーは、ディスク上のアプリケーション実行可能ファイルの位置を特定し、アプリケーションを起動します。

One-Click Install インストールの手順

One-Click Install セットアップを使用すると、エンドユーザーはインストールを素早く簡単に行うことができます。One-Click Install セットアップは以下のように行われます。

1. エンドユーザーが Web ページにアクセスし、[ダウンロード] ボタンをクリックします。
2. ダイアログが開きます。ダイアログにファイル転送の進行状況が表示されます。インストールがダウンロードされると、すぐにインストールが実行されます。
3. エンドユーザーは、アプリケーションの使用を開始します。

One-Click Install セットアップでは、エンドユーザーが実行可能ファイルを探す必要がなく Web ページからの移動は不要です。One-Click Install セットアップにより、エンドユーザーはダウンロードに苦労することなくアプリケーションの使用に専念することができます。

InstallScript プロジェクトの One-Click Install インストール

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install インストールには、適切なコマンドラインを使って、Setup.exe ファイルのダウンロードと起動を行うセットアップ プレーヤー (Setup.ocx) が含まれています。これにより、限定された権限で Windows Vista 以降を使用しているエンドユーザーもインストールを実行できるようになります。インストールのマニフェストで指定された必要実行レベルにより昇格された権限が必要な場合、適切なユーザー アカウント制御 (UAC) プロンプトが Setup.exe ファイルが起動されたときに表示されます。エンドユーザーのシステムに以前のバージョンの Windows がある場合、マニフェストは無視されますが、もう一方の動作は同じです。

One-Click Install インストールの Setup.exe ファイルに COM 情報は含まれていません。

他のインストールと同様、インターネット インストールは、アップデート、メンテナンス モード、複数インスタンス インストール、サイレント インストールのような機能を利用することができます。つまり、InstallShield インストール用に提供されている機能がすべて利用できるということです。

次の InstallScript 関数は、インターネットで有効です:

テーブル 11 • インターネットで有効な InstallScript 関数

関数	説明
CopyFile	有効な URL によって指定されたファイルをコピーしたり、または CGI や ASP の要求を送信したりします。
ExistsDir	有効な URL によって指定されたフォルダーの存在を確認します。
GetFileInfo	有効な URL によって指定されたファイルに関する情報を取得します。
GetLine	有効な URL によって指定されたファイルから行を読み取ります。
Is	次を確認します。 • インストールがインターネットから実行されているかどうか • 有効な URL によって指定されたファイルやフォルダーの存在 • 指定された文字列が URL かどうか • URL の有効性
OpenFile	有効な URL によって指定されたファイルを開きます。
ParseUrl	指定した URL の一部を取得します。
ReadBytes	有効な URL によって指定されたファイルからバイトを読み取ります。
SeekBytes	有効な URL によって指定されたファイル内でファイルポインターを再配置します。

One-Click Install オブジェクト

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install のセットアップ プレーヤー内のオブジェクトは、Player または Ether です。Player オブジェクトは、オブジェクト モデルの中で最上位を占め、次に Ether と続きます。

Player オブジェクト

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

セットアップ プレーヤーを使用できるようにするには、インスタンスを作成する必要があります。詳細については、「Setup Player を使用する」を参照してください。

Playerには、メソッドとプロパティが 1 つずつあります。

- Open (URL)—データ .cab ファイルの場所を指定し、Ether オブジェクトのリファレンスを返します。引数は、Web サーバー、UNC パス、またはローカル ドライブのファイルの場所へのパスを示す URL から構成されます。
- LastError—メソッド (通常 Open()) によってスローされたエラー コードを示す整数値。

例

```
Player.Open("http://www.mydomain.com/myproduct");
Player.Open("file://¥¥server¥myproduct");
Player.Open("file://c:/my installations/myproduct/media/default");
```

LastError メソッド

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

これは、Player オブジェクトの LastErrorメソッドです。Player のメソッド (通常、Open()) によってスローされるエラーのエラーコードを示す整数値。

構文

```
var nError = player.LastError;
```

例

```
var ether = Player.Open("http://www.installshield.com")
```

パラメーター

なし。

戻り値

テーブル 12・LastError メソッドの戻り値

リターン コード	説明
-2147186688	アクセス許可は付与されていません。[Java セキュリティ] ダイアログ ボックスの [拒否] をクリックすると、エラーがスローされます。
-2147186686	Setup Player ActiveX コントロールのアップデートかインストールができませんでした。APPLETタグの Codebase 属性で指定されたURLからダウンロードする機能を確認します。
-2147186685	ISPlayer のインスタンス作成か、ISPlayer の初期化ができませんでした。
-2147186684	player.Openを呼び出せませんでした。
-2147186683	Player.Open はアプレットの初期化のタイムアウトを呼び出しました。
-2147166892	要求したURLがありません。player.Open 呼び出し中に指定したURLで使用可能なインストール ファイルを確認してください。
-2147166895	保護されたリソースにアクセスできません。認証が必要で、エンドユーザーが正しい情報を入力していないインストールへのアクセスを行った場合、このエラーが返されます。
-2147012889	サーバー名が解決されませんでした。player.Open の呼び出しで使用される URL を確認してください。ドメイン名のスペル ミスが、このエラーの一番の原因となっていますので注意してください。

***メモ・** リストの最初の 6 つのエラーについては、エンドユーザーが Java コンソールで詳しいエラー情報を確認できます。*

Open メソッド

***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

Player オブジェクトの Open メソッドに、インストールのキャビネット ファイルの URL を指定する文字列引数を使うことができます。

構文

```
var ether = Player.Open("<URL 文字列>")
```

例

```
var ether = Player.Open("http://www.installshield.com")
```

パラメーター

テーブル 13 • Open メソッドの有効なパラメーター

パラメーター	説明
<URL 文字列>	インストールのキャビネット ファイルの URL を指定します。

戻り値

Open メソッドは、Ether オブジェクトの参照を返します。

Ether オブジェクト

Ether オブジェクトは、次のメソッドをサポートします:

テーブル 14 • Ether オブジェクトがサポートするメソッド

メソッド	説明
GetLastError	インストールの結果を示す数値を返します。
IsPlaying	インストールが実行中かどうかを示すブール値を返します。
Play	インストールを開始します。
SetProperty	プロパティを一定の値に設定します。

GetLastError メソッド

Ether オブジェクトの Play メソッドを呼んだあと、GetLastError メソッドを呼び出してインストールの結果を取得します。GetLastError により、エラーが発生したかどうか、インストールが正常に完了したかどうか、またはキャンセルされたかどうかが分かります。

構文

```
ether.GetLastError( );
```

例

GetLastError は、次の例のようにポールすることをお勧めします。

```
var intervalID = 0;

function startInstall() {
    if (ether)
    {
```

```
        ether.Play();
        if (intervalID == 0)
            intervalID = window.setInterval("IsSetupFinished()", 3000);
    }
}

function IsSetupFinished()
{
    var nResult = ether.GetLastError();
    if (nResult != 0)
    {
        if (nResult == SETUP_FINISHED) // セットアップが完了しました
            /* to do */ ;
        if (nResult == SETUP_ERR_CANCELLED) // セットアップがキャンセルされました
            /* to do */ ;

        clearInterval(intervalID);
        intervalID = 0;
    }
}
```

パラメーター

なし。

戻り値

GetLastError の全戻り値の一覧は、リリースの Disk Images フォルダーにある Disk1¥_graphics¥Utilities.js にあり、次でも確認することができます。

- SETUP_RUNNING
- SETUP_FINISHED
- SETUP_ERR_GENERAL
- SETUP_ERR_LOADMEDIA
- SETUP_ERR_INSTALLKERNEL
- SETUP_ERR_STARTKERNEL
- SETUP_ERR_OPENCAB
- SETUP_ERR_INSTALLSUPPORT
- SETUP_ERR_SETTEXTSUB
- SETUP_ERR_INITINFO
- SETUP_ERR_GETSETUPDRIVER
- SETUP_ERR_INITPROPERTIES
- SETUP_ERR_RUNINSTALL
- SETUP_ERR_UNINSTALLSUPPORT
- SETUP_ERR_EXTRACTBOOT

- SETUP_ERR_DOWNLOADFILE
- SETUP_ERR_CANCELLED

IsPlaying メソッド

Ether オブジェクトの IsPlaying メソッドは、インストールが実行されているかどうかを示すブール値を返します。

構文

var <変数> = ether.IsPlaying();

例

var bPlaying = ether.IsPlaying();

パラメーター

なし。

戻り値

テーブル 15 • IsPlaying メソッドの戻り値

戻り値	説明
true	インストールが実行中であることを示します。
FALSE	インストールが実行されていないことを示します。

Play メソッド

Ether オブジェクトの Play メソッドは、インストールを起動します。

このメソッドの使用例を見るには、リリース ウィザードの [インターネット オプション] パネルまたはリリース プロパティ グリッドの [インターネット] ページで「デフォルトの Web ページの作成」オプションを使ってメディアをビルドしてから、生成された Web ページ (Setup.htm) でコードを調べます。

構文

ether.Play();

パラメーター

なし。

戻り値

なし。

SetProperty メソッド

Ether オブジェクトの SetProperty メソッドは、2 つの引数を受け取ります。最初の引数は文字列で、is::CmdLine プロパティの名前を表します。2 番目は Setup.exe コマンドライン パラメーターを指定する文字列です。

構文

ether.SetProperty("is::CmdLine","Setup.exe command-line options");

例

ether.SetProperty("is::CmdLine","-l0x0407");

パラメーター

次のテーブルには、SetProperty メソッドのパラメーターが含まれています。

テーブル 16 • SetProperty メソッドのパラメーター

プロパティ名	プロパティ値
is::CmdLine	Setup.exe コマンドライン パラメーターを指定する文字列。サポートされているパラメーターの一覧は、「Setup.exe および Update.exe コマンドライン パラメーター」をご覧ください。

戻り値

なし。

One-Click Install インストールのデジタル署名

エンドユーザーが One-Click Install インストールを実行すると、セットアップ プレーヤー (Setup.ocx) は常に次のファイルのデジタル署名を検証します:

- Setup.ocx
- Setup.exe
- Data1.hdr
- ISSetup.dll

エンドユーザーが One-Click Install インストールを実行すると、セットアップ プレーヤー (Setup.ocx) は常に次のファイルのデジタル署名を検証します: すべてのファイルが適切に署名されていて、証明書がまだ完全に信頼されていない場合、セットアップ プレーヤーはプロンプトを表示して、エンドユーザーの承認を求めます (プロンプトは、ダウンロードした実行可能ファイルを実行するときに表示されるプロンプトに似ています)。

InstallShield は、Setup.ocx ファイルの署名を行います。Setup.exe ファイル、Data1.hdr ファイル、および ISSetup.dll ファイルの署名に使用する証明書は同じである必要があります。Setup.exe ファイルとメディア ヘッダーを署名するオプションを選択すると、この要件は自動的に満たされます。

古いプロパティとメソッドの置換

InstallShield のインターネット インストール用簡易化モデルでは、InstallShield Professional 6.x で Ether オブジェクトとそのサブオブジェクトが持っていたメソッドとプロパティを消去します。以下は、そのメソッドとプロパティのリスト、および、Professional 6.x インストール Web ページを InstallShield インストールと共に動作させるため変換する際、それらのをどう置換するかの説明です。

AskDestPath

相当するダイアログをインストール中に表示するには、インストールの スクリプトで InstallScript 関数 **AskDestPath**を呼び出します。Web ページからインストールへのあて先パスを渡すには、"スクリプト定義" プロパティを設定します。

CommandLine

次のような行を、

ether.CommandLine = "<文字列値>";

次のように変更します。

ether.SetProperty("is::CmdLine","<文字列値>");Features, SetState

相当する機能をインストール中に提供するには、インストールのスクリプトで InstallScript 関数 **SdFeatureTree** を呼び出します。Web ページからインストールに機能の選択を渡すには、スクリプト定義のプロパティを設定して、インストール スクリプトでその値を使用して **FeatureSelectItem** を条件付きで呼び出します。

Features, GetState

インストール中に機能の選択状態を取得するには、インストールのスクリプトで InstallScript 関数 **FeatureIsItemSelected** を呼び出します。Web ページは、機能の選択状態を取得できません。

FileToOpen

インストール中にファイルを開くには、インストールのスクリプトで InstallScript 関数 **LaunchApplication** を呼び出します。Web ページからインストールへファイル名を渡すには、スクリプト定義のプロパティを設定します。

GetDownloadSize, FileLevelCosting

インストール中に必要なディスク空き容量を取得するには、インストールのスクリプトで InstallScript 関数 **FeatureGetTotalCost** を呼び出します。(**SdFeatureTree** のような機能のダイアログは、必要なディスク空き容量を表示します。)Web ページは必要なディスク空き容量を取得できません。

言語

次のような行を、

ether.Language = "<言語コード>";

次のように変更します。

ether.SetProperty("is::CmdLine","-l<言語コード>");LegacyMode

デフォルトで、インターネット インストールは *レガシー モード* で実行されます。つまり、インストールのユーザー インターフェイス イベントが適切に生成された状態でということです。Web ページで ether.LegacyMode=false; の設定効果を複製する方法が 2 つあります。

- 次の行を使って、サイレントでインストールを実行します。

  ether.SetProperty("is::CmdLine","-s");

- [スクリプト定義] プロパティを設定、または Is(WEB_BASED_SETUP) の戻り値を確認して、OnShowUI ハンドラーでインストールのユーザー インターフェイス イベント ハンドラー関数を条件付で呼び出します。

ProductName

Web ページは、この情報を取得できません。

GetTargetDir

Web ページは、この情報を取得できません。

GetTextSub

Web ページは、この情報を取得できません。

IsInstalled

Web ページは、この情報を取得できません。

SetSetupType

インストール中にセットアップの種類を指定するには、インストール スクリプトで InstallScript 関数 **FeatureSetupTypeSet**を呼び出します。インストール中にセットアップの種類選択ダイアログを表示するには、インストール スクリプトで、**SdSetupType** を呼び出します。Web ページからインストールにセットアップの種類を渡すには、スクリプト定義プロパティを設定します。

SetTargetDir

インストール中にターゲット ディレクトリを指定するには、インストール スクリプトでシステム変数 TARGETDIR に値を割り当てます。Web ページからインストールにターゲット ディレクトリを渡すには、スクリプト定義プロパティを設定します。

SetTextSub

インストール中にテキスト置換を指定するには、インストール スクリプトで "TextSub オブジェクトの値" プロパティに値を割り当てます。Web ページからインストールにテキスト置換を渡すには、[スクリプト定義] プロパティを設定します。

デフォルトの Web ページ (Setup.htm)

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.htm という名前にデフォルトの Web ページは、次のメソッドいずれかを使用して One-ClickInstall インストールをビルドした際に、ディスク イメージ フォルダー内に作成されます。

- リリース ウィザードの [インターネットのオプション] パネルで **[セットアップのデフォルト Web ページを作成]** オプションを選択。
- [リリース] ビューの [インターネット] タブにある "デフォルト Web ページの作成" 設定で [はい] を選択。

このページをそのまま使用する、自由にカスタマイズする、または Setup Player の使用例として参照することもできます。

上記の Web ページの作成に関する設定は、[リリース] ビューの "ClickOnce Install の生成" 設定が [いいえ] に設定されているとき、無効になりますので注意してください。

One-Click Install インストール

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ホスト システム上で HTTP 1.1 をサポートする Web サーバーが必要です。FTP は One-Click Install インストールをサポートしませんので注意してください。

Setup Player の別の呼び出し方法

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Web ページからセットアップ プレーヤーを呼び出す必要はありません。他のソースから Player を呼び出すことができます。たとえば、ログイン スクリプト、Visual Basic、Visual C/C++ アプリケーションなどです。ActiveXコントロールをサポートするプログラムツールは、Playerを呼び出すことができます。

Playerを呼び出すためのサンプルVisual Basic スクリプト

```
Dim player, ether
Set player = CreateObject("Setup.Player.Automation Interface Version")
Set ether = player.open("http://www.mydomain.com/mysetup")

If ether.IsPlaying() Then
    MsgBox "セットアップが実行中です。"
Else
    MsgBox "セットアップが実行していません。"
End If

Set player = Nothing
Set ether = Nothing
```

HTTPS と One-Click Install インストール

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install インストールは、HTTPS (secure hypertext transfer protocol) の下、つまり SSL (Secure Sockets Layer) を使った HTTP の下で実行することができます。HTTPS と共に実行するためにインストールが必要な唯一の変更は、インターネット対応関数に渡される URL を http:// から https:// へ変更することです。

One-Click Install インストールの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

リリース ウィザードを使用して、One Click Install インストールを作成することができます。

タスク ***リリース ウィザードを使用して、One Click Install を作成するには、以下の手順を実行します。***

1. **リリース ウィザード**を起動します。
2. **[インターネットのオプション]** パネルに移動します。
3. **[One-Click Install の生成]** オプションを選択します。
4. **[セットアップにデフォルトの Web ページを作成]** チェック ボックスを選択するか、**[起動する URL の入力]** ボックスでカスタム Web ページの場所を指定します。
5. 必要に応じて、他のオプションを指定します。
6. [リリース ウィザード] パネルの残りの部分を完了させます。
7. リリースをビルドします。

タスク ***基本の MSI または InstallScript MSI プロジェクトの One-Click Install インストールを作成するには、以下の手順を実行します。***

1. **リリース ウィザード**を起動します。
2. **[メディアの種類]** パネルに移動します。
3. **[メディアの種類]** 一覧で、**Web** をクリックします。
4. **[One Click Install]** パネルに移動し、**[One-Click Install の生成]** オプションを選択します。
5. .htm または .cab ファイルのファイル名を指定します。
6. One-Click Install がサポートするブラウザーを指定します。
7. [リリース ウィザード] パネルの残りの部分を完了させます。
8. リリースをビルドします。

Setup Player を使用する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

セットアップ プレーヤーは、Setup.exe ファイルをダウンロードしてから、適切なコマンドラインを使って起動する Setup.ocx ファイルです。Setup.ocx は Setup.exe 以外の名前を持つセットアップ実行可能ファイルの起動もサポートします。これが正しく動作するためには、Setup.ini ファイルの [Startup] セクションに次のキーが含まれている必要があります:

LauncherName=*SetupLauncher*

SetupLauncher は、代替セットアップ ランチャーのファイル名です。

Web ページで Setup Player を使用する例として、リリース ウィザードの [インターネットのオプション] パネルにある [デフォルトの Web ページを作成] オプション、または [リリース] ビューの [インターネット] タブにある "デフォルトの Web ページを作成" 設定を使用してリリースをビルドします。それから、生成された Web ページ (Setup.htm) でコードを調べます。

Player のインスタンスを作成したら、Player 内で各オブジェクトに関連付けられたメソッドおよびプロパティを使用して、インストールをカスタマイズできます。

起動された ClickOnce Install インストールにデータを渡す

起動された One-Click Install インストールにデータを渡すには、is::CmdLine parameter と CMDLINE スクリプト変数を使用します。

インターネット インストールをサイレントで実行する

インターネット インストールをサイレントで実行するには、Ether オブジェクトの SetProperty メソッドを呼び出して Setup.exe コマンドライン オプションを以下の例のように渡します。

```
ether.SetProperty("is::CmdLine","-s -f1¥"C:¥¥My Folder¥¥Mydir.iss¥"");

ether.SetProperty("is::CmdLine","-s -f1¥"C:¥¥My Folder¥¥Mydir.iss¥" -f2¥"C:¥¥My Folder¥¥Mydir.log¥"");
```

次の事項に注意してください。

- サイレントのインターネット インストールを実行する際、-f1 オプションを指定する必要があります。インターネット インストールには、応答ファイル用のデフォルトの場所がありません。
- サイレントのインターネット インストールを実行または記録する際、-f1 または -f2 オプションを使用する場合、完全修飾絶対パスを指定しなければなりません。これらのオプションに、URL は使用できません。
- **SetProperty** への 2 番目の引数には、エスケープ シーケンス (¥¥ と ¥") を使用して、円記号と引用符を上記の例にしたがって指定する必要があります。

ランタイム言語の設定

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

エディション: *複数言語サポートは、InstallShield の Premier Edition で使用することができます。*

次のメソッドで言語を指定すると、InstallShield で指定したデフォルトの言語、またはターゲット システムのデフォルトの言語にかかわらず、One-Click Install インストールはこの言語で実行されます。このメソッドを使用しない場合、One-Click Install インストールは、他のインストールと同じ方法でランタイム言語を判別します。

タスク ***ランタイム メッセージを追加するには、以下の手順を実行します。***

1. 言語のサポートをプロジェクトに追加します。
   a. **[プロジェクト]** メニューで、**[設定]** をクリックします。**[プロジェクトの設定]** ダイアログ ボックスが開きます。
   b. **[言語]** タブをクリックします。
   c. インストールがターゲットにする、言語のチェック ボックスを選択します。
2. リリースに言語を追加します。
   a. **[ビルド]** メニューで **[リリース ウィザード]** をクリックします。
   b. **[セットアップ言語]** パネルから、適切な言語を選択します。
3. Play メソッドを呼び出す前に、InstallShieldの Web ページで、SetProperty メソッドを呼び出して、適切な言語識別子を指定します。

   たとえば、英語の言語 ID は 0009 です: 英語のランタイム言語を設定するには、以下のコードを使用します:

   ```
   ether.SetProperty("is::CmdLine","-l0009");
   ```

デバッグ モードで One-Click Install インストールを実行する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

デバッグモードで One-Click Install を実行するには、ether.Play を呼び出す前に以下を入力します。

```
ether.SetProperty( "is::CmdLine", "/d")
```

InstallScript からユーザーを認証する

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

One-Click Install 実行後に認証が必要な場合、InstallScript を使用して直ちにユーザーを認証できます。認証が必要なとき、Web サーバーからインストールに HTTP エラーコード 401 が送信されて、InternetError イベントが発生します。OnInternetError イベント ハンドラーでユーザーにユーザー名とパスワードを要求して、このイベントに対応します。

これを簡単に処理するには、WinInet.dll の **InternetErrorDlg** API を使用し、OnInternetError から IDRETRY 値を返します。これによって、インストールが HTTP 要求を再度発行してファイルを求めます。ユーザーが間違った情報を入力した場合、再入力を求めるプロンプトが表示されます。

Setup.ini

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.ini は、InstallScript ベース プロジェクトのビルド プロセス中にインストールの一部の要素を制御するために作成される初期化ファイルです。ビルド プロセスは、Setup.ini に一定のキー名と値のみを挿入します。ビルド プロセスが Setup.ini を作成すると、<プロジェクト フォルダー>¥Media¥<メディア名>¥Disk Images の Disk1 フォルダーに配置されます。

さらに Setup.ini をカスタマイズする場合は、テキスト エディターで変更します。希望の変更を行うバッチ ファイルまたは実行可能ファイルを起動するリリース ウィザードの [ポストビルドのオプション] パネル、または [リリース] ビューの "バッチ ファイルの実行" プロパティを使用して、このプロセスを自動化することができます。Setup.ini を、以前のメディアビルドからの変更コピーでただ単に上書きしないでください。上書きするとインストールが正しく動作しないことがあります。

Setup.ini には、次の 2 つのあらかじめ定義されたセクションがあります。

- [Startup]
- [Mif]

追加セクションを Setup.ini に追加して、情報をセットアップ スクリプトに渡すことができます。それから **GetProfString** と **GetProfInt** 関数を呼び出して Setup.ini からの情報をインストールに転送するとができます。

Setup.ini ファイルの例を示します。

```
[Startup]

EnableLangDlg=Y

Product=My Application

ProductGUID=23EAFFCA-361D-11D3-8B0F-00105A9846E9

CompanyName=自分の会社名

Skin=Setup.skin

SmallProgress=N

SplashTime=5

CheckMD5=N

CmdLine=/f1Test1.iss

LauncherName=MyInstall.exe


[Mif]

Type=SMS

Filename=Ishield
```

SerialNo=IS50-32XYZ-12345

Locale=DEU

メモ: *(Disk1 を削除した後に) 後でファイルにアクセスする必要がある場合は、インストールの始めに* Setup.ini *をサポート フォルダー (SUPPORTDIR) にコピーする必要があります。*

[Startup] セクション

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.ini の [Starup]セクションでは、次のキー名を使用できます。

- EnableLangDlg
- 製品
- ProductGUID
- CompanyName
- Skin
- SmallProgress
- SplashTime
- CheckMD5
- CmdLine
- LauncherName

EnableLangDlg

EnableLangDlg キー名は、インストール初期設定中に [言語] ダイアログ ボックスを表示するかどうかをインストールに指示します。[言語] ダイアログを使用して、エンドユーザーはインストールの実行時に使用する言語を使用可能な言語から選択することができます。この値は、**リリース ウィザード**の **[セットアップの言語]** パネル、または **[リリース]** ビューの **"言語ダイアログ"** プロパティで設定できます。

[言語] ダイアログについての詳細は、「インストールがユーザー インターフェイスで使用する言語を判別する方法」を参照してください。

製品

Product キー名は、[スタートアップ メッセージ] ダイアログ ボックスで、テキスト文字列の最初に表示されるアプリケーション名または製品名を識別します。

ProductGUID

ProductGUID キー名は、Setup.exe が初期化中にこのデータにアクセスできるように、インストールの GUID を指定します。メディア ビルダーは自動的に、Setup.ini に正しい値を入力します。値を変更しないでください。

CompanyName

CompanyName キー名は、会社の名前を指定します。

Skin

Skin キー名は、インストールがエンドユーザー ダイアログの表示に使用するスキン ファイルの名前を指定します。このキー名が Setup.ini ではない場合、スキンは使用されません。スキンは、リリース ウィザードの ユーザー インターフェイス パネル、または [リリース] ビューの "スキンの指定" プロパティで指定できます。

SmallProgress

SmallProgress キー名は、インストール初期化ダイアログを、InstallShield Professional バージョン 6.31 以前で作成したインストールによって表示されていた小さいボックスにするか、残りのエンドユーザー ダイアログ ボックスと同じ大きいボックスにするかを指定します。この値を Y に設定すると小さいダイアログ ボックスが表示され、N に設定すると大きいダイアログ ボックスが表示されます。(インストールが [セキュリティ]、[セットアップの保存および/または実行]、[セットアップ言語の選択] または [正規の製品が検出されました] ダイアログ ボックスを表示する場合、インストール初期化ダイアログ ボックスは、このキーの値に関係なく、残りのエンドユーザー ダイアログ ボックスと同じ大きさになります。)この値は、リリース ウィザードの ユーザー インターフェイス パネル、または [リリース] ビューの "小さい初期化ダイアログ" プロパティで設定できます。

SmallProgress が N に設定されていると、インストール初期化ダイアログ ボックス(および [セットアップ言語の選択] ダイアログ ボックスがあればこれも該当)は、スタートアップ グラフィックが閉じるまで表示されません。スタートアップ グラフィックの表示時間を指定するには、SplashTime 値を設定します。

SplashTime

SplashTime キー名は、スタートアップ グラフィックが表示される時間を秒単位で指定します。

CheckMD5

CheckMD5 キー名は、各ファイルの MD5 ハッシュ値をファイル展開中にインストール ヘッダーファイルに保存されている値と比較するかどうかをインストールに指示します。一般オプション – [詳細] ダイアログ ボックスの [MD5 シグネチャの検証] チェック ボックスでこの値を設定できます。

ヒント: *MD5 確認は破損したファイルを検出します。これは インターネットでのインストール中に便利です。MD5 確認を行わなかった場合、ファイル転送処理速度は速くなります。*

CmdLine

CmdLine キー名は、コマンドラインからパラメーターが渡されない時に Setup.exe に渡すコマンドライン パラメーターを識別します。

メモ: Setup.exe *を起動するときにコマンドライン プロンプトで* /v *パラメーターを使用する場合、CmdLine キー名に指定されたパラメーターは一切無視されます。*

使用可能なコマンドライン パラメーターの詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

LauncherName

Setup.exe の名前を変更すると、ファイルの新しい名前は LauncherName キー名によって指定されます。このキー名は、別のインストールが **DoInstall** 関数を使ってこのインストールを起動する場合に必要です。

[Mif] セクション

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

[Mif] セクションがある場合、インストールは自動的に、インストール .mif ファイルを Temp フォルダーに作成します。Setup.exe の –m コマンドライン オプションを使用し、オプションで –m1 と –m2 のオプションを使用することによって .mif ファイルを作成することもできます。

Setup.ini の [Mif] セクションの下には、キー名があります。

- Type
- Filename
- SerialNo
- ロケール

Type

このキーを SMS に設定します。

Filename

このファイル名キーはオプションです。作成される .mif ファイルの代替名を提供します。このキーが含まれていないと、インストールは Setup.ini の [Startup] セクションの下にある AppName キーを .mif ファイル名として使用しようとします。AppName キーも存在しない場合、インストールは Status.mif というデフォルトの名前を持つファイルを作成します。

.mif ファイルには .mif 拡張子が必要なので、ファイル名には拡張子は含めません。ファイル名にはパスも含まれません。パスはデフォルトで Temp フォルダーに置かれます。

SerialNo

SerialNo キーもオプションです。このキーの情報が提供されると、.mif ファイルの [シリアル番号] セクションに配置されます。このキーが存在しない場合、インストールは引用符 (“”) を代わりに置きます。

ロケール

.mif ファイルに指定したロケールを含めます。English (ENU) がデフォルトですが、ロケール文字列の全リストについては Microsoft のマニュアルを参照してください。

次は、自動的に .mif ファイルを作成するインストールの Setup.ini ファイルの例です。

```
[Startup]

AppName=InstallShield
```

```
[Mif]

Type=SMS

Filename=IShield

SerialNo=IS50-32XYZ-12345

Locale=DEU
```

リリースのクローンを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield では、[リリース] ビューでリリースをクローンまたはコピーすることができます。これにより、すべてのプロパティがクローンされたリリースにあったのときと同じ状態でリリースを作成することができます。従って、ほとんど相違点がないリリースを複数作成する場合、リリースをクローンして必要に応じて設定のみ変更するだけですみます。

タスク ***リリースをコピーまたはクローンするには、以下の手順を実行します。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. コピーするリリースを右クリックして **[クローン]** を選択します。

*「リリース x のコピー」*という名前のリリースが作成されます。リリースのコピーには、オリジナルのリリースと同じプロパティがすべてあります。リリースの名前を変更し、必要に応じリリースの設定を変更します。

Windows アプリ パッケージの作成

プロジェクト: *Windows アプリの作成は、基本の MSIのプロジェクト タイプで使用できます:*

重要: *デスクトップ拡張 (デスクトップ ブリッジ) を含む Windows アプリ パッケージ (.appx) のインストールおよびテストを行うには、Windows 10 Anniversary Update が必要です。さらに、Windows アプリ パッケージに署名を行う、Windows アプリ ロゴのバリエーションを活用する (「Windows アプリ ロゴの考慮事項」を参照)、あるいはローカライズされた Windows アプリ パッケージをビルドするためには、InstallShield を Windows 10 マシン上にインストールするか、Windows 10 SDK と共にインストールする必要があります。*

Windows 8.x および 10 上にアプリを配布およびインストールする為に使用される Windows アプリ パッケージ (.appx/.msix) は、シンプルでセキュリティ保護されたパッケージ フォーマットで、UWP (ユニバーサル Windows プラットフォーム) で使用可能な唯一のフォーマットです。Windows アプリ パッケージの利点:

- 高い可用性、信頼性、および耐久性によって、アプリケーションが長期間にわたってエラーなしで継続的に動作し続けます。
- 必要最小限の構成とカスタマイズ不要な UI によるスタティック ビルドを使ったスムーズなインストール経験
- Windows ストアを使ってアプリケーションを販売または提供できるオプション
- UWP API を使用できる機能だけでなく、ライブ タイルなどの UWP 機能を活用
- Windows Nano Server 上で サポートを持つ唯一のパッケージ フォーマット

InstallShield は、Windows アプリ パッケージ フォーマットおよびそのデスクトップ/サーバー拡張の作成をサポートし、Windows アプリ フォーマットに適合しないアイテムを識別するためのテストを提供します。[リリース] ビューでリリースを選択するとき、**[Appx/MSIX]** という名前の新しいリリースごとに提供されるタブに、Windows アプリ パッケージを作成する設定があります。ここで、Windows アプリ パッケージのビルド プロセスに影響する様々な主要な設定を指定できます。

Windows アプリ パッケージは、**[Appx/MSIX]** タブの **[表示プロパティ]** 領域にある設定を使ってローカライズできます。Windowsアプリ パッケージのビルドに使用できるサポート対象言語は、リリースの [ビルド] タブにある **"UI 言語"** 設定の言語と一致します。ただし、ビルド マシンには MakePRI.exe（Windows 10 SDK 内）が必要です。MakePRI.exe がビルド マシン上で見つからない場合、InstallShield はデフォルト言語のみを使って Windows アプリ パッケージを生成します。

次の手順は、Windows アプリ パッケージの作成方法を説明します。

タスク ***基本の MSI プロジェクトを Windows アプリ パッケージ (.appx) フォーマットに変換するには、次の手順に従います:***

1. .msi リリースのビルドの一環として、Windowsアプリ パッケージをビルドする基本の MSI プロジェクトを開きます。
2. **[リリース]** エクスプローラーで、製品構成の下にある Windows アプリ パッケージのビルドを行うリリースの名前をクリックします。
3. **[Appx/MSIX]** タブをクリックして、**"WINDOWS アプリ パッケージのビルド"** 設定で **[はい]** を指定します。
4. **Appx、MSIX または両方**のパッケージ タイプを選択してください。
5. 次の手順に従ってください:
   - **[Appx/MSIX]** タブの **[全般]** 領域を使って、デスクトップ拡張またはサーバー拡張を含めるかどうかを選択するなど、ビルド処理に影響を及ぼす主要な設定を構成します。
   - **[Appx/MSIX]** タブの **[パッケージ ID のオーバーライド]** 領域を使って、Windows に対してパッケージを一意に識別します。
   - **[Appx/MSIX]** タブの **[表示プロパティ]** 領域を使ってエンドユーザーに対して、パッケージを識別します。

   これらの設定についての詳細は、「リリースの [Appx/MSIX] タブ」を参照してください。
6. Windows アプリ パッケージで作成されたタイルを構成するには、**[ショートカット]** ビューを開いて、**[WINDOWS アプリ パッケージ タイルのオーバーライド]** 領域で設定を構成します。**「Windows アプリ パッケージ タイルのオーバーライド"** 設定についての詳細は、「ショートカットの設定」を参照してください。
7. リリースをビルドします。

8. .msi パッケージ内で Windows アプリ パッケージ フォーマットに適さないアイテムの存在をスキャンするには、InstallShield MSIX/UWP アプリ適合性スイートを実行します。**[ビルド]** メニューから **[検証]** をポイントしてから、**[InstallShield MSIX/UWP アプリ適合性スイート]** をクリックします。詳細については、「InstallShield MSIX/UWP アプリ適合性スイート」を参照してください。

   Windows アプリ パッケージの配布方法についての詳細は、「Windows アプリ パッケージの配布」を参照してください。

インストールのテストと実行

インストールを配布する前にテストを行い、顧客に頼らずに問題をすべて見つけることはとても重要です。InstallShield では、(ファイルをターゲット システムにコピーする手間をかけずに) エンドユーザー ダイアログをテストしたり、転送するファイルを含むインストール全体を実行することができます。

- インストールのテスト実行
- InstallShield インターフェイスからインストールを実行する
- サイレント モードでインストールを実行する
- InstallScript MSI プロジェクトの MSI サイレント モード インストール
- コマンドラインを使ったサイレント アンインストール (InstallScript MSI のみ)
- スイート インストールをコマンドラインから最小ユーザー インターフェイスを使って実行する
- コマンドラインからアップデートのダウンロードを行わずにスイートのサイレント インストールを実行する

インストールのテスト実行

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

インストールをテスト実行してエンドユーザー ダイアログを確認することができます。テスト実行によって、完全ユーザー インターフェイスが表示されますが、ターゲット システムにファイルがコピーされたり、レジストリが更新されることはありません。

タスク ***プロジェクトをテスト実行するには、以下のいずれかを行ってください。***

- **[リリース]** ビューで、リリース名を右クリックしてから **[セットアップのテスト]** をクリックします。
- ツールバーにある**[テスト]** ボタンをクリックします。

InstallShield インターフェイスからインストールを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

リリース ウィザードを実行してインストールのリリースをビルドすると、それを InstallShield 内から実行することができます。

タスク ***ビルドしたインストール パッケージを実行するには、以下のいずれかを実行します。***

- ツールバー上の **[実行]** ボタンをクリックします。
- **[リリース]** ビューで、リリース名を右クリックしてから **[セットアップの実行]** をクリックします。
- **[ビルド]** メニューで、**[ReleaseName の実行]** をクリックします。*ReleaseName* は、現在 **[リリース]** ビューでフォーカスされているリリースの名前です。

InstallShield は、[リリース] ビューで現在強調表示されているリリースを実行します。

[オプション] ダイアログ ボックスで [インストール前にアンインストールする] チェック ボックスを選択すると、これまでに実行されたすべてのリリースが現在のリリースが実行される前にアンインストールされます。このオプションは、[オプション] ダイアログ ボックスの [プリファレンス] タブから使用できます。

InstallShield インターフェイスからインストールを起動するとき、コマンドライン パラメーターを指定することはできません。Windows Installer にパラメーターを渡すには、コマンドラインからインストールを起動する必要があります。詳細については、「MsiExec.exe コマンドラインのパラメーター」を参照してください。

サイレント モードでインストールを実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

サイレント インストールはエンド ユーザー インターフェイスなしで実行されるインストールです。インストールをサイレントで実行する場合、InstallShield では、基本の MSI、InstallScript MSI および InstallScript プロジェクトにサイレント インストールを作成できます。

基本の MSI のサイレント インストール

基本の MSI インストールをサイレントで実行するには、コマンドラインで次のように入力します。

msiexec /i Product.msi /qn

リリース設定に Setup.exe が含まれている場合、次のコマンドを実行することができます:

Setup.exe /s /v"/qn"

基本の MSI インストールは、応答ファイルの作成や読み込みは行いません。基本の MSI プロジェクトのインストール プロパティを設定するには、コマンドラインから次のように入力します。

msiexec /i Product.msi /qn INSTALLDIR=D:¥ProductFolder USERNAME="お客様各位"

InstallScript MSI および InstallScript サイレント インストール

InstallScript MSI および InstallScript プロジェクトでは、エンドユーザーの対応を記録する応答ファイルに記録を作成する必要があります。この応答ファイルは、インストールが実行できるように、Setup.exe に渡されます。従来のサイレント インストールは、通常のインストールとほとんど同じように動作します。このインストールは、通常のインストールと同じスクリプト論理に従っています。

Setup.exe を使用せずに InstallScript MSI インストールをインストールする必要がある場合、MSI サイレント モードを使用することができます。

InstallScript MSI と InstallScript のサイレント インストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield Silent では、サイレント インストールとしても知られる自動化された電子的なソフトウェアの配布が可能です。InstallShield Silent を使用すると、エンド ユーザーがインストールを監視したり、ダイアログ ボックスに情報を入力する必要はありません。InstallShield Silent インストールは、エンドユーザーの操作なしで自動的に実行されます。

マシン上にアプリケーションの複数のインスタンスがあり、そのマシン上でアプリケーションのサイレント アップデート インストールを実行する場合、最初に見つかったインスタンスにアップデートが適用され、[正規の製品が検出されました] ダイアログ ボックスは表示されません。

InstallShield Silent を起動させるには、Setup.exe –s コマンドライン パラメーターを使用します。

Windows ロゴ: *Windows 95 ロゴ要件に準拠するため、デフォルトのインストール オプションが選択される応答ファイルを、サイレントインストールで作成する必要があります。*

インストールを Setup.exe –r パラメーターで実行して、インストール オプションを選択し、InstallShield Silent 応答ファイルを自動的に記録するか、独自のファイルを作成することができます。応答ファイルの実例については、InstallShield インストールの Disk1 にある Setup.iss ファイルを参照してください。応答のファイル形式の説明については、「応答ファイルを作成する」を参照してください。

Setup.exe を使用せずに InstallScript MSI インストールをインストールする必要がある場合、MSI サイレント モードを使用することができます。

タスク ***サイレント インストールを作成するには、以下の手順を実行します。***

1. インストールを作成します。
2. 応答ファイルを作成します。
3. サイレント インストールを実行します。
4. エラーを確認します。

サイレント インストールの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

サイレント モードで実行されるインストールを作成するには、通常の方法でインストールを作成してから、非サイレント モードでスクリプトをテストします。

インストール スクリプトのロジックを変更し、インストールが サイレント モードが実行されているかどうかによって、簡単にフロー制御を含めることができます。InstallShield では、*MODE* というシステム変数が提供されています。*MODE* 変数には、インストールの現在のモードを識別する次の定数のうちの 1 つが含まれます。

- **NORMALMODE**—インストールが通常モードで実行されていることを示します。
- **RECORDMODE**—Setup.exe によって、Windows フォルダーにサイレント インストール ファイル (.iss ファイル) が自動的に生成されることを示します。.iss ファイルは、インストールの入力値を記録したものです。
- **SILENTMODE**—インストールがサイレント モードで実行されていることを示します。

詳細については、「MODE」を参照してください。

ヒント: *すべての InstallShield ビルトインおよび Sd ダイアログは、InstallShield サイレント応答ファイル (.iss ファイル) に格納された値を自動的に処理します。カスタム ダイアログを作成する場合、***SilentReadData*** *を呼び出し、ダイアログの戻り値をサイレント モードで処理する必要があります。*

インストールを作成または編集したら、応答ファイルを作成してサイレント インストールの作成を続けます。

応答ファイルを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

通常の (非サイレント モード) インストールは、ダイアログ ボックスへの応答という形でエンドユーザーから必要な入力を受け取ります。逆に、サイレント インストールは、ユーザーに入力をプロンプトしません。したがって、別のソースからエンドユーザー インプットを取得する必要があります。その方法が、InstallShield サイレント応答ファイル (.iss ファイル) です。

応答ファイルには、通常のインストール実行時にエンドユーザーがダイアログへの応答として入力するだろうと思われる情報が含まれています。InstallShield サイレントでは、応答ファイルから実行時に必要な情報を読み込みます。

応答ファイルの形式は .ini ファイル形式に類似していますが、拡張子には .iss が付きます。応答ファイルはデータ エントリを含むセクションから構成されたプレーン テキスト ファイルです。

InstallShield サイレント応答ファイルの作成するには、InstallShield を使って自動的に応答ファイルを記録/作成する方法と、直接応答ファイルを作成する方法の 2 種類があります。

応答ファイルの記録

InstallShield で自動的に応答ファイルを作成するオプションがあります。インストールを、Setup.exe –r コマンドライン パラメーターを使用して単純に実行します。InstallShield は、Setup.iss にインストールの設定オプションをすべて記録し、このファイルを Windows フォルダーに格納します。

すべての InstallShield ビルトインおよび Sd ダイアログ関数は、InstallShield が記録モードで実行されたとき (Setup –r)、Setup.iss ファイルに値を書き込むように設計されています。カスタム ダイアログを作成する場合は、**SdMakeName** と **SilentWriteData** を呼び出し、インストールが記録モードで実行されたときにセクションとダイアログ データを応答ファイルに追加します。<InstallShield の場所>Include フォルダー内にある Sd ダイアログのソースコードを見て、これらの関数を使用して Setup.iss に書き込む例を参照してください。**SdMakeName** および **SilentWriteData** を呼び出した際に Setup.iss に追加するデータの詳細については、次のセクションをお読みください。

応答ファイルを手動で作成する

応答ファイルは、手動で初めから作成することもできます。すでに示したように、Setup.iss ファイルは .ini ファイルと類似しています。InstallShield 応答ファイルのセクションは、次の順序で作成しなければなりません。

1. サイレント ヘッダー セクション
2. アプリケーション ヘッダー セクション
3. ダイアログ シーケンス セクション
4. ダイアログ データ セクション (ダイアログ 1 つにつき 1 セクション)

セクション名は **[InstallShield Silent]** というように、角かっこで囲みます。

データ エントリはセクション名の後に続き、次のように <名前=値> という形式のペアにします:

```
Dlg0={23EAFFCA-361D-11D3-8B0F-00105A9846E9}-Welcome-0
```

タスク ***応答ファイルを作成するには、以下の手順を実行します。***

1. テキスト エディターを使用し、Setup.iss という名前のテキストファイルを作成します。
2. Setup.iss にサイレント ヘッダーを入力します。
3. Setup.iss にアプリケーション ヘッダーを入力します。

4. Setup.iss にダイアログ シーケンスを入力します。
5. Setup.iss にダイアログ データを入力します。
6. Setup.iss を保存して閉じます。

この形式に慣れ親しめるよう、サンプル応答ファイルが付属されています。

応答ファイルのサイレント ヘッダー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

すべての応答ファイルは、応答ファイル サイレント ヘッダーで始まります。応答ファイル サイレント ヘッダーを使って、InstallShield はそのファイルが有効な InstallShield 応答ファイルであることを識別することができます。また、応答ファイルが InstallShield の適切なバージョンを使って作成されたインストールに対応することを確認するのにも利用されます。

サイレントヘッダーのフォーマットは、次の通りです。Setup.iss ファイルの始まり部分に次の行を入力します。

```
[InstallShield Silent]
Version=v7.00
File=Response File
```

Version=v7.00 ラインは、InstallShieldSilent 応答ファイルのバージョンを示します。InstallShield のバージョンではありません。すべての応答ファイルに **v7.00** を使用します。

応答ファイルを作成するとき、サイレント インストールの実行に使用される応答ファイルを作成する InstallShield と同じバージョンを使用してください。

応答ファイルのアプリケーション ヘッダー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

応答ファイル アプリケーション ヘッダーは、応答ファイル サイレント ヘッダーのすぐ後に続く応答ファイルの中の 2 番目の情報ブロックです。応答ファイル アプリケーション ヘッダーを使用して、応答ファイルを視覚的に見分けることができます。ヘッダーは、インストール スクリプトや Setup.exe によって使用されます。アプリケーション ヘッダーを使用して、応答ファイルがどのインストールのものかを正確に識別することができます。これはダイアログ データを見ただけでは、概して判断が付きにくいものです。

アプリケーションヘッダーのフォーマットは、次の通りです。名前、バージョン、および会社に割り当てられた値は、インストール スクリプトの **CreateInstallationInfo** 関数への呼び出しでレジストリに書き込まれた値から取得されます。(イベントベースのスクリプトでは、**CreateInstallationInfo** がデフォルト OnMoveData イベント ハンドラー コードで呼び出されます。サイレント ヘッダーの下の Setup.iss ファイルに、次の行を入力します。

```
[Application]
Name=<CreateInstallationInfo からの ProductKey>
```

Version=<CreateInstallationInfo からの VersionKey>
Company=<CreateInstallationInfo からの CompanyKey>

応答ファイルのダイアログ シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

サイレント ヘッダーとアプリケーション ヘッダーの後にある応答ファイルの情報の 3 番目のブロックは、応答ファイルのダイアログ シーケンスです。ダイアログ シーケンス セクションでは、通常のインストールで使用する必要があるダイアログ (カスタム ダイアログを含む) が、表示される順にすべて一覧表示されます。ダイアログ ボックスは、[<PRODUCT_GUID>-DlgOrder] というセクション ヘッダーの下に一覧表示されます。

ダイアログの順番付けシーケンスは 0 から始まります。リストに表示できるダイアログの数に制限はありません。

ダイアログの順序と数は、重要な意味があります。InstallShield Silent が実行されるとき、ダイアログの数または順序のいずれかが、非サイレント インストールのダイアログの順序または数に一致しない場合、サイレント インストールに失敗し、ログ ファイル にエラーが記録されます。ダイアログの発生ごとに、エントリを 1 つ作成します。特定のダイアログは、インストールで複数回表示される場合、複数回一覧表示されます。

以下は、ダイアログ シーケンス エントリの形式です。

Dlg<#>=<PRODUCT_GUID>-<DialogIdentifier>

Dlg<#> の部分は、Dlg という文字列と、シーケンス番号から構成されます。インストールの最初のダイアログは、Dlg0 です。この後の各ダイアログでは、<#> の値が 1 ずつ増加します。

DialogIdentifier の形式は、*functionname*-<#>です。 *functionname* は、ダイアログを作成した関数の名前で、<#> は、インストールにおけるその特定のダイアログのシークエンス番号を意味します。

カスタム ダイアログの場合、<DialogIdentifier> の *functionname* の部分に一意の英数名を使用できます。たとえば、カスタム ダイアログを MyDialog と名前を付けることができます。インストールの 10 番目のダイアログが MyDialog というカスタム ダイアログの 2 番目である場合、ダイアログ シーケンス セクションで、次のようなエントリができます。

Dlg9=<PRODUCT_GUID>-MyDialog-1

<DialogIdentifier> 式は、ダイアログのダイアログ データ セクションを識別するために使用されます。

ダイアログ シーケンス セクションは、常に次の形式のステートメントで終了してください。

Count=<ダイアログの数>

ステートメントは、ダイアログ シーケンス セクションにリストされているダイアログの正確な数を指定します。すべてのエントリをカウントします。ダイアログの順番付けは 0 で開始するため、Count の値は、ダイアログ シーケンスに対する <#> の最大値よりも常に 1 つ多くなります。

次の例は、Welcome と **AskOptions** の 2 つのダイアログです。下の例のように、ダイアログ シーケンスのリストを Setup.iss に入力します。

[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-DlgOrder]
Dlg0={23EAFFCA-361D-11D3-8B0F-00105A9846E9}-Welcome-0

```
Dlg1={23EAFFCA-361D-11D3-8B0F-00105A9846E9}-AskOptions-0
Count=2
```

応答ファイルのダイアログ データ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

応答ファイルの最後の情報ブロックは、応答ファイルのダイアログ データです。応答ファイルのダイアログ データは、[ダイアログ シーケンス] セクションで識別された各ダイアログから返された値を含む、セクションのコレクションです。各ダイアログには、独自のセクションがあります。リストされた値は、エンドユーザーの入力主導の標準のインストールでダイアログが戻す値と同じになります。また、カスタム ダイアログのダイアログデータセクションを作成することもできます。

以下は、ダイアログ データ セクションの形式です。

```
[<PRODUCT_GUID>-<DialogIdentifier>]
Result=value
Keyname1=value
Keyname2=value
```

[<PRODUCT_GUID>-<DialogIdentifier>] セクション ヘッダーによって、特定のダイアログ ボックスが識別され、ダイアログ データ エントリ リストが続きます。<DialogIdentifier> は、[ダイアログ シーケンス] セクションにダイアログを一覧表示するために使用される式と同じです。

データ エントリ項目は、keyname=value という形式になります。キー名は、ダイアログが戻す値の名前で、応答ファイルに記録されます。カスタム ダイアログを含むすべてのダイアログによって、キー名 Result の値が戻され、ダイアログを終了するためにクリックされたボタンを反映します。多数のダイアログによって、変数の値が設定されるか戻されます。

たとえば、非サイレント インストールでは、**AskDestPath** ダイアログによって、svDir パラメーターにインストール先が返されます。スクリプトの行は、次のようになります。

```
nResult = AskDestPath( szTitle, szMsg, svDir, 0 );
```

サイレント インストールの場合、Setup.iss ファイルの対応するダイアログ データ エントリは、次のようになります。

```
[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-AskDestPath-0]
Result=1
szPath=C:¥Program Files¥InstallShield¥2022
```

上の例では、**Result=1** は、ダイアログの [次へ] ボタンをクリックするのに等しく、**szPath=C:¥Program Files¥InstallShield¥2022** は、**AskDestPath** の svDir パラメーターに返される値です。

スクリプトで使用される変数の名前は、Setup.iss ファイルに関して意味がありません。ただし、Setup.iss ダイアログ データ セクションでは、ビルトイン ダイアログ ボックスと Sd ダイアログに、パラメーターにマップする独自のキー名のセットがそれぞれあります。キー名は重要で、各ダイアログに対して定義されたキー名と同じでなければなりません。次の「ダイアログ データのキー名リスト」を参照してください。

カスタム ダイアログを使用する場合、各ダイアログの Result キー名に対するダイアログ データ エントリ、およびカスタム ダイアログによって設定または返される他の値に対するエントリを作成する必要があります。カスタム ダイアログに対する keyname=value 式の作成方法の指針として、下のダイアログ データのキー名リストを使用

してください。**GetProfString** または **GetProfInt** を呼び出して、カスタム ダイアログに対するダイアログ データ情報を確認します。**GetProfString** および **GetProfInt** によって、.iss ファイル、セクション、およびキー名を指定することができ、指定したキー名に割り当てられた値が返されます。

Result キー名の標準値

カスタム ダイアログを含むすべてのダイアログによって、キー名 Result の値が戻され、ダイアログを終了するためにクリックされたボタンを示します。ボタンが押されたことが示されない場合、Result の標準値は以下のようになります。

- 12 – [戻る] ボタン
- 1 – [次へ] または [OK] ボタン

コンポーネントとサブコンポーネントの選択の記録

あるダイアログ関数を使用すると、エンドユーザーがコンポーネントやサブコンポーネントを選択できるようになります。コンポーネントとサブコンポーネントの選択を応答ファイルに記録するために使用されるダイアログ データのキー名エントリには、type、count、および <#> の3 種類があります(次を参照)。

コンポーネント選択の各セットおよびサブコンポーネント選択の各セットには、1 つの type キー名エントリ、1 つの count キー名エントリ、および各コンポーネントまたはサブコンポーネントの選択を記録するために必要な数の <#> キー名エントリがあります。

コンポーネントの選択を記録するためにキー名を作成する際、以下のように、type、count、および <#> キー名エントリの前に "Component" という語を入力します。

Component-typeComponent-countComponent-0

サブコンポーネントの選択を記録するためにキー名を作成する際、以下のように、type、count、および <#> キー名エントリの前に、サブコンポーネントが属するコンポーネントの名前を入力します。

Program Files-typeProgram Files-countProgram Files-0Program Files-1

完全な値エントリを作成するには、等号記号を使用して、値をキー名に付加します(各種類のキー名に割り当てられる値のタイプは、下に説明されています)。次の例は、**Program Files** と **Binary Files** という 2 つのコンポーネントと、**Executables** と **Support Elements** という **Program Files** の 2 つのサブコンポーネントの完全な値エントリです。

```
Component-type=string
Component-count=2
Component-0=Program Files
Component-1=Binary Files
Program Files-type=string
Program Files-count=2
Program Files-0=Executables
Program Files-1=Support Elements
```

type キー名によって、コンポーネントまたはサブコンポーネントリストのデータ型が示されます。InstallShield ダイアログは現在、コンポーネント リストとサブコンポーネント リストに、文字列リストしか使用しないため、Component-type=string のように、type は常に「**string**」に等しくなります。今後のバージョンでは、番号リストが使用される可能性もあり、その場合、type は **number** に等しくなります。

count キー名エントリ

count は、特定のコンポーネントまたはサブコンポーネントエントリに対する選択の数に等しくなります。たとえば、**ComponentDialog** ダイアログで 2 つのコンポーネントがインストールに選択された場合、カウント ダイアログ データ エントリは **Component-count=2** のようになります。**Program Files** コンポーネントの 2 つのサブコンポーネントが選択された場合、**Program Files-count=2** エントリが作られます。

<#> キー名エントリ

<#> キー名エントリの数字部分は、0 で開始して 1 ずつ増加する続き番号で、記録された各コンポーネントまたはサブコンポーネントの選択を番号付けします。番号付けは 0 から開始するため、最大値は、常に count の値より 1 つ少なくなります。

<#> キー名エントリに割り当てられる値は、選択されたコンポーネントまたはサブコンポーネントの表示名 (コンポーネント リストまたはサブコンポーネント リストがビルドされた際に、2 番目のパラメーターとして **ComponentAddItem** に渡された文字列) です。

たとえば、**ComponentDialog** ダイアログによって、**Program Files**、**Help Files**、**Sample Files**、および **Utilities** のコンポーネントの選択がエンドユーザーに提供される例を考えてみましょう。エンドユーザーが **Program Files** と **Help Files** を選択した場合、**ComponentDialog** ダイアログのこのインスタンスに対するダイアログ データ セクションには、2 つのリスト項目エントリが含まれ、次のようになります。

```
[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-ComponentDialog-0]
szDir=C:¥MYAPP¥FILES
Component-type=string
Component-count=2
Component-0=Program Files
Component-1=Help Files
Result=
```

次の例では、サブコンポーネントリストの選択が記録される方法について説明しています。例は、**SdComponentMult** ダイアログのインスタンスの場合です。例では、**Program Files** および **Help Files** の 2 つのコンポーネントが選択されていることが示されています。また、**Main Files**、**Aux.Files**、**Main Help**、および **Tutorial Files** の 4 つのサブコンポーネントも選択されたことを示します。**Main Files** と **Aux.Files** は、**Program Files** のサブコンポーネントであり、**Main Help** と **Tutorial Files** は、**Help Files** のサブコンポーネントです。

```
[{23EAFFCA-361D-11D3-8B0F-00105A9846E9}-SdComponentMult-0]
Component-type=string
Component-count=2
Component-0=Program Files
Component-1=Help Files
Program Files-type=string
Program Files-count=2
Program Files-0=Main Files
Program Files-1=Aux.ファイル
Help Files-type=string
Help Files-count=2
Help Files-0=Main Help
Help Files-1=Tutorial Files
Result=1
```

ダイアログ データのキー名リスト

InstallShield ダイアログに対するダイアログ データのキー名は、次のテーブルに一覧表示されています。最初の列には、ダイアログ名が含まれています。2 番目の列には、各ダイアログに適用できるキー名が一覧表示されています。3 番目の列には、キー名に関連した値の説明が含まれています。

テーブル 17 • ダイアログ データのキー名

ダイアログ	キー名	説明
AskDestPath–<#>	結果	標準値
AskDestPath–<#>	szPath	編集フィールドで設定されたパス
AskOptions–<#>	結果	標準値 戻り値の数はスクリプト呼び出しに基づいて変更できるため、AskOptions は特殊です。<#> が選択変数の数である Sel–<#> という形式のキー名のシーケンスを使用できます。番号は 0 で始まります。Sel–<#> エントリの数は、**AskOptions** への特定の呼び出しの変数の数 (チェック ボックス/ラジオ ボタン) に一致します。次の例を参照してください: • **Sel–0**—**AskOptions** の最初の選択変数にマップします。 • **Sel–1**—**AskOptions** の 2 番目の選択変数にマップします。 • **Sel–2**—**AskOptions** の 3 番目の選択変数にマップします。
AskPath–<#>	結果	標準値
AskPath–<#>	szPath	"インストール先" 編集フィールドに入力されたパス
AskText–<#>	結果	標準値
AskText–<#>	szText	編集フィールドからのテキスト
AskYesNo–<#>	結果	1 = エンドユーザーが [はい] ボタンをクリックしました。 0 = ユーザーが [いいえ] ボタンをクリックしました。
ComponentDialog–<#>	結果	標準値
ComponentDialog–<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
ComponentDialog–<#>	Component–type	文字列 (現在使用できる唯一の値です)
ComponentDialog–<#>	Component–count	コンポーネントの選択の合計数
ComponentDialog–<#>	Component–<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
MessageBox-<#>	結果	1 = ユーザーが [OK] ボタンをクリックしました。
RebootDialog-<#>	結果	0 (Result は常に 0 です)
RebootDialog-<#>	Choice	マシンを再起動する前の最後の再起動選択モードを示します。ラジオ ボタンにマップし、以下の可能値があります。 • 602 = "はい、今すぐコンピューターを再起動します。" • 0 = "いいえ、あとでコンピューターを再起動します。" に設定します。
SdAskDestPath-<#>	結果	標準値
SdAskDestPath-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
SdAskOptions-<#>	結果	標準値
SdAskOptions-<#>	Component-type	文字列 (現在使用できる唯一の値です)
SdAskOptions-<#>	Component-count	コンポーネントの選択の合計数
SdAskOptions-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdAskOptionsList-<#>	結果	標準値
SdAskOptionsList-<#>	Component-type	文字列 (現在使用できる唯一の値です)
SdAskOptionsList-<#>	Component-count	コンポーネントの選択の合計数
SdAskOptionsList-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdBitmap-<#>	結果	標準値
SdComponentDialog-<#>	結果	標準値
SdComponentDialog-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
SdComponentDialog-<#>	Component-type	文字列 (現在使用できる唯一の値です)

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdComponentDialog-<#>	Component-count	コンポーネントの選択の合計数
SdComponentDialog-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentDialog2-<#>	結果	標準値
SdComponentDialog2-<#>	Component-type, <サブコンポーネント>-type	文字列 (現在使用できる唯一の値です)
SdComponentDialog2-<#>	Component-count, <サブコンポーネント>-count	コンポーネントまたはサブコンポーネントの選択の合計数
SdComponentDialog2-<#>	Component-<#>, <サブコンポーネント>-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentDialogAdv-<#>	結果	標準値
SdComponentDialogAdv-<#>	szDir	[インストール先] ディレクトリ フィールドに入力されたパス
SdComponentDialogAdv-<#>	Component-type	文字列 (現在使用できる唯一の値です)
SdComponentDialogAdv-<#>	Component-count	コンポーネントの選択の合計数
SdComponentDialogAdv-<#>	Component-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentMult-<#>	結果	標準値
SdComponentMult-<#>	Component-type, <サブコンポーネント>-type	文字列 (現在使用できる唯一の値です)

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdComponentMult-<#>	Component-count, <サブコンポーネント>-count	コンポーネントまたはサブコンポーネントの選択の合計数
SdComponentMult-<#>	Component-number, <サブコンポーネント>-<#>	選択した項目のリスト内の番号 (番号付けは 0 で開始します)
SdComponentTree-<#>	結果	標準値
SdConfirmNewDir-<#>	結果	1 = エンドユーザーが [はい] ボタンをクリックしました。 0 = ユーザーが [いいえ] ボタンをクリックしました。
SdConfirmRegistration-<#>	結果	1 = エンドユーザーが [はい] ボタンをクリックしました。 0 = ユーザーが [いいえ] ボタンをクリックしました。
SdDisplayTopics-<#>	結果	標準値
SdFinish-<#>	結果	1 = 完了
SdFinish-<#>	bOpt1	1 = "はい、README ファイルを開きます。" が選択されました。 0 = "はい、README ファイルを開きます。" が選択されていません。
SdFinish-<#>	bOpt2	1 = "はい、今すぐ [app name] を再起動します。" が選択されました。 0 = "はい、今すぐ [app name] を再起動します。" が選択されていません。
SdFinishEx		特殊処理を必要とします。
SdFinishReboot-<#>	結果	1 = 完了
SdFinishReboot-<#>	BootOption	0 = Windows をマシンで再起動しない 2 = Windows を再起動 3 = マシンの再起動
SdLicense-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 1 = エンドユーザーが [はい] ボタンをクリックしました。

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdOptionsButtons-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 または、[次へ] ボタンがクリックされたことを示します。 101 = オプション ボタン 1 (上) が選択されました。 102 = オプション ボタン 2 が選択されました。 103 = オプション ボタン 3 が選択されました。 104 = オプション ボタン 4 が選択されました。
SdRegisterUser-<#>	結果	標準値
SdRegisterUser-<#>	szName	"名前" フィールドに入力されたテキスト
SdRegisterUser-<#>	szCompany	"会社" フィールドに入力されたテキスト
SdRegisterUserEx-<#>	結果	標準値
SdRegisterUserEx-<#>	szName	"名前" フィールドに入力されたテキスト
SdRegisterUserEx-<#>	szCompany	"会社" フィールドに入力されたテキスト
SdRegisterUserEx-<#>	szSerial	"シリアル (番号)" フィールドに入力されたテキスト
SdSelectFolder-<#>	結果	標準値
SdSelectFolder-<#>	szFolder	"プログラム フォルダー" フィールドに入力されたフォルダー名
SdSetupType-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 または、[次へ] ボタンがクリックされたことを示します。 301 = 標準のラジオ ボタンが現在選択されています。 302 = コンパクト ラジオ ボタンが現在選択されています。 303 = カスタム ラジオ ボタンが現在選択されています。
SdSetupType-<#>	szDir	"インストール先" 編集フィールドに入力されたパス
SdShowDlgEdit1-<#>	結果	標準値
SdShowDlgEdit1-<#>	szEdit1	svEdit1 フィールドに入力されたテキスト
SdShowDlgEdit2-<#>	結果	標準値
SdShowDlgEdit2-<#>	szEdit1	svEdit1 フィールドに入力されたテキスト
SdShowDlgEdit2-<#>	szEdit2	svEdit2 フィールドに入力されたテキスト

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SdShowDlgEdit3-<#>	結果	標準値
SdShowDlgEdit3-<#>	szEdit1	svEdit1 フィールドに入力されたテキスト
SdShowDlgEdit3-<#>	szEdit2	svEdit2 フィールドに入力されたテキスト
SdShowDlgEdit3-<#>	szEdit3	svEdit3 フィールドに入力されたテキスト
SdShowFileMods-<#>	結果	標準値
SdShowFileMods-<#>	nSelection	"セットアップで実施する機能を選択してください" ラジオボタン選択: 101 = セットアップに AUTOEXEC.BAT ファイルを変更させる。 102 = AUTOEXEC.NEW ファイルに指定された変更を保存する。 103 = 変更しない
SdShowInfoList-<#>	結果	標準値
SdStartCopy-<#>	結果	標準値
SdWelcome-<#>	結果	標準値
SdWelcomeMaint-<#>	結果	301 = [次へ] ボタンがクリックされたとき、[変更] ボタンが選択されたことを示します。 302 = [次へ] ボタンがクリックされたとき、[修復] ボタンが選択されたことを示します。 303 = [次へ] ボタンがクリックされたとき、[削除] ボタンが選択されたことを示します。
SelectDir-<#>	結果	標準値
SelectDir-<#>	szDir	このダイアログによって選択されたパス
SelectDirEx-<#>	結果	標準値
SelectDirEx-<#>	szDir	このダイアログによって選択されたパス
SelectFolder-<#>	結果	標準値
SelectFolder-<#>	szResultFolder	このダイアログによって選択されたフォルダー

テーブル 17 • ダイアログ データのキー名 (続き.)

ダイアログ	キー名	説明
SetupType-<#>	結果	12 = エンドユーザーが [戻る] ボタンをクリックしました。 または、[次へ] ボタンがクリックされたことを示します。 301 = 標準のラジオ ボタンが現在選択されています。 302 = コンパクト ラジオ ボタンが現在選択されています。 303 = カスタム ラジオ ボタンが現在選択されています。
SprintfBox-<#>	結果	1 = ユーザーが [OK] ボタンをクリックしました。
Welcome-<#>	結果	標準値

SdFinishEx の特殊処理

SdFinishEx は、システム変数 *BATCH_INSTALL* の値に応じて **SdFinish** または **SdFinishReboot** のどちらかを呼び出します。このため、*SdFinishEx* を応答ファイルで処理することはできません。*SdFinishEx* を呼び出すスクリプトのサイレント インストールを作成するには、次のようなスクリプト コードを利用します:

```
if (MODE = NORMALMODE) then
   SdFinishEx (szTitle, szMsg1, szMsg2, szOpt1, szOpt2, bvOpt1, bvOpt2);
else
   if !BATCH_INSTALL then
      /* 再起動が必要ないサイレント インストール中に
      bvOpt1 と bvOpt2 に持たせる値を設定します。
      例: */
      bvOpt1 = FALSE;
      bvOpt2 = TRUE;
   else
      /* サイレント インストールが再起動を必要なとき、
      即再起動する場合、
      次のようなシステム関数を呼びます:*/
      System (SYS_BOOTMACHINE);
   endif;
endif;
```

サンプル応答ファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

次の応答ファイルは、サイレント InstallShield インストール用の .iss ファイルです。

```
[InstallShield Silent]
Version=v7.00
File=Response File

[Application]
```

```
Name=InstallShield
Version=10.50.000
Company=ソフトウェア会社名

[{77AB941D-5876-11D4-A4A2-006067620F66}-DlgOrder]
Dlg0={77AB941D-5876-11D4-A4A2-006067620F66}-SdBitmap-0
Count=8
Dlg1={77AB941D-5876-11D4-A4A2-006067620F66}-Welcome-0
Dlg2={77AB941D-5876-11D4-A4A2-006067620F66}-SdRegisterUser-0
Dlg3={77AB941D-5876-11D4-A4A2-006067620F66}-AskDestPath-0
Dlg4={77AB941D-5876-11D4-A4A2-006067620F66}-SetupType-0
Dlg5={77AB941D-5876-11D4-A4A2-006067620F66}-SelectFolder-0
Dlg6={77AB941D-5876-11D4-A4A2-006067620F66}-SdStartCopy-0
Dlg7={77AB941D-5876-11D4-A4A2-006067620F66}-SdFinish-0

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdBitmap-0]
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-Welcome-0]
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdRegisterUser-0]
szName=John Doe
szCompany=ソフトウェア会社名
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-AskDestPath-0]
szPath=C:¥Program Files¥製品名バージョン
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SetupType-0]
Result=301
szDir=C:¥Program Files¥製品名バージョン

[{77AB941D-5876-11D4-A4A2-006067620F66}-SelectFolder-0]
szResultFolder=製品名バージョン
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdStartCopy-0]
Result=1

[{77AB941D-5876-11D4-A4A2-006067620F66}-SdFinish-0]
Result=1
bOpt1=1
bOpt2=0
```

サイレント インストールの実行

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

インストールと応答ファイルを作成した後、次の操作を行います:

1. [サポート ファイル/ビルボード] ビューで、(デフォルトで Windows フォルダーに格納されている) 応答ファイルをアドバンス ファイル の下にある Disk1 フォルダーに追加します。
2. リリースをビルドします。インストールに自己展開型実行可能ファイルを作成する場合、リリース ウィザードの [一般オプション] パネルにある "セットアップ コマンドライン" プロパティか、または [リリース] ビューの "セットアップ コマンドライン" プロパティに -s を入力します。

これで、InstallShield Silent を使ってインストールをサイレント モードで実行する準備が整いました。インストールをサイレント モードで実行する際、メッセージが表示されない点に注意してください。代わりに、Setup.log という名前のログ ファイルによって、インストールが正常に終了したかどうかを含むインストール情報がキャプチャされます。ログファイルを確認して、インストールの結果を判断することができます。(一部のインストール初期化エラーでは、ログファイルの名前が Setupexe.log で、インストールがインターネットから実行される場合はそれが SUPPORTDIR に作成され、それ以外の場合は SRCDIR に作成されることがありますのでご注意ください。)

InstallShield Silent を起動するには、-s オプションで Setup.exe を実行します。自己展開型実行可能ファイルを作成した場合、上記手順の 2 で -s オプションを含んでいるため、単に起動するだけです。

InstallShield には、-f1 および -f2 スイッチがあり、応答ファイルの名前と場所、およびログファイルの場所を指定できます。詳細については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

サイレント インストールが正常に終了したかどうかを検証するには、Setup.log の [ResponseResult] セクションの ResultCode 値を確認します。InstallShield は、ResultCode キー名のあとに、適切な戻り値を書き込みます。

Setup.log ファイルを使用してエラーを確認する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

エンドユーザーが Setup.exe を /s 引数で実行した場合に生成される Setup.log は、サイレント インストール ログファイルのデフォルトの名前です。Setup.log は、デフォルトで応答ファイル Setup.iss のあるディレクトリに作成されます。/f1 および /f2 スイッチを Setup.exe に使用することで、Setup.logに異なる名前と場所を指定することができます。

Setup.log ファイルには 3 つのセクションがあります。最初のセクション [InstallShield Silent] は、サイレント インストールで使用した InstallShield Silent のバージョンを識別します。また、ファイルがログファイルであることも表します。

2 番目の Application セクションは、インストールされたアプリケーションの名前とバージョン、および会社名を識別します。

3 番目の ResponseResult セクションには、サイレント インストールが成功したかどうかを示す結果コードが含まれます。[ResponseResult] セクションの ResultCode キー名に、整数値が割り当てられます。インストールは、ResultCode キーに以下の戻り値のいずれかを設定します。

プロジェクト: *InstallScript MSI インストールでは、応答ファイルの読み取りまたは書き込み時に初期化プロセスはありません。そのため、InstallScript MSI インストールで発生する可能性のあるエラーは、サイレント インス*

トールの場合は -3、記録インストールの場合は -6 です。以下の表は、該当する適切なプロジェクトの種類を示します。

テーブル 18 • Setup.log ファイルの値を返す

結果コード	プロジェクトの種類	説明
0	InstallScript、InstallScript MSI	成功
-1	InstallScript	一般エラー。
-2	InstallScript	モードが間違っています。
-3	InstallScript、InstallScript MSI	必要なデータが Setup.iss ファイルに見つかりません。
-4	InstallScript	実行に十分なメモリがありません。
-5	InstallScript	ファイルが存在しません。
-6	InstallScript、InstallScript MSI	応答ファイルに書き込むことができません。
-7	InstallScript	ログファイルに書き込むことができません。
-8	InstallScript	InstallShield Silent 応答 (.iss) ファイルのパスが間違っています。
-9	InstallScript	リストの種類が無効です (文字列 /番号)。
-10	InstallScript	データ 型が無効です。
-11	InstallScript	セットアップ中に不明のエラーが発生しました。
-12	InstallScript	ダイアログ ボックスに不具合があります。
-51	InstallScript	指定したフォルダーを作成することができません。
-52	InstallScript	指定したファイルまたはフォルダーにアクセスできません。
-53	InstallScript	選択したオプションは無効です。

サイレント インストールが成功した場合、Setup.log ファイルの内容は次のようになります。

```
[Application]
Name=Sample App 3000
Version=1.00.0000
Company=Sample Software Corporation
Lang=0409

[ResponseResult]
```

ResultCode=0

InstallScript MSI プロジェクトの MSI サイレント モード インストール

プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

Setup.exe を使用せずに InstallScript MSI プロジェクトをサイレントでインストールする必要がある場合、MSI サイレント モードを使用することができます。

InstallScript MSI プロジェクトの MSI サイレント インストールを起動する

MSI サイレント インストールは、次のような場合に起動されます。

- コマンドラインで次のように入力してインストールを起動した場合：

  msiexec product.msi /qn

- アドバタイズされたショートカットからインストールを起動した場合。
- 要求時インストールでインストールを起動した場合。
- アップデート パッケージ実行中に製品が削除された場合。

従来のサイレントモードとは異なり、MSI サイレントモードはスクリプトで提供される通常の論理に従いません。単に InstallExecuteSequence テーブルを通して実行されます。(このテーブルを表示するには、ダイレクト エディターに移動します。)その結果、次を含む一部のイベントは呼び出されません。

- すべての UI インストール イベント—OnFirstUIBefore と OnFirstUIAfter
- すべての機能イベント

OnMsiSilentInstall イベント

このイベントの機能

InstallScript MSI ベースのインストールでは、製品がターゲット システムにインストールされていない場合、InstallShield は OnMsiSilentInstall イベント ハンドラーを起動します。これは、インストールがアドバタイズされたショートカットからアクティベートされた場合、または、インストールが次のコマンドラインを使用して起動された場合に発生します。

msiexec product.msi /qn

MSI サイレントインストールモードをサポートする場合は、OnMsiSilentInstall イベントを上書きする必要があります。これにより、通常 OnFirstUIBefore、OnFirstUIAfter、および機能イベント ハンドラーで実行されるタスクを実行できます。

イベントの上書き

デフォルトでは、**OnMsiSilentInstall** によりメッセージが表示され、インストールは中止されます。このイベント ハンドラーは、この関数の独自の実装を作成することにより上書きできます。次にこの機能のプロトタイプを示します。

external prototype OnMsiSilentInstall(HWND hInstall);

ここで、hInstall はインストールのハンドラーです。

最も簡単な方法は、インストールが中止しないようにこのイベントの空の本文を実装することです。例:

```
function OnMsiSilentInstall(hInstall)
begin
  //何もしないで、インストールを継続します。
end;
```

MSI サイレント インストール、およびアドバタイズされたショートカットのアクティベーションにより **OnMsiSilentInstall** が再び起動されます。オンデマンド インストール、自動修復、およびアンインストール モードでは呼び出されません。

サイレント インストールが実行されているモードの検出

InstallShield スクリプトから従来のサイレント インストールが実行されているかどうかを検出するには 次を使用します。

```
if (MODE = SILENTMODE)
```

InstallShield スクリプトから MSI サイレント モード インストール (/q、アドバタイズ、自動修復、アンインストールまたはオンデマンド インストールを含む) が実行されているかどうかを検出するには 、Windows Installer API 関数の **MsiGetProperty** を使用して Windows Installer プロパティ ISSETUP_UISEQUENCE_PROCESSED を確認します。このプロパティが設定されていない場合は、サイレント インストールです。(InstallUISequence が実行されないことを示します。)

コマンドラインを使ったサイレント アンインストール (InstallScript MSI のみ)

プロジェクト: *この情報は、InstallScript MSI プロジェクトに適用します。*

InstallScript MSI インストールを使ってインストールされた製品をサイレント アンインストールするには、応答ファイルを使います。

タスク ***応答ファイルを使ってアンインストールを実行するには、以下の手順に従います:***

1. Setup.exe を /r 引数と共に実行してアンインストール用の応答ファイル (.ss) を準備します。

   Setup.exe /r

2. 次の行をコマンドラインに入力します (イタリック体で表示されているアイテムは、製品のアンインストールに固有のデータを示します):

   Setup.exe /s /f1"*FullPath¥YourResponseFile.iss*"

スイート インストールをコマンドラインから最小ユーザー インターフェイスを使って実行する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

進行状況パネルを表示するのみの最小 UI モードでスイート インストールを実行するコマンドライン パラメーターを使用できます。

スイート インストールを最小 UI モードで実行するには、コマンドラインで /passive パラメーターを使用します:

```
Setup.exe /passive
```

最小 UI モードを使ってアンインストールするには、次のコマンドを使用します:

```
Setup.exe /passive /remove
```

コマンドラインからアップデートのダウンロードを行わずにスイートのサイレント インストールを実行する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

/skipupdate というコマンドライン パラメーターを使ってスイート/アドバンスト UI サイレント インストールを構成し、自動的にユーザーによるダウンロード可能なアップデートの適用を強制する代わりに、インストーラーがアップデートのダウンロードをスキップしてサイレント インストールを続行するように指示します。

ユーザーによるアップデートのダウンロードを強制せずにスイート/アドバンスト UI サイレント インストールを実行するには、次のコマンドラインで /skipupdate パラメーターを使用します:

```
Setup.exe /silent /skipupdate
```

このコマンドライン パラメーターは、サイレント モードによるスイート/アドバンスト UI インストール機能を強化します。

プロジェクトの検証

プロジェクトを検証するには、一連の内部整合性評価プログラム (ICE) のルールをインストールまたはマージ モジュール パッケージに適用します。ICE はアクションが正常に実行するために必要な有効データベースがパッケージに含まれているかどうかを判断するのに役立ちます。インストール プロジェクトがこれらの ICE のいずれかに違反する場合、InstallShield は違反の対象となった ICE 規則をレポートし、問題を解決するための追加情報を提供します。

InstallShield では、Microsoft 作成による多くの ICE を利用できます。また、Revenera が作成したカスタム InstallShield ICE (ISICE) も、InstallShield インストールとマージ モジュール検証スイートの一部で利用することができます。ISICE は、パッケージを Windows ベースのインストール用のベスト プラクティスに対して検証するのに使用できます。

Windows ロゴ: *インストールがマイクロソフトの Windows ロゴ プログラムの要件を満たしているかどうかを判別するために、プロジェクトの検証が役に立つ場合があります。従って、Windows ロゴの認証に関心があるユーザーは、**InstallShield 検証スイート – Windows 8** および**完全 MSI 検証スイート**を使ってインストール パッケージの検証を行うことをお勧めします。InstallShield でマージ モジュールを作成した場合、**InstallShield マージ モジュー***

ル検証スイート – Windows 8 およびマージ モジュール検証スイートを利用して、マージ モジュールを検証することができます。詳細については、「Windows ロゴ プログラムの要件」を参照してください。

Windows ロゴの取得に関する詳細は、MSDNを参照してください。

***エディション:** InstallShield Premier Edition には、次の一連の検証スイートが含まれています。*

InstallShield Premier Edition には、次の一連の検証スツールが含まれています:

- ***InstallShield 仮想化適合性スイート**—このスイートに含まれる ISVICE 検証ツールを使って、Microsoft App-V 4.x、Microsoft App-V 5.x、Microsoft Server App-V、VMware ThinApp、および Citrix XenApp との適合性を確認することで、製品が仮想化に必要な準備が整っているかどうかを判断することができます。顧客に仮想バージョンの提供を考慮する場合に、検証スイートを使用して、製品をどのようにビルドするか豊富な情報に基づいた意思決定を行うことができます。*
- ***InstallShield ベスト プラクティス スイート**—インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの ISBP 検証ツールによってアラートされます。*
- ***InstallShield MSIX/UWP アプリ適合性スイート**—このスイートに含まれる ISUWP 検証ツールは、.msi パッケージで Windows アプリ パッケージ フォーマットに適さないアイテムの存在をスキャンして、既知の Windows アプリのバリエーション (ユニバーサル アプリ、デスクトップ ブリッジ、Windows ストア、WSA、および Nano Server) に対する適合性を示すレポートを提供します。*

InstallShield では、アップグレード検証とパッチ検証のエンジンも提供しています。このエンジンへは、アップグレード検証ウィザードを通してアクセスできます。

インストール パッケージまたはマージ モジュールを検証する

InstallShield 内からインストール パッケージまたはマージ モジュールの検証方法には 2 種類あります:

- InstallShield を構成して、リリースのビルド時に毎回インストール パッケージとマージ モジュールの検証を行うようにすることができます。これは、デフォルトで無効になっています。詳細については、「ビルド時に検証を行うかどうかを指定する」を参照してください。

  ビルド時に検証を行う場合、複数の検証スイートを指定できます。

- ビルド済みのリリースに対してオンデマンドで検証を行うことができます。詳しくは、「インストール パッケージまたはマージ モジュールをオン デマンドで検証する」をご覧ください。

  オンデマンドで検証を行う場合、検証スイートは 1 回の実行につき 1 つのみ指定できます。

***ヒント:** 代わりの方法として、ビルドが完了したあと、ISCmdBld.exe を使ってコマンドラインからインストール パッケージやマージ モジュールを検証することもできます。詳細については、「ISCmdBld.exe」を参照してください。*

***プロジェクト:** MSI データベース プロジェクトまたは MSM データベース プロジェクト場合、パッケージの検証はオンデマンドでのみ行うことができます。*

メモ: インストールまたはマージ モジュールに適用する検証の警告を見るには、オンデマンドで検証を実行する必要があります。このタイプの検証メッセージは、ビルド時に検証を実行しても見ることができません。InstallShield では、どちらの検証方法でも、他のタイプの検証メッセージ (エラーおよび失敗) が報告されます。異なるタイプの検証メッセージについては、「検証結果を表示する」を参照してください。

ビルド時に検証を行うかどうかを指定する

InstallShieldでは、リリースが正常にビルドされるたびにインストール パッケージおよびマージ モジュールを検証するかどうかを指定することができます。InstallShield ではまた、検証に使う検証スイートも指定できます (複数可)。

タスク ***検証の設定を構成するには、***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[検証]** タブをクリックします。
3. InstallShield がビルド時に行う検証の種類に当てはまるチェック ボックスを選択します。ビルド時ではなくオンデマンドで検証を行う場合は、チェック ボックスの選択を解除します。
4. **[次を使用して検証を実行する]** チェック ボックスまたは **[次を使用してマージモジュールを検証する]** チェック ボックスを選択したとき、実行する検証タイプも選択します (複数可)

   新しい、またはカスタム検証ツール (.cub file) を追加するには、**[参照]** ボタンをクリックして、適切な検証ツールを作成します。

メモ: インストールまたはマージ モジュールに適用する検証の警告を見るには、オンデマンドで検証を実行する必要があります。このタイプの検証メッセージは、ビルド時に検証を実行しても見ることができません。InstallShield では、どちらの検証方法でも、他のタイプの検証メッセージ (エラーおよび失敗) が報告されます。

検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する

InstallShieldでは、インストール パッケージおよびマーモジュールの検証に使用する ICE、ISICE、ISVICE、ISUWP および ISBP の項目をカスタマイズすることができます。

タスク ***検証中に実行する ICE、ISICE、ISVICE、ISUWP および ISBP を指定するには、以下の手順に従います:***

1. **[ツール]** メニューから **[オプション]** を選択します。**Options** ダイアログ ボックスが開きます。
2. **[検証]** タブをクリックします。
3. **[カスタマイズ]** ボタンをクリックします。**[検証設定のカスタマイズ]** ダイアログ ボックスが表示されます。
4. **[カスタマイズする CUB ファイルを選択]** リストで、カスタマイズするスイートをクリックします。
5. ICE 一覧で、インストール パッケージまたはマージ モジュールの評価に使用する各検証ツールのチェック ボックスを選択します。検証に使用しない検証ツールの各チェック ボックスの選択を解除します。

連続する複数の検証ツールを構成するには、最初のファイルを選択してから SHIFT キーを押したまま最後のファイルを選択します。そのあと、該当する検証ツールのチェック ボックスを選択またはクリアします。

6. [OK] をクリックします。

検証スイートの ICE、ISICE、ISVICE、ISUWP および ISBP のリストをカスタマイズすると、(ビルド時またはオンデマンドで) 検証が実行されるたびに選択された ICE、ISICE、ISVICE、ISUWP または ISBP のみが使用されます。

インストール パッケージまたはマージ モジュールをオン デマンドで検証する

InstallShield では、ビルド プロセスとは切り離してインストール パッケージまたはマージ モジュールを検証することができます。これは、ビルドに成功したあと検証が毎回実行されないように InstallShield を構成してあるけれども、ある時点で製品が Windows ロゴ コンプライアンスまたはベスト プラクティス コンプライアンスに準拠しているかどうかテストするために検証を行う必要がある場合、特に便利です。

タスク ***リリースをオン デマンドで検証するには、以下の手順に従います。***

1. 適切なリリースのビルドを完了します。
2. **[ビルド]** メニューで **[検証]** をポイントして、実行する検証タイプをクリックします。

検証結果を表示する

検証スキャンの結果は、出力ウィンドウの [検証] タブで表示されます。また、この結果は、リリース フォルダー にある ValidationFiles フォルダーに XML ドキュメントとして保存されます。このファイルは、ビルド ディレクトリ、または、[リリース] ビューに移動し、リリースの下で Validations フォルダーを選択すると表示できます。

検証メッセージ

検証メッセージは 3 つに分類されています。

- **エラー**—重複コンポーネント GUID など、データベースに関する問題を説明します。
- **警告**—特定の状況下で発生する可能性がある、データベースに関する問題を説明します。
- **失敗**—検証ツールも実行できなくなるほどの深刻な問題がデータベースにあるときに表示されます。

プロジェクトのスキャン結果が検証メッセージを含む場合、メッセージおよび関連するコードも出力ウィンドウの [タスク] タブに一覧表示されます。

メモ: *インストールまたはマージ モジュールに適用する検証の警告を見るには、オンデマンドで検証を実行する必要があります。このタイプの検証メッセージは、ビルド時に検証を実行しても見ることができません。InstallShield では、どちらの検証方法でも、他のタイプの検証メッセージ (エラーおよび失敗) が報告されます。これらの 2 つの検証方法については、「インストール パッケージまたはマージ モジュールを検証する」を参照してください。*

ヒント: 出力ウィンドウの [タスク] タブの検証コードをクリックすると、対応するナレッジ ベース記事または HelpNet のトピックを表示することができます。

また、[タスク] タブで検証メッセージをクリックすると、その検証メッセージに対応するダイレクト エディター の領域にすぐ移動することができます。この機能は、ICE、ISICE、ISVICE、ISUWP および ISBP で提供されています。

ICE

検証ツールにより、プロジェクトが各内部整合性評価プログラム (ICE) に準拠しているかどうかが確認されます。これらの ICE は、Msival2.exe (Microsoft Windows Installer Platform SDK の一部) が、インストール およびマージ モジュール パッケージの Windows ロゴ コンプライアンスを検証するときに使用されます。

インストール プロジェクトまたはマージ モジュール プロジェクトがこれらの ICE のいずれかに違反する場合、InstallShield は違反の対象となった ICE 規則をレポートし、問題を解決するための追加情報を提供します。

特定の ICE については、Windows Installer ヘルプ ライブラリの「ICE Reference」を参照してください。

ISICE

InstallShield 検証スイートは、プロジェクトが Windows ロゴ プログラムに準拠していることを確認するのに役立つ多数の InstallShield ICE (ISICE) で構成されています。

InstallShield で使用できる ISICE リストは次の表のとおりです。

テーブル 19 ・ ISICE

ISICE	プロジェクトの種類	説明
ISICE01	基本の MSI、InstallScript MSI、MSI データベース	デフォルトでアプリケーションが ProgramFiles またはユーザーの AppData フォルダーにインストールされることを検証します。
ISICE02	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	インストールまたはマージ モジュールに含まれているすべての実行可能ファイルにデジタル署名が行われていることを検証します。
ISICE03	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	ネストされたインストール カスタム アクション (タイプ 7、23、および 39) が存在しないことを検証します。

テーブル 19 • ISICE (続き.)

ISICE	プロジェクトの種類	説明
ISICE04	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	標準 .msi テーブルにカスタム列が追加されていないことを検証します。
ISICE05	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	「MSI」で始まるプロパティまたはカスタム テーブルが存在しないことを検証します (大文字と小文字を区別しません)。
ISICE06	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	インストールに Windows Resource Protection で保護されているファイルが含まれていないことを検証します。
ISICE07	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	MSI コンポーネントが適切なインストールおよびアンインストールにオーサされていることを検証します。
ISICE08	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	遅延カスタム アクションが偽装ビット セットを含んでいないことを検証します。
ISICE09	基本の MSI、InstallScript MSI、MSI データベース	インストールに MsiPatchCertificate テーブル エントリが含まれていることを検証します。
ISICE10	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	各カスタム アクションに [カスタム アクション] ビューの [ヘルプ ファイル パス] 設定で指定されたパスが含まれていることを検証します。
ISICE11	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	.exe ファイルがマニフェストを必要な情報と共に埋め込んだことを検証します。

テーブル 19 • ISICE (続き.)

ISICE	プロジェクトの種類	説明
ISICE12	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	保護されたレジストリ キーが変更されていないことを検証します。
ISICE13	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	.hlp ファイルまたは WinHelp ランタイム ファイルがインストールに含められていないことを検証します。この ICE は現在 ISBP17 です。
ISICE14	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	古い形式の API がインポートされていないことを検証します。この ICE は現在 ISBP18 です。
ISICE15	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	使用されていない API がインポートされていないことを検証します。この ICE は現在 ISBP19 です。
ISICE16	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	16 ビット コンポーネントが 64 ビット システムをターゲットにするインストールで配布されていないことを検証します。
ISICE17	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース	インストールまたはマージ モジュールが再起動を強制しないことを検証します。
ISICE18	基本の MSI、MSI データベース	MsiRMFilesInUse ダイアログが含まれていることを検証します。
ISICE19	基本の MSI、InstallScript MSI、MSI データベース	アップグレードに関するプロパティ、Upgrade テーブルが存在すること、インストールがダウングレードを防ぐことを検証します。

テーブル 19 • ISICE (続き.)

ISICE	プロジェクトの種類	説明
ISICE20	基本の MSI、InstallScript MSI、MSI データベース	マシン全体に渡る設定が権限のないインストールによってインストールされることを検証します。
ISICE21	基本の MSI、InstallScript MSI、MSI データベース	アプリケーションが実行時に VB6 に依存しないことを検証します。
ISICE22	基本の MSI、InstallScript MSI、MSI データベース	アプリケーションがシステム起動時に自動的に開始しないことを検証します。
ISICE23	基本の MSI、InstallScript MSI、MSI データベース	アプリケーションが強く適切な ACL を設定することを検証します。
ISICE24	基本の MSI、InstallScript MSI、MSI データベース	インストールが 16 ビット コンポーネントをインストールしないことを検証します。
ISICE25	基本の MSI、InstallScript MSI、MSI データベース	インストールが Windows ディレクトリまたはサブディレクトリにいずれのファイルもインストールしないことを検証します。
ISICE26	基本の MSI、InstallScript MSI、MSI データベース	アプリケーションが、アクセス許可を適切に制御して、Windows セキュリティ機能をサポートすることを検証します。

ヒント: *場合によって、次の検証エラー メッセージが ISICE で表示されることがあります:*

コンポーネント [2] のファイル [1] が見つかりませんでした。このファイルのコンテンツに対するすべてのテストが無効である可能性があります。

指定されたファイルがないとき、また、関連する ISICE がファイルに対して検証できなかったとき、このエラーが発生します。たとえば、実行可能ファイルが不足しているとき、ISICE11 はファイルが埋め込みマニフェストがあるかどうかを検証することができません。

上記の検証エラー メッセージが表示された場合、不足しているファイルについての問題を解決し、プロジェクトの検証を再度実行します。

ISICE01

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

アプリケーションはデフォルトで、ProgramFiles フォルダーまたは AppData フォルダーにインストールされなくてはなりません。現在のデフォルトの場所は [1] です。

[1] が製品のデフォルト インストールの場所です。

説明

ユーザーは必要な場所に製品をインストールすることができ、デフォルトでファイルが保存される場所には一貫性を持たせる必要があります。

ISICE01 は、製品のデフォルト インストール先が Program Files フォルダーまたは Application Data フォルダーであることを検証します。異なる場所が設定されている場合、検証中にこのエラー メッセージが表示されます。¥

修正アクション

デフォルトの場所を変更します。詳細については、「デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定」を参照してください。

ISICE02

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] がデジタル署名されていません。Windows Vista 以降に配布されたすべての [3] ファイルは、署名が必要です。

[1] は実行可能ファイルの名前、[2] はそのファイルを含むコンポーネントの名前、そして、[3] は署名が必要なファイルの種類です。

説明

ISICE02 は、インストールに含まれるすべての実行可能ファイルがデジタル署名されたことを検証します。これには、次の拡張子を持つファイルを含まれます: exe、dll、ocx、sys、cpl、drv、および scr。インストールに含まれる実行可能ファイルにデジタル署名が行われていない場合、検証中にこのエラー メッセージが表示されます。

修正アクション

インストールに含まれるすべての実行可能ファイルがデジタル署名されたことを確認してください。詳細については、「デジタル署名とセキュリティ」を参照してください。

ISICE03

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

ネストされたカスタム アクション [1] (タイプ [2]) は無効です。

[1] はプロジェクトに含まれるカスタム アクションの名前で、[2] はカスタム アクションの Windows Installer タイプ番号です。

説明

ISICE03 は、インストールにネストされたインストール カスタム アクションが含まれていないことを検証します。Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。

詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。

修正アクション

このエラーを解決するためには、ネストされたインストール カスタム アクションをプロジェクトから削除します。[カスタム アクションとシーケンス] ビュー (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または [カスタム アクション] ビュー (マージ モジュール プロジェクトおよび MSM データベース プロジェクトの場合) におけるネスト インストール カスタム アクションの MSI タイプ番号は、7、23、または 39 です。

ネストされたインストールの代替として、InstallShield 前提条件を作成してから、それをインストール プロジェクトに追加するという方法があります。詳細については、「InstallShield 前提条件を定義する」を参照してください。

Summary List of All Custom Action Types (Windows Installer ヘルプ ライブラリ)

ISICE04

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

標準 MSI テーブル [1] が、schema.msi で定義された MSI スキーマと一致しません。

[1] はプロジェクトに含まれるテーブルの名前です。

説明

ISICE04 は、インストールが標準テーブルにカスタム テーブル列を追加しないことを検証します。標準テーブルに列を追加できるのは、Windows Installer の今後のバージョンだけです。

修正アクション

このエラーを解決するには、エラーメッセージで通知されたテーブルに追加されたカスタム テーブル列すべてを削除します。これは、ダイレクト エディターからテーブルをエクスポートしてテキスト エディターでこれを編集してから、変更済みのテーブルをインポートする方法で行います。

データベース テーブル (Windows Installer ヘルプ ライブラリ)

ISICE05

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

MSI プロパティ [1] が予約された文字で始まっています。MSI プロパティ名を「MSI」(大文字と小文字を区別しません) で始めることはできません。

[1] はプロジェクトに含まれるプロパティの名前です。

メッセージ 2(エラー)

テーブル [1] が予約された文字で始まっています。カスタム テーブル名を「MSI」(大文字と小文字を区別しません) で始めることはできません。

[1] はプロジェクトに含まれるテーブルの名前です。

説明

ISICE05 は、インストールに「**MSI**」(大文字と小文字を区別しません) で始まる名前を持つカスタム プロパティまたはカスタム テーブルが含まれていないことを検証します。「**MSI**」で始まる名前は、今後の新しい標準プロパティおよびテーブル用に予約されています。

修正アクション

プロパティ関連のエラーを解決するためには、[プロパティ マネージャー] を使ってエラー メッセージで通知されたカスタム プロパティの名前を変更します。

テーブル関連のエラーを解決するためには、ダイレクト エディター を使ってテーブルをエクスポートし、エクスポートされた .idt ファイルのテーブル名をテキスト エディターを使って編集してから、再びダイレクト エディターを使って更新済みのテーブルをインポートします。その後、元のカスタム テーブルを削除します。

Naming Custom Tables, Properties, and Actions (Windows Installer ヘルプ ライブラリ)

ISICE06

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のファイル [1] は保護された Windows ファイルです。保護されたファイルを Windows Vista 以降に配布することはできません。

[1] はプロジェクトに含まれるファイル名で、[2] はそのファイルを含むコンポーネントの名前です。

メッセージ 2(警告)

コンポーネント [2] のファイル [1] は、保護された Windows ファイルと同じファイル名前です。たたし、異なるパスへ配布されているように見えます。

[1] はプロジェクトに含まれるファイル名で、[2] はそのファイルを含むコンポーネントの名前です。

説明

ISICE06 は、作成中のインストーラーが Windows Resource Protection (WRP) で保護されているファイルをインストールしないことを検証します。WRP は、Windows Vista 以降のマシン上で保護されたシステム リソースを更新する場合、必ず Microsoft 承認済みのインストールまたはアップデート方法を通してそれが行われるように設計されています。

インストールが保護された Windows ファイルと同じファイル名を持つファイルをインストールするとき、そのファイルがその保護された Windows ファイルと同じ場所にインストールされない場合、警告メッセージ 2 が表示されます。

修正アクション

この検証エラーを解決するには、エラー メッセージで通知されたファイルをプロジェクトから削除します。

検証警告に遭遇したとき、メッセージで通知されたファイルが保護されたファイルかどうかを判別します。

- ファイルが保護されたファイルでなく、他の保護されたファイルと同じ名前を持つファイルである場合、この警告は無視することができます。
- ファイルが保護されたファイルの場合、それをプロジェクトから削除し、この警告を解決します。

製品が新しいバージョンのシステム コンポーネントを必要とする場合、ターゲット マシン上のコンポーネントは Microsoft サービス パックまたは必要なシステム コンポーネントを含む Microsoft 承認済みのインストール パッケージを通して更新されなくてはなりません。システム コンポーネントをリパッケージしてはいけません。

ISICE07

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [1] の Component.ComponentId がヌルです。正しくインストールおよびアンインストールを行うためには、すべてのコンポーネントに有効な ComponentId が必要です。ヌルのままにする場合、その理由を文書化する必要があります。

[1] は [コンポーネント] ビューの [コンポーネント コード] 設定で指定された GUID を持たない、プロジェクトに含まれるコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [1] は COM データを含みますが、複数のファイルがあります。各 COM コンポーネントは、COM ファイルを 1 つだけ含みます。

[1] はプロジェクト内で複数の COM サーバーを含むコンポーネントの名前です。

メッセージ 3(エラー)

コンポーネント [1] には、[デスクトップ] または [スタート] メニュー ショートカットのターゲットとして複数のファイルが含まれています。

[1] は、プロジェクト内でショートカットのターゲットとして指定された複数のファイルを含むコンポーネントの名前です。

メッセージ 4(エラー)

コンポーネント [1] は複数の機能に関連付けられ、次のテーブルで参照されています: [2]。これらの参照には、コンポーネントは 1 つの機能だけに関連付けられている必要があります。

[1] は複数の機能に関連付けられているプロジェクト内のコンポーネントの名前で、Class、Extension、MsiAssembly、PublishComponent、TypeLib テーブルのいずれかによって参照されています。[2] は、このコンポーネントへの参照を含むカンマ区切りのテーブル一覧です。

説明

ISICE07 は、インストールに含まれるコンポーネントがコンポーネント規則に従っていることを検証します。これによって、ある製品のインストーラがターゲット システム上にあるその他の製品に支障をきたさないようにします。これらの規則はまた、アンインストールされる製品に関係のあるすべてのリソースが Windows Installer によって正しく確実に削除されるように支援します。

修正アクション

エラー 1 を解決するには、[コンポーネント] ビューを開いてエラーメッセージで通知されたコンポーネントを選択してから、[コンポーネント コード] 設定をクリックします。画面の右下に表示される [ヘルプ] ペインで、[GUID 生成] をクリックします。[コンポーネント コード] 設定を空白にしておかなくてはならない理由がある場合、Windows ロゴ プログラムの申請書にその内容を記述します。

エラー 2 を解決するには、COM サーバーのどれかを移動させてコンポーネントを分離し、各 COM サーバーがそのコンポーネントのキー パスとなるようにします。

エラー 3 を解決するには、[ショートカット] ビューを使って、エラー メッセージで通知されたコンポーネントに関連付けられたショートカットのうち 1 つを残して、それ以外をすべて削除します。

エラー 4 を解決するには、[セットアップのデザイン] ビューまたは [機能] ビューを使用して、関連付けられている機能のうち 1 つを残して、それ以外をすべて削除します。

ISICE08

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

カスタム アクション [1] は偽装を使用します。Windows Vista 以降上でセットアップを実行する場合、偽装を使うことはできません。

[1] はプロジェクトに含まれるカスタム アクションの名前です。

説明

ISICE08 は、インストールに偽装を使うカスタム アクションが含まれていないことを検証します。スクリプト内実行タイプ設定の [ターミナル サーバーで使用可能] オプションのひとつを選択した場合 ([カスタム アクションとシーケンス] ビュー、[カスタム アクション] ビュー、またはカスタム アクション ウィザードの [応答オプション] パネルで設定)、ターミナル サーバー マシン上でマシン毎のインストールを実行中に、カスタム アクションはエンドユーザーを偽装します。

修正アクション

この検証エラーを解決するためには、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) を開いて、エラー メッセージで通知されたカスタム アクションをクリックします。"スクリプト内実行" 設定の値を "ターミナル サーバーで使用可能" を含まないオプションのひとつに変更します。

ISICE09

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

UAC (ユーザー アカウント制御) パッチには、MsiPatchCertificate テーブル エントリが必要です。

説明

ISICE09 は、インストールに MsiPatchCertificate テーブル エントリが含まれていることを検証します。ベースとなるインストールでは、この テーブルは署名者証明書を提供します。署名者証明書は、後に続くパッチが管理者以外によって適用された場合、またはシステム権限を使用しない管理者によって適用された場合に、デジタル署名を検証するのに使われます。

修正アクション

この検証エラーを解決するためには、プロジェクトに MsiPatchCertificate テーブルを追加します。詳しくは、「非管理者パッチのインストールを準備する」をご覧ください。

ISICE10

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

カスタムアクション [1] (タイプ [2]) について、テーブル [3] に記述されていません。

[1] はプロジェクトに含まれるカスタム アクションの名前で、[2] は Windows Installer タイプ番号です。[3] は、リストされているカスタム アクションのエントリが不足しているテーブルの名前です。

説明

各カスタム アクションの意図された動作は、Windows ロゴ プログラムに基づいて文書化する必要があります。これは、企業環境でシステム管理者が製品を配布する場合、特にカスタム アクションの動作を把握する必要がある場合に便利です。

ISICE10 は、インストールに含まれる各カスタム アクションが文書化されているかどうかについて、CustomAction テーブルの各エントリに対応する ISCustomActionReference テーブル エントリがあることを確認する方法で検証を行います。

修正アクション

この検証エラーを解決するためには、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) を開いてエラー メッセージで通知されたカスタム アクションを選択してから、"ヘルプ ファイル パス" 設定を使ってカスタム アクションの動作を説明する文書へのパスを指定します。"ヘルプ ファイル パス" 設定の値を指定するとき、ISCustomActionReference テーブルにそのカスタム アクション用の行がない場合、InstallShield がこれを追加します。

また、リリースの製品構成を構成して、ビルド時に InstallShield が、各カスタム アクション ヘルプ ファイルのコンテンツを .msi ファイルにストリームするかどうかを指定することができます。詳細については、[リリース] ビューの [全般] タブ (製品構成) の [カスタム アクションのヘルプを含める] 設定の説明を参照してください。

カスタム アクションがインストール プロジェクトによって消費されるマージ モジュールである場合、マージ モジュール プロジェクトの [カスタム アクション] ビューでパスを指定してから、マージ モジュールを再ビルドします。

ISICE11

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] の EXE [1] にマニフェストがありません。

[1] はマニフェストを持たないプロジェクト内の実行可能ファイルの名前で、[2] は、その実行可能ファイルを含むコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] の EXE [1] のマニフェストに requestedExecutionLevel level 設定がありません。

[1] は Windows XP 用に作成されたあと、Windows Vista 以降用に更新されていない可能性があるマニフェストを持つプロジェクト内の実行可能ファイルの名前です。マニフェストは、実行可能ファイルの実行レベルを定義しません。[2] は実行可能ファイルを含むコンポーネントの名前です。

メッセージ 3(エラー)

コンポーネント [2] の EXE [1] のマニフェストに requestedExecutionLevel uiAccess 設定がありません。

[1] は Windows XP 用に作成されたあと、Windows Vista 以降用に更新されていない可能性があるマニフェストを持つプロジェクト内の実行可能ファイルの名前です。マニフェストは、実行可能ファイルの UI アクセシビリティ値を定義しません。[2] は実行可能ファイルを含むコンポーネントの名前です。

メッセージ 4(警告)

コンポーネント [2] の EXE [1] は、マイクロソフトからのウェーバーを必要とする昇格された権限 (highestAvailable) が必要であるとマークされています。

[1] は、昇格された権限が必要であると指定するマニフェストを持つプロジェクト内の実行可能ファイルの名前で、[2] は、その実行可能ファイルを含むコンポーネントの名前です。

メッセージ 5(警告)

コンポーネント [2] の EXE [1]は、マイクロソフトからのウェーバーを必要とする昇格された権限 (requireAdministrator) が必要であるとマークされています。

[1] は、昇格された権限が必要であると指定するマニフェストを持つプロジェクト内の実行可能ファイルの名前で、[2] は、その実行可能ファイルを含むコンポーネントの名前です。

メッセージ 6(警告)

コンポーネント [2] の EXE [1] は、マイクロソフトからのウェーバーを必要とする uiAccess が必要であるとマークされています。

[1] は、より上位の権限を使用して情報をデスクトップ上の他のウィンドウ上に渡すために、実行可能ファイルが UI 保護レベルをバイパスできることを指定しているマニフェストを持つプロジェクト内の実行可能ファイルの名前です。このとき、uiAccess の値には True が設定されています。[2] は実行可能ファイルを含むコンポーネントの名前です。

説明

ISICE11 は、インストール内のすべての .exe ファイルに実行レベルを定義する埋め込みマニフェストがあることを検証します。実行レベルは、マニフェストで次のように定義されています:

```
<requestedExecutionLevel
  level="asInvoker"
  uiAccess="false"/>
```

レベル属性の他の有効な値は、highestAvailable と requireAdministrator です。レベルの値が highestAvailable または requireAdministrator に設定した場合、マイクロソフトからのウェーバーを申請して、Windows ロゴ証明書を取得する必要があります。

uiAccess 属性は、より上位の権限を使用して情報をデスクトップ上の他のウィンドウ (例、オンスクリーン キーボード) 上に渡すために、実行可能ファイルが UI 保護レベルをバイパスすることができるかどうか示します。この属性は、UI アクセシビリティ アプリケーションのときのみ True に設定してください。UI アクセシビリティの値を True に設定した場合、マイクロソフトからのウェーバーを申請して、Windows ロゴ証明書を取得する必要があります。

修正アクション

検証エラー (メッセージ1 から 3) を解決するには、インストール内の実行可能ファイルを、レベルの値と uiAccess の値が適切に設定されている埋め込みマニフェストを持つ実行可能ファイルで置き換えます。

検証警告 (メッセージ 4 から 6) は、製品に Windows ロゴ プログラムの証明を受けたいが、requestedExecutionLevel 要素を、現在実行可能ファイルのマニフェストで定義されているように保持したい場合、場合によって、マイクロソフトから要件免除を取得する必要があることを通知します。

ISICE12

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のレジストリ キー [1] は保護されたレジストリ キーです。保護されたレジストリ キーは、Windows Vista 以降上で変更されない可能性もあります。

[1] は Trusted Installer グループのみが Windows Vista 以降のシステム上で変更できるレジストリ キーで、[2] はそのレジストリ キーを含むコンポーネントの名前です。

説明

ISICE12 は、インストールが Windows Vista 以降のシステム上で Windows リソース保護 (WRP) によって保護されているレジストリ キーを変更しようとしないことを検証します。Trusted Installer グループのみが変更できるレジストリ キーは WRP によって保護されています。

修正アクション

この検証エラーを解決するには、プロジェクトから保護されているレジストリ キーを削除するか、またはそのレジストリ キーを WRP で保護されていないレジストリの一部に移動します。

Using Windows Installer and Windows Resource Protection (Windows Installer ヘルプ ライブラリ)

ISICE16

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、16 ビット ファイルです。64 ビット システムをターゲットするインストールには、16 ビット ファイルが含まれている可能性があります。

[1] は、プロジェクトに含まれる 16 ビット ファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。

説明

インストールが 64 ビット プラットフォームをターゲットにする場合、ISICE16 は、インストールに 16 ビット ファイルが含まれていないことを検証します。64 ビット プラットフォームは 16 ビット ファイルをサポートしません。

修正アクション

この検証エラーを解決するには、この 16 ビット ファイルを 64 ビット または 32 ビット ファイルで置き換えます。

Windows Installer on 64-bit Operating Systems (Windows Installer Help Library)

ISICE17

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

シーケンス [1] に [2] アクションが含まれています。[2] アクションを使用するには、マイクロソフトからのウェーバーが必要です。

[2] はプロジェクトに含まれるアクションの名前で、[1] は、そのアクションを含むシーケンスの名前です。

メッセージ 2(エラー)

コントロール [1] (ダイアログ [2] 内) は [3] アクションを実行します。[3] アクションを使用するには、マイクロソフトからのウェーバーが必要です。

[3] は、プロジェクトに含まれるアクションの名前です。[2] は、アクションを起動するイベントを含むコントロールの名前です。[2] は、コントロールを持つダイアログの名前です。

説明

ISICE17は、インストールがアクションを使用して、再起動を強制しないことを検証します。現在、検証の対象となるアクションは、ForceReboot のみです。

修正アクション

エラーは、製品に Windows ロゴ プログラムの認定を受けたいが、インストールに指定したアクションを保持したい場合、場合によってマイクロソフトから要件免除を取得する必要があることを通知します。

この検証エラーを解決するには、アクションをメッセージで表示されたシーケンスまたはダイアログから削除します。

プロジェクトが、システムの再起動数を最小化する MsiRMFilesInUse ダイアログを含み、アプリケーションが Restart Manager API を使用するように適切にインストルメントされていることを確認してください。詳細については、「Windows Vista 以降のシステムの再起動を最小限にする」を参照してください。

ForceReboot Action (Windows Installer ヘルプ ライブラリ)

ISICE18

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

ファイル [1] が Dialog テーブルにありません。

[1] は、プロジェクトに必要なダイアログの名前です。

メッセージ 2(エラー)

ダイアログ [1] のタイトルに、[2] または [ProductName] が含まれていません。

[1] はプロジェクト内のダイアログ の名前で、[2] は [ProductName] 変数の値です。

説明

ISICE18 は、インストールが MsiRMFilesInUse ダイアログを含み、ダイアログのタイトル バーに製品名が含まれていることを検証します。

修正アクション

このエラーを解決するには、MsiRMFilesInUse ダイアログがプロジェクトにあることを確認します。すべての基本の MSI プロジェクトには、デフォルトでこのダイアログが含まれています。このダイアログの詳細は、「Windows Vista 以降のシステムの再起動を最小限にする」を参照してください。

ダイアログを基本の MSI プロジェクトに再度追加する場合、このダイアログを含む別の基本の MSI プロジェクトを開くか、または新しい基本の MSI プロジェクトを作成します。次いで、ダイアログを、それを必要とするプロジェクトにエクスポートします。詳細については、「他のプロジェクトへダイアログをエクスポートする」を参照してください。

エラー 2 を解決するには、製品名または [ProductName] プロパティを MsiRMFilesInUse ダイアログの [キャプション] プロパティに追加します。

Using Windows Installer with Restart Manager (Windows Installer ヘルプ ライブラリ)

ISICE19

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

テーブル [1] が、インストール パッケージにありません。

[1] は、パッケージに必要なテーブルの名前です。

メッセージ 2(エラー)

UpgradeCode [1] は、適切にスケジュールされたタイプ 19 カスタム アクションを使用してダウングレードの検知および防止を行いません。パッケージはダウングレードをできないようにする必要があります。

[1] は、UpgradeCode プロパティの値です。

メッセージ 3(エラー)

プロパティ [1] が SecureCustomProperties にリストされていません; これが原因で、このプロパティがダウングレードの防止のために正しく使用されなくなる可能性があります。

[1] は、プロパティの名前です。

メッセージ 4(エラー)

プロパティ [1] には、有効な GUID を指定してください。[2] は有効な GUID ではありません。

[1] は、値に有効な GUID が必要なプロパティの名前です。[2] は、現在このプロパティに使用されている無効な値です。

メッセージ 5(エラー)

列 [1] (Upgrade テーブル内) には、有効な GUID が必要です。[2] は有効な GUID ではありません。

[1] は、Upgrade テーブルの列の名前です、[2] は、指定された列のエントリの値です。

メッセージ 6(エラー)

プロパティ [1] には、有効な数値のバージョンをメジャー.マイナー.ビルド というフォーマットで指定してください。[2] は、このフォーマットに沿っていないバージョンです。

[1] は、値にメジャー.マイナー.ビルド というフォーマットのバージョン番号を指定する必要のあるプロパティです。[2] は、現在このプロパティに使用されている無効な値です。

メッセージ 7(エラー)

Upgrade テーブル内の列 [1] には、有効な数値のバージョンをメジャー.マイナー.ビルド というフォーマットで指定してください。[2] は、このフォーマットに沿っていないバージョンです。

[1] は、Upgrade テーブルの列の名前です、[2] は、指定された列のエントリの値です。

メッセージ 8(エラー)

プロパティ [1] に、ヌル値および空白は指定できません。

[1] は、値がヌル値または空白になっているプロパティの名前です。

メッセージ 9(エラー)

プロパティ [1] (Upgrade テーブル内) に、ヌル値および空白は指定できません。

[1] は、Upgrade テーブルの列の名前です。

メッセージ 10(エラー)

プロパティ [1] を指定してください。[2] はデフォルトの値です。

[1] は、カスタマイズされていないプロジェクト内のプロパティの名前です。[2] は、InstallShield がこのプロパティに使用するデフォルト値です。

メッセージ 11(エラー)

プロパティ [1] には、有効な数値の LANGID が必要です; [2] は、数値の LANGID ではありません。

[1] は、値に有効な言語 ID が必要なプロパティの名前です。[2] は、現在このプロパティに使用されている無効な値です。

説明

ISICE19 は、アップグレードに関するプロパティ、および Upgrade テーブルが存在することを検証します。ISICE19 はまた、インストールが以前のパッケージを新しいパッケージで上書きインストールするのを防ぐかどうかを判断するのに役立ちます。

修正アクション

エラー 1 と 2 を解決するには、Upgrades ビューにメジャー アップグレード アイテムがあること、製品の現在のバージョンによって将来のバージョンが上書きインストールされないようにメジャー アップグレード アイテムが適切に構成されていること、および、製品に適切に構成、スケジュールされたタイプ 19 のカスタム アクションが含まれていることを確認してください。詳しい手順については、「現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ」を参照してください。詳細は、Windows Installer ヘルプ ライブラリの「Preventing an Old Package from Installing Over a Newer Version」を参照してください。

エラー 3 を解決するには、指定されたプロパティを SecureCustomProperties プロパティの値に追加します。プロパティの値は、[プロパティ マネージャー] ビューを使用して設定することができます。複数の値を追加するには、それぞれをセミコロンで区切ります。詳しい情報は、Windows Installer ライブラリの「SecureCustomProperties Property」を参照してください。

エラー 4 と 5 を解決するには、既存の値を有効な GUID で置き換えます。値を変更するには、[一般情報] ビューを開きます。次いで、エラー メッセージにリストされている設定をクリックします。有効な GUID を入力します。新しい一意の GUID を使用するには、この設定で **[新しい GUID の生成]** ボタンをクリックします。詳細については、「GUID」を参照してください。

エラー 6 と 7 を解決するには、指定した値を有効なバージョン番号で置き換えます。出力ウィンドウのエラーメッセージをクリックすると、無効な値があるダイレクト エディターの場所に移動することができます。バージョン番号には、**aaa.bbb.ccccc** という形式で、数字のみを利用します。ここで、 *aaa* はメジャーバージョン番号、*bbb* はマイナーバージョン番号、そして *ccccc* はビルド番号を表します。*aaa* と *bbb* の最大値は 255 です。*ccccc* の最大値は、65,535 です。詳細については、「製品バージョンを指定する」を参照してください。

エラー 8 を解決するには、指定されたプロパティに値を入力します。プロパティの値は、[プロパティ マネージャー] ビューを使用して設定することができます。

エラー 9 を解決するには、指定された列に値を入力します。ダイレクト エディター ビューを使用して、値を設定することができます。出力ウィンドウのエラー メッセージをクリックすると、修正が必要な値があるダイレクト エディターの場所に移動することができます。各 Windows Installer テーブルについて学ぶには、Windows Installer ヘルプ ライブラリの「Database Tables」を参照してください。

エラー 10 を解決するためには、指定されたプロパティのデフォルト値を適切な値で置き換えます。この値は、プロパティ マネージャーを使用して変更することができます。

エラー 11 を解決するためには、指定されたプロパティの既存の値を有効な言語 ID で置き換えます。

Upgrade Table (Windows Installer ヘルプ ライブラリ)

ISICE20

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

行 [1] (テーブル [2]): ルート [3] のレジストリ キーをインストール、変更、または削除するには、昇格された権限が必要です。ただし、パッケージは昇格された権限を要求しません。

[1] は、プロジェクトのレジストリの変更に関する情報を含む行の名前です。[2] は、レジストリに関するデータを含むテーブルの名前です。[3] は、レジストリ データの定義済みルート キーで、指定された行に Root 列でリストされているものと同じです。

メッセージ 2(エラー)

行 [1] (テーブル [2]): ディレクトリ [3] のファイルをインストール、変更、または削除するには、昇格された権限が必要です。ただし、パッケージは昇格された権限を要求しません。

[1] は、エンドユーザーがシステム権限がないとアクセスできないディレクトリに関する情報を含む行の名前です。[2] は、レジストリ データを含むテーブルの名前です。[3] は、ディレクトリの名前です。

メッセージ 3(エラー)

環境変数 [1] は、システム環境変数ですが、このパッケージは昇格された権限を要求しません。

[1] は、プロジェクトに含まれているシステム環境変数の名前です。

説明

[概要情報] ビューの "管理者権限が必要" 設定がプロジェクトで [いいえ] に設定されている場合、ISICE20 は、インストールが管理者権限を必要とするタスク (システム レジストリまたはフォルダーの場所への書き込みなど) を実行しようとしないことを検証します。

修正アクション

ISICE20 エラーを解決するには、以下のいずれかを実行します。

- [一般情報] ビューの "管理者権限が必要" 設定で [はい] を選択する。詳細については、「概要情報ストリーム データを入力する」を参照してください。
- システムの場所でデータをインストール、変更、または削除しないようにプロジェクトに変更を加える。たとえば、プロジェクトでレジストリ キーが HKEY_LOCAL_MACHINE に追加される場合、[レジストリ] ビューを使用して、そのレジストリ キーを HKEY_CURRENT_USER に移動します。プロジェクトで、システム環境変数が追加または変更される場合、[環境変数] ビューを使用して、環境変数の [種類] 設定を [システム] から [ユーザー] へ変更します。

Authoring Packages without the UAC Dialog Box (Windows Installer ヘルプ ライブラリ)
Installing a Package with Elevated Privileges for a Non-Admin (Windows Installer ヘルプ ライブラリ)
Guidelines for Packages (Windows Installer ヘルプ ライブラリ)

ISICE21

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、VB6 ランタイムに依存します。

[1] は、Visual Basic 6 ランタイムを必要とするファイルの名前です。[2] はそのファイルを含むコンポーネントの名前です。

説明

ISICE21 は、製品が Visual Basic 6 ランタイムに依存しないことを検証します。

修正アクション

この検証エラーを解決するために、次のオプションを考慮してください:

- 製品が現在使用していないコンポーネントにファイルが含まれている場合、プロジェクトからそのファイルとコンポーネントを削除します。ファイルの作成には、Visual Basic .NET などの比較的新しい言語の使用を考慮してください。
- プロジェクトからファイルとそのコンポーネントを削除してから、Visual Basic 6 ランタイムに依存しない新しいバージョンのファイルとコンポーネントと置換します。

プロジェクトでこれらのオプションのいずれも不可能な場合、インストールが Windows ロゴ プログラムの認証を受けることはできません。

ISICE22

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のショートカット [1] がスタートアップ フォルダーをターゲットとします。

[1] はプロジェクトに含まれるショートカット名で、[2] はそのファイルを含むコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] のレジストリ キー [1] は Run キーを設定します。

[1] はプロジェクトに含まれるレジストリ キーで、[2] はそのショートカットを含むコンポーネントの名前です。

説明

ISICE 22 は、ターゲット システムが開始するときに製品が自動的に開始しないことを検証します。

エラー メッセージ 1 は、インストールが [スタート] メニューのスタートアップ フォルダー内にショートカットを作成する場合に発生します。

エラー メッセージ 2 は、インストールが以下のようなレジストリの場所で Run キーを設定する場合に発生します。

- HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion
- HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion
- HKEY_LOCAL_MACHINE¥Software¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion
- HKEY_CURRENT_USER¥Software¥Wow6432Node¥Microsoft¥Windows¥CurrentVersion

修正アクション

この検証エラーを解決するには、ショートカットまたはレジストリ キーおよび、適切な場合はコンポーネントをプロジェクトから削除します。

ISICE23

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] の LockObject [1] は、セキュリティで保護されていないアクセス許可を設定します。

[1] は、アクセス許可が構成されるファイルの名前、ディレクトリ、またはレジストリ キーです。[2] はそのファイル、ディレクトリ、またはレジストリ キーを含むコンポーネントの名前です。

説明

ISICE23 は、ファイル、ディレクトリ、およびレジストリ キーに対する不適切なアクセス許可の構成によって、インストールがターゲット システムをセキュリティ上の危険にさらさないことを検証します。

たとえば、インストールによってインストールされる特定のフォルダーについて、すべてのユーザーに対する書き込みアクセス許可が割り当てられる場合、ISICE23 はエラーをトリガして、インストールによってターゲット システムがさらされるセキュリティの脆弱性をアラートします。

修正アクション

この検証エラーを解決するには、エラーに記述されているファイル、ディレクトリ、またはレジストリ キーに強力かつ適切なアクセス許可を使用します。詳細については、次を参照してください。

- ファイルとフォルダーのアクセス許可を構成する
- レジストリ キーのアクセス許可を構成する

ISICE24

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、16 ビット ファイルです。Windows 8 システムをターゲットするインストールには、16 ビットファイルを含むことはできません。

[1] は、インストール内の 16 ビット ファイルの名前です。[2] はそのファイルを含むコンポーネントの名前です。

説明

ISICE24 は、インストールに 16 ビット ファイルが含まれていないことを検証します。Windows の 64 ビット バージョンは 16 ビット ファイルをサポートしません。

修正アクション

この検証エラーを解決するには、この 16 ビット ファイルを 64 ビット または 32 ビット ファイルで置き換えます。

ISICE25

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [1] のインストール先は、Windows フォルダーまたはそのサブフォルダーの 1 つです。

[1] は、コンポーネントの名前です。

説明

ISICE25 は、インストールが Windows フォルダーまたはそのサブフォルダーのいずれにも直接書き込まないことを検証します。

修正アクション

この検証エラーを解決するには、次のいずれか 1 つを実行します:

- コンポーネントのインストール先を変更します。
- コンポーネントがフォントをインストールする場合、ベストプラクティスに従ってプロジェクトにフォントファイルが含まれていることを確認してください。詳しくは、「フォントのインストール」をご覧ください。
- コンポーネントがデバイス ドライバーをインストールする場合、ベストプラクティスに従ってプロジェクトにドライバーが含まれていることを確認してください。詳しくは、「デバイスドライバー設定を構成する」をご覧ください。

ISICE26

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] には、オプション ヘッダー [3] が設定されていません。

[1] は、ファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。[3] は、ファイルのコンパイル時には使用されなかったパラメーター (SafeSEH、NXCOMPAT、または DYNAMICBASE) です。

説明

ISICE26 は、インストールが Windows セキュリティ機能をサポートすることを検証します。インストールがインストールする実行可能ファイルの PE ヘッダーをチェックして、次のコマンドライン パラメーターを使ってコンパイルされていることを検証します:

- /SafeSEH—安全な例外処理を確認
- /NXCOMPAT—データの実行を防止
- /DYNAMICBASE—ASLR (Address Space Layout Randomization)

単一ファイルで、前述のパラメーターを 1 つずつ複数の ISICE26 エラーをトリガすることが可能です。

修正アクション

この検証エラーを解決するには、製品のソースコードのビルド オプションを確認して、必要に応じて編集します。Visual Studio の場合、これらは "リンカ" 設定の [詳細プロパティ] ページで構成されます。

InstallShield 仮想化整合性スイート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

InstallShield には、InstallShield 仮想化適合性スイートという名前の一連の検証ツールが含まれています。このスイートに含まれる InstallShield 仮想化適合性検証ツール (ISVICE) は、様々な仮想化テクノロジについてプロジェクトの適合性をチェックします。

InstallShield で使用できる ISVICE のリストは次の表のとおりです。

テーブル 20 · ISVICE

ISICE	プロジェクトの種類	テクノロジ	説明
ISVICE01	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	通常、Windows オペレーティング システムの一部であるファイルの存在を確認します。
ISVICE02	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	デバイス ドライバー ファイルの存在を確認します。

テーブル 20 • （続き.)ISVICE

ISICE	プロジェクトの種類	テクノロジ	説明
ISVICE03	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	ClickOnce 配布ファイルの存在を確認します。
ISVICE04	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。
ISVICE05	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	WMI プロバイダーの一部であるファイルの存在を確認します。
ISVICE06	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	J2EE アプリケーション サーバーの一部であるファイルの存在を確認します。
ISVICE07	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	仮想化を行うと動作しない可能性のあるアプリケーションの一部であるファイルの存在を確認します。
ISVICE08	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	仮想化を行うと失敗することが報告されているアプリケーションの一部であるファイルの存在を確認します。
ISVICE09	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在をチェックします。
ISVICE10	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	MsiDriverPackages テーブルを使ってインストールされたドライバーの存在を確認します。

テーブル 20 • （続き.)ISVICE

ISICE	プロジェクトの種類	テクノロジ	説明
ISVICE11	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	パッケージに少なくとも 1 つのショートカットが含まれていることを確認します。
ISVICE12	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、Server App-V、ThinApp、XenApp	シェル拡張の存在を確認します。
ISVICE13	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、Server App-V、ThinApp、XenApp	URL プロトコル ハンドラーの存在を確認します。
ISVICE14	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、Server App-V、ThinApp、XenApp	デフォルト プログラム リストのサポートの存在を確認します。
ISVICE15	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	ブート スタート サービスの存在を確認します。
ISVICE16	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、Server App-V、XenApp	インストールされたファイルの合計サイズが 4GB 未満であることを確認します。
ISVICE17	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	Windows Installer COM+ テーブルの使用状況を確認します。
ISVICE18	基本の MSI、InstallScript MSI, MSI データベース	App-V 4.x、App-V 5.x、ThinApp、XenApp	COM DLL サロゲート ファイルの存在を確認します。
ISVICE19	基本の MSI、InstallScript MSI, MSI データベース	ThinApp、XenApp	64 ビット パッケージであるかどうかを確認します。ThinApp および XenApp は 64 ビット アプリケーションをサポートしません。

ISVICE01

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE01 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1]が OS と厳密に統合されたファイルのようです。Internet Explorer、Windows Media Player、または .NET Framework のフレームワークなどのアプリケーションの一部であるファイルは、仮想化には向いていません。これらのファイルは仮想ではなく、ローカル マシンにインストールしてください。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE01 は、Windows オペレーティング システムの一部であるファイルの存在を確認します。リリースにこの種類のファイルが 1 つ以上含まれる場合、検証中にこのエラーが表示されます。

修正アクション

エンド ユーザーが製品の仮想バージョンを希望していて、作業中の InstallShield プロジェクトに .NET Framework その他の Windows と密接に統合されているアイテムが含まれている場合、プロジェクトからこれらのアイテムを削除することが推奨されます。エンド ユーザーが仮想バージョンの製品を使用する前に、ローカル マシンにこれらのプロジェクトに必要なテクノロジをインストールする手順を説明するドキュメントの提供を考慮してください。

ISVICE02

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE02 は、次の仮想化テクノロジに適用します:

- App-V 4.x

- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] がデバイス ドライバーのようです。プリンター ドライバーや USB デバイス ドライバーなど、システム レベルのドライバーは、仮想環境からは使用できません。このドライバーを抽出して、ローカル マシンにインストールし、パッケージの残りの部分を仮想化することができる場合があります。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE02 は、デバイス ドライバー ファイルの存在を確認します。プリンター ドライバーや USB デバイス ドライバーなど、システム レベルのドライバーは、仮想環境からは使用できません。

修正アクション

エンド ユーザーが製品の仮想バージョンnを希望していて、InstallShield プロジェクトにデバイス ドライバーが含まれている場合、プロジェクトからデバイス ドライバーを削除して、ドライバー用の個別のインストールを作成することを考慮してください。エンド ユーザーは製品の仮想バージョンを使用する前に、ローカル マシンにそのドライバーをインストールできます。

ISVICE03

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE03 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] が ClickOnce 配置ファイルのようです。ClickOnce はユーザーごとのインストール フォーマットなので、マシンごとの特性を持つ仮想パッケージとは互換性を持たない場合があります。また、ClickOnce アプリケーションは自動的にアップデートを行うため、仮想アプリケーション管理システムでは、無効なバージョン エラーの原因となる可能性があります。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE03 は、ClickOnce 配置ファイルの存在を確認します。プロジェクトに ClickOnce 配置 ファイルが含まれているとき、ISVICE03 が発生します。

修正アクション

ClickOnce 配置ファイルがアプリケーションにおける重要性を検討して、ClickOnce インストールが予定通りに動作することがどれほど重要であるのかを判断してください。ClickOnce 配置 ファイルがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布することが安全策と言えます。ClickOnce 配置ファイルが重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE04

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE04 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

指定されたファイル [1] が ASP.NET/IIS アプリケーションの一部のようです。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE04 は、ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。

修正アクション

ASP.NET または IIS アプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE05

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE05 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] が WMI プロバイダーのようです。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)プロバイダーが、パッケージの重要な部分を構成しない限り、パッケージの残りの部分を仮想アプリケーションとして使用することが可能な場合があります。)

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE05 は、WMI (Windows Management Instrumentation) プロバイダーの一部であるファイルの存在を確認します。

修正アクション

WMI プロバイダーが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE06

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE06 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp

- XenApp

メッセージ（エラー）

ファイル [1] が J2EE アプリケーション サーバーの一部のようです。デスクトップ アプリケーションの仮想化で、これはサポートされていません。）

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE06 は、J2EE アプリケーション サーバーの一部であるファイルの存在を確認します。

修正アクション

J2EE アプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE07

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE07 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ（警告）

ファイル [1] は、仮想化には不向きであるとされているアプリケーションの一部のようです。ただし、適切ではない部分をローカルにインストールして、パッケージの残りの部分だけを仮想化することが可能です。

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE07 は、仮想化を行うと動作しない可能性のあるアプリケーションの一部であるファイルの存在を確認します。これには、SQL Server などのアプリケーションからのファイルが含まれます。

修正アクション

サポートされていないアプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE08

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE08 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

ファイル [1] は、仮想化には不向きであるとされているアプリケーションの一部のようです。

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISVICE08 は、仮想化を行うと動作しない可能性のあるアプリケーションの一部であるファイルの存在を確認します。これには、ウィルス対策ソフトウェアや Exchange Server などのサーバー ソフトウェアからのファイルが含まれます。

修正アクション

サポートされていないアプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE09

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE09 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp

- XenApp

メッセージ (エラー)

IIS [1] アイテム [2] がパッケージに含まれています。デスクトップ アプリケーションの仮想化で、これはサポートされていません。)

[1] はプロジェクトに含まれる IIS アイテムの種類の名前で、[2] はそのアイテムの名前です。

説明

ISVICE09 は ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在をチェックします。

修正アクション

サポートされていない ASP.NET または IIS アプリケーションが製品の重要な部分を構成しない限り、またはターゲット システム上で個別にローカルでインストールすることが可能な場合、この検証エラーを無視することができます。

ISVICE10

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE10 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (エラー)

コンポーネント [1] には、MsiDriverPackages (DIFxApp) テーブルを使ってインストールされたデバイス ドライバーが含まれています。プリンター ドライバーや USB デバイス ドライバーなど、システム レベルのドライバーは、仮想環境からは使用できません。このドライバーを抽出して、ローカル マシンにインストールし、パッケージの残りの部分を仮想化することができる場合があります。)

[1] は、プロジェクトに含まれているデバイス ドライバーの名前です。

説明

ISVICE10 は、MsiDriverPackages テーブルを通してインストールするように構成されたドライバーの存在を確認します。

修正アクション

エンド ユーザーが製品の仮想バージョンを希望していて、InstallShield プロジェクトにドライバーが含まれている場合、プロジェクトからドライバーを削除して、ドライバー用の個別のインストールを作成することを考慮してください。エンド ユーザーは製品の仮想バージョンを使用する前に、ローカル マシンにそのドライバーをインストールできます。

ISVICE11

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE11 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (App-V 4.x and App-V 5.x の警告。ThinApp および XenApp のエラー)

このパッケージには、ショートカットが含まれていません。仮想アプリケーションへのエントリ ポイントを定義するために、通常、ショートカットが必要です。

説明

ISVICE11 パッケージに少なくとも 1 つのショートカットが含まれていることを確認します。ショートカットは、仮想パッケージ内のアプリケーションを起動するときに最も見つけやすいエントリ ポイントです。ほとんどの仮想パッケージには、1 つ以上のショートカットが含まれています。

修正アクション

パッケージが別の仮想パッケージの依存関係である場合、この ISVICE は無視できます。

パッケージがプラグインとして使用される場合、プラグインの対象となるアプリケーションへのショートカットを作成する必要があるかもしれません。

エンド ユーザーが仮想アプリケーションを個別に起動できるようにする場合、メインの実行可能ファイルへのショートカットの追加を考慮してください。

ISVICE12

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *MSI データベース*

ISVICE12 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- Server App-V
- ThinApp
- XenApp

メッセージ (App-V 4.x、App-V 5.x、ThinApp、および XenApp のエラー。Server App-V の警告。)

このパッケージには、シェル拡張 (キー [1]) が含まれています。シェル拡張は、Windows Explorer を拡張するため、仮想パッケージからロードすることはできません。この拡張がアプリケーションにとって大変重要な場合、仮想化した後にアプリケーションが機能しないことがあります。ただし、拡張が重要ではない場合、アプリケーションを使用できることもあります。

[1] は、プロジェクトに含まれているシェル拡張のキーです。

説明

ISVICE12 は、シェル拡張の存在を確認します。

修正アクション

シェル拡張が製品における重要性を検討して、シェル拡張が予定通りに動作することがどれほど重要であるのかを判断してください。シェル拡張がそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。シェル拡張が重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE13

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE13 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- Server App-V
- ThinApp
- XenApp

メッセージ (警告)

このパッケージは URL プロトコル (フレンドリ名 [1]) を登録します。

[1] は、URL プロトコルの表示名です。

説明

ISVICE13 は、URL プロトコル ハンドラーの存在を確認します。

修正アクション

URL プロトコルの登録が製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。URL プロトコルの登録がそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。URL プロトコルの登録が重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

製品に URL プロトコル ハンドラーが含まれているため、App-V 5.x が製品に適した仮想化テクノロジであるかどうかを検証してください。

ISVICE14

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE14 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- Server App-V
- ThinApp
- XenApp

メッセージ (警告)

This package registers its capabilities in the Default Programs list (key '%1').

[1] はデフォルト プログラム リストに含まれるアイテムの登録情報に対応するキーの名前です。

説明

ISVICE14 は、デフォルト プログラム リストのサポートの存在を確認します。

修正アクション

デフォルト プログラム リストの登録が、製品で重要ではない場合、この検証エラーを無視することができます。

製品に デフォルト プログラム リストのアイテムを登録するサポートが含まれているため、App-V 5.x が製品に適した仮想化テクノロジであるかどうかを検証してください。

ISVICE15

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE15 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

このパッケージには、起動時に開始するサービス (名前 [1]) が含まれています。仮想サービスは、仮想アプリケーションが有効である間だけに限られているため、起動時に開始しなくてはならないサービスは、仮想化には向いていません。このサービスを抽出して、ローカル マシンにインストールすることで、パッケージの残りの部分を仮想化することが可能な場合があります。)

[1] はプロジェクトに含まれるサービスの名前です。

説明

ISVICE15 は、ブート スタート サービスの存在を確認します。

修正アクション

サービスが製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。サービスがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。

サービスが重要であるが、それを残りのプロジェクトから切り離すことができる場合、プロジェクトから削除して、サービス用の個別のインストールを作成してください。エンド ユーザーは、製品の仮想バージョンを使用する前に、ローカル マシンにサービスをインストールできます。

サービスが重要な場合で、プロジェクトから切り離すことができないとき、製品の仮想バージョンは正しく機能しない場合があります。

ISVICE16

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE16 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- Server App-V
- XenApp

メッセージ (エラー)

This package contains more than 4 GB of files.Since the target virtualization technology has a 4 GB size limit, this application cannot be successfully virtualized.(このパッケージには、4 GB 以上のファイルが含まれています。ターゲットの仮想化テクノロジには 4 GB のサイズ制限があるため、このアプリケーションを正しく仮想化することができません。)

説明

ISVICE16 は、インストールされたファイルの合計サイズが 4GB 未満であることを確認します。App-V 4.x、Server App-V、および XenApp は、4 GB 以上のファイルを含むパッケージをサポートしません。

修正アクション

App-V 5.x または ThinApp は 4 GB 以上のサイズをサポートするので、App-V 5.x または ThinApp が製品に適した仮想化テクノロジであるかどうかを検証してください。

ISVICE17

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE17 は、次の仮想化テクノロジで使用できます:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

このパッケージには COM+ データ (コンポーネント [1]) が含まれているので、アプリケーションを仮想化した場合に正しく機能しない可能性があります。

[1] はプロジェクトに含まれるコンポーネントの名前です。コンポーネントには COM+ データが含まれています。

説明

ISVICE17 は、Windows Installer COM+ テーブルの使用状況を確認します。

修正アクション

COM+ アプリケーションが製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。COM+ アプリケーションがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。COM+ アプリケーションが重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE18

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE18 は、次の仮想化テクノロジに適用します:

- App-V 4.x
- App-V 5.x
- ThinApp
- XenApp

メッセージ (エラー)

このパッケージには、サポートされていない COM DLL サロゲート (レジストリ キー [1]) が含まれているので、アプリケーションを仮想化した場合に正しく機能しない可能性があります。

[1] はプロジェクトに含まれるレジストリ キーの名前です。

説明

ISVICE18 は、COM DLL サロゲート ファイルの存在を確認します。

修正アクション

COM DLL サロゲートが製品における重要性を検討して、製品が予定通りに動作することがどれほど重要であるのかを判断してください。DLL サロゲートがそれほど重要でない場合、製品の仮想バージョンは少しだけ機能を限定して配布しても恐らく問題はないと言えます。DLL サロゲートが重要な場合、製品の仮想バージョンが正しく機能しない場合があります。

ISVICE19

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

ISVICE19 は、次の仮想化テクノロジに適用します:

- ThinApp
- XenApp

メッセージ (エラー)

The package is 64-bit.ThinApp and XenApp do not support 64-bit applications.(パッケージが 64 ビットです。ThinApp および XenApp は 64 ビット アプリケーションをサポートしません。

説明

ISVICE19 は、64 ビット パッケージをビルドしているかどうかを確認します。ThinApp および XenApp は 64 ビット アプリケーションをサポートしません。

修正アクション

製品に 64 ビット サポートが含まれているため、App-V または Server App-V が製品に適した仮想化テクノロジであるかどうかを検証してください。

InstallShield MSIX/UWP アプリ適合性スイート

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallShield には、InstallShield MSIX/UWP アプリ適合性スイートという名前の一連の検証ツールが含まれています。このスイートに含まれる InstallShield Windows アプリ適合性検証ツール (ISUWP) は、.msi パッケージ内で Windows アプリ パッケージ (.appx) フォーマットに適さないアイテムの存在をスキャンします。このスイートにアクセスするには、**[ビルド]** メニューから **[検証]** をポイントしてから、**[InstallShield MSIX/UWP アプリ適合性スイート]** をクリックします。スイートは、問題が見つかったすべてのテストを [リリース] ビューに表示し、レポート内の各問題に関連付けられた列に既知の Windows アプリ バリエーション (ユニバーサル アプリ、デスクトップ ブリッジ、Windows ストア、WSA、および Nato Server) への適合性が表示されます。従来型の CUB の場合、これらの列は空白のままです。InstallShield で利用できる ISUWP 検証ツールは次の通りです:

テーブル 21 ・ ISUWP

ISUWP	プロジェクトの種類	テクノロジ	説明
ISUWP01	基本の MSI	Windows アプリ	Windows サービスの存在を確認します。
ISUWP02	基本の MSI	Windows アプリ	MsiDriverPackages テーブルを使ってインストールされたドライバーの存在を確認します。
ISUWP03	基本の MSI	Windows アプリ	ショートカットの存在を確認します。InstallShield では、アプリケーションを識別するためにショートカットが必要です。アプリケーションを識別するショートカット無しでは、アプリを開始する方法が無いためパッケージが無効となります。

テーブル 21 • ISUWP

ISUWP	プロジェクトの種類	テクノロジ	説明
ISUWP04	基本の MSI	Windows アプリ	Windows ストア アプリケーションの複数のショートカットの存在を確認します。InstallShield はショートカットをアプリケーションに変換し、Windows ストアでは複数アプリケーションを使用できない場合があります。
ISUWP05	基本の MSI	Windows アプリ	Windows Nano Server アプリケーションのショートカットの存在を確認します。InstallShield はショートカットがパッケージによってインストールされた .exe をポイントする場合、そのショートカットをタイルに変換しますが、Windows Nano Server はアプリケーション タイルを表示できません。
ISUWP06	基本の MSI	Windows アプリ	16 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 16 ビット ファイルをサポートしません。
ISUWP07	基本の MSI	Windows アプリ	32 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 32 ビット ファイルをサポートしません。
ISUWP08	基本の MSI	Windows アプリ	昇格された権限を必要とするアプリケーションの存在を確認します。
ISUWP09	基本の MSI	Windows アプリ	UI 保護レベルをバイパスできるアプリケーションの存在をチェックします。
ISUWP10	基本の MSI	Windows アプリ	COM 登録の存在を確認します。
ISUWP11	基本の MSI	Windows アプリ	カスタム アプリケーション ユーザー モデル ID を設定する実行可能ファイルの存在を確認します。
ISUWP12	基本の MSI	Windows アプリ	Windows Nano Server と互換性を持たない .NET アセンブリの存在を確認します。
ISUWP13	基本の MSI	Windows アプリ	.NET Framework の最新版をサポートしない .NET アセンブリの存在を確認します。
ISUWP14	基本の MSI	Windows アプリ	ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。

テーブル 21 • ISUWP

ISUWP	プロジェクトの種類	テクノロジ	説明
ISUWP15	基本の MSI	Windows アプリ	ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在を確認します。
ISUWP16	基本の MSI	Windows アプリ	SQL スクリプトを実行するビルトイン InstallShield サポートの存在を確認します。
ISUWP17	基本の MSI	Windows アプリ	テキスト ファイルを変更するビルトイン InstallShield サポートの存在を確認します。
ISUWP18	基本の MSI	Windows アプリ	XML ファイルを変更するビルトイン InstallShield サポートの存在を確認します。
ISUWP19	基本の MSI	Windows アプリ	スケジュール タスクを作成するビルトイン InstallShield サポートの存在を確認します。
ISUWP20	基本の MSI	Windows アプリ	予約済み、または禁止されているファイル タイプの関連付けの存在を確認します。

ISUWP01

ISUWP01 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには 1 つ以上の Windows サービスが含まれています。Windows Server 以外のデバイスをターゲットとする Windows アプリ パッケージは、Windows サービスをサポートしません。

説明

ISUWP01 は Windows サービスの存在を確認します。

修正アクション

アプリケーションが 1 つ以上の Windows サービスを必要とする場合、サーバー拡張および WSA が必要です。デスクトップ拡張を含めることは、このシナリオではサポートされていません。

ISUWP02

ISUWP02 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [1] には、MsiDriverPackages (DIFxApp) テーブルを使ってインストールされたデバイス ドライバーが含まれています。Windows アプリ パッケージは、デバイス ドライバーをサポートしません。

[1] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP02 は、MsiDriverPackages テーブルを使ってインストールされたドライバーの存在を確認します。

修正アクション

ドライバーが必要な場合、このアプリケーションは Windows アプリだけではサポートできません。

ISUWP03

ISUWP03 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

このパッケージには、ショートカットが含まれていません。アプリケーションへのエントリ ポイントを定義するために、通常、ショートカットが必要です。

説明

ISUWP03 は、ショートカットの存在を確認します。InstallShield では、アプリケーションを識別するためにショートカットが必要です。アプリケーションを識別するショートカット無しでは、アプリを開始する方法が無いためパッケージが無効となります。

ISUWP04

ISUWP04 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

このパッケージには 1 つ以上のショートカットが含まれています。Windows ストアでは、複数のアプリケーションを使用できない場合があります。

説明

ISUWP04 は、Windows ストア アプリケーションの複数のショートカットの存在を確認します。InstallShield はショートカットをアプリケーションに変換し、Windows ストアでは複数アプリケーションを使用できない場合があります。

修正アクション

このアプリケーションを Windows ストアに配置したい場合、アプリケーションを複数の Windows パッケージに分割しなくてはならない場合があります。

ISUWP05

ISUWP05 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

このパッケージには 1 つ以上のショートカットが含まれています。Windows Nano Server はアプリケーション タイルを表示できません。

説明

ISUWP05 は、Windows Nano Server アプリケーションのショートカットの存在を確認します。InstallShield はショートカットがパッケージによってインストールされた .exe をポイントする場合、そのショートカットをタイルに変換しますが、Windows Nano Server はアプリケーション タイルを表示できません。

修正アクション

この警告は、予定どおりに動作しない可能性のある既知のアイテムのアラートを行います。

ISUWP06

ISUWP06 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

コンポーネント [2] のファイル [1] は、16 ビット ファイルです。64 ビット システムは 16 ビット アプリケーションをサポートしません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP06 は、16 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 16 ビット ファイルをサポートしません。

修正アクション

修正アクションは、実行時にファイルがロードおよび実行する場合のみ必要です。16 ビットまたは 32 ビットのデータ ファイルは問題ありません。フラグされた 16 ビット ファイルが実行時にロードおよび実行する場合、これを 64 ビット ファイルに置換します。

ISUWP07

ISUWP07 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

コンポーネント [2] のファイル [1] は、32 ビット ファイルです。64 ビット システムは 32 ビット アプリケーションをサポートしない場合があります。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP07 は、32 ビット ファイルの存在を確認します。すべてのサーバー インストールは 64 ビット プラットフォームで、64 ビット プラットフォームは 32 ビット ファイルをサポートしません。

修正アクション

修正アクションは、実行時にファイルがロードおよび実行する場合のみ必要です。16 ビットまたは 32 ビットのデータ ファイルは問題ありません。フラグされた 32 ビット ファイルが実行時にロードおよび実行する場合、これを 64 ビット ファイルに置換します。

ISUWP08

ISUWP08 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] の [1] ファイルは、昇格された権限を必要とするアプリケーション実行可能ファイルです。Windows ストアでは、管理者権限を必要とするアプリケーションを使用できません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP08 は、昇格された権限を必要とするアプリケーションの存在を確認します。

修正アクション

昇格された権限の要件を削除します。アプリは昇格された権限を使って、ファイルまたはレジストリ キーの処理におけるベスト プラクティスを妨害する場合がありますが、そのような動作は Windows アプリでは使用できません。

ISUWP09

ISUWP09 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] の [1] ファイルは、デスクトップ上でより高度な権限を要するウィンドウへ打ち込むため、UI 保護レベルのバイパスが許可されているアプリケーション実行可能ファイルです。Windows アプリ パッケージでは、UI 保護レベルのバイパスが許可されているアプリケーションはサポートされていません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP09 は、UI 保護レベルをバイパスできるアプリケーションの存在をチェックします。

修正アクション

UI 保護レベルのバイパスを必要とするアプリケーションはほとんどありませんが、それが必要な場合、現在 Windows アプリではサポートされていません。アプリケーションが UI 保護レベルをバイパスする必要がある場合、Windows アプリ パッケージ以外の異なるパッケージの使用を考慮してください。

ISUWP10

ISUWP10 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

メッセージ 1(警告)

テーブル [1] には、1 つ以上のエントリが含まれています。Windows アプリ パッケージに含まれるアプリケーションは、COM サーバーを別のアプリケーションに公開することができません。

[1] はプロジェクトに含まれるテーブルです。

メッセージ 2(警告)

コンポーネント [2] のレジストリ キー [1] はCOM データです。Windows アプリ パッケージに含まれるアプリケーションは、COM サーバーを別のアプリケーションに公開することができません。

[1] はプロジェクトに含まれるレジストリ キーの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP10 COM 登録の存在を確認します。

修正アクション

COM 登録が問題の場合、またはそうでない場合があります。COM の内部的な使用はデスクトップ ブリッジでサポートされていますが、(他のアプリによる使用に向けて) パブリック登録をサポートする拡張子は現在サポートされていません。後者の場合、Windows アプリ パッケージは適していません。

ISUWP11

ISUWP11 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] にあるファイル [1] は、非推奨の API [3] をインポートします。Windows アプリ パッケージに含まれるアプリケーションは、カスタム アプリケーション ユーザー モデル ID を設定することができません。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

[3] は、API の名前です。

説明

ISUWP11 カスタム アプリケーション ユーザー モデル ID を設定する実行可能ファイルの存在を確認します。

修正アクション

問題のある API 呼び出しのコードベースを削除する必要があります。Windows アプリはアプリケーション ユーザー モデル ID を設定します。

ISUWP12

ISUWP12 は、Windows アプリのシナリオに適用します。

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のファイル[1] は、.NET Framework に対してコンパイルされた .NET アセンブリです。Windows Nano Server .NET コア アプリケーションのみをサポートします。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] のファイル [1] は、[3] に対してコンパイルされた .NET アセンブリです。Windows Nano Server .NET コア アプリケーションのみをサポートします。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

[3] はプロジェクトに含まれる Framework の名前です。

説明

ISUWP12 Windows Nano Server と互換性を持たない .NET アセンブリの存在を確認します。

修正アクション

Nano Server をサポートする場合、.NET コアに対してアプリを再ビルドします。

ISUWP13

ISUWP13 は、Windows アプリのシナリオに適用します。

メッセージ (警告)

コンポーネント [2] のファイル[1] は、.NET Framework の最新版をサポートしない .NET アセンブリです。Windows ストア アプリケーションは、常に .NET Framework の最新バージョン上で実行します。

[1] はプロジェクトに含まれるファイルの名前です。

[2] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISUWP13 .NET Framework の最新版をサポートしない .NET アセンブリの存在を確認します。

修正アクション

App.config を再ビルドまたは調整します。

ISUWP14

ISUWP14 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

指定されたファイル [1] が ASP.NET/IIS アプリケーションの一部のようです。これは Windows アプリに含まれるアプリケーションではサポートされていません。

[1] はプロジェクトに含まれるファイルの名前です。

説明

ISUWP14 は、ASP.NET/IIS アプリケーションの一部であるファイルの存在を確認します。

修正アクション

ASP.NET/IIS アプリケーションの一部であるファイルは Windows アプリではサポートされていないため、パッケージからこのファイルを削除するか、代替のパッケージ フォーマットを使用する必要があります。

ISUWP15

ISUWP15 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

IIS [1] アイテム [2] がパッケージに含まれています。これは Windows アプリに含まれるアプリケーションではサポートされていません。

[1] はプロジェクトに含まれる IIS アイテムの種類の名前で、[2] はそのアイテムの名前です。

説明

ISUWP15 は ASP.NET/IIS アプリケーションがインストール済みであることを示す InstallShield IIS サポートの存在を確認します。

修正アクション

ASP.NET/IIS アプリケーションの一部であるファイルは Windows アプリではサポートされていないため、パッケージからこのファイルを削除するか、代替のパッケージ フォーマットを使用する必要があります。

ISUWP16

ISUWP16 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行する 1 つ以上の SQL スクリプトが含まれています。Windows アプリ パッケージは、SQL スクリプトの実行をサポートしません。

説明

ISUWP16 は、SQL スクリプトを実行するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP17

ISUWP17 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行される 1 つ以上の InstallShield テキスト ファイルの変更アイテムが含まれています。Windows アプリ パッケージは、テキスト ファイルの変更をサポートしません。

説明

ISUWP17 は、テキスト ファイルを変更するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP18

ISUWP18 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行される 1 つ以上の InstallShield XML ファイルの変更アイテムが含まれています。Windows アプリ パッケージは、XML ファイルの変更をサポートしません。

説明

ISUWP18 は、XML ファイルを変更するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP19

ISUWP19 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

このパッケージには、インストール中に実行される 1 つ以上の InstallShield スケジュール タスク アイテムが含まれています。 Windows アプリ パッケージは、スケジュール タスクの作成をサポートしません。

説明

ISUWP19 は、スケジュール タスクを作成するビルトイン InstallShield サポートの存在を確認します。

修正アクション

Windows アプリ パッケージの使用をサポートするためには、フラグされた構成を、インストールからアプリケーションに移動する必要があります。

ISUWP20

ISUWP20 は、Windows アプリのシナリオに適用します。

メッセージ (エラー)

コンポーネント [2] の拡張子 [1] は、Windows アプリ パッケージに含まれるアプリケーションに予約済み、または禁止されているファイル タイプです。

[1] はプロジェクトに含まれる拡張子の名前で、[2] はその拡張子を含むコンポーネントの名前です。

説明

ISUWP20 は、予約済み、または禁止されているファイル タイプの関連付けの存在を確認します。

修正アクション

フラグされた予約済み、または禁止されているファイル タイプの関連付けを削除して、Windows アプリ パッケージの使用をサポートします。

InstallShield ベスト プラクティス スイート

エディション: *InstallShield ベスト プラクティス スイートは、InstallShield の Premier Edition で提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

一部の ISBP は、基本の MSI プロジェクトと MSI データベース プロジェクトに適用します。

InstallShield には、InstallShield ベスト プラクティス スイートという名前の 1 セットの検証ツールが含まれています。インストールがベスト プラクティス ガイドラインに違反している場合、このスイートの InstallShield ベスト プラクティス (ISBP) 検証ツールによって警告されます。

以下は、InstallShield で提供されている ISBP の一覧です。

テーブル 22 • ISBP

ISICE	プロジェクト	説明
ISBP01	基本の MSI、InstallScript MSI、MSI データベース	"ALL" という名前の機能がないことを検証します。
ISBP02	基本の MSI、InstallScript MSI、MSI データベース	"DATABASE" という名前のディレクトリがないことを検証します。
ISBP03	基本の MSI、MSI データベース	50 ユニット未満の ComboBox がないことを確認します。
ISBP04	基本の MSI、MSI データベース	ダイアログで使用されているプロパティがセキュリティで保護された、または制限付きのパブリック プロパティであることを確認します。
ISBP05	基本の MSI、MSI データベース	NULL 値の ControlEvent 条件がないことを確認します。
ISBP06	基本の MSI、InstallScript MSI、MSI データベース	InstallUISequence カスタム アクションが InstallExecuteSequence にもシーケンスされていることを確認します。
ISBP07	基本の MSI、InstallScript MSI、MSI データベース	すべての機能が関連付けられているコンポーネントを持ち、すべてのコンポーネントが機能に関連付けられていることを確認します。
ISBP08	基本の MSI、InstallScript MSI、MSI データベース	ARPINSTALLLOCATION が InstallExecuteSequence の CostFinalize の後に設定されていることを確認します。
ISBP09	基本の MSI、InstallScript MSI、MSI データベース	LIMITUI が ARPNOMODIFY なしに設定されていないことを確認します。
ISBP10	基本の MSI、InstallScript MSI、MSI データベース	AppSearch プロパティがセキュリティで保護された、または制限付きのパブリック プロパティであることを確認します。
ISBP11	基本の MSI、InstallScript MSI、MSI データベース	コンパイル済みの .NET アセンブリが配布されていないことを確認します。

テーブル 22 • ISBP (続き.)

ISICE	プロジェクト	説明
ISBP12	基本の MSI、InstallScript MSI、MSI データベース	自己登録型のファイルがないことを確認します。
ISBP13	基本の MSI、MSI データベース	ダイアログ コントロールによって設定され、インストールで使用されるプロパティにデフォルト値があることを確認します。
ISBP14	基本の MSI、InstallScript MSI、MSI データベース	各ファイルに正しいバージョン情報または MsiFileHash エントリがあることを確認します。
ISBP15	基本の MSI、MSI データベース	RadioButtonGroup に Text が定義されていることを確認します。
ISBP16	基本の MSI、InstallScript MSI、MSI データベース	インストール先が 64 ビットであるコンポーネントがそれぞれ 64 ビット コンポーネントとマークされていることを確認します。
ISBP17	基本の MSI、InstallScript MSI、MSI データベース	.hlp ファイルまたは WinHelp ランタイム ファイルがインストールに含められていないことを検証します。
ISBP18	基本の MSI、InstallScript MSI、MSI データベース	古い形式の API がインポートされていないことを検証します。
ISBP19	基本の MSI、InstallScript MSI、MSI データベース	使用されていない API がインポートされていないことを検証します。
ISBP20	基本の MSI、InstallScript MSI、MSI データベース	Registry テーブルに含まれるレジストリ エントリが、HKLM¥Software などのルート レベルのレジストリ キーや、削除された場合にターゲット マシン上でのトラブルの原因となるような、その他のキーの削除を行わないことを検証します。

ISBP01

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

Feature ALL conflicts with the installation meta-feature ALL.(機能 ALL がインストール メタ機能 ALL と競合します。)

説明

ISBP01 は、インストールが **ALL** という名前が大文字、小文字、または大文字と小文字の両方で指定されている機能がないことを確認します。機能のコンテキストで、Windows Installer は **ALL** という単語を機能の状態プロパティ (ADVERTISE、REINSTALL および ADDLOCAL など) の有効値として使用するために予約しています。機能の状態プロパティには、特定の機能名のほか、すべての機能を意味する **ALL** という単語を指定することができます。したがって、プロジェクト内の個々の機能に **ALL** という名前を指定することはできません。

修正アクション

このエラーを解決するには、機能の名前を ALL 以外の名前に変更します。詳細については、「機能の作成」を参照してください。

このエラーはまた、出力ウィンドウで ISBP01 エラー メッセージをクリックしても解決することができます。[ダイアログ エディター] ビューで Feature テーブルが表示され、**ALL** という名前の機能を含む行がハイライトされます。機能の名前を変更するには、現在の名前を選択して、新しい名前を入力します。

ISBP02

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

Directory DATABASE conflicts with Windows Installer's undocumented directory DATABASE.(ディレクトリ DATABASE が Windows Installer の文書化されていないディレクトリ DATABASE と競合します。)

説明

ISBP02 は、インストールが **DATABASE** という名前が大文字、小文字、または大文字と小文字の両方で指定されている機能がないことを確認します。Windows Installer は、このディレクトリ名を予約語として使用します。

修正アクション

このエラーを解決するには、[出力] ウィンドウで ISBP01 エラー メッセージをクリックします。[ダイアログ エディター] ビューで Directory テーブルが表示され、**DATABASE** という名前のディレクトリを含む行がハイライトされます。ディレクトリの名前を変更するには、現在の名前を選択して、新しい名前を入力します。

ISBP03

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (警告)

ComboBox [1] (ダイアログ [2]) の高さが 50 以下です。ComboBox が低すぎる場合、複数のアイテムを一度に表示できません。

[1] はコンボ ボックス コントロールの名前で、[2] はそのコントロールを持つダイアログの名前です。

説明

コンボ ボックス コントロールの高さが 50 インストーラー ユニット未満の場合、エンドユーザーはこのボックスで一度に 1 アイテムしか見ることができなくなることがあります。ユーザビリティを改善するために、高さを 50 以上にすることをお勧めします。

修正アクション

この警告を解決するには、[出力] ウィンドウで ISBP03 警告メッセージをクリックします。[ダイアログ エディター] ビューで Control テーブルが表示され、コンボボックス コントロールを含む行がハイライトされます。このコントロールの Height 列の値を 50 以上に変更します。

ISBP04

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ

メッセージ 1(警告)

コントロール [2] (ダイアログ [1] 内) はプライベート プロパティ [3] を使用しています。この値は、[実行] シーケンスで利用できません。

[1] はプロジェクト内のダイアログの名前で、[2] はそのダイアログにあるコントロールの名前で、[3] はそのコントロールによって設定また使用されているプライベート プロパティの名前です。

メッセージ 2(警告)

コントロール [2] (ダイアログ [1] 内) はセキュリティで保護されていないカスタム プロパティ [3] を使用しています。この値は、[実行] シーケンスで利用できません。

[1] はプロジェクト内のダイアログの名前で、[2] はそのダイアログにあるコントロールの名前で、[3] はそのコントロールによって使用されているプロパティの名前です。

説明

インストールの [ユーザー インターフェイス] シーケンスでプライベート プロパティを設定し、その値を [実行] シーケンスに渡すことはできません。したがって、ISBP04 は、インストールにプライベート プロパティを使用するダイアログ コントロールが含まれていないことを検証します。ダイアログ コントロールでプライベート プロパティが使用されている場合、場合によって、プロパティの値が [実行] シーケンスに渡されることが許可されるように手続きをとる必要があるというメッセージが警告 1 によって表示されます。

また、[実行] シーケンスで昇格された権限が必要なインストールの [ユーザー インターフェイス] シーケンスでパブリック プロパティを設定してあるときに、そのプロパティの値を [実行] シーケンスに渡す場合、プロパティは SecureCustomProperties プロパティの値として一覧されているか、制限付きパブリック プロパティである必要があります。したがって、ISBP04 はまた、インストールにカスタム パブリック プロパティを使用するダイアログ コントロールがある場合、カスタム パブリック プロパティが **SecureCustomProperties** プロパティの値として一覧されていることも確認します。カスタム パブリック プロパティが **SecureCustomProperties** プロパティの一覧にない場合、場合によってプロパティの値が [実行] シーケンスに渡されることが許可されるように手続きをとる必要があるというメッセージが警告 2 によって表示されます。

修正アクション

[実行] シーケンスでプロパティを使用する必要がない場合、この警告メッセージは無視することができます。ただし、このシーケンスでプロパティを使用する必要ある場合、この警告を解決する必要があります。

警告 1 を解決する必要がある場合、プライベート プロパティをパブリック プロパティに変更します。この変更を行うには、まず [ダイアログ] ビューでダイアログ コントロールを見つけ、コントロールの Property プロパティの文字列をすべて大文字にします (パブリック プロパティはすべて大文字)。また、プロジェクト内の他の場所で使用されているプロパティ名もすべて同様に変更します。

警告 2 を解決する必要がある場合、[プロパティ マネージャー] ビューを開きます。SecureCustomProperties プロパティの Value 列で、警告メッセージで言及されているプロパティの名前を追加します。このプロパティに複数のプロパティが含まれている場合、各プロパティをセミコロン (;) で区切ります。

Private Properties (Windows Installer ヘルプ ライブラリ)
Public Properties (Windows Installer ヘルプ ライブラリ)
SecureCustomProperties Property (Windows Installer ヘルプ ライブラリ)

ISBP05

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (警告)

コントロール [2] (ダイアログ [1] 内) に NULL 値の条件が含まれています。コントロール イベントが常に実行されるように、1 の条件を使用します。

[1] はプロジェクト内のダイアログの名前で、[2] はそのダイアログにあるコントロールの名前です。

説明

ISBP05 は、各ダイアログ コントロールの各イベントの条件が Null 値ではないことを確認します。この検証により、Windows Installer によってコントロールのイベントが適切な状況で起動されるようになります。

修正アクション

コントロールの他のイベントが起動されなかった場合にのみ Null 値の条件があるイベントが Windows Installer によって起動される必要がある場合、この警告メッセージは無視することができます。

この警告を解決する必要がある場合、まず [ダイアログ] ビューでダイアログを見つけます。ダイアログの下にある [動作] をクリックします。コントロールの一覧で、警告メッセージで言及されているコントロールをクリックします。右下のペインに [イベント] タブが表示されていることを確認します。右上に表示されているペインのグリッド内で、イベントの条件を入力します。Windows Installer によってこのイベントが起動されるようにするには、条件として 1 を入力します。

ControlEvent Table (Windows Installer ヘルプ ライブラリ)
Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

ISBP06

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

カスタム アクション [1] は InstallUISequence でスケジュールされていますが InstallExecuteSequence ではスケジュールされていません。このカスタム アクションは、サイレントまたは簡易 UI インストールでは実行されません。

[1] はプロジェクトに含まれるカスタム アクションの名前です。

説明

[インストール] シーケンス中 [実行] シーケンスにスケジュールされたアクションは、サイレント インストール、基本の UI インストール、または簡易 UI インストールでは実行されません。

修正アクション

サイレント インストール、基本の UI インストール、および簡易 UI インストール中 Windows Installer によってカスタム アクションが起動される必要がある場合、カスタム アクションを UI シーケンスに加えて、またはその代わりに、[インストール]-[実行] シーケンスにスケジュールすることも考えられます。

アクションの再スケジュールするには、まず [カスタム アクションとシーケンス] ビューを開いて、警告メッセージの中で言及されているアクションを見つけます。次いで、そのアクションを [インストール] シーケンスの [ユーザー インターフェイス] シーケンスから [インストール] シーケンスの [実行] シーケンスにドラッグします。

両方のシーケンスでアクションをスケジュールするには、[インストール] シーケンスの [ユーザー インターフェイス] シーケンスのアクションを [インストール] シーケンスの [実行] シーケンスにコピーします。[カスタム アクションとシーケンス] ビューでこれを行うには、CTRL を押しながら、アクションを前のシーケンスから後のシーケンスにドラッグします。

サイレント インストール、基本の UI インストール、または縮小 UI インストール中、このカスタム アクションを Windows Installer によって起動しない場合、この警告は無視できます。

ISBP07

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(警告)

機能 [1] にコンポーネントがありません。

[1] は、プロジェクトに含まれる機能の名前です。

メッセージ 2(警告)

コンポーネント [1] は機能に関連付けられていません。

[1] はプロジェクトに含まれるコンポーネントの名前です。

説明

ISBP07 は、プロジェクト内のすべてのコンポーネントが最低 1 つの機能に属していることを確認します。コンポーネントが最低 1 つの機能に関連付けられれていない場合、Windows Installer はそのコンポーネントをインストールすることができません。

ISBP07 はまた、プロジェクト内のすべての機能が最低 1 つのコンポーネントを含んでいることを確認します。コンポーネントには、ファイル、ショートカット、レジストリ エントリなどのインストール要素が含まれています。機能にコンポーネントがない場合、機能にインストールするものが何もないということになります。

修正アクション

これらの警告のいずれかを解決するには、[セットアップのデザイン] ビューを使用して、コンポーネントを機能に関連付けます。詳細については、「既存コンポーネントを機能に関連付ける」を参照してください。

ISBP08

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

InstallExecuteSequence の CostFinalize の後に ARPINSTALLLOCATION を設定する Type 51 アクションがないようです、。

説明

ARPINSTALLLOCATION プロパティは、製品のプライマリ インストール先への完全修飾パスを指定します。Windows Installer はこの値を Uninstall レジストリ キーに書き込みます。

ARPINSTALLLOCATION プロパティは通常、プロパティ設定型のカスタム アクション (タイプ 51) によって設定されます。

SetARPINSTALLLOCATION カスタム アクションは、基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションです。このカスタム アクションをプロジェクトから削除すると、ISBP08 警告に遭遇する場合があります。

修正アクション

この警告を解決するには、プロジェクトにカスタム アクションを次の設定と共に追加します。

- プロパティ名: ARPINSTALLLOCATION
- プロパティ値: [INSTALLDIR]
- 実行スケジュール: Always execute
- インストール実行シーケンス: After CostFinalize

  ***メモ・** ベスト プラクティスに従うには、機能の移行がメジャー アップグレード アイテムで選択されている場合、CostFinalize の後のすべてアクションを MigrateFeatureStates の後にシーケンスします。*

- インストール実行条件: Not Installed

残りの設定はすべてデフォルト値のままにします。

詳細については、「[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する」を参照してください。

ARPINSTALLLOCATION Property (Windows Installer ヘルプ ライブラリ)

ISBP09

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

The property LIMITUI has been set, but ARPNOMODIFY has not.This can lead to undesirable behavior with Add or Remove Programs.(プロパティ LIMITUI は設定されていますが、ARPNOMODIFY は設定されていません。これは [プログラムの追加と削除] での正しくない動作につながります。)

説明

インストールで LIMITUI プロパティを設定した場合、**ARPNOMODIFY** プロパティも設定する必要があります。

LIMITUI プロパティは "ユーザー インターフェイス レベル" を [基本] に設定します。基本の UI で実行されたインストールは、エンドユーザーに進捗状況メッセージのみを表示します。エンドユーザーは、機能を選択したり、フィードバックを提供したりすることをできません。

ARPNOMODIFY プロパティは、エンドユーザーが [プログラムの追加と削除] で製品を変更しようとするのをできないようにします。

修正アクション

このエラー解決するには、[プロパティ マネージャー] ビューで **ARPNOMODIFY** プロパティをプロジェクトに追加し、その値を 1 に設定します。詳細については、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

User Interface Levels (Windows Installer ヘルプ ライブラリ)

ISBP10

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

AppSearch [1] はセキュリティで保護されていないカスタム プロパティ [2] を使用しています。この値は、[実行] シーケンスで利用できません。

[1] はプロジェクトに含まれるシステム検索の名前で、[2] はそのシステム検索によって使用されるプロパティの名前です。

説明

[実行] シーケンスで昇格された権限が必要なインストールの [ユーザー インターフェイス] シーケンスにパブリック プロパティが設定されている場合、そのプロパティの値を [実行] シーケンスに渡すためには、プロパティが SecureCustomProperties プロパティの値としてリストされているか、または制限付きパブリック プロパティである必要があります。したがって、ISBP10 は、プロジェクトの AppSearch テーブルにカスタム パブリック プロパティがある場合、そのカスタム パブリック プロパティが **SecureCustomProperties** プロパティの値として一覧されていることを確認します。カスタム パブリック プロパティが **SecureCustomProperties** プロパティの一覧にない場合、場合によってプロパティの値が [実行] シーケンスに渡されることが許可されるように手続きをとる必要があるというメッセージが警告によって表示されます。

修正アクション

[実行] シーケンスでプロパティを使用する必要がない場合、この警告メッセージは無視することができます。ただし、このシーケンスでプロパティを使用する必要ある場合、この警告を解決する必要があります。

この警告を解決するには、[プロパティ マネージャー] ビューを開きます。SecureCustomProperties プロパティの Value 列で、警告メッセージで言及されているプロパティの名前を追加します。このプロパティに複数のプロパティが含まれている場合、各プロパティをセミコロン (;) で区切ります。

SecureCustomProperties Property (Windows Installer ヘルプ ライブラリ)
AppSearch Table (Windows Installer ヘルプ ライブラリ)

ISBP11

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (警告)

ファイル [1] は .NET アセンブリのプリコンパイル済みネィティブ イメージのようです。これにより、ターゲット マシンでプリコンパイルするよりも効率が悪くなる可能性があります。

[1] は .NET アセンブリのプリコンパイル済みネィティブ イメージの名前です。

説明

ISBP11 は、プロジェクトに .NET アセンブリのプリコンパイル済みネィティブ イメージが含まれていないことを確認します。

修正アクション

この警告を解決するには、メッセージの中で言及されているネィティブ イメージ ファイルを適切な .NET アセンブリ ファイルで置き換えます。

アセンブリがインストール中にマシン コードにコンパイルされるようにする場合、ファイルのコンポーネントの ".NET プリコンパイル アセンブリ" 設定で [はい] を選択します。

プリコンパイル、またはジャストインタイプ コンパイルでは、一部のコードが実行中に呼び出されない可能性があることを考慮に入れています。プリコンパイルは実行中、必要に応じてアセンブリを変化し、後の呼び出しでアクセスできるように結果のコードを格納します。ターゲット マシンでのプリコンパイルでは、プロセスにおいて、実行されているマシンの正確なアーキテクチャを有効に利用することができます。

ISBP12

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

カスタム アクション [1] は RegSvr32 を通して自己登録を呼び出しているようです。ベスト プラクティスでは、COM および Registry テーブル データをインストーラー パッケージに作成することが推奨されています。

[1] は、インストール時に RegSvr32.exe を使用してファイルを登録する可能性があるプロジェクト内のカスタム アクションの名前です。

メッセージ 2(エラー)

ファイル [1] は自己登録型です。ベスト プラクティスでは、COM および Registry テーブル データをインストーラー パッケージに作成することが推奨されています。

[1] は、自己登録型としてマークされているファイルの名前です。

説明

InstallShield では、COM サービスが自己登録するように指定することができますが、好まれる COM サービスの登録方法は、ビルド時またはデザイン時にファイルから COM 情報を抽出する方法です。この方法を使用した場合、InstallShield は COM クラス情報を .msi データベースの Class、ProgID、および Registry テーブルに書き込みます。自己登録には、いくつかの制限があります。詳しくは、「COM サーバーの自己登録」を参照してください。

修正アクション

エラー 1 を解決するには、エラー メッセージで参照されたカスタム アクションを削除して、そのファイルの COM 抽出を適切に構成します。

エラー 2 を解決するには、このファイルの COM 抽出を構成します。

詳細については、「COM 情報を COM サーバーから抽出する」を参照してください。

ISBP13

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (エラー)

プロパティ [1] にデフォルト値がなく、ダイアログ コントロールで設定されていて、テーブル [2] で使用されています。サイレントまたは簡易 UI インストールで、作成されない可能性があります。

[1] はプロジェクト内のプロパティの名前で、[2] は、そのプロパティを使用するテーブルの名前です。)

説明

ほとんど場合、**Property** テーブルの各プロパティにはデフォルト値が必要です。エンドユーザーがインストールをサイレント、簡易 UI、または基本の UI モードでインストールを実行したとき、デフォルト値が使用されます。

修正アクション

プロパティにデフォルト値を追加する必要がない場合、このエラー メッセージは無視することができます。

このエラーを解決するには、[プロパティ マネージャー] ビューを開きます。エラー メッセージで指定されたプロパティの Value 列に値を入力します。

ISBP14

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

ファイル [1] に、ファイル バージョンも MsiFileHash レコードもありません。パッチのインストール時に、ソース パッケージが必要になる可能性があります。

[1] は、プロジェクトに含まれるファイルの名前です。

メッセージ 2(エラー)

ファイル [1] はバージョン [2] として記録されていますが、実際のバージョンは [3] です。パッチのインストール時に、ソース パッケージが必要になる可能性があります。

[1] は、プロジェクトに含まれるファイルの名前です。[2] は、ファイルに構成されたバージョン番号ですが、[3] が実際のバージョン番号です。

説明

ISBP14 は、プロジェクトの各ファイルが正しいバージョン情報または MsiFileHash エントリをもっていることを確認します。バージョン付きファイルの正しいバージョン情報およびバージョンがないファイルの MsiFileHash エントリを使用することにより、Windows Installer は必要のないファイル コピーを検出して、除去することができます。また、以前のバージョンの製品にパッチを適用するときに、ソース パッケージが必要ならないようにします。

修正アクション

エラー 1 を解決する方法として、ファイルのバージョン番号をファイルに追加することができます。ファイルにバージョンがない場合、ファイル ハッシュが使用されるようにファイルを構成します。

InstallShield では、ファイルのバージョン情報をオーバーライドすることができます。ファイルのプロパティがオーバーライドされると、エラー 2 が発生する場合があります。このエラーを解決するために、バージョンのオーバーライドを削除することもできます。

ファイルの設定を構成するには、まず [ファイルとフォルダー] ビューを開きます。**[インストール先コンピューターのファイル]** ペインで、該当のファイルを右クリックしてから、[プロパティ] を選択します。[プロパティ] ダイアログ ボックスが開き、バージョン情報の構成とファイル ハッシュを使用するかどうかの指定を行うことができます。

File Table (Windows Installer ヘルプ ライブラリ)

ISBP15

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *MSI データベース*

メッセージ (エラー)

Control [2] on the Dialog [1] is a borderless RadioButtonGroup with overlapping controls and data in the Text column.This may cause repaint issues on some systems.(コントロール [2] (ダイアログ [1] 内) は、Text 列にオーバーラップするコントロールとデータがある境界線のない RadioButtonGroup です。これにより、一部のシステムで再描画の問題が発生する可能性があります。)

説明

ISBP15 は、ダイアログに境界線のないラジオ ボタン グループ コントロールがあり、かつ 1 つまたは複数の他のコントロールがそのラジオ ボタン グループとオーバーラップするとき、ラジオ ボタン グループの Text 属性に何も指定されていないことを確認します。指定されている場合、Windows Installer の一部のバージョンで再描画に関する問題が発生する可能性があります。

修正アクション

このエラーを解決するには、[出力] ウィンドウで ISBP15 エラー メッセージをクリックします。[ダイアログ エディター] ビューで Control テーブルが表示され、ラジオボタン コントロールを含む行がハイライトされます。ラジオ ボックス コントロールの Text 列の文字列を削除します。

また、ラジオ ボックス コントロールが境界線を使用するように変更することもできます。この変更を行うには、まず [ダイアログ] ビューを開いて、エラー メッセージで言及されているダイアログを見つけます。ラジオ ボタン コントロールをクリックして、Has Border プロパティを True に変更します。

RadioButtonGroup Control (Windows Installer ヘルプ ライブラリ)

ISBP16

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

Component [1] installed to 64-bit location [2] is not marked with the 64-bit component attribute.This may allow files to be installed to an incorrect location.[1] is the name of a component, and [2] is the destination folder for the component's files.(コンポーネント [1] (64 ビットの場所 [2] にインストール済み) が、コンポーネント属性で 64 ビットとマークされていません。このため、ファイルが不適切な場所にインストールされる可能性があります。

[1] はコンポーネントの名前で、[2] はコンポーネントのファイルのインストール先フォルダーです。)

説明

コンポーネントが 64 ビットとして構成されていない場合、Windows Installer はそのコンポーネントのファイルを適切な 64 ビットの場所にインストールしない可能性があります。

修正アクション

コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。

"64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。

ISBP17

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

コンポーネント [2] のファイル [1] は、WinHelp で、Windows Vista 以降に配布されない可能性があります。

[1] は、プロジェクト内の .hlp ファイルの名前で、[2] は、その .hlp ファイルを含むコンポーネントの名前です。

メッセージ 2(エラー)

コンポーネント [2] のファイル [1] は、WinHelp ランタイムで、Windows Vista 以降に配布されない可能性があります。

[1] はプロジェクトに含まれる Windows Help アプリケーション ファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。

説明

ISBP17 は、プロジェクトに .hlp ファイルが含まれていないことを検証します。Windows Vista 以降は、この種類のファイルをサポートしていません。

また、ISBP17 は、WinHlp32.exe と WinHelp.exe がプロジェクトに含まれていないことも検証します。これらのアプリケーションは再配布できません。

修正アクション

この検証エラーを解決するためには、プロジェクトからヘルプ ファイルを削除します。代案として、ヘルプ システムを別のファイル形式 (例、.chm、.htm、.xml など)に変換することも考えられます。この場合、Windows API からの呼び出しも、新しいヘルプ ファイルに変更する必要があります。

ISBP18

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、古い形式の API [3] をインポートします。

[1] は、プロジェクトに含まれるファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。[3] は、そのファイルが使用する古い形式の API の名前です。

説明

ISBP18 は、プロジェクトに含まれるすべての .dll ファイルおよび .exe ファイルが旧式のカーネル API を使用しないことを検証します。

修正アクション

この検証エラーを解決するには、このファイルを古い形式の API を使用しない更新済みのバージョンで置き換えます。更新済みのバージョンを作成するとき、場合によって、.exe または .dll ファイルを上書きし、再ビルドする必要があります。exe または .dll ファイルがサードパーティ ベンダーからのものである場合、そのベンダーより更新済みのバージョンを取得してください。

ISBP19

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ (エラー)

コンポーネント [2] のファイル [1] は、非推奨の API [3] をインポートします。

[1] は、プロジェクトに含まれるファイルの名前で、[2] は、そのファイルを含むコンポーネントの名前です。[3] は、そのファイルが使用する非推奨の API の名前です。

説明

ISBP19 は、プロジェクトに含まれるすべての .dll ファイルおよび .exe ファイルが非推奨のカーネル API を使用しないことを検証します。

修正アクション

この検証エラーを解決するには、このファイルを非推奨の API を使用しない更新済みのバージョンで置き換えます。更新済みのバージョンを作成するとき、場合によって、.exe または .dll ファイルを上書きし、再ビルドする必要があります。exe または .dll ファイルがサードパーティ ベンダーからのものである場合、そのベンダーより更新済みのバージョンを取得してください。

ISBP20

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

メッセージ

メッセージ 1(エラー)

レジストリ エントリ [1] の Name 列に、アスタリスクが含まれています。これは、Windows Installer がレジストリ キー [2] を削除する原因となります。

[1] は、アスタリスク (*) を含むレジストリ エントリ行の Registry テーブルでの Registry 列の名前です。[2] は、アンインストール中に削除されるレジストリ キーの名前です。

メッセージ 2(警告)

レジストリ エントリ [1] は、ソフトウェア アプリケーションとのインストールで推奨されないキーにインストールします。このエントリをインストールすると、ターゲットマシン ([2]) に悪影響が及ぶ可能性があります。

[1] は、ターゲットシステム上でトラブルの原因となる可能性のあるレジストリ エントリ行の Registry テーブルでの Registry 列の名前です。[2] は、レジストリ キーの名前です。

説明

ISBP20 は、製品がターゲット マシンから削除されるときに、HKEY_LOCAL_MACHINE¥SOFTWARE などのルート レベルのレジストリ キーを削除するようなレジストリ エントリがプロジェクトに含まれていないことを検証します。次のテーブルは、アンインストール中に問題を引き起こすようなレジストリ データの例を示します。この例では、HKEY_LOCAL_MACHINE¥SOFTWARE 全体と、そのサブ キーおよび値のすべてが削除されます。

テーブル 23 ・ ISBP20 エラーをトリガーするレジストリ データの例

レジストリ	ルート	キー	名前	値	コンポーネント
Registry2	2	SOFTWARE	*		MyComponent

ISBP20 は、ターゲット マシン上で悪影響を及ぼすような他のレジストリ エントリがプロジェクトに含まれていないことを検証します。

修正アクション

この検証エラーまたは警告を解決するには、[出力] ウィンドウで ISBP20 メッセージをクリックします。[ダイレクト エディター] ビューに Registry テーブルが表示され、問題を引き起こす可能性のあるレジストリ エントリを含む行がハイライト表示されます。レジストリ エントリを削除するには、その行を右クリックしてから [行の削除] を選択します。

Registry テーブル (Windows Installer ヘルプ ライブラリ)

インストールまたはアンインストールがターゲット システムを再開するタイミングについて

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI — InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合*

この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

BATCH_INSTALL システム変数がゼロ以外の値に設定されているとき、それはターゲット システムが再起動した後に実行が必要な操作が 1 つ以上あることを意味します。たとえば、ファイルがターゲット システム上に既に存在し、またそれがロックされているために、インストールがそのファイルをインストールできないと判断した場合、BATCH_INSTALL がゼロ以外の値に設定されるときがあります。

インストールに含まれるファイルがロックされているためにインストールできないとき、インストールは次回システムが再開するときにそのファイルを更新する必要があることをオペレーティング システムに自動的に通知します。追加の Windows ファイルがロードされる前にファイルを更新しなくてはならない可能性があるため、ファイルは Windows がロードされるときに更新されます。なお、InstallScript インストールではファイルは更新されません。

BATCH_INSTALL は、システムが再開した後にロックされたファイルを削除するためにアンインストール中に設定されます。ただし、再起動後にキャッシュされたインストール ファイルを使用することはできません。したがって、再起動関連の処理で有効なのは、オペレーティング システムが実行する処理に限ります。

If BATCH_INSTALL は、ファイル転送が終了したときにゼロ以外の値に設定され、以下が発生します:

- インストールは、ファイルの自己登録を一切行わない。これは、ファイルが正しく自己登録されるためにインストールのファイル全てが完全にインストールおよび更新されていなくてはならない場合があるためです。インストールはこれらのファイルが他の自己登録ファイルの依存関係である可能性を考慮し、インストールできなかったファイルが自己登録型であるかどうかに関わらず、ファイルの自己登録を一切行わない点にご注意ください。この場合、インストールは自己登録の必要があるファイルを記録して、再開した後にインストールがこれらのファイルを自己登録できるようにします。

- フレームワークは自動的に **SdFinishReboot** (OnFirstUIAfter または OnMaintUIAfter イベント内) を呼び出して、エンド ユーザーがインストール完了後にシステムを自動的に再開するためのオプションを提供します。
- インストールが完了すると適切な RunOnce レジストリを作成し、再開後にインストールが実行するようにします。再開後、インストールは /reboot パラメーターを使って実行します。

  ALLUSERS スクリプト変数が 1 に設定された場合、RunOnce エントリは HKEY_LOCAL_MACHINE の下に作成されます。その他の場合、このエントリは HKEY_CURRENT_USER の下に作成されます。Windows は、これらの両方の場所で RunOnce エントリをサポートします。

  RunOnce レジストリ値の名前は、**InstallShield セットアップ %n** で、**%n** は一意のキー名として使用可能な最も小さい整数です。レジストリ値のデータは、再起動の後にインストールを実行する完全なコマンド ラインです。コマンド ラインには、Setup.exe のパスとファイル名が含まれます。

  再起動の後、インストールは Disk1 の場所である DISK1TARGET から実行します。したがって、再起動の後にインストールを実行する場合、インストールに必要な Disk1 ファイルがインストール済みでなくてはなりません。

  再起動の後にキャッシュされたインストール ファイルを使用することはできないため、これはアンインストールには適用しませんので、ご注意ください。

インストールが再開の後に実行すると (つまり、/reboot パラメーターを使って実行すると)、以下が発生します:

- 登録が必要であるとインストールによって記録された自己登録ファイルはすべて登録されます。
- **OnRebooted** イベント ハンドラーが呼び出されます。

アンインストールでは、インストールは再開の後に実行しません。したがって、これらの処理は一切行われません。

Windows Installer ベースのインストールのデバッグ

MSI デバッガーを使用して、Windows Installer ベースのリリースをデバッグすることができます。MSI デバッガーを使用して、パッケージの [ユーザー インターフェイス] と [実行] シーケンスの順番に従って Windows Installer のプロパティを表示/設定することができます。

ブレークポイントに到達するまで各アクションおよびダイアログが実行され、ブレークポイントに到達した時点で実行が停止します。ここで、ウォッチ ウィンドウと変数 ウィンドウにプロパティを表示し、設定することができます。最後に、残りのアクションを順番に実行するか、または、デバッガーを中止します。

メモ: *MSI デバッガーと InstallScript デバッガーは目的がまったく異なる点にご注意ください。セットアップのデザインについての詳しい情報*

MSI デバッガーを開始する

MSI デバッガーを開始する前に、InstallShield でリリースをビルドし、[MSI デバッガー] ビューを開く必要があります。さらに、特定のアクションまたはダイアログでのインストール状態を表示できるように、常にブレークポイントを設定します。

MSI デバッガーを開いてデバッグを開始すると、その時点で [リリース] ビューで強調されているリリースがデバッグされます。リリースをまだビルドしていない場合、またはリリースが Setup.exe で圧縮されている場合、MSI デバッガーには何も表示されず、ツールバーやメニュー項目も使用できません。パッケージをデバッグするには、ビルドされたリリースに切り替えるか、または必要なオプションで再ビルドします。

タスク ***デバッグを開始するには、以下のいずれかを行います。***

- MSI デバッガーのツールバーで **[MSI デバッガー]** ボタンをクリックする。
- MSI デバッガーのツールバーで **[ステップ オーバー]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[MSI デバッガーの開始]** をクリックします。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ステップ オーバー]** をクリックします。
- F5 を押す。
- F11 を押す。

インストールのデバッグが実行されます。ブレーク ポイントに到達すると再び MSI デバッガーが強調され、インストールの実行は停止します。MSI Debugger で、次のいずれかの操作を実行できます。

- Windows Installer のプロパティを参照および設定する。
- 残りのインストールを順番に実行する。
- MSI デバッガーおよび/またはインストールを終了する。

MSI デバッガーでブレークポイントを設定する

タスク ***MSI デバッガーでブレーク ポイントを設定するには、以下の手順を実行します:***

1. **[リリース]** ビューで、デバッグを行うリリースを選択します。
2. **[MSI デバッガー]** ビューを開きます。MSI デバッガーでは、2 つのリストが表示されます。最初に [ユーザー インターフェイス] シーケンスにおける標準およびカスタム アクションがすべて表示され、次にインストールのダイアログがすべて表示されます。
3. ブレークするアクションまたはダイアログを含む行にカーソルを置きます。
4. 次のいずれかの方法で、この行にブレークポイントを設定します。
   - MSI デバッガーのツールーバー上の **[ブレークポイントの切り替え]** ボタンをクリックします。
   - **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ブレークポイントの切り替え]** をクリックします。
   - F9 を押す。

ブレークポイントは、デバッグを開始する前、またはデバッグ セッションを開いている間に設定できます。

ブレークポイントに到達したら、プロパティを表示して設定するか、またはデバッグを続行します。

ヒント: アクションに条件が存在する場合は、各アクションの後にその条件がリストされます。条件付のアクションには慎重にブレークポイントを設定します。条件の評価が True にならないとアクションは実行されず、その部分は飛ばされます。

ブレークポイントを削除する

タスク ***ブレークポイントを削除するには、次の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[MSI デバッガー]** をクリックします。
2. ブレークポイントを削除するには、ブレークポイントのある行にカーソルを置き、**[ブレークポイントの切り替え]** ボタンをクリックするか F9 を押します。

タスク ***デバッガーですべてのブレークポイントの設定を解除するには、以下のいずれかを行います。***

- [**すべてのブレークポイントの削除**] ボタンをクリックする。
- SHIFT +F5 を押す。

MSI デバッガーでプロパティの表示と設定を行う

インストールを順番に実行しているために MSI デバッガーがブレークポイント、アクションまたはダイアログで停止した場合、Windows Installer のプロパティを表示し、その値を ウォッチ ウィンドウ、または変数ウィンドウで実行時に変更できます。

変数ウィンドウ

変数ウィンドウには、データベースの Property テーブル (InstallShield の [プロパティ マネージャー] ビュー に表示されます) にあるすべてのプロパティとその現在の値が表示されます。

インストールの実行中にプロパティの値を変更するには、[プロパティ値] 列を編集します。

変数ウィンドウが表示されない場合は、[表示] メニューから [変数] を選択します。

ウォッチ ウィンドウ

ウォッチ ウィンドウで、プロパティの名前を入力して、インストールの任意の時点でプロパティの値を確認することができます。たとえば、TARGETDIR が実行時に何に解決されるかを見るには、Name 列に **「TARGETDIR」** と入力します。

インストールの実行中にプロパティの値を変更するには、[プロパティ値] 列を編集します。

ウォッチ ウィンドウが表示されない場合は、[表示] メニューで [監視] をクリックします。

MSI デバッガーでアクションにステップインする

MSI デバッガーでカスタム アクションにステップインすると、コードをデバッグできます。次の種類のカスタム アクションにステップインしたとき、登録済みのデバッガーを起動できます。

- Windows Installer .dll ファイル カスタム アクション
- 標準 .dll ファイル カスタム アクション
- Visual Basic Script または JavaScript カスタム アクション
- InstallScript カスタム アクション

タスク ***MSI デバッガーで現在のアクションにステップインするには、次のいずれかを行います。***

- MSI デバッガーのツールバーで **[ステップイン]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ステップ イン]** をクリックします。
- F11 を押す。

ダイアログが開き、登録済みのデバッガーを起動するかどうかたずねます。[はい] を選択するとデバッガーが起動します。[いいえ] を選択して、次のカスタム アクションにステップ オーバーします。

メモ: *InstallShield ではエントリ ポイントのソース コードが提供されていないため、登録済みデバッガーが起動されると一部のマシン コードが表示されます。コードを見るには、[ステップイン] ボタンを 2 度クリックする必要があります。*

MSI デバッガーでアクションを順番に実行する

タスク ***MSI デバッガーで現在のアクションまたはダイアログを順番に実行してデバッグを続行するには、次のいずれかを実行します:***

- MSI デバッガーのツールバーで **[ステップ オーバー]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[ステップ オーバー]** をクリックします。
- F10 を押す。

MSI デバッガーが標準またはカスタム アクションのブレークポイントに到達すると、インストールのデバッグは停止し、MSI デバッガーが強調されます。ここで、インストーラーのプロパティをウォッチし、設定することができます。

引き続きアクションを順番に実行するには、F10 キーを押し続けます。セットアップ ウィザードの各ダイアログで [次へ] をクリックしないとシーケンスが進行しない場合もあります。アクションの戻り値を表示するには、アクション実行後にカーソルをアクションの上に置きます。戻り値はツール ヒントとして表示されます。アクションが正常に実行されると、ツール ヒントには ERROR_SUCCESS のように表示されます。

アクションにブレークポイントを設定すると、MSI デバッガーは後続のアクションでブレークすることになりますが、ダイアログにブレークポイントを設定すると単にそのダイアログに移動するだけで、次のブレークポイントまで介入することはありません。

MSI デバッガーを停止する

タスク

リリースのデバッグを中止するには、以下のいずれかを行います:

- MSI デバッガーのツールバーで **[デバッガーの中止]** ボタンをクリックする。
- **[ビルド]** メニューで **[MSI デバッガー]** をポイントし、**[MSI デバッガーの停止]** をクリックします。
- SHIFT +F5 を押す。

デバッグ セッションを終了しようとするときにデバッガーがダイアログ上のブレイクポイントで停止した場合、InstallShield は、「デバッグを中止するには、ダイアログをすべて閉じてください。」というメッセージをメッセージ ボックスに表示します。開いているダイアログを閉じるには、まず F11 を押して、現在のブレークポイントをステップ オーバーします。それから開いているすべてのダイアログを閉じます。

インストールをキャンセルしても MSI デバッグを中止できます。

インストールを複数のディスクまたは CD に分割する

プログラムが大きくなるにつれて、ディスクの分割の必要性が増します。このような場合、何年も前には複数のフロッピー ディスクで製品を出荷していました。現在の標準は CD-ROM または DVD を使用することです。CD または DVD の容量はフロッピーディスクよりも非常に大きいですが、多くの製品ではそれ以上のスペースを必要とします。マルチメディア チュートリアル、大きなヘルプ ライブラリ、画像の多いプログラムなどが製品に入っていると、標準 CD サイズである 650 MB を超える場合があります。

CD、DVD、カスタムサイズのメディア ディスクにかかわらず、インストールが 1 枚のディスクに収まりきらない場合は、複数のディスクに分割する必要があります。

リリース ウィザードを使用する

複数のディスクにインストールを分割する方法を定義する方法は、リリース ウィザードを使用することです。ウィザードは、選択したメディアのサイズにかかわりなくリリース作成を手順を追って説明します。ここで複数のディスクにファイルを分割する方法を指定できます。また、これらのファイルに適用する圧縮も指定することができます。

ウィザードでは、InstallShield が必要に応じてインストールを複数のディスク上に自動的に分割するように設定することも、またインストール ファイルの分割方法をカスタマイズすることもできます。インストールを複数のディスクに分割する方法をカスタマイズする場合は、インストールの問題を防ぐために「ディスク分割規則」を参照してください。

制限

Windows Installer サービスの制限のために、リムーバブル ディスク以外のディスク上で複数ディスク インストールを実行することはできません。たとえば、インストールが 2 つの CD に分割されている場合、インストールをテストするためには、物理的にこの CD を作成する必要があります。インストールを固定のディスクから実行しようとすると、このインストールは失敗します。

複数ディスク インストールを圧縮する

Setup.exe も .msi ファイルも複数のディスクに分割することができないので、ソース ファイルはこれらのファイルと別に保管する必要があります。サイズ制限のないネットワーク インストールを作成してすべてのファイルを圧縮するように指定した場合で、.msi ファイルまたはSetup.exeの作成を選択すると、これらの圧縮ファイルは .msi または Setup.exe の中に配置されます。その他のすべてのメディア タイプでは、Setup.exe、.msi ファイル、およびすべてのソースファイルが格納されてい .cab ファイル は個別にメディアに配置されます。

たとえば、1.5 MB のファイル、Setup.exe、および Windows NT 用のインストール ファイルがそれぞれ格納された 3 つの機能を含むインストールがあって、サイズが 2 MB のカスタム メディア タイプを作成するとします。ビルドによって複数のディスクが作成されます。ディスク 1 には Setup.exe、InstMsiW.exe (Windows NT マシンに Windows Installer サービスをインストールするロジックが含まれています)、Setup.ini (Setup.exe を含むインストールに必須) および .msi ファイルが含まれます。残りのディスクには、ソースファイルの圧縮コピーを保管する .cab ファイルが格納されます。

ディスク分割規則

リリース ウィザードで、インストールを必要な数のディスクに自動的に分割させるように選択することができます。ウィザードは以下の規則にしたがいます。すべての複数ディスク インストールはこの規則にしたがう必要があります。複数のディスクにインストールを分割する方法をカスタマイズする場合、これらの規則にしたがいます。

- **Setup.exe は、ディスク 1 上にある必要があります。**この規則は、インストールの出荷にフロッピー ディスクを使用し、Setup.exe を出荷する必要があるときに適用します。Setup.exe のサイズ – Windows 9x、Windows NT、またはその両方など選んだプラットフォームによりますが、可能範囲は 1.31 MB から 2.58 MB です。Windows Installer サービスのインストール ファイルもビルドに含まれています。選択したプラットフォームに応じて InstMsiW.exe または InstMsiA.exe と呼ばれるこのファイルも、ディスク 1 上に置く必要があります。
- **ディスク 1 上に置く必要のある 2 番目のファイルは、ビルドする .msi ファイルです。**.msi ファイル、Setup.exe および Windows Installer インストール ファイルの合計サイズは、標準 1.4 MB フロッピーディスクの最大容量を超えています。インストールをフロッピー ディスクで配布する場合、2 つのオプションがあります。
  - Setup.exe を含めずにインストールを配布する。このオプションは、InstallScript MSI プロジェクトの作成時には使用できません。
  - 2 つ目のオプションは、フロッピーディスク以外のメディアでセットアップを配布することです。
- **インストールのすべてのトランスフォームも最初のディスクに置く必要があります。**たとえば、複数のランタイム言語を使用するインストールを作成する場合には、それぞれの言語に対して .mst ファイルが作成されます。このファイルはインストール データベースに適用され、それによって [言語選択] ダイアログで選択された言語を提供します。
- **インストールに含まれる再配布可能ファイル (マージ モジュールを含む) は、最初のディスク上に置く必要があります。**再配布可能ファイルは、スクリプトのビルド時にインストール データベースへストリームされます。このプロセスは、再配布可能ファイルを別のディスクに配置した場合は完了できません。

カスタム ディスク分割

リリース ウィザードを使うと、ディスクの分割に関する設定を行うパネルが表示されます。必要に応じて、ウィザードによって自動的にディスクでインストールを分割するか、ディスク分割を手動で定義することもできます。各ファイルの配置場所を指定することはできませんが、各機能が存在するディスクを指定することはできます。コンポーネントおよびファイルは、機能と同じディスクに自動的に配置されます。自分でレイアウトするかウィザードで自動的に行うかに関わらず、InstallShield が自動的に従うガイドラインが他にもあります。

各ディスクに配置する機能の定義に加え、各ディスクのボリュームラベルをカスタマイズし、ディスク プロンプトを作成して、ウィザードでディスクのサイズを強制するよう選択することができます。ディスクのレイアウトをカスタマイズする手順については、リリース ウィザードのドキュメントで説明しています。

ディスクのプロンプト

ディスクのプロンプトは、インストールを続行するために別のディスクをドライブに挿入する必要があるときに、エンドユーザーに表示されるメッセージです。標準のディスクのプロンプトがありますが、製品に合わせてプロンプトを作成することもできます。

文字列は、実際には、以下のようにプロジェクト全体の多数のソースで表示されます。

1. 終了プロンプトは、ダイレクト エディターの Error テーブルにある error 1302 として発信されます。
2. この値はプロジェクトの文字列エントリー覧からのもので、日本語のデフォルト値は、「次のディスクを挿入してください: *[2]*」です。
3. Windows Installer は、DiskPrompt プロパティの値で *[2]* を解決します。この値はプロパティ マネージャーで *[1]* に設定されています。
4. 最後に、Windows Installer は、"ディスクのプロンプト" フィールドに入力した文字列に対して、*[1]* を評価します。

ほとんどの場合、ディスク ボリュームと同じ名前を入力します。たとえば、デフォルト値が変更しなかったと想定して、**Disk8** のディスク プロンプトを入力すると、ユーザーに「次のディスクを挿入してください : Disk8」というプロンプトが表示されます。

ディスク番号の表示は、疑問符で代用させることもできます。この疑問符は、自動的に正しいディスク番号に置き換えられます。エンドユーザーに対して「次のディスクを挿入してください: DISK4」とプロンプトする場合、4 番目のディスク名に **DISK?** と入力します。ディスクのプロンプトは、インストールの各ディスクに対して指定する必要があります。

ボリューム ラベル

ボリューム ラベルは、インストールが配置されているディスクの名前です。インストールで新しいディスクが要求されると、Windows Installer エンジンによってディスクのボリューム ラベルがインストール ータベースのボリューム ラベルと一致するかどうかが確認されます。ラベルが一致しない場合、サービスは、異なるディスクがドライブに挿入されていると解釈します。ディスク分割オプションをカスタマイズする際に、ボリュームラベルを変更できます。デフォルトのボリュームラベルは、DISK1、DISK2、DISK3 などです。インストールをビルドする際、これらのディレクトリが出力場所に作成されていることを確認できます。インストールを保存したディスクのボリューム ラベルは、ウィザードで指定された名前と同じである必要があります。

***注意** ・インストールがビルドされた後にボリューム ラベルを変更することはできません。正しいボリューム名を指定しなかった場合、Windows Installer サービスはディスクを認識できず、インストールに失敗します。*

ディスク番号の指定は、疑問符で代用させることができます。たとえば、ディスクのボリューム ラベルを **InstallShield Disk 1** や **InstallShield Disk 2** などと設定する場合、すべてのディスクのボリューム ラベルを **InstallShield Disk ?** と変更しておきます。この疑問符は、ビルド時にディスク番号に置き直されます。

ディスク サイズを強制する

"ディスクサイズの強制" オプションを使って、現在のレイアウトが指定したディスクに確実に合うようにできます。このオプションが選択されておらず、作成するディスクが選択したメディアのサイズを超えた場合、ウィザードにより自動的に別のディスクが作成され、機能が分割されます。このオプションを選択した場合にディスクサイズがメディアサイズを超えると、ビルドは失敗し、ビルドエラー 1531 が発生します。その後、機能のサイズを小さくするか、別のレイアウトを試すことができます。

追加情報

インストールのディスク分割を指定し終わってから、インストールに機能を追加すると、これらの機能はインストールの最後のディスクに自動的に追加されます。リリース ウィザードを使用して、いつでもインストールに新しいディスクを追加できます。

セットアップランチャーの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallScript インストールは、常にセットアップランチャーを含みます。

アドバンスト UI およびスイート/アドバンスト UI インストールでは、常にセットアップ ランチャーが作成されます。セットアップ ランチャーによって、.msi、.msp、InstallScript、および .exe パッケージが実行されます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストールの作成」を参照してください。

InstallShieldでは、インストールに Setup.exe セットアップ ランチャーを含めるかどうかを指定できます。Setup.exe セットアップ ランチャーは、次のような場合に必要です:

- 必要に応じて、自動的にターゲット システムの Windows Installer エンジンを更新またはインストールする。
- 複数言語のインストール プロジェクトをビルドしていて、[言語の選択] ダイアログを表示する場合。
- プロジェクトに InstallShield 前提条件が含まれている場合。
- プロジェクトに .NET Framework が含まれている場合。
- InstallScript MSI プロジェクトのリリースをビルドしていて、ユーザー インターフェイスに従来のスタイルを用いる場合。
- 製品構成に製品の複数インスタンスをインストールするサポートが含まれているリリースをビルドして、かつインスタンスの選択ダイアログを適宜表示する場合。
- ビルボードを含む基本の MSI プロジェクトのリリースをビルドする場合。

Setup.exe セットアップ ランチャーは、上記のシナリオを制御するブートストラップ アプリケーションです。

[リリース] ビューにあるリリースの Setup.exe タブでは、Setup.exe ランチャーを使用するかどうかなどの情報を指定することができます。詳細については、「リリースの [Setup.exe] タブ」を参照してください。

ヒント: セットアップランチャーの要件は、リリース ウィザードの [セットアップランチャー] パネルでも指定できます。

Windows Installer と Setup.exe

Windows Installer がターゲット システムに存在しない可能性がある、または、インストールが特定のバージョンの Windows Installer でのみ使用可能な機能に依存している可能性がある場合、InstallShield では、Windows Installer をインストールする再配布可能ファイルをインストールに含めるオプションが提供されます。このオプションを選択すると、InstallShield はターゲット システム上で Windows Installer の存在を確認する Setup.exe ランチャーを作成します。Windows Installer がインストールされていない場合や、または新しいバージョンをインストールする必要がある場合、 Setup.exe は Windows Installer インストールを起動してからインストール パッケージを起動します。

詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

複数言語のサポートと Setup.exe

[リリース] ビューの [ビルド] タブで選択した言語関連の設定に応じて、Setup.exe を実行するエンドユーザーはインストールを実行する言語を選択することができます。

複数の言語でインストールを配布する際に、実行するインストールの言語バージョンの選択オプションをエンドユーザーに提供する場合、Setup.exe が常に必要になります。複数言語のサポートに関する詳しい情報は、「複数言語インストールの作成」をご覧ください。

InstallShield 前提条件と Setup.exe

InstallShield 前提条件を含むプロジェクトでは、Setup.exe によってターゲット システムが InstallShield 前提条件を満たしているかどうかが確認されるため、Setup.exe が必須になります。条件が満たされた場合、Setup.exe は InstallShield 前提条件をインストールします。詳細については、「インストール プロジェクトに含まれている InstallShield 前提条件を利用する」を参照してください。

.NET Framework と Setup.exe

.NET Framework を含むプロジェクトでは、Setup.exe がターゲット システムを検索して .NET Framework が存在するかどうかを確認するため、Setup.exe が必要です。適切なバージョンの .NET Framework が存在しない場合、Setup.exe によってそのバージョンがインストールされます。

.NET Framework の追加については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。

InstallScript MSI プロジェクトと Setup.exe

従来のスタイルの InstallScript MSI インストールでは、Setup.exe が InstallScript エンジンを開始してユーザー インターフェイスを表示するため、Setup.exe が必須になります。この従来のスタイルでは、InstallScript エンジンが外部 UI ハンドラーの役割を果します。

InstallScript MSI インストールで新しいスタイルの UI を使用すると、InstallShield が InstallScript エンジンを .msi パッケージ内部に埋め込むため、セットアップランチャーが必須ではありません。

2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

複数インスタンスのサポートと Setup.exe

製品を構成して、エンドユーザーがターゲット システムで製品を各コンテキストで複数回インストールできるようにする場合、インスタンスの選択ダイアログを適宜表示するには、Setup.exe セットアップ ランチャーが必ず必要になります。Setup.exe ファイルがインスタンスの選択ダイアログを表示します。

複数インスタンスのサポートに関する詳しい情報は、「製品の複数のインスタンスをインストールする」をご覧ください。

ビルボードと Setup.exe

ビルボードは、実行時に Setup.exe によって表示されるため、ビルボードを含む基本の MSI プロジェクトには、Setup.exe が必要です。

ビルボードに関する詳細については、「基本の MSI インストールでビルボードを表示する」を参照してください。

セットアップランチャーのファイルのプロパティをカスタマイズする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

プロジェクトによって異なる情報がある場合は、その内容が記載されています。

InstallShield では、Setup.exe セットアップ起動プログラムのバージョン リソースにカスタム情報を使用できます。情報が、セットアップ起動プログラムの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。

Setup.exe のプロパティを構成するための InstallShield の設定

次のテーブルは、Windows の [前提条件] ダイアログ ボックスに含まれている様々なプロパティと、それらを構成するのに使用できる InstallShield で対応する設定の一覧です。

メモ: *[プロパティ] ダイアログ ボックスは、Windows のバージョンによって異なります。たとえば、Windows 7 システムでは、バージョン リソース情報は [プロパティ] ダイアログ ボックスの [詳細] タブに表示されます。一方、Windows XP システムでは、バージョン リソース情報は、同じダイアログボックスの [バージョン] タブに表示されます。*

また、Windows の一部のバージョンは、[プロパティ] ダイアログ ボックスの一部の設定を表示しません。

テーブル 24 • Setup.exe のプロパティの情報ソース

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
会社名	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI および InstallScript MSI プロジェクトでは、[一般情報] ビューの "発行者" 設定です。 • InstallScript プロジェクトでは、[一般情報] ビューの "会社名" 設定です。 • アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは、InstallShield で提供されたカスタム値、またはデフォルトの InstallShield 固有の名前が使用されます。 InstallShield を使って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの "会社名" フィールドに、独自のカスタム値を使用するには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **"会社名"** 設定に、Setup.exe ファイルのプロパティ内の "会社名" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"会社名" 設定を空白のままにすると、InstallShield は [一般情報] ビュー "発行者" 設定を使用します。この設定が空白の場合、InstallShield はデフォルトの InstallShield 固有の名前を使用します。
製品名	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合、[リリース] ビューにある製品構成の "製品名" 設定です。設定が空白の場合、InstallShield は [一般情報] ビューの "製品名" 設定を使います。 • アドバンスト UI、InstallScript、およびスイート/アドバンスト UI プロジェクトでは、[一般情報] ビューの "製品名" 設定です。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
製品バージョン	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合、[リリース] ビューにある製品構成の "製品バージョン" 設定です。設定が空白の場合、InstallShield は [一般情報] ビューの "製品バージョン" 設定を使います。 • InstallScript プロジェクトでは、[一般情報] ビューの "製品バージョン" 設定です。 • アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、[一般情報] ビューの "製品名" 設定です。この値は、[リリース] ビューを使用して変更することができます。 アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、[一般情報] ビューの値をオーバーライドするには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **"バージョン"** 設定に、Setup.exe ファイルのプロパティ内の "バージョン" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"バージョン" 設定を空白のままにすると、InstallShield は [一般情報] ビューの "バージョン" 設定を使用します。この設定が空白の場合、InstallShield はデフォルトの InstallShield 固有の値を使用します。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
ファイル バージョン	このフィールドの値を含む設定は、使用するプロジェクトの種類によって異なります。 • 基本の MSI、InstallScript MSI、InstallScript: **[リリース]** ビューの **[カスタム バージョンのプロパティを使用する]** の下にある Setup.exe の **"ファイル バージョン"** 設定 を使用します。 この設定が空白の場合、InstallShield はデフォルトで "製品バージョン" の値を使用します。 ***メモ:*** *ファイル バージョンは、常に 4 つのフィールドが含まれます。"ファイルバージョン" に 4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、ファイル バージョンとして 1.1 を指定すると、Setup.exe のバージョン リソースで使用されるファイル バージョンは 1.1.0.0 となります。* **"ファイル バージョン"** を上書きする場合、**"カスタム バージョンのプロパティを使用する"** を **[はい]** に設定して、**"ファイル バージョン"** フィールドにバージョンを入力してください。 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **Setup.exe** タブで **"カスタム バージョンのプロパティを使用する"** 設定に **[はい]** を選択します。 3. **"ファイル バージョン"** フィールドに、Setup.exe ファイルのプロパティ内の "ファイル バージョン" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択した場合、または "ファイル バージョン" 設定を空白のままに残した場合、InstallShield は基本の MSI および InstallScript MSI の場合は **[製品の構成]** ビューから、および InstallScript の場合は **[一般情報]** ビューから製品バージョンの "ファイル バージョン" 設定を使用します。この設定が空白の場合、はデフォルトの InstallShield 固有の値を使用します。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
著作権情報	InstallShield は、提供されたカスタム値、またはデフォルトの InstallShield 著作権情報を使用します。 "著作権情報" フィールドの値に独自の値を使用するには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **" ランチャーの著作権"** 設定に、Setup.exe ファイルのプロパティ内の "著作権" フィールドに使用するテキストを入力します。 "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"起動ツールの著作権" 設定を空白のままにすると、InstallShield はデフォルトの InstallShield 著作権情報を使用します。
ファイルの説明	InstallShield は、提供されたカスタム値またはデフォルトの InstallShield Setup.exe の説明を使用します。 "説明" フィールドの値に独自の値を使用するには、以下の手順に従います: 1. **[リリース]** ビューで、構成するリリースを選択します。 2. **"カスタム バージョンのプロパティを使用する"** 設定での **Setup.exe** タブで **[はい]** を選択します。 3. **"ファイルの説明"** 設定に、Setup.exe ファイルのプロパティ内の "説明" フィールドに使用するテキストを入力します。 ***プロジェクト:*** *アドバンスト UI、InstallScript またはスイート/アドバンスト UI プロジェクトの場合: "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択するか、"ファイルの説明" 設定を空白のままにすると、InstallShield はデフォルトの InstallShield* Setup.exe *の説明を使用します。* *基本の MSI または InstallScript MSI プロジェクトの場合: "カスタム バージョンのプロパティを使用する" 設定に [いいえ] を選択すると、InstallShield はデフォルトの InstallShield* Setup.exe *の説明を使用します。[はい] を選択した場合に "ファイルの説明" 設定を空白のままにすると、InstallShield は [リリース] ビュー内にある製品構成の "コメント" 設定の値を使用します。この設定が空白の場合、InstallShield は [一般情報] ビューの "[概要情報ストリーム] のコメント" 設定を使います。この設定が空白の場合、InstallShield はデフォルトの InstallShield* Setup.exe *の説明を使用します。*
言語	このフィールドには、[リリース] ビュー内の現在ビルド中のリリースの "デフォルト言語" 設定で指定された言語が使用されます。

テーブル 24 • Setup.exe のプロパティの情報ソース (続き.)

Setup.exe プロパティ	Setup.exe プロパティを構成するための InstallShield 設定
元のファイル名	InstallShield は、提供されたカスタム値、またはデフォルトの InstallShield 名 setup.exe のどちらかを使用します。 "元のファイル名" フィールドの値に独自の値を使用するには、以下の手順に従います: 1. [メディア] の下にあるビュー リストで、**[リリース]** をクリックします。 2. **[リリース]** エクスプローラーを右クリックし、**[新しい製品構成]** をクリックします。 リリース エクスプローラーに、新しい製品構成が追加されます。 InstallShield の [全般] タブに製品構成の設定が一覧表示されます。 3. **"セットアップ ファイル名"** 設定の **[全般]** タブに、"元のファイル名" フィールドに使用するテキストを入力します。 InstallShield がビルド時に生成する元のファイル名に使用するファイル名 (.exe ファイル拡張子を除く) を指定します。

Setup.exe のプロパティ ダイアログ ボックスのサンプル

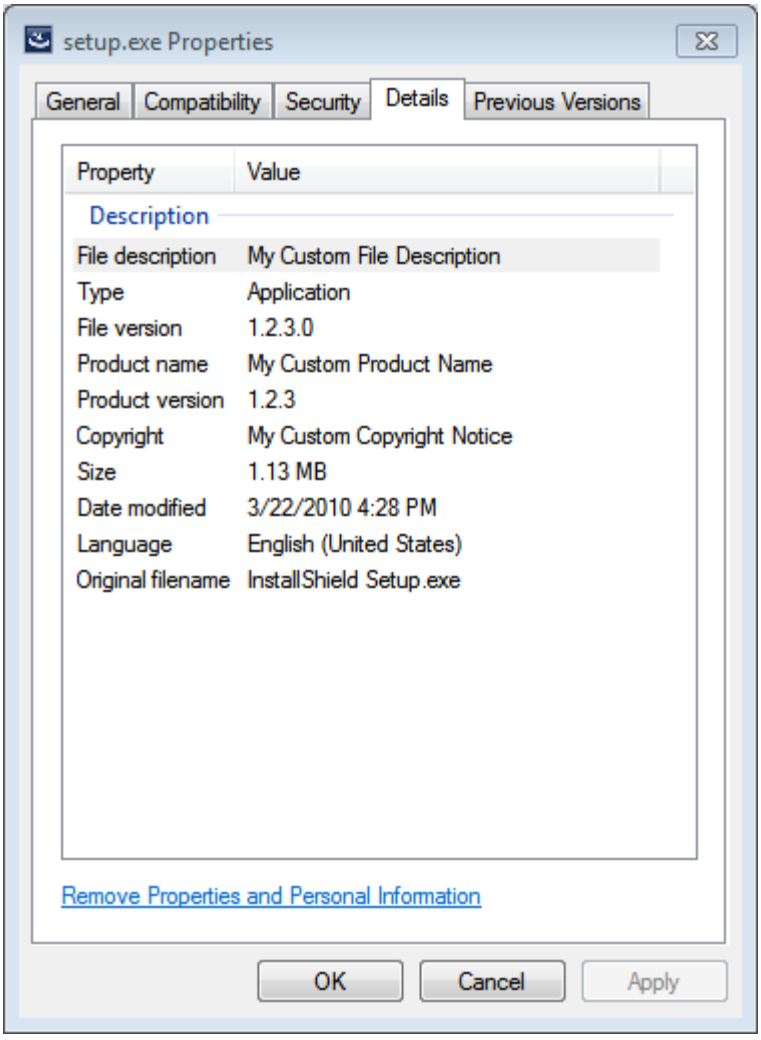


図 1: Windows 7 マシン上における Setup.exe のプロパティのサンプル

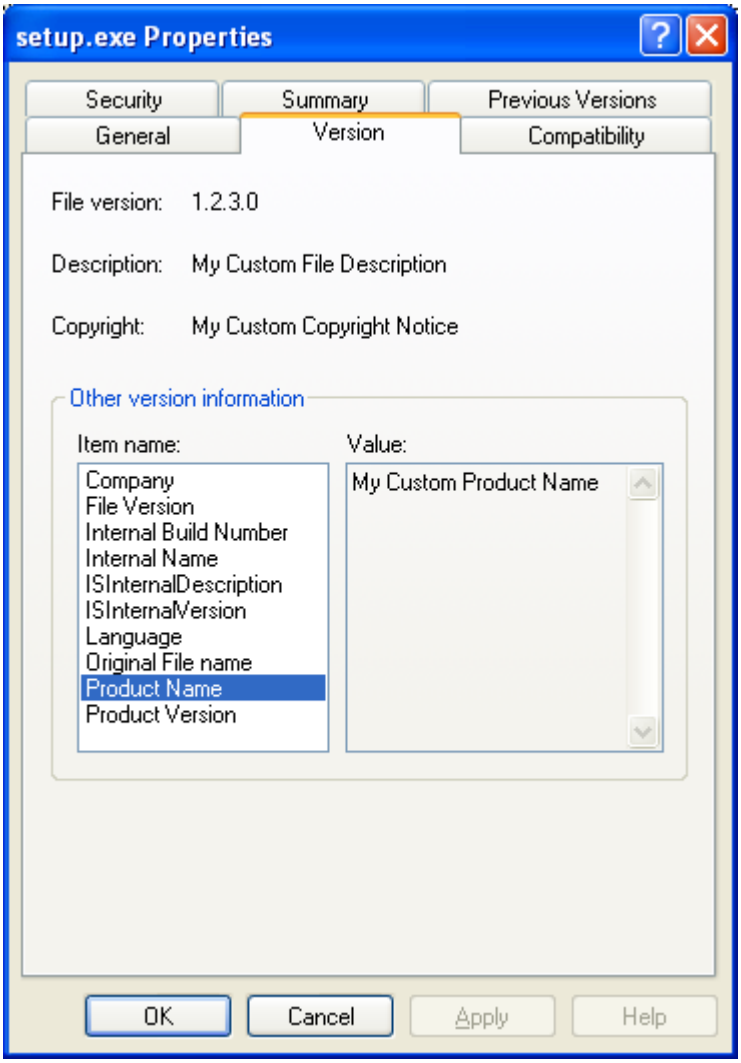


図 2: Windows XP マシン上における Setup.exe のプロパティのサンプル

セットアップランチャーのファイルのアイコンを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield では、Setup.exe セットアップ ランチャーに使用するアイコンを指定できます。アイコンには、.exe、.dll、または .ico ファイルを使用できます。

エンド ユーザーが Windows Explorer で Setup.exe ファイルを参照したときにも、このアイコンが表示されます。アイコンはまた、Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。

アイコンを指定しなかった場合、InstallShield は Setup.exe ファイルのデフォルト アイコンを使用します。

タスク ***Setup.exe ファイルのアイコンを指定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **Setup.exe** タブをクリックします。
4. [**Setup.exe アイコン ファイル**] 設定には、InstallShield がビルド時に Setup.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタン (...) をクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

リリースを次回ビルドするとき、InstallShield が Setup.exe ファイルに指定されたアイコンを使用します。

インストールを配布する

インストールを作成したあと、場合により、指定した場所へ配布する必要があります。これにはネットワーク ドライブ、CD、ローカル ドライブの別の場所、または FTP サイトが可能です。インストールを配布すると、インストールをビルドしたときに作成されたディスク イメージが、指定の場所にコピーされます。

フォルダーまたは FTP サイトにリリースを自動的に配布する

リリースのビルドとテストが完了すると、残る作業は、それを適切な場所に配布するのみです。リリースを適切な場所に手動でコピーすることもできますし、[リリース] ビューの [イベント] タブを使って、InstallShield がリリースを適切な場所 (ローカル/ネットワークの場所、または FTP サイト) へ自動的にコピーするように構成することもできます。

タスク ***InstallShield を構成して、リリースを特定の場所へ自動的に配布するようにするには、以下の手順に従います。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[イベント]** タブをクリックします。

4. 設定を適切に構成します。**[イベント]** タブにある設定に関する詳しい情報は、「リリースの [イベント] タブ」を参照してください。

メモ: *インストールが 1 つのディスクのみで構成される場合は、リリースの保存場所に Disk1 フォルダーの内容がコピーされますが、フォルダーそのものはコピーされません。インストールが複数のディスクで構成される場合は、フォルダーおよびその内容がリリースの保存場所にコピーされます。*

プロジェクト: *InstallScript および InstallScript Object プロジェクトの場合、リリースがビルドされるたびに、InstallShield が、自動的にそのリリースを [イベント] タブで指定した場所にコピーします。*

その他のプロジェクトの種類の場合、[リリース] ビューでリリースを右クリックして、[配布] をクリックすると、リリースに関連するファイルがすべて指定された場所にコピーされます。

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

[イベント] タブでフォルダーおよび FTP サイトを指定した場合、ファイルは FTP ロケーションにのみコピーされます。

基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトで、ビルドするたびに、ビルド エンジンがリリースを指定の場所にコピーするように設定する場合、"ビルド後、配布する" 設定で [はい] を選択します。

ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する

エディション: *仮想マシンへのリリース配布機能は、InstallShield Premier Edition でご利用いただけます。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し (指定されている場合)、VM の電源をオンにしてインストールを VM にコピーし、テストが可能な状態になるようプロジェクトのリリースを構成できます。また、これらのテスト準備を必要なときにオンデマンドで行うこともできます。このテスト準備機能により、手作業で行う処理を減らし、テストにかかる時間を短縮できます。

VM は Microsoft Hyper-V Server、VMware ESX または ESXi Server、または VMware Workstation を指定できます。

VM に配布するための要件

サポートされている VM の1 つに配布する場合、InstallShield では必要な仮想化テクノロジとの対話が必要です。VMWare 仮想化テクノロジ (VMware ESX、ESXi Server oまたは VMware Workstation) を使用する場合、InstallShield と同じマシン上に VMware VIX API が インストールされていなくてはなりません。マシン上に VMware Workstation をインストールするか、http://www.vmware.com/support/developer/vix-api から VMware VIX API をダウンロードおよびインストールすることができます。

VMware Workstation を使用する場合、InstallShield と同じマシン上に VMware Workstation をインストールすることが推奨されます。これによって、InstallShield が特定の VMware Workstation バージョン用に設計された VIX API のバージョンを使用します。VIX API の新しいバージョンも使用できる場合がありますが、現在使っている VMware Workstation にバンドルされている VIX API のバージョンを InstallShield で使用する方法が最も推奨されます。

VM 関連の設定を構成する

タスク

ビルド時またはオンデマンドで VM に配布できるよう、リリースの VM 関連設定を構成するには、以下の手順に従います:

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[イベント]** タブをクリックします。
4. **[仮想マシン]** 領域を展開します。
5. **"構成"** 設定を使って、選択されたリリースを VM に配布するときに使用する VM 構成設定グループの名前を選択します。
   - 新しい VM 構成を作成するには、この設定で省略記号ボタン (...) をクリックします。
   - 既存の VM 構成を編集するには、その構成を選択してから省略記号ボタン (...) をクリックします。

   どちらの場合も **[仮想マシンの構成を編集]** ダイアログが開いて、マシン全体に適用する VM の詳細を構成することができます。このダイアログ ボックスの使い方については、「[仮想マシンの構成を編集] ダイアログ ボックス」を参照してください。

   [仮想マシンの構成を編集] ダイアログ ボックスにある "VM 構成" 設定を指定すると、InstallShield によって指定されたデータが VMConfigurations.xml という名前のファイルに書き込まれます。この VMConfigurations.xml ファイルは、マシン全体に適用するファイルで、一度構成を行うと他の InstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルをビルド マシンで使用することも可能です。詳細については、「リリースの配布用の仮想マシン設定を共有する」を参照してください。
6. 必要に応じて、**仮想マシン**の残りの設定を構成します。

   これらの残りの設定を使って VM の詳細を構成すると、[仮想マシンの構成を編集] ダイアログ ボックスで入力済みの、マシン全体に適用される値が上書きされます。

   [仮想マシン] 領域にある各設定についての詳細は、「リリースの [イベント] タブ」を参照してください。

ヒント: *ビルドが成功するたびに指定された VMにリリースを配布するには、"ビルド後に配布" 設定を [はい] に選択します。*

オンデマンドでビルド済みのリリースをステージングする場合、リリースを右クリックしてから [VM に配布] をクリックします。

インターネットで配布するインストールを準備する

ユーザーがソフトウェアを受け取る方法は急激に変化しています。インターネット テクノロジがそれほど発展していなく、高速のインターネット接続が導入される以前は、すべてのソフトウェアが、フロッピー ディスクや CD-ROM などのリムーバブル メディアで発送されていました。ソフトウェアは現在、インターネットから直接ダウンロードされることがほとんどです。この時間およびコストを節約できるソフトウェア配布プロセスを利用するため、より簡単にダウンロードおよびインストールできる方法で、インストールをパッケージする必要があります。

Web 用のインストールには、満たす必要のある条件がいくつかあります。

圧縮サイズ

現在、多くの人が、高速ケーブル モデムや DSL 回線を使用してインターネットに接続していますが、低速度モデムを使用している人も多くいます。アプリケーションのダウンロードに必要なオンライン時間が増加するため、パッケージ サイズは、低速の接続を使用している人にとって非常に重要です。

自己展開

多くのファイル圧縮ユーティリティは、アプリケーション ファイルを解凍するための特別なクライアント側アプリケーションが必要です。別のユーティリティでは、このような必要性のために、エンドユーザーのダウンロードやインストール プロセスが複雑になります。インストール プロセスをシンプルにするには、使用する圧縮ユーティリティが他のアプリケーションを必要としないように、自己展開できる必要があります。

デジタル署名

顧客がソフトウェアをダウンロードおよびインストールするときの安全を確保するため、アプリケーションパッケージをデジタルで署名できます。デジタル署名により、ソフトウェア作成者または企業がエンドユーザーに識別され、発行以来アプリケーション コードが変更または改ざんされていないことが保証されます。アプリケーションにデジタル署名を付加する詳しい方法については、「デジタル署名とセキュリティ」を参照してください。

使いやすさ

インターネット配布のインストールのパッケージにおいて最も重要なことは、使いやすさです。すべてのエンドユーザーが、インストール ファイルを保存する場所を指定したり、コンピューターを検索してこれらのファイルを自分で探すことは望んでいるとは限りません。インストールを圧縮パッケージにシームレスに統合し、必要なインストール開始手順を 1 つだけにする必要があります。

プロキシ サーバーのサポート

特定のファイルがターゲット システム上で必要な場合のみ、インストールがそのファイルをダウンロードするように構成できます。たとえば、Windows Installer エンジン、.NET Framework、および一部の InstallShield 前提条件が、一部またはほとんどのターゲット システム上に既在する可能性があります。これらのファイルをインストールに埋め込む代わりに、必要なファイルだけを実行時にダウンロードするようにプロジェクトを構成することができます。こうすることで、インストール全体のサイズを抑えることができます。

エンド ユーザーがプロキシ サーバーを使ってインターネットにアクセスする場合で、インストールがファイルをダウンロードするように構成されているとき、インストールはダウンロード中に、Internet Explorer で手動で構成されたシステム プロキシ設定を使用します。これは、ターゲット システム上で別のブラウザーがデフォルトとして設定されている場合でも同じです。

InstallShield は、Internet Explorer の "設定を自動的に検出する" 設定をサポートしませんので、ご注意ください。(エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されているときに、インストールでファイルのダウンロードが必要な場合、ファイルのダウンロードができないため、インストールは失敗します。エンドユーザーが使用している Internet Explorer で、LAN 接続に対して [設定を自動的に検出する] チェック ボックスが選択されている可能性があるとき、ダウンロードされるように構成する代わりに、すべてのファイルをインストールに埋めこんだ方が良い場合があります。ファイルが埋めこまれている場合、失敗は避けられます。)ただし、InstallShield は Internet Explorer の LAN 接続用にセットアップされた自動構成スクリプト機能をサポートします。

InstallScript インストールと共に配布される再配布可能ファイル

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

次のファイルは、InstallShield によって自動的に作成され、(エンドユーザー使用許諾契約で述べられているように) ソフトウェア配布と共に配布することが可能です。メディアのビルドは、ディスク イメージ フォルダーに必要なため、自動的に次のファイルを含みます。

メイン メディア ファイル

- data<2, 3, 4, ⋯>.cab
- data1.cab
- data1.hdr
- ISSetup.dll
- layout.bin
- setup.exe
- setup.ini
- setup.inx

オプション ファイル

- setup.isn—スキン ファイル
- setup.isn—Skin Filesetup.htm—One Click Install .htm ページ

64 ビット セットアップ ランチャーのビルド

プロジェクト: この情報は、基本の MSI プロジェクトに適用します。

InstallShieldでは、インストールに 64 ビット**Setup.exe** セットアップ ランチャーを含めるかどうかを指定できます。64 ビット **Setup.exe** セットアップ ランチャーは、インストールが 64 ビット システムをターゲットとする場合のみ必要です。デフォルトで、InstallShield は 32 ビット システムだけでなく 64 ビットシステム上の WOW64 上で実行可能な 32 ビット セットアップ ランチャーをビルドします。一部の 64 ビット ターゲット システム、たとえば Windows Server Core システム は、WOW64 (32-bit Windows-on-Windows) をサポートしない場合があります。これらの 64 ビット ターゲット システムは 32 ビット セットアップランチャーを実行できません。

[リリース] ビューにある製品構成の **全般** タブでは、64 ビット セットアップ ランチャーをビルドするかどうかなどの情報を指定することができます。

64 ビット セットアップ ランチャーをビルドしたとき、対応する 64 ビット MSI パッケージがデフォルトでビルドされることはありません。64 ビット MSI パッケージをビルドするには、適切な 64 ビット値 (x64 または Intel64) を使って "テンプレート概要" を構成する必要があります。ピュア 64 ビット MSI パッケージは、インストール中に 32 ビット製品ファイルをコピーまたは 32 ビット カスタム アクションを実行することができません。これは、32 ビット バイナリを WOW64 サポートなしで 64 ビット ターゲット システム上にロードすることができないためです。**[リリース]** ビューにある製品構成の "InstallShield アーキテクチャの検証" 設定を使って、ピュア 32 ビットまたは 64 ビット MSI パッケージをビルドすることができます。InstallShield では、リリース フラグの構成に基づいて、単一プロジェクトから 2 つのインストーラー (32 ビットおよび 64 ビット) を作成することができます。

64 ビット MSI パッケージのビルドについての詳細は、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。

64 ビット セットアップ ランチャーを含む 32 ビット MSI パッケージをビルドする

MSI パッケージで 64 ビット セットアップ ランチャー オプションを含む 32 ビット パッケージのビルドが構成された場合、InstallShield は次の警告メッセージを表示します:

ISDEV: 警告 -7372: セットアップ ランチャーでは 64 ビットのビルドが構成されていますが、テンプレートの概要では 64 ビット MSI パッケージのビルドが構成されていません。

これは 64 ビット セットアップ ランチャーがピュア 32 ビット システム上での実行に失敗し、32 ビット MSI パッケージがピュア 64 ビット システム上でのインストールに失敗することを通知することが目的です。この警告を回避するため、インストーラーがピュア 32 ビット システムおよび 64 ビット システム上の WOW64 をターゲットにする必要があるときは、64 ビット セットアップ ランチャーを使用しないでください。または、MSI テンプレートの概要で、ピュア 64 ビット システムをターゲットにする 64 ビット MSI パッケージを生成するように変更してください。

64 ビット セットアップ ランチャーを使用する InstallScript カスタム アクションのビルド

InstallScript カスタムアクションは、32 ビットセットアップ ランチャーに加えて、64 ビットセットアップ ランチャーでサポートされています。インストール要件に応じて、**[カスタムアクションとシーケンス] ビューの [アクション]** 領域で、**"64 ビット InstallScript を実行"** 設定を構成できます。

InstallScript カスタムアクションが 64 ビット DLL ファイルを読み込む必要がある場合、または 64 ビット ファイル システムと 64 ビットレジストリの場所にアクセス/構成する必要がある場合、**"64 ビットInstallScriptを実行する"** 設定を **[はい]** に設定することをお勧めします。これによって、InstallScript カスタム アクションが 64 ビット プロセスで実行され、64 ビットシステム リソースと正しく対話することが保証されます。

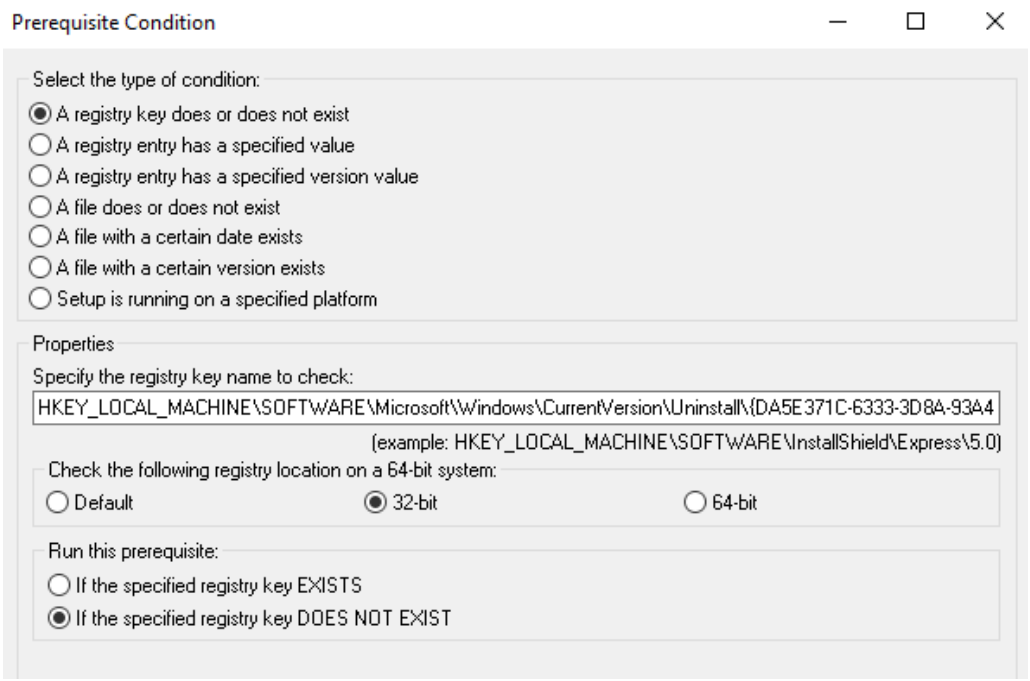
詳細については、「カスタム アクションの設定」を参照してください。

InstallShield 前提条件および 64 ビット セットアップ ランチャー

一般的に InstallShield 前提条件を含むプロジェクトでは、**Setup.exe** によってターゲット システムで InstallShield 前提条件が満たされているかどうかが確認されるため、**Setup.exe** が必須になります。条件が満たされている場合、**Setup.exe** が InstallShield 前提条件をインストールします。

64 ビット セットアップ ランチャーを使用する InstallShield 前提条件を含むプロジェクトのビルドを構成する場合、必ず前提条件のインストーラーが 64 ビットであり、ピュア x64 システムにインストールが可能で、前提条件がレジストリまたはフォルダーの場所で 64 ビットの場所を確認するように構成します。

場合によって、64 ビット セットアップ ランチャーを使う 32 ビット前提条件のインストーラーをビルドしたい場合、32 ビットの場所を確認する条件を構成します。前提条件に含まれるほとんどのレジストリ条件はデフォルトの場所を確認するように構成されています (32 ビット セットアップ ランチャーの場合は 32 ビットの場所、64 ビット セットアップ ランチャーの場合は 64 ビットの場所)。ただし、32 ビットの前提条件インストーラーは、32 ビット レジストリハイブの下にレジストリ キーを作成します。この問題を解決するためには、32 ビットの場所を確認するようにレジストリの場所を InstallShield 前提条件エディターで明示的に変更する必要があります。前提条件は次の通りです:



Windows アプリ パッケージの配布

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

重要: *デスクトップ拡張 (デスクトップ ブリッジ) を含む Windows アプリ パッケージ (.appx) のインストールおよびテストを行うには、Windows 10 Anniversary Update が必要です。Windows アプリ パッケージにデジタル署名を行う場合、InstallShield を Windows 10 または Windows 10 SDK がインストールされているマシン上にインストールする必要があります。*

ヒント: *Windows アプリ パッケージを正しくサイドロードするためには、サイドローディングを有効化する必要があります。サイドローディングに関する詳細は、MSDN 記事「Enable your device for deployment 」を参照してください。*

次のテーブルは、Windows アプリパッケージの配布方法についての概要を説明します:

テーブル 25 • Windows アプリの配布方法

配布方法	説明	配布方法の設定	署名の考慮
Windows ストア	Windows アプリ パッケージを Windows ストアにアップロードします。詳細については、「Windows アプリのパッケージ化」(MSDN 記事) を参照してください。 Windows ストアを通して配布されるパッケージにはすべて、コンテンツを制限する追加ポリシーが適用されます。	Windows ストア	この種類のアプリは、パッケージが Microsoft によって署名されるため、Microsoft にアップロードする前に署名する必要がありません。
スイート/アドバンスト UI プロジェクトにサイドロードを追加	サイドローディング Windows アプリ パッケージをスイート/アドバンスト UI プロジェクトに追加する方法は「サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する」に説明されています。 サイドローディング アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。 WINDOWS アプリ パッケージ (.appx) の **[Windows アプリ パッケージの対象]** 条件や、終了条件あるいはパッケージの対象条件にある **[Windows タイプが存在]** 条件といった条件の確認、およびその他の条件確認を行い、この種類の配布方法を採用します。この種類のシナリオでは、エンド ユーザーが条件付で .msi パッケージまたは Windows アプリ パッケージをインストールできる柔軟性を提供できます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。	直接配布 (サイドロード)	パッケージがサイドロードされ、**また**スイートに追加される場合、Windows アプリ パッケージはスイート/アドバンスト UI プロジェクトにビルドされる前に署名を行う必要があります。
サイドロードおよび Windows アプリ パッケージ (.appx) としての配布	カスタマに対し、Windows アプリ パッケージ (.appx) ファイルをダブルクリックでインストール開始可能な実行可能インストーラーとして配布します。 サイドローディング アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。	直接配布 (サイドロード)	パッケージがサイドロードされる場合、カスタマが独自に署名することを希望しない限り、一般的に署名が必要です。パッケージのダブルクリック インストールを行う前に、あらかじめ署名が必要です。

トランスフォームの作成

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

トランスフォーム (.mst ファイル) は、2 つの .msi データベース間の差分を含むシンプルな Windows Installer データベースです。トランスフォームを使うと、管理者は、インストール パッケージを配布する際、変更した設定をデータベースに適用することができます。

インストールのカスタマイズ

たとえば、ユーザーは会社の部署ごとに異なる方法でインストールをカスタマイズしなくてはならない場合があります。通常のオフィススイートは、スプレッドシート プログラム、ワードプロセッサ、およびプレゼンテーション ツールを備えています。会計部ではスプレッドシートとプレゼンテーション プログラムだけを必要とし、書類作成部ではワードプロセッサとスプレッドシートだけを必要とするような場合があります。その他の部署にはアプリケーションの全スイートが必要となるかもしれません。

その場合各社員ごとに手作業でセットアップしなくても、ユーザーはスイート全体のオリジナルインストールを使用して各部署の必要に応じてカスタマイズし、その後 2 つのパッケージ間のトランスフォームを作成することができます。トランスフォームは、個別の製品構成についてそれぞれ作成する必要があります。

トランスフォームの作成

InstallShield では、製品の複数の構成を容易に作成できるようにウィザードがその手順をガイドします。以下の 2 つの方法のいずれかを利用してトランスフォームを作成することができます。

2 つの .msi パッケージを比較してトランスフォームを作成する

トランスフォームを作成する一つ目の手法では、ベース .msi パッケージはすべてのカスタマイズの起点としての役割を果たし、ターゲットパッケージはインストールに反映させたいすべての変更を含みます。一度ベース .msi パッケージ とターゲット .msi パッケージが準備されると、トランスフォームウイザードを使用してトランスフォームを作成することができます。ウィザードを終了すると InstallShield は 2 つのパッケージを比較し、パッケージ間の差分を含むトランスフォーム (.mst ファイル)を作成します。

単一の .msi パッケージを基にトランスフォームを作成する

別のトランスフォームの作成方法として、新規トランスフォームプロジェクトの作成があります。新規トランスフォーム プロジェクトを作成すると、トランスフォーム オープン ウィザード が起動します。トランスフォーム オープン ウィザードを使用して、ダイレクト MST モードで既存の .msi パッケージを開き必要に応じてプロジェクトを編集することができます。InstallShield は、 ベース .msi パッケージとダイレクト MST モードで加えた変更との差分を評価し、その差分を含むトランスフォーム プロジェクト (.mst ファイル) を作成します。

この手法でトランスフォームを作成すると、トランスフォーム プロジェクトに組み込む追加トランスフォーム プロジェクトを指定することができます。また、エンドユーザーへのデフォルトのユーザー インターフェイス応答も指定することができます。

トランスフォームの適用

トランスフォームの作成後は、アプリケーションがインストールされているコンピューター別に実行時にトランスフォームを適用できます。たとえば、ターゲット コンピューターが経理部門のコンピューターかどうかを検査することができます。それが経理部門にある場合、経理部用のトランスフォームが元のインストールに適用され、スプレッドシートとプレゼンテーション ツールのみがインストールされます。

2 つの .msi パッケージを比較してトランスフォームを作成する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

メモ: *トランスフォーム プロジェクトが新規ファイル（オリジナル .msi ファイルには含まれないファイル）を含む場合、これらのファイルはインストール実行中には見つかりません。この問題を避けるため、トランスフォーム プロジェクトをベース .msi パッケージと同じ場所に配置するか、または手動でソース ファイルをベース .msi パッケージの場所へコピーします。*

タスク ***2 つの .msi パッケージを比較してトランスフォームを作成するには、以下の手順を実行してください。***

1. InstallShield で、ベースの .msi パッケージを開きます。
2. プロジェクトを新しい名前で保存します。この新しいプロジェクトがターゲット プロジェクトとなります。詳細については、「新しい名前と場所でプロジェクトを保存する」を参照してください。
3. インストールに反映させたい変更をすべて加え、リリースをビルドします。
4. **[ツール]** メニューで、**[トランスフォームを作成/適用]** をクリックします。トランスフォームウィザードが開きます。
5. トランスフォーム ウィザードのパネルを完成して、トランスフォームを作成します。

単一の .msi パッケージを基にトランスフォームを作成する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

メモ: *トランスフォーム プロジェクトが新規ファイル（オリジナル .msi ファイルには含まれないファイル）を含む場合、これらのファイルはインストール実行中には見つかりません。この問題を避けるため、トランスフォーム プロジェクトをベース .msi パッケージと同じ場所に配置するか、または手動でソース ファイルをベース .msi パッケージの場所へコピーします。*

タスク ***単一の .msi パッケージを基にトランスフォームを作成するには、以下の手順を実行してください。***

1. **[ファイル]** メニューで、**[新規]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[Windows Installer]** タブで **[トランスフォーム]** をクリックします。
3. **[プロジェクト名]** ボックスで、トランスフォーム プロジェクトの名前を入力します。

4. **[場所]** ボックスに、トランスフォーム プロジェクトが保存される場所へのパスを入力する、または **[参照]** をクリックして場所を探します。
5. [OK] をクリックします。**トランスフォーム オープン ウィザード**が開きます。
6. **トランスフォーム オープン ウィザード**のパネルを完成してトランスフォームを作成します。

トランスフォームの適用

***プロジェクト・** この情報は、トランスフォーム プロジェクトに適用します。*

トランスフォームを適用すると、ベース .msi パッケージを変更して、トランスフォーム (.mst ファイル) に含めた変更を組み込むことになります。

トランスフォームを適用する方法として 2 つの方法が以下に記述されています。

コマンドラインからのトランスフォームの適用

1 つの方法としては、コマンドラインを Msiexec.exe または Setup.exe に渡す方法があります。コマンドラインからトランスフォームを適用すると、.msi パッケージのデータベースは、実行時に変更されます。トランスフォームを実行時に適用するには、次のコマンドライン ステートメントのように、TRANSFORMS プロパティで目的のトランスフォームを指定します。

```
msiexec /i "ProductName.msi" TRANSFORMS="YourTransform.mst"
```

複数のトランスフォームはセミコロンで区切ります。

***注意 ・** Windows Installer はセミコロンをファイル名の区切り文字として解釈するため、トランスフォーム名にセミコロンを使用しないでください。*

コマンドラインで 2 つ以上のトランスフォームを指定した場合、トランスフォームは指定した順番で適用されます。たとえば、ベース パッケージに機能が 1 つだけあって、2 つ目の機能を追加する 1 つのトランスフォームと、2 つ目の機能を変更するもう 1 つのトランスフォームがあったとします。機能を追加するトランスフォームは、2 つ目の機能を変更するトランスフォームの前に指定しなければなりません。その順番で指定しない場合、変更は正しく行われません。

トランスフォームの適用にトランスフォーム ウィザードを利用する

トランスフォーム ウィザード を使ってトランスフォームを適用する際、トランスフォーム ファイルに含まれる変更すべてを反映したインストール パッケージ (.msi パッケージ) を作成します。これは、あらゆるユーザーのアプリケーションをカスタマイズしているネットワーク管理者にとって便利な方法です。

タスク ***トランスフォーム ウィザードを起動するには、次を実行します。***

[ツール] メニューで、**[作成/適用]** をクリックします。

トランスフォームの編集

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

InstallShield を使用すると、.msi パッケージを、InstallShield (.ism) プロジェクトに変換することなく編集することができます。変更点はトランスフォーム (.mst) ファイルとして保存できます。

編集するため既存するトランスフォームを開く

InstallShield で、既存の .mst ファイルを参照することができます。.mst ファイルを開く前に、InstallShield はこのファイルを適用する ベース .msi を必要とします。また、編集するために .mst ファイルを開く前にベース .msi ファイルに適用するその他の .mst ファイルの名前も必要です。

InstallShield で、.mst ファイルを初めて開いたとき、トランスフォーム オープン ウィザードが起動します。このウィザードでは、 .mst ファイル、ベース .msi ファイル、および編集の前に適用する追加トランスフォーム指定することができます。このウィザードの最後のパネル、[応答トランスフォームの作成] パネルでは、そのトランスフォームが応答トランスフォームであるかどうかを指定できます。応答トランスフォームには、インストールのユーザー インターフェイス部分のデフォルトの応答も含まれています。応答トランスフォーム作成を選択すると、インストールのユーザー インターフェイス要素が実行され、インストール ウィザードの各パネルのオプションをカスタマイズすることができます。

トランスフォーム オープン ウィザードが完了したとき、トランスフォームが適用され、.mst ファイルが InstallShield のダイレクト MST モード (ダイレクト編集モードに似ています) で開きます。ウィザードで指定されたデータは、次回 .mst ファイルを開いたとき、トランスフォーム プロジェクトが InstallShield で直接開くように保存されます。

タスク ***既存のトランスフォームを開くには、以下の手順を実行してください。***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログが開きます。
2. 必要な .mst ファイルを選択します。
3. **[開く]** をクリックします。**トランスフォーム オープン ウィザード**が開きます。

既存のトランスフォーム ファイルを InstallShield トランスフォーム ウィザードで開く

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

InstallShield トランスフォーム ウィザードで、既存のトランスフォーム ファイルを開くことができます。トランスフォーム ウィザードで初めて開いたときと同様に、トランスフォーム ファイルのベースとなる MSI パッケージを選択するためのプロンプトが表示されます。これによって、複数の MSI パッケージに共通の汎用トランスフォーム ファイルを使用することができます。

InstallShield トランスフォーム ウィザードで既存のトランスフォーム ファイルを開くには、Windows エクスプローラーでトランスフォーム ファイルを右クリックしてから、コンテキスト メニューから **[InstallShield トランスフォーム ウィザードで開く]** を選択します。

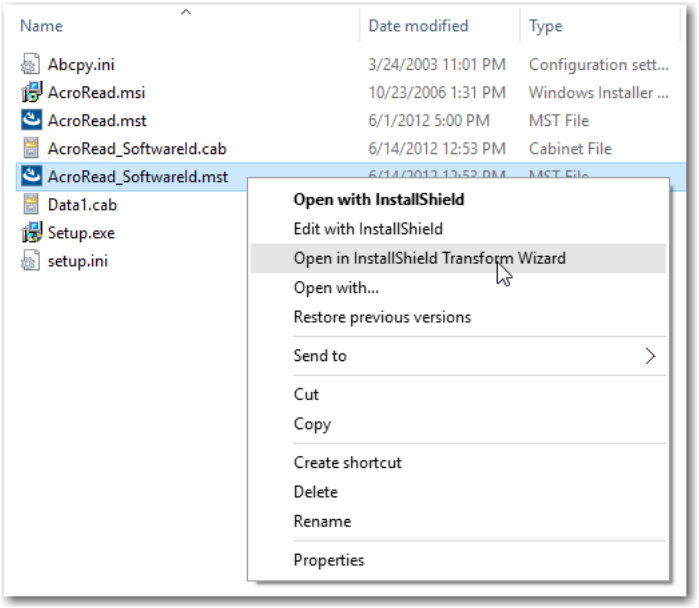


図 1: [InstallShield トランスフォーム ウィザードで開く] オプション

.msi ファイルをトランスフォームとして保存する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

ダイレクト編集モードで .msi ファイルを編集中、そのファイルを .mst ファイルとして保存することができます。

タスク ***.msi ファイルを .mst ファイルとして保存するには、以下の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. **[ファイルの種類]** ボックスで **[Windows Installer トランスフォーム (*.mst)]** を選択します。

最後に保存してからそれ以降に変更された部分を指定された .mst ファイルに保存します。

トランスフォームを .msi ファイルとして保存する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

.mst ファイルを .msi ファイルとして保存することもできます。

タスク ***トランスフォームを .msi ファイルとして保存するには、以下の手順を実行します。***

1. まだトランスフォーム プロジェクトを開いていない場合、InstallShield で開きます。
2. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
3. **[ファイルの種類]** ボックスで **[Windows Installer パッケージ (*.msi)]** を選択します。

.msi ファイルが作成されます。ベース .msi ファイルのデータすべて、編集した .mst ファイル、およびその他適用された .mst ファイルが含まれます。プロジェクトは ダイレクト モード MSI プロジェクトとして取り扱われます。詳細については、「ダイレクト編集モードで .msi/.msm データベースを編集する」を参照してください。

応答トランスフォーム

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

応答トランスフォームは、管理者がユーザー インターフェイス シーケンスを実行して希望の値を入力するだけでインストールのデフォルト値を設定することができる種類のトランスフォームです。たとえば、インストールされる機能、アプリケーションのインストール場所、および会社名のデフォルト値等を含んだ応答トランスフォームを作成することができます指定するデフォルト値はトランスフォームの開始位置として利用されます。

応答トランスフォームを作成するには、トランスフォーム オープン ウィザードを利用します。

応答トランスフォームの作成

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

タスク ***応答トランスフォームを作成するには、以下の手順に従います:***

1. トランスフォーム オープン ウィザードを開きます。
2. 必要に応じてウィザードの各パネルを完成します。[応答トランスフォームの作成] パネルで **[応答トランスフォームの作成]** を選択します。
3. **[コマンドラインのプロパティ (オプション)]** ボックスで、応答トランスフォームに渡すコマンドラインのプロパティ (セミコロンで分かれた一対のプロパティ名/値) を指定します。
4. **[完了]** をクリックします。インストールのユーザー インターフェイス要素が実行されます。
5. 必要に応じてインストールウィザードの各パネルで利用可能なオプションをカスタマイズして、シミュレート インストールを完成します。ファイルは転送されず、コンピューターに変更を加えることもありません。
6. インストールを最後まで進み、**[インストール]** をクリックします。

シミュレート インストールは終了し、InstallShield はトランスフォーム プロジェクトの [プロパティ] テーブルの中の値としてシミュレート インストールの間に加えられたすべての変更を保存します。これらの値はインストールでデフォルト値として使用されます。

応答トランスフォームの変更

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

タスク ***既存のトランスフォームプロジェクト内の個別応答を変更するには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. 応答を変更するトランスフォーム プロジェクト (.mst ファイル) を選択します。
3. **[開く]** をクリックします。InstallShield はダイレクト MST モードでトランスフォーム プロジェクトを開きます。
4. **ダイレクト エディター**で、**Property** テーブルをクリックします。
5. 必要に応じて任意の応答プロパティの値を変更します。COMPANYNAME と USERNAME はよく変更されるプロパティの代表例です。

また、別の方法として、トランスフォームの作成時と同様の手順を実行することもできます。これらの手順に従うと、インストールのユーザー インターフェイスを起動し、適切な応答を選択することができます。

応答トランスフォームのテスト

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

応答に加えた変更を確認するために、次のいずれかを実行してトランスフォームのユーザー インターフェイス部分をテストすることができます。

- **[ビルド]** メニューで **[テスト]** をクリックします。
- **[テスト]** ボタンをクリックします。
- CTRL+T を押します。

サーバーの場所の構成

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

エンド ユーザーが製品をネットワーク サーバーまたは Web サイトからインストールする場合で、機能をそのサーバーまたは Web サイトから実行できるようにするには、初回インストールの後、製品によるサーバーまたは Web サイト上の .msi パッケージおよび関連ファイルへのアクセスが必要になる場合があります。また、インストールが問題のあるファイルを自動的にサーバーまたは Web サイトからコピーすることができるため、一部のファイルが削除されているまたは破損している場合も、アクセスが必要になることがあります。

InstallShield では、トランスフォームに製品のインストール パッケージへのネットワークまたは URL ソース パスの一覧を指定することができます。このリストにあるパスは SOURCELIST プロパティに格納されています。実行時に Windows Installer は、エンド ユーザーの既存の製品のソース リストの最後に、このリストを追加します。

SOURCELIST プロパティ (Windows Installer ヘルプ ライブラリ)

サーバーの場所の追加

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

タスク ***サーバーの場所としてネットワーク サーバー パスまたは Web サイト URL を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. 必要に応じて、サーバーの場所を指定します。
4. **[OK]** をクリックします。

InstallShield が、"サーバーの場所" 設定の値を指定された場所で更新します。

サーバーの場所の変更

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

タスク ***サーバー リストにある場所を変更するには、以下の手順を実行します:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. 必要に応じて、サーバーの場所を変更します。
4. **[OK]** をクリックします。

InstallShield が、必要に応じて "サーバーの場所" 設定の値を更新します。

サーバーの場所の削除

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

タスク ***サーバーの場所のパスを削除するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. **[ソース パス]** リストで、削除するパスを選択します。
4. **[削除]** ボタンをクリックします。
5. **[OK]** をクリックします。

サーバーの場所の順番を変更する

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

インストールがネットワーク サーバーまたは Web サイト上に格納されているソース パスへのアクセスを必要とする場合、サーバーの場所一覧で最初にアクセス可能なパスを使用します。InstallShield では、必要に応じてパスの順序を並べ替えることができます。

タスク ***サーバーの場所の順番を変更するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"サーバーの場所"** 設定で、省略記号ボタン (...) をクリックします。**[サーバーの場所]** ダイアログ ボックスが開きます。
3. **[ソース パス]** リストで、移動するパスを選択します。
4. **[上へ移動]** と **[下へ移動]** ボタンをクリックして、必要に応じて順番を変更します。
5. **[OK]** をクリックします。

5

アドバンスト UI およびスイート/アドバンスト UI インストールの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールは、インストールと InstallShield 前提条件を合わせてパッケージ化し、カスタマイズ可能な統合ユーザー インターフェイスを持つ単一インストールを作成するブートストラップ アプリケーションです。これらのインストールでは、条件付きで、必要に応じて、ターゲット システムでパッケージを起動するセットアップ ランチャー (Setup.exe) を使用します。

以下は、スイート/アドバンスト UI インストール内の各パーツを示したダイアグラムです。このインストールでは、複数のプライマリ パッケージ (.exe パッケージ、サイドローディング Windows アプリ パッケージ (.appx)、および Windows Installer トランザクションを含む) および複数の InstallShield 前提条件パッケージを実行するサポートが提供されています。アドバンスト UI インストールのパーツは、アドバンスト UI インストールでは、プライマリ パッケージは 1 つのみ実行できるけれども、InstallShield 前提条件パッケージは複数実行できるという点を除き、類似しています。

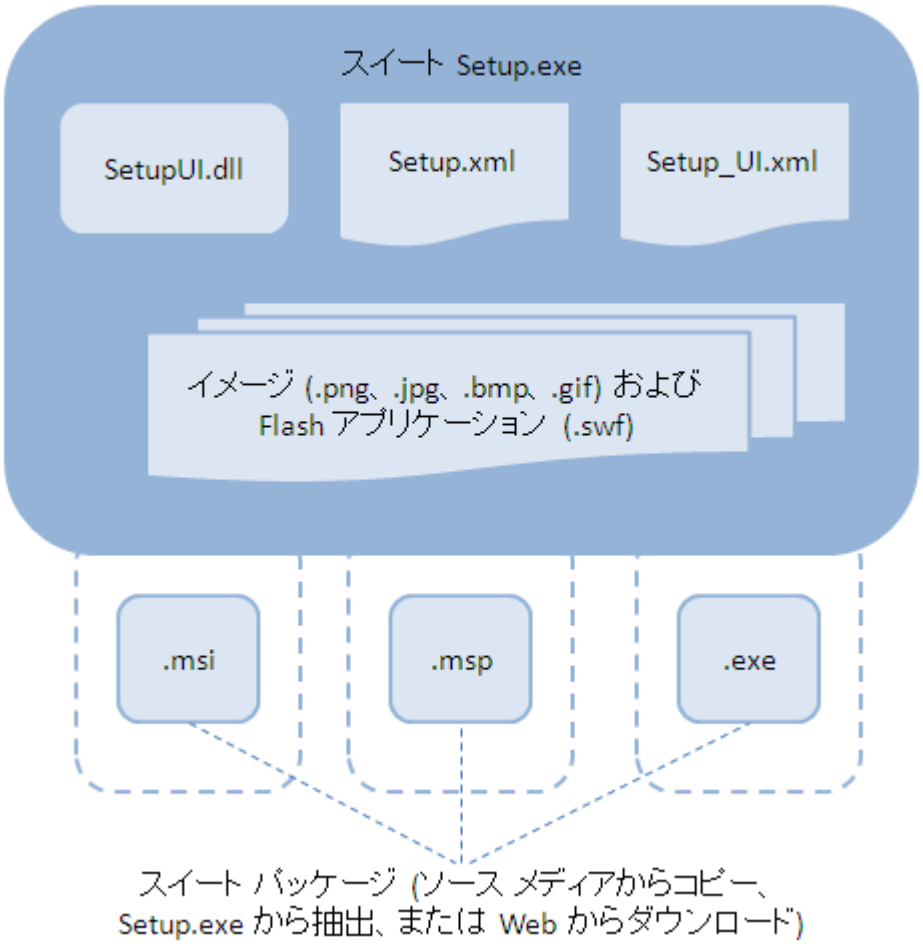


図 1: スイート/アドバンスト UI インストールのパーツ

ビルド時に InstallShield は .issuite ファイル (アドバンスト UI プロジェクトの xml プロジェクト ファイル、または、スイート/アドバンスト UI プロジェクトの XML プロジェクト ファイル) の様々なセクションを使用して、Setup.xml および Setup_UI.xml を作成し、これらを、アドバンスト UI またはスイート/アドバンスト UI パッケージとアドバンスト UI または スイート/アドバンスト UI インストールの実行に必要なその他のファイルと共にバンドルします。作成されたアドバンスト UI またはスイート/アドバンスト UI インストールには、次のファイルが含まれています:

- Setup.exe – アドバンスト UI またはスイート/アドバンスト UI エンジンが含まれています。
- Setup.xml – アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージ、パッケージの起動条件、およびアドバンスト UI またはスイート/アドバンスト UI インストールのウィザード ユーザー インターフェイスで使用される文字列エントリが記述されています。
- Setup_UI.xml – ウィザード インターフェイスのスタイル設定、レイアウト情報、その他の UI パラメーターを定義します。

- SetupUI.dll – アドバンスト UI およびスイート/アドバンスト UI インストールに使用されるアドバンスト UI およびスイート/アドバンスト UI ライブラリで構成されています。

InstallShield はまた、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを集めて、適切な場合はパッケージと関連する非圧縮ファイルのフォルダー構造を作成します。また、InstallShield には、アドバンスト UI またはスイート/アドバンスト UI インストールのサポート ファイルも組み込まれています。これらのサポート ファイルは、ウィザード ページで表示される使用許諾契約や画像ファイルです。

エンド ユーザーがアドバンスト UI またはスイート/アドバンスト UI インストールを起動すると、アドバンスト UI またはスイート/アドバンスト UI エンジンは Setup.xml ファイルを読み込み、SetupUI.dll ファイルをロードして、スイートならびにそのパッケージに定義された条件を評価し、Setup.xml ファイルの指示に従い、適切な場合は、ターゲット システムで必要なパッケージとファイルをダウンロードし、実行する必要があるパッケージを実行して、リソースをクリーンアップします。

アドバンスト UI またはスイート/アドバンスト UI インストールには、ターゲット システム上で Windows Installer 3.1 以降が必要になりますのでご注意ください。また、Windows XP 以降または Windows Server 2003 以降も必要です。

アドバンスト UI またはスイート/アドバンスト UI タイプのプロジェクトに関する詳しい情報は、ドキュメントのこのセクションを参照してください。

アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、インストールのブートストラップ アプリケーションを開発したり、完全にカスタマイズ可能なユーザー インターフェイスを作成することができます。次のテーブルは、これら 2 つのプロジェクト タイプの異なる機能と特徴の比較です。

テーブル 1 ・ アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
InstallShield の Edition Premier および Professional Edition の違いついては、「InstallShield の他のエディションからアップグレードする」をご覧ください。	InstallShield Professional	InstallShield Premier

テーブル 1 • アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点 (続き.)

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
プロジェクト ファイルの拡張子	.issuite	.issuite
サポートされているプライマリ パッケージのファイル形式 適切なパッケージのファイル形式の選択については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」をご覧ください。	.msi パッケージ .msp パッチ InstallScript パッケージ	.msi パッケージ .msp パッチ InstallScript パッケージ .exe パッケージ .appx パッケージ Web 配置パッケージ (.zip) Windows Installer トランザクション スイート/アドバンスト UI インストールに、基本の MSI および InstallScript プロジェクト (.ism) をパッケージとして含めることもできます。ビルド時に InstallShield は、InstallShield プロジェクト内で指定されたリリースを最初にビルドして、生成するスイート/アドバンスト UI インストールにそれらをパッケージとして含めます。 サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。
プロジェクトに含めることができるプライマリ パッケージの数 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。	1 この機能を利用すると、単一のパッケージに対して、最新のカスタマイズ可能なユーザー インターフェイスを作成することができます。	1 つ以上 この機能を利用すると、単一のパッケージまたは複数のパッケージに対して、最新のカスタマイズ可能なユーザー インターフェイスを作成することができます。 また、この機能を使うと、複数の個別のインストールを、統合されたユーザー インターフェイスを使用する単一のインストールにパッケージ化することもできます。

テーブル 1 • アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点 (続き.)

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
InstallShield 前提条件をプロジェクトに含めることができるサポート 詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。	必要に応じて、InstallShield 前提条件を、.msi パッケージ、.msi パッケージ、.msp パッケージ、および .exe パッケージとして (前提条件に対して実行されるように設定されたファイルの種類に応じます)、数に制限なく含めることができます。	必要に応じて、InstallShield 前提条件を、.msi パッケージ、.msi パッケージ、.msp パッケージ、および .exe パッケージとして (前提条件に対して実行されるように設定されたファイルの種類に応じます)、数に制限なく含めることができます。
プライマリ パッケージとセカンダリ パッケージを識別できるサポート 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。	制限付きで柔軟性あり。1 つの .msi、.msp、または InstallScript パッケージをプライマリ パッケージとしてプロジェクトに追加できます。 プロジェクトに含める InstallShield 前提条件はすべて、依存パッケージとして追加されます。これは変更不可能です。	完全に柔軟性あり。任意のサポートされている種類のパッケージをプロジェクトに追加して、それらがプライマリ パッケージまたは依存パッケージであるかを容易に指定できます。 プロジェクトに含めた InstallShield 前提条件はすべて、依存パッケージとして追加されますが、それを上書きして、プライマリ パッケージとしてマークすることも可能です。
アクションを作成して、適切なランタイム イベント中にそれらの実行をスケジュールできる機能のサポート 詳細については、「スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する」を参照してください。	利用できません。	スイート/アドバンスト UI プロジェクトには、[イベント] ビューが含まれており、実行時にターゲット システム上で発生するアクションを定義することができます。[イベント] ビューで特定のイベント中に発生させるアクションをスケジュールするか、[パッケージ] ビューでパッケージに割り当てることができます。さらに、ウィザード ページまたは 2 番目のウィンドウを開くまたは閉じるとき、またはエンド ユーザーがコントロールを使用したとき (たとえば、ボタンをクリックしたとき) に発生させるアクションをスケジュールすることもできます。

テーブル 1 • アドバンスト UI およびスイート/アドバンスト UI プロジェクトの相違点 (続き.)

比較カテゴリ	アドバンスト UI プロジェクト	スイート/アドバンスト UI プロジェクト
ビルド イベントのサポート 詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。	利用できません。	スイート/アドバンスト UI プロジェクトでは、InstallShield でリリースのビルドが開始する前、および、InstallShield がリリースをビルドして署名をした後で実行するコマンドをスケジュールすることができるリリース設定が提供されています。
ランタイム言語サポート 詳細については、「InstallShield 実行時の言語サポート」を参照してください。	アドバンスト UI インストールのエンドユーザー インターフェイスは、InstallShield をインストールしたときに選択した 1 言語に限られます。	エンドユーザーのテキストを複数の言語で表示する単一インストールを作成できます。翻訳済みの文字列を利用して、ウィザード ページおよびメッセージを 34 の追加言語の 1 つに変更することができます。
ビルド時に InstallShield が初期化する仮想マシンへインストールを配布できる機能 詳細については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。	利用できません。	インストールのビルドが成功するたびに InstallShield が仮想マシン (VM) を指定のスナップショットに戻し、VM の電源をオンにしてスイート/アドバンスト UI インストールを VM にコピーし、テストが可能な状態にします。この機能を使って手作業で行う処理を減らし、テストにかかる時間を節約できます。

アドバンスト UI またはスイート/アドバンスト UI インストールで機能、パッケージ、前提条件、およびファイルを編成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールはブートストラップ アプリケーションで、通常、その主要な目的はターゲット システム上で必要に応じて条件付きでパッケージを起動することです。アドバンスト UI またはスイート/アドバンスト UI インストールでは、製品が適切に機能するようにサードパーティまたはカスタム再配布可能ファイルをインストールする InstallShield 前提条件を実行する必要がある場合もあります。

機能は、エンド ユーザーから見て、アドバンスト UI またはスイート/アドバンスト UI インストールの個別にインストール可能な最小構成単位です。各パッケージと、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれる InstallShield 前提条件は、プロジェクト内の機能と関連付けなくてはなりません。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるパッケージと InstallShield 前提条件の追加および構成については、ドキュメントのこのセクションを確認してください。このセクションには、次のトピックが含まれています:

- パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン
- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮
- スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する
- InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮
- スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する
- 実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する
- Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する
- サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する
- サイドローディング Windows アプリ バンドル (.appxbundle | .msixbundle) をスイート/アドバンスト UI プロジェクトに追加する
- Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い
- InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、パッケージとトランザクションのインストール順序を指定する

- アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける
- 異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する
- アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージ操作を構成する
- アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する

パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

推奨されるファイル フォーマット

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージ用に使用するファイル形式は、ブートストラップ アプリケーションまたはセットアップ ランチャー (Setup.exe) を含まないファイル形式をお勧めします。アドバンスト UI またはスイート/アドバンスト UI プロジェクトには、次のパッケージの種類が推奨されています:

- .msi パッケージ
- .msp パッチ
- InstallScript パッケージ パッケージは、InstallShield 2012 Spring 以降でビルドされた非圧縮の InstallScript インストールでなければなりません。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおける、これらの種類のパッケージの使用については、次」を参照してください:

- .msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する
- InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

スイート/アドバンスト UI インストールには、IIS サーバーおよびクラウドに Web 配置パッケージ (.zip) を展開できるサポートが含まれています。詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

さらに、InstallShield にはサイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加するためのサポートが含まれています。アプリのサイドローディングとは、Windows Store を介さずにアプリケーションをインストールするプロセスです。詳細については、「サイドローディング Windows アプリ パッケージ (.appx| msix) をスイート/アドバンスト UI プロジェクトに追加する」を参照してください。Windows アプリ パッケージは限られた状況下でのみサポートされているため、通常これらのパッケージは .msi パッケージなどのより一般的なインストール形式と共に使用されます。

InstallShield では、実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加できます。以下に例を示します:

- Windows Installer ベースのインストール用に InstallShield でビルドされたセットアップ ランチャー 実行可能ファイル
- Windows Installer ベースのパッチ用に InstallShield でビルドされたセットアップ ランチャー 実行可能ファイル
- InstallScript インストール用のセットアップ ランチャー ファイル (これには、InstallShield 2012 以前でビルドされた InstallScript インストールおよび圧縮済み InstallScript インストールが含まれます。)
- InstallShield 以外のツールでビルドされたセットアップ ランチャー実行可能ファイル

.msi、.msp、および InstallScript パッケージは、さまざまな理由で、セットアップ ランチャー実行可能ファイルよりも利点があります:

- たとえば、アドバンスト UI およびスイート/アドバンスト UI セットアップ ランチャーのウィザード インターフェイスでは、ターゲット システム上で実行中、推奨されるファイル フォーマットの進行状況を段階的に表示できますが、実行可能パッケージの進行状況は表示できません。
- パッケージとして推奨されるファイル フォーマットの 1 つをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加したとき、アドバンスト UI またはスイート/アドバンスト UI インストールで、以前のバージョンのパッケージが新しいバージョンを上書きするのを防ぐための適切な対象条件が自動的に追加されます。実行可能パッケージをスイート/アドバンスト UI プロジェクトに追加する場合、このようなダウングレードを防ぐ対象条件を手入力で定義する必要があります。
- パッケージとして推奨されるファイル フォーマットの 1 つをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する場合、そのパッケージに対して、ターゲット システムでパッケージが既にインストールされているかどうかを評価する判別条件を定義する必要はありません。これは、このようなシナリオは、アドバンスト UI またはスイート/アドバンスト UI インストールでブロックされるからです。実行可能パッケージをスイート/アドバンスト UI プロジェクトに追加する場合、この種類の判別条件は、パッケージに対して手入力で定義する必要があります。
- アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーでは、アドバンスト UI またはスイート/アドバンスト UI インストールのウィザード インターフェイスが優先されるため、推奨されるファイル フォーマットのユーザー インターフェイスは自動的に抑制されます。プロジェクトに実行可能パッケージが含まれている場合、そのユーザー インターフェイスを手動で抑制し、アドバンスト UI またはスイート/アドバンスト UI インストールで、そのユーザー インターフェイスをサイレントで起動する必要があります。そうしなかった場合、インストールの実行時に、アドバンスト UI またはスイート/アドバンスト UI のユーザー インターフェイスと .exe パッケージのユーザー インターフェイスの 2 つの異なるユーザー インターフェイスが表示されることになります。.msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

詳細については、「実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

メディアの種類の要件

アドバンスト UI またはスイート/アドバンスト UI インストールでは、複数のディスク (CD や DVD など) に分割されたパッケージをインストールできません。また、ネットワークなどの固定ディスクからの複数ディスク パッケージもインストールすることができません。従って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるパッケージ (.msi パッケージ、.msp パッケージ、InstallScript パッケージ、.exe パッケージ、Web 配置パッケージ、または Windows アプリ パッケージ(.appx)) はどれも、単一ディスク リムーバブル メディア タイプ、ネットワーク イメージ タイプ、または Web タイプである必要があります。

.msi パッケージ、.msp パッチ、トランザクションをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、次の種類の Windows Installer ベースのインストールを、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに追加することができます:

- Windows Installer パッケージ (.msi パッケージ)
- Windows Installer 修正プログラム (パッチ) (.msp パッケージ)

InstallShield ではまた、Windows Installer トランザクションをスイート/アドバンスト UI プロジェクトに含めることもできます (アドバンスト UI プロジェクトには追加できません)。

トランザクションは、Windows Installer 4.5 以降の機能を使った、ターゲット システム上でトランザクション処理を使って連鎖させる Windows Installer パッケージ (.msi) および Windows Installer パッチ (.msp) で構成されます。パッケージを連鎖させることによって、単一のトランザクションとして処理します。各スイート/アドバンスト UI インストールには、複数の個別のトランザクションを含めることができます。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は現在のトランザクションに含まれる連鎖パッケージ全てのロールバックを開始して、システムを以前の状態に復元します。

トランザクション処理なしに実行する .msi または .msp パッケージを追加する

タスク

トランザクション処理なしで実行する .msi または .msp パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックして、適切なコマンド (**新しい Windows Installer パッケージ (.msi)** または **新しいパッチ パッケージ (.msp)**) をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. アドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加したいパッケージ ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

トランザクション処理を使って実行するパッケージを追加する

タスク

トランザクション処理を使って実行するパッケージをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しいトランザクション]** をクリックします。**[パッケージ]** エクスプローラーに、**[トランザクション]** フォルダーが追加されます。
3. **[パッケージ]** エクスプローラーで、**トランザクション** フォルダーを右クリックして、適切なコマンド (**新しい Windows Installer パッケージ (.msi)** または **新しいパッチ パッケージ (.msp)**) をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
4. スイート/アドバンスト UI プロジェクトに追加したいパッケージ ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
5. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

重要: *スイート/アドバンスト UI インストールからトランザクション内のパッケージを使用するには、Windows Installer 4.5 以降が必要になります。トランザクション内のパッケージが、以前のバージョンの Windows Installer があるターゲット システム上のスイート/アドバンスト UI インストールによって起動された場合、スイート/アドバンスト UI インストールは失敗します。従って、スイート/アドバンスト UI インストールに含まれるパッケージにトランザクション処理を使用する場合、以下のタスクから 1 つ以上を実行することが推奨されます:*

- *以前のバージョンがあるターゲット システム用のスイート/アドバンスト UI プロジェクトに Windows Installer 4.5 再配布可能ファイルをパッケージとして含めて、トランザクション処理を使用するパッケージの前にスケジュールする。*
- *Windows Installer 4.5 以降が搭載されていないシステム上でスイート/アドバンスト UI インストールが実行することを防ぐ終了条件を作成する。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する」を参照してください。*
- *トランザクションに含まれる各パッケージに Windows Installer 4.5 以降の存在を確認する対象条件を作成して、トランザクションに含まれるこれらのパッケージが起動しないように防ぐ。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。*

InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。スイート/アドバンスト UI とアドバンスト UI プロジェクト タイプの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。*

InstallScript インストールを、実行可能パッケージとしてではなく InstallScript パッケージとしてアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するための要件

InstallShield では、InstallScript パッケージが次の要件を満たした場合にのみ、InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加することができます。

- InstallScript パッケージは非圧縮です。
- InstallShield 2012 Spring 以降が、InstallScript パッケージまたはアドバンスト UI またはスイート/アドバンスト UI インストールを作成するために使われている。
- InstallScript パッケージで、イベント ベースのスクリプトが使用される。program...endprogram スタイルのスクリプトは使用しない。program...endprogram ブロックを使用するスクリプトをイベント ベースのスクリプトに変換する必要がある場合、「OnShowUI」を参照できます。

これらの条件のいずれかが満たされなかった場合、InstallScript インストールを、InstallScript パッケージとしてではなく、実行可能ファイル パッケージ (.exe) として、スイート/アドバンスト UI プロジェクトに追加することができます。ただし、InstallScript インストールを、実行可能パッケージ (.exe) としてではなく、InstallScript パッケージとして含める方が優先されます。詳細については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」を参照してください。

メモ: *InstallShield 2012 以前の InstallScript インストールをパッケージとしてアドバンスト UI またはスイート/アドバンスト UI インストールに含める場合、アドバンスト UI またはスイート/アドバンスト UI インストールで、InstallScript パッケージが起動されたとき、ランタイム エラーが発生します。*

InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときの手順

タスク

InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[InstallScript パッケージ (.hdr)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. InstallScript インストールの data1.hdr ファイルを選択して、**[開く]** をクリックします。data1.hdr ファイルは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージとして含める非圧縮 InstallScript インストールの残りのファイルと同じフォルダーに格納します。

ヒント: *InstallScript インストールの* data1.hdr *ファイルを選択すると、InstallScript パッケージが起動されたときに必要な .cab ファイル、InstallScript ファイル (.rul)、および、その他のファイルがアドバンスト UI またはスイート/アドバンスト UI インストールに含められます。*

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

ヒント: *アドバンスト UI およびスイート/アドバンスト UI プロジェクトに InstallScript パッケージを含める手順についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。スイート/アドバンスト UI とアドバンスト UI プロジェクト タイプの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに InstallScript パッケージを含める場合、または スイート/アドバンスト UI プロジェクトに InstallScript プロジェクト (.ism) を含める場合、以下の詳細に注意が必要です。

ランタイム ユーザー インターフェイス、イベント、および関数

アドバンスト UI またはスイート/アドバンスト UI インストールで InstallScript パッケージが起動されたとき、それ自身のユーザー インターフェイス (UI) が表示され、InstallScript パッケージの UI は自動的に抑制されます。アドバンスト UI またはスイート/アドバンスト UI インストールでは、InstallScript パッケージに対して、進行状況も表示されます。これらの変更を可能にするために、アドバンスト UI またはスイート/アドバンスト UI インストールでは、デフォルトで、いくつかのアドバンスト UI およびスイート/アドバンスト UI 固有の InstallScript イベントおよび関数が使用され、一部の標準 InstallScript イベントおよび関数は無視されます。アドバンスト UI およびスイート/アドバンスト UI 固有の InstallScript イベントによって、ファイルの転送がどう開始されるかも補強されています。

InstallScript Setup.exe ファイルから起動される (スイート/アドバンスト UI インストールから起動されない) 標準の InstallScript インストールでは、ほとんどのイベントは OnShowUI イベントから直接呼ばれます。InstallScript パッケージを起動するアドバンスト UI またはスイート/アドバンスト UI インストールでは、OnShowUI が OnSuiteShowUI によって置き換えられます。OnSuiteShowUI を使ってデフォルトで起動されるアドバンスト UI およびスイート/アドバンスト UI 固有のイベントなどに関する詳しい情報は、「OnSuiteShowUI」をご覧ください。

InstallScript 言語には、InstallScript パッケージとそれを実行するアドバンスト UI またはスイート/アドバンスト UI インストールの間の対話を可能にするアドバンスト UI およびスイート/アドバンスト UI 固有の関数がいくつか含まれています。詳しくは、「アドバンスト UI およびスイート/アドバンスト UI の対話関数」を参照してください。

次の事項に注意してください。

- アドバンスト UI またはスイート/アドバンスト UI インストールでは、InstallScript パッケージの標準 InstallScript UI が抑制されるため、すべてのスクリプト ダイアログ (例、**DefineDialog**、**EzDefineDialog**) で、エラーが返されます。従って、ダイアログの呼び出しは、アドバンスト UI およびスイート/アドバンスト UI 固有のイベントまたは機能の転送イベントで行う必要があります。
- アドバンスト UI およびスイート/アドバンスト UI インストールでは、すべてのスクリプトから提供されたメッセージ ボックス関数 (例、**MessageBox**、**SprintfBox**) は SuiteErrorReport 関数を介して送られます。これらの関数への呼び出しは、エラー ケース以外、ユーザー入力を要求する方法として使用することはできません。すべてのユーザー入力は、アドバンスト UI またはスイート/アドバンスト UI のウィザード ページから取得する必要があります。この設定は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [ウィザード インターフェイス] ビューで構成することができます。この構成により、アドバンスト UI またはスイート/アドバンスト UI インストール内のすべてのパッケージで、一貫した UI 使用感を提供することができます。
- 非 InstallScript UI (DLL または 他の方法によって実装されたカスタム UI など) は、アドバンスト UI およびスイート/アドバンスト UI 固有の InstallScript イベントまたは機能転送イベントから呼び出すことはできません。すべてのユーザー入力は、アドバンスト UI インストールまたはスイート/アドバンスト UI インストールの UI で取得する必要があります。

アドバンスト UI またはスイート/アドバンスト UI インストールからのデータをアドバンスト UI またはスイート/アドバンスト UI インストール内の InstallScript パッケージに渡す

コマンドラインを使って、アドバンスト UI またはスイート/アドバンスト UI インストールからのデータをスイート内の InstallScript パッケージに渡すには、CMDLINE 変数を使用します。詳細については、「CMDLINE」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのプロパティの値を InstallScript パッケージに渡すには、InstallScript 関数 SuiteGetProperty または SuiteFormatString を使用します。

アドバンスト UI またはスイート/アドバンスト UI インストールから起動するためのランタイム検出

InstallScript インストールが、アドバンスト UI またはスイート/アドバンスト UI インストール内の InstallScript パッケージとして実行されているかどうかを判別するには、InstallScript コードで SUITE_HOSTED 変数を使用します。詳細については、「SUITE_HOSTED」を参照してください。

スイート/アドバンスト UI プロジェクトに InstallShield プロジェクト (.ism) をパッケージとして追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、スイート/アドバンスト UI プロジェクトに基本の MSI および InstallScript プロジェクト ファイル (.ism) をパッケージとして追加できます。1 つ以上の InstallShield プロジェクト パッケージを含むスイート/アドバンスト UI プロジェクトのリリースをビルドすると、関連する InstallShield プロジェクト内の指定されたリリースが最初にビルドされて、生成されるスイート/アドバンスト UI インストールにそれらがパッケージとして含まれます。InstallShield は基本の MSI プロジェクトおよび InstallScript プロジェクトの出力を .msi パッケージおよび .hdr パッケージとしてスイート/アドバンスト UI インストールに追加します。

スイート/アドバンスト UI プロジェクトに基本の MSI または InstallScript プロジェクトを含める場合、それらの基本の MSI または InstallScript パッケージはスイート/アドバンスト UI プロジェクトと同じバージョンの InstallShield で保存されていなくてはなりません。

タスク

基本の MSI プロジェクトまたは InstallScript プロジェクトをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[プロジェクト パッケージ (.ism)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。

3. スイート/アドバンスト UI プロジェクトに追加したい .ism ファイルを選択してから、**[開く]** をクリックします。

[パッケージ] エクスプローラーにプロジェクトが追加されます。右側のペインで設定を構成します。"製品構成" 設定 (基本の MSI プロジェクト パッケージの場合) および "リリース" 設定 (基本の MSI または InstallScript プロジェクト パッケージの場合) を使って、InstallShield でスイート/アドバンスト UI インストールに含めるリリースを選択します。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトに InstallScript パッケージを含める手順についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトの InstallScript パッケージについての特別考慮」を参照してください。

実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、実行可能パッケージ (.exe) をスイート/アドバンスト UI プロジェクトに追加できます。このフォーマットのスイート/アドバンスト UI パッケージは、優先されるフォーマット タイプの 1 つが使用できないときのみ推奨されます。詳細については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」を参照してください。

タスク

.exe パッケージをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しい実行可能パッケージ (.exe)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい .exe ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

スイート/アドバンスト UI プロジェクトで実行可能パッケージの構成するときの注意事項

次のタスクは、スイート/アドバンスト UI プロジェクトで実行可能パッケージを構成するときに推奨されます。

- スイート/アドバンスト UI インストールに対して異なるシナリオごとの動作を指示するための条件を定義します (たとえば、パッケージが既にインストール済みかどうかの検証、要件や依存関係の確認など)。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。
- 実行可能パッケージのユーザー インターフェイスを抑制し、スイート/アドバンスト UI インストールをサイレントで起動させます。そうしなかった場合、インストールの実行時に、スイート/アドバンスト UI ユーザー インターフェイスと .exe パッケージのユーザー インターフェイスの 2 つの異なるユーザー インターフェイスが表示されることになります。詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、Web 配置パッケージ (.zip) をスイート/アドバンスト UI プロジェクトに追加できます。実行時、スイート/アドバンスト UI インストールは、エンド ユーザーの入力に基づいて IIS Web サーバーまたはクラウド上に Web 配置パッケージを配置します。Web 配置パッケージは、Visual Studio などの IIS または Web アプリケーション開発環境を使って作成できます。

タスク

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しい Web 配置パッケージ (.zip)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい Web 配置パッケージ ファイルを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。通常、Web 配置パッケージは自己完結型で、.zip ファイルのみを必要とします。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

スイート/アドバンスト UI プロジェクトで Web 配置パッケージを構成するときの注意事項

Web 配置パッケージのライフサイクル

Web 配置パッケージは一般的に、初回インストールと同様、または既存の古いバージョンあるいは同一バージョンを上書きする場合と同様に動作するように設計されています。その結果、Web 配置パッケージではアンインストールを行うことができません。つまり、スイート/アドバンスト UI インストールはこの種類のパッケージのメンテナンスを行うことができません。1 つ以上の Web 配置パッケージ、または 1 つ以上の従来型パッケージ（メンテナンス オプションが含まれている）をプライマリ パッケージとして含むスイート/アドバンスト UI インストールの作成を避けることが推奨されます。

スイート/アドバンスト UI プロパティと、ハードコード化された Web 配置構成値の設定との違い

[パッケージ] ビューに Web 配置パッケージを選択すると、右側のペインにパッケージに関連する設定が表示されます。右側のペインの [共通] タブにある [構成] 領域には、Web 配置の構成関連の設定、たとえばインストール先（多くの場合はリモート サーバー）や資格情報などが含まれています。さらに、[構成] 領域には Web 配置パッケージのパラメーター XML ファイルで定義されるパラメーターの設定も含まれています。デフォルトでは、スイート/アドバンスト UI プロパティが構成設定すべての構成設定に使用され、エンド ユーザーはウィザード インターフェイスまたはコマンドラインを使って実行時にそれらを設定することができます。

プロパティのデフォルト値を含める場合、[プロパティ マネージャー] ビューを使って、対応するプロパティの値を指定します。

一部の状況において、1 つ以上のパラメーターにハードコード化された値を使用したい場合があります。たとえば、特定の Web 配置パラメーターを特定の値に設定するとき、そのパラメーターのデフォルト プロパティ名をハードコード化された値に置換する必要があります。

Web 配置パッケージを構成するためのウィザード インターフェイスについて

Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加すると、そのパッケージ用の定義済みの Web 配置ウィザード ページが自動的にプロジェクトに追加されます。Web 配置ウィザード ページを使って、エンド ユーザーは特定のパッケージをローカル IIS サーバー、リモート サーバー、またはクラウドベースのサーバーのいずれに配置するのかを指定することができます。また、ビルトイン Web 配置関連アクションを使うと、エンド ユーザーがパブリッシャー プロファイル ファイル（.publishsettings）から構成設定をロードすることも可能です。

エンド ユーザーが Web 配置パッケージのパラメーター XML ファイルで定義されているパラメーターを構成できるようにするには、必要に応じてウィザード インターフェイスにコントロールを追加して、それらのコントロールを [パッケージ] ビューの対応するパラメーター設定に指定されているプロパティに関連付けます。

サイドローディング Windows アプリ パッケージ（.appx|msix）をスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: この情報は、スイート/アドバンスト UI プロジェクトに適用します。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

メモ: *サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。*

InstallShield では、サイドローディング Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加できます。

アプリのサイドローディングとは、Windows Store を介さずにアプリケーションをインストールするプロセスです。この種類のアプリケーションは、企業環境で配布されることがあります。サイドロード アプリケーションの要件は以下の通りです:

- ターゲット システムに Windows 8 以降、または Windows Server 2012 以降が搭載されている。
- ターゲット システム上で **"すべての信頼されているアプリケーションのインストールを許可する"** グループ ポリシー設定が有効になっている。

サイドロードおよびこれらの種類のアプリケーションを実行するためのその他の要件は、MSDN Web サイトに掲載されている Windows ドキュメントを参照してください。

スイート/アドバンスト プロジェクトでは、ARM プロセッサ アーキテクチャをターゲットとする Windows アプリ パッケージを含める機能はサポートされていません。

タスク

Windows アプリ パッケージ (.appx) をスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しいサイドローディング Windows アプリ パッケージ (.appx)]** をクリックします。**[このパッケージのインストール ファイルを選択します]** ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい Windows アプリ パッケージを選択してから、**[開く]** をクリックします。**[このパッケージのファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加のフォルダーとファイルを含めるかどうかを指定します。含める場合、ダイナミック リンク ファイルを含めるかどうかも指定します。通常、Windows アプリ パッケージは自己完結形式で、Windows アプリ パッケージおよび証明書ファイル (.cer) のみを必要とします。

InstallShield によって、[パッケージ] エクスプローラーにパッケージおよび追加のファイルおよびフォルダーが追加されます。

サイドロード アプリケーション パッケージをスイート/アドバンスト UI プロジェクトに追加すると、InstallShield によってそのパッケージに Windows 8 以降または Windows Server 2012 以降の存在を確認する対象条件が自動的に追加されます。この対象条件は、スイート/アドバンスト UI インストールが Windows の以前のバージョンが搭載されているシステム上にパッケージがインストールされないように防ぎます。

サイドローディング Windows アプリ バンドル (.appxbundle | .msixbundle) をスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: *この情報は、スイート/アドバンスト UI に適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

メモ: *サイドローディング Windows アプリ バンドルを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows 8 以降または Windows Server 2016 以降が必要です。*

InstallShield では、サイドローディング Windows アプリ バンドル (.msixbundle) をスイート/アドバンスト UI プロジェクトに追加できます。

アプリのサイドローディングとは、Windows Store を介してアプリ バンドルを取得せずにインストールするプロセスです。この種類のアプリ バンドルは、企業環境で配布されることがあります。サイドロード アプリ バンドルの要件は以下の通りです:

- ターゲット システムに Windows 10、または Windows Server 2019 以降が搭載されている。
- ターゲット システム上で **"すべての信頼されているアプリケーションのインストールを許可する"** グループ ポリシー設定が有効になっている。

サイドロードおよびこれらの種類のアプリ バンドルを実行するためのその他の要件は、MSDN Web サイトに掲載されている Windows ドキュメントを参照してください。

タスク

Windows アプリ バンドル (.msixbundle) をスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. [パッケージ] エクスプローラーを右クリックしてから、**[新しい UWP アプリ バンドル (.appxbundle | .msixbundle)]** をクリックします。
   [このパッケージのインストール ファイルを選択します] ダイアログ ボックスが開きます。
3. スイート/アドバンスト UI プロジェクトに追加したい Windows アプリ バンドルを選択してから、**[開く]** をクリックします。
   このパッケージの **[ファイルを追加する]** ダイアログ ボックスが開きます。
4. パッケージ ファイルに追加フォルダーおよびファイルを含めるかどうかを指定します。その場合、ダイナミックにリンクされたファイルを含めるかどうかを指定します。通常、Windows アプリ バンドルは自己完結形式で、Windows アプリ バンドルおよび証明書ファイル (.cer) のみを必要とします。

5. InstallShieldによって、[パッケージ] エクスプローラーにバンドルおよび追加のファイルおよびフォルダーが追加されます。

メモ: *サイドロード アプリケーション バンドルをスイート/アドバンスト UI プロジェクトに追加すると、InstallShield によってそのバンドルに Windows 8 以降または Windows Server 2016 以降の存在を確認する対象条件が自動的に追加されます。この対象条件は、スイート/アドバンスト UI インストールが Windows の以前のバージョンが搭載されているシステム上にパッケージがインストールされないように防ぎます。*

Windows パッケージ マネージャー パッケージをスイート/アドバンスト UI プロジェクに追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Windows パッケージ マネージャー (WPM) パッケージをスイート/アドバンスト UI プロジェクに追加することができます。Windows パッケージ マネージャー (WPM) パッケージをインストールするには、ターゲット システム上に Windows パッケージ マネージャー (WPM) ソリューションがインストールされている必要があります。Windows パッケージ マネージャー(WPM) ソリューションは、アプリ インストーラー MSIX バンドルでインストールされます。Windows パッケージ マネージャー (WPM) パッケージが スイート/アドバンスト UI プロジェクトに追加されると、InstallShield は、アプリ インストーラー MSIX バンドル パッケージを依存ファイルとして追加します。

ターゲットマシン上でアプリ インストーラーが使用できない場合、InstallShield はアプリ インストーラー MSIX バンドル パッケージをインストールします。Windows パッケージ マネージャー は、Windows 10 1809 からアプリ インストーラーを通してサポートされています。

タスク

Windows パッケージ マネージャーをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[新しい Windows パッケージ マネージャー パッケージ]** をクリックします。

InstallShield の **[パッケージ]** エクスプローラーにパッケージが追加され、右側にあるペインの **[共通]** タブにある **[全般]** 領域で、その設定を構成することができます。**[全般]** 領域を使って、スイート/アドバンスト UI インストールを使ってパッケージ起動する条件とその詳細を指定します。

*メモ: スイート/アドバンスト UI プロジェクトをビルドする前に、**[全般]** 領域の下にある **"WPM パッケージ ID"** 設定で有効な WPM パッケージ ID が 不足している場合、Windows パッケージ マネージャー (WPM) パッケージが赤いアイコンで表示されます。*

スイート/アドバンスト UI プロジェクトで Windows パッケージ マネージャー パッケージを構成するときの注意事項

スイート/アドバンスト UI インストールに対して異なるシナリオごとの動作を指示するための条件を定義する必要があります (たとえば、パッケージが既にインストール済みかどうかの検証、要件や依存関係の確認など)。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い

***プロジェクト:** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション:** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、.msi、.msp、または .exe パッケージを追加または構成するとき、パッケージに、パッケージ ファイルの近くに保存されている追加ファイルが必要かどうかを示すことができます。たとえば、追加するパッケージが非圧縮の .msi パッケージである場合、その他のファイル (近くのサブフォルダーに保存されている .cab ファイルや非圧縮データ ファイルなど) をパッケージとファイルに含める必要が、場合によって、あります。

InstallShield では、パッケージの追加ファイルをプロジェクトに追加するとき、いくつかの方法がサポートされています:

- **すべての追加ファイルにスタティック リンクを使用する**—この方法は、パッケージの作成作業がすべて完了し、変更の可能性がなくなった場合に便利です。この方法は、アドバンスト UI またはスイート/アドバンスト UI インストールを設計中、すべての追加ファイルに対して、名前変更の可能性がほぼなくなった場合にも便利です。ビルド時に、[パッケージ] ビューに表示されているパッケージのすべてのスタティック ファイルが、アドバンスト UI またはスイート/アドバンスト UI インストールにコピーされます。
- **すべての追加ファイルにダイナミック リンクを使用する**—この方法では、追加ファイルのソースの場所を指定します。ビルド時に、ソースの場所に保存されているファイルがすべて、アドバンスト UI またはスイート/アドバンスト UI インストールにコピーされます。

この方法は、パッケージとアドバンスト UI またはスイート/アドバンスト UI インストールの両方を作成しているとき、および、ビルドとビルドの間で、パッケージに必要な追加ファイルのリストに変更が発生する可能性があるときに便利です。

- **追加のファイルにスタティックとダイナミック ファイル リンクを組み合わせて使用する**—この方法では、デザイン時に名前が分かっているすべての重要な追加ファイルに対してスタティック リンクを使い、その他の追加ファイルに対し、ダイナミック リンクを使うことができます。ビルド時に、[パッケージ] ビューに表示されているパッケージのすべてのスタティック ファイルが、アドバンスト UI またはスイート/アドバンスト UI インストールにコピーされます。また、必要に応じて、すべてのダイナミック ファイルもコピーされます。

メモ: *InstallShield では、パッケージ ファイル (.ms、.msp、InstallScript パッケージ、.exe、その他のパッケージ ファイル) に対して、常にスタティック リンクが含まれます。従って、パッケージ ファイルの名前は、ビルドが実行される度に変わることはありません。*

重要: *ダイナミック ファイル リンクの使用は注意が必要です。ダイナミック リンクが参照するソース ファイルからダイナミック リンクがあるファイルを誤って削除してしまった場合、そのファイルは、次回リリースをビルドしたとき、リリースに含まれません。このとき、ビルドの警告やエラーも表示されません。製品は問題なくインストールされる場合がありますが、誤って削除されたダイナミック リンクがあるファイルがインストールされないため、適切に動作しない可能性があります。*

InstallShield で、スタティックのソース場所とダイナミックのソース場所を識別する

スタティック リンク フォルダーが [パッケージ] ビューで表示されたとき、フォルダー アイコンは普通の黄色のフォルダーで表示されます。

ダイナミック リンク フォルダーが [パッケージ] ビューで表示されたときは、フォルダーのアイコン上に、ダイナミックにリンクされていることを示すイメージが表示されます。

InstallShield では、ダイナミック ファイルのイメージの他に、ダイナミックにリンクされているサブフォルダーを 1 つ以上含むフォルダーのアイコン上に、矢印も追加されます。

どちらの特殊フォルダー アイコンをクリックしても、コンテンツのソース場所などの設定を構成できます。また、ビルド時に使用されるフィルターを設定して、特定のファイルをビルドに含めたり、ビルドから除外したりできます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでダイナミック リンクを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[パッケージ] ビューを使って、プロジェクト内のパッケージに必要な追加ファイルのダイナミック リンクを作成できます。

ダイナミック ファイル リンクは、パッケージをプロジェクトに追加するときに含めることができます。また、ダイナミック ファイル リンクを、後でプロジェクトに追加することもできます。以下は、その両方の手順です。

パッケージを追加するときにダイナミック ファイル リンクを作成する

[パッケージ] ビューでプロジェクトにパッケージを追加すると、[このパッケージのファイルを追加する] ダイアログ ボックスが表示されます。このダイアログ ボックスで、追加するパッケージ ファイルに、追加するパッケージの近くにあるその他のファイルとフォルダーが必要であるかどうかを指定できます。たとえば、非圧縮の .msi パッケージをプロジェクトに追加する場合、場合によって、その他のファイルおよび、.msi パッケージによってターゲット システムにインストールされるファイルを含むフォルダーを含める必要があります。これらのフォルダーおよびその他のファイルは通常、.msi パッケージが含まれている同じフォルダーに保存されます。

[このパッケージのファイルを追加する] ダイアログ ボックスが表示されたら、次のいずれかを実行します:

- スタティック ファイルのみ含める場合、このダイアログ ボックスの **[ビルド時に見つかったファイルをダイナミックに追加する]** チェックボックスをクリアします。
- ダイナミック ファイルを含める場合は、スタティック ファイルを含める含めないにかかわらず、このダイアログ ボックスの **[ビルド時に見つかったファイルをダイナミックに追加する]** チェックボックスをクリアします。

次に、パッケージ ファイルに相対してソース ファイルがどこに存在するか表しているオプションを選択します。異なるオプションについての詳細は、「[このパッケージのファイルを追加する] ダイアログ ボックス」を参照してください。

パッケージが追加された後、ダイナミック ファイル リンクを作成する

タスク

パッケージがプロジェクトに追加されてから、追加ファイルのダイナミック ファイル リンクを追加するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、追加ファイルを必要とするパッケージを見つけて、対応するダイナミック リンクを持たせるフォルダーをクリックします。**[ファイル]** ペインで、現在のファイルとダイナミック フォルダーのリストが表示されます。
3. **[ファイル]** ペインで右クリックして **[ダイナミック リンクの追加]** をクリックします。**[フォルダーの参照]** ダイアログ ボックスが開きます。
4. ダイナミックに含めるファイルを含むフォルダーを参照して、**[OK]** をクリックします。

ダイナミック ファイル リンクの構成

いったんダイナミック ファイル リンクをプロジェクトに追加すると、ダイナミック ファイル リンクにファイルを含めるとき、および、除外するときのフィルター基準などの設定を構成することができます。詳しくは、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

いったんダイナミック ファイル リンクをプロジェクトに追加すると、ダイナミック ファイル リンクにファイルを含めるとき、および、除外するときのフィルター基準を指定することができます。

タスク ***ダイナミック ファイル リンクのフィルター基準を指定するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、ダイナミック リンク フォルダーを選択します。
3. **[ファイル]** ペインで、フィルター基準を構成するフォルダーをクリックします。
4. **[ダイナミック リンク]** ペインで、**"フィルター"** 設定の下にあるサブ設定を確認します。
5. 以下のいずれかを実行します。
   - フィルターを追加するには、以下の手順を実行します。
     a. **"フィルター"** 設定で、**[新しいフィルター]** ボタンをクリックして、追加するフィルターの種類に応じて、**[含めるファイル]**、**[除外するファイル]**、**[含めるフォルダー]**、または **[除外するフォルダー]** を選択します。新しい行が追加され、フィルターを定義できます。
     b. 新しい行で、ファイルまたはフォルダーの名前またはパターンを指定します。

        ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、すべての .htm ファイルを含める場合、**"含めるファイル"** 設定で ***.htm** と入力します。すべてのファイルを含める場合、***.*** と入力できます。

        パスでルート フォルダー (パッケージ ファイルを含むフォルダー) を参照する場合、ドットと円記号を使用します:

.¥

たとえば、**MyDirectory** という名のサブフォルダー内 (**MyDirectory** は、パッケージを含むフォルダーのサブフォルダーです) にある **ReadMe.txt** ファイルを参照する場合、次のようなフィルターを使うことができます:

.¥MyDirectory¥ReadMe.txt

- フィルターを変更するには、変更するフィルターの設定で、フィルターの適切な名前またはパターンを入力します。
- フィルター基準の順番を変更するには、移動するフィルターの設定で、**[フィルターの移動]** ボタンをクリックして、**[上に移動]** または **[下に移動]** をクリックします。
- フィルターを削除するには、削除するフィルターの設定で、**[このフィルターを削除]** ボタンをクリックします。

アドバンスト UI またはスイート/アドバンスト UI インストールのビルド時におけるフィルター基準の評価

InstallShield では、ビルド時に、ダイナミック リンク フォルダーに設定されたフィルターが使われて、特定のパッケージのリリースに含めるフォルダーとファイル、および、除外するフォルダーとファイルが判別されます。

InstallShieldで、ダイナミック リンクのソース フォルダー (パッケージを含むフォルダー) が確認されるとき、フィルターは、"フィルター" 設定の下に表示されている順番で上から下へと確認されます。最初にファイルに一致したファイル フィルターが、ファイルをビルドに含めるかどうかを判別するフィルターです。同様に、最初にフォルダーに一致したフォルダー フィルターが、フォルダーのファイルをビルドに含めるかどうかを判別するフィルターです。

たとえば、リスト内の最初のフィルターが次のような場合、**ReadMe.txt** という名前のファイルを含め、ソース フォルダー内にあるすべてのファイルが含められます:

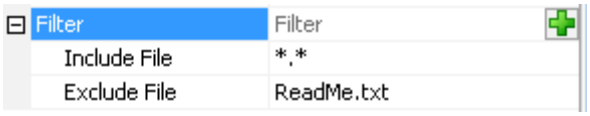


図 2: *.**.* フィルターが ReadMe.Txt フィルターの上に表示されているため、ReadMe.txt "除外するファイル" フィルターは、*.* "含めるファイル" フィルターによってオーバーライドされます。

ReadMe.txt 除外フィルターを *.* インクルード フィルターの上に移動した場合、ビルドに **ReadMe.txt** という名前のファイルを除いてすべてのファイルが含められます。

ダイナミック リンク フォルダーにフィルターが含まれていない場合、ソース フォルダー内のすべてのファイルが含められますが、サブフォルダー内のファイルは無視されます。

InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。既存の InstallShield 前提条件 (.prq ファイル) は、数に制限なく、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに追加することができます。InstallShield 前提条件を自作で作成して、それをアドバンスト UI およびスイート/アドバンスト UI プロジェクトに追加することもできます。

タスク

InstallShield 前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーを右クリックしてから、**[前提条件 (.prq) のインポート]** をクリックします。

メモ・ *スイート プロジェクトで Windows Package Manger (WPM) 前提条件のインポートはサポートされていません。*

InstallShield 前提条件は、[パッケージ] エクスプローラーに、前提条件に対して実行するように設定されたファイルの種類に応じて、.msi パッケージ、.msp パッケージ、.exe パッケージとして追加されます。

InstallShield 前提条件に依存関係がある場合 (つまり、1 つまたは複数の .prq ファイルが、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する InstallShield 前提条件で、依存関係として指定されている場合)、依存関係にある前提条件が [パッケージ] エクスプローラーに別のパッケージとして自動的に追加されます。

ヒント: *InstallShield 前提条件をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加すると、前提条件パッケージの設定に対して、デフォルトの値が自動的に構成されます。これらの設定は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでパッケージの設定を変更するのと同様、必要に応じて変更が可能です。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する」を参照してください。*

メモ: *アドバンスト UI およびスイート/アドバンスト UI パッケージで Windows Package Manger (WPM) および アプリ パッケージの前提条件はサポートされていません。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージとして含まれている InstallShield 前提条件の検出条件と評価条件を確認する

InstallShield 前提条件をアドバンスト UI またはスイート/アドバンスト UI プロジェクトをインポートしたとき、すぐにパッケージの検出条件と評価条件を確認し、必要に応じて変更を加えます。

たとえば、Windows オペレーティング システムのバージョン 5.0 以降すべてをターゲットする InstallShield 前提条件をインポートした場合、前提条件がプロジェクトに追加された時にデフォルトで構成される条件を変更した方が良い場合があります。たとえば、複数存在するプラットフォーム条件を、OS バージョン条件のサブ設定が **5.0–**（バージョン 5.0 以降という意味）に設定されている単一のプラットフォーム条件で置き換えることができます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクト内の各パッケージは、プライマリ パッケージまたは依存パッケージとして識別される必要があります。パッケージの種類によって、そのパッケージがターゲット システムにあることにより、アドバンスト UI またはスイート/アドバンスト UI インストールが初回インストール モードとして実行されるか、またはメンテナンス モードで実行されるかが決定されるかどうかが識別できます:

- **プライマリ パッケージ**—プライマリ パッケージは、アドバンスト UI またはスイート/アドバンスト UI インストールのメインの部分です。

  実行時、ターゲット システム上からインストールのプライマリ パッケージのすべてが不足している場合、インストールは初回インストールとして実行します。ターゲット システム上にスイート インストールのいずれかのプライマリ パッケージが存在する場合、インストールはメンテナンス モードで実行します。

- **依存パッケージ**—依存パッケージがターゲット システムにあるかないかは、インストールが実行されるときのモードに影響しません。

[パッケージ] ビューでパッケージを選択したときに使用できる "パッケージの種類" では、パッケージがプライマリ パッケージであるか依存パッケージであるかが示されます。

スイート/アドバンスト UI プロジェクトでは、プロジェクト内のパッケージに対して設定されている "パッケージの種類" 設定の値を必要に応じて変更することができます。ただし、アドバンスト UI プロジェクトでは、プライマリ パッケージの数が 1 つのみサポートされているため、この設定は読み取り専用になっています。

InstallShield 前提条件をアドバンスト UI プロジェクトまたはスイート/アドバンスト UI プロジェクトにインポートすると、InstallShield 前提条件が依存パッケージとしてプロジェクトに追加されます。スイート/アドバンスト UI プロジェクトでは、適宜 "パッケージの種類" 設定を使って、デフォルト値を上書きし、パッケージを

プライマリ パッケージとして指定することができます。ただし、InstallShield の Professional Edition では、アドバンスト UI プロジェクト内の InstallShield 前提条件は、常に依存パッケージとして識別され、デフォルトの動作を上書きすることはできません。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、パッケージとトランザクションのインストール順序を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI プロジェクトに、1 つ以上のパッケージ、トランザクション、または InstallShield 前提条件が含まれている場合、[パッケージ] ビューを使って、実行時のターゲット システムおける、パッケージのインストール順序とトランザクションの起動順序を指定できます。

同様に、アドバンスト UI プロジェクトに、パッケージおよび 1 つ以上の InstallShield 前提条件が含まれている場合、[パッケージ] ビューを使って、実行時のターゲット システムおける、パッケージのインストール順序を指定できます。

この [パッケージ] ビューでは、パッケージ、InstallShield 前提条件 パッケージ、およびトランザクション (サポートされている場合) が、アドバンスト UI またはスイート/アドバンスト UI インストールの実行時と同じ順番で表示されています。

タスク

ターゲット システムにパッケージをインストールする順番を変更するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、移動するパッケージまたはトランザクションを右クリックし、**[上に移動]** または **[下に移動]** をクリックします。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールにパッケージを追加するとき、設定を構成して、表示名や、アドバンスト UI またはスイート/アドバンスト UI インストールで、パッケージが起動されるときの条件などの情報を指定できます。

タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトでパッケージを構成するには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージをクリックします。
3. **[共通]** タブにある設定を使って、パッケージの条件を定義、およびその他の設定を構成します。

パッケージの設定についての詳細は、「共通」を参照してください。

実行時に、[パッケージ] ビューでリストされたパッケージに定義した条件とパッケージの順番に従って、Setup.exe ファイルが各パッケージを起動します。

ヒント: *構成中のパッケージが .msi パッケージである限り、MSI パッケージの対象条件および検出条件を定義する際に、現在のパッケージの製品コード、パッケージ コード、および製品バージョンのプレースホルダーとしてアスタリスクを使用することができます。*

また、構成中のパッケージが .msi パッケージである限り、MSI アップグレードの対象条件および検出条件を定義する際に、現在の製品のアップグレード コードおよび最小バージョン番号/最大バージョン番号のいずれかのプレースホルダーとしてアスタリスクを使用することができます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる各パッケージは、プロジェクト内の機能と関連付けなくてはなりません。

タスク ***ターゲット システムの再起動を必要とするアドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージを関連付けるには、次の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
3. **[機能]** タブをクリックします。
4. パッケージを含める各機能のチェック ボックスを選択します。

異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、複数のアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有することができます。2 以上のアドバンスト UI およびスイート/アドバンスト UI インストールが 1 つのパッケージを共有している場合に、アドバンスト UI およびスイート/アドバンスト UI 製品のすべてが削除されるまで、ターゲット システムにそのパッケージが保持されるようにします。

パッケージを共有としてマークするかどうかを判別するときのガイドライン

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージを構成するとき、これを共有としてマークすることができます。パッケージを共有としてマークする場合、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで、必ず同じパッケージ GUID を使用してください。

共有パッケージの機能は、通常プライマリ パッケージではなく、依存パッケージで使用できます。.NET Framework をインストールするパッケージなど、広く配布されるパッケージは、ほとんどの場合ターゲット システムにインストールされたままとなるため、通常共有としてマークされません。

共有機能はほとんどの種類のパッケージで使用できます (.msi、.msp、.exe、.appx、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。ただし、通常アンインストールされることがない Web 配布パッケージでは使用できません。

パッケージが共有かどうかを指定する

タスク

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでパッケージの共有を指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
3. **"共有"** 設定で、適切な値を選択します。
   - **いいえ**—パッケージは別のアドバンスト UI またはスイート/アドバンスト UI プロジェクトと共有されません。これがデフォルトの値です。
   - **はい**—パッケージは別のアドバンスト UI またはスイート/アドバンスト UI プロジェクトと共有されます。

     "共有" 設定に **[はい]** を選択した場合、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで必ず同じ パッケージ GUID を使用してください。**"パッケージ GUID"** 設定を使って、パッケージの GUID を確認および変更することができます。

共有パッケージの実行時の動作

実行時、アドバンスト UI またはスイート/アドバンスト UI インストールが既存しない共有パッケージをインストールするとき、インストールは次のレジストリ キーの下にある共有パッケージのパッケージ GUIDの 新しいレジストリ キーを作成します:

- 32 ビット システムの場合: HKEY_LOCAL_MACHINE¥SOFTWARE¥InstallShield¥SuiteInstallers¥Parcels
- 64 ビット システムの場合: HKEY_LOCAL_MACHINE¥SOFTWARE¥Wow6432Node¥InstallShield¥SuiteInstallers¥Parcels

インストールはまた、スイート GUID をパッケージ GUID レジストリ キーのレジストリ データとして追加します。別のアドバンスト UI またはスイート/アドバンスト UI インストールで同じ共有パッケージをインストールする必要がある場合、これらのアドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID がパッケージ GUID レジストリ キーのレジストリ データに追加されます。任意のアドバンスト UI またはスイート/アドバンスト UI プロジェクトのアンインストール中に、スイート GUID がパッケージ GUID レジストリ キーから削除されます。共有パッケージおよびパッケージ GUID レジストリ キーは、最後のアドバンスト UI またはスイート/アドバンスト UI 製品が削除されるまでターゲット システムに残ります。

2 つのアドバンスト UI またはスイート/アドバンスト UI インストールが、同じパッケージの異なるバージョンを共有する場合、どちらのアドバンスト UI またはスイート/アドバンスト UI インストールが先に実行されたかにかかわらず、新しいバージョンのパッケージは両方が実行された後にターゲット システムにインストールされたバージョンです。また、インストールは必要に応じて Parcels レジストリ キーをアップデートします。共有パッケージとパッケージ GUID レジストリ キーは、最後のアドバンスト UI またはスイート/アドバンスト UI 製品が削除されるまでターゲット システムに残ります。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、プロジェクト内の各パッケージに対して、異なるランタイムの場所を指定することができます。たとえば、パッケージとそのファイルをソース メディアに格納するか、必要な場合に Web からダウンロード可能にできます。または、パッケージをアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルに圧縮して、実行時に必要な場合に抽出することができます。

タスク

プロジェクト内の各パッケージに対して、異なるランタイムの場所を指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
3. **[場所]** リストで、適切なオプションを選択します。

指定した場所は、リリース レベルでオーバーライドすることができます。個々のパッケージに選択した値のオーバーライドを禁止するには、リリース レベルの "パッケージの場所" 設定を [個々の選択に従う] に設定します。詳細については、「リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する」を参照してください。

ヒント: *アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート セットアップ ランチャーの構成を行う場合、パッケージのランタイムの場所は、セットアップ ランチャーから抽出するか、Web からダウンロードします。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。*

詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージ操作を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

アドバンスト UI プロジェクトでは、.exe パッケージが InstallShield 前提条件からプロジェクトに含められた場合のみサポートされています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI の操作とは、アドバンスト UI またはスイート/アドバンスト UI インストールが実行されているときの状態を意味します。操作には、インストール、削除、修復、変更、および保守という異なる種類があります。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージを追加すると、そのパッケージに対して、様々な操作設定を構成することができます。設定を使って、操作ごとに、それらがパッケージに適用可能かどうかを指定できます。また、設定を使って、モード別に、アドバンスト UI またはスイート/アドバンスト UI のインストールで、パッケージが起動されたときに使用するコマンドラインを指定することもできます。コマンドラインは、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスと共に実行されたときのコマンドライン、または、サイレント（ユーザー インターフェイスなし）で実行されたときのコマンドラインと、別々に指定することができます。

インストール操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、パッケージをターゲット システムにインストールするとき、インストール操作が実行されます。インストール操作は、初回インストール、および、メンテナンス モードでインストールする機能が選択されたときに発生します。すべてのパッケージには、通常、インストール操作が設定されています。

アドバンスト UI またはスイート/アドバンスト UI インストールで、.msi または .msp、または InstallScript パッケージに対してインストール操作が実行されるとき、それらのパッケージは自動的にサイレントで起動されます。ただし、アドバンスト UI またはスイート/アドバンスト UI インストールで、.exe パッケージに対してインストール操作が実行されるときは、異なる種類の .exe パッケージで、ユーザー インターフェイスなしの異なる方法が使用される可能性があるため、パッケージを自動的にサイレントで起動することはできません。アドバンスト UI またはスイート/アドバンスト UI インストールで、.exe パッケージをサイレントで実行する必要がある場合、そのパッケージのインストール操作のサブ設定で適切なコマンドラインを入力します。

エンドユーザーによって構成可能なアドバンスト UI またはスイート/アドバンスト UI のパッケージが存在する場合、エンドユーザーがプロパティを構成できるウィザード ページを作成し、コマンドラインを使って、その関連付けられたプロパティをそれらのパッケージに渡します。

削除操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、パッケージをターゲット システムからアンインストールするとき、削除操作が実行されます。プライマリ パッケージとしてマークされているパッケージには、通常、削除操作が設定されています。逆に、依存パッケージとしてマークされているパッケージは、通常ターゲット システムにそのまま残されるため、概して削除操作は設定されていません。.NET Framework をインストールするパッケージは、依存パッケージとしてマークされるパッケージの 1 つの例です。パッケージの "パッケージの種類" 設定では、パッケージがプライマリ パッケージであるか、または依存パッケージであるかが識別されます。

メンテナンスの実行時に、アドバンスト UI またはスイート/アドバンスト UI の起動が無効にされたとき、また、アドバンスト UI またはスイート/アドバンスト UI 製品がアンインストールされたとき、削除操作が実行されます。インストール時同様、パッケージは、.exe パッケージのコマンドラインを使って、サイレントで実行します。

削除操作は、.msp パッケージには適用できませんので注意してください。また、.msp パッケージを削除するとき、ベースの .msi パッケージを削除する必要があることにも注意してください。

修復操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、一部のパッケージでサポートされている修復機能を実行する修復操作を実行することができます。修復機能をサポートするアドバンスト UI またはスイート/アドバンスト UI パッケージを構成する場合、パッケージの修復操作設定を構成して、修復機能を有効にします。アドバンスト UI またはスイート/アドバンスト UI インストールでは、アドバンスト UI またはスイート/アドバンスト UI インストールがメンテナンスの一部として修復されるとき、および、パッケージが既にインストールされている場合にのみ、修復操作が実行されます。

.msi パッケージに対して修復操作を構成する場合、アドバンスト UI またはスイート/アドバンスト UI インストールでは、修復機能が必要に応じて自動的に起動することが可能なため、サイレントで修復を実行する特別のコマンドラインを指定する必要はありません。ただし、.exe パッケージに対して修復操作を構成する場合、適切なコマンドラインを入力する必要があります。たとえば、基本の MSI の Setup.exe パッケージの場合、"修復操作" 設定の下にある "EXE コマンドライン" と "EXE サイレント コマンド ライン" サブ設定で次のコマンドラインを入力できます:

[SystemFolder]msiexec.exe /f {8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC} /qn

カッコ内の GUID は、Windows Installer パッケージの製品コードです。

修復操作は、.msp パッケージに適用できませんので注意してください。

変更操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、機能選択の実行に変更操作が実行されます。機能の選択を提供するアドバンスト UI またはスイート/アドバンスト UI パッケージを構成する場合、パッケージの変更操作設定を構成して、変更を有効にします。変更操作は、アドバンスト UI またはスイート/アドバンスト UI インストールの機能の選択で、パッケージの機能の追加または削除が指示されたときに実行されます。

変更操作は、.msp パッケージに適用できませんので注意してください。

保守操作

アドバンスト UI またはスイート/アドバンスト UI インストールでは、次のようなシナリオで、メンテナンス操作が実行されます:

- エンドユーザーが、[プログラムの追加と削除] のエントリから製品の変更を選択したとき。

- エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールを再実行したとき。

保守操作は、操作設定から制御できないという点で、他の種類の操作と異なります。

アドバンスト UI または スイート/アドバンスト UI インストールのパッケージにカスタム フォルダー名を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールをビルドしたとき、インストールに含まれている各パッケージに対してフォルダーが 1 つ作成されます。これらのフォルダーは、アドバンスト UI またはスイート/アドバンスト UI のセットアップ ランチャーと同じフォルダー内で作成されます。デフォルトで、各フォルダーの名前には、InstallShield で生成された GUID が使用されます。

InstallShield の [パッケージ] ビルドでは、今回から、これらのパッケージ フォルダーに対して、この GUID 名をオーバーライドして、ユーザーに分かりやすい名前を使用できるようになりました。

タスク

パッケージ フォルダーの名前に任意のテキストを指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、フォルダー名をカスタマイズするパッケージのノードを展開します。
3. **[パッケージ ファイル]** フォルダーを右クリックして、**[名前の変更]** をクリックします。

   メモ: *デフォルト名 (**パッケージ ファイル**) をそのまま使用する場合、パッケージ フォルダーの名前に GUID が使用されます。*
4. フォルダーの新しい名前を入力します。複数のパッケージ フォルダー名をカスタマイズする場合、各フォルダー名に異なる名前を使用してください。

ビルド時に、リリースが作成された時、パッケージ フォルダーの名前に、GUID ではなく、入力されたテキストが使用されます。

スイート/アドバンスト UI インストールの動作をカスタマイズする

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールを作成するとき、製品の要件およびエンド ユーザーのニーズに応じてカスタマイズすることが重要です。InstallShield では、スイート/アドバンスト UI インストールで、インストールに含めるパッケージの範囲を超えた様々なランタイム タスクを実行する機能を拡張することができます。詳細については、次を参照してください。

- スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する
- スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

一部の Windows の機能は、ターゲット システム上でデフォルトで無効です。また、エンド ユーザーまたは IT 管理者は、一部の Windows の機能を無効にしたい場合があります。そのような Windows の役割と機能を手動で有効または無効にするには、[コントロール パネル] の [プログラムと機能] の **[Windows の機能の有効化または無効化]** を使います。

スイート/アドバンスト UI インストール内の特定のパッケージが、ターゲット システム上で 1 つ以上の Windows の役割または機能が有効化されていることを必要とする場合、スイート/アドバンスト UI プロジェクトでパッケージを構成中に、必要な役割と機能を指定できます。実行時、インストールされる対象のパッケージが無効化されている 1 つ以上の Windows の役割または機能を必要とする場合、スイート/アドバンスト UI インストールは、パッケージを起動する前にこれらの役割および機能を有効化します。

たとえば、スイート/アドバンスト UI プロジェクトで Windows のインターネット インフォメーション サービス機能が有効であることを必要とする IIS Web サイトをインストールパッケージがあります。同じプロジェクト内の別のパッケージでは、PowerShell 機能が有効であることが必要です。プロジェクトのこれらのパッケージの設定

を構成するとき、必要な Windows 機能を指定します。実行時、パッケージがターゲット システム上でインストールの対象となっているが、必要な Windows 機能のいずれかが無効である場合、スイート/アドバンスト UI インストールは、起動前にこれらの無効化されている Windows 機能を有効にします。パッケージがインストールの対象ではない場合、必要な Windows 機能は有効化されません。

スイート/アドバンスト UI インストールがインストール中に有効化した任意の Windows の役割と機能は、アンインストール中に削除されません。そのため、ターゲット システム上でその他の製品が必要とするこれらの役割および機能の破損を回避することができます。

InstallShield には、いくつかの Windows 機能のビルトインサポートが搭載されています:

- インターネット インフォメーション サービス
- PowerShell
- .NET Framework 3.x

InstallShield ではまた、パッケージに必要な上記以外の Windows 役割と機能を指定することもできます。

タスク

特定のパッケージがインストールの対象であるときに有効化する Windows の役割または機能を指定するには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、構成するパッケージをクリックします。
3. **[共通]** タブにある **"Windows の機能"** 設定で **[Windows の機能を追加]** ボタンをクリックしてから、有効な機能オプションの 1 つを選択します:
   - **インターネット インフォメーション サービス**
   - **PowerShell**
   - **Microsoft .NET Framework 3.x**
   - **カスタム** – オプションを使って、この設定にオプションとしてリストされていない任意の Windows の機能を指定できます。

   InstallShield は、**"Windows 機能"** 設定の下に新しい **"Windows の機能"** 行を追加し、任意のビルトイン機能オプション (インターネット インフォメーション サービス、PowerShell、または Microsoft .NET Framework 3.x) の必要に応じて構成します。**[カスタム]** オプションでは、**[Windows の機能]** 行には機能名のプレースホルダ—文字列が含まれます。
4. ビルトイン オプションの 1 つを選択した場合、InstallShield によって **"Windows 機能"** 設定に追加された値をそのままに残します。

   [カスタム] オプションを選択した場合: **"Windows の機能"** 設定に、有効にする Windows の役割または機能の名前を入力します。カスタム オプションに入力する名前はすべて、Deployment Image Servicing and Management (DISM.exe) および Package Manager (Pkgmgr.exe) などのツールが Windows の特定のバージョンでロールまたは機能を識別するのに使用する文字列です。

   各 Windows のバージョンで使用可能な Windows の役割および機能の一覧は、Microsoft TechNet を参照してください。

Windows の役割および機能を有効化する際の特別考慮

様々な Windows バージョンで Windows の役割と機能が使用可能であるかどうか

Windows の役割と機能を有効化できるサポートは、Windows Vista 以降、または Windows Server 2008 以降を搭載するターゲット システム上で使用できます。指定のターゲット システム上で有効化することができる Windows の役割および機能のリストは、存在する Windows のバージョンによって異なります。また、異なるバージョンの Windows では、同じロールまたは機能に異なる文字列を使用する場合があります。

スイート/アドバンスト UI インストールがターゲット システム上で使用できない Windows の役割または機能を有効化しようと試みた場合、スイート/アドバンスト UI インストールは失敗して、ログファイルにエラー、0x800f080c が書き込まれます。

したがって、必要な Windows の役割と機能を指定する一部の状況下で、パッケージの "対象条件" 設定を使って、パッケージに選択している必要な Windows の役割または機能に対応する、適切なオペレーティング システムのバージョンを識別したい場合があります。有効にする Windows の役割または機能の名前にそれぞれ異なる文字列を使った複数バージョンの Windows をターゲットとするには、複数回にわたってパッケージをプロジェクトに追加する方法を考慮してください。パッケージの各インスタンスで、"Windows の機能" および "対象条件" 設定を使って、ターゲット オペレーティング システムに対応する適切な文字列を指定します。

Windows の一部のバージョンで、一部のロールおよび機能が無効化されないだけでなく、インストールもされない場合があります。スイート/アドバンスト UI インストールがターゲット システム上にインストールされていない Windows の役割または機能を有効化するように構成した場合、スイート/アドバンスト UI インストールはその役割または機能のインストールを試みます。ロールまたは機能をインストールできないとき、スイート/アドバンスト UI インストールは失敗します。このシナリオが当てはまる Windows の機能の例として .NET Framework 3.x 機能があり、これは Windows 8 システム上ではデフォルトでインストールされません。この機能のペイロードは、Windows アップデートを通してのみ使用可能です。インターネットへの接続が利用できないとき、または Windows アップデートの代わりに Windows Server Update Service を通してアップデートを取得するように構成されている場合、スイート/アドバンスト UI インストールで .NET Framework 3.x 機能をインストールおよび有効化することはできません。

必要な Windows の役割または機能の依存関係を識別する

一部の Windows の役割および機能は、その他の Windows の役割または機能に依存します。無効化されているロールまたは機能に依存関係があるかどうかを判別する方法の 1 つとして、次のコマンドを権限が昇格されたコマンド プロンプト ウィンドウ (cmd.exe のショートカットを右クリックしてから **[管理者として実行]** をクリック) から実行する方法があります:

```
dism /online /enable-feature FEAT-NAME
```

この例で、FEAT_NAM はチェックする無効化された機能の名前です。前述のコマンドが動作する場合、機能には依存関係がありません。これが失敗するとき、エラーに依存関係がリストされます。

パッケージで有効化するロールまたは機能が、ターゲット システムで無効化されている可能性のある 1 つ以上のロールまたは機能を必要とする場合、必ずそのパッケージの "Windows の機能" 設定で依存関係の機能も指定してください。

Windows の役割または機能を有効化するランタイムの動作

スイート/アドバンスト UI インストールで、一部のターゲット システム上で Windows の役割または機能を有効化する場合、スイート/アドバンスト UI インストールの InstallationProgress ウィザード ページで有効化処理中の役割または機能の進行状況を表示できます。この進行状況の情報は、Windows 8 以降または Windows Server 2012

以降が搭載されたターゲット システム上で使用できます。これらのターゲット システム上で、スイート/アドバンスト UI インストールは DISM API を使ってロールおよび機能を有効化し、これらの API には進行状況の情報をレポートする機能をサポートします。

Windows Vista、Windows Server 2008、Windows 7、および Windows Server 2008 R2 ターゲット システム上で、スイート/アドバンスト UI インストールは Pkgmgr.exe を使ってロールおよび機能を有効化します。Pkgmgr.exe は、進行状況の情報をレポートする機能をサポートしません。したがって、これらのシステム上ではロールおよび機能の有効化処理中に進行状況バーを更新することはできません。

Windows 7 および Windows Server 2008 R2 システム上で DISM.exe を使用できますが、これらの Windows バージョンでは API サポートが提供されていません。また、API DLL 依存関係は、ターゲット システムで再配布することができません。

スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールで、インストールに含めるパッケージの範囲を超えた様々な実行時タスクが必要な場合があります。たとえば、スイート/アドバンスト UI インストールで、以下の 1 つ以上の操作が必要な状況が考えられます。

- スイート/アドバンスト UI インストールのユーザー インターフェイスが表示される前にアプリケーションをインストールする。

  この状況が必要な例として、スイート/アドバンスト UI インストール内のパッケージの 1 つが Oracle データベースをインストールする SQL スクリプトを実行する時があります。このパッケージを実行する前に、スイート/アドバンスト UI インターフェイスで、エンド ユーザーがネットワーク上で使用可能なサーバーの一覧から適切なサーバーを選択できるウィザード ページを表示したい場合があります。この種類の UI サポートを使用するには、ターゲット システム上に ODBC ドライバーがインストールされていることが必要です。

- 特定の製品、テクノロジ、フォルダー、ファイル、レジストリ エントリ、その他のアイテムの存在の有無をターゲット システム上で検索する。

  ターゲット システムを検索するサポートを使って、特定の条件が満たされているかどうかに基づいて動作をトリガすることができます。たとえば、ターゲット システム上で特定のファイルが不足している場合、スイート/アドバンスト UI プロパティを適切に設定してから、そのプロパティを使ってスイート/アドバンスト UI インストールに含まれる 1 つ以上のパッケージでプロパティを設定できます。

- スイート/アドバンスト UI インストール内のパッケージを実行する前後にターゲット システムを構成する。

一部の状況、たとえばスイート/アドバンスト UI インストールがサーバー上の IIS アプリケーションをインストールする場合、ターゲット システム上で IIS 関連の Windows の役割および機能を有効化したい場合があります。その他、エンド ユーザーがスイート/アドバンスト UI インストールに含まれるパッケージのいずれを実行するか選択した内容によって、ターゲット システム上の構成ファイル、レジストリ データ、その他の項目を編集したい場合があります。

- エンド ユーザーによって、特定のウィザード インターフェイス コントロールを使用したとき、またはウィザード ページ または 2 番目のウィンドウが開いた/閉じたときにトリガされる初期設定を行う

  たとえば、ウィザード インターフェイス上のコンボ コンボ ボックスにアクションを追加して、コンボ ボックスに値をダイナミックに挿入したい場合があります。

スイート/アドバンスト UI インストールの機能を拡張して、前述のタスクその他を実行できるようにするには、スイート/アドバンスト UI プロジェクトの [イベント] ビューを使って、実行時に実行可能ファイルの実行、DLL 関数の呼び出し、PowerShell スクリプトの実行、スイート/アドバンスト UI プロパティの設定、InstallScript コードの実行、またはマネージ アセンブリのパブリック メソッドの呼び出しを行うアクションを作成することができます。

タスク ***アクションを作成してスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:***

1. インストールで必要な機能を処理する独自の .exe ファイル (適切な場合) 、DLL ファイル、PowerShell スクリプト、InstallScript ファイル (.rul)、またはマネージ アセンブリ (Visual Basic .NET または C# といったマネージ コードで作成) を作成します。

   InstallScript アクションの場合、プロジェクトの [InstallScript] ビューを使います。

2. [イベント] ビューで、ステップ 1 で作成したファイルを使用するアクションを作成します。詳細については、「アクションをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

3. アクションを実行するタイミングを決めて、スケジュールします。そのためには、アクションをイベントに追加するか、ウィザード インターフェイス要素に追加するか、またはそれをプロジェクト内の特定のパッケージに含まれるイベントに割り当てます。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI インストールに含まれるアクションの .exe ファイルでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールで .exe アクションを使って、インストールに含まれている (一時サポート ファイル、または製品とともにインストールされる) またはターゲット システムに既存する実行可能ファイルを起動できます。

次に、スイート/アドバンスト UI インストールに .exe アクションを追加する例を説明します。

- 独自の、またはサード パーティが作成した Setup.exe インストールを起動する .exe アクションを追加する。
- 既存ディレクトリのアクセス許可を設定する .exe アクションを追加する。
- ReadMe ファイルを、それに関連付けられたアプリケーション (メモ帳、または Adobe Reader) で開く .exe アクションを追加する。
- 製品に含まれるデータ ファイルを開く .exe アクションを追加する。

.exe アクションを スイート/アドバンスト UI プロジェクトに追加する最初の手順は、(適切な場合) .exe ファイルを作成または取得するか、実行可能ファイルが起動するドキュメントその他のファイルを作成することです。

.exe アクションは、実行中のインストール セッションにアクセスすることができません。したがって、(コマンド ライン引数を除いて) スイート/アドバンスト UI プロパティを .exe ファイルに渡したり、.exe ファイルから戻したりはできません (レジストリやファイルなどの外部ストレージ経由の場合を除く)。

タスク ***スイート/アドバンスト UI インストールで実行可能ファイルを実行するアクションを追加するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい EXE]** をクリックします。**[アクション]** エクスプローラーに実行可能ファイル アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

メモ: *.exe アクションからインタラクティブなアプリケーションを起動する場合、スイート/アドバンスト UI インストールがサイレント (ユーザー インターフェイスなし) で実行中に、それが起動することを防ぐための条件をアクションに追加した場合があります。この種類の条件で* ISSilentInstall *プロパティを使用できます。スイート/アドバンスト UI インストールがサイレント (ユーザー インターフェイスなし) で実行している場合、このプロパティは True に設定します。*

別の方法として、インストールのユーザー インターフェイスからインタラクティブなアプリケーションを起動したい場合もあります。つまり、.exe アクションを使って PDF ファイルを起動する代わりに、インストールのユーザー インターフェイスに含まれるウィザード ページの 1 つから PDF ファイルを開く機能を追加することができます。

スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DLL アクション用の DLL ファイルを作成する

スイート/アドバンスト UI プロジェクトに DLL アクションを追加する最初の手順は、DLL の作成です。DLL は、Visual C++ の最近のバージョン、または、COM および DLL のエクスポートをサポートしているすべてのツールや言語を使って作成できます。

スイート/アドバンスト UI エンジンでは、各アクションにエントリ ポイントが必要です。エントリ ポイントを指定するには、それを [イベント] ビューにあるアクションの "関数名" 設定に入力します。関数は、次のように定義します:

```
HRESULT _stdcall MyFunction(IDispatch *pAction);
```

この関数をアドバンスト UI またはスイート/アドバンスト UI インストールから呼び出せるようにするには、関数を適切にエクスポートする DLL をビルドするときに、定義ファイル (.def) を含める必要があります。次は、サンプル .def ファイルのコンテンツです。LIBRARY の後の名前は、DLL に使用した名前です。

```
LIBRARY "MyActionLibrary"
EXPORTS
    MyAction
```

スイート/アドバンスト UI インストールは、インストールに含まれるイベントまたはパッケージのアクションの実行がスケジュールされているときに関数のエントリ ポイントを呼び出します。エントリ ポイント関数は、アクションが正しく終了したことを示す ERROR_SUCCESS を戻します。アクションが失敗した場合、ゼロ以外の値が戻されます。値は、Windows Installer カスタム アクションが戻す値と同じです。

スイート/アドバンスト UI インストールでは任意のアクションを実行できますが、アクションはそれに構成されている権限を使って実行されます。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。その場合、アクションが必要なデータに正しくアクセスできるように、アクションが管理者権限を要求するように構成する必要があります。つまり、[イベント] ビューにあるアクションの "管理者権限が必要" 設定で [はい] を選択する必要があります。別の方法として、スイート/アドバンスト UI インストールの Setup.exe ファイルを管理者権限を使って実行するか、管理者マニフェストを含みます。ただし、管理者権限を昇格するのは、できるだけ短時間に抑える必要があります。

アクションのエントリ ポイントに渡される IDispatch インターフェイスは、ISuiteExtension2 インターフェイス (すべての ISuiteExtension メソッドとプロパティを含む) を実装します。ISuiteExtension2 へのポインターを取得するには、IDispatch インターフェイスで QueryInterface メソッドを呼び出します。ISuiteExtension2 インターフェイスでは、アクション関数を使って、スイート/アドバンスト UI プロジェクトでアクションに定義された属性パラメーターにアクセスすることができます。プロジェクト内の各アクションは、それぞれのアクション インスタンスに固有の異なるインターフェイス ポインターが渡されます。したがって、エントリ ポイント関数に渡されたインターフェイス ポイントを常に使用し、それを他のアクション DLL の呼び出し用に保存しないようにしてください。

インターフェイスは、次のように定義されています。

```
interface ISuiteExtension2 : IDispatch {
  [propget, id(1), helpstring("Attribute")]
    HRESULT Attribute([in]BSTR bstrName, [out, retval]BSTR *bstrValue);
```

```
  [id(2), helpstring("method LogInfo")]
    HRESULT LogInfo([in]BSTR bstr);

  [propget, id(3), helpstring("Property")]
    HRESULT Property([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  [propput, id(3), helpstring("Property")]
    HRESULT Property([in]BSTR bstrName, [in]BSTR bstrValue);

  [id(4), helpstring("FormatProperty")]
    HRESULT FormatProperty([in]BSTR bstrValue, [out, retval]BSTR *bstrReturnValue);

  [id(5), helpstring("ResolveString")]
    HRESULT ResolveString([in]BSTR bstrStringId, [out, retval]BSTR *bstrReturnValue);

  [id(6), helpstring("ProgressMessage")]
    HRESULT SendProgressMessage([in]BSTR bstrMsg, [in]INT iCurrent, [in]INT iMax,
                                [in]EnumProgressFlags eFlags);
};
```

スイート/アドバンスト UI プロジェクトでは、関数プロトタイプを使って、ISuiteExtension2 COM インターフェイス ポインターがアクションに渡されます。これにより、次の関数にアクセスできるようになります:

- **LogInfo**

  HRESULT LogInfo([in]BSTR bstr);

  このメソッドにより、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。

- **get_Property**

  HRESULT get_Property([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  このメソッドによって、現在実行中のスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。

- **put_Property**

  HRESULT put_Property([in]BSTR bstrName, [in]BSTR bstrValue);

  このメソッドを使って、現在実行中のスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。

- **FormatProperty**

  HRESULT FormatProperty([in]BSTR bstrValue, [out, retval]BSTR *bstrReturnValue);

  この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

- **ResolveString**

HRESULT ResolveString([in]BSTR bstrStringId, [out, retval]BSTR *bstrReturnValue);

このメソッドはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されているスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

- **SendProgressMessage**

  HRESULT SendProgressMessage([in]BSTR bstrMsg, [in]INT iCurrent, [in]INT iMax,
  [in]EnumProgressFlags eFlags);

  このメソッドは、スイート/アドバンスト UI インストールで InstallationProgress ウィザード ページのステータス メッセージと進行状況バーを更新します。進行状況バーを更新するには、eFlags の epfProgressValid を指定し、ステータス メッセージを更新するには、epfMessageValid を指定します。ステータス メッセージと進行状況バーの両方を更新するには、両方のフラグと共にビット単位の OR 演算子 (|) を使用します。

get_Attribute メソッドはスイート/アドバンスト UI インストールの DLL アクションでは使用できないので、これを呼び出すことはできません。

作成した DLL から ISuiteExtension2 インターフェイスにアクセスするには、#import を使って、InstallShield と共にインストールされている SetupSuite.exe ファイルからタイプ ライブラリ情報を組み込むことができます。次は、そのパスです:

InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386¥SetupSuite.exe"

たとえば、VC++ プロジェクト内のタイプ ライブラリ (例、stdafx.h) をインポートするには、次のステートメントを使用します:

```
#import "C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386¥SetupSuite.exe" no_namespace raw_interfaces_only named_guids
```

InstallShield が別の場所にインストールされている場合、#import ステートメントのパスを、それに従って変更する必要があります。

DLL アクションを追加する

タスク ***スイート/アドバンスト UI インストールの実行中に DLL ファイルの関数を呼び出すアクションを追加するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい DLL]** をクリックします。**[アクション]** エクスプローラーに DLL アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Windows PowerShell は、構成タスクのオートメーション化を可能にする .NET Framework ベースのコマンドラインシェルおよびスクリプト言語です。InstallShield では、PowerShell スクリプト (.ps1) を実行するスイート/アドバンスト UI インストールのアクションを含めることができます。この種類のアクションは、インストールの実行時にシステムの構成タスクを実行するプロジェクトに追加することができます。PowerShell アクションには、ターゲット システム上に PowerShell 2.0 以降が必要です。

PowerShell アクションを追加する

タスク

スイート/アドバンスト UI インストールで PowerShell スクリプトを実行するアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい PowerShell]** をクリックします。**[アクション]** エクスプローラーに PowerShell アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

[アクション] エクスプローラー、または [イベント] エクスプローラーで PowerShell アクションを選択すると、InstallShield の右側に [PowerShell スクリプト] タブが表示されます。このタブを使ってスクリプトを編集できます。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

PowerShell アクションの実行時要件

スイート/アドバンスト UI プロパティ IS_CLR_VERSION を使って、PowerShell スクリプトを実行するためにアクションがロードする、セミコロン区切りの .NET Framework バージョン一覧を識別することができます。IS_CLR_VERSION プロパティについては、「 アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

実行中のスイート/アドバンスト UI インストールとのインタラクション

実行中の スイート/アドバンスト UI インストールとのインタラクションを可能にする、いくつかの cmdlet があります:

テーブル 2 ・ 実行中のスイート/アドバンスト UI インストールとインタラクションする cmdlet

Cmdlet	説明
get-suiteproperty -name [プロパティ名]	cmdlet がスイート/アドバンスト UI プロパティの値を取得します。Name パラメーターは、位置指定パラメーターです。 次のサンプル コードは、MYPROPERTY という名前のプロパティの値を取得します。 $Value = get-suiteproperty -name MYPROPERTY
set-suiteproperty -name [プロパティ名] -value [値]	この cmdlet がスイート/アドバンスト UI プロパティの値を設定します。 次のサンプル コードは、MYPROPERTY プロパティの値を使って、NEWPROPERTY と呼ばれる 2 番目のプロパティの値を設定します。 $Value = get-suiteproperty -name MYPROPERTY set-suiteproperty -name NEWPROPERTY -value $Value
format-suiteproperties -Value [format string]	この cmdlet は、形式化された式の値を展開します。Name および Value パラメーターは位置指定パラメーターです。 次のサンプル コードは、NEWPROPERTY と呼ばれるスイート/アドバンスト UI プロパティを含む文字列を解決します。プロパティ名と、その周りの角かっこは、実行時にプロパティの値で置換されます。 $myFormat = format-suiteproperties -Value "これは、[NEWPROPERTY] が埋め込まれたテキスト文字列です。"
resolve-suitestring -StringId [文字列 ID]	この cmdlet は、スイート/アドバンスト UI 文字列識別子を、現在選択されている UI 言語でそれに対応する文字列値で解決します。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。StringId パラメーターは、位置指定パラメーターです。 次のサンプル コードは、**IDS_MY_MESSAGE** と呼ばれる文字列識別子を解決します: $StringValue = resolve-suitestring -StringId IDS_MY_MESSAGE

テーブル 2 ・ 実行中のスイート/アドバンスト UI インストールとインタラクションする cmdlet (続き.)

Cmdlet	説明
trace-suiteinfo -LogMessage [長い文字列]	この cmdlet により、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。LogMessage パラメーターは、位置指定パラメーターです。 次のサンプル コードは、**IDS_MY_MESSAGE** と呼ばれる文字列識別子を解決してから、その値をスイート/アドバンスト UI デバッグ ログ記録に書き込みます。 $StringValue = resolve-suitestring -StringId IDS_MY_MESSAGE trace-suiteinfo -LogMessage $StringValue

スクリプトが成功したことを示すには、必ず 0 を戻します。たとえば、次の行でスクリプトを終了します:

exit(0)

プロジェクトの [イベント] ビューで PowerShell アクションを追加した後、[アクション] エクスプローラーまたは [イベント] エクスプローラーでそれを選択し、InstallShield の右に表示される [PowerShell スクリプト] タブを使ってスクリプトを編集できます。

スイート/アドバンスト UI インストールに含まれるプロパティを設定するアクションでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、スイート/アドバンスト UI プロジェクトに、実行時にスイート/アドバンスト UI プロパティを設定するアクションを追加できます。

タスク

スイート/アドバンスト UI インストールで プロパティを設定するアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しいプロパティ セット]** をクリックします。**[アクション]** エクスプローラーにプロパティ アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。

4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、スイート/アドバンスト UI プロジェクトに、実行時に InstallScript コードを呼び出すアクションを追加できます。スイート/アドバンスト UI ベースの InstallScript アクションは、基本の MSI プロジェクトの InstallScript カスタム アクションとほぼ同じ要領で機能します。

スイート/アドバンスト UI プロジェクトに IInstallScript アクションを追加するには、まず始めにインストールから実行する InstallScript コードを作成します。この作業には、プロジェクトの [InstallScript] ビューを使います。コードを作成したら、プロジェクトの [イベント] ビューを使って InstallScript アクションを追加します。

タスク

スイート/アドバンスト UI インストールで InstallScript コードを実行するアクションを追加するには、以下の手順に従います:

1. InstallScript コードをプロジェクトに追加するには、以下の手順に従います:
   a. **[動作とロジック]** の下のビュー リストで、**InstallScript** をクリックします。
   b. **InstallScript** エクスプローラーで、**[ファイル]** を右クリックして、**[新しいスクリプト ファイル]** を選択します。**[ファイル]** ノードの下に Setup.rul という名前の新しいファイルが追加されます。
   c. 新しい .rul ファイルを選択してから、実行時に起動するコードを右側のペインで作成します。

      スイート/アドバンスト UI アクションで InstallScript 関数を呼び出す場合、次の例のように、**export** キーワードを使って関数をプロトタイプ化します。

      ```
      export prototype MyFunction(OBJECT);
      ```

2. InstallScript コードを呼び出す InstallScript アクション を追加します:
   a. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
   b. **[アクション]** エクスプローラーを右クリックしてから、**[新しい InstallScript]** をクリックします。**[アクション]** エクスプローラーに InstallScript アクションが追加されます。
   c. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。

d. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

スイート/アドバンスト UI アクションの InstallScript コードを書くためのヒント

一部の InstallScript 関数は InstallScript アクションのエントリ ポイントに渡される ISuiteExtension オブジェクトをラップします。これらの InstallScript 関数を使って、InstallScript アクションが実行中のスイート/アドバンスト インストールとインタラクトできるようにします。

- **SuiteFormatString**
- **SuiteGetProperty**
- **SuiteLogInfo**
- **SuiteResolveString**
- **SuiteSetProperty**

これらの関数についての詳細は、「スイート/アドバンスト UI およびアドバンスト UI の対話関数」を参照してください。

スイート/アドバンスト UI インストール内のアクションから呼び出される InstallScript 関数を使用する際、次の詳細にご注意ください:

- InstallScript UI 関連の関数を、スイート/アドバンスト UI プロジェクトで使用することはできません。これを使用すると、スクリプトのコンパイル エラーが発生するか、コンパイルが完了しても、実行時にエラーが発生します。。InstallScript アクションで表示される UI はすべて、スイート/アドバンスト UI インストールの UI が表示されるときまで遅延させることが推奨されます。
- InstallScript アクションは、スイート/アドバンスト UI プロパティを読み書きして、それらをスイート UI とのコミュニケーション手段として使用します。(スイート/アドバンスト UI アクションは、基本の MSI カスタム アクションと同様に、UI にアクセスすることができません。)ビルトイン プロパティの一覧は、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。
- InstallScript 実行時パス変数 ((PROGRAMFILES、WINDIR など) は使用できません。これらの変数は通常、「*バインドされた変数*」として知られるスクリプト エンジンの概念を使って実装されます。バインドされた変数は、変数そのものがテキストサブ名値のマッピングに直接関連付けられています。これらの変数に対応するテキストサブは引き続き使用できます。 たとえば、<PROGRAMFILES> は、InstallScript 関数 **TextSubGetValue** または **TextSubSubstitute** を通して解決することができます。パス型スイート/アドバンスト UI プロパティも、スイート/アドバンスト UI エンジンを通して定義され、**SuiteGetProperty** で読み込むことができます。
- LAAW_PARAMETERS 構造体変数の szStatusText メンバーを、スイート/アドバンスト UI インストールで呼び出される InstallScript アクションに使用することはできません。

InstallScript デバッガーを使って、スイート/アドバンスト UI インストールの InstallScript アクションをデバッグできます。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。

スイート/アドバンスト UI インストールでマネージ コード アクション を使用する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield を使って、実行時に Visual Basic .NET または C# などのマネージ コードで書かれたマネージ アセンブリでパブリック メソッドを呼び出すスイート/アドバンスト UI プロジェクトに、マネージ コード アクション タイプを追加することができます。

マネージ コード アクション用のマネージ アセンブリを作成する

スイート/アドバンスト UI プロジェクトにマネージ コード アクションを追加する最初の手順は、マネージ アセンブリの作成です。マネージ アセンブリは、任意の最近のバージョンの .NET 言語 (Visual Basic .NET または C# など) を使って作成できます。

スイート/アドバンスト UI エンジンでは、各マネージ コード アクションにつき 1 つのエントリ ポイントが必要です。この場合のエントリ ポイントとは、スタティック メソッドか、パラメーターなしで構成可能なクラス上にあるインスタンス メソッドのどちらかです。

エントリ ポイントを指定するには、**[イベント]** ビューにあるアクションの **"クラス"** 設定および **"メソッド"** 設定を使います。

スイート/アドバンスト UI インストールは、インストールに含まれるアクションの実行がスケジュールされているときにエントリ ポイントを呼び出します。エントリ ポイント メソッドは、アクションが正しく終了したことを示す 数値 0 を戻します。アクションが失敗した場合、ゼロ以外の値が戻されます。戻されるエラーの値は、Windows Installer カスタム アクションで定義されている戻り値と同じです。

アクションのエントリ ポイントに渡されるインターフェイスは、オブジェクトとして渡される ISuiteExtension インターフェイスを実装します。戻り値の型は 32 ビット整数型でなくてはなりません。

スイート/アドバンスト UI 拡張インターフェイスを使用する場合、たとえば Visual Studio で C# コードを作成するときに次のようなコードを含めることができます:

```
[Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
[TypeLibType(4288)]
public interface ISuiteExtension
  {
     [DispId(1)]
     string get_Attribute(string bstrName);
     [DispId(2)]
     void LogInfo(string bstr);
     [DispId(3)]
     string get_Property(string bstrName);
     [DispId(3)]
     void set_Property(string bstrName, string bstrValue);
     [DispId(4)]
```

```
        string FormatProperty(string bstrValue);
        [DispId(5)]
        string ResolveString(string bstrStringId);
    }
```

マネージ コードで次のメソッドを使用できます:

- **LogInfo** — このメソッドにより、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。
- **get_Property** — このメソッドによって、現在実行中のスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。
- **set_Property** — このメソッドを使って、現在実行中のスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。
- **FormatProperty**—この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用可能な構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
- **ResolveString** — このメソッドはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されているスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

次のサンプル C# コードは、TestCLRAction、TestCLRClass の "クラス" 設定値、TestCLRMethod の "メソッド" 設定値に対応します:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using Microsoft.WindowsAzure.Common;
using Microsoft.WindowsAzure.Management;
using Microsoft.WindowsAzure.Management.WebSites;
using Microsoft.WindowsAzure.Management.WebSites.Models;
using Microsoft.WindowsAzure;
using System.Windows.Forms;

namespace TestCLRAction
{
    [Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
    [TypeLibType(4288)]
    public interface ISuiteExtension
    {
        [DispId(1)]
        string get_Attribute(string bstrName);
```

```
    [DispId(2)]
    void LogInfo(string bstr);
    [DispId(3)]
    string get_Property(string bstrName);
    [DispId(3)]
    void set_Property(string bstrName, string bstrValue);
    [DispId(4)]
    string FormatProperty(string bstrValue);
    [DispId(5)]
    string ResolveString(string bstrStringId);
}

public class TestCLRClass
{
    const UInt32 ERROR_INSTALL_FAILURE = 1603;
    const UInt32 ERROR_SUCCESS = 0;

    ISuiteExtension GetExtensionFromDispatch(object pDispatch)
    {
        if (Marshal.IsComObject(pDispatch) == false)
            throw new System.ContextMarshalException("無効な
            ディスパッチ オブジェクトが CLR メソッドに渡されました");

        return (ISuiteExtension)pDispatch;
    }
    public UInt32 TestCLRMethod(object pDispatch)
    {
        try
        {
            ISuiteExtension suiteExtension = GetExtensionFromDispatch(pDispatch);
            suiteExtension.set_Property("TESTPROP", "hello, world");
        }
        catch (System.ContextMarshalException)
        {
            return ERROR_INSTALL_FAILURE;
        }

        return ERROR_SUCCESS;
    }
}
```

メモ: *スイート/アドバンスト UI インストールでは任意のアクションを実行することが可能ですが、アクションは構成された特権を使って実行されます。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。その場合、アクションが必要なデータに正しくアクセスできるように、アクションが管理者権限を要求するように構成する必要があります。つまり、[イベント] ビューにあるアクションの "管理者権限が必要" 設定で [はい] を選択する必要があります。別の方法として、スイート/アドバンスト UI インストールの Setup.exe ファイルを管理者権限を使って実行するか、管理者マニフェストを含みます。ただし、特権の昇格はできるだけ短時間に限ります。*

マネージ コード アクションを追加する

タスク

スイート/アドバンスト UI インストールの実行中にマネージ アセンブリのパブリック メソッドを呼び出すアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しいマネージ コード]** をクリックします。**[アクション]** エクスプローラーに マネージ コード アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

マネージ コード アクションの実行時要件

スイート/アドバンスト UI プロパティ IS_CLR_VERSION を使って、マネージ コードを実行するためにアクションがロードする、セミコロン区切りの .NET Framework バージョン一覧を識別することができます。IS_CLR_VERSION プロパティについては、「 アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

マネージ コード アクションに依存関係がある場合、[サポート ファイル] ビューを使って、それらをプロジェクトに追加します。実行時、スイート/アドバンスト UI エンジンは、 .NET Framework に [SETUPSUPPORTDIR] 内に依存関係があることを指定します。

スイート/アドバンスト UI インストールで .NET Core アクションを使用する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield を使って、実行時に Visual Basic .NET または C# などのマネージ コードで書かれたマネージ アセンブリでパブリック メソッドを呼び出すスイート/アドバンスト UI プロジェクトに、.NET Core アクション タイプを追加することができます。

.NET Core アクションのマネージ アセンブリを作成する

スイート/アドバンスト UI プロジェクトに .NET Core アクションを追加する場合、最初にマネージ アセンブリを作成します。マネージ アセンブリは、任意の最近のバージョンの .NET 言語 (Visual Basic .NET または C# など) を使って作成できます。

スイート/アドバンスト UI エンジンは、各 .NET Core アクションにエントリ ポイントが必要です。この場合のエントリ ポイントとは、スタティック メソッドか、パラメーターなしで構成可能なクラス上にあるインスタンス メソッドのどちらかです。

エントリ ポイントを指定するには、**[イベント]** ビューにあるアクションの **"クラス"** 設定および **"メソッド"** 設定を使います。

スイート/アドバンスト UI インストールは、インストールに含まれるアクションの実行がスケジュールされているときにエントリ ポイントを呼び出します。エントリ ポイント メソッドは、アクションが正しく終了したことを示す 数値 0 を戻します。アクションが失敗した場合、ゼロ以外の値が戻されます。戻されるエラーの値は、Windows Installer カスタム アクションで定義されている戻り値と同じです。

アクションのエントリ ポイントに渡されるインターフェイスは、オブジェクトとして渡される ISuiteExtension インターフェイスを実装します。戻り値の型は 32 ビット整数型でなくてはなりません。

スイート/アドバンスト UI 拡張インターフェイスを使用する場合、たとえば Visual Studio で C# コードを作成するときに次のようなコードを含めることができます:

```
[Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
[TypeLibType(4288)]
public interface ISuiteExtension
  {
    [DispId(1)]
    string get_Attribute(string bstrName);
    [DispId(2)]
    void LogInfo(string bstr);
    [DispId(3)]
    string get_Property(string bstrName);
    [DispId(3)]
    void set_Property(string bstrName, string bstrValue);
    [DispId(4)]
    string FormatProperty(string bstrValue);
    [DispId(5)]
    string ResolveString(string bstrStringId);
  }
```

マネージ コードで次のメソッドを使用できます:

- **LogInfo** — このメソッドにより、スイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。
- **get_Property** — このメソッドによって、現在実行中のスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。
- **set_Property** — このメソッドを使って、現在実行中のスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。

- **FormatProperty**—この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用可能な構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
- **ResolveString** — このメソッドはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されているスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

次のサンプル C# コードは、TestCLRAction、TestCLRClass の "クラス" 設定値、TestCLRMethod の "メソッド" 設定値に対応します:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Runtime.InteropServices;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using Microsoft.WindowsAzure.Common;
using Microsoft.WindowsAzure.Management;
using Microsoft.WindowsAzure.Management.WebSites;
using Microsoft.WindowsAzure.Management.WebSites.Models;
using Microsoft.WindowsAzure;
using System.Windows.Forms;

namespace TestCLRAction
{
    [Guid("BAFAEAED-08C6-4679-B94E-487A4D89DE63")]
    [TypeLibType(4288)]
    public interface ISuiteExtension
    {
        [DispId(1)]
        string get_Attribute(string bstrName);
        [DispId(2)]
        void LogInfo(string bstr);
        [DispId(3)]
        string get_Property(string bstrName);
        [DispId(3)]
        void set_Property(string bstrName, string bstrValue);
        [DispId(4)]
        string FormatProperty(string bstrValue);
        [DispId(5)]
        string ResolveString(string bstrStringId);
    }

    public class TestCLRClass
    {
        const UInt32 ERROR_INSTALL_FAILURE = 1603;
        const UInt32 ERROR_SUCCESS = 0;

        ISuiteExtension GetExtensionFromDispatch(object pDispatch)
        {
            if (Marshal.IsComObject(pDispatch) == false)
```

```
            throw new System.ContextMarshalException("無効な
            ディスパッチ オブジェクトが CLR メソッドに渡されました");

        return (ISuiteExtension)pDispatch;
    }
    public UInt32 TestCLRMethod(object pDispatch)
    {
        try
        {
            ISuiteExtension suiteExtension = GetExtensionFromDispatch(pDispatch);
            suiteExtension.set_Property("TESTPROP", "hello, world");
        }
        catch (System.ContextMarshalException)
        {
            return ERROR_INSTALL_FAILURE;
        }

        return ERROR_SUCCESS;
    }
}
```

メモ: *スイート/アドバンスト UI インストールでは任意のアクションを実行することが可能ですが、アクションは構成された特権を使って実行されます。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。その場合、アクションが必要なデータに正しくアクセスできるように、アクションが管理者権限を要求するように構成する必要があります。つまり、[イベント] ビューにあるアクションの "管理者権限が必要" 設定で [はい] を選択する必要があります。別の方法として、スイート/アドバンスト UI インストールの Setup.exe ファイルを管理者権限を使って実行するか、管理者マニフェストを含みます。ただし、特権の昇格はできるだけ短時間に限ります。*

.NET Core アクションを追加する

タスク

スイート/アドバンスト UI インストールの実行中にマネージ アセンブリのパブリック メソッドを呼び出すアクションを追加するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、**[新しい .NET Core アクション]** をクリックします。
   [アクション] エクスプローラーに .NET Core アクションが追加されます。
3. 新しい名前を入力するか、または名前を後で右クリックして、**[名前の変更]** をクリックして新しい名前を割り当てます。このアクションに、プロジェクト内の他のアクションから容易に区別できる名前を使用します。
4. 必要に応じて、アクションの設定を構成します。

プロジェクトにアクションを追加したあと、それを必要に応じてスケジュールします。詳細については、「スイート/アドバンスト UI アクションをスケジュールする」を参照してください。

.NET Core アクションの実行時要件

.NET Core タイプのカスタム アクションを使用するには、ターゲット システム上に Microsoft .NET Core 8.0 が必要です。

.NET Core アクションに依存ファイルがある場合、**[サポート ファイル]** ビューを使ってこれらをプロジェクトに追加します。実行時、スイート/アドバンスト UI エンジンが .NET Core に対して、関連ファイル が [SETUPSUPPORTDIR] にあることを指定します。

アクションをスイート/アドバンスト UI プロジェクトに追加する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

タスク ***アクションをスイート/アドバンスト UI プロジェクトに追加するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーを右クリックしてから、追加するアクションの種類をクリックします。

   PowerShell アクションを作成する場合、**[開く]** ダイアログ ボックスが開いて、使用する PowerShell スクリプト ファイル (.ps1) を選択することができます。
3. オプションで、新しいアクションを選択し、F2 キーを押し、新しい名前を入力して名前を変更できます。
4. アクションの設定を構成します。

実行時にアクションが起動またはロードできない場合、インストールは中止します。

スイート/アドバンスト UI アクションの設定を構成する

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

タスク ***スイート/アドバンスト UI プロジェクトでアクションを構成するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[アクション]** エクスプローラーで、構成するアクションの種類をクリックします。
3. 右側に表示されているグリッド内の設定を構成します。

実行時にアクションが起動またはロードできない場合、インストールは中止します。

スイート/アドバンスト UI アクションをスケジュールする

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

プロジェクトにアクションを追加すると、それを実行時に起動するタイミングをスケジュールできます。スイート/アドバンスト UI エンジンがアクションを起動する前に評価する 1 つ以上の条件を定義することもできます。

InstallShield では、アクションを種類の異なる場所でスケジュールすることができます:

- スイート/アドバンスト UI が管理する 1 つ以上のビルトイン イベントの実行中にアクションをスケジュールできます。たとえば、スイート/アドバンスト UI インストールがロードされた直後にスイート/アドバンスト UI エンジンは OnBegin と呼ばれるイベントの実行をトリガします。このイベントは、OnBegin イベントが発生するときに実行する操作を指定します。
- スイート/アドバンスト UI エンジンがパッケージをインストール、削除、変更、または修復する前後にアクションの実行をスケジュールできます。
- アクションをウィザード インターフェイスの要素 (たとえば、ウィザード ページまたはウィザード ページのコントロール) と関連付けることができます。

詳細については、「スイート/アドバンスト UI インストールにおけるイベントの種類」を参照してください。

イベント中にアクションをスケジュールする

[イベント] ビューの [イベント] エクスプローラーの下にリストされているイベントとアクションは、実行時に起動される順番に時系列で編成されています。アクションをイベントに追加する際、アクションを適切なイベントに追加してアクションが起動されるタイミングを指定します。

タスク ***イベント中にアクションをスケジュールするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[イベント]** をクリックします。
2. **[イベント]** エクスプローラーで、そのアクションの直前に位置するイベントまたはアクションを右クリックして、**[既存アクションを追加]** をポイントしてから、スケジュールするアクションの名前をクリックします。**[イベント]** エクスプローラーにアクションが追加されます。
3. 右側に表示されているグリッド内の設定を構成します。設定の 1 つを使って、アクションが起動されるかどうかを制御する、条件ステートメントをビルドすることができます。この条件の定義方法に関する詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」をご覧ください。

   アクションが特定のモードでのみ実行することを示すには、"プロパティの比較" 条件でスイート/アドバンスト UI プロパティ ISInstallMode を使用します。このプロパティ、およびその他のビルトイン プロパティの詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

ヒント: *[イベント] ビューでは、ドラッグアンドドロップ編集がサポートされています:*

- *新しいアクションをシーケンスするには、[アクション] エクスプローラーからそれを、[イベント] エクスプローラーの下にある適切なイベントにドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。*
- *アクションを、イベント内の別の位置へ (または、あるイベントから別のイベントへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。*
- *アクションをあるイベントから別のイベントへコピーするには、CTRL を押しながら、そのアクションをもう 1 つのイベントまでドラッグし、直前にくるアクションまたはイベントの上にドロップします。*

スイート/アドバンスト UI エンジンがパッケージをインストール、削除、変更、または修復する前後にアクション行をスケジュールする

タスク ***パッケージがインストール、削除、変更、または修復する前後にアクションをスケジュールするには、以下の手順に従います:***

1. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
2. **[パッケージ]** エクスプローラーで、アクションと関連付けるパッケージをクリックします。
3. **[共通]** タブで、**[イベント]** 領域が展開されていない場合、これを展開します。
4. 以下のいずれかを実行します。
   - パッケージがインストール、削除、変更、または修復する前にアクションをスケジュールする場合: **"構成済みパッケージ"** 設定で、**[新しいアクション]** ボタンをポイントしてから、スケジュールするアクションの名前をクリックします。InstallShield によって、**"アクション"** 設定が追加され、その値が選択されたアクションの名前に設定されます。
   - パッケージがインストール、削除、変更、または修復した後にアクションをスケジュールする場合: **"構成済みパッケージ"** 設定で、**[新しいアクション]** ボタンをポイントしてから、スケジュールするアク

ションの名前をクリックします。InstallShield によって、**"アクション"** 設定が追加され、その値が選択されたアクションの名前に設定されます。

5. オプションで、アクションの起動時により詳細な制御を行うための条件ステートメントをビルドしたい場合は、**[条件の追加]** ボタンをクリックします。この条件の定義方法に関する詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」をご覧ください。

   インストールに含まれる特定のパッケージが、特定の状態で実行している場合のみアクションを実行することを示すには、[パッケージの操作] 条件を作成します。

   アクションが特定のモードでのみ実行することを示すには、"プロパティの比較" 条件でスイート/アドバンスト UI プロパティ ISInstallMode を使用します。このプロパティ、およびその他のビルトイン プロパティの詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。

ウィザード インターフェイス内の要素にアクションを追加する

InstallShield では、ウィザード ページまたは 2 番目のウィンドウが開いているまたは閉じているときに起動されるアクションを定義することができます。また InstallShield では、エンド ユーザーがボタンをクリックしたときなど、様々なウィザード インターフェイス コントロールを使用するとトリガされるアクションを定義できます。ウィザード インターフェイスを使ってアクションをスケジュールする方法については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。

スイート/アドバンスト UI インストールにおけるイベントの種類

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

スイート/アドバンスト UI インストールは、特定の順番で一連のイベントを管理するスイート/アドバンスト UI インストールによって駆動します。これらのイベントは、プロジェクトに追加されたアクションを実行します。

次のテーブルは、スイート/アドバンスト UI インストールの [イベント] ビューにリストされている各イベントについて説明します:

テーブル 3 - [イベント] ビューにリストされているイベント

イベント名	説明
OnBegin	OnBegin イベントはスイート/アドバンスト UI インストールの最初のイベントです。このイベントにスケジュールされているアクションは、そのアクションに定義されている条件ステートメントが False 評価されない限り、すべてのモード (インストール、メンテナンス、修復など) で実行されます。 アクションが特定のモードでのみ実行することを示すには、"プロパティの比較" 条件でスイート/アドバンスト UI プロパティ ISInstallMode を使用します。このプロパティ、およびその他のビルトイン プロパティの詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。 OnBegin 中にスケジュールされたアクションは、スイート/アドバンスト UI インストールがウィザード インターフェイスを表示する前に実行します。このイベント中に長時間実行するアクションは、エンド ユーザーがインストールを起動した時点からウィザード インターフェイスが初めて表示されるまでに長い遅延をもたらす可能性があります。その結果、インストールが正しく起動されていないと、エンド ユーザーが勘違いする場合があります。これを避けるために、OnBegin 中に長時間実行するアクションをスケジュールしないことが推奨されます。 また、起動時に独自のユーザー インターフェイスを表示するアクションも避けて下さい。スイート/アドバンスト UI インストールのウィザード インターフェイスは、既に開いている別のウィンドウの背面に表示され、エンド ユーザーを混乱させる可能性があります。
OnResuming	OnResuming イベントは、ターゲット マシンが再起動された後、スイート/アドバンスト UI インストールが再開するときに、そのアクションを起動します。
OnStaging	OnStaging イベントは、スイート/アドバンスト UI インストールがステージングされる直前に、そのアクションを起動します。つまり、(適切な場合) スイート/アドバンスト UI インストールがパッケージをダウンロードして、(適切な場合) Setup.exe ファイルからパッケージを抽出し、BrowseStageFolder ウィザード ページでエンド ユーザー が指定するディレクトリ、またはコマンドラインを使って **ISRootStagePath** プロパティで設定されたディレクトリにパッケージをコピーする前にアクションを起動します。
OnStaged	OnStaged イベントは、(適切な場合) スイート/アドバンスト UI インストールがパッケージをダウンロードし、Seup.exe ファイルから抽出して、ステージング ディレクトリにコピーした後に、そのアクションを起動します。
OnPackagesConfiguring	PackagesConfiguring イベントは、スイート/アドバンスト UI インストールがインストール、変更、修正、または削除される直前に、そのアクションを起動します。

テーブル 3 ・ [イベント] ビューにリストされているイベント (続き.)

イベント名	説明
OnPackagesConfigured	OnPackagesConfigured イベントは、スイート/アドバンスト UI インストールがインストール、変更、修正、または削除された時に、そのアクションを起動します。
OnRebooting	OnRebooting イベントは、スイート/アドバンスト UI インストールがマシンの再起動をトリガする直前に、そのアクションを起動します。 OnRebooting 中にスケジュールされたアクションは、短時間で実行しなくてはなりません。そうでない場合、前に実行されているアクションによって再起動が開始され、再起動イベントが完了する前に Windows がスイート/アドバンスト UI プロセスをシャットダウンした場合、アクションが完了せずに失敗することがあります。
OnEnd	OnEnd イベントは、スイート/アドバンスト UI インストールが正常に終了する直前に、そのアクションを起動します。エンド ユーザーがスイート/アドバンスト UI インストールをキャンセルした場合、またはインストールでエラーが発生した場合、OnEnd イベントは実行しません。

スイート/アドバンスト UI エンジンは、実行済み/未実行のイベントの状況を管理しません。そのため、再起動によってスイート/アドバンスト UI インストールが再開した後に、アクションが再実行されることを避けるため、アクションの [プロパティの比較] 条件を定義しなくてはならない場合があります。これらの条件で ISOnRebooted プロパティを使用できます。スイート/アドバンスト UI インストールが再起動の後に再開する場合、このプロパティは True に設定します。

アクションに "機能の操作" 条件または "パッケージの操作" 条件を定義する場合、アクションをスケジュールできる最初のイベントは OnStaging です。

アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI インストールがインストールの終了の前に、様々な条件に応じて表示する終了エラー メッセージを指定できます。たとえば、アドバンスト UI またはスイート/アドバンスト UI インストールに Windows Vista 以降が必要な場合、エンド ユーザーに対して要件を通知する終了メッセージを設定できます。また、終了メッセージに条件を定義して、ターゲット システムに Windows の以前のバージョンが搭載されているかどうかを評価することもできます。エンドユーザーがアドバンスト UI またはスイート/アドバンスト UI インストールを起動したとき、定義された条件が True となった場合、エラー メッセージが表示されます。エンドユーザーがエラー メッセージ ボックスを閉じると、インストールは終了します。

タスク

アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件およびメッセージを定義するには、以下の手順に従います:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"終了条件"** 設定で、**[新しい条件]** ボタンをクリックします。**"終了条件"** 設定の下に終了メッセージのサブ設定が追加されます。
3. **"終了メッセージ"** 設定に、構成中の条件が実行時に True 評価された場合にアドバンスト UI またはスイート/アドバンスト UI インストールで表示するエラー メッセージを入力します。
4. **"終了メッセージ"** 設定を拡張してから、そのサブ設定を使って実行時にアドバンスト UI またはスイート/アドバンスト UI インストールが評価を行う条件ステートメントを定義します。
   - 最初のサブ設定では、その設定内のリストを使って、構成している条件、または条件グループに使用するオプションを選択します。選択可能なオプションは以下のとおりです:
     - Any—Any 条件グループは、論理演算子 OR と同じ要領で動作します。Any グループに含まれる任意の条件が True 評価された場合、条件グループ全体が True 評価されます。Any グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が False 評価されます。
     - All—All 条件グループは、論理演算子 AND と同じ要領で動作します。All グループが True 評価されるためには、All グループに含まれる条件のすべてが True 評価される必要があります。
     - None—None 条件グループは、論理演算子 NOR と同じ要領で動作します。None グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が True 評価されます。None 条件グループに含まれる任意の条件が True 評価された場合、条件グループ全体が False 評価されます。None 条件グループが 1 つの条件ステートメントで構成される場合、これは論理演算子 NOT と同じ要領で動作します。
   - サブ条件グループを追加するには、この設定で **[新しい条件]** ボタンをクリックしてから、必要に応じて **Any**、**All**、または **None** を選択します。条件の定義に使用できる新しい行が追加されます。
   - 条件を追加するには、この設定で **[新しい条件]** ボタンをクリックしてから、使用可能な条件の種類から 1 つを選択します。条件の定義に使用できる新しい行が追加されます。

条件の構成についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

各条件設定についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールは、次のいずれかのモードで実行されます:

- インストール モード。初回インストールとして動作するインストール。
- メンテナンス モード。インストールで、エンド ユーザーが製品の削除、修復、変更を選択できるウィザード ページが表示されます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、インストール モードおよびメンテナンス モードの条件が自動的に作成されます。モード条件は、内では編集できません。また、モード条件はプロジェクト ファイル (.issuite) でのみ表示されます。これらのモードは、アドバンスト UI またはスイート/アドバンスト UI インストールを初回インストール モードで実行するか、メンテナンス モードで実行するかを判別します。

インストール モードの条件

インストール モードの条件は、次の要因に基づきます:

- **検出条件**—インストール モード条件の一部は、スイート/アドバンスト UI プロジェクトに含まれるすべてのプライマリ パッケージ (アドバンスト UI プロジェクトの場合、1 つのプライマリ パッケージ) の検出条件から構成されています。検出条件が True に評価されたプライマリ パッケージが存在しない場合 (既にインストールされているプライマリ パッケージが存在しない場合)、インストール モード条件の検出条件の部分は True と評価されます。検出条件が True に評価されたプライマリ パッケージが 1 つ以上存在する場合 (既にインストールされているプライマリ パッケージが最低 1 つ存在する場合)、インストール モード条件の検出条件の部分は False と評価されます。
- **インストール済みスイート条件**—インストール モード条件のもう片方の部分は、インストール済みスイート条件で構成されています。この条件により、同じバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールがターゲット システムに存在しないとき、インストールは初回インストール モードでトリガされることがあります。インストール モード条件のンストール済みスイート条件は、適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID およびバージョンを、ターゲット システムに既にインストールされているアドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID およびバージョンと比較します。これらの値が異なる場合、インストール モード条件のンストール済みスイート条件によって、アドバンスト UI またはスイート/アドバンスト UI インストールが存在していないということが示唆され、これにより、初回インストールが実行されることがあります。[インストール済みスイート] タイプの条件チェックに関する詳細は、「特定のバージョンのアドバンスト UI またはス

イート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」を参照してください。

検出条件が False に評価されたプライマリ パッケージが 1 つ以上存在する場合、または、インストール済みスイート条件で、アドバンスト UI またはスイート/アドバンスト UI インストールがまだインストールされていないと判断された場合、インストール モード条件は True として評価され、インストールが初回インストールとして実行されます。

メンテナンス モードの条件

メンテナンス モード条件は、スイート/アドバンスト UI プロジェクトに含まれるすべてのプライマリ パッケージ（アドバンスト UI プロジェクトの場合、1 つのプライマリ パッケージ）の検出条件に基づきます。検出条件が True に評価されたプライマリ パッケージが 1 つ以上存在する場合（既にインストールされているプライマリ パッケージが最低 1 つ存在する場合）、モード条件は True として評価され、インストールはメンテナンス モードで実行されます。

スイート/アドバンスト UI プロジェクトと Web 配置パッケージに関する特別考慮

Web 配置パッケージを検出することはできません。これらのパッケージは一般的に、初回インストールと同様、または既存の古いバージョンあるいは同一バージョンを上書きする場合と同様に動作するように設計されています。その結果、Web 配置パッケージではアンインストールを行うことができません。つまり、スイート/アドバンスト UI インストールはこの種類のパッケージのメンテナンスを行うことができません。1 つ以上の Web 配置パッケージ、または 1 つ以上の従来型パッケージ（メンテナンス オプションが含まれている）をプライマリ パッケージとして含むスイート/アドバンスト UI インストールの作成を避けることが推奨されます。

アドバンスト UI またはスイート/アドバンスト UI インストールの [プログラムの追加または削除] エントリ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、アドバンスト UI またはスイート/アドバンスト UI インストールで、[プログラムの追加または削除] エントリを作成するかどうかを指定できます。このエントリを使って、エンド ユーザーは必要に応じてアドバンスト UI またはスイート/アドバンスト UI インストールの管理、変更、または削除を行うことができます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [一般情報] ビューには、適切な動作を指定できる ″[プログラムの追加と削除] エントリの表示″ 設定があります。

アドバンスト UI またはスイート/アドバンスト UI インストール全体に対して単一のエントリのみを表示する場合、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるパッケージからのエントリを非表示にしてください。

たとえば .msi パッケージの場合、[パッケージ] ビューにあるパッケージのコマンドライン設定を使って、Windows Installer プロパティ ARPSYSTEMCOMPONENT の値を 1 に設定できます。

[プログラムの追加または削除] エントリを表示" 設定は、アドバンスト UI またはスイート/アドバンスト UI 製品が [プログラムの追加または削除] にエントリされるかどうかが決定される唯一の要因ではないので注意してください。メンテナンス モード条件によっても、エントリされるかどうかが決定されます。メンテナンス モード条件は、インストール内のすべてのプライマリ パッケージの検出条件をすべて組み合わせた条件です。アドバンスト UI またはスイート/アドバンスト UI インストール内のすべてのプライマリ パッケージに対する検出条件が、初回インストール後、False と評価された場合、アドバンスト UI またはスイート/アドバンスト UI 製品の [プログラムの追加または削除] エントリは追加されません。アドバンスト UI またはスイート/アドバンスト UI インストール内のすべてのプライマリ パッケージに対する検出条件が、アドバンスト UI またはスイート/アドバンスト UI 製品が削除された後で True に評価された場合、既存の [プログラムの追加または削除] エントリはそのまま残されます。

コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールプロジェクトの [パッケージ] ビューでパッケージを構成するとき、コマンドラインおよびサイレント コマンドライン設定を使って、パッケージを起動したときにアドバンスト UI またはスイート/アドバンスト UI インストールで使用するコマンドラインを指定できます。これらの設定は、[パッケージ] ビューのグリッド内にある [操作] 領域のサブ設定で使用できます。

アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスと共に実行される場合、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルは、"MSI コマンドライン" 設定、"MSP コマンドライン" 設定、"コマンドライン" 設定、または "EXE コマンドライン" 設定 (パッケージの種類によって異なる) に入力されたコマンドラインを使用してパッケージを起動します。インストールがサイレント (ユーザー インターフェイスなし) で実行されるとき、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルは、"MSI サイレント コマンドライン" 設定、"MSP サイレント コマンドライン" 設定、"サイレント コマンドライン" 設定、または "EXE サイレント コマンドライン" 設定に入力されたコマンドラインを使用してパッケージを起動します。

これらの設定に値を入力するとき、構成するパッケージの種類によって以下の背景情報を参照してください。

.msi パッケージ

.msi パッケージのインストールおよび削除操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer プロパティです。アドバンスト UI またはスイート/アドバンスト UI インストールは MsiInstallProduct を使って、.msi パッケージをインストール モードまたは削除モードで起動します。この関数には、コマンドラインとして Windows Installer プロパティのみを使用できます。

この種類のパッケージの修復および変更操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer 機能プロパティのみです。

- ADDDEFAULT
- ADDLOCAL
- ADDSOURCE
- ADVERTISE
- COMPADDDEFAULT
- COMPADDLOCAL
- COMPADDSOURCE
- FILEADDDEFAULT
- FILEADDLOCAL
- FILEADDSOURCE
- MSIDISABLELUAPATCHING
- MsiPatchRemovalList
- MSIRESTARTMANAGERCONTROL
- MSIDISABLERMRESTART
- MSIRMSHUTDOWN
- MSIPATCHREMOVE
- PATCH
- REINSTALL
- REINSTALLMODE
- REMOVE

アドバンスト UI またはスイート/アドバンスト UI インストールは MsiConfigureProductEx を使って .msi パッケージを修復 モードまたは変更モードで起動します。この関数には、Windows Installer プロパティのみを使用できます。

パッケージの "MSI コマンドライン" 設定および "MSI サイレント コマンドライン" 設定に Windows Installer プロパティを使用している場合、以下の形式を使用してください:

MYPROPERTYNAME=MyPropertyValue

/l コマンドライン オプションをパッケージに渡してログ ファイルを生成することはできません。インストールをログ記録する場合、[パッケージ] ビューで、パッケージに対してログを有効にできます。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。別の方法として、インストールをログ記録したい場合は、logging system policyまたは MsiLogging プロパティのどちらかを使用します。

アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msi パッケージは常にサイレントで起動されますので注意してください。そのため、.msi パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。

.msp パッケージ

.msp パッケージのインストール操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer プロパティのみです。アドバンスト UI またはスイート/アドバンスト UI インストールは MsiApplyPatch を使って .msp パッケージをターゲット システムに適用します。この関数には、コマンドラインとして Windows Installer プロパティのみを使用できます。

.msp パッケージを使ってすべての機能をアップデートするには、以下のようなコマンドライン プロパティを入力します:

REINSTALLMODE=vomus REINSTALL=ALL

.msp パッケージに含まれている特定の機能のみをアップデートするには、アップデートするコンマ区切りの機能リストに REINSTALL を設定します。

REINSTALLMODE=vomus REINSTALL=Feature1,Feature3,Feature5

アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msp パッケージは常にサイレントで起動されますので注意してください。そのため、.msp パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。

InstallScript パッケージ

InstallScript パッケージの [インストール] 操作と [変更] 操作を構成する場合、コマンドラインで、アドバンスト UI およびスイート/アドバンスト UI のプロパティ ISFeatureInstall と ISFeatureRemove を使用できます。InstallScript 関数 FeatureConfigureFeaturesFromSuite は、これらのプロパティを使って、InstallScript パッケージの機能を設定します。

"コマンドライン" 設定と "サイレント コマンドライン" 設定では、次のように、これらのプロパティに、複数の機能名をカンマ区切りで設定することができます:

ISFeatureInstall=Feature1,Feature2 ISFeatureRemove=Feature3

上記の例では、**Feature1** と **Feature2** は、インストールされるように選択されています。**Feature3** は、存在していれば、削除されるように選択されています。

機能が、両方のプロパティの値で使用されている場合、**ISFeatureRemove** プロパティに設定された機能は、**ISFeatureInstall** プロパティに優先されます。

アドバンスト UI およびスイート/アドバンスト UI インストールでは、デフォルトで、InstallScript パッケージはサイレントで起動されますので注意してください。従って、InstallScript パッケージのユーザー インターフェイスを非表示にするコマンドライン パラメーターを渡す必要はありません。また、サイレント応答ファイルを使用する必要もありません。

.exe パッケージ

.exe パッケージのインストール、削除、修復、および変更操作を構成している場合、.exe ファイルがサポートしている任意のコマンドライン パラメーターを入力できます。

エンド ユーザーによる操作なしですべてのカスタマイズを完了する場合、.exe パッケージをサイレントで実行するコマンドライン パラメーターを含めることができます。

Windows Installer ベースのインストール用に InstallShield でビルドされた Setup.exe ファイルをサイレントで実行する

アドバンスト UI またはスイート/アドバンスト UI インストールで、Windows Installer ベースの Setup.exe ファイルをサイレントで実行する場合、"EXE コマンドライン" 設定と "EXE サイレント コマンド" 設定で、次の構文を使用します:

```
Setup.exe /s /v"/qn"
```

Windows Installer ベースのパッチ用に InstallShield でビルドされた Update.exe ファイルをサイレントで実行する

アドバンスト UI またはスイート/アドバンスト UI インストールで、Windows Installer ベースの Update.exe パッチをサイレントで実行する場合、"EXE コマンドライン" 設定と "EXE サイレント コマンド" 設定で、次の入力します:

```
Update.exe /s /v"/qn"
```

InstallScript インストールのセットアップ ランチャー ファイルをサイレントで実行する

アドバンスト UI またはスイート/アドバンスト UI インストールで、InstallScript の Setup.exe インストールをサイレントで起動する場合、サイレント応答ファイルを作成して、それを、スイート プロジェクトの [サポート ファイル] ビューに、サポート ファイルとして追加します。[インストール] 操作の "EXE コマンドライン" 設定と "EXE サイレント コマンドライン" 設定で、次の構文を使用します:

```
Setup.exe /s /f1"[SETUPSUPPORTDIR]¥Setup.iss"
```

アンインストールで、元の Setup.exe ファイルをサイレントで起動する代わりに、アンインストール用にサイレント応答ファイルを作成して、それを、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [サポート ファイル] ビューに、サポート ファイルとして追加します。[削除] 操作の "EXE コマンドライン" 設定と "EXE サイレント コマンドライン" 設定で、次の構文を使用します:

```
"[ProgramFilesFolder]InstallShield Installation Information¥[PRODUCT-GUID-HERE]¥Setup.exe" /s /f1"[SETUPSUPPORTDIR]¥Uninstall.iss" -remove_only -runfromtemp
```

アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI 形式の式を使って、アドバンスト UI またはスイート/アドバンスト UI インストールで、アドバンスト UI またはスイート/アドバンスト UI パッケージを起動するときに使用するコマンドラインを動的に構成することができます。形式化された式は、プロパティ名、環境変数リファレンス、その他の特殊文字列から構成されます。実行時に、インストールはこれらの式の値を拡張します。

一部のシナリオでは、単一のアドバンスト UI またはスイート/アドバンスト UI で形式化された式を使ってパッケージのほとんどまたはすべてを構成できます。その他のシナリオでは、各パッケージに異なるアドバンスト UI またはスイート/アドバンスト UI の形式化された式が必要です。

これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

次のサンプル シナリオは、アドバンスト UI またはスイート/アドバンスト UI プロパティを形式化された式として使用して、インストール内で値をパッケージに渡す方法を説明します。

アドバンスト UI またはスイート/アドバンスト UI プロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストール内の複数のパッケージに同じ値を渡す

アドバンスト UI またはスイート/アドバンスト UI インストールを作成するとき、アドバンスト UI またはスイート/アドバンスト UI プロパティを使って Windows Installer プロパティを設定し、各 .msi パッケージをインストールする方法を変更する場合があります。たとえば、複数言語のインストールを作成している場合、インストールする言語を制御する言語トランスフォームを選択する必要があります。この特定の例で、言語 (つまりプロパティ値) は恐らくアドバンスト UI またはスイート/アドバンスト UI プロジェクトの各パッケージの言語と同じです。

アドバンスト UI プロジェクトでは、アドバンスト UI またはスイート/アドバンスト UI プロパティ ISSelectedLanguage は、アドバンスト UI またはスイート/アドバンスト UI インストールに使用する言語を識別します。リリースでエンド ユーザーが言語を選択できるように構成する場合、InstallationLanguage ウィザード ページがエンド ユーザーに対して言語のプロンプトを行います。インストールは ISSelectedLanguage プロパティをユーザーが選択した言語の 10 進数言語 ID を含む文字列に設定します。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる .msi パッケージを適切な言語で実行するには、各 .msi パッケージのコマンドライン設定に入力する値に以下を含みます:

TRANSFORMS="[ISPREREQDIR]¥[ISSelectedLanguage].mst"

.mst ファイルは、.msi パッケージと同じフォルダーに配置します。

基本の MSI .exe パッケージのコマンドライン設定に次の構文を使います:

/L[ISSelectedLanguage]

アドバンスト UI またはスイート/アドバンスト UI プロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストール内の複数のパッケージに異なる値を渡す

アドバンスト UI またはスイート/アドバンスト UI のプロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストール内の各パッケージまたは一部のパッケージに、異なるプロパティの値を渡すことができます。たとえば、エンド ユーザーが INSTALLDIR を各パッケージごとに異なる場所にオーバーライドできるようにしたい場合が考えられます。この例では、INSTALLDIR の値は各パッケージごとに異なります。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるパッケージの 1 つで、プロジェクトに 定義済み BrowseFolder ウィザード ページを追加する と、InstallShield は [パッケージ] ビューのインストール操作で、以下をパッケージのコマンドライン設定に追加します。

INSTALLDIR="[ISInstallDir_PackageDisplayName]"

適切な場合、そのコマンドラインをオーバーライドすることができます。パッケージが .msi パッケージである場合、そのコマンドラインはパッケージの INSTALLDIR 値をエンド ユーザーがそのパッケージの BrowseFolder ウィザード ページで指定したパスに設定します。

パッケージが基本の MSI または InstallScript MSI .exe パッケージの場合、デフォルトのコマンドラインを変更してこの構文を使用します:

/v"INSTALLDIR=¥"[ISInstallDir_PackageDisplayName]¥""

アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスを定義する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールのウィザード インターフェイス要素は、ウィザード ページおよび 2 番目のウィンドウ (ポップアップ ウィンドウとも呼ばれます) で構成されます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトには、標準の定義済みウィザード ページおよび 2 番目のウィンドウが多く含まれています。ページやウィンドウを追加および削除、またそのレイアウトや動作をカスタマイズすることができます。

アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーでは、アドバンスト UI またはスイート/アドバンスト UI インストールのウィザード インターフェイスが優先されるため、.msi、.msp、および InstallScript パッケージ フォーマットのユーザー インターフェイスは自動的に抑制されます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトに .exe パッケージが含まれている場合、そのユーザー インター

フェイスを手動で抑制し、アドバンスト UI またはスイート/アドバンスト UI インストールで、そのユーザー インターフェイスをサイレントで起動する必要があります。そうしなかった場合、インストールの実行時に、アドバンスト UI またはスイート/アドバンスト UI のウィザード ページと .exe パッケージのユーザー インターフェイスの 2 つの異なるユーザー インターフェイスが表示されることになります。詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスの定義についての詳細は、「ウィザード インターフェイスを使って作業する」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストールでハイコントラストモードをサポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンストUIまたはスイート/アドバンスト UI インストール中、各ウィザードページには、ウィザード アイコン、ウィザード キャプション、ヘッダー イメージ、ヘッダー テキスト、本文テキスト、ナビゲーション ボタン、ナビゲーション ボタン ラベルなどのコア要素が表示されます。ヘッダー テキスト、本文テキスト、およびナビゲーション ボタンのラベルの読みやすさは、ターゲット システムに設定されているコントラスト モードによって異なります。InstallShield は、一般的に、各ウィザード ページのヘッダー テキスト、本文テキスト、およびナビゲーション ボタンのラベルが、すべてのコントラスト モード、特にエンドユーザーのシステムで構成された高コントラスト モードではっきりと読み取り可能であるように構成されています。InstallShield は、ターゲットまたはエンド ユーザーのシステムで設定されたコントラスト モードにかかわらず、自動的にアクセシビリティと読みやすさを維持します。

一般的に、単色とグラデーションは、各ウィザード ページのヘッダーテキスト、本文テキスト、およびナビゲーション ボタン用に、**ウィザードインターフェイス** エクスプローラーのブラシを使用して最初に指定されますが、これらの定義された単色とグラデーションは、エンドユーザーのシステムで設定された高コントラスト モードの色によって上書きされます。InstallShield は、一貫したアクセシビリティを確保するために、この動作を完全にサポートしています。

アドバンスト UI またはスイート/アドバンスト UI インストールのエンドユーザー インターフェイスの定義についての詳細は、「ウィザード インターフェイスを使って作業する」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの UI で機能ステータスおよびその他の機能データを参照する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト インストールの実行時に、機能関連の情報に依存する特定の UI 動作をトリガする必要がある場合があります。この情報の使用例を次に示します:

- 特定の機能のインストールが選択されなかった場合に、特別な対応ウィザード ページを省略することができます。
- 特定の機能が既にインストールされている場合に、ウィザード ページのチェックボックスを無効にすることができます。
- 拡張条件 DLL を作成して、インストールが実行時に行う機能関連のチェックするカスタム条件を定義できます。
- ターゲット システム上で各機能が必要とする空き容量を表示するカスタム機能ツリー ウィザード ページを表示するように、インストールを構成できます。

アドバンスト UI またはスイート/アドバンスト UI インストールの実行中にチェックすることができる、様々な機能関連の疑似プロパティを以下に示します。いずれも、*名前*部分を機能の名前に置換します。

テーブル 4 ・ アドバンスト UI またはスイート/アドバンスト UI インストールの UI における機能関連の疑似プロパティ

疑似プロパティ	種類	説明
FEATURE[*name*].actionState	読み取り/書き込み	これは、指定された機能の次の動作を示します。次の値が可能です: ・ **インストール**—この状態は、機能のインストールが設定されていることを示します。この状態は機能が既にインストール済みでない場合のみ有効です。 ・ **削除**—この状態は、機能の削除が設定されていることを示します。この状態は機能が既にインストールされている場合のみ有効です。 ・ **ヌル (空白文字列)**—空白文字列の状態は動作を要求しません。これは常に有効です。
FEATURE[*name*].installState	読み取り専用	これは、指定された機能の現在の状態を示します。次の値が可能です: ・ **0**—機能は現在ターゲット システムに存在しません。 ・ **1**—機能の一部がターゲット システムに存在します。つまり、この機能の一部のパッケージが存在します。 ・ **2**—一部の機能の子機能のみがターゲット システムに存在します。 ・ **3**—機能が、現在ターゲット システムに存在します。
FEATURE[*name*].displayName	読み取り専用	これは、実行時に適切なウィザード ページ上で指定された機能に表示される名前を示します。
FEATURE[*name*].description	読み取り専用	これは、実行時にウィザード ページ上に表示される特定の機能の説明を示します。
FEATURE[*name*].cost	読み取り専用	これは、指定された機能がターゲット システム上で必要とする空き容量を示します。

読み取り/書き込み疑似プロパティ FEATURE[name].actionState は、例えば拡張条件やウィザード インターフェイスのコントロールの "動作" 設定を通して設定できます。

これらの任意の疑似プロパティは、拡張条件またはウィザード インターフェイスのコントロールの "表示" または "有効化" 設定のバインド条件を通して読み込むことができます。

アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域で、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む形式化された式を埋め込むことができます。実行時、インストールはこれらの式の値を拡張します。次の種類の形式化された式がサポートされています。

- **プロパティ**—プロパティの値を解決するには、プロパティ名を角括弧で囲みます。たとえば、**[MYPROPERTY]** は実行時に MYPROPERTY というプロパティの値に解決します。プロパティが定義されていない場合、この式は空白文字列に解決します。
- **再帰的なプロパティ**—プロパティの解決済みの値を解決するには、プロパティ名を追加の角括弧のセットで囲みます。たとえば **[[PROPERTY1]]** は、**PROPERTY1** に名前が格納されているプロパティの値に解決します。つまり、**PROPERTY1** に **PROPERTY2** という値が格納されている場合、**PROPERTY2** に格納されている値に解決されます。
- **エスケープされた文字**—式に特殊文字 (リテラル括弧、コンマ、または角括弧など) を含めるには、特殊文字の前に円記号を付けて、結果となる文字列を角括弧で囲みます。たとえば、**[¥[]Text[¥]]** と入力すると **[Text]** に置換されます。
- **ヌル文字**—式にヌル文字を埋め込むには、チルダを角括弧で囲む、つまり**[~]** を埋め込みます。
- **環境変数**—環境変数の値を使用するには、環境変数の名前の前にパーセント記号を付けて、結果文字列を角括弧で囲みます。たとえば、**[%PATH]** は PATH 環境変数の値として解決します。
- **オブジェクト式**—ターゲット システム上でファイル、レジストリ エントリ、オペレーティング システム、その他の項目についての情報を検索するには、オブジェクト式を使用します。これによって、ターゲット システム固有の条件に基づいて、実行時にアドバンスト UI またはスイート/アドバンスト UI の多くの設定をダイナミックに構成することができます。オブジェクト式は、規則 **[@Object(Parameters, ...).Property(Parameters, ...)]** を使用します。オブジェクト式の書き方についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する」を参照してください。

プロパティ式の中など、その他の形式化された式内にオブジェクト式を埋め込むことができます。オブジェクト式のパラメーター内に、プロパティ式およびその他の形式化された式を埋め込むこともできます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する」を参照してください。

構文エラー、オブジェクト式のオブジェクト エラー、および無効な形式化された式は、実行時に空白文字列として評価されます。一部の状況において、解析エラーまたは形式化された式の評価についての情報は、アドバンスト UI またはスイート/アドバンスト UI インストールのデバッグ ログ ファイルに記録されます。

形式化された式を使用できる、アドバンスト UI およびスイート/アドバンスト UI プロジェクトの領域

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な領域にある多くの設定で、形式化された式を使用できます。一部の種類のアクションでは、形式化された式を使用することができます。InstallShield の次の領域は、形式化された式を指定できる場所の例です。

- [パッケージ] ビュー: パッケージのファイルを起動するために使用するコマンドラインを入力する設定

  詳細については、「アドバンスト UI およびスイート/アドバンスト UI 形式の式を使用して、コマンドラインを動的に構成する」を参照してください。

- [プロパティ マネージャー] ビュー: プロパティの [値] 列[形式化済み] チェックボックスを確実に選択して、プロパティの値の形式化された式が実行時に解決されるようにします。実行時の解決処理は、インストール中に OnBegin イベントのアクションが実行される前、早い段階で発生します。

  詳細については、「[プロパティ マネージャー] ビュー」を参照してください。

- たとえば、[機能] ビューの機能の "表示名" 設定や、[ウィザード インターフェイス] ビューの UI コントロールの "テキスト" 設定など、文字列エントリを使用する設定。

- 終了条件、検出条件、インストール条件、その他の種類の条件を構成するためのテキストを入力することができるサブ設定の一部。たとえば、"ファイルの比較"、"レジストリの比較"、および "プロパティの比較" 条件確認の "比較対象" サブ設定内。

  詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

- プロパティを設定するアクションの [イベント] ビューの場合: "プロパティ値" 設定内"プロパティ値の形式化" 設定で [はい] を選択して、プロパティの値の形式化された式が実行時に解決されるようにします。

  詳細については、「スイート/アドバンスト UI インストールに含まれるプロパティを設定するアクションでの作業について」を参照してください。

- ISuiteExtension ポインターを受け取る任意のアクション: FormatProperty メソッドの bstrValue パラメーター

  詳細については、次を参照してください。

  - アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する
  - スイート/アドバンスト UI インストールに含まれるアクションの DLL ファイルでの作業について
  - スイート/アドバンスト UI インストールでマネージ コード アクションを使用する
  - スイート/アドバンスト UI インストールで .NET Core アクションを使用する

- InstallScript アクション コード: InstallScript 関数 **SuiteFormatString** の szValue パラメーター

  詳細については、「SuiteFormatString」を参照してください。

- PowerShell アクションの場合: format-suiteproperties cmdlet は形式化された式の値を展開します。

  詳細については、「スイート/アドバンスト UI インストールに含まれるアクションの PowerShell スクリプトでの作業について」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでオブジェクト式を書いて、ターゲット システムを検索する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な設定では、ファイル、レジストリ エントリ、その他のアイテムについての情報をターゲット システムでクエリするオブジェクト式を埋め込むことができます。オブジェクト式には、この規則を使用します:

[@Object(Parameters, ...).Property(Parameters, ...)]

各オブジェクト式には、オブジェクト固有プロパティの集まりである、オブジェクトへの参照が含まれています。オブジェクトとプロパティにはパラメーターを含めることができます。

たとえば、次の Platform オブジェクト式は、アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンのアーキテクチャ (x86、x64、IA64、ARM、または不明) を取得します:

[@Platform.Architecture]

スケジュール済みの処理が完了した後、次の Package オブジェクト式は、パッケージ GUID 0F283EC0-E9D1-4D18-801D-0014F6CA96DF を持つパッケージの終了コードを取得します。

[@Parcel(0F283EC0-E9D1-4D18-801D-0014F6CA96DF).ExitCode]

オブジェクト式書き込みについてのガイドライン

オブジェクト式を書き込む際には、次のガイドラインに留意してください:

- 構文エラー、オブジェクト式のオブジェクト エラー、および無効な形式化された式は、実行時に空白文字列として評価されます。一部の状況において、解析エラーおよび形式化された式の評価についての情報は、アドバンスト UI またはスイート/アドバンスト UI インストールのデバッグ ログ ファイルに記録されます。
- プロパティ式その他のオブジェクト式のような、別の形式化された式内にオブジェクト式を埋め込むことができます。

  次の式では、Registry オブジェクト式が File オブジェクト式にパラメーターの一部として埋め込まれています。

  [@File([@Registry(HKLM¥Software¥MyProduct).KeyValue(MyProductPath)]¥MyProduct.exe).Version]

  MyProduct.exe ファイルが **MyProductPath** 値データに指定された場所にあるとき、File オブジェクト式はファイルのバージョンを返します。ファイルがその場所に見つからなかった場合、またはレジストリ値が存在しなかった場合、File オブジェクト式は空白文字列を返します。

- オブジェクト式のパラメーター内に、プロパティ式およびその他の形式化された式を埋め込むことができます。

  次の File オブジェクト式は、32 ビットの ProgramFilesFolder にある MyProduct.exe と呼ばれるファイルの最終更新日時を取得します。(*false* のパラメーター値は、インストールが 64 ビットの場所を確認しないことを示します。)

  [@File([ProgramFilesFolder]MyCompany¥MyProduct.exe,false).Date(modified)]

- アドバンスト UI およびスイート/アドバンスト UI インストールの形式化された式のパーサーに、引用符は不要です。たとえば、オブジェクト式内のパラメーターに 1 つ以上のスペースが含まれるパスまたは別のアイテムが含まれる場合、そのパスを引用符で囲む必要はありません。
- オブジェクト式のパラメーター内にリテラル角括弧、丸括弧、またはコンマを含む必要がある場合、パラメーター内に特殊文字を含む値に解決する形式化されたプロパティ式 ([SpecialCharacterString])、またはエスケープ文字式 (A[¥,] B[¥,] および C) を埋め込みます。
- オブジェクト式を使ってシステム検索を行うことができます。ある検索の出力を、異なるオブジェクト式のパラメーターへのインプットとして使用することができます。
- オブジェクト式の実行時の解決には、2 つの異なる段階 (解析と評価) があります。実行時、任意の形式化された式が解析されるとき 、まず始めにオブジェクト式のパラメーターが解析されます。すべての形式化された式が解析されたあと、解析された出力が評価されます。その他の解析は、最初の解析が完了した後に始まります。

  したがって、式 [MYPROP] の値が [MYPROP] の場合、[MYPROP] がターゲット システム上で評価される解析された出力となります。この状況で、再帰ルックアップが発生することはありません。
- サンプル オブジェクト式、および使用可能な各オブジェクトについての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

特定の詳細の一部は、スイート/アドバンスト UI プロジェクト タイプにのみ適用します。これらの違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Windows Vista 以降およびユーザー アカウント制御 (UAC) の目的は、ユーザーが常に標準ユーザーとして実行できるようにすることです。昇格が必要になることは、ほとんどありませんが、必要となる場合、できるだけ短い時間に限られる必要があります。

InstallShield のいくつかの異なる領域にある設定は、アドバンスト UI またはスイート/アドバンスト UI インストールで昇格された権限の UAC の同意または資格情報プロンプトをトリガするかどうかに影響します。さらに、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの設定は、昇格された権限の UAC プロンプトをエンド ユーザーに表示するかどうかにも影響します。Windows Vista 以降のシステムおよび Windows Server 2008 以降のシステム上でエンド ユーザーがインストールを実行する際の UAC の動作を適切に定義するためには、これらの異なる設定をよく理解することが必要です。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおける UAC 関連の設定

アドバンスト UI またはスイート/アドバンスト プロジェクトの次の設定は、UAC プロンプトを表示するかどうかを決定するのに役立ちます:

- **必要実行レベル**—[リリース] ビューにあるこの設定を使用して、インストールの Setup.exe ファイルが必要とする最小実行レベルを指定します。InstallShield は、Setup.exe ランチャーに埋め込まれるアプリケーション マニフェストで選択した値 (管理者、最高権限、または起動者) を使用します。詳細については、「Windows Vista 以降のプラットフォームでのセットアップランチャーの必要実行レベルを指定する」を参照してください。
- **昇格された権限が必要**—[パッケージ] ビューのパッケージでこの設定を使って、パッケージに昇格されたシステム権限が必要かどうかを指定します。
- **管理者権限が必要**—この設定を使って、[イベント] ビューのプロジェクトに追加したアクションが管理者権限を必要とするかどうかを指定します。

***プロジェクト・**スイート/アドバンスト UI プロジェクトには [イベント] ビューが含まれていて、アクションをイベントに割り当てることができます。*

アドバンスト UI またはスイート/アドバンスト UI インストールに含まれているパッケージの UAC 関連の設定

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるパッケージで、昇格された権限のための UAC プロンプトをトリガする様々な方法があります。詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールの UAC 関連のランタイム動作

Windows Vista 以降および Windows Server 2008 以降における、次の UAC 関連の動作にご注意ください:

- ″必要実行レベル″ が [起動者] に設定されている場合で、プロジェクト内のアクションには管理者権限が不要、またインストール内のパッケージにも管理者権限が不要なとき、インストール中にエンド ユーザーに UAC プロンプトが表示されることはありません。インストール全体 (UI の表示、アクションの起動、およびパッケージの実行) は、昇格されないプロセスで実行します。
- ″必要実行レベル″ が [起動者] に設定されている場合で、プロジェクト内のアクションには管理者権限が不要、インストール内の 1 つ以上のパッケージで管理者権限が必要な場合、インストール中エンド ユーザーに UAC プロンプトが 1 回表示されます (さらに、再起動の度に追加の UAC プロンプトが表示されます)。UAC

プロンプトは、エンド ユーザーが UI 上でインストール ボタンをクリックした後に表示されます。"昇格された権限が必要" に [はい] が設定されているパッケージは、新しい個別のプロセスで昇格された権限で起動されます。"昇格された権限が必要" に [いいえ] が設定されている、UI のその他の部分、アクション、およびパッケージは、元の昇格されないプロセスで起動されます。

- "必要実行レベル" が [起動者] に設定されている場合で、プロジェクト内の 1 つ以上のアクションで管理者権限が必要な場合、インストール中エンド ユーザーに UAC プロンプト 1が 回表示されます (さらに、再起動の度に追加の UAC プロンプトが表示されます)。UAC プロンプトのタイミングは、昇格が最初に必要な時によって異なります。

  たとえば、一番最初にスケジュールされた管理者権限が必要なアクションが OnBegin イベント中に発生するように構成されている場合 (パッケージが実行される前)、エンド ユーザーが Setup.exe ファイルを起動したすぐ後に UAC プロンプトが表示されます。エンド ユーザーが同意または認証情報を入力した後、インストールは新しい昇格されたプロセスでこのアクション、およびそれに続く昇格が必要なアクションとパッケージを起動します。後に続く昇格が不要なアクションとパッケージは、UI の残りの部分と共に、元の昇格されないプロセスで起動されます。

  ただし、最初にスケジュールされている管理者権限が必要なアクションが、権限の昇格が必要なパッケージの後に発生するように構成されている場合、エンド ユーザーが UI 上でインストール ボタンをクリックした後に UAC プロンプトが表示されます。エンド ユーザーが同意または認証情報を入力した後、インストールは新しい昇格されたプロセスでパッケージ、およびそれに続く昇格が必要なアクションとパッケージを起動します。後に続く昇格が不要なアクションとパッケージは、UI の残りの部分と共に、元の昇格されないプロセスで起動されます。

アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield はいくつかの方法を使って、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージを実行した後にターゲット システムを再起動するかどうかを判別します。

.exe パッケージにおけるレジストリ キーの変更

アドバンスト UI またはスイート/アドバンスト UI に含まれる .exe パッケージが実行された後に次のレジストリ キーのどれかが変更されたとき、ターゲットマシンの再起動が必要な場合があります。

- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnce

- HKEY_CURRENT_USER¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- (64 ビット) HKEY_LOCAL_MACHINE¥Software¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx
- (64 ビット) HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx
- HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥Session Manager、PendingFileRenameOperations の値

レジストリキーは パッケージが実行された後に考慮されます。これらの数字が異なる場合、ファイルがシステムの再起動を試みると同時にインストールを終了するものと見なされます。

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに指定されたリターン コード

アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの終了コードが、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューにある "再起動コード" 設定でパッケージの適切な操作の種類 (インストール、削除、修復、または変更) に指定されたリターン コードと一致する場合、ターゲット システムの再起動が必要な場合があります。

リターンコード 1641 および 3010

アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージで実行される .msi ファイルに再起動ステートが戻された場合、ターゲット システムの再起動が必要な場合があります。標準 Windows Installer の再起動リターンコードは以下の通りです。

- ERROR_SUCCESS_REBOOT_INITIATED (1641)
- ERROR_SUCCESS_REBOOT_REQUIRED (3010)

パッケージに指定された動作

アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューにあるパッケージの "再起動の要求" 設定を使って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに前述の条件が当てはまる場合、または当てはまらない場合の動作を指定できます。各操作の種類 (インストール、削除、修復、および変更) に対して異なる動作を指定できます。

詳細については、「再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する」を参照してください。

再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで新しいパッケージを構成するとき、または既存パッケージの設定を変更するとき、ターゲット システムの再起動が必要と見られるときに、アドバンスト UI またはスイート/アドバンスト UI インストールが行う処理を指定できます。たとえば、ターゲット システムを再起動する前にインストールがエンド ユーザーに対してプロンプトを行う場合、または再起動そのものをスキップする場合が考えられます。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューでは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる各パッケージに対して、また各操作の種類 (インストール、削除、修復、または変更) に対して適切な対応を行うために必要な柔軟性が提供されています。

タスク

ターゲット システムの再起動を必要とするアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定するには、次の手順に従います:

1. 再起動動作を構成したいパッケージを含むアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。
3. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
4. 適切な操作 (**インストール**、**削除**、**修復**、または**変更**) の下にある **"再起動の要求"** 設定で適切なオプションを選択します。

次のテーブルは、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージが行う様々な再起動関連の動作のための適切なオプションについて説明します。

テーブル 5 · アドバンスト UI またはスイート/アドバンスト UI パッケージの再起動動作

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージの動作	"再起動の要求" 設定のガイドライン
エンド ユーザーにプロンプトを表示して、エンド ユーザーの応答に従ってターゲット システムを再起動します	"再起動の要求" 設定で以下のオプションから 1 つを選択することを考慮します: • **コンピューターの再起動を許可する**—このオプションは、ほとんどの状況で使用できます。 • **常にコンピューターを再起動する**—このオプションは、エンド ユーザーがターゲット マシンを再起動しないオプションを選択している場合に、エンド ユーザーによるパッケージのプロンプトに対する応答と競合する可能性があります。この状況が発生する恐れがある場合に、常にシステムを再起動させるためには、パッケージのプロンプトを抑制して、パッケージの "再起動の要求" 設定で適切なオプションを選択してください。

テーブル 5 • アドバンスト UI またはスイート/アドバンスト UI パッケージの再起動動作 (続き.)

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージの動作	"再起動の要求" 設定のガイドライン
常に再起動をトリガする。エンドユーザーにプロンプトを行う、または行わない。	"再起動の要求" 設定で以下のオプションから 1 つを選択することを考慮します: • **コンピューターの再起動を許可する**—このオプションは、ほとんどの状況で使用できます。 • **常にコンピューターを再起動する**—このオプションは、エンドユーザーがターゲット マシンを再起動しないオプションを選択している場合に、エンド ユーザーによるパッケージのプロンプトに対する応答と競合する可能性があります。この状況が発生する恐れがある場合に、常にシステムを再起動させるためには、パッケージのプロンプトを抑制して、パッケージの "再起動の要求" 設定で適切なオプションを選択してください。
リターン コードまたはレジストリを使って、再起動が必要であることを示す。エンド ユーザーにプロンプトを行わない。	"再起動の要求" 設定で以下のオプションから 1 つを選択することを考慮します: • **再起動の要求を遅延して、Setup.exe を終了する、またはコンピューターを再起動する**—後続のパッケージのためにターゲットシステムを再起動する必要がある場合は、このオプションを選択します。 • **常に再起動の要求を表示して、Setup.exe を終了する、またはコンピューターを再起動する**—ターゲット システムを即時に再起動する必要がある場合は、このオプションを選択します。 • **再起動の要求を無視する**—パッケージの再起動要求が不要な場合は、この無視オプションを選択します。 • **再起動の要求を遅延して、Setup.exe を終了する、またはコンピューターを再起動する**—後続のパッケージに再起動が不要な場合は、このオプションを選択します。アドバンスト UI またはスイート/アドバンスト UI インストールに含まれる 1 つ以上のパッケージで再起動が必要な場合、このオプションを使用すると再起動の回数を最小限に抑えることができます。 後続のパッケージを実行する前に再起動が必要な場合で、そのパッケージの実行前に必ず再起動が行われる保証がない場合、この遅延オプションの選択は避けてください。
再起動を必要とするが、エンド ユーザーにプロンプトを表示せず、再起動の必要性も表示しない。	"再起動の要求" 設定で以下のオプションの選択を考慮してください: • **常に再起動の要求を表示して、Setup.exe を終了する、またはコンピューターを再起動する**—再起動が必要な場合は、このオプションを選択します。

アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI インストールがインストール モードまたはメンテナンス モードのどちらで実行されているかによって、インストールは Web サイトでホストされているアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーが更新されているかどうかを確認し、それをダウンロードするか、エンド ユーザーにダウンロードのオプションを提供します:

- インストールが [インストール] モードで実行中の場合、Web サイトでホストされているアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーが更新されているかどうかを自動的に確認します。アップデートが利用可能な場合、インストールはそれをダウンロードしてから起動します。
- インストールが [メンテナンス] モードで実行中の場合、Web サイトでホストされているアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーが更新されているかどうかを確認します。アップデートが利用可能な場合、インストールで表示されるメンテナンス ウィザードに **[アップデート]** ボタンが含まれます。エンド ユーザーがこのボタンをクリックすると、スイート エンジンがアップデートをダウンロードして起動します。

アプリケーションがターゲット マシン上に配置されたとき、アプリケーションのアップデートを定期的に確認するパラメーターをビルド前に構成する方法についての詳細は、「アプリのアップデートを確認する」を参照してください。

ダウンロード可能なアップデートをサポートするための要件と推奨事項

アドバンスト UI またはスイート/アドバンスト UI インストールにダウンロード可能なアップデートのサポートを追加して、アドバンスト UI またはスイート/アドバンスト UI プロジェクト用に、ベースのセットアップ ランチャーとアップデート用セットアップ ランチャーを作成する場合、次のガイドラインが推奨されます。

- isupdate.xml という名前のメタデータ ファイルをホストするのに使用する URL を決定します。

  ビルド時に InstallShield は isupdate.xml ファイルを作成して、プロジェクトのビルド フォルダー内の UpdateMetadata サブフォルダーに配置します。この isupdate.xml ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるバージョンとそのダウンロード場所を識別します。このファイルをサイトにアップロードする前に、必要に応じて手動でアップデートすることができます。

  ベースのアドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート URL パスを指定する必要があります。

- アップデートのリリース準備ができたら、isupdate.xml とアップデートをサイトにアップロードします。

- 最低限として、アップデート セットアップ ランチャーは、デジタル署名が必要です。エンドユーザー側での使用感を考慮した場合、アップデート セットアップ ランチャーとベース セットアップ ランチャーに同一のデジタル署名を使用することをお勧めします。
- アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、アップデート セットアップ ランチャーの構成を行うとき、パッケージのランタイムの場所は、セットアップ ランチャーからの抽出か、Web からのダウンロードである必要があります。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。

  ベース セットアップ ランチャーは、ソース メディアに格納されているパッケージを使用することができます。

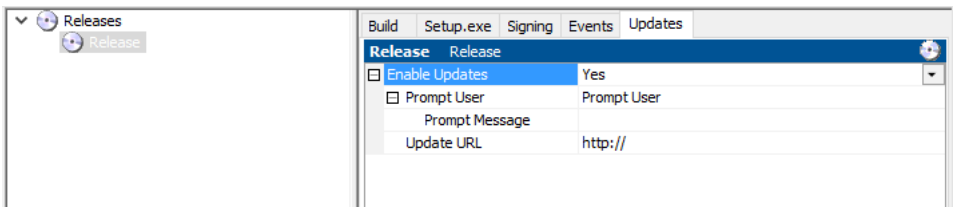
  アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する方法についての詳細は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する」を参照してください。

  アドバンスト UI またはスイート/アドバンスト UI プロジェクトでは、すべてのパッケージのランタイムの場所を上書きできます。詳細については、「リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する」を参照してください。
- アドバンスト UI またはスイート/アドバンスト UI インストールには、[プログラムの追加と削除] エントリが必ず必要です。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールの [プログラムの追加または削除] エントリ」を参照してください。
- アップデート アドバンスト UI またはスイート/アドバンスト UI プロジェクトのスイート GUID は、ベース プロジェクトのスイート GUID と一致しなくてはなりません。この設定は [一般情報] ビューにあります。

ダウンロード可能なアップデートのサポートを有効化する

タスク ***ダウンロード可能なアップデートのサポートをベースのアドバンスト UI またはスイート プロジェクトで有効化するには、以下の手順に従います:***

1. 将来的にダウンロード可能なアップデートを有効にするアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. 次いで、**[アップデート]** タブの **"アップデートの有効化"** 設定で、適切な値を選択します。
   - **いいえ** – ダウンロード可能なアップデートを無効化します。
   - **はい** – ダウンロード可能なアップデートを有効化します。

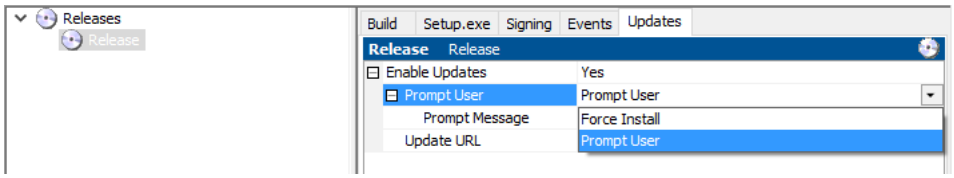


ダウンロード可能なアップデートのより新しいバージョンを構成する

タスク

ダウンロード可能なアップデートのサポートをベースのアドバンスト UI またはスイート プロジェクトで構成するには、以下の手順に従います:

1. アドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. 次いで、**[アップデート]** タブの **"ユーザーに確認"** 設定で、適切な値を選択します。
   - **強制インストール** – エンド ユーザーはスイート パッケージ (使用可能な場合) の新しいバージョンを強制的にダウンロードおよびインストールすることができます。
   - **ユーザーに確認する** – エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するかどちらかのオプションを受け取ります。

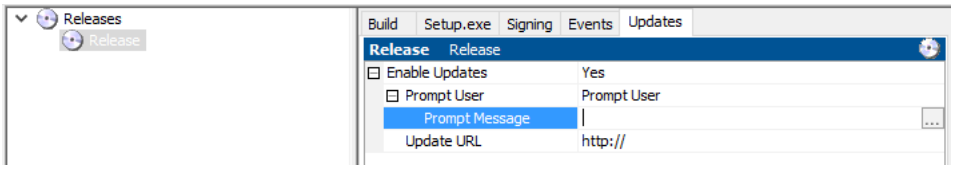


確認メッセージを構成する

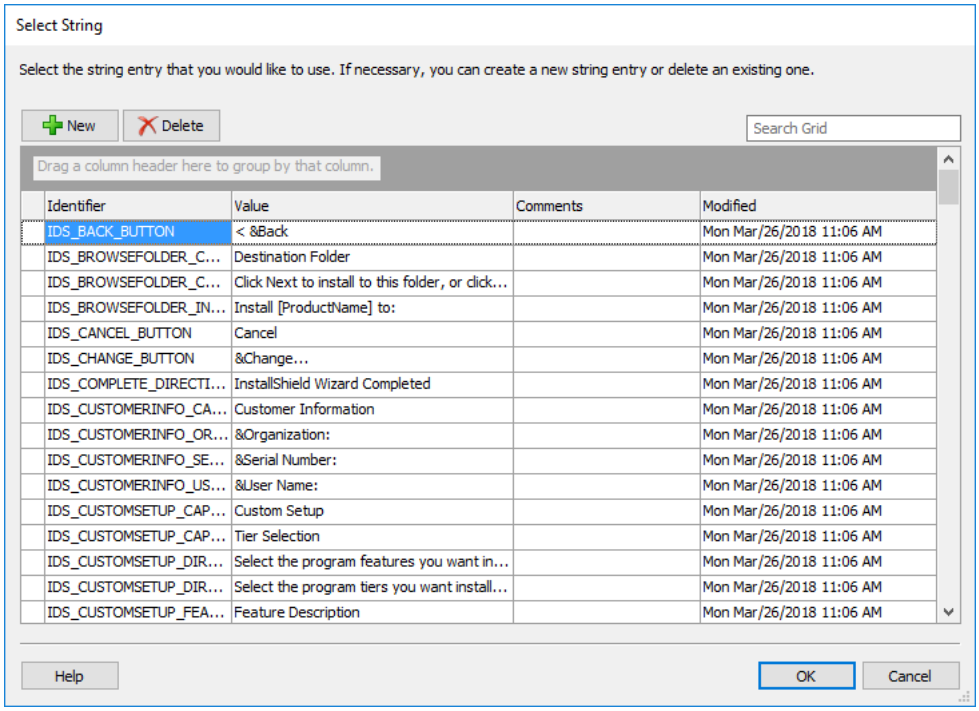
タスク

ベースのアドバンスト UI またはスイート プロジェクトにより新しいバージョンの確認メッセージを構成するには、以下の手順に従います:

1. **[アップデート]** タブにある **"確認メッセージ"** 設定で、
   - エンド ユーザーに表示するメッセージを入力するか、
   - 省略記号ボタン (...) をクリックして、新しいメッセージを作成、または文字列一覧からローカライズ済みのメッセージを選択します。



2. **[アップデート]** タブにある **"確認メッセージ"** 設定で、**[文字列の選択]** ダイアログを表示する省略記号ボタン (...) をクリックします。
3. **[文字列の選択]** ダイアログで **[新規作成]** をクリックして新しい文字列を作成してから **[OK]** をクリックします。



***メモ・** より新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するための [はい] または [いいえ] オプションを表示する確認メッセージがエンド ユーザーに表示されます。*

ダウンロード可能なアップデートのサポートを追加する

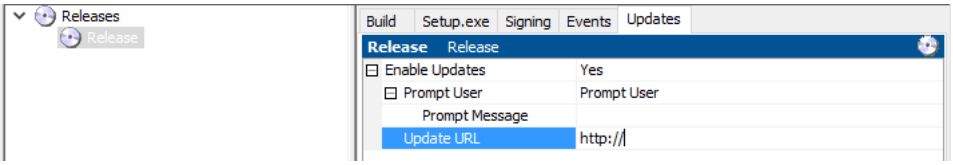
タスク ***ダウンロード可能なアップデートのサポートをベースのアドバンスト UI またはスイート プロジェクトに追加するには、以下の手順に従います:***

1. 将来的にダウンロード可能なアップデートを有効にするアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. **[アップデート]** タブにある **"アップデート URL"** 設定で、ターゲット システムでダウンロード可能にするアップデート セットアップ ランチャーのパスとして今後使用する絶対パス URL (**http://** または **https:/** で始まります) を入力します。

   isupdate.xml を "アップデート URL" 設定で指定されているのと同じパスにアップロードする必要があります。たとえば、"アップデート URL" 設定の値が **http://MyWebSite.com/path/setup.exe** の場合、メタデータの URL は **http://MyWebSite.com/path/isupdate.xml** となります。

   "アップデート URL" 設定で指定されているセットアップ ランチャーの場所は、isupdate.xml ファイルに適切なパスを指定する方法でオーバーライドすることができます。ただし、isupdate.xml ファイルが "アップデート URL" で指定されている .exe ファイルと同じフォルダーに存在しなくてはなりません。そうでない場合、エンドユーザーがダウンロード可能なアップデートを使用することはできません。

5. **[署名]** タブでは、デジタル署名情報を入力して、Setup.exe ランチャーが署名されるようにリリースを構成することをお勧めします。



セットアップランチャーを作成する

タスク

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのアップデート セットアップ ランチャーを作成するには、以下の手順に従います:

1. 更新するアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開き、必要に応じて、更新します。
2. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
3. **[リリース]** エクスプローラーで、構成するリリースを選択します。
4. **[署名]** タブで、デジタル署名情報を入力して、Setup.exe ランチャーが署名されるようにリリースを構成します。ここでは、ベースの Setup.exe ランチャーの署名に使用したのと同じデジタル署名を使用することをお勧めします。
5. リリースをビルドします。

   InstallShield は isupdate.xml ファイルを作成して、プロジェクトのビルド フォルダー内の UpdateMetadata サブフォルダーに配置します。XML ファイルには、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるバージョンの場所を識別する次のような要素が含まれています:

   <Update Url="http://www.mysite.com/MyProduct/Setup.exe" Version="1.00.0000" Id="{8E23F36C-5EA8-475D-8EAE-09FA36103975}"/>

   このファイルをサイトにアップロードする前に、必要に応じて編集することができます。

   たとえば、起動後にアップデート セットアップ ランチャーへのパスを変更する場合、Url 属性の値を変更してから指定の場所に isupdate.xml ファイルを再びアップロードします。

   同じ isupdate.xml を使って複数製品のアップグレードを定義したい場合、各製品の Update 要素をそのファイルに追加することができます。
6. isupdate.xml ファイルとアップデート セットアップ ランチャーをサイトにアップロードします。

ダウンロード可能なアップデートの実行時の動作

ダウンロード可能なアップデートをサポートするインストールの実行時の動作は、インストールが実行中のモードによって異なります。

インストール モードの動作

ベースのアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーがインストール モードで実行するとき、セットアップ ランチャーはアップデート URL パスにある isupdate.xml ファイルを確認します。

1. isupdate.xml ファイルが存在し、指定のパスにある新しいバージョンのインストールが参照されている場合、ベース アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーはそれを自動的にダウンロードしてデジタル署名を検証します。
2. isupdate.xmlが新しいバージョンのインストールを参照しない場合、またはisupdate.xml が存在しない場合、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーはベース プロジェクトで構成されているアップデート URL パスにダウンロードがあるかどうかを確認します。ダウンロード可能なアイテムが存在する場合、自動的にダウンロードが実行され、デジタル署名が確認されます。
3. ベース プロジェクトで構成されているアップデート URL パスにダウンロードが存在しない場合、ベース アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーは、アップデートをダウンロードしないまま続行します。

メンテナンス モードの動作

アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーがメンテナンス モードで実行するとき、アドバンスト UI または スイート/アドバンスト UI セットアップ ランチャーはアップデート URL パスにある isupdate.xml ファイルを確認します。

1. isupdate.xml ファイルが存在し、指定のパスにある新しいバージョンのインストールを参照する場合、インストールは ISUpdateAvailable プロパティを 1 に設定して、[アップデート] ボタンが含まれまれている MaintenanceUpdateWelcome ウィザード ページを表示します。また、インストールは ISUpdateVersion プロパティをダウンロード可能なスイート/アドバンスト UI またはアドバンスト UI インストールのバージョンと等しく設定します。

   エンド ユーザーが [アップデート] ボタンをクリックすると、セットアップ ランチャーがそれをダウンロードしてからデジタル署名を検証します。
2. isupdate.xml ファイルにインストールの新しいバージョンへの参照が含まれていない場合、またはisupdate.xml ファイルが存在しない場合、インストールは [アップデート] ボタンを含まない標準のメンテナンス ウィザード ページ、MaintenanceWelcome を表示します。

デジタル署名の検証

アドバンスト UI または スイート/アドバンスト UI インストールがアップデートをダウンロードした場合、それを起動する前にデジタル署名が検証されます。

1. アップデート セットアップ ランチャーのデジタル署名がベースのセットアップ ランチャの署名と一致すると、アップデート セットアップ ランチャーが自動的に実行されます。
2. デジタル署名が一致しなかった場合、または、ベース セットアップ ランチャーがデジタル署名されていない場合、エンドユーザーがアップデート セットアップ ランチャーの実行を続けるかどうかを選択できるセキュリティ警告が表示されます。
3. アップデート セットアップ ランチャーにデジタル署名がない場合、インストールは失敗し、デバッグ ログによって、デジタル署名問題が報告されます。

［新しいアップデート利用可能］ダイアログのカスタマイズ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

［新しいアップデート利用可能］ダイアログ ボックスは、アプリケーションのアップデートまたは新しいバージョンがターゲット システムでダウンロード可能になったときに表示される実行時のインターフェイスです。このダイアログ ボックスは通常、**［メディア］**ビューの **［リリース］**エクスプローラの **［アップデート］**タブにある **［アプリのアップデートを確認する］**領域で指定されたアップデート構成に基づいて、スケジュールどおりにトリガされます。**［新しいアップデート利用可能］**ダイアログ ボックスには、通常、次の内容が表示されます:

- アップデートが利用可能であることを示すメッセージ。
- **［後で通知］**、**［このアップデートをスキップ］**、**［今すぐダウンロード］**などのアクション ボタン。

アドバンスト UI またはスイート/アドバンスト UI インストーラのインストール中に表示される **［新しいアップデート利用可能］**ダイアログ ボックスを設定するには、isupdate.xml ファイルに以下のプロパティを必要な値とともに追加し、更新した XML ファイルを指定の場所に再アップロードします:

- DialogBackgroundColor— このプロパティを使って、**［新しいアップデート利用可能］**ダイアログ ボックスの背景色を指定します。
- DialogHeaderFontSize — このプロパティを使って、**［新しいアップデート利用可能］**ダイアログ ボックスのダイアログ ヘッダーのフォントサイズを定義します。読みやすくするため、28 以上のサイズを定義することが推奨されます。
- DialogBodyFontSize— このプロパティを使って、**［新しいアップデート利用可能］**ダイアログ ボックスのダイアログ テキストおよびボタン テキストのフォント サイズを定義します。読みやすくするため、18 以上のサイズを定義することが推奨されます。
- DialogTextColor— このプロパティを使って、**［新しいアップデート利用可能］**ダイアログ ボックスのダイアログ テキスト、ボタン テキスト、ダイアログ ヘッダーの色を指定します。

色の値は、0xRRGGBBのような 16 進数形式で定義する必要があります:

- 0x は後に続く文字が 16 進数であることを示します。
- RR、GG、および BB は 00 から FF までの範囲で、それぞれ赤、緑、および青を意味します。

  例:

  - 0xFF0000 は赤を意味します。
  - 0x00FF00 は緑を意味します。
  - 0x0000FF は青を意味します。
  - 0xFFFFFF は白を意味します。

- 0x000000 は黒を意味します。

以下の例に示すように、それぞれの値を持つプロパティを isupdate.xml ファイルに追加することができます:

<Update Url="http://www.mysite.com/MyProduct/Setup.exe" Version="1.00.0000" Id="{8E23F36C-5EA8-475D-8EAE-09FA36103975}"DialogBackgroundColor="0xFF0000" DialogHeaderFontSize="38" DialogBodyFontSize="24" DialogTextColor="0xFFFFFF"/>

Windows タスク スケジューラでのアプリケーション更新スケジュール タスクの管理

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI のインストーラが実行されると、通常、ターゲット システム上の Windows タスク スケジューラにスケジュールタスクが作成されます。これらのタスクは、アプリケーションの更新を管理するように設定され、インストールされたアプリケーションが定期的に更新をチェックして適用できるようにします。これらのタスクの頻度と動作は、**[アップデート]** タブの **[アプリのアップデートを確認する]** 領域で定義した設定によって決まります。

ただし、インストール中にアプリケーション更新のスケジュール タスクの作成をスキップしたい場合は、インストール ダイアログに表示されるチェック ボックスを選択できます。このタイプのチェック ボックスでは、最初に、**[ユーザー インターフェイス]** ビューの **[ウィザード インターフェイ スエクスプローラ]** の任意のウィザード ページの下にチェックボックス コントロールを追加し、ISSKIPAPPUPDATES プロパティを使用して構成する必要があります。以下の手順では、プロジェクトに必要なチェックボックス コントロールを追加して構成する方法について説明します。

タスク

インストール中にアプリケーションの更新をスキップするチェックボックス コントロールをプロジェクトに追加して設定するには、以下の手順に従います:

1. **[動作とロジック]** の下のビュー リストで、**[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンをクリックしてプロパティを作成し、**[名前]** 列に **ISSKIPAPPUPDATES** と入力します。
3. **[値]** 列に、新しいプロパティの値を入力します。デフォルト値は **False** です。
4. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。
5. **[ウィザード インターフェイス]** エクスプローラで、**[ウィザード ページ]** ノードを展開し、インストール中にチェックボックスを表示するウィザード ページ(**InstallationWelcome** ウィザード ページなど)の下にチェックボックス・コントロールを追加します。

6. チェックボックス コントロールの **"プロパティ"** 設定を **ISSKIPAPPUPDATES=true** のみに設定します。
7. チェックボックス コントロールの **"テキスト"** 設定を設定して、インストール ダイアログのチェックボックスに必要なテキストを表示します。たとえば、この設定を **[インストール中に更新をスキップする]** に設定できます。
8. チェックボックスコントロールの **"幅"** 設定を必要に応じて設定し、インストール ダイアログで定義済みのテキストを含むチェックボックスを見やすくします。たとえば、この設定を **200** に設定できます。

インストール プロセス中、**"テキスト"** 設定で指定した定義済みテキストのチェックボックスが定義済みインストール ダイアログに表示されます。このチェック ボックスを選択すると、ターゲット システムへのインストール中に Windows タスク スケジューラでアプリケーション更新のスケジュール タスクの作成を妨げるまたはスキップすることができます。逆に、このチェックボックスの選択を解除すると、これらのタスクの作成が許可されます。デフォルトで、このチェック ボックスは選択されていません。

メモ: *ターゲットシステムへのインストール中に Windows タスクス ケジューラでアプリケーション更新のすべてのスケジュール済みタスクの作成をスキップするには、ターゲットシステムでインストーラを実行する前に、ISSKIPAPPUPDATES プロパティを使用してチェックボック スコントロールを構成する必要があります。*

アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

InstallShield では、アドバンスト UI またはスイート/アドバンスト UI インストールのセットアップ起動プログラムをパスワードで保護することができます。セットアップ起動プログラムをパスワードで保護すると、インストールを実行するエンド ユーザーは、大文字小文字の区別があるパスワードを入力する必要があります。

タスク

アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護するには、以下の手順に従います:

1. セットアップ起動プログラムにパスワード保護を追加します:
   a. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
   b. **[リリース]** エクスプローラーで、構成するリリースを選択します。
   c. **Setup.exe** タブをクリックします。セットアップ起動プログラムの設定が表示されます。

d. **"ランチャーをパスワードで保護"** 設定で、**[はい]** を選択します。

e. **"ランチャーのパスワード"** 設定に、エンド ユーザーが指定しなくてはならない、大文字と小文字を区別するパスワードを指定します。

2. オプションで、エンド ユーザーがパスワードを入力できる定義済みのウィザード ページを追加することもできます。

a. **[ユーザー インターフェイス]** の下のビュー リストにある **[ウィザード インターフェイス]** をクリックします。

b. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。**ユーザー インターフェイス ウィザード**が開きます。.

c. **[タスク ページ]** リストで、**[インストールのパスワードを入力]** を選択します。

d. **[後続ページ]** リストを使って、ウィザード ページ シーケンスでデータベース サーバー ウィザード ページをスケジュールする場所を指定します。実行時に、このリストで選択したページが、新しいパスワード ウィザード ページの直後に表示されます。

***メモ・** 定義済みのパスワード ウィザード ページをプロジェクトに追加すると、InstallShield によってプレースホルダー イメージ コントロールが追加され、ページにロック イメージを表示することができます。コントロールの "リソース" 設定を構成してこのコントロールと共に表示するファイルを指定するか、イメージを除外する場合は、イメージ コントロールを削除します。*

デフォルトで、定義済みパスワード ウィザード ページは、ISPassword プロパティを使って、エンド ユーザーが入力したパスワード値を格納します。アドバンスト UI またはスイート/アドバンスト UI インストールは、ISPassword プロパティの値を Setup.xml ファイルのパスワードに格納されているパスワードと比較します。エンド ユーザーが入力したパスワードとプロジェクトで構成されたパスワードとが一致した場合、インストールが次のウィザード ページに進みます。

サイレント アドバンスト UI および スイート/アドバンスト UI インストールの場合、エンド ユーザーは /password コマンドライン パラメーターを使ってコマンドラインからパスワードを指定することができます。詳細については、「アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター」を参照してください。

アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

以下は、アドバンスト UI またはスイート/アドバンスト UI インストールに関する問題を解決するときに役立つ背景情報です。

アドバンスト UI またはスイート/アドバンスト UI インストールを、メンテナンス モードで実行される可能性がある理由

アドバンスト UI またはスイート/アドバンスト UI インストールは、次のシナリオで、メンテナンス モードで実行されることがあります:

- エンドユーザーが、[プログラムの追加と削除] から製品を修正する選択をしたとき。
- エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールを再実行したとき。
- アドバンスト UI またはスイート/アドバンスト UI インストール内に、検出条件が適切に構成されていないプライマリ パッケージが 1 つ以上存在するとき。アドバンスト UI またはスイート/アドバンスト UI パッケージにおける条件の構成については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」をご覧ください。
- アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージが、依存パッケージではなく、プライマリ パッケージとして間違って構成され、かつ、パッケージが既にターゲット システムにインストールされているとき。

  InstallShield は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるすべてのプライマリ パッケージの検出条件に基づいて、メンテナンス モード条件を自動的に作成します。アドバンスト UI またはスイート/アドバンスト UI エンジンは、これらの条件を使って、アドバンスト UI またはスイート/アドバンスト UI インストールがメンテナンス モードで実行するかどうかを判断します。検出条件が True に評価されたプライマリ パッケージが 1 つ以上存在する場合 (つまり、既にインストールされているように見えるプライマリ パッケージが最低 1 つ存在する場合)、メンテナンス モード条件は True として評価され、アドバンスト UI またはスイート/アドバンスト UI インストールはメンテナンス モードで実行されます。

  [パッケージ] ビューの "パッケージの種類" 設定を利用して、パッケージがプライマリ パッケージか依存パッケージかを指定できます。

  モード条件に関する詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする」をご覧ください。

トラブルシューティングのためのアドバンスト UI またはスイート/アドバンスト UI インストールのログ

アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティングは、まずデバッグ ログ ファイルの検証から始めます。デバッグ ログ ファイルでは、実行時にアドバンスト UI またはスイート/アドバンスト UI インストールの操作がどのように展開されたかについての詳しい情報が提供されています。デバッグ ログ ファイルを作成するには、/debuglog コマンドライン パラメーターを使って、コマンドラインからアドバンスト UI またはスイート/アドバンスト UI インストールを実行します。Setup.exe ファイルと同じディレクトリで InstallShield.log という名前のログ ファイルを生成するには、コマンドライン パラメーターのみを渡します。Setup.exe ファイルが読み取り専用の場所にあるとき、この処理は実行できません。例:

```
Setup.exe /debuglog
```

ログ ファイルの名前と場所を指定するには、パスと名前を次の要領で渡します。

```
Setup.exe /debuglog"C:¥PathToLog¥setupexe.log"
```

アドバンスト UI またはスイート/アドバンスト UI のデバッグ ログ ファイルには、次の情報が含まれています:

- アドバンスト UI またはスイート/アドバンスト UI エンジンからの初期化情報
- プロパティの値とプロパティの値の変更
- 機能のアクションとインストール状態に関する情報
- パッケージの検出状態と適格性に関する情報
- パッケージのコマンドラインと起動の結果情報
- InstallScript パッケージの場合—InstallScript 関数 SuiteLogInfo によってログ記録された情報

アドバンスト UI またはスイート/アドバンスト UI プロジェクトのデフォルトのアドバンスト UI またはスイート/アドバンスト UI のユーザー インターフェイスでも、情報はデバッグ ログ ファイルにログされますが、一般的に、これらの情報は無視することができます。(ログの UI 情報は通常 UI DLL: というプレフィックスがついています。)

機能、パッケージの状態、および、アドバンスト UI またはスイート/アドバンスト UI のモード情報がログ ファイルに書き込まれるとき、状態の数値のみ書き込まれます。次のテーブルは、数値とそれが表す状態の説明です。

テーブル 6 ・ 機能の操作ステータス

ログの値	状態
0	機能に対して発生する操作はありません。
1	機能がインストールされます。
2	機能が削除されます。

テーブル 7 ・ 機能のインストール状態

ログの値	状態
0	機能は現在インストールされていません。
3	機能はインストールされています。

テーブル 8 ・ パッケージ状態

ログの値	インストール状態	操作状態
0	パッケージはインストールされていません。	—
1	パッケージは現在インストールされています。	インストール操作が実行されます。
2	—	変更操作が実行されます。

テーブル 8 • パッケージ状態 (続き.)

ログの値	インストール状態	操作状態
3	—	修復操作が実行されます。
4	—	削除操作が実行されます。
5	—	パッケージに対して発生する操作はありません。

テーブル 9 • アドバンスト UI またはスイート/アドバンスト UI インストールのモード

ログの値	インストール モード
0	初回インストール
1	メンテナンス/UI メンテナンス モードの選択肢
2	メンテナンス/変更
3	メンテナンス/削除
4	メンテナンス/修復
5	ステージングのみ (インストールは、アドバンスト UI またはスイート/アドバンスト UI インストールがステージングされる、または圧縮解除および特定の場所にコピーされる、ステージ モードで実行中。インストール内のいずれのパッケージも、このモードでは実行しません。)

デバッグ方法 (例)

デバッグ ログ ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールで予期されなかった動作が認められた様々な異なる状況で役に立ちます。たとえばここで、スイート/アドバンスト UI プロジェクトに、.exe ファイルと .msi ファイルという 2 つのパッケージが含まれていると仮定します。スイート/アドバンスト UI インストールでは、それぞれのファイルに対して、それ自身の機能が関連付けられています。あるマシンで実行中に、両方の機能が選択されているにもかかわらず、.exe パッケージのみインストールされます。この動作のトラブルシューティングを開始するには、まず、この問題が発生しているマシン上でデバッグ ログ ファイルを生成します。ログ ファイルが生成されたら、両方の機能が実際にインストール用に選択されていたかどうかを検証します。ログ ファイルに、各機能について次のような情報が記録されています:

9-21-2011[03:07:04 PM]: Engine: determining suite feature states

9-21-2011[03:07:04 PM]: Initializing state for feature 'NewFeature'

9-21-2011[03:07:04 PM]: Default action state 1 for mode 0

9-21-2011[03:07:04 PM]: Initial feature state: 1

9-21-2011[03:07:04 PM]: Final feature state: 1

...

9-21-2011[03:07:04 PM]: Initializing state for feature 'NewFeature1'

9-21-2011[03:07:04 PM]: Default action state 1 for mode 0

9-21-2011[03:07:04 PM]: Initial feature state: 1

9-21-2011[03:07:04 PM]: Final feature state: 1

上記の情報では、機能 NewFeature と NewFeature1 の初期状態が示されています。両方の機能とも初期状態は、インストールが選択されたことを示す 1 であったことが分かります。従って、スイート/アドバンスト UI インストール内の機能は、問題であるようには見えません。さらに続けてログを見ていくと、次の情報が提供されています:

9-21-2011[03:07:10 PM]: Engine: setting parcel states as determined by feature selections

9-21-2011[03:07:10 PM]: Feature NewFeature setting parcel states, parent override: no, override state: 0

9-21-2011[03:07:10 PM]: Requesting action state 1 for parcel '{BA3EAE7A-0701-4CE0-9224-4F5C0F135792}'

9-21-2011[03:07:10 PM]: Containing feature is request state change to parcel {BA3EAE7A-0701-4CE0-9224-4F5C0F135792}, feature request: 1

スイート/アドバンスト UI インストールのこの時点までで、UI DLL は、UI 選択段階が完了し、インストールを開始するために必要な情報がすべて収集しています。ここで、スイート/アドバンスト UI エンジンは、関連付けられている機能を基に、インストールするパッケージ (ログ ファイルでは*パーセル*という言葉を使用しています) を判別します。この情報は、{BA3EAE7A-0701-4CE0-9224-4F5C0F135792} というパッケージが (この GUID は、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内の各パッケージで設定された "パッケージ GUID" 設定と同じ値です)、NewFeature (インストールが選択されている機能) という機能からインストール (action state 1) するように要求されていることを示しています。さらに見ていくと、次の情報が提供されています:

9-21-2011[03:07:10 PM]: Parcel is ineligible or the current parcel state and install mode to not allow parcel configuration

この行では、パッケージ (期待通りにインストールされない .msi パッケージ) が非対象になってなっていることが分かります。アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューで、提供されたパッケージ ID でこのパッケージを探すと、パッケージに対象条件が設定されています。条件は、パッケージは、特定の .msi 製品コードとパッケージ コードを持つパッケージがインストールされている場合、または、特定の製品コードがある .msi パッケージがインストールいる場合、および、製品バージョンが、.msi パッケージの製品バージョンより大きくない場合に対象となるように設定されています。この条件は、より新しいバージョンが既にインストールされている場合、パッケージがインストールされるのを防ぐ目的で設定されています。

この動作が発生したコンピューターをさらに検証すると、マシン上で新しいバージョンが既に存在し、これが原因で、スイート/アドバンスト UI インストール内の .msi パッケージが対象外なったことが分かります。

スイート/アドバンスト UI インストールにおける InstallScript アクションのデバッグ

スイート/アドバンスト UI インストールで InstallScript アクション の問題をトラブルシューティングするには、InstallScript デバッガーが利用できます。スイート/アドバンスト UI インストールの InstallScript アクションをデバッグするには、コマンドラインから /d パラメーターを使ってスイート/アドバンスト UI インストールを実行します。

メモ: *InstallScript コードをデバッグするには、デバッグ情報ファイルの* Setup.dbg *が使用可能である必要があります。また、開発マシン以外のシステムでインストールをデバッグする場合、特定のファイルを開発マシンからデ*

バッグ マシンにコピーする必要があります。詳細については、「任意のコンピューターでのインストールのデバッグ」を参照してください。

InstallScript デバッガーで次のコマンドを使って、InstallScript カスタム アクションを実行します:

Setup.exe /d

インストールが元のコンパイル場所であるパスにある場合、デバッグ シンボル ファイル (.dbg) へのパスも指定する必要があります:

Setup.exe /d"Path-to-Setup.dbg"

アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

/log コマンドライン パラメーターを使って、アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動する場合、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージ設定の構成で、ログ ファイルを生成するかどうかを指定できます。パッケージの種類に応じて、提供されている設定を構成して、その他の情報 (ログ ファイルが作成されるときにパッケージに渡すログ オプションなど) も指定できます。

パッケージにログ機能がサポートされていない場合、そのパッケージに対して、ログ機能を無効にしておくことをお勧めします。

パッケージのログ機能を有効にする

タスク

アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージにログのサポートを有効にするには、以下の手順に従います:

1. ログを有効にするパッケージを含むアドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
2. **[編成]** の下のビュー リストにある **[パッケージ]** をクリックします。

3. **[パッケージ]** エクスプローラーで、構成するパッケージを選択します。
4. **[共通]** タブにある **"ログの有効化"** 設定で、**[はい]** を選択します。
5. .msi または .msp パッケージで、必要に応じて、次のオプションのサブ設定を構成します:

   - **ログ ファイル**—ログ ファイルの名前を指定します。ファイルのパスを使用しないでください。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターによって、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。

     この設定を空白のままにしておいた場合、インストールで作成されたログ ファイルに、*PackageGUID*.log という名前が使用されます。*PackageGUID* は、[パッケージ] ビューの "パッケージ GUID" 設定でパッケージに割り当てられた GUID です。

   - **ログ オプション**—ログ ファイルを生成するとき、パッケージで使用する Windows Installer ログ /L フラグを指定します。たとえば、すべてを詳細に記録するログ ファイルを作成する場合は、この設定で、次のように入力します:

     *v

     その他の使用可能なフラグについては、/L の説明を参照してください。

     この設定を空白のままにしておくと、アスタリスク (*) と v フラグがログ ファイルの生成時に使用されます。

   .exe パッケージの場合、次のサブ設定を構成します:

   - **ログのコマンドライン**—アドバンスト UI またはスイート/アドバンスト UI インストールで、ログ機能を有効にするために .exe パッケージに渡すコマンドラインを指定します。サポートされている適切なフラグを含めます。構成中の .exe パッケージでログ機能がサポートされている場合、ファイルが作成されるディレクトリのパスの代わりに、アドバンスト UI またはスイート/アドバンスト UI のプロパティ ISLogDir を参照するパスを角かっこで囲んで含めます。

     たとえば、InstallShield でビルドされた Setup.exe ファイルによって実行される .msi パッケージについて、すべてを詳細に記録するログ ファイルを生成する場合、次のコマンドラインを入力します:

     /v"/l*v ¥"[ISLogDir]FileName.log¥""

     アドバンスト UI またはスイート/アドバンスト UI インストールによって [ISLogDir] が、ログ ファイルを含むフォルダーへのパスで置き換えられます。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターを使用すると、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。

アドバンスト UI またはスイート/アドバンスト UI インストールの実行中に、パッケージ ログ ファイルを生成する

パッケージ ログ ファイルを作成するには、/log コマンドライン パラメーターを使って、コマンドラインからアドバンスト UI またはスイート/アドバンスト UI インストールを実行します。/log パラメーターを使用すると、ログが有効されたアドバンスト UI またはスイート/アドバンスト UI インストール内の各パッケージに対して、ログ ファイルが作成されます。

特定のフォルダー内でパッケージ ログ ファイルを生成するには、フォルダーのパスを /log パラメーターと一緒に、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルに渡します。オプションで、/log パラメーターとパスをコロンで区切ることもできます。たとえば、次のどちらのコマンドラインを使用しても、ログ ファイルは PathToLogFiles フォルダーに作成されます:

Setup.exe /log"C:¥PathToLogFiles"

Setup.exe /log:"C:¥PathToLogFiles"

ログ ファイルの場所へのパスは、事前に存在している必要があります。

パッケージのログ ファイルを *%TEMP%* ディレクトリに生成する場合、パスを空白のままにしておきます。例:

Setup.exe /log

アドバンスト UI またはスイート/アドバンスト UI インストールのプロパティ ISLogDir は、パッケージ ログ ファイルを含むディレクトリへのパスを格納します。

ヒント: *アドバンスト UI またはスイート/アドバンスト UI インストールでは、logging system policy または MsiLogging プロパティを使用して、Windows Installer ベースのパッケージのログ ファイルを生成することもできます。*

64 ビット セットアップ EXE のビルド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

InstallShieldでは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトでインストールに 64 ビット **Setup.exe** セットアップランチャーを含めるかどうかを指定できます。64 ビット **Setup.exe** セットアップ ランチャーは、インストールが 64 ビット システムをターゲットとする場合のみ必要です。デフォルトで、InstallShield は 32 ビット システムだけでなく 64 ビットシステム上の WOW64 上で実行可能な 32 ビット セットアップ ランチャーをビルドします。一部の 64 ビット ターゲット システム、たとえば Windows Server Core システム- は、WOW64 (32-bit Windows-on-Windows) をサポートしない場合があります。これらの 64 ビット ターゲット システムは 32 ビット セットアップランチャーを実行できません。

[リリース] ビューにあるリリースの **[Setup.exe]** タブを使って、64 ビット セットアップ ランチャーをビルドするための情報を指定できます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの 64 ビット セットアップ EXE の構成にともない、WOW64 サポートを搭載していないピュア 64 ビット オペレーティング システムをターゲットすることが可能なピュア 64 ビット セットアップ EXE をビルドするためには、64 ビット パッケージおよび 64 ビット アクション イベントを構成する必要があります。ピュア 64 ビット スイート セットアップ EXE は、インストール中に 32 ビットパッケージをインストールまたは 32 ビット アクション イベントを実行することができません。これは、32 ビット バイナリを WOW64 サポートなしで 64 ビット ターゲット システム上にロードすることができないためです。ただし、InstallShield では、オペレーティング システムのアーキテクチャに基づいて選択的にパッケージをインストールするためのプラットフォーム条件を構成することが可能です。この機能によって、単一のスイート セットアップ インストーラーに複数のプラットフォームをターゲットとするパッケージを含めることができます。64 ビット セットアップ ランチャーをビルドするとき、32 ビット DLL、23 ビット マネージ コア、32 ビット .NET Core、および InstallScript アクション イベントはサポートされません。

条件ステートメントの構成方法については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

64 ビット セットアップ EXE ランチャーを含む 32 ビット パッケージをビルドする

パッケージに 64 ビット セットアップ EXE ランチャー オプションを持つ 32 ビット Windows インストーラー パッケージ (.msi) および 32 実行可能パッケージ (.exe) が含まれている場合、InstallShield でアドバンスト UI およびスイート/アドバンスト UI プロジェクトのビルドを行うと次の警告メッセージが表示されます。

- ISDEV: 警告 -7373: 64 ビット セットアップ (x86 Windows Installer パッケージ - 'パッケージ名' を含む) をビルドしています。
- ISDEV: 警告 -7373: 64 ビット セットアップ (x86 実行可能パッケージ - 'パッケージ名' を含む) をビルドしています。

これは 64 ビット セットアップ ランチャーがピュア 32 ビット システム上での実行に失敗し、32 ビット パッケージがピュア 64 ビット システム上でのインストールに失敗することを通知することが目的です。このエラーを回避するため、インストーラーがピュア 32 ビット システムおよび 64 ビット システム上の WOW64 をターゲットにする必要があるときは、64 ビット セットアップ EXE ランチャーを使用しないでください。または、すべての 32 ビット パッケージをピュア 64 ビット システムをターゲットにする 64 ビット パッケージに移行することも可能です。

64 ビット セットアップ ランチャーを使った InstallScript イベント アクションのビルド

InstallScript イベント アクションは、64 ビット セットアップ EXE ランチャーでサポートされていません。InstallShield では、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに InstallScript イベント アクションが含まれている場合、次のエラーが発生します:

ISDEV: エラー -7371: InstallScript カスタム アクションは、64 ビット セットアップ ランチャーでサポートされていません。

InstallScript イベント アクションが必要な場合、アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは 64 ビット セットアップ ランチャーを使用しないビルドを構成することができます。これは、InstallScript のランタイムが 32 ビット セットアップ EXE ランチャーを必要とする 32 ビットのままであるためです。または、InstallScript イベント アクションが不要な場合、プロジェクトから削除して 64 ビット セットアップ EXE ランチャーのビルドを続けることもできます。

64 ビット セットアップ ランチャーで 32ビット DLL、32 ビット マネージ コード、および 32 ビット.NET Core イベント アクションをビルドする

64 ビット セットアップ EXE ランチャーで 32ビット DLL、32 ビット マネージ コード、および 32 ビット.NET Core イベント アクションはサポートされていません。InstallShield のビルドでは、32 ビット DLL および 32 ビット マネージ コード イベント アクションを含む 64 ビット セットアップ EXE ランチャーのビルド中に次のエラーが発生します。

ISDEV: エラー -7374: 64 ビット セットアップ (x86 DLL アクション 'アクション名') をビルドしています。

ISDEV: エラー -7374: 64 ビット セットアップ (x86 EXE アクション 'アクション名' を含む) をビルドしています。

ISDEV: エラー -7392: x86 .NET Core アクション 'アクション名' を含む 64 ビット セットアップをビルドしています。

ISDEV: エラー -7339: x86 .マネージ コード アクション 'アクション名' を含む 64 ビット セットアップをビルドしています。

このエラーを解決するには、64 ビット セットアップ ランチャーに 64ビット DLL、32 ビット マネージ コード、および 64 ビット.NET Core イベント アクションを含みます。

64 ビット セットアップ ランチャーを使った 32 ビット EXE イベント アクションのビルド

InstallShield では、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで 64 ビット セットアップ EXE ランチャーおよび 32 ビット EXE イベント アクションを含むビルドが構成されている場合、ビルド中に次のエラー メッセージが表示されます。

ISDEV: エラー –7375: 64 ビット セットアップ (x86 EXE アクション 'アクション名' を含む) をビルドしています。

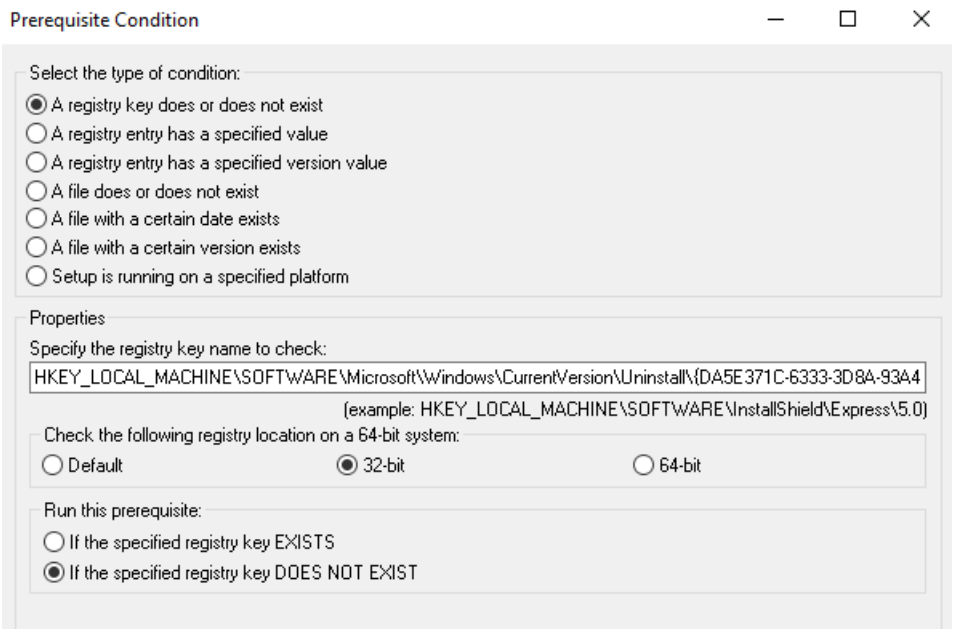
これは 64 ビット セットアップ ランチャーがピュア 32 ビット システム上での実行に失敗し、32 ビット EXE アクション イベントがピュア 64 ビット システム上での実行に失敗することを通知することが目的です。このエラーを回避するため、インストーラーがピュア 32 ビット システムおよび 64 ビット システム上の WOW64 をターゲットにする必要があるときは、64 ビット セットアップ EXE ランチャーを使用しないでください。または、すべての 32 ビット EXE アクション イベントをピュア 64 ビット システムをターゲットにする 64 ビット EXE に移行することも可能です。

InstallShield 前提条件および 64 ビット セットアップ ランチャー

一般的に InstallShield 前提条件を含むプロジェクトでは、Setup.exe によってターゲット システムで InstallShield 前提条件が満たされているかどうかが確認されるため、Setup.exe が必須になります。条件が満たされている場合、Setup.exe が InstallShield 前提条件をインストールします。

64 ビット セットアップ ランチャーを使用する InstallShield 前提条件を含むプロジェクトのビルドを構成する場合、必ず前提条件のインストーラーが 64 ビットであり、ピュア x64 システムにインストールが可能で、前提条件がレジストリまたはフォルダーの場所で 64 ビットの場所を確認するように構成します。

場合によって、64 ビット セットアップ ランチャーを使う 32 ビット前提条件のインストーラーをビルドしたい場合、32 ビットの場所を確認する条件を構成します。前提条件に含まれるほとんどのレジストリ条件はデフォルトの場所を確認するように構成されています (32 ビット セットアップ ランチャーの場合は 32 ビットの場所、64 ビット セットアップ ランチャーの場合は 64 ビットの場所)。ただし、32 ビットの前提条件インストーラーは、32 ビット レジストリハイブの下にレジストリ キーを作成します。この問題を解決するためには、32 ビットの場所を確認するようにレジストリの場所を InstallShield 前提条件エディターで明示的に変更する必要があります。前提条件についての情報は次の通りです:



32 ビット拡張条件 DLL および UI リソース DLL

64 ビット セットアップ ランチャーでは、実行時に 32 ビット拡張条件 DLL および UI リソース 拡張 DLL のロードでエラーが発生します。プロジェクトに 32 ビット拡張 DLLがバインドされている場合、対応する 64 ビット拡張 DLL をビルドして、64 ビット セットアップ ランチャーのビルドを構成する必要があります。

6

InstallShield 前提条件およびその他の再配布可能ファイルをデザインする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript オブジェクト*
- *InstallScript MSI*
- *マージ モジュール*

InstallShield は、InstallShield 前提条件、マージ モジュール、および InstallScript オブジェクトとしてプロジェクトに組み込むことができる多くのサード パーティ再配布可能ファイルを提供します。独自の再配布可能ファイルを作成して、他のインストール開発者に配布することもできます。これらの再配布可能ファイルの種類ごとに、以下のセクションで説明されています。

InstallShield 前提条件

プロジェクト: *次のプロジェクト タイプは、InstallShield 前提条件のサポートを含みます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。InstallShield で提供されている InstallShield 前提条件の一例として、Java Runtime Environment (JRE) および SQL Server Express Edition があります。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

プロジェクト: *Windows Installer ベースのインストールでは、プロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖することができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。*Setup.exe *セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。*

メモ: *Windows Installer 4.5 のチェーンとは異なり、InstallShield 前提条件のインストールは単一のトランザクションとして処理されません。つまり、正常に完了したインストールは、後続の前提条件のインストールが失敗した場合でもロールバックされることはありません。Windows Installer 4.5 のチェーン サポートの詳細については、「トランザクション処理を使って複数パッケージ インストールを構成する」を参照してください。*

独自の InstallShield 前提条件の作成方法についての詳細だけでなく、InstallShield で提供されている InstallShield 前提条件の構成方法については、「InstallShield 前提条件を定義する」を参照してください。

InstallShield 前提条件の種類

基本の MSI、InstallScript、または InstallScript MSI プロジェクトに InstallShield 前提条件を追加すると、デフォルトで *セットアップ前提条件*タイプの InstallShield 前提条件とみなされます。つまり、製品がインストールされる前にターゲット マシン上にインストール済みでなくてはならない、ベース アプリケーションまたはコンポーネントとして処理されます。前提条件がシステム上にインストール済みでない場合、メイン インストールが実行する前にセットアップ前提条件のインストールが実行します。

基本の MSI プロジェクトでは、InstallShield 前提条件をメイン インストールに含まれる機能に関連付けることができます。InstallShield 前提条件が 1 つまたは複数の機能に関連付けられているとき、これは *機能前提条件*と呼ばれます。機能前提条件は、実行時にその前提条件を含む 1 つまたは複数の機能がインストールされるとき、その前提条件がシステム上にインストール済みではない場合にインストールされます。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

これら 2 種類の InstallShield 前提条件についての詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。

マージ モジュール

プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

InstallShield では、独自の マージ モジュールを作成して、インストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

詳細については、「マージ モジュールのデザイン」を参照してください。

メモ: *[再配布可能ファイル] ビューに含まれているマージ モジュールの多くは Microsoft またはその他のサードパーティによるものです。InstallShieldでは、これらのモジュールを無料配布することによって、インストール プロジェクトの作成を支援します。ただし、サードパーティが作成したモジュールに存在する問題を InstallShield が修正したり直すことはできません。サード パーティが作成したモジュールに関する問題は、ベンダーへお問い合わせください。*

InstallScript オブジェクト

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield では、独自の InstallScript オブジェクトを作成して、他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

詳細については、「InstallScript オブジェクトを作成する」を参照してください。

InstallShield 前提条件を定義する

プロジェクト: 次のプロジェクト タイプは、InstallShield 前提条件のサポートを含みます:

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

これら全てのプロジェクト タイプは、InstallShield 前提条件タイプのセットアップ前提条件のサポートを含みます。基本の MSI プロジェクトは、機能前提条件タイプのサポートを含みます。

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

メモ: Windows Package Manger 前提条件を、スイート プロジェクトでインポートすることはできません。

WPM 前提条件のランタイムは、Windows 10 以降のワークステーションでサポートされています。その他のオペレーティング システムの場合、WPM 前提条件はランタイム中にスキップされます。

InstallShield の InstallShield 前提条件エディターを利用すると、InstallShield に含まれているすべての前提条件の設定を変更することができます。このツールを使ってカスタム前提条件を作成し、プロジェクトに追加することもできます。

独自の InstallShield 前提条件を指定したり既存の前提条件を変更することで、たとえば次のようなオプションの設定が可能になります。

- 前提条件ファイルをターゲットマシンへダウンロードする URL
- 前提条件をインストールする条件
- 特定の前提条件が依存している別の前提条件
- 前提条件のコマンドライン
- インストールがサイレントモードで実行中の場合に使われるコマンドライン
- 前提条件をインストールした後にターゲットマシンを再起動するかどうか
- 前提条件をインストールするために管理者権限が必要かどうか
- エンド ユーザーが前提条件をインストールするかどうかを選択するためのメッセージ ボックスを表示するかどうか
- Windows Package Manager からの前提条件の構成をサポートします。

このセクションには、次のトピックが含まれています:

- InstallShield 前提条件を作成する
- 既存の InstallShield 前提条件を変更する
- InstallShield 前提条件のプロパティを設定する
- nstallShield Windows Package Manager 前提条件の作成

- .prq ファイルの代替 URL を指定する
- InstallShield 前提条件のインストール条件を設定する
- InstallShield 前提条件のファイルを指定する
- InstallShield 前提条件をダウンロードする URL を指定する
- InstallShield 前提条件をインストールするためのパラメーターを指定する
- InstallShield 前提条件のインストールでターゲットマシンを再起動する
- InstallShield 前提条件のインストール動作を指定する
- InstallShield 前提条件の依存関係を指定する
- InstallShield 前提条件を保存する

InstallShield 前提条件を作成する

タスク ***新しい InstallShield 前提条件を作成するには、以下の手順に従います:***

1. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター**が開きます。
2. 各タブにある設定を適宜構成します。
3. **[ファイル]** メニューで、**[保存]** をクリックします。

既存の InstallShield 前提条件を変更する

タスク ***既存の InstallShield 前提条件を変更するには、以下の手順に従います:***

1. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター**が開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. InstallShield 前提条件ファイル (.prq) を参照および選択してから、**[開く]** をクリックします。
4. 必要に応じてフォントの変更を行います。
5. **[ファイル]** メニューで、**[保存]** をクリックします。

InstallShield 前提条件のプロパティを設定する

InstallShield 前提条件エディターの [プロパティ] タブでは、InstallShield 前提条件ファイルの説明および一意の ID を指定することができます。

タスク ***InstallShield 前提条件のプロパティを設定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[プロパティ]** タブをクリックします。
3. **InstallShield 前提条件の固有の ID**ボックスで前提条件ファイルに一意のファイル拡張子を入力するか、デフォルト値をそのまま利用します。これは前提条件アプリケーションの名前、コンポーネント、または GUID 名前、どちらでも構いません。

   メモ・新しい前提条件を作成するために [InstallShield 前提条件エディター] を開けるたびに、新しい GUID が生成され、InstallShield 前提条件の固有の IDボックスにリストされます。

4. **[説明]** ボックスに InstallShield 前提条件の概要を入力します。この説明は **[再配布可能ファイル]** ビューで InstallShield 前提条件ファイルを選択したときに、**概要**タイトルの下に表示されます。

nstallShield Windows Package Manager 前提条件の作成

タスク ***新しい InstallShield Windows Package Manager 前提条件を作成するには、次の手順に従います。***

1. **[ツール]** メニューで **[前提条件エディター]** をクリックします。**InstallShield 前提条件エディター** が開きます。
2. 次の手順に従って、インストールするパッケージの **[Windows Package Manager]** タブの設定を構成します:
   1. **[Windows Package Manager からインストール]** チェックボックスを選択します。
   2. いストールする WPM パッケージの**パッケージ ID** を指定します。
   3. 最新版か、特定のバージョンを指定します。
   4. 追加のパラメーターを指定します。
3. **[ファイル]** メニューで、**[保存]** をクリックします。

.prq ファイルの代替 URL を指定する

次の条件がどちらも true の場合、.prq ファイルに代替 URL を指定することもできます。

- インストールで、1 つまたは複数の InstallShield 前提条件を、Setup.exe ファイルまたはソース メディアに含めず、Web からダウンロードするように指定した。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。
- 後日、InstallShield 前提条件のアプリケーション ファイルをダウンロードするための、違う URL にエンドユーザーをリダイレクトする可能性がある。

この場合、代替 URL が、Setup.exe インストールと共に含めた .prq ファイルに指定されます。エンドユーザーがインストールを実行すると、ターゲットマシンは Setup.exe .prq ファイルで指定した代替 URL を参照して、その .prq ファイルをダウンロードし、代替 .prq ファイルで指定した URL ファイルを使用して必要条件アプリケーションに必要なファイルをダウンロードします。

タスク ***.prq ファイルに代替 URL を指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[プロパティ]** タブをクリックします。
3. **[前提条件ファイルをダウンロードする場合、.prq をダウンロードするための別のロケーション]** ボックスで、.prq ファイルの代替 URL を入力します。例:

   http://www.mywebsite.com/MyPrq.prq

メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

InstallShield 前提条件のインストール条件を設定する

ターゲット マシン上に InstallShield 前提条件が既にインストールされているかどうかを判断するインストール条件を指定する必要があります。これを行わなかった場合、ターゲット システムは前提条件が適切にインストールされなかったと見なして作動するため、問題が発生します。また、オペレーティング システム、レジストリ、またはファイル要件を指定するインストール条件を作成することも可能です。InstallShield 前提条件エディターの [条件] タブでインストール条件を設定することができます。

InstallShield 前提条件が (すべてではなく) いくつかのオペレーティング システム上にインストールされる場合、前提条件の複数オペレーティング システム条件を作成することができます。ターゲットマシンが 1 つの独立したオペレーティング システム条件を満たす場合、その前提条件用のオペレーティング システムが満たされます。

実行時に条件が True と評価すると、条件が満たされます。InstallShield 前提条件は、以下が当てはまる場合にターゲット マシンにインストールされます:

- ターゲット マシンがすべてのオペレーティング システム条件、および [条件] タブにリストされているその他すべての条件を満たしている。
- 機能前提条件のみ (つまり、メイン インストールに含まれる 1 つまたは複数の機能に関連付けられている InstallShield 前提条件) — 機能前提条件を含む機能は、必ずインストールされなくてはなりません。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

メモ: *Windows サービス パックなどの前提条件のインストールがサポートされていないため、オペレーティング システム バージョン情報に変更がないものと想定されます。したがって、オペレーティング システム条件は前提条件が正しくインストールされているかどうかを検証する際に考慮されません。*

InstallShield 前提条件の新しいインストール条件を追加する

タスク ***InstallShield 前提条件のインストール条件を新たに追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。
3. **[追加]** をクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
4. 条件の種類によって適切なオプションを選択し、必要に応じて条件のプロパティを設定します。

InstallShield 前提条件エディターが、[条件] タブに条件を追加します。

実行時に条件が True と評価すると、条件が満たされます。InstallShield 前提条件は、以下が当てはまる場合にターゲット マシンにインストールされます:

- ターゲット マシンがすべてのオペレーティング システム条件、および [条件] タブにリストされているその他すべての条件を満たしている。
- 機能前提条件のみ (つまり、メイン インストールに含まれる 1 つまたは複数の機能に関連付けられている InstallShield 前提条件) — 機能前提条件を含む機能は、必ずインストールされなくてはなりません。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

たとえば、いくつかの異なる条件を持つ InstallShield 前提条件を含んでいると想定します:

- ターゲット マシンに特定のファイルが存在しない。この例で、このファイルが不足している場合、製品が必要とするアプリケーションがインストールされることを示します。そうでない場合、製品はインストールされません。
- ターゲットマシンに Windows XP がある。
- ターゲットマシンに Windows Vista がある。

この例では、ターゲット マシンで Windows XP または Windows Vista が実行されている場合で、InstallShield 前提条件がどの機能にも関連付けられていないとき、さらに指定されたファイルが存在しない場合のみ、前提条件がインストールされます。InstallShield 前提条件が、インストールによって実行時にインストールされる機能と関連付けられている場合にも当てはまります。

メモ: *Windows サービス パックなどの前提条件のインストールがサポートされていないため、オペレーティング システム バージョン情報に変更がないものと想定されます。したがって、オペレーティング システム条件は前提条件が正しくインストールされているかどうかを検証する際に考慮されません。*

InstallShield 前提条件のオペレーティング システム条件を追加する

InstallShield 前提条件が (すべてではなく) いくつかのオペレーティング システム上にインストールされる場合、前提条件の複数オペレーティング システム条件を作成することができます。ターゲットマシンが 1 つの独立したオペレーティング システム条件を満たす場合、その前提条件用のオペレーティング システムが満たされます。

タスク ***InstallShield 前提条件に新しいインストール要件を追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。
3. **[追加]** ボタンをクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
4. [セットアップが指定のプラットフォーム上で実行されている] を選択します。
5. 以下のいずれかを実行します。
   - [**前提条件を実行するプラットフォームを選択します**] ボックスで、ターゲット システム上で前提条件に必要なオペレーティング システムを選択する。
   - **[前提条件を実行するプラットフォームを選択します]** ボックスで、**[カスタム]** を選択する。次いで、オペレーティング システムの設定を必要に応じて構成します。

   ヒント・ *既にある定義済みのオペレーティング システム条件からの値を使用して、新しいカスタム条件を作成することもできます。そのためには、**[前提条件を実行するプラットフォームを選択します]** ボックスで適切な定義済みオペレーティング システムを選択するか、設定を **[カスタム]** に変更します。**[カスタム]** 領域で追加の値を指定し、オペレーティング システム条件を続けて構成します。*
6. この前提条件が True 評価されるサービス パックの番号を指定するには、**[サービス パック]** ボックスを使用します。たとえば、ターゲット システムにサービス パック 2、3、または 4 が必要な場合、最初のボックスに 2、2 番目のボックスに 4 を入力します。将来のサービス パックもすべて含めるには、2 番目のボックスを空白のままに残します。サービス パック 3 のみをターゲットとする場合は、両方のボックスに 3 と入力します。
7. **[OK]** をクリックします。

InstallShield 前提条件エディターが、[条件] タブにオペレーティング システム条件を追加します。

メモ・ *InstallShield 前提条件エディターは、**[前提条件を実行するプラットフォームを選択します]** ボックスで選択された値を .prq ファイルに格納する代わりに、選択した値に関連付けられている基のオペレーティング システムの設定を格納します。したがって、[カスタム] オプションを選択したあと、[カスタム] 領域で設定の変更または構成を行わなかった場合、選択された条件の [前提条件設定] ダイアログ ボックスが再度開かれたとき、InstallShield 前提条件エディターは、このボックスの値を [カスタム] に設定しません。代わりに、ボックスの値を定義済みのオペレーティング システム オプションに設定します。*

InstallShield 前提条件のインストール条件を変更する

タスク ***既存の条件を変更するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。

3. 変更する条件を選択します。
4. **[変更]** ボタンをクリックします。**[前提条件設定]** ダイアログ ボックスが開きます。
5. 必要に応じて条件を変更します。

InstallShield 前提条件エディターが、[条件] タブの変更済み条件を更新します。

InstallShield 前提条件からインストール条件を削除する

タスク ***InstallShield 前提条件から条件を削除するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[条件]** タブをクリックします。
3. 削除する条件を選択します。
4. **[削除]** をクリックします。

InstallShield 前提条件のファイルを指定する

新しい InstallShield 前提条件を作成する時、前提条件と共に含む必要があるインストール ファイルを指定しなくてはなりません。既存する InstallShield 前提条件に必要なファイルのリストを変更することもできます。InstallShield 前提条件エディターの [含めるファイル] タブでファイルを指定します。

タスク ***InstallShield 前提条件に必要なファイルをひとつ追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. [**含めるファイル**] タブをクリックします。
3. **[追加]** をクリックします。**[新規ファイル]** ダイアログ ボックスが開きます。
4. **[ファイル]** ボックスで、追加するファイルの名前とパスを入力します。ファイルを参照して見つけるには、省略記号 (...) ボタンをクリックします。
5. エンドユーザーが Web からファイルをダウンロードすることが可能な場合、URL を **[ファイルへの URL]** ボックスで指定します。この URL は、インストール作成者が **[再配布可能ファイル]** ビューを利用してインターネットからローカル マシンへ InstallShield 前提条件をダウンロードする時に InstallShield が利用する URL と同じです。

   たとえば、そのファイルの URL が **http://www.mywebsite.com/Folder1/Notepad.exe** の場合、次をこのボックスに入力します。

   http://www.mywebsite.com/Folder1

メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

タスク ***InstallShield 前提条件に必要なファイルを一度に複数追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、**[含めるファイル]** タブをクリックします。
2. **[複数ファイルの追加]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. 追加するファイルを指定します。 連続する複数のファイルを選択するには、最初のファイルを選択してから SHIFT キーを押しながら最後のファイルを選択します。連続しない複数のファイルを選択するには、最初のファイルを選択してから CTRL キーを押しながらその他の各ファイルを選択します。
4. **[開く]** をクリックします。

メモ: *この手法でファイルを追加すると、URL(ファイルをダウンロードできる場所)は設定されません。複数ファイルに URL を設定する場合は、「InstallShield 前提条件をダウンロードする URL を指定する」を参照してください。*

タスク ***既存するファイルの設定を変更するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、**[含めるファイル]** タブをクリックします。
2. ファイル名、パス、または URL を変更するファイルを指定します。
3. **[変更]** をクリックします。**[新規ファイル]** ダイアログ ボックスが開きます。
4. 必要に応じて設定を変更します。

タスク ***InstallShield 前提条件からファイルを削除するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、**[含めるファイル]** タブをクリックします。
2. 削除するファイルを選択します。
3. **[削除]** をクリックします。

InstallShield 前提条件をダウンロードする URL を指定する

エンドユーザーはインストールに含まれるセットアップ前提条件に必要なファイルのダウンロードが可能でなくてはなりません。InstallShield 前提条件エディターの [含めるファイル] タブで必ず URL を指定してください。

メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

タスク ***InstallShield 前提条件に必要なファイル (複数可) のURL を指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. [**含めるファイル**] タブをクリックします。

3. その URL を指定するファイル (複数可) を指定します。連続する複数のファイルを選択するには、最初のファイルを選択してから SHIFT キーを押しながら最後のファイルを選択します。連続しない複数のファイルを選択するには、最初のファイルを選択してから CTRL キーを押しながらその他の各ファイルを選択します。
4. [**ファイルの URL を設定する**] をクリックします。**[ファイル URL の設定]** ダイアログ ボックスが開きます。
5. **[ファイルの親フォルダーへの URL]** ボックスで、選択されたファイルの親ディレクトリの URL を入力します。この URL は、インストール作成者が **[再配布可能ファイル]** ビューを利用してインターネットからローカル マシンへ InstallShield 前提条件をダウンロードする時に InstallShield が利用する URL と同じです。

   たとえば、**Notepad.exe** および **Paint.exe** というファイルを選択して、これらのファイルの URL がそれぞれ **http://www.mywebsite.com/Folder1/Notepad.exe** および **http://www.mywebsite.com/Folder1/Paint.exe** となっている場合、次の通りこのボックスに入力します。

   http://www.mywebsite.com/Folder1
6. **[OK]** をクリックします。

ヒント: 上記手順とは別の方法として、[含めるファイル] タブにある前提条件ファイルのリストに各ファイルを追加する際に URL を指定することも可能です。詳細については、「InstallShield 前提条件のファイルを指定する」を参照してください。

InstallShield 前提条件をインストールするためのパラメーターを指定する

InstallShield 前提条件エディターの [実行するアプリケーション] タブでは、ターゲット マシン上への前提条件のインストール方法を指定することができます。

タスク ***InstallShield 前提条件をインストールするためのパラメーターを指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[実行するアプリケーション]** タブをクリックします。
3. **[起動するアプリケーションを指定してください]** リストで、InstallShield 前提条件がインストールされている場合にターゲット マシンで起動するファイル (通常、Setup.exe セットアップランチャーまたは .msi ファイル) を選択します。**[含めるファイル]** タブで指定したファイルのみがこの一覧に含まれます。

   メイン インストールが基本の MSI インストールまたは InstallScript MSI インストールの場合で、.msi ファイルを指定して、[動作] タブで進行状況の表示を指定した場合、Setup.exe セットアップランチャーは進行状況メッセージをキャプチャして、実行時に MsiExec.exe の代わりに Windows Installer API を使って .msi パッケージを起動します。

   その他の種類のファイルを指定した場合、または進行状況が表示されない .msi ファイルを指定した場合、Setup.exe セットアップランチャーは実行時に Open 動詞 (.msi と .exe ファイルの場合)、またはデフォルトの動詞 (その他の種類のファイル) のどちらかを使ってファイルを実行します。

*プロジェクト・.msi ファイルを指定し、[動作] タブで進行状況の表示を指定した場合、*Setup.exe *セットアップランチャーは進行状況メッセージをキャプチャして、実行時に* MsiExec.exe *の代わりに Windows Installer API を使って .msi パッケージを起動します。*

その他の種類のファイルを指定した場合、進行状況が表示されない .msi ファイルを指定した場合、またはメイン インストールが InstallScript インストールの場合、Setup.exe セットアップランチャーは実行時に Open 動詞 (.msi と .exe ファイルの場合)、またはデフォルトの動詞 (その他の種類のファイル) のどちらかを使ってファイルを実行します。

4. Windows Installer エンジン、.NET Framework またはその両方をこの InstallShield 前提条件のインストールの前にインストールする必要がある場合、[**Windows Installer エンジンおよび/または .NET Framework を最初にインストールする必要がある**] チェック ボックスを選択します。

メモ・このチェック ボックスを選択しても Windows Installer エンジンまたは .NET Framework をインストールに追加しません。これらがインストールに含まれている場合、InstallShield 前提条件がインストールされる前にこれらをインストールしなくてはならないことを指定するだけです。Windows Installer エンジンまたは .NET Framework をインストールに含める場合、それらをプロジェクトに追加する必要があります。詳しくは、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」または「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」をご覧ください。

5. 必要に応じて、[**アプリケーションのコマンドラインを指定する**] ボックスで [**起動するアプリケーションを指定する**] リストで選択したファイルのコマンドラインを入力します。このボックスにファイル名を含めないでください。

   詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

6. 必要に応じて、[**セットアップがサイレントモードで実行中にアプリケーションのコマンドラインを指定する**] ボックスに適切なコマンドラインを入力します。このボックスにファイル名を含めないでください。

   詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

7. 選択された InstallShield 前提条件アプリケーションがインストール後にターゲットマシンの再起動を要求する場合、[**再起動が必要な場合、アプリケーションが戻すコードを指定してください(十進法)**] ボックスに戻りコードを入力します。

ヒント・複数のリターンコードが存在する場合、各コードはカンマで区切ってリストしてください。

- *InstallShield 前提条件として起動するファイルの戻りコードが分からない場合は、ファイルの作成者に問い合わせてください。*
- *再起動が必要な InstallShield 前提条件の詳細は、「InstallShield 前提条件のインストールでターゲットマシンを再起動する」を参照してください。*

InstallShield 前提条件のコマンドライン パラメーターを指定する

InstallShield 前提条件エディターの [実行するアプリケーション] タブでは、ターゲット システム上で InstallShield 前提条件が起動されたときに使用されるコマンドライン パラメーターを指定できます。

タスク

InstallShield 前提条件のインストールが起動されたときに使用するコマンドライン パラメーターを指定するには、以下の手順に従います:

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[実行するアプリケーション]** タブをクリックします。
3. **[アプリケーションのコマンドラインを指定する]** ボックスで、**[起動するアプリケーションを指定する]** リストで選択したファイルの有効な任意のコマンドライン パラメーターを入力します。このボックスにファイル名を含めないでください。
4. メインのインストールがサイレント モードで実行中される場合に、InstallShield 前提条件に使用するコマンドラインを指定するには、**[セットアップがサイレントモードで実行するときのアプリケーションのコマンドラインを指定してください]** ボックスで任意の有効なコマンドライン パラメーターを入力します。このボックスにファイル名を含めないでください。

   この設定に入力されたコマンドライン パラメーターは、その前提条件が表示しないように構成されている場合、つまり [動作] タブで [**前提条件をインストール一覧に表示しない**] チェック ボックスが選択されている場合にも使用されます。

メモ: *InstallShield 前提条件を含むインストールを /s コマンドライン パラメーターを使って起動すると、前提条件のインストールは自動的にサイレントで実行されません。必要に応じて、[実行するアプリケーション] タブの "**セットアップがサイレントモードで実行中にアプリケーションのコマンドラインを指定する**" 設定で、InstallShield 前提条件用の有効なサイレント コマンドライン パラメーターを指定してください。*

*[**セットアップがサイレント モードで実行するときのアプリケーションのコマンドラインを指定してください。**] ボックスを空白のままに残した場合で、次の条件の 1 つまたは両方が True の場合、InstallShield 前提条件にコマンドライン パラメーターは渡されません:*

- *メイン インストールがサイレントで実行されている。*
- *前提条件について、[動作] タブにある [**前提条件をインストール一覧に表示しない**] チェック ボックスが選択されている。*

このため、標準のコマンドライン パラメーターを指定する場合、サイレント コマンドライン パラメーターも指定することが推奨されます。

セットアップランチャーが InstallShield 前提条件にコマンドライン パラメーターを渡す仕組み

InstallShield 前提条件のインストールが .msi パッケージで、[動作] タブで進行状況の表示を選択した場合、セットアップランチャーは MsiExec.exe の代わりに Windows Installer API を使って 、[実行するアプリケーション] タブで指定されたコマンドライン パラメーターで .msi パッケージを起動します。これらの状況下では、コマンドラインに以下の種類のパラメーターをどれでも入力できます:

- 次のフォーマットで表記された Windows Installer プロパティ名と値:

  PROPERTY=VALUE

- /L (/L*v などの任意の変数も使用可能)
- /log
- /q (/qb+! などの任意の変数も使用可能)

- /quiet

進行状況の表示を指定すると、デフォルトで、前提条件の .msi パッケージのユーザー インターフェイスは表示されません。この動作をオーバーライドするには、コマンドライン パラメーターとして /qf を指定できます。

InstallShield 前提条件のインストールがその他の種類のファイルである場合、または進行状況が表示されない .msi パッケージである場合、セットアップランチャーは実行時に Open 動詞 (.msi と .exe ファイルの場合)、またはデフォルトの動詞 (その他の種類のファイル) のどちらかを使ってファイルを実行します。したがtって、進行状況が表示されない .msi パッケージの場合、MsiExec.exe が受け付けるコマンドライン パラメーターであればどれでも使用できます。サポートされているコマンドライン パラメーターのリストは、Windows Installer ヘルプ ライブラリの「Command-Line Options」と「Standard Installer Command-Line Options」を参照してください。

InstallShield 前提条件のインストールが、InstallShield で作成された Setup.exe ファイルの場合、 InstallShield 前提条件がプロジェクトの機能と関連付けられているかどうかによって、[実行するアプリケーション] タブで指定した前提条件コマンドライン パラメーターの形式とは異なります。InstallShield 前提条件が機能と関連付けられていない場合、それは *セットアップ前提条件*と呼ばれます。InstallShield 前提条件が 1 つまたは複数の機能と関連付けられている場合、それは *機能前提条件*と呼ばれます。

メモ: *「セットアップ前提条件」と「機能前提条件」の違いについての詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。*

セットアップ前提条件に関して、セットアップランチャーは以下のプロパティについてコマンドライン パラメーターを使ったプロパティの置換をサポートします。

- SETUPEXEDIR – セットアップランチャー ファイルへのパスを識別します。たとえば、セットアップランチャーへのパスが C:¥MySetups¥MyApp¥Setup.exe とすると、[SETUPEXEDIR] の値は C:¥MySetups¥MyApp になります。
- SETUPEXENAME – プロジェクトがビルドされるときに作成される、セットアップランチャー ファイルの名前を識別します。セットアップランチャー ファイルの名前が変更された場合、インストールは実行時に、このプロパティ値を更新します。以下のパスは、このファイルへの完全パスを示します:

  [SETUPEXEDIR]¥[SETUPEXENAME]
- ISPREREQDIR – 実行中の InstallShield 前提条件へのパスを識別します。このパスは、末尾のスラッシュを含みます。このプロパティを使って、InstallShield 前提条件内のファイルを参照することができます(例、**[ISPREREQDIR]MyConfigFile.ini**) 。
- ProductLanguage – データベースに書き込まれていない、ユーザー インターフェイスに含まれる任意の文字列用にインストールが使用する言語を識別します。

機能前提条件に関して、[起動するアプリケーション] タブで指定したコマンドライン パラメーターは、実行時にフォーマットされたテキスト フィールドと同様にフォーマットされます。したがって、Setup.exe がサポートする標準コマンドライン パラメーターだけでなく、前述のプロパティもコマンドラインで使用することができます。サポートされているコマンドライン パラメーターの一覧は、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

InstallShield 前提条件のインストールでターゲットマシンを再起動する

InstallShield はいくつかの方法を使って、InstallShield 前提条件の実行後にターゲット マシンの再起動が必要かどうかを決定します。

レジストリキーの変更

InstallShield 前提条件が実行された後に次のレジストリキーのどれかが変更された場合、ターゲットマシンの再起動が必要な可能性があります。

- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnce
- HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥RunOnceEx
- HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Control¥Session Manager¥PendingFileRenameOperations

レジストリキーは InstallShield 前提条件が実行された後に考慮されます。これらの数字が異なる場合、ファイルがシステムの再起動を試みると同時にインストールを終了するものと見なされます。

.prq ファイルで指定されたリターンコード

.exe ファイルの exit コードが InstallShield 前提条件 (.prq) ファイルのリターン コードと一致する場合、ターゲットマシンの再起動が必要な可能性があります。.prq ファイルのリターン コードは InstallShield 前提条件エディターの **[再起動が必要な場合、アプリケーションが戻すコードを指定してください]** に入力されます。詳細については、「InstallShield 前提条件をインストールするためのパラメーターを指定する」を参照してください。

リターンコード 1641 および 3010

InstallShield 前提条件用に実行される .msi パッケージに reboot ステートが返された場合、ターゲットマシンの再起動が必要な可能性があります。標準 Windows Installer の再起動リターンコードは以下の通りです。

- ERROR_SUCCESS_REBOOT_INITIATED (1641)
- ERROR_SUCCESS_REBOOT_REQUIRED (3010)

.prq ファイルで指定された動作

InstallShield 前提条件エディターの [動作] タブに表示されるフィールドの 1 つを利用して、前述の条件が特定の InstallShield 前提条件に適合するかどうかによって後に続く動作を指定することができます。詳細については、「再起動を必要とする InstallShield 前提条件の動作を指定する」を参照してください。

InstallShield 前提条件のインストール動作を指定する

新しい InstallShield 前提条件を作成する、または既存の InstallShield 前提条件を変更するとき、エンドユーザーが前提条件のインストールをスキップすることができるかどうかを指定できます。また、ターゲット マシンの再起動の要否に応じて、前提条件のインストールをどのように進めるかを指定することもできます。InstallShield 前提条件エディターの [動作] タブでは、こういった種類の InstallShield 前提条件のインストール動作を指定することができます。

InstallShield 前提条件に管理者権限が必要であることを指定する

Windows Vista 以降の環境では、エンド ユーザーは通常、管理者権限がない標準ユーザーとしてシステムを実行します。InstallShield 前提条件が管理者権限を必要とする場合、Windows Vista は、ユーザー アカウント制御 (UAC) プロンプトを表示して、エンドユーザーに同意 (コンセント) または認証情報を提供するように要求します。

InstallShield 前提条件に管理者権限が必要な場合、または、InstallShield 前提条件をマシンごとにインストールする場合、InstallShield 前提条件エディターの [動作] タブでそれを指定することができます。

タスク ***InstallShield 前提条件をインストールするために管理者権限が必要であると指定するには、次の手順を実行します。***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **[前提条件は管理者権限を必要とする]** チェック ボックスを選択します。

メモ: *この設定は Windows Vista 以降のシステム上で実行されるインストールに適用します。以前のバージョンの Windows はこの設定を無視します。*

InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるようにする

InstallShield 前提条件のインストールの要否をエンド ユーザーが選択できるメッセージ ボックスを表示することができます。InstallShield 前提条件エディターの [動作] タブでこれを指定します。

タスク ***エンド ユーザーが InstallShield 前提条件のインストールの要否を選択できるようにするには、次の手順に従います。***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **"ユーザーは前提条件をオプションでスキップすることができる"**チェック ボックスを選択します。

ターゲットシステムで [インストールされるセットアップ前提条件] リストに、前提条件の名前を含めるかどうかを指定する

ターゲット システムに 1 つまたは複数のセットアップ前提条件のインストールが必要な場合、通常は実行時にメイン インストールが実行する前にセットアップ前提条件ダイアログが表示されます。エンド ユーザーがこのダイアログで [インストール] ボタンをクリックすると、セットアップ前提条件がインストールされます。

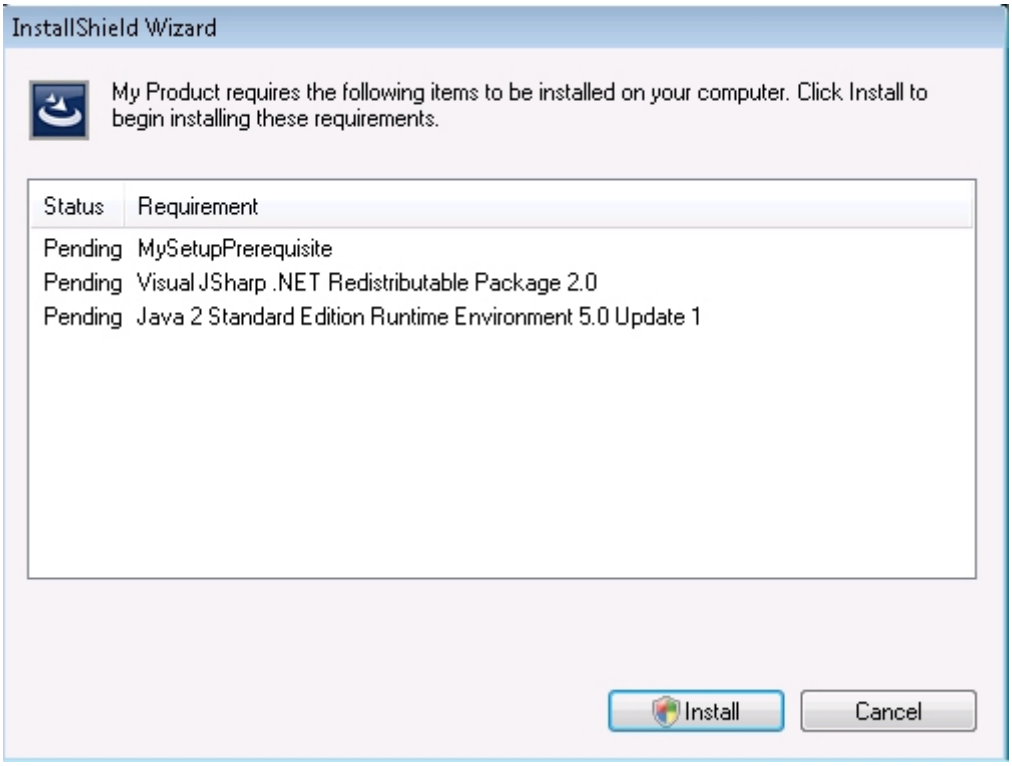


図 1: [インストールするセットアップ前提条件] リストを表示するサンプル [セットアップ前提条件] ダイアログ

InstallShield 前提条件エディターの [動作] タブで、前提条件ダイアログの一覧にセットアップ前提条件を表示しないかどうかを指定できます。前提条件を隠した場合、インストールが必要な前提条件の 1 つとしてリストされていなくても、条件がそれを必要したときに前提条件がインストールされます。

インストールに含まれるすべてのセットアップ前提条件が隠されている場合、インストールは実行時にセットアップ前提条件ダイアログを表示しません。

タスク ***実行時に、セットアップ前提条件ダイアログでセットアップ前提条件をリストするかどうかを指定するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. 以下のいずれかを実行します。
   - 前提条件を一覧に表示しない場合、**[前提条件をインストール一覧に表示しない]** チェック ボックスを選択します。

     セットアップ前提条件が非表示の場合、標準のコマンドライン パラメーターではなく、サイレント コマンドライン パラメーターを使って起動されます。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。
   - セットアップ前提条件を一覧に表示する場合、**[前提条件をインストール一覧に表示しない]** チェック ボックスをクリアします。

メモ: [前提条件をインストール一覧に表示しない] チェック ボックスが選択またはクリアされているかに関わらず、機能前提条件が実行時にセットアップ前提条件ダイアログに表示されることはありません。つまり、InstallShield 前提条件をプロジェクトに追加して、それを機能に関連付けた場合、実行時、メイン インストールが実行する前に表示されるセットアップ前提条件ダイアログで、機能前提条件は一覧に表示されません。機能前提条件に関する詳細は、「セットアップ前提条件と機能前提条件の違い」を参照してください。

実行時に、InstallShield 前提条件のインストールの進行状況を表示するかどうかを指定する

前提条件エディターの [動作] タブでは、実行時に Windows Installer によって表示されるインストールの進行状況メッセージと共に、InstallShield 前提条件のインストールがインストールの実際の進行状況を表示するステータス バーを表示するかどうかを指定します。この機能は、前提条件が .msi ファイルを起動する場合のみ使用が可能で、Setup.exe ファイルを起動する場合は使用できません。さらに、InstallShield 前提条件を含むインストールは基本の MSI インストールまたは InstallScript MSI インストールでなくてはなりません。

進行状況が表示されない場合、または InstallShield 前提条件を含むインストールが InstallScript インストールの場合、前提条件のインストールは実行時ダイアログの「**[PrerequisiteName] をインストールしています**」メッセージを表示して、ステータス バーは前提条件のインストールが完了するまでループします。

タスク

InstallShield 前提条件のインストールについて進行状況メッセージとステータス バーを表示するかどうかを指定するには、以下の手順に従います:

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. 以下のいずれかを実行します。
   - 進行状況を表示するには、**[前提条件のウィンドウに進行状況を表示する (未加工の MSI ファイルのみ)]** チェック ボックスを選択します。
   - 進行状況を表示しない場合は、**[前提条件のウィンドウに進行状況を表示する (未加工の MSI ファイルのみ)]** チェック ボックスをクリアします。

メモ: 進行状況の表示を選択した指定した場合、一部のコマンドライン パラメーターしかサポートされません。コマンドライン パラメーターは、InstallShield 前提条件エディターの [実行するアプリケーション] タブで指定されます。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。

前提条件の進行状況の表示を指定すると、デフォルトで、前提条件の .msi パッケージのユーザー インターフェイスは表示されませんので、ご注意ください。この動作をオーバーライドするには、InstallShield 前提条件エディターの [実行するアプリケーション] タブで、コマンドライン パラメーターとして /qf を指定できます。

InstallShield 前提条件のインストールに伴う可能性のある問題の対処方法について

InstallShield 前提条件のインストール実行後、引き続き InstallShield 前提条件のインストールが必要であると表示される。原因として、次の理由が考えられます。

- InstallShield 前提条件が正しくインストールされなかった。
- InstallShield 前提条件に不正確な条件が指定されていた。

たとえば、ターゲット マシンに特定のファイルが存在しない場合に InstallShield 前提条件をインストールする必要があると条件付けたとします。InstallShield 前提条件がインストールされた後にもそのファイルが不足している場合、その条件は適切ではなかったと考えられます。InstallShield 前提条件エディターの [動作] タブでは、これらの状況下でインストールをどのように進めるのかを指定することができます。

タスク

InstallShield 前提条件のインストールに関連して起こりうる問題の対処には、次の手順に従います:

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **前提条件をインストールした後も、その前提条件のインストールがまだ条件で要求された場合** リストで、適切な項目をクリックします。

再起動を必要とする InstallShield 前提条件の動作を指定する

新しい InstallShield 前提条件を作成または既存の InstallShield 前提条件を変更するとき、ターゲット マシンの再起動の要否に従って 前提条件のインストール処理をどのように進めるのかを指定することができます。たとえば、マシンを再起動する前にインストールがエンド ユーザーに対してプロンプトを行う場合、または再起動そのものをスキップする場合が考えられます。InstallShield 前提条件エディターの [動作] タブでは、各 InstallShield 前提条件に適切な応答を行うための柔軟性を提供します。

タスク

ターゲット マシンの再起動を要求する InstallShield 前提条件の動作を指定するには、次の手順に従います。

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[動作]** タブをクリックします。
3. **前提条件が再起動を必要とする場合**リストで、適切な項目をクリックします。

使用可能なオプションについて詳しい情報は、[動作] タブを参照してください。

InstallShield 前提条件の依存関係を指定する

新しい InstallShield 前提条件を作成する、または既存の InstallShield 前提条件を変更するとき、この InstallShield 前提条件が依存する他の 前提条件 (.prq ファイル) を指定することができます。依存関係は、InstallShield 前提条件エディターの依存関係タブで指定します。

InstallShield 前提条件の依存関係を追加する

タスク ***InstallShield 前提条件の依存関係を追加するには、以下の手順に従います:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[依存関係]** タブをクリックします。
3. **[追加]** ボタンをクリックします。**[新しい依存関係]** ダイアログ ボックスが開きます。
4. **[ファイル]** ボックスで、現在の InstallShield 前提条件に必要な .prq ファイルの名前とパスを入力します。.prq ファイルを参照して見つけるには、省略記号 (...) ボタンをクリックします。InstallShield 前提条件ファイル (.prq) は、次の場所に格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites
5. **[OK]** をクリックします。

InstallShield 前提条件の依存関係を変更する

タスク ***既存の依存関係の設定を変更するには、以下の手順を実行します:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[依存関係]** タブをクリックします。
3. 変更する依存関係を選択します。
4. **[変更]** をクリックします。**[新しい依存関係]** ダイアログ ボックスが開きます。
5. 必要に応じて設定を変更します。

InstallShield 前提条件の依存関係を削除する

タスク ***前提条件から依存関係を削除するには、以下の手順を実行します:***

1. **InstallShield 前提条件エディター**で、変更する前提条件を開きます。
2. **[依存関係]** タブをクリックします。
3. 削除する依存関係を選択します。
4. **[削除]** をクリックします。

InstallShield 前提条件を保存する

新しい InstallShield 前提条件を作成したとき、または InstallShield の InstallShield 前提条件エディターを利用して既存の前提条件を変更したとき、InstallShield 前提条件 (.prq) ファイルを保存しなくてはなりません。

タスク ***InstallShield 前提条件エディターで開いている InstallShield 前提条件を保存するには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. InstallShield 前提条件ファイルの名前と .prq 拡張子を指定しますInstallShield 前提条件ファイル (.prq) は、次の場所に格納されています:

   InstallShield Program Files フォルダー¥SetupPrerequisites
3. **[保存]** ボタンをクリックします。

SetupPrerequisites に格納されている InstallShield 前提条件は、[再配布可能ファイル] ビューにリストされています。

マージ モジュールのデザイン

プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

マージ モジュールを利用することにより、複数の開発チームが互いに独立したコンポーネントについて作業できるようになります。各チームは、自分たちの作業が他のコンポーネントにどう影響するかを考慮せずに、そのインストールデータベースに値を設定できます。各コンポーネントの開発が完全に終了すると、マージ モジュールデータベースの各コンポーネントを中央の製品インストールデータベースにマージできます。

InstallShield では、独自の マージ モジュールを作成して、インストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

マージ モジュールを作成する最も重要なステップは、その設計にあります。上手く設計されたマージ モジュールにするためには、ファイルをプロジェクトの開発者的視点から見た構成による、複数のコンポーネントに分割する必要があります。機能、コンポーネント、ファイルを含む完全なインストールを作成する場合とは違い、マージ モジュールには、コンポーネントのみが含まれます (コンポーネントの中にファイルがあります)。

マージ モジュールの作成には、主に以下のものが関係します。

テーブル 1 ・ マージ モジュール作成の概観

マージ モジュール要素	説明
コンポーネント	アプリケーション データをグループ化できます。コンポーネントは、プロジェクトを開発者側の視点から見た構成であり、ファイル、レジストリ エントリ、およびショートカットなどのデータを含みます。コンポーネント ビューでは、マージ モジュールのコンポーネントを設計します。
ファイル、レジストリデータ およびショートカット	上記で示した階層構造を完成するためにコンポーネントに追加される要素です。
詳細設定	詳細設定を指定して、コンポーネントのパブリッシュ、COM サーバー、ファイル拡張サーバー、および MIME の種類の登録を行なうことができます。コンポーネントの詳細設定を使って、レジストリにアプリケーションパスのエントリを作成することもできます。

このセクションには、次のトピックが含まれています:

- マージ モジュール プロジェクトを作成する
- マージ モジュールのデフォルトのインストール先フォルダーを指定する
- マージ モジュール言語を選択する
- マージ モジュールに依存関係を追加する
- マージ モジュールへ除外を追加する
- マージ モジュールをマージ モジュール プロジェクト内からリポジトリにパブリッシュする

マージ モジュール プロジェクトを作成する

プロジェクト- この情報は、マージ モジュール プロジェクトに適用します。

マージ モジュール (.msm ファイル) には、個別機能をインストールするために必要なロジックとファイルのすべてが含まれています。たとえば、多くのアプリケーションには、Visual Basic ランタイム DLL が必要です。コンポーネントにファイルを含めてインストール要件を調べる必要はなく、プロジェクトの機能に Visual Basic 仮想マシン マージ モジュールを添付するだけでこれを実行できます。

マージ モジュールプロパティには、以下の項目が含まれます。

- 製品名
- 製品バージョン
- 言語
- INSTALLDIR

マージ モジュールの依存関係と除外は、[一般情報] ビューにある "モジュールの依存関係" および "モジュールの除外" 設定で指定できます。

マージ モジュールを作成するには、まずマージ モジュール プロジェクトを作成します。

タスク *マージ モジュール プロジェクトを作成するには、以下の手順に従います:*

1. **[ファイル]** メニューで、**[新規]** をクリックします。
2. **[Windows Installer]** タブをクリックします。
3. [プロジェクトの種類] ボックスで、**[マージ モジュール プロジェクト]** を選択します。
4. **[プロジェクト名]** ボックスに、プロジェクトの名前を入力します。

   重要 *- 新しいマージ モジュール プロジェクトを作成するとき、[新規プロジェクト] ダイアログ ボックスで指定する名前は 35 文字以下でなくてはなりません。これは、入力した名前がマージ モジュールファイルの* ModuleSignature *テーブルにある ModuleID フィールドで GUID と共に利用されるため、また ModuleID フィールドのオブジェクト名部分の最大文字数が 35 文字のためです。*
5. **[場所]** ボックスに、プロジェクトを保存する場所を指定するか、**[参照]** ボタンをクリックして、その場所に移動します。
6. **[OK]** をクリックします。

マージ モジュールのデフォルトのインストール先フォルダーを指定する

プロジェクト- この情報は、マージ モジュール プロジェクトに適用します。

[一般情報] ビューの "INSTALLDIR" 設定に入力された値は、マージ モジュールのすべてのファイルのデフォルト フォルダーとなります。したがってその値は、デフォルトのコンポーネントインストール先フォルダーである、Windows Installer フォルダープロパティ **INSTALLDIR** に割り当てられます。

このマージ モジュールのユーザーが、デフォルトのインストール先ディレクトリを上書きできるようにするには、"INSTALLDIR" 設定に **[TARGETDIR]** を入力します。詳細については、「マージ モジュールのインストール先をオーバーライドする」を参照してください。

インストール先フォルダーに関するその他の考慮事項

マージ モジュール プロジェクトに含まれる各コンポーネントにも "インストール先" 設定があります。コンポーネントの "インストール先" 設定は、[一般情報] ビューの "INSTALLDIR" 設定をオーバーライドします。したがって、マージ モジュールの全ファイルがデフォルトでマージ モジュールのインストール先フォルダーにインストールされるようにする場合、すべてのコンポーネントの "インストール先" 設定に [INSTALLDIR] を入力します。

マージ モジュール言語を選択する

プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

[一般情報] ビューの "言語" 設定では、マージ モジュールの言語を選択できます。たとえば、モジュールでユーザー インターフェイスをインストールする場合、そのユーザー インターフェイスを作成する言語を指定します。モジュール言語の設定と一致しないマシン上でモジュールを実行しても、モジュールはインストールされません。

ヒント: *すべての言語でモジュールを実行できるようにするには、言語非依存を選択します。*

マージ モジュールに依存関係を追加する

プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

マージ モジュールのなかには、それ自体で独立したものもありますが、正しく機能するために他のマージ モジュールを必要とするものもあります。たとえば、マージ モジュールには Visual Basic ランタイム ライブラリが必要なことがあります。その場合、そのモジュールを依存関係として新しいマージ モジュールに追加することが必要になります。

ファイルをマージ モジュールに追加する場合、そのファイルがシステムに格納された別のマージ モジュールに含まれていれば、セットアップベストプラクティス違反となります。この場合、マージ モジュールを依存関係として追加する必要があります。

注意: *典型的な基本の MSI プロジェクトまたは InstallScript MSI プロジェクトを作成中、マージ モジュール内で利用可能なファイルを追加すると、自動的にマージ モジュールがプロジェクトに追加されます。マージ モジュールの作成中に別のマージ モジュール内で利用可能なファイルを追加すると、そのマージ モジュールを依存関係として設定するように注意を促すセットアップ ベストプラクティス アラートが表示されます。InstallShield はマージ モジュール プロジェクトに依存関係を作成しません。*

[一般情報] ビューに入力した依存関係情報は、プロジェクトの ModuleDependency テーブルに書き込まれます。このテーブルは、ダイレクト エディターを使って参照できます。

再配布可能ファイル ギャラリーからの依存関係の追加

マージ モジュールが [オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されたパスのいずれかに存在する場合、それは再配布可能ファイルギャラリー内にあります。

タスク ***再配布可能ファイル ギャラリーから 1 つ以上の依存関係を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの依存関係"** 設定で、省略記号ボタン (...) をクリックします。**[モジュールの依存関係]** ダイアログ ボックスが開きます。
3. 作成中のマージ モジュールが必要とするマージ モジュールのチェック ボックスを選択します。
4. **[OK]** をクリックします。

InstallShield が [一般情報] ビューのグリッドへ依存関係を追加します。必要に応じて、依存関係の設定を変更できます。

再配布可能ファイル ギャラリーに存在しない依存関係を追加する

タスク ***マシンにない依存関係を追加するには、次の手順を実行します。***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの依存関係"** 設定で、**[新しいモジュールの依存関係を追加]** ボタンをクリックします。

InstallShield が [一般情報] ビューのグリッドへ依存関係を追加します。必要に応じて、各設定を構成します。

メモ: *"モジュール ID" 設定を入力するとき、InstallShield はそれが正しいかどうかをチェックしませんので、ご注意ください。インストールされたプログラムを実行するまで、失敗するかどうかわかりません。*

マージ モジュールへ除外を追加する

プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

ときどき、作成中のモジュールに関連付けられているモジュールから、特定のマージ モジュールを除外することが必要なことがあります。旧バージョンのモジュールが新しいモジュールと互換性がないためにこのような作業が必要になることがあります。結果的に、新しいモジュールが関連するどれかのセットアップからそのモジュールを除外する必要が出てきます。

再配布可能ファイル ギャラリーからの除外を追加する

マージ モジュールが [オプション] ダイアログ ボックスの [マージ モジュール] タブで指定されたパスのいずれかに存在する場合、それは再配布可能ファイルギャラリー内にあります。

タスク ***再配布可能ファイル ギャラリーからの除外を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの除外"** 設定で、省略記号ボタン (...) をクリックします。**[モジュールの除外]** ダイアログ ボックスが開きます。
3. 作成中のマージ モジュールと互換性のないマージ モジュールのチェック ボックスを選択します。
4. **[OK]** をクリックします。

InstallShield が [一般情報] ビューのグリッドへ除外を追加します。必要に応じて、除外の設定を変更できます。

再配布可能ファイル ギャラリーに存在しない新しい除外を追加する

タスク ***マシンに存在しない除外を追加するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"モジュールの除外"** 設定で、**[新しいモジュールの除外を追加]** ボタンをクリックします。

InstallShield が [一般情報] ビューのグリッドへ除外を追加します。必要に応じて、各設定を構成します。

メモ: *"モジュール ID" 設定を入力するとき、InstallShield はそれが正しいかどうかをチェックしませんので、ご注意ください。インストールされたプログラムを実行するまで、失敗するかどうかわかりません。*

マージ モジュールをマージ モジュール プロジェクト内からリポジトリにパブリッシュする

プロジェクト: *この情報は、マージ モジュール プロジェクトに適用します。*

Windows Installer ベースのインストールで再利用可能な、または他のユーザーと共有可能なマージ モジュールを設計中の場合、それをリポジトリへパブリッシュすることができます。InstallShield では、ビルドの度にマージ モジュールがリポジトリへパブリッシュされるよう、リリースを構成することが可能です。

タスク ***マージ モジュール プロジェクト内部から、マージ モジュールをリポジトリへパブリッシュするよう設定するには、次の手順に従ってください。***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、設定を行うリリースを選択します。

3. **[イベント]** タブをクリックします。
4. **"マージ モジュールのパブリッシュ"** 設定で、**[リポジトリにパブリッシュ]** オプションを選択します。
5. **"リポジトリ名"** 設定で、適切なリポジトリを選択します。アクセス可能なすべてのリポジトリは、この設定にリストされます。
6. **"表示名"** 設定に、リポジトリにパブリッシュするマージ モジュールの名前を指定します。この名前は、[再配布可能ファイル] ビューでマージ モジュールの名前として表示されます。
7. **"発行者"** 設定に、発行者の名前を入力します。
8. **"説明"** 設定に、マージ モジュールの説明を入力します。

選択したリリースをビルドすると常に、InstallShield によって指定されたリポジトリへマージ モジュールがパブリッシュされます。

ビルド時にマージ モジュールがインストールへ組み込まれます。リポジトリ マージ モジュールに変更を加えてからリポジトリへ再パブリッシュすると、プロジェクトのインストールが次回再ビルドされたときに、古いバージョンのリポジトリマージ モジュールを持つプロジェクトはすべて更新されます。

InstallScript オブジェクトを作成する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield では、独自の InstallShield オブジェクトを作成して、他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。オブジェクトによって、InstallShield の以前のバージョンにあったテンプレートが置き換えられます。

メモ: *作成するオブジェクトは、InstallShield X 以降、InstallShield DevStudio、または InstallShield Professional 6.1 以降でのみ利用することができます。*

オブジェクトの作成は、標準インストール プロジェクトの作成と多少異なります。唯一の違いはスコープの重要性です。オブジェクトを作成するとき、オブジェクトのスコープがその利用目的に対して最適な状態にして下さい。オブジェクトのスコープはその有用性に影響します。スコープが狭すぎるとオブジェクトがニーズを十分に満たさないものになることがあります。スコープが広すぎるとオブジェクトが厄介で実用性に欠けるものになることがあります。

次のセクションでは、オブジェクトの作成プロセスの手順が説明されています。

オブジェクトの作成

現在、オブジェクトの作成方法には 2 通りあります。[新規のプロジェクト] ダイアログ ボックスから新規の InstallScript Object プロジェクトを作成するか、既存の InstallScript プロジェクトを InstallScript オブジェクトに変換することができます。

タスク ***InstallScript プロジェクトを InstallScript オブジェクトに変換するには、以下の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. プロジェクトのコピーの作成先となるフォルダーを選択します。新しいオブジェクトを元のプロジェクトと同じフォルダーに保存することはできません。
3. **[ファイル名]** ボックスで、新規プロジェクトの名前を入力します。
4. **[ファイルの種類]** リストで、**[InstallScript オブジェクト プロジェクト (*.ism)]** を選択します。
5. **[保存]** をクリックします。

指定した名前を持つオブジェクト プロジェクト ファイル (..ism ファイル) とすべてのプロジェクトのサブフォルダーが指定の場所に作成されます。プロジェクトが InstallShield で開きます。

オブジェクトのデザイン

オブジェクトの目的は、別個の機能をインストールするためです。たとえば、アプリケーションを実行するためには、最新バージョンの DirectX をインストールする必要がある場合があります。必要なすべてのファイルを個別にインストール プロジェクトに含めるかわりに、InstallShield に付属の DirectX オブジェクトを含めることができます。この同じオブジェクトに、Visual Basic ランタイム .dll ファイルや、MDAC サポートが含まれていると想定してください。オブジェクトが、ニーズに対して大きすぎます。必要のないファイルをインストールし、インス

トール プロセスの所要時間がかかるだけです。したがって、オブジェクトをできるだけ合理化する必要があります。4 つの異なるテクノロジを1つのオブジェクトに含めないでください。4 つの別のプロジェクトを作成するようにしてください。

オブジェクトは従来のインストールと同じ構造をとりますが、オブジェクトはセットアップの種類をサポートしない点で、大きく異なります。オブジェクトの構成の最高レベルにあるのは機能です。機能の下にはコンポーネントが続きます。さらに、オブジェクトはシェル オブジェクトもサポートしていません。オブジェクトからシェルオブジェクトを作成する場合は、スクリプトから行う必要があります。

詳細については、「オブジェクトの設計」を参照してください。

オブジェクトのビルド

オブジェクトのビルドは、インストール プロジェクトのビルドに非常によく似ています。この作業を完了する最も簡単な方法は、リリース ウィザードを使うことです。オブジェクトのビルドとインストール プロジェクトのビルドの主な違いは、オブジェクトのメディアの種類を指定する必要がないことです。ウィザードでは、CD-ROM 形式のメディアのみ作成できます。

詳細については、「ユーザー インターフェイスからオブジェクトをビルドする」を参照してください。

オブジェクトのテスト

どのようなソフトウェア作業においても、一般にリリースする前にソフトウェアをテストすることが重要です。オブジェクトのテストは簡単です。標準インストール プロジェクトを作成してから、オブジェクトを追加し、ビルドして実行します。すべてが予定通りに実行されると、次に、オブジェクトをプロジェクトに含めます。予定通りに行かなければ、デバッグする必要があります。「オブジェクトのテストとデバッグ」を参照してください。(オブジェクトプロジェクトがあり、関連したインストール プロジェクトを InstallShieldの別のインスタンスで開いて、オブジェクトを変更および再ビルドした場合、それらの変更を統合するには関連したインストール プロジェクトを閉じてから再度開く必要があります。)

オブジェクトの配布

ここで使用されている*オブジェクトの配布*とは、そのオブジェクトをインストールに含めて、ターゲットマシンでそのオブジェクトからファイルをインストールすることではありません。ここで言う*オブジェクトの配布*とは、オブジェクトを他のユーザーのマシンにおいてインストール プロジェクトで使用できるようにするインストールを作成することを意味します。オブジェクト内に含まれているロジックはインストールしたくないので、通常行うようにロジックをコンポーネントにただ追加することはできません。オブジェクトを簡単に登録するには、InstallShield オブジェクト インストーラー オブジェクトを使用することができます。このオブジェクトを機能と関連付ける際、配布するローカル ギャラリーからオブジェクトを選択するよう求めるプロンプトが表示されます。

詳細については、「オブジェクトの配布」を参照してください。

オブジェクトの設計

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのデザインとはコンポーネントとファイル グループをいかに配置するかを意味し、オブジェクトの実際の内容を扱うこの作業は、安定した使用可能なオブジェクトを実現するために克服しなくてはならない最大の難関です。

オブジェクトのデザインには、いくつかの重要点があります。それぞれを以下で説明します。

オブジェクトのプロジェクト設定

標準のインストール プロジェクトとオブジェクトの違いは、そのプロジェクト設定にあります。オブジェクト作成用のタブは、[プロジェクトの設定] ダイアログ ボックスで利用可能です。これらのタブで、オブジェクトの開発言語、表示名、オブジェクトに関連付けられているアイコンとヘルプトピックを定義することができます。また、作成するオブジェクトに、好みのウィザード インターフェイスを選択することみできます。

詳細については、「オブジェクトのウィザードの設計」を参照してください。

コンポーネントの管理

プロジェクトは、時間をかけて適切に各機能に分割することが重要です。機能を設計する場合、機能のローカリゼーションや条件選択などの状況を考慮に入れる必要があります。詳細については、「オブジェクト用機能の設計」を参照してください。

プロパティとメソッド

プロパティとメソッドを使うと、デザイン時と実行時に、ユーザーがオブジェクトと相互作用することができます。詳細については、「プロパティの作成」および「メソッドの作成」を参照してください。

デザインタイム ウィザードの作成

オブジェクトにプロパティを含める場合、オブジェクトのユーザーが必要に応じてプロパティを簡単に設定できるように、グラフィカル インターフェイスを提供すると良いでしょう。詳細については、「オブジェクトのウィザードの設計」を参照してください。

ランタイム ユーザー インターフェイスの作成

作成したオブジェクトで、このオブジェクトを含むインストールのエンドユーザーにダイアログを表示できます。オブジェクトのランタイム ユーザー インターフェイスの作成と、オブジェクトを含むインストールからのインターフェイスの呼び出しについての詳細は、「オブジェクトの実行時のユーザー インターフェイスを作成する」を参照してください。

オブジェクトのデザイン環境をローカライズする

InstallShield インターフェイスは英語と日本語にローカライズされています。オブジェクトをこれらの言語にローカライズする方法についての詳細は、「オブジェクトのデザインタイム環境のローカライズ」を参照してください。

オブジェクトのランタイム動作をインターナショナル化する

アプリケーションを複数言語で実行するようにしている場合は、インターナショナル化されたオブジェクトを準備する必要があります。言語固有のファイルをオブジェクトと一緒にインストールすることは、通常のインストールとは若干異なります。詳細については、「インターナショナル化のためのオブジェクトの作成」を参照してください。

実行時に、オブジェクトとインストールとの互換性を持たせる

オブジェクトはオブジェクトを含むインストールによってサポートされている、すべてのオペレーティング システムと言語をサポートしています。詳細については、「実行時に、オブジェクトとインストールとの互換性を持たせる」を参照してください。

オブジェクト用機能の設計

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

機能のデザインの良し悪しは、作業全体に影響するため、注意が必要です。機能のデザインに落ち度がある場合、オブジェクトをインターナショナル化したり、ファイルを選択してオブジェクトにビルドしたり、または条件を付けてファイルをインストールするなどの作業をより困難なものにします。このセクションでは、オブジェクトの機能レイアウトを正確に効率良くデザインするためのガイドラインを示します。

オブジェクトの機能を選択してインストール

通常、オブジェクトの機能の選択について条件を設定します。たとえば、DCOM をインストールするオブジェクトを作成する場合、まず、そのターゲット システムがすでに DCOM をインストールしているかどうかを確認する必要があります。DCOM がすでにインストールされている場合、新たにオブジェクトを作成する作業は時間の無駄であり、また面倒な問題を起こす可能性があります。オブジェクトの機能を、オブジェクトを含むプロジェクトに悪影響を与えずに、条件を付けて選択するには、次のガイドラインに従ってオブジェクトを設計する必要があります。

オブジェクトはその機能を選択しませんが、必要な場合に機能の選択を解除します。(オブジェクトの機能はデフォルトで選択されます。)使用するオブジェクトが 1 つ、または複数の機能を選択する場合、選択する機能がすでにエンドユーザーやプロジェクトの内部ロジックによって選択を解除されていても、そのオブジェクトを含む機能が選択されます。

しかし、オブジェクトによる機能の選択解除は、ある問題を引き起こす可能性もあります。下記にその問題と解決法について示します。

プロジェクトに2つの主機能があるとします(機能 A と B)。B には、サブ機能はありませんが、機能 C をもつオブジェクトがあります。InstallScript エンジンは、子機能が選択されていない時、自動的に機能を選択解除します。したがって、機能 C が選択されていない時、DCOM オブジェクトは選択されません。DCOM オブジェクトが選択されていない場合、機能 B も選択されません。メイン機能に関連付けられたコンポーネントがある場合、この動作が原因で問題が生じることがあります。この例では機能 B がこれに該当します。コンポーネントを含む機能が選択されていないため、直接関連付けられたコンポーネントもインストールされません。

DCOM が選択されているかどうかに関わらずコンポーネントが選択されるようにするには、オブジェクト内にサンプル機能機能を作成します。機能 C が選択されない場合も、サンプル機能が選択されます。サンプル機能が選択されるため、機能 B も選択され、そのコンポーネントと共にインストールされます。

言語依存関係のインストール

複数言語をターゲットにするインストールを開発するには、同じファイルの別の言語バージョンを使用することがよくあります。これらのファイルに共通する名前がついていると、InstallShield でインストールをビルドして、すべてのインストール ファイルをメディアにコピーする際に、問題が発生する可能性があります。リリース ウィザードを実行する際に、機能内のすべてのファイルをコピーし、リリース上の同じディレクトリに保管します。同じ名前が付いた 2 つのファイルが別々のコンポーネントに存在する場合、その 2 つのコンポーネントが同じ機能に属するとそのうちの 1 つが上書きされてしまいます。この問題を解決するには、ターゲットとする各言語の機能を別に作成する必要があります。詳細については、「インターナショナル化のためのオブジェクトの作成」を参照してください。

オブジェクト ファイルの管理

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトをビルドする際には、そのすべてのファイルは、圧縮されていない形式でリリースへ送られます。その結果、同じ名前をもつ複数のファイルを含むオブジェクトが問題を引き起こすことがあります。同じ名前のファイルが同じ機能内にある場合は、それらのうちの一つだけがリリースへ送られることになります。同じ名前をもつ残りのファイルは、リリースへ送られた最後のファイルによって上書きされてしまいます。

同じ名前のファイルをもつ主な理由は、複数の言語をサポートするためです。その場合、サポートする各言語について別々の機能を作成することを推奨します。詳細については、「インターナショナル化のためのオブジェクトの作成」を参照してください。

問題の防止をより確実にするためには、すべての機能について CD-ROM フォルダーを定義します。CD-ROM フォルダーを定義することで、各機能のファイルが送られるリリース上の場所を指定します。CD-ROM フォルダーを定義するには、[機能] ビューへ移動し、"CD-ROM フォルダー" プロパティでフォルダーの名前を入力するだけです。

同じ名前があるファイルをもつ機能には、CD-ROM フォルダーを定義しなければなりません。(例: Setup.inx というコンパイル済みのスクリプト ファイル、または [サポート ファイル/ビルドード] ビューの [アドバンス ファイル] ノードの [Disk1] サブノードに配置したファイル)

オブジェクトのウィザードの設計

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトにプロパティを含める際、オブジェクトのユーザーにインターフェイスを提供して、そのプロパティを変更できるようにすることをお勧めします。InstallShield では、作成したプロパティに基づいてストック ウィザードを作成したり、Visual Basic や Visual C++ を使用して独自のウィザードを作成できます。

InstallShield ストック ウィザード

InstallShield ストック ウィザードは、オブジェクトの社内使用に最適です。ストックウィザードには、すべての読み取り専用プロパティが表示され、読み書きが可能なプロパティには編集フィールドが表示されます。

ストック ウィザードにはいくつかの制限がありますが、カスタム ウィザードを使用して回避することができます。たとえば、ストック ウィザードをご自分のニーズに合わせてカスタマイズすることはできません。表示するプロパティを選択したり、ウィザードの表示方法をカスタマイズすることもできません。ウィザード用にカスタマイズしたインターフェイスを作成する場合、Visual C++ や Visual Basic で独自のウィザードを作成する必要があります。ストック ウィザードのもう 1 つの欠点は、配列プロパティをサポートしていないことです。このように、ストックウィザードにはいくつかの制限がありますが、簡単に実装および使用できます。

カスタム ウィザード

ウィザードによりプロフェッショナルな印象を持たせたり、機能性を向上、または表示をカスタマイズする必要がある場合、Visual C++ または Visual Basic を使用して独自のウィザードを作成します。これらのアプリケーションを InstallShield インストールを実行する前にインストールした場合、InstallShield オブジェクト ウィザードを作成する機能が追加されます。InstallShield をインストールした後にこれらのプログラムをインストールした場合は、InstallShield インストールをメンテナンス モードで実行し、ウィザード機能を Visual Studio アプリケーションに追

加する必要があります。Visual C++ で InstallShield オブジェクト ウィザードを作成するには、[ファイル] メニューから [新規作成] を選択します。その後、プロジェクトの種類のリストから [InstallShield オブジェクト ウィザード] を選択します。Visual Basic の場合は、[新規プロジェクト] ダイアログ ボックスから [InstallShield オブジェクト ウィザード] を選択します。

Visual C++ で InstallShield オブジェクト ウィザードを作成すると、VC ウィザード フレームワーク プロジェクトをベースとした新しい ATL-MFC が表示されます。Visual Basic の場合は、テンプレートプロジェクトが作成されます。各プロジェクトの種類内のコメント化されたコードは、オブジェクトプロパティの取得方法と保存方法を示します。詳細については、「カスタム オブジェクト ウィザード」を参照してください。

ウィザードの作成が完了したら、コンパイルします。

メモ: *カスタムウィザードがあるオブジェクトを配布する場合、配布インストール プロジェクトのウィザード .dll に依存関係を含めるようにしてください。[セットアップのデザイン] ビューまたは [機能] ビューの InstallShield オブジェクトインストーラー機能の上に配置する機能に必要なファイルを含めてください。スタティック(静的) ウィザード .dll を作成すると依存関係を使用せずに済みます。*

InstallShield ストック ウィザード、またはカスタム ウィザードのどちらを使用するかを指定する

タスク ***InstallShield ストック ウィザードまたは作成したカスタム ウィザードのどちらを使用するかを指定するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. 以下のいずれかを実行します。
   - ストック ウィザードを使用する場合: ***"オブジェクト ウィザード"*** 設定で、**[InstallShield オブジェクト ストック ウィザードを使用する]** を選択します。
   - 独自のカスタム ウィザードを使用する場合: ***"オブジェクト ウィザード"*** 設定で、省略記号ボタン (...) をクリックします。**[マイ カスタム オブジェクトの参照ウィザード]** ダイアログ ボックスが表示されます。オブジェクト用に作成したウィザード ファイルを選択します。

プロジェクトのリリースをビルドするとき、指定したオブジェクトがウィザードに含まれます。

カスタム オブジェクト ウィザード

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクト用に InstallShield ですでに定義されているストック ウィザードではなくて、独自のウィザードを使用する場合、インプロセスの COM サーバーを提供する必要があります。このサーバーは、IDevWizard インターフェースを実装しなければなりません。このインターフェースは 、オブジェクトが最初にプロジェクトに含められる際と、[オブジェクト] ビューの [機能] ペインのすでに含められているオブジェクトのコンテキスト メニューから [変更] が選択される際に、InstallShieldによって呼び出されます。このインターフェースはオブジェクトのプロパティの値を表示したり、これらの値を変更するのに便利です。

このインターフェースは、次の 2 つの引数と共に Display メソッドを提供する必要があります。

- オブジェクトのプロパティへのアクセスを提供する IDispatch ポインター。
- 既存のオブジェクトのプロパティに変更を加えられているかどうか、既存のオブジェクトが存在するかしないか、新しいオブジェクトを作成する必要があるかどうかなどを通知するブール戻り値。

C++ 言語の場合、このインターフェースは下記のように定義されます。

```
interface IDevWizard : IDispatch
{
HRESULT Display([in] IDispatch *pObject,
            [out, retval] VARIANT_BOOL* Confirmed);
};
```

サーバーは、ISetupDesignObjectBuild インターフェースも実装できます。InstallShield リリース ビルダーは、このインターフェイスを呼び出して、オブジェクトのどの機能がそのオブジェクトを含むプロジェクトのビルドから除外されるべきかを判別します。(このインターフェイスが実装されない場合、すべてのオブジェクトの機能がビルドに含まれます。)

このインターフェイスは、インストール作成者がオブジェクトのウィザードで選択を実行するため、ビルドされたプロジェクトからこれからインストールされないオブジェクトの機能を除外する際に便利です。たとえば、オブジェクトがオプションで様々なドライバーに対するサポートを提供し、プロジェクトの作成者がそのドライバーのうちの幾つかを含めないと選択する場合、そのドライバーを含む機能をビルドされたプロジェクトから除外することができます。

このインターフェースは、実装されている場合、次の 3 つの引数と共に IsIncluded メソッドを提供する必要があります。

- オブジェクトのプロパティへのアクセスを提供する IDispatch ポインター。
- 機能を指定する文字列値 (例、**Feature 1¥Subfeature 1A**)。
- このコンポーネントをビルドに含める必要があるかどうかを通知するブール戻り値。

C++ 言語の場合、このインターフェースは下記のように定義されます。

```
interface ISetupDesignObjectBuild : IDispatch
{
HRESULT IsIncluded([in] IDispatch *pObject,
                [in] BSTR FeaturePath,
                [out, retval] VARIANT_BOOL* Included);
};
```

メモ: *カスタムウィザードがあるオブジェクトを配布する場合、配布インストール プロジェクトのウィザード .dll に依存関係を含めるようにしてください。[セットアップのデザイン] ビューまたは [機能] ビューの InstallShield オブジェクトインストーラー機能の上に配置する機能に必要なファイルを含めてください。スタティック(静的)ウィザード .dll を作成すると依存関係を使用せずに済みます。*

オブジェクトのデザインタイム環境のローカライズ

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield インターフェイスでローカライズする各言語について、別々の表示名、短い名前、.htm ファイル、およびオブジェクトのアイコンを指定できます。このローカライズは、[プロジェクトの設定] ダイアログ ボックスの [言語固有オブジェクト プロパティ] タブで行います。

[言語固有オブジェクト プロパティ] タブの最初のオプションは [IDE 言語] リスト ボックスです。サポートする最初の言語を選択し、その言語が英語でない場合は [デフォルトを使用する] ボックスのチェックを外します。[言語固有オブジェクトのプロパティ] タブ の残りを入力します。

サポートしている言語が 1 つだけの場合、このダイアログを閉じることができます。しかし、さらに言語を追加してサポートする場合、その言語をリストから選択して、それに必要な情報を提供します。

メモ: *独自のウィザードを作成する場合は、サポートする各言語に対して、ウィザードの実行可能ファイルでランタイム文字列を提供します。*

インターナショナル化のためのオブジェクトの作成

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

ターゲットマシンに基づいて様々なファイルをインストールするオブジェクトをデザインする作業は、同様の機能によってインストールをデザインする作業と類似しています。オブジェクトと標準インストール プロジェクトの重要な違いは、オブジェクトをインターナショナル化する作業はインストールをデザインするよりも微妙に複雑であるというところにあります。この違い、つまり、リリースビルダーは非圧縮したオブジェクトのプロジェクト ファイルをディスク イメージに格納するので、インターナショナル化したファイルは機能レベルで処理する必要があります。次のセクションでは、オブジェクトをインターナショナル化するのに必要な手順と方法について説明します。

オブジェクトに言語を追加する

InstallShield の Premier Edition がある場合、[プロジェクト設定] ダイアログ ボックスの [言語] タブ を使用してオブジェクトにサポート言語を関連付けることができます。

言語固有のファイル グループを設計する

インターナショナル化されたオブジェクトを作成する上で最も重要な手順は、使用する言語固有のデータをコンポーネントから分離させることです。この処理をわかりやすいように例を示して説明します。英語、フランス語、ドイツ語の 3 つの言語をサポートし、そのうち英語をデフォルトの言語にする場合、少なくともコンポーネントが 4 つ必要になります。最初のコンポーネントには英語のみのファイルが格納されます。このコンポーネントの言語プロパティでは、英語と間接的にサポートしている言語すべてを設定する必要があります。この操作を実行することにより、オブジェクトを使用可能にして、英語、フランス語、ドイツ語以外の言語でもインストールできるようになります。ターゲットマシンがサポートされる 3 言語 (英語、フランス語、ドイツ語) 以外の言語で稼動している場合、英語のファイルがインストールされます。

フランス語とドイツ語のコンポーネントには、単純にそれぞれのコンポーネントに対してフランス語とドイツ語のファイルを追加します。フランス語のコンポーネントの言語を フランス語と設定し、ドイツ語のコンポーネントの言語を ドイツ語と設定します。

最後に、使用する言語非依存関係を別のコンポーネントに追加します。このコンポーネントを言語非依存としてマークします。4 つのファイルグループのレイアウトを下記の表に示します。

テーブル 1 ・ ファイル グループ レイアウトのサンプル

ファイルの言語	コンポーネント言語
英語固有ファイルすべて	英語 + サポートされない言語すべて
フランス語固有ファイルすべて	フランス語
ドイツ語固有ファイルすべて	ドイツ語
言語非依存関係すべて	言語非依存

言語固有機能を設計する

オブジェクトの言語固有データを含める場合、ターゲットする言語のそれぞれに対する機能を作成する必要があります。実際には機能の言語は定義できないことから、これは不必要な手順のように見えます。しかし、ディスクイメージ内のファイルの保管形態上、コンポーネントを 複数の言語から 1 つの機能に追加できなくなっています。リリース ウィザードを実行する際に、機能内のすべてのファイルをコピーし、リリース上の同じディレクトリに保管します。同じ名前が付いた 2 つのファイルが別々のコンポーネントに存在する場合、その 2 つのコンポーネントが同じ機能に属するとそのうちの 1 つが上書きされてしまいます。この問題を解決するには、ターゲットとする各言語の機能を別に作成する必要があります。次に、対応するコンポーネントを追加します。

実行時に、オブジェクトとインストールとの互換性を持たせる

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトは、オブジェクトを含むインストールによってサポートされている、すべてのオペレーティング システムと言語をサポートしています。オペレーティング システムが一致しないと、インストールが正しく実行されません。言語が一致しないと、インストールのエンドユーザーが混乱します。そのような不一致を避けるために、以下の 1 つを実行します:

- InstallShield でサポートされているすべてのオペレーティング システムと言語のオブジェクトにサポートを追加する。オブジェクト プロジェクトにオペレーティング システム サポートを追加するには、[一般情報] ビューに移動して、"プラットフォーム フィルター" 設定を構成します。オブジェクト プロジェクトに言語サポートを追加するには、「インターナショナル化のためのオブジェクトの作成」を参照してください。
- インストール作成者にサポートされているオペレーティング システムと言語を通知する文書をオブジェクトに含める。これを行うには、必要な情報を含む .htm ファイルをまず作成し、[プロジェクト] メニューで [設定] をクリックします。[言語非依存オブジェクト プロパティ] タブをクリックして、"HTML ヘルプ" フィールドで .htm ファイルを指定します。

プロパティとメソッド

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

定義として、プロパティはデータ構造に関連付けられている属性を記述するのに使用され、メソッドは、オブジェクトがメッセージを受け取ったときに実行されるプロシージャのことを指します。

インストールの異なる側面に関連付けられたプロパティは、InstallShield のどの箇所でも、追加、作成および設定することができます。プロパティは、オブジェクト スクリプトで役に立つことがあります。

メソッドを使用することにより、プロパティと同様、オブジェクトがそれを含むインストールとインタラクトできるようになります。実際的な目的に対するメソッドとは関数のことです。メソッドと関数の違いは、メソッドはメソッド キーワードで宣言されるということだけです。メソッドを使用すると、オブジェクトの関数を他のオブジェクトやインストールで使用できるようにすることができます。

プロパティの作成

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

プロパティは、オブジェクト スクリプトで役に立つことがあります。オブジェクト スクリプトでは、プロパティを使うと、オブジェクトのユーザーは、InstallShield オブジェクト ウィザード によるデザインを行っている最中、またはスクリプトによってパラメーターを渡して実行している最中に特定のオブジェクトの設定を変更することができます。共通プロパティには、実行時やデザイン 時にオブジェクトからインストールへのエラー メッセージの転送、ユーザー名やオブジェクトに必要な情報の取得などがあります。プロパティの使用例については、[オブジェクト] ビューにある InstallShield オブジェクトのインライン ヘルプを参照してください。

プロパティについて以下の点にご注意ください。

- Boolean プロパティの値がオブジェクトスクリプトまたはそのオブジェクトを含むインストールのスクリプトによって TRUE に設定されている場合、後に続くプロパティの値の以降のチェックには 1 ではなく -1 の値が表示されます。
- オブジェクトとインストール スクリプトの間にある LIST 変数を渡すことはできませんが、文字列配列をパスすることは可能です。

プロパティの作成

タスク

プロパティを作成するには、以下の手順に従ってください。

1. **InstallScript** ビューを開きます。
2. **InstallScript** エクスプローラーで、**[プロパティ]** を右クリックして、**[新しいプロパティの追加]** をクリックします。**[新しいプロパティの追加]** ダイアログ ボックスが開きます。
3. プロパティの設定を定義します。
4. **[OK]** をクリックします。

[新しいプロパティの追加] ダイアログ ボックスにより、以下のスクリプトが追加されます。

- プロパティを宣言する構文:

  property(<プロシージャ>) <プロパティ データ型> <プロパティ名> ( );

  <プロシージャ> は読み取り専用プロパティの場合 **get**、書き込み専用プロパティの場合 **put** で、読み取り/書き込みプロパティの場合 **get,put** です。例:

  property(get,put) STRING MyProperty ( );

  プロパティ名の後のかっこ内に引数のデータ型を追加して、プロパティの引数を指定することができます。
  例:

  property(get,put) STRING MyProperty (NUMBER);

  プロパティの宣言に引数のデータ型を追加した場合、プロパティのプロシージャ関数の定義に対応する引数の変数を追加する必要があります。

- スクリプト内にプロパティの値を保存する変数の宣言例:

  STRING m_strMyProperty;

  [新しいプロパティの追加] ダイアログ ボックスで ([配列] チェック ボックスを選択して) 配列プロパティを指定する場合、プロパティの値を保存する VARIANT 型の変数と、プロパティの値を初期化するために **InitProperties** 関数ブロックで使用するサイズ指定のない配列変数の 2 つの変数を宣言します。例:

  STRING arrayMyArrayProperty();
  VARIANT m_arrayMyArrayProperty;

- プロパティの値を保存する変数を初期化する、**InitProperties** 関数ブロック内のステートメント。例:

  m_strMyProperty = "デフォルト値";

  [新しいプロパティの追加] ダイアログ ボックスで ([配列] チェック ボックスを選択して) 配列プロパティを指定した場合、プロパティの値を保存する変数は、配列変数の値に割り当てられます。例:

  m_arrayMyArrayProperty = arrayMyArrayProperty;

  配列プロパティの要素をヌルやゼロ以外の値に初期化するには、上記ステートメントの前に配列変数の要素の割り当てステートメントを追加します。例:

  ```
  /* サイズなしと宣言された配列変数のサイズを変更する。*/
  Resize ( arrayMyArrayProperty , 3 );

  /* 配列変数の要素に値を割り当てます。*/
  arrayMyArrayProperty(0) = "zero";
  arrayMyArrayProperty(1) = "one";
  arrayMyArrayProperty(2) = "two";

  m_arrayMyArrayProperty = arrayMyArrayProperty;
  ```

- プロパティ バッグ オブジェクトに保存されているプロパティの値を取得する、**ReadProperties** 関数ブロック内での関数呼び出し。この呼び出しの構文は次のとおりです。

  <Read*xxxx*Property> (PropertyBag, "<プロパティ名>", <プロパティ値変数>);

  <Read*xxxx*Property> は、プロパティの値のデータ型に適した Read*xxxx*Property 関数です。例:

  ReadStringProperty(PropertyBag, "MyProperty", m_strMyProperty);

- プロパティ バッグ オブジェクトにプロパティの値を保存する、**WriteProperties** 関数ブロック内での関数呼び出しこの呼び出しの構文は次のとおりです。

  <Write*xxxx*Property> (PropertyBag, "<プロパティ名>", <プロパティ値変数>);

  <Write*xxxx*Property> は、プロパティの値のデータ型に適した Write*xxxx*Property 関数です。例:

  WriteStringProperty(PropertyBag, "MyProperty", m_strMyProperty);

- 関数ブロックは、プロパティのプロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザイン時のウィザード (該当する場合) によってプロパティの値が読み取られたときに実行されるアクションを定義します。

  get_<プロパティ名> 関数は読み取り専用または読み書き可能なプロパティに対して定義し、**put_<プロパティ名>** 関数は書き込み専用または読み書き可能なプロパティに対して定義します。例:

  ```
  function STRING get_MyProperty()
  begin
     return m_strMyProperty;
  end;

  function void put_MyProperty(newVal)
  begin
     m_strMyProperty = newVal;
  end;
  ```

  (引数のデータ型をプロパティ名の後のカッコ内のプロパティ宣言に追加することによって)プロパティの引数を指定する場合、プロパティのプロシージャ関数の定義に対応する引数変数を追加する必要があります。例:

  ```
  function STRING get_MyProperty( szAddedArgument )
  function void put_MyProperty( szAddedArgument, newVal )
  ```

  put_ 関数に引数を追加する場合、newVal 引数の前に配置してください。newVal 引数は、プロジェクトの呼び出しの割り当てステートメント、またはオブジェクトのデザインタイム ウィザード (該当する場合) によりプロパティに書き込まれた値を取ります。

メモ: *get_ および put_ 関数の名前を変更しないでください。変更した場合、スクリプトが正常に機能しません。*

構造評価プロパティの作成

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

値がデータ構造のプロパティを作成するには、オブジェクト スクリプトに次の内容を追加します。

- データ構造のグローバル宣言。以下に例を示します。

  ```
  typedef STRUCT
  begin
     NUMBER Mem1;
     NUMBER Mem2;
  end;
  ```

- データ型が OBJECT である読み取り専用プロパティの宣言。(プロパティに値を書き込むには、メソッドを定義します。)例:

  property(get) OBJECT StructProp();

  この宣言をオブジェクト スクリプトの Properties Section ブロックに配置します。

- オブジェクト スクリプト内にプロパティの値を格納する変数の宣言例:

  STRUCT m_structStructProp;

  この宣言をオブジェクトスクリプトの "ローカル変数セクション" ブロックに配置します。

- データ構造の各メンバーに対応する適切なデータ型の引数を持つメソッドの宣言。上記で定義した STRUCT 構造について、適切なメソッド宣言の例を以下に示します。

  method NUMBER SetStructProp(NUMBER, NUMBER);

  この宣言をオブジェクト スクリプトの "メソッド セクション" ブロックに配置します。

- プロパティの値を保存する変数のメンバーを初期化する、**InitProperties** 関数ブロック内のステートメント。例:

  m_structStructProp.Mem1 = 3;
  m_structStructProp.Mem2 = 5;

- プロパティ バッグ オブジェクトに保存されているプロパティの値を取得する、**ReadProperties** 関数ブロック内での関数呼び出しとステートメント関数呼び出しは次のような構文を持っています:

  <Read*xxxx*Property> (PropertyBag, "<メンバー値のストレージ名>", <メンバー値変数>);

  <Read*xxxx*Property> は、プロパティ メンバーの値のデータ型に適した Read*xxxx*Property 関数です。例:

  ReadNumberProperty(PropertyBag, "StructProp_Mem1", nStructPropMem1);

  このような各関数呼び出しについては、関数が取得する値と同等のプロパティ変数のメンバーを設定するステートメントを下に追加します。例:

  m_structStructProp.Mem1 = nStructPropMem1;

- プロパティ バッグ オブジェクトにプロパティメンバーの値を保存する、**WriteProperties** 関数ブロック内での関数呼び出し関数呼び出しは次のような構文を持っています:

  <Write*xxxx*Property> (PropertyBag, "<メンバー値のストレージ名>", <プロパティ変数メンバー>);

  <Write*xxxx*Property> は、プロパティ メンバーの値のデータ型に適した Write*xxxx*Property 関数です。例:

  WriteNumberProperty(PropertyBag, "StructProp_Mem1", m_structStructProp.Mem1);

- get_<プロパティ名> 関数ブロックは、プロパティの get プロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザインタイム ウィザード (該当する場合) により、プロパティの値が読み取りまたは書き込まれた際に実行されるアクションを定義します。例:

  ```
  function OBJECT get_StructProp()
  begin
     return m_structStructProp;
  end;
  ```

- 宣言されたメソッドを定義する関数ブロック。このメソッドはメソッドに渡された引数と等しいプロパティ変数のメンバーを設定します。例:

```
function NUMBER SetStructProp( nMem1, nMem2 )
begin
    m_structStructProp.Mem1 = nMem1;
    m_structStructProp.Mem2 = nMem2;
end;
```

オブジェクトとインストールの間で文字列配列を渡す

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトとインストール間で文字列配列を渡すには、以下の手順を実行します。

- 以下のオブジェクト スクリプトを追加します。
  - データ型が 異なる読み書きのプロパティの宣言。例:

    ```
    property(get,put) variant Test();
    ```

    この宣言をオブジェクト スクリプトの Properties Section ブロックに配置します。
  - オブジェクト スクリプト内にプロパティの値を保存する変数に必要な宣言。

    ```
    STRING arrayTest();
    variant m_arrayTest;
    ```

    この宣言をオブジェクト スクリプトの Local Variables Section ブロックに配置します。
  - プロパティの値を保存する変数のメンバーを初期化する、InitProperties 関数ブロック内のステートメント。例:

    ```
    m_arrayTest = arrayTest;
    ```
  - プロパティバッグオブジェクトに保存されている値を取得する、ReadProperties 関数ブロック内での関数呼び出し。例:

    ```
    ReadArrayProperty(PropertyBag, "Test", m_arrayTest);
    ```
  - プロパティ バッグ オブジェクトに値を保存する、WriteProperties 関数ブロック内での関数呼び出し。例:

    ```
    WriteArrayProperty(PropertyBag, "Test", m_arrayTest);
    ```
  - 関数ブロックは、プロパティの get プロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザインタイム ウィザード (該当する場合) により、プロパティの値が読み込まれた時に実行されるアクションを定義します。例:

    ```
    function variant get_Test()
    begin
        return m_arrayTest;
    end;
    ```
  - 関数ブロックは、プロパティの put プロシージャを定義します。つまり、オブジェクトを含むプロジェクト、またはオブジェクトのデザインタイム ウィザード (該当する場合) により、プロパティの値が書き込まれた時に実行されるアクションを定義します。例:

    ```
    function void put_Test(newVal)
    begin
        m_arrayTest = newVal;
    ```

```
end;
```

- 以下のインストールスクリプトを追加します。
  - オブジェクトからの文字列配列の読み取りやオブジェクトへの書き込みをする各関数ブロックでは、オブジェクト変数の宣言や文字列配列。例:

    ```
    OBJECT oObject;
    string szTest(3);
    ```

  - オブジェクトへのリファレンスを取得するための GetObject への呼び出し。例:

    ```
    set oObject = GetObject("Object");
    if (!IsObject(oObject)) then
       MessageBox( "オブジェクトのリファレンスを取得できませんでした。", INFORMATION );
       abort;
    endif;
    ```

  - "オブジェクトの文字列配列" プロパティの値を設定するには、以下のコードを使用します:

    ```
    oObject.Test = szTest;
    ```

    "オブジェクトの文字列配列" プロパティの値を取得するには、以下のコードを使用します:

    ```
    szTest = oObject.Test;
    ```

オブジェクトのプロパティにアクセスする

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトをコンパイルした後にプロパティを変更する方法は 2 つあります。最初の方法では、オブジェクトがインストール プロジェクトに追加された時に変更を行います。オブジェクトのウィザードインターフェイスを提供する場合、インストールのデザイン タイム中に読み書きのプロパティが変更されます。2 番目の方法では、実行時にスクリプトを使用してプロパティを変更します。オブジェクトのユーザーが実行時にプロパティにアクセスできるようにするには、オブジェクトのヘルプトピックにこれらのプロパティをドキュメント化する必要があります。これにより、ユーザーは必要に応じてプロパティを変更できます。

InstallShield インターフェイス からプロパティへアクセスする

インストール プロジェクトでオブジェクトのプロパティにアクセスすると、そのオブジェクトのスクリプトの一部を InstallShield インターフェイスで実行することになります。たとえば、オブジェクトをインストール プロジェクトと関連付ける場合、そのオブジェクトのプロパティは **get_PropertyName** 関数によって読み込まれます。この関数は、プロパティを含むすべてのオブジェクトに組み込まれています。オブジェクトウィザードでオブジェクトのプロパティを変更する場合は、**put_***PropertyName* 関数が呼び出されます。ここで、*PropertyName* は変更するプロパティの名前になります。

スクリプトを使用してプロパティにアクセスする

オブジェクトのプロパティにアクセスするには、まずそのオブジェクトへの参照を取得します。その後、アクセスするオブジェクトのプロパティを指定します。以下のコードは、オブジェクトのステータス プロパティを返す方法を示します:

```
function OnBegin()
  /*オブジェクトへの参照を保持するオブジェクト変数を宣言します。*/
```

```
    OBJECT oObject;
begin
    /* "カスタム オブジェクト" という名前のカスタム オブジェクトの参照を取得します。*/
      set oObject = GetObject("カスタム オブジェクト");

    /*Status.Number プロパティを確認。*/
    if oObject.Status.Number != OBJ_STATUS_SUCCESS then
        /*セットアップに合わせてエラーに応答。
        たとえば Status.description テキストを表示して
        セットアップを中止できます。*/
        MessageBox ( oObject.Status.Description, SEVERE );
        abort;
    endif;
end;
```

実行時にプロパティを変更する場合、その情報は保存されず、今後のインストールでは使用されません。たとえば、後でインストールをメンテナンス モードで実行すると、最初のインストールで与えられた値はプロパティに保持されません。

Boolean プロパティの値がオブジェクトスクリプトまたはそのオブジェクトを含むインストールのスクリプトによって TRUE に設定されている場合、後に続くプロパティの値の以降のチェックには 1 ではなく -1 の値が表示されます。

ヒント: *InstallScript Object で、単にオブジェクトの現在のステータス (Status.Number の現在の値) を取得するだけのときは、***GetStatus** *関数を使用できます。詳細については、「GetStatus」を参照してください。*

メソッドの作成

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

メソッドは、プロパティと同様に、オブジェクトとそれを含むインストールが相互に対応するようにすることができます。実際的な目的に対するメソッドとは関数のことです。メソッドと関数の違いは、メソッドはメソッド キーワードで宣言されるということだけです。メソッドを使うと、オブジェクトの関数を他のオブジェクトやセットアップで使用することができます。

メソッドの作成

タスク ***メソッドを作成するには、以下の手順を実行します。***

1. **InstallScript** ビューを開きます。
2. **InstallScript** エクスプローラーで、**[メソッド]** を右クリックして、**[新しいメソッドの追加]** をクリックします。**[新しいメソッドの追加]** ダイアログ ボックスが開きます。
3. メソッドの設定を定義します。
4. **[OK]** をクリックします。

[新しいメソッドの追加] ダイアログ ボックスにより、以下のオブジェクト スクリプトが追加されます。

- メソッドを宣言する構文

  method <メソッドデータタイプ> <メソッド名> (<引数データタイプ>);

  例:

  method STRING MyMethod(NUMBER, BOOL);

- メソッドを定義する関数ブロック例:

```
function STRING MyMethod( /*NUMBER*/ arg1, /*BOOL*/ arg2 )
begin
end;
```

メソッドを使用する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのメソッドには実行時に限りアクセスできます。オブジェクトのメソッドをインストール プロジェクト内から、または別のオブジェクトから呼び出すことができます。オブジェクトのメソッドにアクセスするには、まずそのオブジェクトの参照を取得します。次のコードはオブジェクトの現在の状態についての説明を表示します:

```
Set oObj = GetObject("ATL 3.0")
MessageBox(oObj.Status.Description, INFORMATION);
```

オブジェクトステータスのプロパティ

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

あらかじめ定義されているものも、ユーザーによって定義されたものも含め、すべての InstallScript オブジェクトは以下のような読み取り専用プロパティを持っています。

- Status
- StatusBase
- StatusStruct

StatusBase と StatusStruct は、高度な知識をもつユーザー向けです。これらのプロパティに関する詳細については、次のセクションを参照してください。

Status

Status プロパティは、次のプロパティをメンバーに持つ構造です。

テーブル 2 - Status プロパティのメンバー

プロパティ	説明
Status.Number	数値のオブジェクト ステータス。

テーブル 2 • Status プロパティのメンバー (続き.)

プロパティ	説明
Status.Description	オブジェクトのステータスを説明する文字列。
Status.szSource	エラーの場合、エラーのソース。通常、オブジェクトの文字列エントリから読み込まれたオブジェクトの名前。
Status.szScriptFile	エラーの場合、エラー発生時に実行されていたスクリプト ファイルの名前。
Status.nScriptLine	エラーの場合、エラー発生時に実行されていたスクリプト行。
Status.nScriptError	エラーの場合、エラーのリターン コード。

オブジェクト スクリプトで、**SetStatusEx** を呼んでこれらの値を設定することができます。オブジェクトのステータスをクエリすると、デフォルトでは最初サブオブジェクトのステータス (あれば) をクエリします。1 つ以上のオブジェクトのサブオブジェクトがエラーを報告 (OBJ_STATUS_SUCCESS より少ない Status.Number プロパティ) すると、最初に失敗したオブジェクトのステータスプロパティが親オブジェクトのステータスとして返されます。ユーザーによって定義されたオブジェクトのスクリプトは、(引数なしで) get_Status 関数を明示的に定義することによってデフォルトの動作をオーバーライドすることができます。

オブジェクトの Status.Number プロパティの値を確認して、オブジェクトを正常にインストールすることが可能かどうか、可能ではない場合、その原因を、判断し、エンドユーザーに Status.Description の値を表示することができます。この確認は、セットアップのどの時点でも行うことができますが、最低、以下の例にあるように、**OnBegin** イベント ハンドラーで成功または失敗したかを確認することをお勧めします。

```
function OnBegin()
    /*オブジェクトへの参照を保持するオブジェクト変数を宣言します。*/
    OBJECT oObject;
begin
    /* "カスタム オブジェクト" という名前のカスタム オブジェクトの参照を取得します。*/
    set oObject = GetObject("カスタム オブジェクト");

    /*Status.Number プロパティを確認。*/
    if oObject.Status.Number != OBJ_STATUS_SUCCESS then
        /*セットアップに合わせてエラーに応答。
        たとえば Status.description テキストを表示して
        セットアップを中止できます。*/
        MessageBox ( oObject.Status.Description, SEVERE );
        abort;
    endif;
end;
```

以下のテーブルには、すべてのオブジェクトに共通な Status.Number および Status.Description の可能な値がリストされています。一部のオブジェクトには、固有のプロパティに可能な値があります。可能な値のリストについては、個別のオブジェクトのヘルプ ページを参照してください(**[オブジェクト]** ビューまたは **[セットアップのデザイン]** または関連機能の **[機能]** ビューでオブジェクトを選択すると表示できます)。

テーブル 3 ・ すべてのオブジェクトに共通な Status.Number および Status.Description の可能な値

oObject.Status.Number	oObject.Status.Description
OBJ_STATUS_SUCCESS	操作は成功しました。
OBJ_STATUS_OSINVALID	ターゲット OS が無効です。
OBJ_STATUS_DISKSPACE	十分なディスク容量がありません。
OBJ_STATUS_LANGUAGESUPPORT	サポートされていない言語です。
OBJ_STATUS_NOTADMINISTRATOR	管理者特権が必要です。
OBJ_STATUS_SETKEY	レジストリキーを設定できません。
OBJ_STATUS_GETKEY	レジストリキーを取得できません。
OBJ_STATUS_SETVALUE	レジストリ値を設定できません。
OBJ_STATUS_GETVALUE	レジストリ値を取得できません。
OBJ_STATUS_SETTARGET	スクリプトで定義されたフォルダー値を設定できません。
OBJ_STATUS_LOADDLL	DLL をロードすることができません。
OBJ_STATUS_FILELOCKED	必要なファイルがロックされました。
OBJ_STATUS_FILENOTFOUND	ファイルが見つかりません。
OBJ_STATUS_FILECOPY	ファイルのコピーに失敗しました。
OBJ_STATUS_FILELAUNCH	ファイルを起動できませんでした。
OBJ_STATUS_SETLOGDATA	セットアップ ログにデータを設定できません。
OBJ_STATUS_GETLOGDATA	セットアップ ログからデータを取得できません。
OBJ_STATUS_INITVARS	変数の初期化ができません。

StatusBase

StatusBase プロパティは、Status プロパティと同じメンバーを持つ構造です。このプロパティは、通常、セットアップがこのプロパティを直接クエリする必要がない特殊なケース（セットアップはオブジェクトから提供されるプロパティを処理してすべてのカスタムステータスをバイパスすることができるため）を除き、カスタマイズされた get_Status メソッドを持つオブジェクトによってのみ使用されます。

StatusBase プロパティを使って、オブジェクトのサブオブジェクトのステータスのチェックなど、オブジェクトのデフォルトのステータス ハンドリングを利用することができます。オブジェクトに カスタム get_Status 関数を作成すると、以下の例にあるように、get_Status 関数内の StatusBaseプロパティを使用して、デフォルトの ステータス ハンドリングを利用することができます。

```
function object get_Status()
    object oThis;
begin
    //ここでカスタム処理を行います。

    //オブジェクトのリファレンスを取得します。
    set oThis = GetObject( "" );

    //これがオブジェクトでない場合は失敗します。
    if( !IsObject( oThis ) ) then
        return NULL;
    endif;

    //標準のステータス情報を返します。
    return oThis.StatusBase;
end;
```

StatusStruct

StatusStruct プロパティは、Status プロパティと同じメンバーを持つ構造です。このプロパティは、通常、インストールがこのプロパティを直接クエリする必要がない特殊なケース（セットアップはオブジェクトから提供されるプロパティを処理してすべてのカスタムステータスをバイパスすることができるため）を除き、カスタマイズされた get_Status メソッドを持つオブジェクトによってのみ使用されます。

StatusStruct プロパティを使って、デフォルトのオブジェクト ステータス ハンドリングをバイパスすることができます。オブジェクトにサブオブジェクト ステータス値を無視させたい場合、次の例にあるように、このプロパティを使用することができます。

```
function object get_Status()
    object oThis;
begin
    //オブジェクトのリファレンスを取得します。
    set oThis = GetObject( "" );

    //これがオブジェクトでない場合は失敗します。
    if( !IsObject( oThis ) ) then
        return NULL;
    endif;

    //ステータス情報を直接返します。
    return oThis.StatusProp;
end;
```

オブジェクトの実行時のユーザー インターフェイスを作成する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

作成したオブジェクトで、このオブジェクトを含むインストールのエンドユーザーにダイアログ ボックスのシーケンスを表示できます。このダイアログのシーケンスを定義するには、オブジェクト スクリプトの UI イベント ハンドラー (OnFirstUIBefore、OnFirstUIAfter、OnMaintUIBefore、および OnMaintUIAfter) を使用してください。(デフォルトでは、これらのハンドラーはダイアログをオブジェクトのプロジェクトで表示しません。)

ダイアログ シーケンスを使ったオブジェクトを含むプロジェクトでは、オブジェクトのダイアログを表示するには 2 つの方法があります。

- **ShowObjWizardPages** 関数を呼び出して、プロジェクトのある時点ですべてのオブジェクトのダイアログ シーケンスを表示します。
- オブジェクトの Show*xxxxx*UI*yyyyy* メソッド (つまり、ShowFirstUIBefore、ShowFirstUIAfter、ShowMaintUIBefore、および ShowMaintUIAfter) をいずれか、またはすべてを選択して、各オブジェクトのダイアログ シーケンスを表示します。

ShowObjWizardPages 関数

ShowObjWizardPages 関数はデフォルトで InstallShield X、InstallShield DevStudio、または InstallShield Professional 6.1 以降を使って作成した InstallScript インストール プロジェクトの各 UI イベント ハンドラーの中で呼び出されます。デフォルトのイベント ハンドラーでのこの関数の利用方法をガイドラインとして参考にしてください。**ShowObjWizardPages** をプロジェクトの UI イベント ハンドラーの中で呼び出すと、インクルードした各オブジェクトの対応 UI コードが表示されます。たとえば、**ShowObjWizardPages** をプロジェクトの OnFirstUIBefore イベント ハンドラーの中で呼び出すと、インクルードした各オブジェクトの対応 OnFirstUIBefore コードが表示されます。オブジェクトがプロジェクトの [コンポーネント] ビューで表示される順番で、インクルードしたオブジェクトの UI イベント ハンドラーが実行されます。

エンドユーザーは、[次に] ボタンと [戻る] ボタンを使用して、インストール ダイアログ シーケンス全体を移動できます。それには、UI イベント ハンドラーコードのデフォルトどおりに各ダイアログ ボックス関数の戻り値 (NEXT または BACK) を変数 nResult に割り当てて、**ShowObjWizardPages** を次の要領で呼び出します。

```
nResult = ShowObjWizardPages( nResult );
```

ShowxxxxxUIyyyyy メソッド

ユーザー作成のすべてのオブジェクトは内部で、ShowFirstUIBefore、ShowFirstUIAfter、ShowMaintUIBefore、および ShowMaintUIAfter のメソッドをサポートしています。オブジェクトの Show*xxxxx*UI*yyyyy* メソッドを呼び出すと、オブジェクトの対応 UI コードが実行されます。たとえば、次の呼び出しは、オブジェクトの OnFirstUIBefore コードを実行します :

```
nResult = oObject.ShowFirstUIBefore(nResult);
```

ShowObjWizardPages 関数と同様に、エンドユーザーのナビゲーションを可能にするには、各ダイアログ関数の戻り値を nResult に割り当て、前述の例のように how*xxxxx*UI*yyyyy* メソッドを呼び出します。

オブジェクト UI イベント ハンドラーのコード

デフォルトのコードを変更して、オブジェクト UI イベント ハンドラーのコードを作成します。このデフォルトのコードは、ダイアログを表示しませんが、エンドユーザーのナビゲーションを可能にするために必要なコード構造を持っています。

- ダイアログを 1 つ表示するには、FIRST_DIALOG か LAST_DIALOG ラベルのすぐ後にダイアログ関数呼び出しを配置します。
- ダイアログを 2 つ表示するには、FIRST_DIALOG ラベルの後に 1 つめのダイアログ関数呼び出しを配置して、LAST_DIALOG ラベルのすぐ後に 2 つめのダイアログ関数呼び出しを配置します。
- ダイアログを 2 つ以上表示するには、FIRST_DIALOG ラベルの後に 1 つめのダイアログ関数呼び出しを配置して、LAST_DIALOG ラベルのすぐ後に 2 つめのダイアログ関数呼び出しを配置します。中間にある各ダイアログに対しては、FIRST_DIALOG と LAST_DIALOG ブロックのようなコードブロック構造で、ダイアログ関数呼び出しを配置します。

上記のそれぞれのケースで、ダイアログ関数呼び出しの戻り値を変数からの戻り値を変数 *nDirection* に割り当てます。

オブジェクトでサポートされていない機能

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトの目的と実装のために、InstallShield インターフェイス内でオブジェクトを作成する際には特定の機能がサポートされていません。これらのサポートされていない機能を以下の一覧に示します。オブジェクトとインストール プロジェクトとの詳しい設計概念の違いについては、「オブジェクトの設計」を参照してください。

テーブル 4 ・ オブジェクトでサポートされていない機能

サポートされていない機能	説明
セットアップの種類	セットアップの種類はオブジェクトではサポートされていません。既存のインストール プロジェクトの外にオブジェクトを作成する場合、セットアップの種類は無視されます。
メディアの種類	オブジェクトを作成する際には、ビルドするメディアの種類を選択することはありません。すべてのオブジェクトは CD-ROM オプションで圧縮されずにビルドされます。
ショートカットとフォルダー	[ショートカット] ビューはオブジェクトではサポートされていません。オブジェクト内にショートカットまたはフォルダーを作成するには、**AddFolderIcon**、**CreateProgramFolder**、または **CreateDir** 関数を使ってスクリプトから作成する必要があります。
サポート ファイル	標準インストールで、[サポート ファイル] ビューでのスプラッシュ クリーン、ビルボード、言語依存、言語非依存、およびアドバンスファイルを指定するための場所。オブジェクトは言語依存と言語非依存のファイルのみを許可します。
製品のレジストリ キー	 ***Windows ロゴ ・*** *InstallShield で作成されたインストールが実行される際には、製品のレジストリ キーが自動的にターゲットマシン上に作成されます。製品のレジストリ キーは、HKLM¥SOFTWARE¥<会社名>¥<製品名>¥<バージョン> に作成され、マイクロソフトのロゴに準拠するために必要です。したがって、作成されたこれらのレジストリ キーが必要な場合は、自分で追加する必要があります。*

オブジェクトのビルド

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのビルドは、本質的には、標準インストールのビルドと同じです。InstallShield ユーザー インターフェイスからオブジェクトをビルドするには、リリース ィザードを使用します。リリース ウィザードでビルドに必要な情報すべてが、プロンプトによって聞かれます。

ユーザー インターフェイスからオブジェクトをビルドする

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

メモ: *オブジェクトは、ライブラリ ファイル* Ifxobject.obl *と* Isrt.obl *を使用してコンパイルする必要があります。[新規プロジェクト] ダイアログ ボックスの InstallScript オブジェクト プロジェクトタイプを使って、または [ファイル] メニューから [名前を付けて保存] コマンドを選択してプロジェクトが作成された場合、これらのファイルは、オブジェクトプロジェクトによって自動的に参照されます。オブジェクト プロジェクトがその他の方法で作成されている場合は、[ビルド] メニューから [設定] を選択します。次に、[コンパイル/リンク] タブのライブラリ (.obl) ボックスに、*Ifxobject.obl *と* Isrt.obl *がリストされていることを確認します。*

オブジェクトのビルドは、本質的には、標準インストールのビルドと同じです。オブジェクトをビルドするには、リリース ウィザードを使用します。リリース ウィザードでビルドに必要な情報すべてが、プロンプトによって聞かれます。リリース ウィザードではオブジェクトプロジェクト用に次のパネルが表示されます。

テーブル 5 ・ リリース ウィザードのオブジェクト固有のパネル

リリース ウィザード パネル	説明
[リリースの指定] パネル	新しいリリースを作成したり、既存のリリースを選択できます。
[一般オプション] パネル	プリプロセッサ変数定義とリリースの詳細プロパティを指定できます。
[プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル	オブジェクトがサポートするオペレーティング システムを選択できます。
[セットアップ言語] パネル	[プロジェクトの設定] プロパティシートの [言語] ページで、複数の言語が選択されている場合に表示されます。オブジェクトでサポートする言語、デフォルトのオブジェクト言語、およびエンドユーザーがオブジェクトを実行する言語を選択できるようにするかどうかを選択します。
[デジタル署名] パネル	オブジェクトにデジタル署名することができます。オブジェクトにデジタル署名をすることで、オブジェクト内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証します。

テーブル 5 • リリース ウィザードのオブジェクト固有のパネル (続き.)

リリース ウィザード パネル	説明
[ポストビルドのオプション] パネル	次回リリースをビルドした後に、ディスク イメージ フォルダーをフォルダーまたは FTP サイトにコピーするか、またはバッチ ファイルを実行します。
[リリース設定の概要] パネル	リリース情報の概要を表示します。 *メモ• オブジェクトをビルドすると、そのオブジェクトは [オブジェクト] ビューに自動追加され、それ以降に作成するすべてのインストールで使用できます。オブジェクトの以前のバージョンがギャラリーに存在する場合、新しくビルドされたオブジェクトで上書きされます。複数のメディアをビルドする場合、最新のメディアが [オブジェクト] ビューにコピーされます。*

コマンドラインからオブジェクトのコンパイルとビルドを行う

プロジェクト• *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

コマンドラインからインストールをコンパイルおよびビルドするのと同じ形式で、コマンドラインからオブジェクトをコンパイルおよびビルドします。コマンドラインからオブジェクトをコンパイルおよびビルドする方法については、「ISCmdBld.exe を使用して、コマンドラインからリリースをビルドする」を参照してください。

オブジェクトのテストとデバッグ

プロジェクト• *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトのテストは標準インストールのテストに類似しています。この 2 つのテストの違いは、オブジェクトは、テストするには標準インストールに含めなくてはならないということです。このため、まずオブジェクトをビルドし、次にそれをインストール プロジェクトに含めてから、インストール プロジェクトをビルドして実行する必要があります。次のデバッグ方法は、オブジェクトを開発する際に適用できます。

メモ• *オブジェクトプロジェクトがあり、関連したインストール プロジェクトを InstallShieldの別のインスタンスで開いて、オブジェクトを変更および再ビルドした場合、それらの変更を統合するには関連したインストール プロジェクトを閉じてから再度開く必要があります。*

スクリプト デバッガー

スクリプト デバッガーて、実行するインストールのコードで手順をすすめることができます。スクリプト デバッガーでオブジェクトをデバッグするには、標準インストール プロジェクトにそのオブジェクトを含め、コンパイルし、デバッグする必要があります。もう 1 つの方法として、オブジェクトを標準インストールとしてデザイン、テスト、デバッグし、そして次の手順に従ってオブジェクトに変換することもできます。

タスク ***オブジェクトを変換するには、以下の手順を実行します。***

1. **[ファイル]** メニューで、**[名前を付けて保存]** をクリックします。**[名前を付けて保存]** ダイアログ ボックスが開きます。
2. プロジェクトのコピーの作成先となるフォルダーを選択します。新しいオブジェクトを元のプロジェクトと同じフォルダーに保存することはできません。
3. **[ファイル名]** ボックスで、新規プロジェクトの名前を入力します。
4. **[ファイルの種類]** リストで、**[InstallScript オブジェクト プロジェクト (*.ism)]** を選択します。
5. **[保存]** をクリックします。

指定した名前を持つオブジェクト プロジェクト ファイル (..ism ファイル) とすべてのプロジェクトのサブフォルダーが指定の場所に作成されます。プロジェクトが InstallShield で開きます。

詳細については、「InstallScript デバッガー」を参照してください。

メモ: *後でオブジェクトに変換することを考慮に入れてオブジェクトを標準インストールとしてデザインしている場合、次の制限があることに注意します。*

- *オブジェクトはセットアップの種類を含みません。さらに、定義するセットアップの種類やプロジェクトで呼び出すセットアップの種類関数は、オブジェクトがオブジェクトプロジェクトに変換される際に無視されます。*
- *オブジェクトでサポートされない関数やイベントが数多くあります。*
- *プロジェクトが変換される際に、ビルドされたメディアに関する情報はすべて失われるため、プロジェクトを再ビルドする必要があります。*

別のマシンでのオブジェクトのデバッグ

[**インライン デバッグ情報を生成する**] は、[ビルドの設定] ダイアログ ボックスの [コンパイル/リンク] タブにあります。オブジェクトが作成されたマシン以外のマシン上でオブジェクトをデバッグすることができます。このオプションを使用しないと、デバッグ情報は Setup.dbg という名前のファイルに配置され、その場所は Setup.inx のヘッダーに格納されます。このスキーマは、コンパイルを行ったのと同じマシン上でスクリプトをデバッグする場合、問題はありません。しかし、別のマシンを使用する場合、.dbg ファイルは同じ場所に保管されない可能性があります。この問題を緩和するために、[**インライン デバッグ情報の生成**] オプションはデザインされました。このオプションを使用する場合、デバッグ情報はすべて Setup.inx に保管されます。このため、インストールがデバッグ情報を検索する必要はありません。

下記の 2 つの理由により、最後の製品と共に出荷する Setup.inx をコンパイルする前に **[インライン デバッグ情報の生成]** オプションをクリアすることが推奨されます。1 つ目の理由は、デバッグ情報は Setup.inx のサイズを増やし、インストール実行にかかる時間をさらに遅くします。2 つめは、インラインデバッグ情報によって、他人がコードを容易にリバースエンジニアリングしてしまう可能性があります。

InstallShield キャビネット & ログ ファイル ビューアーを使用する

InstallShield キャビネット & ログ ファイル ビューアーを使って、以下を行います:

- InstallScript キャビネット ファイル (.cab) または InstallScript ヘッダー ファイル (.hdr)、およびそれらの圧縮ファイル、レジストリ エントリ、コンポーネント、機能、その他のデータ を参照する。このツールを使って、.cab ファイルからファイルを抽出することもできます。
- InstallScript インストールが作成した InstallScript ログ ファイル (.ilg) を参照する。このツールを使って、インストールがログ ファイルに記録した内容を参照できます。ログ ファイルには、重要なアンインストール情報がバイナリ形式で保存されます。

詳細については、「InstallShield キャビネット & ログ ファイル ビューアー」を参照してください。

カスタム デザインタイム ウィザードのデバッグ

オブジェクトに対してカスタム デザインタイム ウィザードを作成した場合、そのウィザードをデバッグする必要があります。

タスク ***ウィザードをデバッグするには、次の操作を行います。***

1. 開発環境からウィザードのデバッグバージョンを起動し、ブレークポイントを希望の場所に設定します。
2. InstallShield インターフェイスで、別のプロジェクトの機能にオブジェクトを挿入してオブジェクトのウィザードを起動します。
3. 開発環境が Microsoft Visual Basic の場合、次の操作を行う必要があります。
   a. インターフェイスを最小化します。**[サーバー使用中]** メッセージ ボックスが開きます。(メッセージ ボックスが開く前に、タイトルバーの最小化ボタンを何度かクリックしなければならないこともあります。)
   b. **[サーバー使用中]** メッセージ ボックスで、**[切り替え]** ボタンをクリックするか Enter キーを押します。

オブジェクトの配布

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

オブジェクトの主な利点は、オブジェクトがマシン上にインストールされて登録されている限り、InstallShield インストールを作成している人なら誰でも最低限の作業でインストールにオブジェクトを含めることができるという点です。オブジェクトを登録する最も簡単な方法は、InstallShield オブジェクト インストーラーを使用することです。このオブジェクトを使うと、InstallShield X 以降、InstallShield DevStudio、または、InstallShield Professional 6.1 以降があれば配布するオブジェクトをパッケージにしてターゲットマシンに登録することができます。

タスク ***インストールに InstallShield オブジェクト インストーラーを追加するには、以下の手順を実行します。***

1. **[オブジェクト]** ビューを開きます。**[InstallShield オブジェクト/マージ モジュール]** ペインには、使用可能なすべてのオブジェクトが一覧表示されます。
2. **[機能]** ペインから InstallShield オブジェクト インストーラーを追加する機能を選択します。
3. **[InstallShield オブジェクト インストーラー]** を右クリックして、**[選択された機能に追加]** を選択します。

関連したウィザードが表示されるので、これに従ってカスタマイズします。

メモ- 配布するオブジェクトが InstallShield オブジェクトギャラリーに存在する必要があります。これが存在しない場合、そのオブジェクトを登録する必要があります。詳しくは、「InstallScript プロジェクトでオブジェクトを登録する」をご覧ください。

オブジェクトのスクリプト

***プロジェクト-** この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShieldでは、追加の関数を定義することでオブジェクトのサポートをInstallScript ビューにスクリプトを追加することでオブジェクト プロジェクトでサポートされていない定数のサポートを拡張することができます。基本的に、InstallShield インターフェイスでサポートされている関数を強化するスクリプトを書くことができです。

オブジェクト プロジェクトでサポートされていない関数

***プロジェクト-** この情報は、InstallScript オブジェクトプロジェクトに適用します。*

以下に、オブジェクト プロジェクトでサポートされていない InstallScript 関数の一覧を示します。

- OnFileReadOnly
- OnRemovingSharedFile
- OnFileLocked
- OnNextDisk
- OnMD5Error
- OnFileError
- OnComponentError
- SdSetupType
- SdSetupTypeEx
- SetupType
- ComponentSetupTypeSet
- ComponentSetupTypeEnum
- ComponentSetupTypeGetData

オブジェクト プロジェクトでサポートされていない定数

***プロジェクト-** この情報は、InstallScript オブジェクトプロジェクトに適用します。*

以下に、オブジェクト プロジェクトでサポートされていない InstallScript 定数の一覧を示します。

- IFX_ONNEXTDISK
- IFX_ONNEXTDISK_MSG
- IFX_MAINTUI_MSG
- IFX_ONMAINTUI_CAPTION
- IFX_SDFINISH_MSG1
- IFX_SDFINISH_MAINT_MSG1
- IFX_SDFINISH_MAINT_TITLE

オブジェクト プロジェクトでのみサポートされている関数

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

以下に、オブジェクト プロジェクトでのみサポートされている InstallScript 関数の一覧を示します。

- SetStatus
- WizardDirection

ScriptDefinedVar プロパティをオブジェクトに追加する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

InstallShield のオブジェクトには、ScriptDefinedVar プロパティを含むものがあります。このプロパティで、デザイン時にパス (例、**<MYSERVICEFOLDER>¥MyService.exe**) を定義するのに使用するスクリプト定義の変数値を実行時に設定することができます。たとえば、以下のようなメイン スクリプトのコードを使うことにより、エンドユーザーは*<MYSERVICEFOLDER>* の値を指定することができます:

```
/* 名前を付けたオブジェクトへの参照を取得します。
オブジェクト名は [コンポーネント] ペイン の「新しい カスタムオブジェクト 1」です。*/
set oObj = GetObject("New Custom Object 1");

/* エンドユーザーからフォルダーの選択を取得します。*/
svDir = TARGETDIR;
AskDestPath ( "" , "サービスファイルの場所を指定してください:" ,
    svDir , 0 );

/* 希望する <MYSERVICEFOLDER> の値をオブジェクトに伝えます。*/
oObj.ScriptDefinedVar("<MYSERVICEFOLDER>") = svDir;
```

タスク　　オブジェクト スクリプトに ScriptDefinedVar プロパティを含めるには、下記の手順を実行します。

1. **InstallScript** ビューを開きます。
2. **InstallScript** エクスプローラーで、**[プロパティ]** を右クリックして、**[新しいプロパティの追加]** をクリックします。**[新しいプロパティの追加]** ダイアログ ボックスが開きます。
3. 以下をエントリを行ってください。
   c. **[プロパティ名]** ボックスで、次のように入力します。

      ScriptDefinedVar

   d. **[データ型]** リストで、**[文字列]** を選択します (プロパティへ文字列データを書き込むため)。

   e. **[アクセス メソッド]** リストで、**[読み取り/書き込み]** を選択します。(オブジェクトのユーザーがメイン スクリプトから現在のプロパティ値を確認しないことが明確な場合は、代わりに **書き込み専用** を選択できます。)
4. [OK] をクリックしてダイアログ ボックスを閉じ、必要なコードをすべてオブジェクト スクリプトに自動追加します。
5. 自動追加されたプロパティのスクリプト コードに対して、次の変更を行います。
   a. **InstallScript** ビューを開きます。

   b. **InstallScript** エクスプローラーの **[プロパティ]** の下にある **ScriptDefinedVar** をダブル クリックして、スクリプト エディター ウィンドウのキャレットを ScriptDefinedVar 宣言に移動します。宣言を以下のように変更して、宣言に文字列パラメーターを追加します:

      ```
      property(get,put) STRING ScriptDefinedVar();
      ```

      変更後、

      ```
      property(get,put) STRING ScriptDefinedVar( STRING ) へ変更。
      ```

   c. **InstallScript** エクスプローラーの **[関数]** の下にある get_ScriptDefinedVar をダブル クリックして、スクリプト エディター ウィンドウのキャレットを **get_ScriptDefinedVar** 関数ブロックに移動します。次の関数ブロックを検索します。

      ```
      function STRING get_ScriptDefinedVar()
      begin
         return m_strScriptDefinedVar;
      end;
      ```

      次のように変更します。

      ```
      function STRING get_ScriptDefinedVar( szScriptVar ) /* 文字列引数を追加します。*/
      begin
         return TextSub.Value( szScriptVar ); /*  szScriptVar に関連した値を取得します。*/
      end;
      ```

   d. **InstallScript** エクスプローラーの **[関数]** の下にある **put_ScriptDefinedVar** をダブル クリックして、スクリプト エディター ペインのキャレットを **put_ScriptDefinedVar** 関数ブロックに移動します。次の関数ブロックを検索します。

      ```
      function void put_ScriptDefinedVar(newVal)
      begin
         m_strScriptDefinedVar = newVal;
      ```

```
end;
```

次のように変更します。

```
function void put_ScriptDefinedVar( szScriptVar, newVal ) /* 文字列引数を追加します。*/
begin
    TextSub.Value( szScriptVar ) = newVal; /* newVal を szScriptVar に関連付けます。*/
end;
```

オブジェクトでシステム変数を使用する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

システム変数は、ソースパス、ターゲットパス、Windows フォルダー、および Windows システムフォルダーのような情報を含む、あらかじめ定義された変数です。スクリプト中でこれらの変数を定義することはできません。InstallShield では、インストール プロセスの開始時に自動的にシステム変数が初期化されます。

オブジェクト内のシステム変数を変更したり初期化したりすることは可能ですが、お勧めしません。オブジェクト内のシステム変数を変更すると、その影響はインストール全体に及びます。たとえば、オブジェクトによって TARGETDIR に特定の値が初期値として設定された場合、その値は、インストール プロジェクト内ですでに設定された値を上書きしてしまいます。従って、インストール全体にわたって、ファイルはインストール開発者によって指定されたディレクトリではなくオブジェクトで指定されたディレクトリへインストールされることになります。

スクリプトを使用してオブジェクトのインストール先を設定する

プロジェクト: *この情報は、InstallScript オブジェクトプロジェクトに適用します。*

スクリプト定義の変数を使用して、オブジェクトコンポーネントのインストール先のプロパティの一つを定義できます。そのためには、オブジェクトのユーザーは、スクリプト中で **ComponentCompareSizeRequired** が呼び出される前に、スクリプト定義の変数値を設定する必要があります。そうでない場合は、関数は不充分なディスク容量を報告します。次のうちのいずれかまたは両方の方法で、スクリプト定義の変数値を設定することができます。

- 書き込み専用、または読み取り/書き込みオブジェクトのプロパティの put_ function が **ComponentSetTarget** を呼び出すようにする。以下に例を示します。

  ```
  function VOID put_SetTargetDest ( szScriptVar , szValue )
  begin
      ComponentSetTarget ( MEDIA , szScriptVar , szValue );
  end;
  ```

- オブジェクトの OnFirstUIBefore イベント ハンドラーによって、エンドユーザーのダイアログを表示させる。

7

開発プロセスを再利用/分担するためのインストール プロジェクトのモジュール化

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

InstallShield のデベロッパー インストール マニフェスト (DIM) は、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。DIM は、機能サイズのプロジェクトで、インストール パッケージの論理的に別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。以下は、DIM を利用した時のいくつかの利点です:

- DIM を利用することにより、複数のチーム メンバーが、インストールの開発を同時に携わることができます。各ソフトウェア開発者またはチームメンバーは、異なる DIM について個別で作業することができ、リリース エンジニアは、1 つまたは複数のインストール プロジェクトでそれらを参照することができます。
- リリース エンジニアは、DIM を複数のインストール プロジェクトで繰り返し使用できるため、非常に効率的です。
- DIM では、基本の MSI プロジェクトで提供されている機能とほぼ同じ機能がサポートされています。このため、DIM の作成者には、インストールの部分開発を行うために必要な柔軟性がすべて提供されています。

この章では、InstallShield における DIM プロジェクトの使い方について説明し、次のトピックが含まれています:

- DIM のインストール情報を指定する
- DIM でパス変数を使用する
- DIM のファイルを編成する
- DIM のターゲット システムの構成
- DIM のインストール動作をカスタマイズする
- DIM のサーバーを構成する

DIM のインストール情報を指定する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

InstallShield は、プロジェクトの設定を、単一インストール プロジェクト ファイル (.dim) に保管します。このファイルには、DIM プロジェクトについてのすべての情報が含まれています。[一般情報] ビューでは、作成者名、プロジェクトについての内部メモ、リリース エンジニアが DIM をインストール プロジェクトに含める時に留意すべき手順の説明など、DIM についての基本的な情報を編集できます。

このセクションには、次のトピックが含まれています:

- プロジェクト ファイルを XML またはバイナリ形式で保存する
- DIM のデフォルトのインストール先フォルダーを指定する

プロジェクト ファイルを XML またはバイナリ形式で保存する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

InstallShield では、.ism プロジェクトファイルを XML またはバイナリ形式で保存することができます。

タスク

プロジェクト ファイルの形式を指定するには、以下の手順に従います:

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"プロジェクト ファイルの形式"** 設定で、適切なオプションを選択します: 選択可能なオプションは以下のとおりです:
   - **バイナリ** – プロジェクト ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。
   - **XML** – プロジェクト ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。

メモ: *DIM プロジェクト ファイル (.dim) を XML またはバイナリ形式で保存すると、プロジェクト ファイルは .ism 拡張子を保持します。*

DIM のデフォルトのインストール先フォルダーを指定する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

[一般情報] ビューの "INSTALLDIR" 設定に入力された値は、DIM のすべてのファイルのデフォルト フォルダーとなります。したがってその値は、デフォルトのインストール先フォルダーである、Windows Installer フォルダー プロパティ INSTALLDIR に割り当てられます。インストール プロジェクトでは、通常これは次のように評価されます:

[ProgramFilesFolder]会社名¥*製品名*

この DIM のユーザーが、デフォルトのインストール先を上書きできるようにするには、"INSTALLDIR" 設定に **[TARGETDIR]** を入力します。詳細については、「DIM リファレンスのインストール先のオーバーライド」を参照してください。

DIM でパス変数を使用する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

DIM プロジェクトに製品ファイルを追加すると、デフォルトで、製品ファイルのソースとして使用されるパス変数が作成されます。パス変数は、プロジェクトのソース ファイルを含むディレクトリを表すために使われる変数です。パス変数はデザイン時とビルド時の両方で使用されますが、インストールがエンドユーザーのマシンで実行されるインストールの実行時には使用できません。

ハードコード化されたパスの代わりにパス変数を使う利点として、たとえばソール ファイルが開発システム上にある別のディレクトリに移された場合、パス変数の値を変更するだけで済みます。つまり、ハードコード化されたすべてのソース ファイルのパスをそれぞれ更新する必要はありません。

DIM プロジェクトでパス変数を使用すると、ソース ファイルの管理を簡略化できます。特に、DIM プロジェクトが DIM プロジェクトを含める基本の MSI プロジェクトとは異なる場所にある場合、これは有用です。

また、すべてのコードと InstallShield プロジェクト ファイルを格納するためにソース コード管理を使っている場合や、ソース コード管理から適切なファイルを取得し、コードをコンパイルし、インストールをビルドするビルド マシンを使用している場合も、パス変数は有用です。このようなシナリオでは、異なるリリース エンジニアや開発者が、ソース管理から最新のファイルを取得するために別のローカル作業ディレクトリを使うことがあります。パス変数を使って場所を定義すると、ソースの場所が異なるために発生する問題を最小限に抑えることができます。

DIM プロジェクトにおける定義済みパス変数

以下は、DIM プロジェクトにおける定義済みパス変数、および通常それらに設定される値です。

テーブル 1 ・ DIM プロジェクトにおける定義済みパス変数

定義済みパス変数	通常値
CommonFilesFolder	C:¥Program Files¥Common Files¥
ISProductFolder	C:¥Program Files¥InstallShield¥2022¥
ISProjectDataFolder.DIM_GUID	<ISProjectFolder>¥ProjectName¥
ISProjectFolder.DIM_GUID	C:¥InstallShield 2022 Projects¥

テーブル 1・DIM プロジェクトにおける定義済みパス変数 (続き.)

定義済みパス変数	通常値
ISRedistPlatformDependentExpressFolder	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386 Express
ISRedistPlatformDependentFolder	C:¥Program Files¥InstallShield¥2022¥Redist¥Language Independent¥i386
ProgramFilesFolder	C:¥Program Files¥
SystemFolder	C:¥Windows¥System32¥
WindowsFolder	C:¥Windows¥

パス変数に *DIM_GUID* が含まれる両方のケースでは、*DIM_GUID* は、DIM プロジェクトの [一般情報] ビューの "DIM GUID" 設定で定義される識別子になります。次は、DIM プロジェクトにおけるサンプル パス変数です:

- **ISProjectDataFolder.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC**
- **ISProjectFolder.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC**
- **MyNewPathVariable.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC**

GUID は、DIM プロジェクトからの適用可能なパス変数を判別するのに役立ちます。また、同じパス変数が、ある DIM プロジェクトとその DIM を含む基本の MSI プロジェクトで使用されていて、異なる値がこれらの 2 つのパス変数に割り当てられている場合に発生する競合を最小化または削除する必要がある時にも役立ちます。

InstallShield*SystemFolder* および *WindowsFolder* など、GUID を含まないパス変数名は、で開いているプロジェクト専用ではないため、すべてのマシンで使用可能です。これらの種類のパス変数の値は、オペレーティング システムに応じて異なる場合があります。たとえば、32 ビット マシンでは、**ProgramFilesFolder** の値は C:¥Program Files¥ の可能性があります。一方、64 ビット マシンでは、C:¥Program Files (x86)¥ の可能性があります。

パス変数を参照するための構文

パス変数が InstallShield で使用されている場合、パス変数は、 山かっこで囲まれます。例:

<ISProjectDataFolder.8356F8B7_8DE5_4E04_A77A_6FA722CBE1CC>

場合によっては、別のパス変数の相対的になるようにパス変数を定義することできます。たとえば、**MySourceFiles** という名前の標準パス変数を作成したと仮定します。そしてこの時、**BinFiles** という名前の別のパス変数を使って、先の **MySourceFiles** ディレクトリ内にある **Bin** フォルダーへのパスも定義する必要もあると仮定します。この場合、**BinFiles** 変数の Defined Value 列に相対パス変数を入力する時、 でベースのパス変数を囲む必要があります。したがって、**BinFiles** 変数の Defined Value 列は、次のようになります:

<MySourceFiles>¥Bin

MySourceFiles の値が C:¥CheckoutRoot¥ だとすると、BinFiles の値は、C:¥CheckoutRoot¥Bin¥ になります。

サンプル パス変数のシナリオ

1 台またはそれ以上のマシンを使って InstallShield プロジェクトの作成/編集を行っているが、インストールのビルドには別のマシンを使ってるというような環境では、プロジェクトでパス変数をどうしようするか前もって考えておく必要があります。そうしなかった場合、InstallShield でインストールをビルドした時、ファイルが不足しているためにビルド エラーが発生したり、間違ったファイルがビルドに取り込まれたりします。

次の例では、InstallShield プロジェクトでパス変数を使う時の、いくつかの異なる方法が説明されています。最も容易な方法は、例 3 で説明されている方法です。

例 1: DIM を含む基本の MSI プロジェクトで DIM のパス変数の値を上書きする

次のテーブルでは、DIM プロジェクトで、1 つの定義済みパス変数と、いくつかの標準パス変数がどう使用されているかが説明されています。

テーブル 2 ・ 例 1 DIM プロジェクトにおけるパス変数

名前	定義された値	現在の値
ISProjectFolder.4E846FD3_5307_4CF5_9FF1_C476E5505666	(これは、InstallShield プロジェクト ファイルを含むフォルダーの定義済みパス変数です。この値は、手動で定義することはできません。)	C:¥Source¥Units¥Utility¥
CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666	C:¥Source	C:¥Source
PATH_TO_RELEASE_FILES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥SpellLib¥Bin¥Release	C:¥Source¥SpellLib¥Bin¥Release
PATH_TO_DICTIONARIES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥Dictionaries	C:¥Source¥Dictionaries

この例では、DIM プロジェクトの作成者は、パス変数 **CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666** を、自分のソース ルート フォルダー **C:¥Source** (ソース管理からファイルをチェックアウトしているマシン上の作業ディレクトリ) に割り当てています。ソール ファイルを含む他のフォルダーのパス変数は、このソース ルートパス変数に相対的に定義されています。

リリース エンジニアが、ビルド マシン上の異なる作業ディレクトリを使っている場合、場合によって、DIM を含むプロジェクトで基本の MSI インストールをビルドする時、このパス変数の値を上書きする必要があります。リリース エンジニアは、コマンドラインからインストールをビルドする時、-I パラメーターを ISCmdBld.exe に渡すことで、ソース ルート パス変数の新しい値を設定することができます。他のパス変数の値は、ソース ルート パス変数を使って定義されるため、これらの他のパス変数の値は、それに沿って更新されます。

この方法は、上書きを必要とするパス変数をそれぞれ含む複数の DIM プロジェクトがある基本の MSI プロジェクトのインストールをビルドする場合など、場合によって、手間がかりエラーが発生しやすいという欠点があるで注意が必要です。

例 2: ビルド環境にしたがって、DIM パス変数の値を変更する

DIM 開発マシンのソール ルート パスが、基本の MSI をビルドしているマシン上の対応するパスと一致しない場合、DIM を直接開いて、パス変数の値を必要に応じて更新することができます。DIM プロジェクトを基本の MSI プロジェクト内から開く方法については、「インストール プロジェクト内から参照された DIM プロジェクトを開く」を参照してください。

この方法は、テスト目的で毎日ビルド処理を実行している環境で、DIM プロジェクトにまだ変更を行っている場合、理想的ではありません。

例 3: ISProjectFolder.DIM_GUID のような定義済みパス変数に相対するパス変数を使用する

DIM プロジェクト内のパス変数を管理する最も簡単な方法は、**ISProjectFolder** などの定義済みパス変数に相対するパスにソール ファイルを常に保存しておくことです。**DIM_GUID**実際の手順を以下で説明します。

テーブル 3 · 例 3 DIM プロジェクトにおけるパス変数

名前	定義された値	現在の値
ISProjectFolder.4E846FD3_5307_4CF5_9FF1_C476E5505666	(これは、InstallShield プロジェクト ファイルを含むフォルダーの定義済みパス変数です。この値は、手動で定義することはできません。)	次は、考えられる開発マシンにおける現時点での値です: C:¥Source¥Units¥Utility¥ 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source¥Units¥Utility¥
CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666	<ISProjectFolder.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥..¥.. ***メモ:*** *このパスの値に定義済みのプロジェクト フォルダー パス変数を使用して、それから他のパス変数の値にチェックアウト ルート パス変数を使用すると、InstallShield プロジェクトおよびその他すべてのソース ファイルが、マシンからマシンに移動できるポータブルになります。*	次は、考えられる開発マシンにおける現時点での値です: C:¥Source 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source
PATH_TO_RELEASE_FILES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥SpellLib¥Bin¥Release	次は、考えられる開発マシンにおける現時点での値です: C:¥Source¥SpellLib¥Bin¥Release 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source¥SpellLib¥Bin¥Release

テーブル 3 - 例 3 DIM プロジェクトにおけるパス変数 (続き.)

名前	定義された値	現在の値
PATH_TO_DICTIONARIES.4E846FD3_5307_4CF5_9FF1_C476E5505666	<CHECKOUT_ROOT.4E846FD3_5307_4CF5_9FF1_C476E5505666>¥Dictionaries	次は、考えられる開発マシンにおける現時点での値です: C:¥Source¥Dictionaries 次は、考えられるビルド マシンにおける現時点での値です: C:¥BuildWorkspace¥Source¥Dictionaries

チェックアウト ルート パス変数の値に定義済みのプロジェクト フォルダー パス変数を使用して、それから他のパス変数の値にチェックアウト ルート パス変数を使用すると、InstallShield プロジェクトおよびその他すべてのソース ファイルが、マシンからマシンに移動できるポータブルになります。

DIM のファイルを編成する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

DIM プロジェクトでは、ファイルは階層構造で編成されます。インストールの自分の担当の部分のファイルをコンポーネントに追加します。DIM を基本の MSI プロジェクトに含めると、DIM のコンポーネントは、基本の MSI プロジェクトの 1 つまたは複数の機能に追加されます。エンドユーザーがインストールする機能および除外する機能を選択できるようにした場合、基本の MSI インストールが実行された時、。エンドユーザーはインストールする機能を選択できるようになります。

DIM プロジェクトでコンポーネントにファイルを追加する場合、セットアップ ベスト プラクティスに留意する必要があります。デフォルトでは、DIM ベストプラクティス ウィザードがセットアップのデザインを監視し、ベストプラクティスに反した場合には警告を表示します。

ファイルの編成についての詳細は、ドキュメントの次のセクションを参照してください:

- ファイルとフォルダーを含める
- コンポーネントを使用する
- セットアップ ベスト プラクティス

DIM のターゲット システムの構成

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

インストールでは、ターゲット システムの変更が必ず発生します。簡単なインストールでは、ファイルをコピーするだけのものもあります。より複雑なインストールでは、レジストリの変更、.ini ファイルの編集、ショートカットの作成、ODBC リソースの構成、環境変数の使用、XML ファイルの変更、およびテキスト ファイルの変更が行われます。DIM プロジェクトでこれらの種類の構成に関する詳細は、ドキュメントの次のセクションを参照してください:

- ショートカットおよびプログラム フォルダーの作成
- レジストリの編集
- .ini ファイル データの変更
- ODBC リソースの構成
- 環境変数を使用する
- XML ファイルの変更
- テキスト ファイルの変更

DIM のインストール動作をカスタマイズする

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

インストール作成の重要な要素は、それをエンドユーザーのニーズに合わせてカスタマイズすることです。たとえば、Windows Installer が直接サポートしていない機能を追加するカスタム アクションを作成するのに役立ちます。また、検索の結果によって、インストールが続行できるかどうかが判別される場合など、ターゲット システムでインストール済みデータを検索する必要が出てくる場合もあります。Windows Installer のプロパティを使用すると、プロジェクトに対してプロジェクト全体の値を使用できます。DIM プロジェクトのインストール動作のカスタマイズに関する詳細は、ドキュメントの次のセクションを参照してください:

- カスタム アクションを使用
- インストールされたデータの検索
- Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方

DIM のサーバーを構成する

プロジェクト: *この情報は、DIM プロジェクトに適用します。*

DIM プロジェクトを作成しているとき、ターゲット システムにインストールされるテクノロジに対してサーバー側のサポートを提供する必要があることに気がつくことがあります。InstallShield では、インターネット インフォメーション サービス (IIS) Web サイトの作成と管理、COM+ アプリケーションおよびコンポーネントの管理、およびサーバー接続および設定による SQL スクリプトの管理および編成がサポートされています。DIM プロジェクトのサーバーの構成に関する詳細は、ドキュメントの次のセクションを参照してください:

- SQL サポートの構成
- COM+ アプリケーションとコンポーネントの管理
- インターネット インフォメーション サービス

8

アプリケーションのアップデート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*
- *QuickPatch*

アプリケーションのアップグレードをインストールすることは、アプリケーションのオリジナルリリースのインストールよりもはるかに一般的な操作です。この意味で、効率的で、信頼のおけるアップグレードを作成することは、とても重要なタスクです。

[アップデートの更新] は、異なるタイプのアップグレードについての確かなバックグラウンド情報が提供されており、一般的なパッチについての誤解が分かりやすく説明されています。"アプリケーションの更新" では、また、製品に最適なアップグレード ソリューションを判断するのに役に立ち、アップグレード、パッチ、QuickPatch プロジェクト、差分リリース、および、完全リリースの作り方を手順を追って案内します。またこの章には、FlexNet Connect を利用してエンドユーザーに製品の新しいバージョンのリリースついて通知する方法について説明する次のトピックが含まれています:

- アップグレードの概要
- 最適なアップグレード ソリューションの決め方
- アップグレード、パッチ、および QuickPatch プロジェクトを使用する

アップグレードの概要

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*
- *QuickPatch*

ソフトウェア アプリケーションの保守は、開発にかかったコストよりも高くかかる場合があります。そのため、効率的で信頼のおけるアップグレードを作成することは重要なタスクです。アプリケーションに対する堅牢なアップグレードの配布が可能かどうかは、オリジナル インストールパッケージがどのように構成され、配布されたかによって左右されます。ヘルプトピックには、堅牢なアップグレードパッケージ作成の基盤となる情報が掲載されています。

基本の MSI プロジェクトと InstallScript MSI プロジェクトのアップグレード方法

Windows Installer は 3 種類 (スモール アップデート、マイナー アップグレード、およびメジャー アップグレード) の製品アップグレードをサポートしています。アップグレードは、完全インストールまたはパッチとしてパッケージすることができます。パッチはアップグレードを実装するための 1 つの仕組みに過ぎません。ただし完全リリースとは違って、パッチはインストール済みのファイルを最新版に変更するために必要な部分のみをユーザーに配布します。

MSI データベース、およびトランスフォーム プロジェクトのアップグレード方法

MSI データベースおよびトランスフォーム プロジェクトは、Windows Installer のメジャー アップグレードをサポートします。

InstallScript プロジェクトのアップグレード方法

InstallScript インストール用には、完全リリース パッケージ、または差分リリースのどちらかを使ってアプリケーションを更新することができます。最適なアップグレード方法を決定するには、ファイルサイズ、以前のリリースからのファイルの有無など様々な要素を考慮する必要があります。

メモ: *InstallShield Professional 6.0 よりも古いバージョンを利用して InstallScript プロジェクト用のメディアを作成した場合、メディアをアップデートすることはできません。*

このセクションには、次のトピックが含まれています:

- メジャーアップグレード
- マイナーアップグレード
- スモール アップデート
- パッチの適用
- QuickPatch プロジェクト
- 差分リリース
- 完全リリース

メジャーアップグレード

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

メジャーアップグレードでは製品の変更規模が大きいため、パッケージコードだけでなく製品バージョン番号および製品コードの両方を変更する価値があります。たとえばバージョン 1.2 製品の 2.0 へのアップデートです。メジャーアップグレードは、以前のバージョンが存在しない場合は初回インストールと同様に作動します。以前のバージョンが存在する場合、メジャーアップグレードは通常それをアンインストールしてから新しいバージョンをインストールします。また、メジャーアップグレードにまず以前のバージョンを上書きインストールさせてから、使われないファイルを削除することも可能です。方法としては、この方がより効率的ですが、コンポーネント作成規則の厳密な順守が必要です。

メジャーアップグレードの動作の仕方

Windows Installer のヘルプで示唆されているように、メジャー アップグレードを行うにはインストールの最新バージョンの製品コードを変更する必要があります。

メモ: インストールの製品コードは、[一般情報] ビューで設定できます。新しい製品コードは、一意である限り、特に制約はありません。

一意の製品コードを使用することで、エンド ユーザーのマシン上で、最新のインストールを Windows Installer サービスに独自に登録することができます。言い換えるとこれは、何かしらの手段が執られないかぎり、Windows Installer は、以前のバージョンとは無関係に製品の最新のバージョンをインストールするということを意味します。これは、2 つの異なるバージョンの製品が同じマシン上に共存できる場合は問題はありませんが、アップグレードされたことにはなりません。したがって、便宜上、ここでは 2 つのバージョンが同じマシン上で共存できないと仮定します。

最新インストールの製品コードを更新した場合、[アップグレード] ビューを使って、アップグレードを行う以前のすべてのバージョンについての情報を指定します。

メージャー アップグレードの作成に関する詳細については、"メジャー アップグレードを作成する" ヘルプを参照してください。

実行時のメージャーアップグレード

製品の以前のバージョンがターゲット マシン上に存在しない場合にエンド ユーザーがメジャー アップグレードを実行すると、初回インストールとしてインストールが行われます。

製品の以前のバージョンがターゲット マシン上に存在している状態でエンド ユーザーがメージャー アップグレードを実行すると、エンドユーザーはあたかも以前のインストールが存在しないマシン上に最新のアプリケーションをインストールするかのような印象を受けます。唯一の異なる点は、新しいリソースがインストールされる前に、インストールはまずターゲット マシンから以前のバージョンをアンインストールするということです。

この削除過程は、[セットアップ進行状況] ダイアログの進行状況バーで表示され、エンドユーザーはアンインストールの処理状況を直接見ることができます。以前のインストールの削除が完了すると、最新のインストールからのリソースがターゲット マシンにインストールされます。

このタイプのアップグレードは、つまり、完全アンインストールとそれに続くアプリケーションに関連付けられたリソースすべての再インストールからなります。したがって、エンドユーザーによって構成されたアプリケーションのためのすべてのデータはエンドユーザーのマシーンから完全に削除される可能性があります。エンドユーザー データの一部を残しておく必要がある場合、このデータのバックアップをとってから、新しいデータのインストールが完了した後、それを置き換えるカスタム アクションを作成してください。

マイナーアップグレード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

製品データベースおよびファイルへの変更規模が ProductCode プロパティを変更するほどではないけれども、ProductVersion プロパティを変更をするだけの価値があるとき、これをマイナー アップグレードといいます。つまり、マイナーアップグレードの場合、パッケージコードおよび製品バージョン番号両方とも以前のインストールパッケージと異なりますが、製品コードは変わりません。バージョン 1.1 の製品からバージョン 1.2 へアップデートはひとつの例です。マイナーアップグレードでは、通常、異なるバージョンの間でインストールの構成において重要な変更は行われません。フル インストールとしてパッケージされたマイナー アップグレード は、以前のバージョンが存在しない場合は初回インストールと同様に動作しますが、製品が既にインストールされている場合、その上からインストールを行います。既存のインストールをアップグレードする場合、マイナーアップグレードは基本的にバージョン 1.2 と 1.1 のアプリケーションの差分のみをインストールします。

メモ: *InstallScript MSI マイナー アップグレード中、製品のベース バージョンがデフォルト以外の場所にインストールされていて、新しいコンポーネントが INSTALLDIR のサブフォルダ—に追加された場合、新しいコンポーネント ファイルは、ユーザーが指定したサブフォルダ—の場所ではなく、デフォルトの場所にインストールされます。*

回避策については、「InstallScript MSI Installs to Default Location During Minor Upgrades (マイナー アップグレード中に IInstallScript MSI を既定の場所にインストールする)」を参照してください。

ヒント: *アップグレードの製品バージョンは、[一般情報] ビューで変更できます。*

スモール アップデート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

本質的にスモール アップデートは、インストール済みのアプリケーションのいくつかのファイルを変更するのに使用されるアップデート タイプで、一般的に小さいバグ修正の配布に使用されます。スモール アップデートには、ホットフィックスなどの製品バージョンの変更が必要なほど大きくはない変更が含まれます。スモールアップデートでは、パッケージコードの変更が必要です。

3.0 以前のバージョンの Windows Installer と共にインストールされるスモールアップデートの短所は、製品のより新しいバージョンのためのインストーラーをはじめとする、外部プログラムは、オリジナルバージョンとアップデートされたバージョンを区別することができないということです。また、3.0 以前のバージョンの Windows Installer は、スモールアップデートを正しい順序で適用することができないこともあります。

パッチの適用

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチは、以前のバージョンの Windows Installer インストールパッケージをアップデートしてアプリケーションをアップデートするための効率的な仕組みです。パッチはインストール済みのファイルを新しいファイルへ変更するのに必要な部分のみをカスタマーに配布します。パッチが優れている理由のひとつは、アップグレードを配布するために必要なアップグレードパッケージのサイズを、完全インストール パッケージのそれと比べて格段に小さく収めることが可能な点です。アップグレードパッケージをできるだけ小さくすることで、インターネットを通してアップグレードを簡単に配布することが可能になります。

但し、パッチが必ずしも最善のソリューションとは限らないので注意してください。たとえば、インストールを圧縮形式から非圧縮形式に変更する場合、またはその逆を行なう場合、アップグレードはパッチではなく完全インストールとしてパッケージしなくてはなりません。アップグレードの最適なパッケージ オプションの選び方については、「アップグレードのパッケージ オプション」を参照してください。

パッチはパッチパッケージ (.msp) ファイルの形式で配布され、これをエンドユーザーがインストール済みの製品に適用します。パッチパッケージは、インストールの旧バージョンをその数に関わらずアップデートすることができます。パッチパッケージには、指定した以前のバージョンをアップデートするための個別のトランスフォームおよび指示が含まれています。

パッチ作成の重要な側面は、パッチパッケージが作成されるパラメーターを定義するパッチ作成プロパティ (.pcp) ファイルの生成です。.pcp ファイルは特定のスキーマを持つデータベースですが、InstallShield または Orca を使って直接開くことができます。

[パッチのデザイン] ビューは、パッチ作成処理を簡素化する、使いやすいインターフェイスを提供します。このビューは、パッチ作成で必要な処理すべてを論理的にグループにまとめます。

QuickPatch プロジェクト

プロジェクト: *以前のバージョンが Windows Installer を使用してインストールされている場合のみ、QuickPatch を作成して製品をアップデートすることができます。初期のバージョンの製品のインストールは、次のプロジェクト タイプのいずれかに作成されている必要があります:*

- *基本の MSI*

- *InstallScript MSI*
- *QuickPatch*

QuickPatch プロジェクトは、規模の小さいシングル アップグレードをユーザーへ配布したいインストール作成者へお勧めするプロジェクトの種類です。カスタム アクションの追加、.ini データの変更などのより広範囲におよび変更には通常、標準パッチが必要です。

QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ構成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

QuickPatch では、次のすべてを実行することができます。

- 元のインストールまたは以前の QuickPatch へ新しいファイルを追加する。
- 元のインストールのファイルを削除する。
- 以前の QuickPatch と共に追加されたファイルを削除する。
- 上記と同じ操作をレジストリ エントリで実行する。
- 元のインストールに含まれていたが、現在の QuickPatch プロジェクトには適用しないカスタム アクションを削除する。

QuickPatch プロジェクトの作成は常に、[新規 QuickPatch 作成] ウィザードから始めます。ウィザードを完了すると InstallShield で QuickPatch プロジェクトが開き、プロジェクトの設定を構成できます。

差分リリース

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、このタイプのリリースには指定された 1 つ以上の既存リリースのセットのに存在していなかった (および/または、それらのバージョンと日付、時間、サイズおよび属性が違う) ファイルを含む差分リリースを定義およびビルドすることができます。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。このタイプのアップグレードには、すべてのメンテナンス/アンインストール機能のファイルも含まれます。Data1.hdr、Data1.cab および Layout.bin の新しいバージョンは既存のバージョンを上書きせず、新規のフォルダーに配置されます。

メモ: *差分リリースは、指定された既存のメディアと新しいメディアの間で変更があったファイルからのみ構成されているため、指定された以前のバージョンの製品を持たないシステムに差分リリースをインストールすると、有効なアプリケーションは、ほとんどの場合、作成されません。以前のバージョンの製品がインストールされていないシステムをアップデートするには、フル リリースを作成します。*

差分リリースに含まれるファイル

差分リリースを定義する場合、新しい差分リリース作成時に現在のプロジェクトと比較する複数の既存リリースを指定します。これらの既存のリリースは、リリース ウィザードの [アップデート] パネルで指定します。現在のプロジェクトのファイルは、次のいずれかが該当する場合、差分リリースに含まれます。

- ファイルは、指定された比較リリースの少なくとも 1 つで欠けているか、そのコピーと異なる日時、サイズまたは属性を持つ場合。

- ファイルが、"差分"プロパティが "常に含める" に設定されているコンポーネントにある場合。
- ファイルが、指定の比較リリースの少なくとも 1 つと比べて欠けているコンポーネントにある場合。
- ファイルが、指定の比較リリースの少なくとも 1 つと比べて異なる名前を持つコンポーネントにある場合。
- 指定した比較リリースの少なくとも 1 つとインストール先、言語、またはオペレーティング システムのプロパティが異なるコンポーネントにファイルがある場合。
- ファイルが、指定の比較リリースの少なくとも 1 つと異なる、コンポーネント階層内の名前またはパスを持つコンポーネントと関連付けられたコンポーネントにある場合。

差分リリースのインストール

インストールが初期化されたときに ADDREMOVE システム変数が非ゼロ値になっていると、インストールは自動的に同じ製品コードを持つ以前にインストールされた差分リリースを初期化しようとします。インストールはすべての DISK1TARGET フォルダーのサブフォルダーを列挙し、差分リリースのサブフォルダーを検索します。次の条件がすべて満たされていると、差分リリースが見つかった場合、それがロードされます。

- サブフォルダーには、差分リリースの有効な InstallShield X 以降、InstallShield DevStudio 9 または InstallShield Professional 6 以降の Data1.hdr ファイルである Data1.hdr が含まれます。
- ヘッダーファイルに保存されている製品コードはインストールの製品コードに一致します。
- 差分リリースの製品バージョンは、インストールの製品バージョンより新しくなっています。

完全リリース

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

InstallScript プロジェクトでは、アップグレードのための完全リリースを定義、ビルドすることができます。完全リリースには、インストール プロジェクトにリンクするすべてのファイルが含まれます。製品の以前のバージョンがターゲット システムに既に存在していない場合、このタイプのリリースは、デフォルトで、初回インストールとして動作します。また、以前のバージョンを完全に上書きインストールすることも可能です。

完全リリースを作成して 1 つ以上の以前のバージョンをアップデートするとき、そのリリースが特定のバージョンのみ対象にするかどうかを指定する必要があります。

- **非バージョン固有** – 既定のオプションです。非バージョン固有の完全リリースは、以前のバージョンの製品がインストールされていないシステムにインストールすることができます。また、以前のバージョンの製品はどれもアップデートすることができます。
- **バージョン固有** – バージョン固有の完全リリースは、更新するように指定したバージョンのみ更新します。完全リリースをバージョン固有と設定して、バージョン番号を指定しないでおくと、アップデートは以前のバージョンすべてに適用されます。

最適なアップグレード ソリューションの決め方

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*
- *QuickPatch*

製品のアップグレードの作成に使用される方法は、オリジナルのインストール パッケージがどのように開発されたかによって変わってきます。オリジナルのインストールが Windows Installer ベースのプロジェクトで作成された場合、基本的に製品のアップデートにはメジャー、マイナーまたはスモールアップデートを作成します。元のインストールが InstallScript プロジェクトで作成された場合は、差分または完全リリースを作成して製品を更新します。これらの両方の種類についての詳細については以下をお読みください。

基本の MSI プロジェクトおよび InstallScript MSI プロジェクト

アップグレードのインストール作成は、そのタイプに関わらず、製品の以前のバージョンを持っていないターゲット システムを対象にするかどうかを決めるところから始まります。それが決まると、アップグレードのパッケージにどの種類の方法を使用するかを決めることができます。以下のテーブルは最適な方法を選ぶときに目安となる一般的な概要です。アップグレードのパッケージに関するテクニックについてより掘り下げた内容については、「アップグレードのパッケージ オプション」を参照してください。

テーブル 1 ・ Windows Installer ベースのプロジェクトの可能なアップグレード ソリューション

ターゲット システムの状況	必要なインストール の種類	アップデート パッケージの技法
ターゲット システムの中には以前のバージョンの製品がインストールされているものもあり、製品がまったくインストールされていないものもある。	ファイルのサイズが問題ではない場合、以下の両方を行うインストールを作成することも可能です。 ・ ターゲット システムに以前のバージョンが存在していない場合、初回インストールとして作動する。 ・ 既にターゲット システムにインストールされている場合、既存の製品をアップデートする。	メジャーまたはマイナーアップグレードを作成して、それを完全インストールとしてパッケージします。

テーブル 1・Windows Installer ベースのプロジェクトの可能なアップグレード ソリューション（続き.）

ターゲット システムの状況	必要なインストール の種類	アップデート パッケージの技法
ターゲット システムの中には以前のバージョンの製品がインストールされているものもあり、製品がまったくインストールされていないものもある。	以前のバージョンの製品をアップグレードを必要とするエンドユーザーのためのスモール インストールが必要な場合、2 つのインストールを分けて作成することができます。 ・ 初回インストールとして作動する完全インストール。 ・ 既にインストールされている製品の 1 つまたは複数の以前のバージョンをアップデートする小規模のインストール。このインストールにはバージョンとバージョンの間で変更があったデータのみが含まれているので、完全インストール パッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。	初回インストールの場合、メジャーまたはマイナー アップグレードを作成して、それを完全インストールとしてパッケージします。製品の以前のバージョンを使用しているエンドユーザーのためには、メジャーまたはマイナーアップグレードを作成して、それをパッチとしてパッケージします。 ***ヒント:*** *場合によって、パッチが適当ではないことがあります。パッチがアップグレードに適切かどうかを判断するための詳しいガイドラインは、「アップグレードのパッケージ オプション」を参照してください。* ***メモ:*** *パッチ パッケージはアップグレードとは違いますので注意してください。パッチは簡単にいうと、省スペースでアップグレードを配布するためのメカニズムです。*
すべてのターゲット システムが製品の以前のバージョンを持っている。	既にインストールされている製品の 1 つまたは複数の以前のバージョンをアップデートする小規模のインストールを作成することができます。このインストールにはバージョンとバージョンの間で変更があったデータのみが含まれているので、完全インストール パッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。	以下のいずれかを選択します ： ・ メジャーまたはマイナーアップグレードを作成して、それを標準パッチとしてパッケージ。 ・ QuickPatch プロジェクトの作成。 InstallShield の QuickPatch 技術を利用して以前にインストールされた製品のためのシンプルなパッチを作成することができます。QuickPatch プロジェクトを使ってパッチを作成する場合、カスタマイズの範囲に制限がありますが、標準パッチに較べると QuickPatch プロジェクトのほうが簡単に作成することができます。詳細については、「パッチと QuickPatch プロジェクトの違い」を参照してください。 ***メモ:*** *標準のパッチ同様、QuickPatch は、省スペースでアップグレードを配布するためのメカニズムです。*

InstallScript プロジェクト

元のインストールが InstallScript プロジェクトで作成された場合は、2 種類のアップデート リリースの中の 1 つを作成することができます。

- **差分リリース** – この種類のリリースには、1 つ以上の指定された既存リリースのセットに存在していなかった (および/または、それらのバージョンと日付、時間、サイズおよび属性が違う) ファイルが含まれています。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。
- **完全 (非差分) リリース** – ターゲット システムに以前のバージョンの製品が存在していない場合、このタイプのリリースは初回インストールとして動作します。また、以前のバージョンを完全に上書きインストールすることも可能です。

差分インストールはバージョンとバージョンの間で変更があったデータのみが含まれているので、完全インストール パッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。但し、完全リリースのみ、製品が全くインストールされていないシステムへのインストールが可能です。詳細については、「差分リリース」および「完全リリース」を参照してください。

このセクションには、次のトピックが含まれています:

- メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い
- 自動アップグレード
- パッチと QuickPatch プロジェクトの違い
- アップグレードのパッケージ オプション
- 差分リリースと完全リリース

メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

Windows Installer は 3 種類 (メジャーアップグレード、マイナーアップグレード、スモールアップデート) の製品アップグレードをサポートしています次のテーブルは、ユーザーのニーズに最も適したアップグレードの種類を決定するお手伝いをします。アップグレードの要件がマイナーアップグレードまたはスモールアップデートにひ

とつでも適さない場合、メジャーアップグレードを作成します。Windows Installer ベースのプロジェクトにどのタイプのアップグレードを利用すべきかが分からない場合、または特に希望するタイプがない場合は、自動アップグレードを作成することができます。

テーブル 2 - メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い

アップグレードの要件	メジャー アップグレードの使用	マイナー アップグレードの使用	スモール アップグレードの使用	メモ
.msi パッケージ名の変更	はい	いいえ	いいえ	.msi ファイルが Setup.exe インストールのランチャ に圧縮されていなければ、デフォルトのファイル名は ″製品名″ プロパティから取得されます。
エンドユーザーが以前のバージョンと最新のバージョンを同一マシンにインストールできるようにする	はい	いいえ	いいえ	
新しいサブ機能を追加する	はい	場合による	場合による	新規のサブ機能が新規のコンポーネントのみで構成されている場合、スモールアップデート、マイナーアップグレード、またはメジャーアップグレードを利用することができます。新規のサブ機能が既存のコンポーネントで構成されている場合、メジャーアップグレードを利用しなければなりません。
製品ツリーで機能を移動または削除する	はい	いいえ	いいえ	
新規の機能に新規のコンポーネントを追加する	はい	はい	はい	
既存の機能に新規のコンポーネントを追加する	はい	可。ただし Windows Installer が 2.0 以降の場合	可。ただし Windows Installer が 2.0 以降の場合	Windows Installer 1.x には、アップグレード パッケージで、マイナーアップグレードおよびスモールアップデートの新規の機能に配置する新規のコンポーネントが必要です。また、特殊コマンドライン処理も同様に必要です。
製品ツリーでコンポーネントを移動または削除する	はい	いいえ	いいえ	

テーブル 2・メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い (続き.)

アップグレードの要件	メジャー アップグレードの使用	マイナー アップグレードの使用	スモール アップグレードの使用	メモ
既存のコンポーネントのコンポーネントコードを変更する	はい	いいえ	いいえ	
コンポーネントのキーファイルを変更する	はい	いいえ	いいえ	
ファイル、レジストリキー、またはショートカットのいずれかを追加、削除または変更する	はい	はい	はい	ファイル、レジストリキーまたはショートカットが複数のコンポーネントに存在して、そのコンポーネントが 2 つ以上の機能で共有されている場合、メジャーアップグレードを利用する必要があります。
MSI データベースまたはトランスフォーム プロジェクトで始める	はい	いいえ	いいえ	

異なるタイプのアップグレードに関連するコード

いくつかの Windows Installer コードは製品を識別するのに役立ちます。

- **パッケージ コード** – 概要情報ストリームの一部。特定のデータベースを認識します。パッケージコードは、Windows Installer プロパティではありません。同じパッケージコードを持つ 2 つの .msi データベースの内容は同じでなければなりません。したがって、ビルドごとにパッケージコードを変更する必要があります。
- **ProductVersion** – これは、製品バージョンを含む Windows Installer プロパティです。Windows Installer は、バージョンを比較する際、ProductVersion プロパティの最初の 3 つのフィールドのみを利用することに注意してください。たとえば、1.2.3.4 の製品バージョンの場合、4 は無視されます。(これは、ProductVersion 値の比較にのみ当てはまり、ファイル バージョンの比較には当てはまらないので注意してください。)
- **ProductCode** – これは、製品の GUID を含む Windows Installer プロパティです。たとえ ProductName プロパティの値が同じでも、Windows Installer は ProductCode GUID が異なる 2 つの製品は相互に関連が無いものとして扱います。
- **UpgradeCode** – これは、関連製品を示す GUID を識別する Windows Installer プロパティです。インストール済みの製品の関連バージョンを検索できるように、UpgradeCode は、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があります。[アップグレード] ビューでアップグレードの UpgradeCode を設定することができます。

どの種類の再配布可能ファイルを表示するのかを指定するには、[表示するオブジェクトの種類] リストで適切なオプションを選択します。次のテーブルで、異なるタイプのアップグレードに各コードをいつ変更すべきかが分かります。

テーブル 3 • 異なる種類のアップグレードごとに変更する必要があるコード

	パッケージ コード	製品バージョン	製品コード	アップグレード コード
スモール アップデート	X			
マイナーアップグレード	X	X		
メジャー アップグレード	X	X	X	

自動アップグレード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

アップグレード作成過程を簡素化するため、InstallShield には自動アップグレードの作成機能が搭載されています。自動アップグレードは、どの種類のアップグレードの種類を利用すれば良いのか分からない場合、または特にアップグレードの種類にこだわらない場合に利用できる、特殊な種類のアップグレードです。自動アップグレードを利用すると、指定したパッケージの新旧を比較して InstallShield がメジャーアップグレードまたはマイナーアップグレードのどちらが最適かを判断します。

パッチと QuickPatch プロジェクトの違い

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

完全に更新されたインストール パッケージを配布せずに製品を更新する場合は、Windows Installer パッチ パッケージ (.msp) を作成することができます。InstallShield では、異なるシナリオに対応する異なる 2 つのパッチ作成方法 ([パッチのデザイン] ビューを使用する方法、および QuickPatch プロジェクト タイプを使用する方法) が提供されています。次の両方の選択肢を読んで、製品に適切なソリューションを見極めます。

メモ: メジャーアップグレードは、通常パッチとしてパッケージされません。従って、下のテーブルは [パッチのデザイン] ビューで作成されたパッチはマイナーアップグレードまたはスモールアップデート用と仮定します。QuickPatch プロジェクトは、マイナーアップグレードとして機能する QuickPatch パッケージを作成します。

テーブル 4 · パッチと QuickPatch プロジェクトの違い

パッチの要件	パッチの使用	QuickPatch の使用	メモ
ベース パッケージをアップデートする多くの累積パッチを適用できる機能	はい	可 (QuickPatch の簡素化を使用する場合)	QuickPatch の簡素化を使用しない場合、15 回以上パッチを適用することはできません。詳細については、「QuickPatch パッケージを簡素化するかどうかを指定する」を参照してください。
新しいサブ機能を追加する	はい	いいえ	
新規のサブ機能に新規のコンポーネントを追加する	はい	いいえ	
既存の機能に新規のコンポーネントを追加する	可。ただし Windows Installer が 2.0 以降の場合	いいえ	Windows Installer 1.x には、アップグレード パッケージで、マイナーアップグレードおよびスモールアップデートの新規の機能に配置する新規のコンポーネントが必要です。また、特殊コマンドライン処理も同様に必要です。
ファイルの追加、変更、または削除	はい	はい	QuickPatch で、新しいファイルに新しいターゲットの場所を使用することはできません。新しいファイルには、元のインストールで既に定義されている場所のみを使用できます。[パッチのデザイン] ビューを使って作成したパッチには、この制限はありません。

テーブル 4 • パッチと QuickPatch プロジェクトの違い (続き.)

パッチの要件	パッチの使用	QuickPatch の使用	メモ
レジストリデータの追加、変更、または削除	はい	はい	QuickPatch で追加する新規レジストリ データはすべて、既に元のインストールに存在する機能に関連付ける必要があります。
ショートカットの追加、変更、または削除	はい	いいえ	
カスタム アクションの追加、変更、または削除	はい	オリジナルのベースインストールに含められたカスタム アクションのみ削除することができます。	
再配布可能ファイルの追加または削除	はい	いいえ	
ODBC リソースの追加、変更、または削除	はい	いいえ	
.ini ファイルの編集	はい	いいえ	
.xml ファイルの編集	はい	いいえ	
IIS Web サイト、コンポーネントサービス、および SQL スクリプト等のサーバー設定を構成する	はい	いいえ	
パッチ パッケージ内のファイル (例、アプリケーションの実行可能ファイル) にデジタル署名する。	はい	自動ではない	QuickPatch プロジェクトの場合、個々のファイルを手動で署名してから、プロジェクトに追加する必要があります。

[セットアップデザイン] ビューの使用

[パッチのデザイン] ビューではビジュアルに統合された方法でパッチを作成すると共に、各パッチ構成に関連する適切な設定を選択することができます。[パッチのデザイン] ビューでは複数パッチ構成を作成することも可能です。各パッチ構成にはパッチをビルドするために必要な設定とデータが含まれます。

ほとんどの場合、[パッチのデザイン] ビューからパッチ作成を開始します。簡単なパッチソリューションが必要な場合は QuickPatch プロジェクトを作成することができます。

QuickPatch プロジェクトを利用する

QuickPatch プロジェクトを作成したときに起動される新規 QuickPatch 作成ウィザードは、規模の小さいアップデートをエンドユーザーに向けに作成するインストール作成者を対象としています。これは [パッチのデザイン] ビューでのパッチ作成よりも簡単です。

QuickPatch プロジェクトを使ってパッチを作成する場合、カスタマイズの範囲に制限がありますが [パッチのデザイン] ビューと同じ種類のパッチ (.msp と .exe) が作成されます。

アップグレードのパッケージ オプション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

エンド ユーザーのマシン上にインストール済みの製品のバージョンを更新するインストールを作成する場合、メジャー アップグレード、マイナー アップグレード、またはスモール アップデートをパッケージする 2 つの選択肢があります。

- 以前のバージョンがインストールされている場合は既存製品をアップデートし、以前のバージョンが存在しない場合は初回インストールとして作動する完全インストールとして、アップグレードをパッケージすることが可能です。
- アップデートするバージョン間で変更されているデータ (.msi データおよびバイトレベルのファイル差分) のみを含むパッチとしてアップグレードをパッケージすることが可能です。

フルインストール パッケージ

マイナーアップグレードとスモールアップデートは共に、以前のバージョンが存在しない場合は初回インストールと同様に作動し、製品が既にインストールされている場合、その上からインストールを行います。メジャーアップグレードも、以前のバージョンが存在する場合は初回インストールと同様に作動しますが、メジャーアップグレードは一般的に以前のバージョンすべてをアンインストールしてから新しいバージョンをインストールします。

パッチパッケージ

パッチを利用すると、アプリケーションのファイルを特定バージョンにアップデートするのに必要な、データベースの一部のみを配布することができます。このため、完全インストールとしてパッケージされたアップグレードに比べて規模の小さいパッケージを作成することが可能です。つまり、完全インストールパッケージの配布よりも狭い帯域幅を利用したアップグレードの配布が可能となります。

メモ: *パッチはアップグレードの一種ではありません。パッチは簡単にいうと、省スペースでメジャーアップグレード、マイナーアップグレード、またはスモールアップデートを配布するためのメカニズムです。実際、パッチを作成するためにはまず、アップグレードを設計してからそれをパッチとしてパッケージします。パッチを作成する前に、アップグレードをフル インストールパッケージとしてテストすることをお勧めします。*

アップグレードの最適なパッケージオプションの決定方法

「最適なアップグレード ソリューションの決め方」トピックには、以前のバージョンの更新のために最も適切なパッケージの種類を判断するときに参考となるテーブルが掲載されています。標準パッチまたは QuickPatch がアップグレードのパッケージ方法として最も適切なメカニズムの場合もあります。しかし状況によって、パッチの代わりに完全インストールとしてアップグレードをパッケージする方法が相応しい場合もあります。例:

- ターゲットイメージが Windows Installer 1,2 以前で作成され、アップグレードイメージが Windows Installer 2.0 以降で作成された場合、アップグレードはパッチではなく完全インストールとしてパッケージする必要があります。このスキーマの違いを超えたパッケージにパッチを作成すると、問題が発生します。詳しい情報については、アップグレードおよびパッチ検証用の関連検証ツールついて説明されている Val0011 をご覧ください。
- アップグレードがターゲット システム上で 1 つまたは複数のファイルを別の場所に移動する場合、パッチではなく完全インストールとしてアップグレードをパッケージしなくてはなりません。エンドユーザーがターゲット システム上にあるファイルを移動するアップグレード用のパッチをインストールすると、問題が発生する可能性があります。たとえば、パッチが作動しない、システム修復が作動しない、後続のパッチが作動しない、またエンドユーザーが製品をアンインストールできない等。ワークアラウンドとして、古いロケーションにあるファイルを削除して新しいロケーションにファイルを追加するパッチを作成することができます。
- インストールを圧縮形式から非圧縮形式に変更する場合、またはその逆を行なう場合、アップグレードはパッチではなく完全インストールとしてパッケージしなくてはなりません。このシナリオでパッチを利用すると、ターゲット システムの修復が作動しない、後続のパッチが作動しない、またはエンドユーザーが製品をアンインストールできない場合があります。
- ファイルを .cab ファイル間で移動させる必要がある場合、または .cab ファイル内でファイル順を変更する必要がある場合、アップグレードをパッチではなく完全インストールとしてパッケージしなくてはなりません。
- オリジナルインストールのファイル数が 32,767 より大きいときに、最新版のインストールに含まれるファイルが 32,767 未満の場合、パッチは失敗します。同様に、オリジナルインストールのファイル数が 32,767 未満のときに、最新版のインストールに含まれるファイルが 32,767 より大きい場合、パッチは失敗します。どちらの場合も、アップグレードをフルインストールとしてパッケージする必要があります。

  オリジナルインストールと、最新版インストールの両方のファイル数が 32,767 より大きい (または両方のファイル数が 32,767 未満) 場合、アップデートをパッチとしてパッケージすることが可能です。
- パッチは、InstallScript MSI プロジェクトのメージャーアップグレードには作成できません。従って、InstallScript MSI プロジェクトのメジャーアップグレードを配布する必要がある場合、完全インストールとしてアップグレードをパッケージします。

差分リリースと完全リリース

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

製品の元のインストールが InstallScript プロジェクトで作成された場合、製品のアップデートに差分または完全リリースを作成する必要があります。

以前のバージョンを持つターゲット システムや、製品がまったくインストールされていないターゲット システムがある場合、アップグレードを完全リリースとしてパッケージすることができます。以前のバージョンがターゲット システムに既に存在していない場合、この種類のリリースは、デフォルトで初回インストールとして動作します。また、以前のバージョンを完全に上書きインストールすることも可能です。

すべてのターゲット システムが以前のバージョンを持っている場合、アップグレードを差分リリースとしてパッケージできます。この種類のリリースには、1 つ以上の指定された既存リリースのセットに存在していなかった (および/または、それらのバージョンと日付、時間、サイズおよび属性が違う) ファイルが含まれています。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。

メモ: *InstallShield Professional 6.0 よりも古いバージョンを利用して InstallScript プロジェクト用のメディアを作成した場合、メディアをアップデートすることはできません。*

アップグレード、パッチ、および QuickPatch プロジェクトを使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

MSI データベースおよびトランスフォーム プロジェクトはメジャー アップグレードをサポートしますが、マイナー アップグレード、スモール アップデート、パッチ、または QuickPatch パッケージはサポートしません。

アップグレードをフルインストール、または標準パッチとしてパッケージする場合、まずプロジェクトの最新版を開いて、必要に応じてファイルおよびレジストリ エントリを追加するなどの変更を加えます。

QuickPatch としてアップグレードをパッケージする場合、まず新しい QuickPatch プロジェクトを作成します。QuickPatch プロジェクトでは、QuickPatch を使ってどの以前のリリースをパッチするかを指定します。

メジャーアップグレード、マイナーアップグレード、スモールアップデート、自動アップグレード、パッチ、および QuickPatch プロジェクトの作成方法についての詳細は、このセクションのトピックをご覧ください。このセクションではまた、でき上がったパッケージが適切に役割を果たすかどうか、アップグレードまたはパッチを検証する方法を説明します。

このセクションには、次のトピックが含まれています:

- ファイルの上書き規則について
- アップグレード関する考慮事項
- InstallShield を構成してアップグレードの種類を自動判別する
- メジャー アップグレードを作成する
- マイナー アップグレードを作成する
- スモール アップデートを作成する
- パッチ時の考慮事項
- パッチの作成と適用
- QuickPatch プロジェクトを作成する

- アップグレード、パッチ、および QuickPatch パッケージを検証する
- グローバル アセンブリ キャッシュのアセンブリをパッチする
- 差分リリースと完全リリース

ファイルの上書き規則について

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

Windows Installer サービスは、デフォルトで、アップグレードが含むファイルがターゲット システムに既に存在するファイルを上書きするかどうかを判断するときいくつかのファイルの上書き規則を使用します。これらの規則は、REINSTALLMODE プロパティが o 設定を使用してターゲット システムの古いファイルを上書きインストールするときに適用されます。この動作を変更するには、o オプションを次の値の中から 1 つを使って置き換えます。

- p ― ターゲット システム上に対応するファイルがない場合のみ再インストールします。
- e ― ファイルが見つからないか、または、バージョンが古いか同じなとき再インストールします。
- d ― ファイルが見つからないか異なるとき、再インストールします。
- a ― バージョンに関わらず、すべてのファイルを再インストールします。

REINSTALLMODE の設定は、インストール時にすべての機能に適用されるので、個別に設定することはできません。また、REINSTALLMODE に a を含めるように設定すると、パッチの適用中にオリジナルのインストールソースを求めるプロンプトを引き起こす可能性があります。

File Versioning Rules (Windows Installer ヘルプ ライブラリ)
REINSTALLMODE Property (Windows Installer ヘルプ ライブラリ)

アップグレード関する考慮事項

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

MSI データベースおよびトランスフォーム プロジェクトはメジャー アップグレードをサポートしますが、マイナー アップグレード、スモール アップデート、パッチ、または QuickPatch パッケージはサポートしません。

以下は、アップグレード作成のためのいくつかのガイドラインです。

パッケージ コード、製品バージョン、および製品コードのアップデート

どの種類の再配布可能ファイルを表示するのかを指定するには、[表示するオブジェクトの種類] リストで適切なオプションを選択します。詳細については、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。

アップグレードとパッチの最適化

マイナー アップグレードまたはスモール アップデートを構成するとき、ビルドの設定にあるパッチの最適化機能を利用します。

パッチの最適化を行うと、コンポーネントの名前、コンポーネント GUID、File テーブルのキー、および Directory テーブルのキーを、アップグレードされたイメージとターゲットのイメージとの間で同期することができます。これにより、パッチのサイズを最も小さくすることができます。パッチの最適化を行わなかった場合、パッチとターゲットの .msi パッケージのサイズが同じになってしまったり、Windows Installer のコンポーネントの規則に違反したりします。

パッチ作成プロセスは File テーブル キーを使って、2 つのファイルが同じファイルかどうかを判断します。(実際のファイル名は使用することができません。これは、パッケージが異なる条件下でインストールされた同じ名前のファイルを複数含んでいる可能性があるからです。)パッチの最適化で以前のパッケージを指定すると、同一のファイルに対して同一の File テーブル キーが使用されます。

ヒント: *パッケージがダイナミック ファイル リンクを使用する場合、アップグレードをビルドするときにパッチの最適化を使用することが推奨されます。これによって、リリース間でキーファイルの一貫性が保たれます。*

パッチの最適化機能を利用するには、リリース ウィザードの [詳細設定] パネルで Windows Installer パッケージを指定します。[リリース] ビューでそのリリースについての [ビルド] パネルにある "以前のパッケージ" 設定でも以前の Windows Installer パッケージを指定することができます。

マイナーアップグレードでシグネチャを追加する

一般的に、マイナーアップグレードには新規の最上位機能を含めることはできません。但し、既存の機能の新規サブ機能は許可されます。マイナー アップグレードに新規のサブ機能を追加するとき、それらがマイナーアップグレード中に適切にインストールされるように次のようにサブ機能の設定を 2 つ設定します。

- リモート インストール— **[親を優先]** を選択します。
- 必須—**[はい]** を選択します。

マイナーアップグレードのユーザー インターフェイスは、通常機能ツリーを表示しませんが、アップデートされたインストールのメンテナンス モードは、通常、通常機能ツリを表示します。エンドユーザーが新規のサブ機能を選択解除できないようにそれを機能ツリーから取り除くには、サブ機能の "表示" 設定を [非表示] に設定します。

インストール済みデータの削除

マイナーアップグレードがファイル、レジストリ設定、または、.ini ファイル等のデータを削除する場合、以前の製品バージョンに存在していたデータをプロジェクトから単純に削除しても、ファイルはアップグレード適用時に削除されません。このようなデータの削除方法については、「マイナー アップグレードを構成してインストール済みのデータを削除する」をご覧ください。

InstallShield を構成してアップグレードの種類を自動判別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer ベースのプロジェクトにどのタイプのアップグレードを利用すべきかが分からない場合、または特に希望する種類がない場合は、自動アップグレードを作成することができます。自動アップグレードを利用すると、指定したパッケージの新旧を比較して InstallShield がメジャーアップグレードまたはマイナーアップグレードのどちらが最適かを判断します。このセクションのトピックでは、自動アップグレードの作成および処理方法を説明します。

自動アップグレード アイテムの追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***自動アップグレード アイテムを追加するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで **[Windows Installer インストールのアップグレード]** を右クリックして、**[自動アップグレード アイテムの追加]** をクリックします。自動アップグレード項目がエクスプローラーへ追加されます。
3. インストールの最新版に対応するようにアイテムの名前を変更します。

自動アップグレード アイテムの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

自動アップグレード アイテムをプロジェクトに追加する場合、自動アップグレード アイテムの設定を構成する必要があります。設定の構成には、[アップグレード] ビューの [アップグレード シナリオのセットアップ準備] エクスプローラーで新規の自動アップグレード アイテムをクリックします。

また、InstallShield が自動アップデートアイテムにメジャーまたはマイナーアップグレードが必要と判断するかどうかによって、InstallShield がアップグレードの作成に使用する一般アップグレードの設定も構成する必要があります。[アップグレード] ビューのエクスプローラーの [アップグレードシナリオのセットアップ準備] エクスプローラーの [Windows Installer セットアップのアップグレード] をクリックしたときに表示されるタブでこれらの全般設定を構成することができます。

- [共通] タブ
- [詳細] タブ

自動アップグレードからメジャーまたはマイナーアップグレードへ変換する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***自動アップグレード アイテムをメジャーまたはマイナーアップグレードへ変換するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. 変換する自動アップグレード アイテムを右クリックしてから **[マイナーアップグレード アイテムに変換]** または、**[メジャーアップグレード アイテムに変換]** をクリックします。

自動アップグレード アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***自動アップグレード アイテムを追加するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. 削除する自動アップグレード アイテムを右クリックしてから、**[削除]** をクリックします。

メジャー アップグレードを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

メジャー アップグレードが状況に適した最も有効なアップグレード ソリューションであると判断したならば、[アップグレード] ビューでメジャー アップグレード アイテムを作成開始します。メジャーアップグレードは機能の重要な変更を意味し、[一般情報] ビューでアップデートする ProductCode プロパティの変更を必要とします。

メモ: *製品コードを変更すると、*ProductName *値が同じでも、Windows Installer は製品の最新版と以前のバージョンが関連しないものとして処理します。製品の両方のバージョンを同じシステム上にインストール可能にするには、製品コードおよびメインとなるインストールディレクトリ (多くの場合、INSTALLDIR) の両方を変更するだけです。*

基本的に、メジャー アップグレードでは 2 つの処理が 1 つのインストール パッケージに組み込まれています。メジャー アップグレードは、製品の新しいバージョンをインストールしてから古いバージョンをサイレントでアンインストールか、古いバージョンをサイレントでアンインストールしてから新しいバージョンをインストールします。これら 2 つの個別の処理のシーケンスは、製品の新しいバージョンのインストール パッケージ構成方法によって異なります。

InstallShield を使用してのメジャー アップグレードの作成に関する詳しい情報ついては、InstallShield ヘルプライブラリ内の該当セクションを参照してください。

メジャー アップグレード アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***メジャー アップグレード アイテムを追加するには、以下の手順に従います:***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ準備]** エクスプローラーで **[Windows Installer セットアップのアップグレード]** を右クリックして、**[メジャー アップグレード アイテムの追加]** をクリックします。

新しいメジャー アップグレード アイテムが追加されます。

メジャー アップグレードの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アップグレード] ビューでメジャーアップグレード アイテムを作成する場合、変更可能な一般設定は、RemoveExistingProducts アクションのスケジュール場所です。この設定は、"Windows Installer セットアップのアップグレード" を選択してから [共通] タブをクリックし、[アップグレード] ビューで変更することが可能です。

メジャーアップグレード アイテムの構成可能な設定は、ダイレクト エディターの Upgrade テーブルよりもユーザーフレンドリーなフォーマットで表示されます。

アップグレードアイテムの [共通] タブでは、アップデートする製品バージョンと共に、アップデートする製品のアップグレードコード値を指定します。[詳細]タブで、ISSetAllUsers カスタム アクションはデフォルトで含まれます。このアクションは、メジャー アップグレード中に ALLUSERS を適切に設定します。(さらに、メジャーアップグレードが実行される場合、ISSetAllUsers はカスタムプロパティ IS_MAJOR_UPGRADE を設定します。)ISSetAllUsers アクションを含めるかどうかを指定するには、[オプション] ダイアログ ボックスの [一般] タブを使用します。[オプション] ダイアログ ボックスにアクセスするには、[ツール] メニューから [オプション] を選択します。そのタブで "**ISSetAllUsers アクションを自動的に作成**" チェック ボックスを選択または非選択します。

メジャー アップグレード アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

タスク ***[アップグレード] ビューからメジャー アップグレード アイテムを削除するには、以下の手順を実行します。***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで メジャー アップグレード アイテムを右クリックして、**[削除]** をクリックします。

マイナー アップグレードを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

マイナー アップグレードが状況に適した最も有効なアップグレードソリューションであると判断すれば、[アップグレード] ビューで マイナー アップグレード アイテムを追加して作業を開始することができます。マイナー アップグレード アイテムでインストールをビルドすると、ビルドエンジンは一連のテストを実行し、最新版のインストールが以前のバージョンのインストールを確実にアップグレードするかどうかを判別します。ただし、マイナー アップグレード アイテムの追加は、マイナー アップグレードが適切に機能するために不可欠という訳ではありません。詳細については、「マイナー アップグレードの実行時の動作」を参照してください。

インストールの最新バージョンに追加する新しい機能は、既に存在する機能のサブ機能として追加しなくてはなりません。これらの新しい機能を使用する場合、[機能] ビューで "リモート インストール" 設定に [親を優先]、"必須" 設定に [はい] を選択する必要があります。

マイナー アップグレード アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***[アップグレード] ビューでマイナー アップグレード アイテムを追加するには、以下の手順を実行します。***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで **[Windows Installer インストールのアップグレード]** を右クリックして、**[マイナー アップグレード アイテムの追加]** をクリックします。

新しいメジャー アップグレード アイテムが追加されます。

ヒント: *複数の以前のバージョンをアップデートするアップグレードを作成する場合、追加のマイナーアップグレード アイテムを作成することができます。*

マイナー アップグレードの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[アップグレード] ビューでマイナーアップグレード アイテムの追加を行なう場合に、変更可能な一般設定は、製品の以前のバージョンが存在する場合に (ビルド構成がセットアップランチャーを含む場合) セットアップランチャーがどのように作動するかです。この設定は、"Windows Installer セットアップのアップグレード" を選択してから [共通] タブをクリックし、[アップグレード] ビューで変更することが可能です。

マイナーアップグレード アイテムを [アップグレード] ビューに追加してからそれが選択されていることを確認した後、そのマイナーアップグレード アイテムの "アップグレードするセットアップ" 設定でターゲットとする製品の以前のバージョンの .msi データベースを参照することができます。

"リリースフラグ" 設定のリリースフラグを指定し、ビルドからマイナーアップグレードアイテムを除外することもできます。

マイナー アップグレードを構成してインストール済みのデータを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield は、インストール済みデータを削除するマイナーアップグレードの構成について、2 つの異なる方法をサポートします。次のセクションで、2 つの方法について説明します。

方法 1

以前のリリースでインストールされたデータを削除するマイナーアップグレードを構成するには、アップグレードを構成する InstallShield プロジェクトを開いて、[ファイルとフォルダー] ビューでそのファイルを削除します。そしてダイレクト エディター内の Remove テーブルに対応するレコードを作成します。ダイレクト エディターの Remove テーブルには、RemoveFile、RemoveIniFile、および RemoveRegistry テーブルがあります。

マイナーアップグレードで、コンポーネントからレジストリデータを削除するには、RemoveRegistry テーブルにレコードを作成する必要があります。RemoveRegistry テーブルのレコードは、関連するコンポーネントがインストールされた時に削除するレジストリキーおよび値について説明します。RemoveFile テーブルとは異なり、RemoveRegistry テーブルは関連するコンポーネントがアンインストールされた時に指定されたレジストリデータを削除するオプションを受け付けません。その代わり、"**すべてのキーをアンインストール**" フラグを使ってレジストリ値を作成することが可能です。コンポーネントが、[名前] フィールドにハイフンを用いたレジストリ値を含み、[値] フィールドが空白の場合、指定したレジストリキーおよびその内容は、コンポーネントが削除される時に削除されます。

その他のデータ型では通常、Windows Installer テーブルでアンインストールフラグの利用が可能か、対応するアンインストールテーブルを利用することができます。たとえば .ini データを削除する場合、RemoveIniFile テーブルを利用することができます。環境変変数データには、対応するアンインストールフラグがあります。

メモ: *コンポーネントが別の製品と共有されている場合、リソースを削除する時にコンポーネントコードを変更する必要があります。さらに、既存するコンポーネントのコンポーネントコードを変更するには、メジャーアップグレードが必要です。メジャーアップグレードの場合、完全なアンインストールが必要なため、[Remove] テーブルにデータを入力する必要はありません。*

方法 2

コンポーネントを 移行可能と設定し、マイナーアップグレードが適用された時に false 評価されるコンポーネントの条件を設定します。

RemoveFile Table (Windows Installer ヘルプ ライブラリ)
RemoveIniFile Table (Windows Installer ヘルプ ライブラリ)
RemoveRegistry Table (Windows Installer ヘルプ ライブラリ)

マイナー アップグレード アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***[アップグレード] ビューでマイナー アップグレード アイテムを削除するには、以下の手順を実行します。***

1. **[アップグレード]** ビューを開きます。
2. **[アップグレード シナリオのセットアップ作成]** エクスプローラーで、マイナー アップグレード アイテムを右クリックして、**[削除]** をクリックします。

マイナー アップグレードの実行時の動作

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Setup.exe を使ってマイナー アップグレードを実行する

Setup.exe 起動プログラムを含んだリリースをビルドすると、最新のインストールはマイナー アップグレードが可能になります。Setup.exe はアプリケーションの古いバージョンが存在するときにそれを検出することができます。Setup.exe が以前のバージョンを検出した場合、インストールの続きはマイナーアップグレードモードで実行します。

製品の以前のバージョンがインストールされているターゲット システム上でエンド ユーザーがマイナー アップグレードを実行した場合、アップグレードが行われることを通知するメッセージ ボックスが表示されます。このメッセージ ボックスは以前の製品名を指定し、エンドユーザーに対してどの既存アプリケーションをアップデートするのかを警告します。この時点で、エンドユーザーは必要に応じてインストール処理を中止することができます。このプロンプトの後には [ようこそ] ダイアログが続き、エンドユーザーが [ようこそ] ダイアログの [次へ] をクリックすると、セットアップが新しいリソースをインストールします。リソースのインストール中、インストールは [セットアップの進行状況] ダイアログの進行状況バーを更新して、ユーザーにアップグレードの処理状況を追跡できるようにします。

ヒント: *[アップグレード] ビューでは、エクスプローラーの Windows Installer セットアップのアップグレードをクリックして、スモール/マイナーアップグレード用の [共通] タブの設定を構成し、最初のアップグレード プロンプトを無効にすることができます。*

メモ: *マイナー アップグレードの実行時に、コンポーネントのキー ファイルが処理中にダウングレードされる場合、Windows Installer はそのコンポーネントの再インストールを行いません。このため、アップグレードのキー ファイルは、古いパッケージのキー ファイルのバージョン番号と等しいか、それよりも大きい番号でなくてはな*

りません。例外は、エンド ユーザーが REINSTALLMODE プロパティに ***a*** *を設定した場合です。この設定は、バージョンに関係なくすべてのファイルを強制的にインストールします。ファイルの上書き規則に関する情報は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。*

マイナーアップグレードは、ターゲットマシンに既に存在するアプリケーションの上に新しいリソースをインストールします。アプリケーションを完全にアンインストールしてから再インストールするには、メジャーアップグレードの利用を検討してください。

Setup.exe を使わないマイナー アップグレードを実行する

インストールを Setup.exe ランチャーなしで配布する場合、エンド ユーザーはコマンドラインを使用してインストールを開始する必要があります。このため、Setup.exe を使わなくても同じような結果が得られるものの、Setup.exe の利用をお勧めします。

インストーラープロパティ、REINSTALLや REINSTALLMODE はアップグレードモードでインストールをスタートさせるためにコマンドラインから設定する必要があります。最も複雑なシナリオ以外、通常は REINSTALLMODE プロパティは、**vomus** に、REINSTALL プロパティは **ALL** に設定しなくてはなりません。標準的なコマンド次のとおりです:

```
msiexec.exe /i ¥product.msi REINSTALLMODE=vomus REINSTALL=ALL
```

アップデートを行なわない機能が含まれる場合、次のコマンドの通り REINSTALL をコンマ区切りのアップデートする機能リストに設定します。

```
msiexec /i ¥product.msi REINSTALLMODE=vomus REINSTALL=F1,F3,F5
```

REINSTALL プロパティで利用する機能名は大文字と小文字を区別します。

メモ: *初回インストールで* REINSTALL = **ALL** *と設定しないで下さい。*

ここで重要なのは、インストールの以前のバージョンがターゲットマシンに存在しない限りこれらのプロパティはコマンドラインで設定しない方が良いということです。コマンドラインで設定した場合、インストールはマイナー アップグレード モードで実行しているかのように見えますが、アプリケーション ファイルがインストールされない可能性があります。エンドユーザーがコマンドラインを使用すべきかそうでないかを識別できるように配慮してください。

メモ: *マイナーアップグレードを実行する場合、*REINSTALLMODE *属性 v は非常に重要です。このパラメーターはインストーラーに対して、新しいインストール パッケージを使って製品の内部セットアップキャッシュを更新するよう指示します。このパラメーター無しでは、インストールは最新のインストール パッケージに施された変更に関する情報を全く持ちません。*

スモール アップデートを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

スモール アップデートの作成手順は、重要な例外をひとつ除いては、マイナー アップグレードの作成手順と同じです。例外とは、スモールアップ デートとは違って、マイナーアップグレードでは製品バージョンを変更するということです。

Windows Installer 3.0 より以前のバージョンを利用して製品に適用することができるアップグレードを作成する必要がある場合、スモールアップデート以外の方法を利用することが推奨されます。スモールアップデートは製品バージョンを変更しないので、製品の最新版に利用されるインストーラーをはじめとする外部プログラムは、スモールアップデートが適用された製品とそうではない製品の違いを認識できません。

スモールアップデートは、通常、完全インストールとしてではなくパッチとしてパッケージされます。作成するアップグレードが、Windows Installer 3.0 以降を利用して製品に適用されることが分かっている場合、パッチ シーケンス サポートを利用しスモールアップデート パッチを使用して、以前のバージョンの製品を更新することができます。

パッチ時の考慮事項

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

以下は、パッチと QuickPatch プロジェクトを作成するときのいくつかのガイドラインです。

パッチ作成前にアップグレードをテストする

パッチ作成前にフル インストールとしてアップグレードのテストを行うことが推奨されます。フルインストール パッケージがアプリケーションの以前のバージョンを確実にアップデートすると、パッチを作成することができます。

アップデート起動ツール (Update.exe) を作成する

InstallShield では、アップデートに Update.exe アップデート ランチャーを含めるかどうかを指定できます。Update.exe アップデート ランチャーは次のような場合に必要です。

- 必要に応じて、自動的にターゲット システムの Windows Installer エンジンを更新またはインストールする。
- InstallScript MSI プロジェクトのパッチをビルドしている。
- 必要に応じて、.NET Framework をターゲット システムに自動的にインストールする。
- 製品構成に製品のインスタンスを複数インストールできるサポートが含まれているパッチをビルドし、かつインスタンスの選択ダイアログを必要に応じて表示する。

  複数インスタンスのサポートに関する詳しい情報は、「製品の複数のインスタンスをインストールする」および「製品の複数インスタンスをインストールするためのサポート」をご覧ください。

Update.exe アップデート ランチャーは、上記のシナリオを制御するブートストラップ アプリケーションです。

InstallShield で Update.exe アップデートランチャーを作成しないように構成した場合、.msp ファイルが作成されます。

詳細については、次を参照してください。

- パッチに Update.exe アップデート ランチャをビルドをするかどうかを指定する
- QuickPatch パッケージに Update.exe アップデート起動ツールをビルドをするかどうかを指定する

アップデート ランチャー名のカスタマイズ

InstallShield では、現在のパッチの指定された名前を持つ (.exe) 用にアップデート ランチャーを作成するかどうかを指定できます。デフォルトで、InstallShield はアップデート ランチャーの名前として Update.exe を使用します。

大きなファイル用のパッチの最適化

[パッチのデザイン] ビューで**[大きなファイル用のパッチの最適化]** チェック ボックスを選択すると、ファイル サイズが 4MB 以上のアプリケーションに対してより小さいビットレベルのパッチが作成されます。

QuickPatch プロジェクトでは、ビルドされた QuickPatch はそれぞれ自動的に大きいファイル用に最適化されます。

圧縮されたインストールのパッチ

パッチの作成プロセスには、以前のインストールと最新のインストールの非圧縮リリースが必要です。インストールが圧縮されているインストールの場合は、リリースの管理用イメージを利用できます。

InstallShield は、[パッチのデザイン] ビューでパッチ構成に圧縮されたインストールを指定すると、自動的に管理用イメージを作成します。InstallShield はまた、QuickPatch プロジェクトで非圧縮のイメージを指定すると、自動的にユーザーのために管理イメージを作成します。

パッチと REINSTALL プロパティ

パッチのテスト中にオリジナルインストールのソースがある場所を問い合わせるプロンプトが表示される場合、REINSTALL プロパティで変更した機能に明示的に設定することで問題を回避できる場合があります。このプロパティはデフォルトで **"ALL"** に設定されます。

パッチまたは QuickPatch パッケージにおける COM サーバーの自己登録

次の条件のすべてが当てはまる場合、QuickPatch プロジェクトに含まれる自己登録とマークされている新しい COM サーバーには、ISSelfReg テーブルを使った InstallShield 自己登録が使用されます。

- ISSelfReg テーブルは、QuickPatch の作成時に使用された、アップグレード済みの .msi ファイルにあります。
- ISSelfRegisterFiles カスタム アクションが、[実行] シーケンスにある。
- InstallShield の [オプション] ダイアログにある [プリファレンス] タブで、ISSelfReg オプションが選択されている。

前述の条件のうち、1 つでも当てはまらない場合、InstallShield は Windows Installer 自己登録 (SelfReg テーブル) を使用します。

前述の条件の 1 つでも当てはまらない場合に、パッチで InstallShield 自己登録を使用するには、QuickPatch プロジェクトの代わりに [パッチのデザイン] ビューを使用します。

Windows Installer 3.0 (およびそれ以上) とパッチ

Windows Installer 3.0 以上には、多数のパッチ関連の強化点があります。パッチまたは QuickPatch が Windows Installer 3.0 以上を用いてターゲットマシンで適用される場合、これらの強化点を利用することができます。詳細は、次を参照してください:

- パッチ シーケンス
- パッチのアンインストール
- 非管理者パッチ

Windows Installer 3.0 以上は、シーケンスデータを含まないメジャーアップグレードパッチをサポートします。ただし、インストーラーは MsiPatchSequence テーブルに格納されるシーケンスデータを含むメジャーアップグレードパッチはサポートしません。

Windows Installer 1.2 とパッチ

Windows Installer 1.2 エンジンをターゲットとするパッチを作成する場合、[パッチのデザイン] ビューで適切なパッチ構成を選択します。そして [詳細] タブで "最小 Windows Installer バージョン" プロパティを **120** に設定します。

製品の複数インスタンスをパッチするためのサポート

同じコンテキスト (環境) および同じマシン上で製品のインスタンスを複数インストールできるサポートを含む基本の MSI プロジェクトにパッチを作成したとき、InstallShield では、エンドユーザーがそのパッチを 1 つまたはすべてのインスタンスに適用することができるように構成されます。詳細については、「製品の複数インスタンスをインストールするためのサポート」を参照してください。

アップデート起動ツールのファイルのプロパティをカスタマイズする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、Update.exe アップデート ランチャーのバージョン リソースにカスタム情報を使用できます。この情報は、アップデート ランチャーの [プロパティ] ダイアログ ボックスに表示されます。[プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックしたときに表示されます。

メモ: *[プロパティ] ダイアログ ボックスは、Windows のバージョンによって異なります。たとえば、Windows 7 システムでは、バージョン リソース情報は [プロパティ] ダイアログ ボックスの [詳細] タブに表示されます。一方、Windows XP システムでは、バージョン リソース情報は、同じダイアログボックスの [バージョン] タブに表示されます。*

また、Windows の一部のバージョンは、[プロパティ] ダイアログ ボックスの一部の設定を表示しません。

Update.exe *ファイルの [プロパティ] ダイアログ ボックスと同じ概観を持つ* Setup.exe *ファイルの [プロパティ] ダイアログ ボックスのスクリーンショットを見るには、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。*

タスク ***Update.exe 設定のデフォルトの InstallShield 値を独自のカスタム値でオーバーライドするには、以下の手順に従います:***

1. この手順は、作成中のパッケージが標準のパッチか QuickPatch パッケージかによって異なります。

   基本の MSI または InstallScript MSI プロジェクトで標準のパッチを作成している場合:

   a. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。

   b. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成を選択します。

   QuickPatch パッケージを作成している場合:

   a. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。

   b. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。

2. **[詳細]** タブをクリックしてから、**[アップデート ランチャーの設定]** 領域を探します。
3. 以下の設定に、Update.exe ファイルのプロパティに使用する値を入力します。
   - 会社名
   - 製品名
   - 製品バージョン
   - 説明
   - 著作権情報

ヒント: *"製品バージョン" 設定は、Update.exe のファイル バージョンと製品バージョンを更新します。ファイル バージョンは、常に 4 つのフィールドで構成されます。4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、製品バージョンとして* **1.1** *を指定すると、Update.exe のバージョン リソースで使用されるファイル バージョンは* **1.1.0.0** *となります。*

アップデート起動ツールの設定を空白のままに残すと、InstallShield はデフォルトの InstallShield 値を使用します。

アップデート起動ツールのアイコンを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、Update.exe アップデート ランチャーに使用するアイコンを指定できます。アイコンには、.exe、.dll、または .ico ファイルを使用できます。

エンド ユーザーが Windows Explorer で Update.exe ファイルを参照したときにも、このアイコンが表示されます。このアイコンは、Update.exe の [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。

アイコンを指定しなかった場合、InstallShield は Update.exe ファイルのデフォルト アイコンを使用します。

タスク ***Update.exe ファイルのアイコンを指定するには、以下の手順に従います:***

1. この手順は、作成中のパッケージが標準のパッチか QuickPatch パッケージかによって異なります。

   基本の MSI または InstallScript MSI プロジェクトで標準のパッチを作成している場合:

   a. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。

   b. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成を選択します。

   QuickPatch パッケージを作成している場合:

   a. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。

   b. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。

2. **[詳細]** タブをクリックしてから、**[アップデート ランチャーの設定]** 領域を探します。

3. **"アイコン"** 設定には、InstallShield がビルド時に Update.exe ファイルを作成するときに使用するアイコンを含むファイルの完全修飾名を指定します。

   ファイルを指定するには、明示的パスまたはパス変数を入力するか、省略記号ボタン (...) をクリックして **[アイコンの変更]** ダイアログ ボックスを開いて **[参照]** ボタンをクリックするとファイルを選択できます。

   デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、**[アイコンの変更]** ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

リリースを次回ビルドするとき、InstallShield が Update.exe ファイルに指定されたアイコンを使用します。

パッチ シーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

InstallShield では、ターゲット マシンにパッチが提供された順番に関係なく、Windows Installer バージョン 3.0 以降がインストール済みの製品にスモール アップデート パッチを適用する順番を指定することができます。パッチ シーケンス データを使って、Windows Installer が同じパッチ ファミリー内でパッケージされた各アップグレードの適切な関係を認識できるようにします。従って、製品にパッチ 2 を適用した後にパッチ 1 を適用すると、パッ

チ 2 のファイルを上書きせずにパッチ 1 が登録されます。Windows Installer バージョン 3.0 以前では、パッチ シーケンスは無視され、すべてのスモール アップデート パッチは、ターゲット マシンに提供された順番で製品に適用されます。

Windows Installer 3.0 以降で利用可能なパッチシーケンス機能は、パッチ作成処理を簡素化します。次のセクションでは、その方法を説明します。

Windows Installer 3.0 以前のバージョンを使って適用されるパッチを作成する

Windows Installer 3.0 より以前のバージョンを利用して製品に適用することができるパッチを作成する必要がある場合、スモールアップデート以外の方法を利用することが推奨されます。スモール アップデートは製品バージョンを変更することはありません。したがって製品の最新版に利用されるインストーラーを含み、外部プログラムは、スモール アップデートを伴う製品およびスモール アップデートを伴わない製品の違いを認識しません。Windows Installer 3.0 以前のバージョンに限ったシナリオでは、こういったインストーラーの制限事項について考慮する必要があるため、以前の製品状態について様々な可能性をターゲットにしなくてはなりません。次のテーブルに図式化されたサンプル アプリケーションのライフサイクルで、それによって生じる複雑さを説明します。

テーブル 5 • Windows Installer 3.0 以前のバージョンを使って適用されるパッチのサンプル アプリケーション ライフサイクル

アプリケーション パッケージ	製品バージョン	パッケージがターゲットにする以前のセットアップ
1.基本のインストール	1.0	—
2.マイナー アップグレード	1.1	1.0
3.マイナー アップグレード	1.2	1.0, 1.1
4.マイナー アップグレード	1.3	1.0、1.1、1.2、4
5.マイナー アップグレード	1.4	1.0、1.1、1.2、1.3
6.メジャー アップグレード	2.0	1.0, 1.1, 1.2, 1.3, 1.4

Windows Installer 3.0 を使って適用されるパッチを作成する

Windows Installer 3.0 以降で利用可能なパッチ シーケンス機能を使うと、スモール アップデートは製品バージョンを変更することはありませんが、製品の複数の異なるバージョン用に連続性を持たない 1 つのアップグレードを配布するスモール アップデートを安心して利用することができます。スモール アップデートとは違って、マイナー アップグレードは製品バージョンを変更します。マイナー アップグレードはまた、スモール アップデート パッチのシーケンス用フレームワークを形成します。製品のバージョン 1.1 用のスモール アップデートがバージョン 1.2 に適用された場合、インストーラーはスモール アップデートをターゲット システムに登録して、1.2 マイナー アップグレードが適用される前の段階と同じ要領でそれを適用します。

スモール アップデートパッチはまた、Windows Installer 3.0 以降を有効にし、製品に別のパッチが個別に適用または削除されても製品の有効な状態を保ちます。さらに、パッチ シーケンスを利用すると、ターゲット マシン上に存在する可能性のあるパッチの組み合わせすべてを考慮する必要なく、以前の比較的小さな製品状態の集まりからアップグレード パッケージを生成することができます。次のテーブルに図式化されたサンプル アプリケーション ライフサイクルは、この利点を説明します。

テーブル 6 • Windows Installer 3.0 を使って適用されるパッチのサンプル アプリケーション ライフサイクル

アプリケーション パッケージ	パッチ シーケンス番号	製品バージョン	パッケージがターゲットにする以前のセットアップ
1.基本のインストール	—	1.0	—
2.スモール アップデート	1	1.0	1.0
3.スモール アップデート	2	1.0	1.0
4.スモール アップデート	3	1.0	1.0
5.スモール アップデート	4	1.0	1.0
6.マイナー アップグレード	—	1.1	1.0

上のテーブル内のスモール アップデートのすべては、同じパッチ ファミリーに所属します。Windows Installer 3.0 以降はパッチ ファミリーを利用して、同じファミリー内でスモール アップデート パッチとその他すべてのパッチを比較し、各パッチをターゲット マシンに適用する順番を決定します。パッチ シーケンスは、パッチ パッケージ データベースの MsiPatchSequence テーブル に追加されます。このテーブルは、同じパッチ ファミリーをターゲットにするパッチの関係を定義します。

Windows Installer 3.1 を使って適用されるパッチを作成する

Windows Installer 3.1 で利用可能なパッチ シーケンス機能は、マイナー アップグレードに関してパッチ作成処理をさらに簡素化します。[パッチのデザイン] ビューの [詳細] タブにある **"RTM バージョンをターゲットにするマイナー アップグレード (MSI 3.1 が必要)"** プロパティを利用すると、マイナー アップグレード パッチで製品の RTM (Release to Manufacturing) バージョン(インストールされている場合、製品の最新メジャー アップグレード) をターゲットするかどうかを指定することができます。このプロパティには次のような 2 つのオプションがあります。

- 現在のマイナーアップグレード パッチを適用する前に、マイナー アップグレード パッチで製品の RTM バージョン (または、インストールされている場合、製品の最新メジャー アップグレード) までのすべてのパッチを実質的に削除する場合、このプロパティで [はい] を選択します。このオプションを選択すると、すべてのパッチが、シーケンシング データがあるないにかかわらず、削除されます。追加のベースライン バージョンをターゲットにする必要はないので、結果的にパッチのペイロードを増やすことになります。すべてのエンドユーザーは、中間のパッチを適用することなく、正常にこのパッチを適用することができます。

- このオプションを選択すると、場合により、RTM (または、存在する場合、製品の最新メジャー アップグレード) の後に作成された以前の各マイナー アップグレードをターゲットするのに必要な情報をパッチに含める必要があります。

  たとえば、SP 2 のマイナー アップグレード パッチを作成していて、このプロパティで [いいえ] を選択した場合、パッチは SP 1 のマイナー アップグレードをターゲットにする必要があります。またオプションで、他のベースライン (RTM など) をターゲットにすることもできます。これにより、ペイロードが増加します。RTM バージョンをターゲットにしない場合、RTM バージョンを持っているが SP 1 のマイナー アップパッチがないエンドユーザーは、SP 2 の前に SP 1 をインストールしなければなりません。

Windows Installer 4.5 を使って適用されるパッチを作成する

優先されるパッチが製品のコンポーネントをインストールするが、後で優先するパッチがそのコンポーネントを削除する場合、コンポーネントの機能の状態がアドバタイズに変更され、再インストールされない可能性があります。さらに、その機能に関連付けられた残りのコンポーネントは一切保持されません。これは、Windows Installer 4.0 以前を持つターゲット システム上ではトラブルの原因となります。Windows Installer 4.5 で利用できるパッチのシーケンス機能は、この問題を回避するための強化された堅牢な優先機能を提供します。

現在のパッチに含まれるコンポーネントをアンインストール用にフラグして、優先するパッチが適用された後にターゲット システム上でこのコンポーネントが孤立しないように指定できます。後続のパッチがインストールされ、またそれが最初のパッチよりも優先することがフラグされている場合、適切な場合に Windows Installer 4.5 はこのコンポーネントを登録解除およびアンインストールします。

Windows Installer 4.5 は、置き換えられて不必要になったコンポーネントをアンインストール用にフラグするための 2 つの方法をサポートします:

- 個別のコンポーネントの "置き換えられたコンポーネントのアンインストール" 設定に [はい] を選択して、適切な場合にこれをアンインストールすることを示します。[はい] を選択すると、InstallShield はコンポーネントの Component テーブルに msidbComponentAttributesUninstallOnSupersedence 属性を追加します。

  "置き換えられたコンポーネントのアンインストール" 設定にアクセスするには、[コンポーネント] ビューを開いてから、構成するコンポーネントを選択します。右側のグリッドに設定が表示されます。この設定のデフォルト値は [いいえ] です。

- 優先されたパッチに含まれる不必要なコンポーネントすべてをアンインストールすることを示すには、Windows Installer プロパティ MSIUNINSTALLSUPERSEDEDCOMPONENTS の値を 1 に設定します。[プロパティ マネージャー] ビューを使ってプロジェクトにこのプロパティを追加すると、プロパティの値をコマンドラインから設定できます。

Windows Installer 4.0 以前は、msidbComponentAttributesUninstallOnSupersedence 属性と MSIUNINSTALLSUPERSEDEDCOMPONENTS プロパティを無視します。

MsiPatchSequence Table (Windows Installer ヘルプ ライブラリ)

パッチのアンインストール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI – InstallScript ユーザー インターフェイス (UI) のスタイルが、埋め込み UI ハンドラーとして InstallScript エンジンを使用する新しいスタイルの場合*
- *QuickPatch*

この情報は、InstallScript UI に従来型のスタイル (InstallScript エンジンを外部 UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

Windows Installer 3.0 以降はスモールアップデートまたはマイナーアップグレードのパッチアンインストールを サポートします。アンインストールするパッチは Windows Installer 3.0 以降を利用してインストールされたものでなくてはなりません。パッチがアンインストールされた時、製品はパッチがインストールされる前の状態に戻ります。以前のバージョンの Windows Installer では、パッチの削除を希望するエンドユーザーはパッチされた製品をアンインストールしてから、パッチを適用せずに製品を再インストールする必要がありました。この場合、削除しないパッチについては再び適用する必要があります。

Windows XP SP2 以降を実行中のシステムでは エンドユーザーは [プログラムの追加と削除]を使ってパッチをアンインストールすることができます。以前のバージョンの Windows で Windows Installer 3.0 以降を実行中のシステムでは、パッチはコマンドラインからアンインストールします。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「Uninstalling Patches」を参照してください。

すべてのパッチがアンインストール可能ではないため、パッチのアンインストールはデフォルトでは無効に設定されています。パッチのアンインストールが不可能である、または特定の機能がパッチのアンインストール中に動作しない具体例は次のとおりです。

- パッチとしてパッケージされたメジャーアップグレードはアンインストールすることができません。
- パッチが特定の行を Windows Installer テーブルに追加する場合、パッチをアンインストールすることはできません。パッチがこれらのテーブルのいずれかに行を追加する場合、パッチ検証の 1 つ (Val0015) によって、パッチがアンインストール不可能であることが警告されます。該当するテーブルの一覧は、「Val0015」を参照してください。
- RemoveFile テーブルまたは RemoveRegistry テーブルに内容を追加するパッチをエンド ユーザーが 削除できない場合があります。オリジナルパッケージに含まれていないファイルまたはレジストリ エントリを削除するようにパッチが設計されている場合、そのパッチをアンインストールしてもファイルまたはレジストリ エントリは修復されません。
- [SQL スクリプト] ビューを通して追加された SQL スクリプト サポートを含むアンインストール可能なパッチを作成した場合、その SQL スクリプトは、パッチのアンインストール中に実行されません。

エンドユーザーへ配布する前にパッチのアンインストールを充分テストすることが推奨されます。アンインストール不可能なパッチについてさらに詳しい情報は、Windows Installer ヘルプ ライブラリの「Uninstallable Patches」を参照してください。

非管理者パッチ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者によるパッチは次のすべての条件が満たされた場合に利用することができます。

- Microsoft Windows XP 以降のクライアント プラットフォームの場合、ターゲットマシンが Windows Installer 3.0 以降を実行中。サーバー プラットフォームはサポートしていない。
- アプリケーションは CD-ROM または DVD などのリムーバブル メディアからインストールされている。
- アプリケーションはマシンごとにインストールされている。

メモ: *ALLUSERS プロパティがコマンドラインで上書きされた場合、非管理者パッチは失敗します。*

- ベース インストールには、後に続くすべてのパッチの署名に使用される証明書を含まれていなくてはなりません。
- ベース インストールには、MsiPatchCertificate テーブルを含まれていなくてはなりません。このテーブルは非管理者が後に続くパッチを適用した際、そのデジタル署名を照合するために使用される署名者証明書を提供します。このテーブルには必要に応じて複数の証明書を含むことが可能で、あとに続くパッチは最低 1 つの証明書と照合することが可能でなくてはなりません。詳細については、「非管理者パッチのインストールを準備する」を参照してください。
- 非管理者パッチには、MsiDigitalCertificate テーブルが含まれていなくてはなりません。このテーブルには、署名済みパッチの署名証明書が含まれています。詳細については、「パッチ パッケージに署名する」または「QuickPatch パッケージに署名する」を参照してください。

前述の基準のどれにも当てはまらない場合、エンドユーザーはロックダウンされた環境でデジタル署名済みパッチをインストールすることができません。

非管理者パッチの一般的なシナリオは、コンピューター ゲーム業界で見られます。コンピューター ゲーム ユーザーの一部は子供で、自分のユーザープロファイルおよび HKEY_CURRENT_USER の下にあるレジストリ キー内のフォルダー以外のシステム領域にアクセスが不可能な場合があります。子供がマシンにインストールする内容およびアクセスできる範囲を、彼等の両親が管理者アクセスを利用してコントロールする例が考えられます。両親がすべてのアプリケーションをインストールした場合で、インストール済みのソフトウェアに対するパッチが利用可能なときに、前述の基準がすべて満たされた場合、子供は両親が付き添わなくても自分で非管理者パッチをダウンロード並びにインストールすることができます。

メモ: *ベース インストールが期限切れのデジタル証明書で署名してあり、MsiDigitalCertificate および MsiPatchCertificate テーブルに以前の証明書と新しい証明書を含む中間パッチが適用されなかった場合、非管理者パッチは、ユーザー アカウント管理 (UAC) を表示します。*

User Account Control (UAC) Patching (Windows Installer ヘルプ ライブラリ)

パッチの作成と適用

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ヘルプのこのセクションでは、インストールのパッチの作成およびその適用方法について説明します。このセクションは、パッチのアンインストールを有効にする方法、パッチのシーケンスの仕方、およびパッチパッケージの署名方法についても網羅しています。

新しいパッチ構成アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***新しいパッチ構成アイテムを追加するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** を右クリックして、**[新しいパッチ構成の追加]** をクリックします。

InstallShieldは、新しいパッチ構成アイテムを追加します。

新しい最新セットアップ アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***最新のセットアップ アイテムを追加するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、適切なパッチ構成アイテムを右クリックして、**[最新のセットアップの新規追加]** をクリックします。

InstallShieldが、最新のセットアップを新規追加します。

新しい以前のセットアップ アイテムを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***既存のセットアップ アイテムを追加するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、適切なパッチ構成アイテムを右クリックして、**[既存セットアップの新規追加]** をクリックします。

InstallShieldが、以前のセットアップを新規追加します。

外部ファイルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***外部ファイルを追加するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、**追加の外部ファイル**を右クリックして、**[外部ファイルの新規追加]** をクリックします。

InstallShieldが、外部ファイルを新規追加します。右側にあるペインで、このファイルを構成することができます。

パッチの設定を構成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチの作成および構成には、[パッチのデザイン] ビューを利用するのが最善の方法です。[パッチのデザイン] ビューで作業中に、製品の以前のバージョン、および、より最新のバージョンのためのプロジェクトが既にビルドされてい必要があります。

[パッチのデザイン] ビューでデザインされた各パッチは、パッチ構成として開始します。パッチ構成は、少なくとも、最新セットアップ アイテム 1 つと 1 つまたは複数の以前のセットアップ アイテムからなります。パッチ構成段階では、次のプロパティタブでプロパティを構成することができます。

- 共通
- 識別
- デジタル署名
- Sequence

- アドバンスト UI

パッチ構成プロパティを設定した後、エクスプローラーで順に階層を展開し、最新および以前のセットアップ アイテムを構成します。

パッチの識別情報を指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 3.0 以降は、パッチがアンインストール不可能な場合も、適用された各パッチについての [プログラムの追加と削除] エントリをターゲット システムに追加します。

ヒント: *[パッチのデザイン] ビューでパッチ構成の最新セットアップを変更するたびに、その最新セットアップからの [プログラムの追加と削除] 情報が [識別] タブにある設定用の値として使用されます。必要に応じて、[識別] タブの値をオーバーライドすることもできます。*

タスク ***パッチの識別情報を指定するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. パッチ構成がまだ作成されていない場合、それを作成します。
3. **[パッチのデザイン]** エクスプローラーで、パッチ構成をクリックします。
4. **[識別]** タブをクリックします。
5. 個々の設定を構成します。

パッチのアンインストールを有効にする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI ― InstallScript ユーザー インターフェイス (UI) のスタイルが、埋め込み UI ハンドラーとして InstallScript エンジンを使用する新しいスタイルの場合*

この情報は、InstallScript UI に従来型のスタイル (InstallScript エンジンを外部 UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

Windows Installer 3.0 以降は、スモール アップデートおよびマイナー アップグレード用のパッチのアンインストールをサポートします。ただし、すべての パッチがアンインストール可能という訳ではありません。パッチのアンインストールについての詳細および、関連する制限事項については、「パッチのアンインストール」を参照してください。

タスク ***パッチのアンインストールを有効にするには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[共通]** タブをクリックします。
4. **[パッチのアンインストールを許可する (Windows Installer 3.0 が必要)]** チェック ボックスを選択します。

パッチのシーケンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチ シーケンスを定義する前に、[アップグレード] ビューで必要なアップグレード アイテムを作成する必要があります。その後、[パッチのデザイン] ビューでパッチの構成アイテムを追加します。これらのタスク完了後はじめて、パッチのシーケンスを指定することができます。

タスク ***パッチ シーケンスを定義するには、以下を手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[シーケンス]** タブをクリックします。
4. 以下のいずれかを実行します。
   - InstallShield が生成するデフォルトのパッチ シーケンスを使用する場合、[デフォルトのシーケンス エントリの使用] チェック ボックスを選択します。
   - パッチ シーケンスを使用しない場合、[デフォルトのパッチシーケンスを使用する] チェック ボックスをクリアします。
   - カスタム シーケンスを指定する場合、次の手順に従います:
     a. **"デフォルトのパッチ シーケンスの使用"** チェック ボックスをクリアします。
     b. **[追加]** をクリックします。[**パッチ シーケンス**] ダイアログ ボックスが開きます。
     c. 必要に応じてシーケンス情報を定義します。

パッチ パッケージに署名する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者によるパッチは厳しい条件が満たされたときのみ利用することができます。たとえば、パッチが更新するベース インストールは、パッチ パッケージの署名に使用される証明書を含まなくてはなりません。満たされなければならない他の基準については、「非管理者パッチ」を参照してください。

メモ: *パッチのファイル (アプリケーションの実行可能ファイルなど) にデジタル署名を行う場合、[リリース] ビューの [署名] タブで署名するファイルを指定することができます。*

タスク ***パッチパッケージを署名するには、以下を手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[デジタル署名]** タブをクリックします。
4. **[パッチ パッケージの署名]** チェック ボックスを選択します。
5. デジタル署名 設定を構成します。

パッチ パッケージをパスワードで保護する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

セキュリティをさらに向上させるため、パッチをパスワードで保護することができます。パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。

タスク ***パッチパッケージをパスワードで保護するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[詳細]** タブをクリックします。
4. **"ランチャーをパスワードで保護"** 設定で、**[はい]** を選択します。

5. **"ランチャ パスワード"** 設定に、パッチで使用するパスワードを指定します。

メモ: *パスワードは大文字と小文字を区別します。*

パッチに Update.exe アップデート ランチャをビルドをするかどうかを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

InstallShield では、InstallShield で、作成中のパッチ パッケージに Update.exe アップデート ランチャを作成するかどうかを指定できます。

必要に応じて自動的にターゲット システムで Windows Installer サービスを更新またはインストールする場合、Update.exe アップデート ランチャーが必要です。Update.exe ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。

InstallShield で Update.exe アップデートランチャーを作成しないように構成した場合、.msp ファイルが作成されます。

タスク ***パッチに Update.exe アップデート ランチャを含めるかどうかを指定するには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、構成するパッチ構成をクリックします。
3. **[共通]** タブをクリックします。
4. Update.exe を含める場合、**[Update.exe を作成する]** チェック ボックスを選択します。

   Update.exe を含めない場合、**[Update.exe を作成する]** チェック ボックスをクリアします。

ヒント: *[詳細] タブを利用しても、Update.exe を含めるかどうかを指定することができます。*

最新セットアップ アイテムを別のパッチ構成にコピーする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

複数のパッチ構成アイテムを [パッチのデザイン] ビューに追加した場合、ひとつのパッチ構成内の最新セットアップ アイテムを別のパッチ構成にコピーすることができます。

タスク ***ひとつのパッチ構成内の最新セットアップ アイテムを別のパッチ構成にコピーするには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、コピーする最新セットアップアイテムを、それを持っていないパッチ構成アイテムまでドラッグします。

InstallShield は、前回のセットアップ アイテムおよび追加の外部ファイルをすべて含む最新のセットアップ アイテムをパッチ構成に追加します。

パッチ構成アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***パッチ構成アイテムを削除するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、削除するパッチの構成を右クリックしてから、**[削除]** をクリックします。

最新セットアップ アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***最新のセットアップ アイテムを削除するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、削除する最新のセットアップ アイテムを右クリックしてから、**[削除]** をクリックします。

以前のセットアップ アイテムを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

タスク ***既存のセットアップ アイテムを削除するには、次の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、削除する既存のセットアップ アイテムを右クリックしてから、**[削除]** をクリックします。

パッチのビルド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成を設定し、[リリース] ビューでパッチの新しいリリースを作成すると、アップグレードのためのパッチをビルドする準備ができたことになります。

タスク ***パッチをビルドするには、以下の手順に従います:***

1. ビューリストで **[メディア]** の下にある **[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、パッチをビルドするパッチ構成アイテムを右クリックしてから、**[パッチのビルド]** をクリックします。

InstallShield はパッチ構成アイテムに指定した設定に従ってパッチをビルドします。パッチは、パッチ構成の [共通] タブで指定した場所でビルドされます。InstallShield は次のようなサブディレクトリを指定した出力場所へ追加します。

¥*PatchConfigName*¥Patch

InstallShield は、また、デフォルトでパッチをビルドするごとにアップグレードとパッチの検証を実行します。詳細については、「アップグレード、パッチ、および QuickPatch パッケージを検証する」を参照してください。

最新のセットアップ バージョンが圧縮でビルドされると、ビルドはエラーを返して途中終了します。

パッチの適用

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチパッケージ (.msp) ファイルには、インストールされた 1 つ以上の製品バージョンをアップグレードするのに必要なトランスフォームおよび指示が含まれています。

パッチパッケージを適用するには、Windows Installer v. 1.1 以降がシステムにインストールされている必要があります。またアップデートするパッケージは、管理イメージとして、またはローカルにインストールされている必要もあります。さらに別の要件として、既存のセットアップパッケージが、パッチパッケージと同じ権限を使って同じユーザー向けにインストールされている必要があります。たとえば製品がすべてのユーザー用としてインストールされている場合は、アップデートもすべてのユーザー用にインストールする必要があります。

パッチを適用する場合、Windows エクスプローラーで .msp ファイルをダブルクリックするか、またはファイルを右クリックして [開く] を選択するのが最も簡単です。マイナーアップグレードのパッチを適用すると、最初に開くダイアログは PatchWelcome ダイアログです。メジャー アップグレードのパッチを適用すると、インストールをスタンドアロンで実行したときに表示されるような完全ダイアログ シーケンスが表示されます。

MsiExec.exe /p オプションを使用して、コマンドラインからパッチを適用できます。次のステートメントを入力すると、X:¥Product Updates¥Build 36¥PatchForV1.msp にあるパッチパッケージが適用されます。

```
msiexec /p "X:¥Product Updates¥Build 36¥PatchForV1.msp"Update.exe
```

パッチ構成に [パッチのデザイン] ビューで [Update.exe の作成] チェック ボックスを選択すると、.msp ファイルが実行可能ファイルにラップされます。次のコマンド ライン指定すると、Update.exe によってパッチ パッケージが起動されます。

```
msiexec /p <path to .msp file> REINSTALL=ALL REINSTALLMODE=omus
```

サイレントモードでのパッチの適用

サイレント モードでパッチを適用する場合、2 種類の方法があります。/qn コマンド ライン パラメーターを使って MsiExec.exe を起動するか、/s を Update.exe へ渡します。

サイレントモードでパッチを適用する場合、考慮を要する重要な点があります。正しく動作させるには、パッチを適用する際に、Windows Installer のプロパティ REINSTALL が **ALL** に、REINSTALLMODE が **omus** に設定されている必要があります。Update.exe は常にこれらのプロパティをコマンドラインで設定するため、パッチパッケージが Update.exe とともに適用されていた場合は、ユーザーが特別な操作を行う必要はありません。

完全なユーザー インターフェイスを使ってパッチパッケージを適用すると、インストールのデフォルトのダイアログの 1 つ、PatchWelcome が表示されます。ここには、正しいオプションを使って REINSTALL および REINSTALLMODE を設定するためのコントロール イベントが含まれています。ただしこのダイアログは、エンド ユーザー インターフェイスが抑制されていると表示されないため、次のようにコマンドラインでプロパティを設定する必要があります。

```
msiexec /p <.msp ファイルへのパス> /qn REINSTALL=ALL REINSTALLMODE=omus
```

パッチは既存する .msi データベースのキャッシュを変更しないため、REINSTALLMODE に **v** 設定を含む必要があります。

QuickPatch プロジェクトを作成する

プロジェクト: *以前のバージョンが Windows Installer を使用してインストールされている場合のみ、QuickPatch を作成して製品をアップデートすることができます。初期のバージョンの製品のインストールは、次のプロジェクト タイプのいずれかに作成されている必要があります:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

QuickPatch プロジェクトは、規模の小さいシングル アップグレードをユーザーへ配布したいインストール作成者へお勧めします。QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ作成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

既存の .msi ファイルまたは既存の QuickPatch をパッチする QuickPatch プロジェクトを作成するには、新規 QuickPatch 作成ウィザードを利用します。

既存の QuickPatch をパッチする QuickPatch プロジェクトを作成する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

既存の QuickPatch プロジェクトをベースにして、新規 QuickPatch プロジェクトを作成することができます。これは、オリジナルアプリケーションまたは特定のパッチ済みバージョンに対してパッチをあてる必要があるエンドユーザすべてに配布することができるパッチを作成します。

注意 - *[一般情報] ビューの [履歴] 領域に表示されるリリースを開いて変更を加えた場合、最新プロジェクトは動作不可能となります。これは [履歴] にあるリリース間で共有されている中間データが不足または改定された可能性があるためです。つまり、既存する QuickPatch プロジェクトをパッチする QuickPatch プロジェクトを作成するたびに InstallShield は既存の QuickPatch プロジェクト (.ism ファイル) のモードを読み取り専用に変更します。*

タスク

既存の QuickPatch をパッチする QuickPatch プロジェクトを作成するには、以下の手順を実行します。

1. InstallShield で、パッチをあてる QuickPatch プロジェクト (.ism ファイル) を開きます。
2. **[一般情報]** ビューを開きます。
3. **[一般情報]** エクスプローラーで、**[履歴]** を選択します。
4. **[次のバージョンの作成]** をクリックします。

新規 QuickPatch プロジェクトが InstallShield で開きます。

新規 QuickPatch 作成ウィザードを使用して、既存の QuickPatch に QuickPatch を作成することもできます。

QuickPatch パッケージを簡素化するかどうかを指定する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトの構成を行うとき、InstallShield で QuickPatch パッケージの作成を簡素化して、最もシンプルなパッケージをビルドするかどうか指定することができます。QuickPatch の簡素化を行うと、通常の QuickPatch パッケージに比べて新しいサブ機能とカスタム アクションの数が少ない QuickPatch パッケージを生成できます。

たとえば、QuickPatch プロジェクトに新しいファイルとレジストリ エントリが含まれていて、 が QuickPatch の簡素化を行わなかった場合、そのファイルとレジストリ エントリ用に新しいサブ機能が作成されます。さらに、特定の Windows Installer パッチ要件に対応するため、1 つまたは複数のビルド済み カスタム アクションが追加されます。これに対し、InstallShield が QuickPatch の簡素化を行う場合、ファイルまたはレジストリ エントリは既存する機能に追加されるため、特別なビルド済み InstallShield カスタム アクションは必要ありません。

メモ: *次のシナリオにおいて、InstallShield が QuickPatch パッケージの作成処理を簡素化することはできません。*

- *QuickPatch パッケージがインストール済みのファイルを削除する。*
- *QuickPatch パッケージがレジストリ キーを削除する、またはその名前を変更する。*
- *QuickPatch パッケージが、簡素化されていない通常の QuickPatch イメージをターゲットとする。つまり、[一般情報] ビューの [履歴] 領域で、簡素化を行わなかった QuickPatch のチェック ボックスを選択した場合、QuickPatch の簡素化はできないということです。1 つまたは複数の簡素化されていない QuickPatch イメージをターゲットとする簡素化された QuickPatch のビルドを試みると、ビルド警告が表示され、簡素化は行われません。*

また、15 未満の簡素化されていない累積 QuickPatch パッケージをベース .msi パッケージまたはメジャー アップグレード パッケージに適用できる点にご注意ください。制限数を超えると、パッチを適用中に エラー 2701 が発生します。正確な制限は、パッケージの機能ツリーおよび簡素化されていない各 QuickPatch がそのツリーに追加するサブ機能の階層数によって決まります。Windows Installer の機能ツリー階層数の制限は 16 レベルです。

簡素化された QuickPatch パッケージには、新しいサブ機能が含まれないため、この制限がありません。

タスク ***QuickPatch パッチを簡素化するかどうかを指定するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[詳細]** タブをクリックします。
4. **"QuickPatch の簡素化"** 設定で、適切なオプションを選択します:
   - QuickPatch パッケージを簡素化する場合、**[はい]** を選択します。この値は、新しい QuickPatch プロジェクトのデフォルト値です。
   - QuickPatch パッケージを簡素化を選択しない場合、**[いいえ]** を選択します。

パッチのターゲット リリースを指定する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

タスク ***QuickPatch プロジェクトでどのリリースをパッチするのかを指定するには、次の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[履歴]** を選択します。

3. 中間リリース（ベース QuickPatch イメージと現在の QuickPatch イメージの中間となるリリース ）の隣にあるチェック ボックスを選択またはクリアして、現在のプロジェクトでパッチするのかどうかを指定します。

メモ: *ベース QuickPatch イメージまたは現在のプロジェクトのチェック ボックスをクリアすることはできません。中間 QuickPatch プロジェクトのチェック ボックスをクリアすると、現在の QuickPatch プロジェクトを使って、アプリケーションの最新イメージとして中間イメージを持つマシンをアップグレードすることはできません。*

たとえば、インストールとしてバージョン 1.0 を配布した場合で、その後オリジナルインストールをバージョン 1.1 にアップグレードするパッチをリリースした場合。バージョン 1.2 QuickPatch を作成して、履歴で 1.1 QuickPatch のチェック ボックスが選択されていない場合、 1.2 QuickPatch は 1.0 リリースのアップグレードに利用できますが、1.1 リリースには利用できません。

QuickPatch を実行するカスタム アクションを指定する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

タスク ***QuickPatch の適用時に実行するカスタム アクションを指定するには、以下の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで **[カスタム アクション]** をクリックします。
3. カスタム アクションの隣にあるチェック ボックスを選択またはクリアして、QuickPatch のインストール中にそれらを実行するかどうかを指定します。

QuickPatch へファイルを追加する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

タスク ***QuickPatch にファイルを追加するには、以下の手順を実行します。***

メモ: *QuickPatch プロジェクトは、追加する新しいファイルに新しいターゲットの場所を定義できるサポートを含みません。そのため、QuickPatch プロジェクトに新しいファイルを追加する場合、そのインストール場所は、元のインストールで定義されたフォルダーでなくてはなりません。*

1. **[パッチ設定の定義]** の下にあるビュー リストで、**[ファイル]** をクリックします。
2. **[パッチするファイル]** エクスプローラーを右クリックし、**[新規ファイルの挿入]** をクリックします。**[開く]** ダイアログ ボックスが開きます。

3. 追加するファイルをクリックして **[開く]** をクリックします。**[パッチするファイル]** エクスプローラーにファイルが追加されます。
4. 新しいファイルをクリックしてから、その設定を構成します。

QuickPatch の識別情報を指定する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

Windows Installer 3.0 以降は、QuickPatch がアンインストール不可能な場合も、適用された各 QuickPatch についての [プログラムの追加と削除] エントリをターゲット システムに追加します。

タスク ***QuickPatch の識別情報を指定するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[識別]** タブをクリックします。
4. 個々の設定を構成します。

QuickPatch のアンインストールを有効にする

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

Windows Installer 3.0 以降は、スモール アップデートおよびマイナー アップグレード用の QuickPatch パッケージのアンインストールをサポートします。ただし、すべての QuickPatch パッケージがアンインストール可能という訳ではありません。標準パッチおよび QuickPatch パッケージのアンインストールについての詳細および、関連する制限事項については、「パッチのアンインストール」を参照してください。

タスク ***QuickPatch パッケージのアンインストールを有効にするには、以下の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[共通]** タブをクリックします。
4. **[パッチのアンインストールを許可する (Windows Installer 3.0 が必要)]** チェック ボックスを選択します。

QuickPatch パッケージをシーケンスする

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

タスク ***QuickPatch パッケージのシーケンスを定義するには、以下の手順を実行します。***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[詳細]** タブをクリックします。
4. **"パッチ シーケンスのエントリ作成"** 設定で、適切なオプションを選択します:
   - InstallShield が生成するデフォルトのパッチ シーケンスを使用する場合、**[はい]** を選択します。
   - パッチ シーケンスを使用しない場合、**[いいえ]** を選択します。

QuickPatch パッケージに署名する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

Windows Installer 3.0 以上では、管理者以外によるインストールが可能なパッチを作成することができます。非管理者による QuickPatch パッケージは厳しい条件が満たされたときのみ利用することができます。たとえば、パッチが更新するベース インストールは、パッチ パッケージの署名に使用される証明書を含まなくてはなりません。満たされなければならない他の基準については、「非管理者パッチ」を参照してください。

メモ: *QuickPatch プロジェクトでは、パッチ パッケージと* Update.exe *ファイルにデジタル署名することができます。QuickPatch パッケージの各ファイル (アプリケーションの実行可能ファイルなど) にデジタル署名を行う場合、それらを手動で署名してから、プロジェクトに追加しなければなりません。Windows SDK に含まれている* SignTool.exe *を使って、手動でファイルに署名を行うことができます。*

タスク ***パッチパッケージを署名するには、以下を手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[デジタル署名]** タブをクリックします。
4. **[パッチ パッケージの署名]** チェック ボックスを選択します。
5. デジタル署名 設定を構成します。

QuickPatch パッケージをパスワードで保護する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

セキュリティをさらに向上させるため、QuickPatch パッケージをパスワードで保護することができます。QuickPatch パッチをパスワードで保護すると、QuickPatch パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。

タスク ***QuickPatch パッケージをパスワードで保護するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[詳細]** タブをクリックします。
4. **"ランチャーをパスワードで保護"** 設定で、**[はい]** を選択します。
5. **"ランチャーのパスワード"** 設定に、パッチで使用するパスワードを指定します。

メモ: *パスワードは大文字と小文字を区別します。*

QuickPatch パッケージに Update.exe アップデート起動ツールをビルドをするかどうかを指定する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

InstallShield では、InstallShield で、作成中の QuickPatch パッケージに Update.exe アップデート ランチャを作成するかどうかを指定できます。

必要に応じて自動的にターゲット システムで Windows Installer サービスを更新またはインストールする場合、Update.exe アップデート ランチャーが必要です。Update.exe ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。

InstallShield で Update.exe アップデートランチャーを作成しないように構成した場合、.msp ファイルが作成されます。

タスク ***QuickPatch に Update.exe アップデート起動ツールを含めるかどうかを指定するには、以下の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[共通]** タブをクリックします。
4. Update.exe を含める場合、**[Update.exe を作成する]** チェック ボックスを選択します。

   Update.exe を含めない場合、**[Update.exe を作成する]** チェック ボックスをクリアします。

ヒント: *[ビルドの設定] 領域にある [詳細] タブを利用しても、Update.exe を含めるかどうかを指定することができます。*

[パッチするファイル] エクスプローラーからファイルを削除する

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

タスク ***[パッチするファイル] エクスプローラーからファイルを削除するには、以下の手順に従います:***

1. **[パッチ設定の定義]** の下にあるビュー リストで、**[ファイル]** をクリックします。
2. 削除するファイルを右クリックして、**[削除]** をクリックします。

QuickPatch を使用したインストール済みファイルの変更と削除

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

タスク ***QuickPatch と共にインストールされたファイルを変更または削除するには、以下の手順に従います:***

1. **[パッチ設定の定義]** の下にあるビュー リストで、**[ファイル]** をクリックします。
2. **[パッチするファイル]** エクスプローラーを右クリックし、**[既存のファイルのパッチ]** をクリックします。**[ファイルの選択]** ダイアログ ボックスが開きます。
3. 変更または削除するファイルをクリックします。**[パッチするファイル]** エクスプローラーにファイルが追加されます。
4. **[パッチするファイル]** エクスプローラーに追加したばかりのファイルをクリックして、設定を構成します。

ヒント: *また、上記 3、4 の手順を踏まずに、直接ファイルまたはフォルダーを [**元のセットアップ ファイル**] から [**パッチするファイル**] エクスプローラーへドラッグ アンド ドロップすることもできます。*

QuickPatch を使用したレジストリデータの追加、変更および削除

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトでレジストリデータを追加、変更または削除するときの手順は、基本的に、オリジナルのインストールにそれを行うときと同じ要領です。唯一異なる点は、QuickPatch プロジェクトのレジストリデータを追加する前に、[レジストリ] ビュー上部にある [ビューフィルター] リスト内で既存の機能を選択する必要があるという点です。QuickPatch では、新規の機能やコンポーネントを追加することができないため、レジストリデータはすべて、既にオリジナルの製品に存在する機能に関連付ける必要があります。

ヒント: *QuickPatch プロジェクト用に変更を加えたレジストリの設定を変更するには、まずアイテムを右クリックしてから [取り消す] をクリックします。*

QuickPatch プロジェクトのファイル形式を指定する: XML またはバイナリ

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトのファイル形式はバイナリまたは XML のどちらかで指定できます。

タスク ***QuickPatch プロジェクトのファイル形式を指定するには、次の手順に従います:***

1. **[パッチの設定]** の下にあるビュー リストで、**[一般情報]** をクリックします。
2. [一般情報] エクスプローラー選択、**[ビルドの設定]** を選択します。
3. **[詳細]** タブを開きます。
4. **[全般]** の下にある **"プロジェクト ファイル形式"** を次のいずれかに設定します。
   - **バイナリ** – ism ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。これは、現在のプロジェクト タイプのデフォルト形式です。現在のプロジェクトにこのプロジェクト ファイル形式を選択すると、Windows Installer データベース エディターで .ism ファイルを更新できます。また、Windows Installer API またはそのオートメーション インターフェイスを使って、.ism ファイルを更新することもできます。
   - **XML** – .ism ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。XML ツールを使用して、プロジェクト ファイルを更新できます。

メモ: *どちらのプロジェクト ファイル形式でも、コマンドラインからリリースをビルドできます。*

アップグレード、パッチ、および QuickPatch パッケージを検証する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

検証は、既存の Windows Installer ベースのインストールをアップグレードしようとするときに生じる可能性がある一般的な問題を識別するのに便利です。

メモ: *アップグレードおよびパッチの検証は、非圧縮 のリリースに対してのみ行うことができます。*

アップグレードおよびパッチの検証エンジンは現在、特定の条件およびレポートの失敗を必要に応じてテストする目的で設計された異なる検証ツール からなります。アップグレードおよびパッチの検証は、アップグレード シナリオに関する問題の検出を容易にすることを目的に設計されており、初回インストール問題を検証するのにより適している Windows Installer 検証とは異なります。

検証ツール

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

次の表は、アップグレードおよびパッチの検証のための検証ツールの一覧です。

テーブル 7 ・ アップグレードとパッチの検証ツール

検証ツール番号	説明
Val0001	この検証ツールは、最新バージョンのインストールから削除されたファイルが、アップグレード適用時にターゲットマシンからアンインストールされることを確認ます。
Val0002	この検証ツールは、RemoveFile テーブルエントリのみがインストール中にのみ実行されるようスケジュールされているケースを検出します。
Val0003	この検証ツールは、アップデートが正常に開始するために設定が適切であることを確認します。
Val0004	この検証ツールは、アップグレード適用時に、最新のインストールで変更されたすべてのファイルが、ターゲット システムに正しくインストールされることを確認します。
Val0005	この検証ツールは、アップグレードで無視される機能がないように、機能の InstallLevel 値が適切に構成されていることを確認します。
Val0006	この検証ツールは、メジャーアップグレードが実行されるべきときに、マイナーアップグレードが実行されようとしていると、それを検出します。
Val0007	この検証ツールは、.msi パッケージの名前が変わっていないことを確認します。

テーブル 7 • アップグレードとパッチの検証ツール (続き.)

検証ツール番号	説明
Val0008	この検証ツールは、Upgrade テーブルのすべてのレコードの ActionProperty と Version 要素が適切に統合されていることを確認します。
Val0009	この検証ツールは、インストールから削除されたレジストリ エントリが、インストールがアップグレードとして実行されたときに適切に削除されることを確認します。
Val0011	この検証ツールは、スキーマに互換性がない場合、それを検出します。スキーマに互換性がない場合、パッチを作成することはできません。
Val0012	この検証ツールは、アップグレード実行時に新しい機能がインストールされるよう、それらが最新のインストールに適切に追加されているかどうかを確認します。
Val0013	この検証ツールは、エンド ユーザーがメジャー アップグレード アイテムの [削除 (Remove)] 属性の機能について通知されることを確認します。
Val0014	この検証ツールは、Upgrade テーブル内のあるエントリのため、メジャーアップグレードが適用されたときに既存製品の一部がターゲットマシン上に残ることを警告します。
Val0015	この検証ツールは、特定の状況下でアップグレードをパッチとしてパッケージする場合、エンド ユーザーがパッチをアンインストールできなくなることを警告します。この検証ツールは、スモールアップデートおよびマイナーアップグレードに適用されます。

Val0001

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ (エラー)

ファイル [1] (ターゲットパス : [2]) は、セットアップから削除されているようですが、RemoveFile テーブルに表示されていません。RemoveFile テーブルが作成されていないと、このファイルは、アップグレードが実行されたとき、ターゲットマシンから削除されません。

[1] は、削除されたファイルの名前で、[2] は、このファイルのターゲット パスです。

説明

スモール アップデートまたはマイナー アップグレードが適用されるとき、最新のセットアップから削除されたすべてのファイルは、マイナーアップグレードがローカル .msi データベースを再キャッシュするため、ターゲットマシンから自動的にアンインストールされません。この再キャッシュされたコピーは、ファイルが既にセットアップから削除されているので、ファイルへのリファレンスを含んでいません。

アップグレードが実行されたときに、このファイルを削除するには、RemoveFile テーブルにエントリを追加する必要があります。アップグレードの実行時、これらの RemoveFile テーブルのエントリが実行され、ファイルは削除されます。

メモ: *この検証ツールは、メジャーアップグレードには適用しません。メージャーアップグレードの "アンインストールしてから再インストールする" という性質がこの種類の配慮に適合しないからです。*

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

以前のバージョンのセットアップパッケージからのファイル名、ファイルのインストール先を使用して、ダイレクト エディターを使用して RemoveFile テーブルでエントリを作成します。ファイル名には、ワイルドカード文字を使うことができます。

メモ: *RemoveFile テーブルを作成するとき、コンポーネントの指定が必要です。このコンポーネントのインストール状況は、この RemoveFile シグネチャと一致するファイルを削除すべきかどうかを判断するのに使用されます。アップグレード適用時に必ず再インストールするコンポーネントを指定してください。*

Val0002

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ (メモ)

RemoveFile テーブル エントリ [1] の InstallMode が 1 になっています。アップグレード シナリオで、コンポーネント [2] が再インストールを必要としない場合、このテーブル エントリに属するファイルは削除されません。InstallMode の値を 3 に変えて、ファイルが少なくともアンインストール時に削除されるようにするか、この RemoveFile テーブル エントリが再インストールされるコンポーネントに関連付けられていることを確認します。

[1] は、このメッセージの表示の原因になった RemoveFile テーブルの最初の列にあるエントリ名です。[2] は、RemoveFile テーブル エントリが関連付けられているコンポーネントの名前です。

説明

RemoveFile テーブルでエントリを作成するとき、InstallMode 列で値を指定する必要があります。この値が 1 に設定されていると、インストーラーは、指定されたファイルを関連付けられたコンポーネントがインストールされたときのみ削除するということが分かります。アップグレード シナリオでは、関連付けられたコンポーネントの再インストールが必要ではない場合、RemoveFile テーブル エントリは実行されず、ファイルはターゲットマシンから削除されません。

この後のアンインストールでは、ファイルはターゲットマシンにまだ存在しているので、このファイルとそれを含むディレクトリをマニュアルで削除する必要があります。

メモ: *この検証ツールは、メジャーアップグレードには適用しません。メージャーアップグレードの "アンインストールしてから再インストールする" という性質がこの種類の配慮に適合しないからです。*

この検証テストを行うために、検証エンジンは最新のセットアップを前回のバージョンと必ず比較します。

修正アクション

これは検証ツールは通知のみで、アップグレードを作成するときに、潜在的に存在するセットアップに関する問題を警告するものです。アップグレードの実行を予定しない場合や、この RemoveFile エントリに関連付けられているコンポーネントがアップグレードで再インストールされないことが分かっている場合、この通知を無視することができます。

InstallMode の値を 3 に変えることもできます。3 に変えると、ファイルはインストール時には削除されませんが、アンインストール時は必ず削除されます。

Val0003

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(メモ)

このセットアップは、リファレンスされた以前のセットアップの 'スモール' アップグレードを実行します。

メッセージ 2(メモ)

このセットアップは、リファレンスされた以前のセットアップの 'マイナー' アップグレードを実行します。

メッセージ 3(メモ)

このセットアップは、リファレンスされた以前のセットアップの 'メジャー' アップグレードを実行します。

メッセージ 4(エラー)

最新セットアップのパッケージコードが、以前のバージョンのパッケージコードと違っていません。アップグレードを実行するには、パッケージコードの変更が必要です。

メッセージ 5(メモ)

製品バージョン [1] は、製品バージョン [2] と異なっていますが、3 番目の要素以降のみです。Windows Installer は、3 番目の要素以降の製品バージョンの違いを検出しません。

[1] は、以前のインストールのバージョン番号です。[2] は、最新のインストールのバージョン番号です。

メッセージ 6(エラー)

セットアップは、指定された以前のインストールの 'メジャーアップグレード' を実行する必要があります。ただし、以前のセットアップ [1] のアップグレードがアップグレード テーブルに存在しません。アップグレードは、実行されません。

[1] は、以前のインストールのアップグレード コードです。

メッセージ 7(エラー)

セットアップは、指定された以前のインストールの 'メジャーアップグレード' を実行する必要があります。アップグレード テーブル エントリはアップグレード コードに存在していますが、製品バージョン [1] または 製品の言語 [2] にマッチしているアップグレード テーブル エントリがありません。

[1] は、バージョン番号で、[2] は、インストールの言語 ID です。

メッセージ 8(エラー)

以前のパッケージには UpgradeCode プロパティが含まれていません。メジャー アップグレードは、UpgradeCode を持たないすべての以前のバージョンのアンインストールに失敗します。

説明

この検証ツールは、以前のインストールをアップデートするのに必要なアップグレードの種類を判断します。たとえば、マイナー アップグレードの作成中に、インストールがメジャーアップグレードを実行するように構成されていることが検証ツールによって通知された場合、マイナーアップグレードをメジャーアップグレードに変更することも考えられます。

この検証ツールは、パッケージコードが変わったかどうかもチェックします。このチェックはすべてのアップグレード タイプに対して実行されます。パッケージコードの変更は、インストールのリリースを決定した場合、それを変更する意図があるないに関わらず、必ず必要です。パッケージコードが変わっていない場合、メッセージ 4 で説明されているエラーが発生します。

製品バージョンの変更が 3 番目より後のバージョン項目のみのとき、検証ツールは、メッセージ 5 で説明された注意を表示します。この注意は、バージョン リソースの 4 番目以降の変更は、実行時にインストールの一意のバージョンとしては表示されないことを警告するためのものです。

さらに、メジャーアップグレードが必要な場合、この検証ツールは、ターゲットマシンから前回のインストールを削除するために適切なエントリが最新インストールに存在するかどうかをチェックします。これらのエントリが存在しない場合、メッセージ 6 または 7 が表示されます。メッセージ 6 は、以前の製品に関連する設定がないときに表示されます。メッセージ 7 は、以前の製品に関連する設定はあるけれども、その設定の署名が前回のセットアップの署名をパッチしないときに表示されます。以前の製品に関連する設定があるが、以前のインストールにアップグレード コードが見つからない場合に、エラー 8 が表示されます。

修正アクション

メッセージ 4 を受け取った場合、最新バージョンのインストールに新しいパッケージコードを生成する必要があります。お勧めするオプションは、"製品構成のパッケージコードの生成" プロパティを [はい] に設定することです。

メッセージ 5 を受け取った場合、このメッセージは情報提供のみが目的なので無視することができます。但し、後日、メジャー アップグレードの適用を試みる場合、バージョンの違いがはっきり分からないと問題が生じることもあるのでご注意ください。

メッセージ 6 を受け取った場合、メジャー アップグレード アイテムの追加が必要です。

メッセージ 7 が表示されるとき、複数の理由が考えられます。1 つ目の可能性は、製品バージョンが、潜在的なターゲット インストールとして指定されたバージョン範囲から外れた場合です。以前のインストールのバージョンが、アップグレードする製品バージョンの受け入れ可能範囲の外側にある場合、メジャー アップグレード アイテムの最小/最大インクルーシブ設定をチェックしてみることも考えられます。また、製品の言語が、アップグレードにサポートされている言語のリストにない可能性もあります。以前のセットアップにメジャーアップグレード アイテムを定義していながら、そのアイテムを **"検出のみ"** とマークした場合も、エラーメッセージ 7 が表示されます。

エラー 8 が発生した場合は、アップグレード コードを持たない以前のパッケージを削除します。

Val0004

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ (エラー)

コンポーネント [2] のキーファイル [1] が、以前のパッケージでバージョン指定されていて、アップグレード済みパッケージでは低いバージョンが指定されているか、バージョンが指定されていません (古いファイル バージョン [3]、新しいファイル バージョン [4])。これによって、アップグレード シナリオでコンポーネントが再インストールされることを防ぎます。

[1] はファイル キーの名前、[2] はキー ファイルを含むコンポーネントの名前、[3] は以前のパッケージに含まれるファイルのバージョン番号、[4] はアップグレードに含まれるファイルのバージョン番号です。

説明

この検証ツールは、アップグレード適用時に、アップグレードで変更されたすべてのファイルが、ターゲット システムで正しく更新されることを確認します。

マイナー アップグレードまたはスモール アップデートの実行時に、コンポーネントのキー ファイルが処理中にダウングレードされる場合、Windows Installer はそのコンポーネントの再インストールを行いません。このため、アップグレードのキー ファイルは、古いパッケージのキー ファイルのバージョン番号と等しいか、それよりも大きい番号でなくてはなりません。

例外は、エンド ユーザーが REINSTALLMODE プロパティに **a** を設定した場合です。この設定は、バージョンに関係なくすべてのファイルを強制的にインストールします。

メモ: *この検証ツールは、メジャーアップグレードには適用しません。メジャーアップグレードの "アンインストールしてから再インストールする" という性質がこの種類の配慮に適合しないからです。*

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

このエラーを解決するためには、アップグレードのキー ファイルのバージョン番号が古いパッケージのキー ファイルのバージョン番号と等しいか、それよりも大きい番号であることを確認してください。代わりに、メジャー アップグレードを作成することもできます。

Val0005

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(警告)

機能 [1] のデフォルトの InstallLevel が ゼロになっています。この機能がインストールで有効にされる場合、似ているロジックを作成してメンテナンス モードでもその機能が有効にされるようにします。そうしない場合、アップグレードで機能は無視されます。

[1] は、InstallLevel の確認が必要と思われる機能の名前です。

メッセージ 2(警告)

機能 [1] の条件は、実行時に、この機能の InstallLevel をゼロに設定する可能性があります。この機能がインストールで有効にされる場合、似ているロジックを作成してメンテナンス モードでもその機能が有効にされるようにします。そうしない場合、アップグレードで機能は無視されます。

[1] は、InstallLevel の確認が必要と思われる機能の名前です。

説明

機能 InstallLevel をゼロに設定されると、実行時に機能が無効として表示され、ユーザーはそれを有効化することができません。デフォルトでこの機能をゼロに設定して、機能条件を使用して必要に応じて有効化できればと考える方はたくさんいます。

アップグレードで、この機能条件が、インストール レベルにゼロ以外の値を設定することができるように、評価されないようになっている場合、この機能の InstallLevelはゼロのままになります。したがって、機能でコンポーネントに加えられた変更はインストールされません。

また別の問題として、機能 InstallLevel は、デフォルトでゼロ以外の値に設定されますが、実行時に、機能 InstallLevel をゼロに設定する機能条件があります。これは、以前のインストールが既にフィールドに存在するので、メジャーアップグレードでも問題を提起します。

Val0005 は、すべてのアップグレードの種類 (メジャー、マイナー、スモール) に適用します。

この検証テストを行うために、検証エンジンは最新バージョン セットアップの検証のみ必要です。

修正アクション

これらの機能条件の評価が実行される可能性がある処理を行う場合、同じ処理がアップグレード実行時にも行われるようにしてください。アップグレードでは、エンドユーザーに表示されるユーザー インターフェイスが、インストールが初回で実行されるときに表示されるユーザー インターフェイスと異なる場合があります。

機能 InstallLevel の値 がデフォルトでゼロになっていないことは、メジャーアップグレードが支障なくファイルをインストールおよびアンインストールできるために大変重要です。

Val0006

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(エラー)

コンポーネント [1] (ComponentID [2] によって識別) が、最新バージョンのセットアップにありません。コンポーネントを削除してしまうと、スモール/マイナーアップグレードの実行はできません。メジャーアップグレードの実行が必要になります。

[1] は、削除されたコンポーネントの名前です。[2] は、削除されたコンポーネントに関連付けられている ComponentID です。

メッセージ 2(エラー)

機能 [1] が、最新バージョンのセットアップにありません。機能を削除してしまうと、スモール/マイナーアップグレードの実行はできません。メジャーアップグレードの実行が必要になります。

[1] は、削除された機能の名前です。

説明

コンポーネントまたは機能がインストールの最新バージョンから削除されている場合、マシンに残っているリソースおよびレジストレーションに関する潜在的な問題を避けるために、メジャーアップグレードを実行する必要があります。

メモ: *インストールで機能またはコンポーネントを他の場所に移動する場合、実質的に、オリジナルの場所からその機能またはコンポーネントを削除したことになります。*

この検証ツールは、メジャーアップグレードには適用されません。マイナーアップグレードまたはスモールアップデートではなく、メジャーアップグレードを実行すべき時を識別します。

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

インストールのアーキテクチャをメジャー変更する場合、メジャーアップグレードを適用します。

RemoveFile、RemoveRegistry および RemoveIniFile テーブルを使って、親を失ったリソースをクリーンアップすることができます。但し、コンポーネントおよび機能の Windows Installer レジストレーション情報はクリーンアップされません。レジストレーション情報をそのまま残しておくと、将来のアップグレード インストールまたは製品のアンインストールで予期しない結果を生じることがあります。

Val0007

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(エラー)

最新のセットアップ [1] の MSI パッケージ名が、前回のセットアップ [2] の MSI パッケージ名と異なります。スモール/マイナーアップグレードでは、パッケージ名の変更はできません。

[1] は、変更された新規の .msi パッケージの名前です。[2] は、元の .msi パッケージの名前です。

説明

マイナーアップグレードまたはスモールアップデートを実行するとき、インストールの前回と最新のバージョンはどちらも同じ .msi パッケージ名でなければなりません。.msi ファイル名が異なる場合にマイナー アップグレードまたはスモール アップデートを実行しようとすると、Windows Installer ランタイム エラー 1316 の原因となる可能性があります。

この検証ツールは、メジャーアップグレードには適用されません。

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

これは、Windows Installer サービスにのみ適用します。ここで唯一できることは、作成中の .msi パッケージ に同じ名前を付けることです。.msi パッケージの名前を構成するには、[リリース] ビューにある製品構成の [全般] タブで ″MSI パッケージ ファイル名″ 設定を使用します。

Val0008

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(警告)

Upgrade テーブルに、アクションプロパティ [1] が SecureCustomProperties のメンバーとしてリストされていません。定義されたアップグレードが適切に機能しない可能性があります。

[1] は、Upgrade テーブルで指定されたアクション プロパティの 1 つの名前です。

メッセージ 2(エラー)

Upgrade テーブルで、アクション プロパティ [1] がパブリックではありません。定義されたアップグレードは適切に機能しません。

[1] は、Upgrade テーブルで指定されたアクション プロパティの 1 つの名前です。

メッセージ 3(エラー)

Upgrade テーブルで、アクション プロパティ [1] が Property テーブルで既に定義されています。定義されたアップグレードは適切に機能しません。

[1] は、Upgrade テーブルで指定されたアクション プロパティの 1 つの名前です。

メッセージ 4(エラー)

Upgrade テーブルで、アクションプロパティ [1] が 複数回使用されています。実行時に予期せぬ結果が生じます。

[1] は、Upgrade テーブルで指定されたアクション プロパティの 1 つの名前です。

メッセージ 5(エラー)

Upgrade テーブルで、MaxVersion [1] が MinVersion [2] より小さいレコードが存在します。実行時に予期せぬ結果が生じます。

[1] と[2] は、Upgrade テーブルで指定されたバージョン番号です。

説明

Upgrade テーブルのすべてのレコードに適用

- ActionProperty がパブリックであることを確認してください。
- ActionProperty が一意であることを確認してください。
- ActionProperty が Property テーブルで定義済みではないことを確認してください。
- ActionProperty が SecureCustomProperties リストに存在することを確認してください。
- MinVersion が MaxVersion より小さいかまたは等しいことを確認してください。

この検証ツールは、メジャーアップグレードにのみ適用されます。

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

Action プロパティの値がどこに入力されたかを見つけるには、[アップグレード] ビューを開きます。メジャーアップグレードアイテムを選択してから、[詳細] タブをクリックします。Action プロパティに、Detect Property 値が使用されます。

メモ: 次の事項に注意してください。

- *プロパテがすべて大文字のとき、それはパブリックプロパティです。*
- *プロパティは、他のメジャーアップグレード アイテムと共有されていない限り一意です。*
- *プロパティが Property テーブルで定義されている場合、プロパティ マネージャーに移動し、それを削除します。*

- *プロパティが SecureCustomProperties リストにない場合、プロパティ マネージャーでこれを SecureCustomProperties プロパティに追加できます。複数プロパティはセミコロンで区切ります。*
- *最小バージョンが最大バージョンより大きいメジャーアップグレード アイテムを [アップグレード] ビューで見つけ、修正します。*

Val0009

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(警告)

レジストリ エントリが、コンポーネント [1] から削除されました。RemoveRegistry テーブルに対応するエントリが存在しますが、そのエントリがこのコンポーネントに関連づけられていません。これにより、アップグレードが実行されたときに、レジストリ エントリが削除されない可能性があります。

[1] は、セットアップの最新バージョンでレジストリのエントリが削除されたコンポーネントの名前です。

メッセージ 2(エラー)

レジストリ エントリが、コンポーネント [1] から削除されました。このキーを RemoveRegistry テーブルに追加する必要があります。追加をしない場合、アップグレードにより親を失くすことになります。[2]

[1] は、セットアップの最新バージョンでレジストリのエントリが削除されたコンポーネントの名前です。[2] は、存在しないレジストリ エントリで、<Root>|<Key>|<Value> 形式です。

説明

スモール アップデートまたはマイナーアップグレードが実行されるとき、最新のインストールから削除されたレジストリ エントリは、ターゲットマシンから自動的にはアンインストールされません。この理由は、マイナーアップグレードはローカル MSI データベースを再キャッシュするからです。これらのレジストリ エントリへのリファレンスは、既にインストールから削除されているので、この再キャッシュされたコピーには含まれていません。

このレジストリ エントリを削除するには、アップグレードの適用時に削除されるように、RemoveRegsitry テーブルにエントリを追加する必要があります。アップグレードの実行時、これらの RemoveRegistry テーブルのエントリが実行され、レジストリ エントリは削除されます。

メモ: *この検証ツールは、メジャーアップグレードには適用しません。メージャーアップグレードの "アンインストールしてから再インストールする" という性質がこの種類の配慮に適合しないからです。*

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

以前のバージョンのインストール パッケージからのレジストリ エントリのルート、キーおよび値を使用して、ダイレクト エディターを使用して RemoveRegistry テーブルでエントリを作成します。Value フィールドは、マイナス記号 (-) を使って作成できます。これにより、レジストリキー全体がそのすべての値とサブキーと共に削除されます。

メモ: *RemoveRegistry テーブルにエントリを追加すると、コンポーネントの指定が必要です。コンポーネントのインストール状態は、この RemoveRegistry シグネチャと一致するレジストリ エントリを削除すべきかどうかを判断するのに使用されます。アップグレード適用時に必ず再インストールされるコンポーネントを指定してください。*

Val0011

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(エラー)

TypeLib テーブルの Version 列のデータ型が、アップグレードされたイメージと異なります。パッチパッケージの作成時に、失敗の原因になります。

メッセージ 2(警告)

ターゲット イメージは、Windows Installer 1.2 またはそれ以前で作成され、アップグレードされた (Upgraded) イメージは Windows Installer 2.0 以降で作成されました。検証はデータ型の競合は検出しませんでしたが、このスキーマの変曲点を分割するパッケージにパッチを作成すると問題が発生します。

説明

Windows Installer 2.0 のリリースで、.msi インストール パッケージのスキーマは変更になりました。TypeLib テーブルの Version 列のデータ型は、I2 から I4 に変わりました。古いスキーマを使用した .msi インストールを新しいスキーマを使用した .msi パッケージでパッチすることはできません。

メモ: *この検証ツールは、メジャーアップグレードには適用しません。メージャーアップグレードの "アンインストールしてから再インストールする" という性質がこの種類の配慮に適合しないからです。*

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

このままでアップグレードをマイナーアップグレードとして配布することは可能ですが、パッチとして配布することはできません。

Val0012

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(エラー)

新規機能 [1] が、ルート階層の機能として定義されています。新規の機能は、オリジナル セットアップに既に存在する機能のサブ機能として定義されなければなりません。

[1] は、検証メッセージの原因になっている機能の名前です。

メッセージ 2(エラー)

検証エンジンは新しい機能 [1] を検出しました。この機能には、アップグレード シナリオでインストールされるために 'FavorParent' と 'Required' 属性に [はい] が設定されている必要があります。

[1] は、検証メッセージの原因になっている機能の名前です。

説明

インストールの最新バージョンをスモール アップデートまたはマイナーアップグレードとして配布予定の場合、インストールに機能を追加するとき、守らなければならない一定のルールがあります。規則 1 は、機能は既存の機能の子として追加されなければなりません。規則 2 は、機能の "リモート インストール" プロパティには FavorParent、"必須" プロパティには [はい] が設定されている必要があります。

この 2 つの規則が守られないと、新しい機能のコンテンツは、アップグレードが適用されたとき、インストールされません。代わりに、エンドユーザーはインストールをメンテナンス モードで実行し、機能をマニュアルで選択してローカルにインストールしなければなりません。

この検証ツールは、メジャーアップグレードには適用されません。

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

新規機能を既存の機能の子として追加し、その [リモート インストール] プロパティに [親を優先 (Favor Parent)] に、および "必須" プロパティを [はい] に設定します。

Val0013

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(警告)

検証ツールは、エントリ [1] に Upgrade テーブルの Remove 列が使われていることを検知しました。これは、リファレンスされたセットアップが不完全にアンインストールされる原因になります。最も一般的なアップグレードのシナリオでは、アップグレード テーブルの Remove 列は作成できません。

[1] は、Upgrade テーブルで参照されている Upgrade コードです。

説明

メジャーアップグレード アイテムの [削除属性 (Remove Attribute)] が空白のままにしておくと、ターゲット製品からすべての機能が削除され、実施的にすべての製品がアンインストールされます。

メジャー アップグレード アイテムの削除属性に機能の名前を 1 つ配置すると、その機能のみが削除され、既存の製品の他のすべての機能はターゲット マシンにターゲットの製品自身と共に残されます。つまり、ターゲットマシンに製品が 2 つと [プログラムの追加と削除]に 2 つのエントリが結果的に残ります。

この機能は、機能の一部の削除だけを行い、エンドユーザーのマシンにターゲット製品を残しておきたいユーザー用です。

この検証ツールは、メジャーアップグレードにのみ適用されます。

この検証テストを行うために、検証エンジンは最新インストールの検証のみ必要です。

修正アクション

これは警告です。上記で説明された動作が目的に適ったものである場合、この警告は無視することができます。そうでない場合は、メジャー アップグレード アイテムの Remove 属性は使用しないでください。

Val0014

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(エラー)

[1] という ID を持つコンポーネントは、最新バージョンのセットアップから削除されています。メージャーアップグレードでは、これらのコンポーネントに含まれているリソースはアンインストールされるので、アプリケーションの実行に必要なリソースが削除される可能性があります。

[1] は、以前のインストールのコンポーネントを識別するコンポーネント コードです。

説明

この検証ツールは、メジャー アップグレードが新規ファイルをインストールしてから、要らないファイルを削除するように構成されている場合のみ、このチェックを行います。メジャー アップグレードが製品をアンインストールしてから再インストールを行うように構成されている場合、この検証ツールが防ごうとしている問題は存在しません。

ターゲット アップグレード モードが選択されると、Installer はターゲットマシンの参照カウント コンポーネントを頼ってこの機能を果たします。必須コンポーネントの参照カウントは最新のアプリケーションがインストールされたときにインクリメントされ、以前のアプリケーションが削除されたときデクリメントされます。但し、参照カウントはゼロに達しないので、永続的なコンポーネントはそのままターゲットマシンに残ります。

したがって、インストールからコンポーネントを削除したり、そのコンポーネントからのリソースを異なるコンポーネントに追加すると、問題が生じます。代わりに、次を実行します。

1. 新しいコンポーネントを新しいリソースと共にインストールする。
2. 古いコンポーネントを古いリソースと共にアンインストールする。

ステップ 2 で古いコンポーネントがアンインストールされると、新しいコンポーネントによってステップ 1 で追加されたレジストリ エントリ、.ini ファイルの変更、環境変数などを削除する場合があります。

コンポーネントは GUID によって参照されるので、新しいコンポーネント GUID はインストール プロジェクトに新しいコンポーネントを追加するたびに生成されます。既に存在するコンポーネントを削除または追加するだけでも、同じことが起こります。

この検証ツールは、メジャーアップグレードにのみ適用されます。

この検証テストを行うために、検証エンジンは最新のインストールを前回のバージョンと必ず比較します。

修正アクション

この問題を修正するには、アップグレードを最初にアンインストールしてから再インストールするように構成するか、このエラーメッセージで指定された GUID を持つコンポーネントを追加します。コンポーネントが空でも、その存在は Windows Installer が正確なリファレンス カウントを記録するのに役立ち、コンポーネントが早まってアンインストールされるのを防ぎます。

Val0015

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

メッセージ 1(警告)

[1] テーブルに新しいコンテンツが含まれています。したがって、このアップグレードをパッチとしてパッケージすると、パッチのアンインストールが不可能になります。

[1] は、このメッセージの原因となったいるテーブルの名前です。

説明

エンドユーザーがパッチをアンインストールすると、製品はパッチがインストールされる前の状態に戻ります。パッチがアンインストール不可能な場合にパッチの削除を希望するエンドユーザーは、パッチが適用された製品をアンインストールしてから、パッチ無しで製品を再インストールする必要があります。

メモ: ターゲットマシンが特定の要件を満たさない場合 (たとえば、Windows Installer 3.0 以上 が必須である等)、検証警告が解決されていてもパッチはアンインストールすることができません。アンインストール可能なパッチの要件について詳しい情報は、「パッチのアンインストール」を参照して下さい。

パッチが次の特定のテーブルに行を追加する場合、たとえ**このパッチのアンインストールを許可する**チェックボックスが選択されていても、パッチをアンインストールすることはできません。

- BindImage
- Class
- Complus
- CreateFolder
- DuplicateFile
- Environment
- 拡張子
- Font
- IniFile
- ISLockPermissions
- IsolatedComponent
- ISSelfReg
- LockPermissions
- MIME
- MoveFile
- MsiServiceConfig
- MsiServiceConfigFailureActions
- MsiLockPermissionsEx
- ODBCAttribute
- ODBCDataSource
- ODBCDriver
- ODBCSourceAttribute
- ODBCTranslator
- ProgId
- PublishComponent
- RemoveIniFile
- SelfReg

- ServiceControl
- ServiceInstall
- TypeLib
- Verb

ある特定の条件下では、RemoveFile テーブルまたは RemoveRegistry テーブルに内容を追加するパッチをエンド ユーザーが 削除できない場合があります。オリジナルパッケージに含まれていないファイルまたはレジストリ エントリを削除するようにパッチが設計されている場合、そのパッチをアンインストールしてもファイルまたはレジストリ エントリは修復されません。

この検証ツールは、スモールアップデートおよびマイナーアップグレードにのみ適用されます。

[このパッチのアンインストールを許可する] チェック ボックスが選択されている場合、この検証テストを実行するために、検証ツールエンジンはインストールの最新バージョンと以前のバージョンとを比較します。

修正アクション

この検証ツールは、パッチの作成中 (または、あとでアップグレードをパッチとしてパッケージすることに決めた場合) に、問題を解決するための修正アクションを取らないと、エンドユーザーがパッチをアンインストールできなくなることを警告します。この警告を解決するには、以下のいずれかを実行します。

- **[このパッチの アンインストールを許可する]** チェック ボックスの選択を解除します。これによって、エンドユーザーがパッチをアンストールすることができなくなります。
- 警告メッセージで示されたテーブルから、新しいデータを削除します。ただしパッチの要件によっては、新しいテーブルデータを削除すると、パッチ作成のもともとの理由であるアプリケーションの問題を修正するという目的を果たさない場合があります。

グローバル アセンブリ キャッシュのアセンブリをパッチする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *QuickPatch*

Windows Installer 3.0 以上で、MsiPatchOldAssemblyFile および MsiPatchOldAssemblyName テーブルを利用すると、パッチ パッケージでオリジナル インストール ソースのランタイム要求なしにグローバル アセンブリ キャッシュ (GAC) 内のアセンブリをパッチすることができます。デフォルトでは、パッチまたは QuickPatch パッケージをビルドすると InstallShield がこれらのテーブル用のエントリを自動的に生成します。(パッチまたは QuickPatch プロジェクトでこの自動エントリ生成を無効にするには、**"MsiPatchOldAssembly テーブルの生成"** プロパティを [いいえ] に設定します。パッチの場合、このプロパティは [パッチのデザイン] ビューのパッチ構成アイテムの [詳細] タブにあります。QuickPatch プロジェクトの場合、このプロパティは一般情報ビューのビルド設定アイテムの [詳細] タブにあります。)

メモ: MsiPatchOldAssemblyFile および MsiPatchOldAssemblyName テーブルのエントリを自動生成するためには、InstallShield はパッチを適用する .msi パッケージの最新バージョンへの書き込みアクセスを必要とします。InstallShield は、このパッケージをパッチ作成の前に変更します。InstallShield はパッケージを書き込み可能にしますが、それが不可能だった場合はビルド警告を生成し、テーブルエントリは作成されません。

これらのテーブルエントリはターゲット システムが Windows Installer 3.0 以上を実行している場合のみ適用されます。システムが Windows Installer 2.0 を実行中の場合、パッチは実行しますがこれらのテーブルは無視され、パッチが GAC 内のファイルを更新する必要がある場合はオリジナルソースパッケージを要求します。Update.exe に Windows Installer 2.0 エンジンのみを含む場合でも、InstallShield は、これらのテーブル エントリを生成します。これは既に Windows Installer 3.0 以上がインストールされたターゲット システムがこれらのテーブル エントリを利用するためです。

差分リリースと完全リリース

プロジェクト: この情報は、InstallScript プロジェクトに適用します。

InstallShield は、InstallScript インストールに対するアップグレード作成方法として、2 つの仕組みを提供します。

- 以前のバージョンがインストールされている場合は既存製品をアップデートし、以前のバージョンが存在しない場合は初回インストールとして作動するフルリリースとして、アップグレードをパッケージすることが可能です。
- アップデートするバージョン間で変更されているデータ (バイトレベルのファイル差分) のみを含む差分リリースとしてアップグレードをパッケージすることが可能です。

差分リリースまたは完全リリースを作成して InstallScript インストールを更新する手順については、ヘルプのこのセクションを参照してください。

以前のバージョンをアップデートする InstallScript リリースを作成する

プロジェクト: この情報は、InstallScript プロジェクトに適用します。

InstallShield では、InstallScript プロジェクトのリリースを作成して、アプリケーションの 1 つまたは複数の既存バージョンをアップデートすることができます。スクリプトで必要な基準をカスタマイズすることができます。

メモ: アップデートする製品が、InstallShield X 以降、InstallShield DevStudio または InstallShield Professional 6.0 以降で作成されたインストールによってインストールされていない場合は、新しいプロジェクトを作成する必要があります。このアップデートに含める予定の変更が広範囲に渡る場合、新規のプロジェクトを作成することも考えられます。

タスク ***アップデート リリースを作成するには、以下の手順に従います:***

1. 新規または既存の InstallScript プロジェクトを開きます。
2. 製品コードが以前のバージョンがインストールされたときの GUID と合致することを確認します。製品コードを調べるには、**[一般情報]** ビュー で **"製品コード"** 設定の値を調べます。

   メモ・ *新しいプロジェクトを作成するときに、元のインストールでインストールされた機能をアップデートする場合、各機能の GUID が、その機能がインストールされた以前のバージョンの GUID と一致することを確認してください。機能の GUID を調べるには、**[機能]** ビューまたは **[セットアップのデザイン]** ビューでそれを選択して GUID プロパティの値を調べます。*
3. 製品アップデートのバージョン番号を指定します。
   c. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
   d. **"バージョン"** 設定に、バージョン番号を入力します。
4. 差分リリースまたは完全リリース をビルドするかどうか指定します。
   a. **[リリース]** ビューで希望のリリースをクリックします。
   b. 差分リリースをビルドする際、**"差分メディア"** プロパティの値を **[はい]** に設定し、ステップ 6 へ進みます。完全リリースをビルドする際、プロパティを **[いいえ]** のままにしておき、ステップ 5 へ進みます。

   メモ・ ***リリース ウィザード****を使用している場合、**[アップデート]** パネルで、リリース フォーマット (完全または差分) を選択します。*
5. 完全リリースをビルドする場合、アップデートを適用する製品のバージョンを指定します。**"サポートされているバージョン"** プロパティで、セミコロンを使って異なるバージョンを区切った適切なバージョン番号 (例、**1.2.3;1.2.4**) を入力します。**"サポートされているバージョン"** プロパティを空白のままにするか、**[非バージョン固有]** すると、製品のすべての以前のバージョンにアップデートが適用されます。ステップ 7 へ移ります。

   メモ・ *これらのバージョン番号は、**[アップデート]** パネルのボックスからバージョンを選択して、リリース ウィザードで指定することもできます。InstallShield Professional 6.0 以前のバージョンで作成されたリリースに対応するバージョン番号を指定しないでください。これらのリリースはアップデートできません。*
6. 差分リリースをビルドする場合、新しい差分リリース作成するとき、**リリース ウィザード**を使用して現在のプロジェクトを比較する 1 つまたは複数の既存のリリースを指定する必要があります。

   メモ・ *InstallShield Professional 6.0 以前のバージョンで作成された既存のリリースを指定しないでください。これらのリリースはアップデートできません。指定された既存のリリースのみが、それから生成された差分リリースでアップデートすることが可能です。現在のプロジェクトのファイルが差分リリースに含まれているかどうかを決定する条件については、「差分リリースと完全リリース」を参照してください。*

- 現在のプロジェクトのリリースを指定するには、次の手順に従います:

  a. **[リリース]** ビューのツリー コントロールで製品名またはリリース名を右クリックし、**[リリース ウィザード]** を選択します。**リリース ウィザード**が開きます。

  b. **[アップデート]** パネルで、**[追加]** をクリックします。**[既存メディアの追加]** ダイアログ ボックスが開きます。

  c. 希望のリリースを選択します。

  d. **[OK]** をクリックします。

- 現在のプロジェクトにないリリースを指定するには、以下の手順に従います:

  a. **[リリース]** ビューのツリー コントロールで製品名またはリリース名を右クリックし、**[リリース ウィザード]** を選択します。**リリース ウィザード**が開きます。

  b. **[アップデート]** パネルで、**[インポート]** をクリックします。**[メディアファイル プロパティ]** ダイアログ ボックスが開きます。

  c. [**メディアヘッダーファイル**] ボックスに完全修飾ファイル名を入力するか、参照ボタンをクリックしてファイルを選択し、希望のリリースの Data1.hdr ファイルを指定します。

  d. **[OK]** をクリックします。

- 指定のリリースのバージョン情報が **[アップデート]** パネルの **[差分メディア]** ボックスに表示されていることを確認してください。バージョン情報が表示されない場合は、以下の手順に従います:

  a. リストボックスでリリースを選択し、[**変更**] をクリックします。**[メディアファイル プロパティ]** ダイアログ ボックスが開きます。

  b. **バージョン情報を以下に指定** オプションを選択して、バージョン番号を入力またはリストで選択します。

  c. **[OK]** をクリックします。

7. リリースをビルドします。

バージョン 6.x で作成されたスクリプトをアップデートが有効にされたインストールで使用する方法についての詳細については、「InstallShield Professional 6.x からの移行」を参照してください。

9

追加のインストール オプション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *スイート/アドバンスト UI*
- *トランスフォーム*

InstallShield では、製品をインストールするインストール パッケージの作成を支援するだけでなく、最終インストール パッケージを強化する他のインストール オプションも提供されています。オプションの中には、複数言語インストールを有効にする、条件ステートメントのビルド、およびインストール前提条件の定義が含まれています。

このセクションには、次のトピックが含まれています:

- 複数言語インストールの作成
- 製品の複数のインスタンスをインストールする
- ターゲット システムの条件を検出する
- インストールされたデータの検索
- インストールおよびプロジェクトのテーブルを編集する
- InstallShield ツールの使用

- Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方
- .msi および .msm データベースを直接編集する

複数言語インストールの作成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

エディション: *InstallShield Premier Edition では、複数言語インストールの作成をサポートします。*

InstallShield は、国際化対応のためにインストールをカスタマイズできる多くの機能をサポートしています。これらの機能を使って、複数の言語でエンド ユーザー テキストを表示する単一のインストール プロジェクトを作成し、言語固有ファイルの条件付きインストールを処理できます。

複数言語インストールを作成するには、まず言語固有のリソースおよびファイルからコードを分離する必要があります。インストールの実行言語によっては、グラフィック、ライセンスファイル、またはカスタム アクション など、別のファイルをインストール用に配布しなければならないこともあります。その他に考慮すべきことは、ターゲット システムの使用地域ごとに、異なるアプリケーション ファイルをインストールする必要があるか、ということです。

タスク

InstallShield では、インストールの作成を以下のようなタスクに分けることによって、複数言語インストールの作成を可能にします。

- インストールに含める言語を指定する。
- 各サポート対象言語の文字列を翻訳する。
- 各言語に必要なエンドユーザー ダイアログを変更する。
- 言語依存コンポーネントをマークする。
- リリースに含める言語を選択する。

このセクションには、次のトピックが含まれています:

- グローバリゼーションのヒント
- デフォルトのプロジェクト言語の設定
- 言語の設定
- インストール言語の選択
- コンポーネントを言語依存としてマークする
- 言語基づいてコンポーネントをインストールする

- リリースへの言語の組み込み
- 文字列エントリの翻訳
- 文字列エントリのエクスポートとインポート
- 各言語ごとのダイアログを変更する
- インストールがユーザー インターフェイスで使用する言語を判別する方法
- 言語サポートのカスタマイズ
- 言語識別子

グローバリゼーションのヒント

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

グローバルなユーザーを対象にプロジェクトの設計を考える場合、次の点に留意してください:

- グローバルな配布の目標は、全世界を対象とし、世界の特定の地域にも対応できる製品にローカライズすることです。
- 国際化のキーはリソースとコードの分離であり、さらに国と言語の非依存性が加わります。
- インストールのグローバリゼーションでは、単純でモジュール式のデザインが必要になります。
- 世界的な規格パッケージの作成においては、最初からグローバルな要件をインストールの規格に組み込みます。
- ビットマップやアイコンを各文化に対応したものにします。ある国で受け入れられるものが、別の国では誤解を招いたり不快感を与えることがあります。
- 英語の文字列は、別の言語の同意義の文字列より通常は短くなっています。文字列を翻訳すると、平均で 30–40% 増加します。これはつまり、静的保存領域と一時保存領域のサイズが大きくなるということです。
- プロンプトを設計する際は、使用できる空き領域の半分だけを使用して、サイズの拡大に備えます。
- 直接入力する要素を画面に配置しないようにします。これらの項目は要素を翻訳した際変更されることがあるためです。

デフォルトのプロジェクト言語の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクトのサポート対象言語の 1 つは、プロジェクトのデフォルト言語として機能する必要があります。デフォルトの言語は以下のすべての事項を決定します:

- 機能の "表示名" 設定またはショートカットの "説明" 設定など、InstallShield の様々なビューで指定した翻訳可能な文字列のすべては、デフォルト言語の文字列であり、これらは [文字列エディター] ビューに表示されます。[文字列エディター] ビュー、または InstallShield 内の別のビューでデフォルト言語の値を編集できます。
- インストールに言語選択ダイアログが含まれておらず、またインストールがターゲット システムの言語をサポートしない場合、デフォルト言語を使ってターゲット システムでインストールが実行されます。(言語選択ダイアログを使って、エンド ユーザーは実行するインストールの言語バージョンを選択できます。)

[一般情報] ビューまたは [文字列エディター] ビューを使って、デフォルト言語を指定または変更できます。

タスク ***[一般情報] ビューを使ってプロジェクトのデフォルト言語を変更するには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **[デフォルト言語]** 設定で、適切な言語を選択します。

タスク ***[文字列エディター] ビューを使ってプロジェクトのデフォルト言語を変更するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[デフォルト言語]** ボックスで、適切な言語を選択します。

ヒント: *特定のリリースにおいて、デフォルトのユーザー インターフェイス言語をオーバーライドできます。そのためには、[リリース] ビューにある [ビルド] タブで、"デフォルト言語" 設定に適切な言語を選択します。*

言語の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

エディション: *InstallShield Premier Edition では、複数言語インストールの作成をサポートします。*

InstallShield では、プロジェクト レベル、コンポーネント レベル、およびリリース レベルで言語を設定できます。言語を指定するレベルによって、プロジェクトに異なる効果があります。次のテーブルでは、様々な言語関連の設定について説明します。

テーブル 1 ・ InstallShield の言語設定

設定	プロジェクトの種類	説明
[一般情報] ビュー設定—セットアップ言語	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	 ***プロジェクト*** *この設定の動作は、プロジェクトの種類によって異なります。* 基本の MSI、InstallScript MSI、マージ モジュール、およびスイート/アドバンスト UI プロジェクトでは、"セットアップ言語" 設定を使って、[リリース] ビューの "UI 言語" 設定にリストする言語を指定します。プロジェクト レベルでこの "セットアップ言語" 設定に言語がリストされていない場合、その特定の UI 言語をプロジェクトのリリースに含めることはできません。 InstallShield Professional Edition で使用可能なアドバンスト UI プロジェクトは、単一の言語のみをサポートします。従って、このプロジェクトの "セットアップ言語" 設定は読み取り専用になっています。 InstallScript および InstallScript オブジェクト プロジェクトでは、"セットアップ言語" 設定を使って、プロジェクトの [コンポーネント] ビューと [リリース] ビューの "言語" 設定にリストする言語を指定します。一般的に、プロジェクト レベルでこの設定に言語がリストされていない場合、プロジェクト内の特定のコンポーネントをその言語に基づいてターゲットにすることはできません。さらに、特定の言語に基づいてコンポーネントと UI 文字列をプロジェクトのリリースに含めることもできません。 この設定を使ってサポート言語を基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、またはスイート/アドバンスト UI プロジェクトに追加すると、InstallShield によってプロジェクトにその言語の文字列エントリが追加されます。文字列エントリには、翻訳済みのビルトイン ユーザー インターフェイス文字列リソースが含まれます。 DIM プロジェクトの "セットアップ言語" 設定を使って、プロジェクトでサポートする言語を指定します。この設定を使ってプロジェクトに言語を追加すると、InstallShield によってその言語の文字列エントリがプロジェクトに追加されます。文字列エントリには、翻訳が必要な文字列リソースが含まれています。 詳細については、「インストール言語の選択」を参照してください。

テーブル 1・InstallShield の言語設定 (続き.)

設定	プロジェクトの種類	説明
"コンポーネント"—設定　言語	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コンポーネントの言語設定を使って、ビルド時のフィルタリングに使用する、コンポーネントが依存する言語を指定できます。デフォルトでは、コンポーネントは言語に依存しません。つまり、コンポーネントのデータ (ファイル、レジストリ エントリなど) はどれも特定の言語に固有ではありません。 ***プロジェクト:*** *InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは、プロジェクトの [一般情報] ビューで言語が選択されていない場合、その言語はプロジェクトのコンポーネント用に利用可能な言語の一覧には表示されません。* コンポーネントが特定の言語にのみ適用することを指定するには、この設定の省略記号ボタン (...) をクリックします。[言語] ダイアログ ボックスが開き、ここでコンポーネントに適切な言語を選択できます。 詳細については、「コンポーネントを言語依存としてマークする」を参照してください。 ***ヒント:*** *実行時にインストールされる言語依存コンポーネントを指定する方法については、「言語基づいてコンポーネントをインストールする」を参照してください。*
[リリース] ビュー 設定 ([ビルド] タブ)—データ言語	基本の MSI、InstallScript MSI、マージ モジュール	各コンポーネントで選択された言語に基づいて特定のコンポーネントを選択または除外する場合、この設定の省略記号ボタン (...) をクリックして、適切な言語を指定します。コンポーネントに指定された言語が、この設定で選択された言語のどれにも一致しない場合、そのコンポーネントはリリースに含まれません。 デフォルトでは、リリースは言語に依存しません。つまり、プロジェクトのコンポーネントはすべてリリースに含まれます。
[リリース] ビュー 設定 ([ビルド] タブ)—UI 言語	アドバンスト UI、基本の MSI、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	"UI 言語" 設定を使って、リリースに含めるユーザー インターフェイス言語を指定できます。適切な言語を選択するには、この設定で省略記号ボタン (...) をクリックします。 プロジェクトの [一般情報] ビューにある "セットアップ言語" 設定で言語が選択されていない場合、その言語は "UI 言語" 設定の利用可能な言語の一覧には表示されません。

テーブル 1・InstallShield の言語設定 (続き.)

設定	プロジェクトの種類	説明
[リリース] ビュー設定 ([ビルド] タブ)—言語	InstallScript、InstallScript オブジェクト	″言語″ 設定を使用して、各コンポーネントで選択された言語に基づいて特定のコンポーネントを含めたり、その他のコンポーネントを除外したりできます。この設定を使って、リリースに含めるユーザーインターフェイス言語を指定することもできます。コンポーネントに指定された言語が、この設定で選択された言語のどれにも一致しない場合、そのコンポーネントはリリースに含まれません。また、プロジェクトに含まれる UI 言語がこの設定で選択された言語の 1 つと一致しない場合、InstallShield はその UI 文字列をリリースに含みません。 デフォルトで、リリースは言語に依存しません。つまり、プロジェクトのコンポーネントまたは UI 文字列はすべてリリースに含まれます。 プロジェクトの [一般情報] ビューで言語が選択されていない場合、その言語は ″ 言語″ 設定の利用可能な言語の一覧には表示されません。

プロジェクトのデフォルト言語に関する詳細については、「デフォルトのプロジェクト言語の設定」を参照してください。

インストール言語の選択

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstalllScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

エディション: *InstallShield Premier Edition では、複数言語インストールの作成をサポートします。*

InstallShield Premier Edition では、インストールに含める言語を選択することができます。プロジェクトに言語を追加すると、その言語用の文字列エントリが追加されます。

プロジェクトにサポート対象言語を追加するには、[一般情報] ビューの ″セットアップ言語″ 設定を使用します。この設定の効果は、使用しているプロジェクトの種類によって異なります。

- 基本の MSI、InstallScript MSI、マージ モジュール、およびスイート/アドバンスト UI プロジェクトでは、[一般情報] ビューの ″セットアップ言語″ 設定を使って、[リリース] ビューの ″UI 言語″ 設定にリストする言語を指定します。このため、プロジェクト レベルでこの ″セットアップ言語″ 設定に言語がリストされていない場合、その特定の UI 言語をプロジェクトのリリースに含めることはできません。

- InstallScript および InstallScript オブジェクト プロジェクトでは、[一般情報] ビューの "セットアップ言語" 設定を使って、プロジェクトの [コンポーネント] ビューと [リリース] ビューの "言語" 設定にリストする言語を指定します。一般的に、プロジェクト レベルでこの設定に言語がリストされていない場合、プロジェクト内の特定のコンポーネントをその言語に基づいてターゲットにすることはできません。さらに、特定の言語に基づいてコンポーネントと UI 文字列をプロジェクトのリリースに含めることもできません。

この設定を使ってプロジェクトにサポート対象言語を追加すると、その言語の文字列エントリがプロジェクトに追加されます。文字列エントリには、翻訳済みのビルトイン ユーザー インターフェイス文字列リソースが含まれます。

InstallShield の新規言語ウィザードを使って、サポートされていない言語をプロジェクトに追加できます。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をプロジェクトに追加すると、その言語は "セットアップ言語" 設定や、InstallShield 内のその他の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

多数のサポート言語で配布するためのインストールをビルドできますが、InstallShield はインストールを 1 つの言語だけで実行します。詳細については、「インストールがユーザー インターフェイスで使用する言語を判別する方法」を参照してください。

コンポーネントを言語依存としてマークする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

すべての新しいコンポーネントは、デフォルトでは言語に依存しません。コンポーネントとそのデータを、その言語をターゲットにするパッケージに組み込む場合、コンポーネントをその言語固有にマークします。

タスク ***コンポーネントを言語依存に指定するには、次の操作を実行します。***

1. [編成] の下にあるビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 言語依存として構成するコンポーネントを選択します。右側のペインにコンポーネントの設定が表示されます。
3. **"言語"** 設定をクリックします。言語の一覧が右下のペインに表示されます。
4. コンポーネントのデータが適用する各言語に対してチェック ボックスを選択します。該当しない言語のチェック ボックスをクリアします。

ヒント: 実行時にインストールされる言語依存コンポーネントを指定する方法については、「言語基づいてコンポーネントをインストールする」を参照してください。

言語基づいてコンポーネントをインストールする

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

リリースの中の言語依存のコンポーネントは、その機能がインストールで選択されることを前提にしているため、ターゲット システムにインストールされます。コンポーネントの言語設定は、リリースにビルドされるコンポーネントが決定されますが、これらのコンポーネントは必ずしもインストールされるとは限りません。

実行時にインストールされる言語依存コンポーネントを指定する (基本の MSI および InstallScript MSI プロジェクト)

ターゲット システムの言語に基づいてコンポーネントをインストールするかどうかを指定するには、コンポーネントの "条件" 設定で Windows Installer プロパティ SystemLanguageID を使用します。

たとえば、次の条件を指定すると、コンポーネントは、(InstallShield でフランス語とマークされているかどうかに関わらず) フランス語(フランス) システムにのみインストールされます。

SystemLanguageID = 1036

実行時に言語特性を決定するために使用できるその他のプロパティには、ユーザーのデフォルト言語を数値識別子で表した UserLanguageID と、インストールが実行中である言語の識別子 ProductLanguage があります。

実行時にインストールされる言語依存コンポーネントを指定する (InstallScript プロジェクト)

InstallScript インストールの実行中に、**FeatureFilterLanguage** 関数を呼び出して、インストールがサポートする言語を制御することができます。

OnFilterComponents イベント ハンドラーで、フレームワークは通常、適切なコンポーネントだけがインストールされるように、この関数をターゲット システムに一致する言語で呼び出します。**FeatureFilterLanguage** を呼び出して、デフォルトの動作をオーバーライドし、指定した言語基準に基づいてコンポーネントをインストールしたり、またはコンポーネントのインストールを防いだりすることができます。

詳細については、「FeatureFilterLanguage」を参照してください。

リリースへの言語の組み込み

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

エディション: *InstallShield Premier Edition では、複数言語インストールの作成をサポートします。*

リリース ウィザードまたは [リリース] ビューを使ってリリースをリリースを構成するとき、インストールのエンドユーザー インターフェイス言語、および言語依存アプリケーション データのフィルターに使用する言語を指定できます。

同じプロジェクトを使って、任意のサポート対象言語用のリリース バージョンをビルドできます。または、リリース ウィザードまたは [リリース] ビューで複数の言語を選択して、インストールに含まれるどの言語ででも実行でき、任意の数の言語に固有のコンポーネントをインストールする単一のインストールを作成することもできます。

言語のユーザー インターフェイス リソースを含める

リリース ウィザードまたは [リリース] ビューで、インストールで使用できる言語を決定します。

複数の言語をサポートするパッケージを作成できますが、インストール自体は 1 つの言語だけで提示されます。詳細については、「インストールがユーザー インターフェイスで使用する言語を判別する方法」を参照してください。

言語に依存するコンポーネントを含める

もう 1 つの考慮事項は、プロジェクトに含めるアプリケーションの言語依存データを決めることです。コンポーネントを言語固有に設定してから、[リリース] ビューの [ビルド] タブで適切な言語関連の設定を構成すると、そのコンポーネントをリリースに自動的に含めたり除外したり (つまり、フィルター)できます。リリース ウィザードでリリースの言語を指定することもできます。

リリース レベルで特定の言語を選択しなかった場合、InstallShield は全てのコンポーネントをリリースに含みます。言語を選択すると、その言語に固有のすべてのコンポーネントとすべての言語依存コンポーネントがリリースにビルドされます。たとえば、英語のみのリリースをビルドする場合、日本語コンポーネント、つまり "言語" 設定に [日本語] が選択されているコンポーネントは除外されます。

ヒント: *[リリース] ビューまたはリリース ウィザードの言語関連の設定で選択した内容によって、いくつかの言語に固有のコンポーネントを設定することもできます。コンポーネントを特定のロケールを持つマシンにのみインストールする方法については、「言語基づいてコンポーネントをインストールする」を参照してください。*

文字列エントリの翻訳

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

*スイート/アドバンスト UI*プロジェクトを通して文字列をハードコード化する代わりに、InstallShield でローカライズ済みテキストを受け入れる領域では、文字列エントリを使用できます。各文字列エントリは、言語非依存の識別子と対応する言語固有の値で構成されます。実行時に、インストールは適切な翻訳済み文字列値を表示します。

プロジェクトのローカライズ処理を簡素化するために、インストール処理中、実行時に表示されるすべてのテキスト文字列は、統合された 1 つのビュー、[文字列エディター] ビューに表示されます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。またこのビューでは、各言語の文字列エントリをファイルにエクスポートして、ファイルにリストされている値を翻訳してから、翻訳済みファイルをプロジェクトにインポートすることができます。

文字列エントリでの作業についての詳細は、「InstallShield で文字列エントリを使用する」を参照してください。

インポートおよびエクスポートの方法については、「文字列エントリのエクスポートとインポート」を参照してください。

文字列エントリのエクスポートとインポート

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクトで使用されているすべてのランタイム文字列を翻訳するタスクを軽減するため、InstallShield では、言語の文字列エントリをタブ区切りテキスト (.txt) ファイルにエクスポートできます。この .txt ファイルを、翻訳されたテキストで更新することができる翻訳者に渡すことができます。エクスポートの完了後、.txt ファイルをインストール プロジェクトにインポートしなおすと、ローカライズされたユーザー インターフェイスが完成します。

ヒント: *エクスポートおよびインポート処理を自動化するために、InstallShield オートメーション インターフェイスが用意されています。詳しくは、「オートメーション インターフェイスを使用した文字列エントリのインポートとエクスポート」をご覧ください。*

言語の文字列エントリを InstallShield からテキスト ファイルにエクスポートする

InstallShield では、言語のすべての文字列エントリ、または一部の文字列エントリをエクスポートできます。特定の日付以降に変更された文字列エントリのみをエクスポートしたい場合が考えられます。これは、新しい文字列エントリ、または最後に翻訳を行った日付以降に変更された文字列エントリのみを翻訳者に渡したい場合に便利です。

タスク ***言語のすべての文字列エントリをエクスポートするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[文字列のエクスポート]** ボタンをクリックします。**[エクスポートするファイルと言語を選択]** ダイアログ ボックスが開きます。
3. InstallShield がテキスト ファイルを保存する場所を参照します。
4. [**ファイル名**] ボックスに、作成するテキスト ファイルの名前を入力します。
5. [**言語**] 一覧で、テキスト ファイルに文字列をエクスポートする言語を選択します。
6. **[保存]** をクリックします。

InstallShield が、選択された全ての言語の文字列エントリを、タブ区切りの Unicode テキスト ファイルにエクスポートします。このテキスト ファイルを翻訳者へ渡します。

ヒント: *デフォルトでは、エクスポート処理は Unicode テキスト ファイルを作成します。[**エクスポートするファイルと言語を選択**] ダイアログ ボックスには、ANSI ファイルに文字列エントリをエクスポートできるチェック ボックスが含まれていますが、ANSI ファイルはデフォルトでは作成されません。ANSI ファイルの場合、2 バイト言語用に適切なコード ページがビルド マシンで必要なため、Unicode テキスト ファイルが推奨されます。詳細については、「言語 サポートのコード ページ要件」を参照してください。*

タスク ***言語の一部の文字列エントリをエクスポートするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. エクスポートする文字列エントリを選択します。
   - 連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。
   - 連続しない複数の行を選択するには、最初の行をクリックしてから CTRL を押しながら追加するそれぞれの行をクリックします。
3. CTRL+C を押して、選択された行の文字列エントリをクリップボードにコピーします。
4. 「メモ帳」などのテキスト エディターで、新しいテキスト ファイルを作成します。

5. CTRL+V を押して、クリップボードの内容をテキスト ファイルに貼り付けます。
6. テキスト ファイルを保存します。Unicode は 2 バイト文字をサポートするため、Unicode エンコードを使ってファイルを保存することが推奨されます。詳細については、「言語 サポートのコード ページ要件」を参照してください。

ヒント: *オートメーション インターフェイスを使って、特定の日付以降に変更された文字列エントリのみをエクスポートする処理を自動化できます。詳細については、「オートメーション インターフェイスを使用した文字列エントリのインポートとエクスポート」を参照してください。*

文字列エントリをテキスト ファイルから InstallShield プロジェクトにインポートする

InstallShield では、翻訳済み文字列エントリをテキスト ファイルから InstallShield プロジェクトにインポートできます。

タスク ***翻訳済みのタブ区切りテキスト ファイルをインポートするには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[文字列エディター]** をクリックします。
2. **[文字列のインポート]** ボタンをクリックします。**[エクスポートするファイルと言語を選択]** ダイアログ ボックスが開きます。
3. インポートするテキスト ファイルを選択します。
4. **[言語]** 一覧から、インポートするテキスト 文字列の言語を選択します。
5. **[開く]** をクリックします。

文字列エントリ インポータが、文字列識別子との競合をチェックします。競合が見つかった場合、プロジェクト内の既存のエントリをテキスト ファイルのエントリで上書きするかどうかを問い合わせるプロンプトが表示されます。

各言語ごとのダイアログを変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

プロジェクトに言語のサポートを追加する場合は、新しく追加された言語に翻訳された各標準ダイアログが提供されます。ダイアログ エディターでそれぞれのサポートされた言語用にこれらのダイアログを編集し、カスタム ダイアログやインポートされたダイアログと共に使用することができます。

タスク ***これらのダイアログを表示するには、次の操作を実行します。***

1. **[ユーザー インターフェイス]** の下のビュー リストにある **[ダイアログ]** をクリックします。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** アイテムを展開します。
3. ダイアログの名前をダブルクリックすると、それぞれのサポート言語についてアイテムが表示されます。
4. ダイアログのレイアウトを変更するには、言語バージョンを選択する必要があります。

留意すべき規則は、それぞれの言語に対してすべてのコントロールが同じプロパティ、文字列エントリ、および動作を持つことです。英語 (US) 版のダイアログにコントロールを追加する場合、ドイツ語版のコントロールにも同じコントロールを追加することになります。ビットマップの Sunken プロパティを True に設定すると、各言語固有版のそのプロパティも True になります。

要素のサイズ変更

上記規則に対する例外は、コントロールの "高さ" および "幅" プロパティです。これらのプロパティは、各言語のバージョンに対して固有です。文字列長は言語によって大幅に異なるため、コントロールのサイズを変更しても、それが他の言語のコントロールのサイズに影響することはありません。

たとえば、ドイツ語に翻訳した後の文字列長に適合させるためにプッシュボタンを大きくする必要があるとしましょう。この場合、すべての版のダイアログは作成されたままのサイズにとどまります。ドイツ語レイアウトでコントロールのサイズを変更したときに、その言語に対してだけサイズが変更されます。

文字列の変更

各ダイアログ中の文字列は、その言語の文字列エントリから取得されます。ローカライズ可能テキストが表示されるコントロールに対して文字列を選択する場合、各言語固有版に対して文字列 ID は同じですが、表示される文字列値は現在の言語の文字列エントリから取得されます。コントロールのプロパティシートでこの値を編集すると、実際にその言語の文字列 ID 値が編集されます。

ファイルリソースの変更

コントロールには、ビットマップやチェック ボックスのようにセットアップパッケージへストリーム入力されるファイルを受け取るものがあります。言語によってファイルリソースが異なる場合があるため、ファイル名の値を編集する際には、その特定の言語に対してのみファイルを提供することになります。

各言語に対してファイル名が異なることがある理由は、すべてのファイル名プロパティが文字列エントリを使用していることです。したがって、すべての言語に対して元は同じファイル名が使用されています。[文字列エディター] ビューで文字列を編集するときや、特定の言語のダイアログ レイアウトを編集するときに、その言語に対してだけ新しいファイル名を入力することができます。

右から左方向へ読み書きされる言語のダイアログ

基本の MSI とマージ モジュール プロジェクトには、右から左に記述する言語のサポートが含まれます。詳細については、「右から左に記述される言語のダイアログ サポート」を参照してください。

インストールがユーザー インターフェイスで使用する言語を判別する方法

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストールをプロジェクトがサポートする言語すべてにローカライズすることができますが、インストールは 1 つの言語でのみ実行可能です。Setup.exe による初期化時、インストールで使用する言語を判別します。

Setup.exe の初期化時における言語の判別

インストールで、複数の言語がサポートされていて (つまり、リリース ウィザードの [セットアップ言語] パネル、または、[リリース] ビューの [ビルド] タブで複数の言語が選択されている状態)、かつ、それらの言語の 1 つが、ターゲット システム上の次のいずれかのアイテムと一致する場合、Setup.exe は一致した言語でインストールを起動します。インストール時、次のアイテムは、表示されている順番で評価され、最初に一致した言語が使用されます:

1. エンドユーザーが /L コマンドライン パラメーターを使って指定した言語
2. 製品の以前のバージョンがターゲット システムにインストールされたことがあり、現在も存在する場合、そのバージョンがインストールされた時に使用された言語
3. ターゲット システムのユーザーのデフォルト言語
4. ターゲット システムのシステムのデフォルト言語
5. ターゲット システムのシステムのデフォルト UI 言語
6. インストールのデフォルト言語。デフォルトの言語は、リリース ウィザードの [セットアップの言語] パネルまたは [リリース] ビューの [ビルド] タブで設定できます。

インストールで 1 言語のみサポートされている場合、Setup.exe は、常にその言語でインストールを起動します。

実行時に [言語の選択] ダイアログを表示する

InstallShield では、Setup.exe の実行時に [言語] ダイアログを表示して、エンドユーザーがインストールの実行言語を選択できるように設定できます。ダイアログには、使用可能な言語の一覧が表示されます。一覧に表示される言語は、リリース ウィザードの [セットアップ言語] パネル、または [リリース] ビューにあるリリースの [ビルド] タブで選択した言語と同じです。[言語] ダイアログのテキストは、Setup.exe が初期化時に選択した言語で表示されます。

エンド ユーザーが [言語] ダイアログでの選択を終了すると、基本の MSI または InstallScript MSI インストールはその言語用のユーザー インターフェイスをすべて含むトランスフォームを適用して、選択された言語でインストールを起動します。

InstallScript スクリプトでは、ユーザー インターフェイス リソースは、_isres*.dll ファイルに格納されています。文字列は、インストールのサポート ファイルに、_isres*.dll ファイルと共に、含まれている文字列テーブル ファイルに組み込まれています。

[言語] ダイアログは Setup.exe によって表示されるため、インストールのユーザー インターフェイスに [言語] ダイアログを含める場合は、セットアップランチャーを作成する必要があります。詳細については、「セットアップランチャーの作成」を参照してください。

メモ: *エンドユーザーが初めて [言語] ダイアログで言語を選択したときは、適切な言語でインストールが実行されます。特定の言語でインストールを実行したあと、同マシンでインストールを他の言語で実行することはできません。これにより、修正モードのインストールを、初回のインストールで使用された言語と異なる言語で実行することができなくなります。また、初回のインストールで使用された言語と異なる言語で機能をインストールすることもできなくなります。*

言語サポートのカスタマイズ

エディション: *新しい言語ウィザードは、InstallShield Premier Edition で提供されています。*

プロジェクト: *新しい言語ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield Premier Edition でサポート対象外の言語でインストールを実行する必要がある場合や、サポート対象の言語で独自の翻訳を作成する場合は、新しい言語ウィザードを使用してこれらの言語のサポートを追加することができます。このウィザードでは、サポートしたい言語、およびそれらを追加するプロジェクトを選択します。選択した言語は、インストールで提供される言語の一覧に追加されます。

このウィザードに関する詳しい情報は、「新しい言語ウィザード」をご覧ください。

言語識別子

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallShield インターフェイス全体を通して、言語 ID または LCID を使って特定の言語を参照しなくてはなりません。ID は、特定の言語を識別する整数の値です。言語 ID は、一般に 16 進数値として指定されますが、Windows Installer の場合は 10 進数で指定する必要があります。

次の表は、すべてのサポート対象言語の LCID 一覧です。言語のサポートに関する詳しい情報は、「InstallShield 実行時の言語サポート」を参照してください。

プロジェクト: *基本の MSI と InstallScript MSI インストールでは、InstallScript インストールとは異なる ID を使用しますので注意してください。*

エディション: *また、アラビア語(サウジアラビア)とヘブライ語は、Premier Edition の基本の MSI、マージ モジュール、およびスイート/アドバンスト UI プロジェクトでのみ使用できます。*

テーブル 2 • 言語識別子

言語名	ID(InstallScript プロジェクト)	ID (Windows Installer ベースのプロジェクトおよびスイート/アドバンスト UI プロジェクト)
アラビア語 (サウジアラビア)	(この言語はサポートされていません。)	1025
バスク語	0x042d	1069
ブルガリア語	0x0402	1026
カタロニア語	0x0403	1027
中国語 (簡体字)	0x0804	2052
中国語 (繁体)	0x0404	1028
クロアチア語	0x041a	1050
チェコ語	0x0405	1029
デンマーク語	0x0406	1030
オランダ語	0x0413	1043
英語 (U.S.)	0x0409	1033
フィンランド語	0x040b	1035
フランス語 (カナダ)	0x0c0c	3084
フランス語 (フランス)	0x040c	1036
ドイツ語	0x0407	1031
ギリシャ語	0x0408	1032
ヘブライ語	(この言語はサポートされていません。)	1037
ハンガリー語	0x040e	1038
インドネシア語	0x0421	1057

テーブル 2 - 言語識別子 (続き.)

言語名	ID(InstallScript プロジェクト)	ID (Windows Installer ベースのプロジェクトおよびスイート/アドバンスト UI プロジェクト)
イタリア語	0x0410	1040
日本語	0x0411	1041
韓国語	0x0412	1042
ノルウェー語	0x0414	1044
ポーランド語	0x0415	1045
ポルトガル語 (ブラジル)	0x0416	1046
ポルトガル語 (ポルトガル)	0x0816	2070
ルーマニア語	0x0418	1048
ロシア語	0x0419	1049
セルビア語 (キリル)	0x0c1a	3098
スロバキア語	0x041b	1051
スロヴェニア語	0x0424	1060
スペイン語	0x040a	1034
スウェーデン語	0x041d	1053
タイ語	0x041e	1054
トルコ語	0x041f	1055

製品の複数のインスタンスをインストールする

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

注意 : *エンド ユーザーが製品の複数インスタンスを同じマシン上に同じコンテキストでインストールできるインストールを作成するには、インストール開発者の洗練されたオーサリング能力と献身的な取り組みが必要です。この機能の使用は、上級インストール開発者にのみお勧めします。*

Windows Installer は、マシン コンテキストでは 1 インスタンスの製品コードのみ、そして各ユーザー コンテキストでは 1 インスタンスのみインストールを許可します。Windows Installer 3.x 以降には、製品コードを変換するトランスフォームがサポートされています。この種類のトランスフォーム (*インスタンス トランスフォーム*) は、各インスタンスの製品コードを変更するため、同じ .msi パッケージを使って、同じ製品の複数インスタンスを同じコンテキストでインストールすることができます。

InstallShield の複数インスタンスの サポートを利用することで、製品の複数インスタンスのサポートに必要な作業を削減することができます。InstallShield では、製品のベース インストール またはマージ モジュールを構成して、サポートする各追加のインスタンスに対応する複数インスタンスを構成することができます。ビルド時に、各インスタンスに対してインスタンス トランスフォームが作成され、インスタンス トランスフォームは .msi パッケージにストリームされます。実行時に、通常インストールはインスタンスの選択ダイアログを表示し、そこでユーザーは新しいインスタンスをインストールするか、既存のインスタンスを保持するかを指定することができます。

このセクションには、次のトピックが含まれています:

- 複数インスタンス サポートの実行時要件
- InstallShield で複数インスタンスを構成する
- 複数インスタンス サポートに関する特別考慮
- 複数インスタンスのサポートを含むリリースの構成とビルド
- 複数インスタンスのサポートを含むメジャー アップグレードの構成とビルド
- 製品の複数インスタンスをインストールするためのサポート
- 製品のインスタンスを複数インストールするときの実行時の動作

複数インスタンス サポートの実行時要件

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

以下のテーブルは、複数インスタンス サポートに関するオペレーティング システムおよび Windows Installer の要件です。

テーブル 1 - 複数インスタンス サポートの実行時要件

要件の種類	要件	影響
ターゲット オペレーティング システム	ターゲット システムには、次のオペレーティング システムまたはそれ以降のバージョンが 1 つ搭載されている必要があります: • Windows Server 2003 • Windows Vista • Windows XP SP1	起動条件を追加して、エンドユーザーが製品のインスタンスを複数インストールすることができるプラットフォームでのみ製品をインストールすることができるようにすることをお勧めします。 ***ヒント:*** *プロジェクト アシスタントの [インストール要件] ページを利用すると、起動条件を製品に素早く追加することができます。詳細については、「プロジェクト アシスタントでオペレーティング システムを指定する」を参照してください。*
Windows Installer のバージョン	Windows Installer 3.0 以降が必要です。	ターゲット システムに Windows Installer 3.0 以降が存在しない可能性がある場合、Windows Installer 再配布可能ファイルを製品に追加することも考慮できます。 詳細については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

Installing Multiple Instances of Products and Patches (Windows Installer ヘルプ ライブラリ)

InstallShield で複数インスタンスを構成する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

次は、複数インスタンスのサポートを基本の MSI プロジェクトに追加するときの標準的なプロセスです:

1. [リリース] ビュー内にある製品構成について、サポートする各インスタンスに新しいインスタンスを追加します。
2. 各インスタンスのプロパティを設定します。
3. ターゲット システムで、各インスタンスにそれぞれ、分離ファイルと非ファイル データが適宜割り当てられることを確認してください。詳細については、「複数インスタンス サポートに関する特別考慮」を参照してください。

4. リリースをビルドします。
5. リリースを十分にテストします。

Installing Multiple Instances of Products and Patches (Windows Installer ヘルプ ライブラリ)

製品構成にインスタンスを追加する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

InstallShield で製品の複数インスタンスを構成するとき、ベースのインストール パッケージによって許可されているベースのインスタンスのほかに、インストールされる各インスタンスにつき、インスタンスを 1 つ追加および定義する必要があります。InstallShield では、インスタンスは製品構成レベルで定義されます。作成した各インスタンスは、InstallShield がビルド時に作成する別のインスタンス トランスフォームに対応します。

メモ: *インストール パッケージが許可するベースのインスタンスのインスタンスは追加しないようにしてください。このインスタンスのインスタンス トランスフォームは必要ありません。*

タスク

製品構成にインスタンスを追加するには、以下の手順に従います:

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、新しいインスタンスを含む製品構成をクリックします。
3. **[複数インスタンス]** タブをクリックします。
4. **[インスタンス]** エクスプローラーを右クリックし、**[新規]** をクリックします。

新しいインスタンスが追加され、数値がその名前として表示されます。また、ProductCode プロパティが、新しい GUID 値と共に、右側のペインに追加されます。

インスタンスが追加されたら、プロパティの設定を行います。詳細については、「インスタンスのプロパティを設定します」を参照してください。

ヒント: *[インスタンス] エクスプローラーに新しいインスタンスが追加されたとき、そのインスタンスの新しい名前を入力できます。あとで名前を変更する場合、インスタンスを右クリックして、[名前の変更] をクリックします。インスタンスの名前付けについては、「インスタンスの名前付け」を参照してください。*

インスタンスの名前付け

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

プロジェクトで製品構成にインスタンスを初めて追加したとき、Windows Installer プロパティ InstanceId がプロジェクトに追加され、その値が 0 に設定されます。0 はベースとなるインストール パッケージの値です。プロジェクトで定義した各インスタンスについては、InstanceId の値に異なる整数が割り当てられる必要があります。

InstallShield は、[リリース] ビューにある [複数インスタンス] タブで表示されているインスタンスの名前を、そのインスタンスに対応する InstanceId プロパティの値として使用します。このため、インスタンスに名前を付けるときは、必ず次のガイドラインに従ってください。

- 製品構成に含まれる各インスタンスには異なる名前を付ける。
- 各インスタンスの名前には整数値を使う (文字その他の記号は使用できません)。

タスク ***新しい InstanceId の値をインスタンスに割り当てるには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、変更するインスタンスを含む製品構成をクリックします。
3. **[複数インスタンス]** タブをクリックします。
4. **[インスタンス]** エクスプローラーで、InstanceId プロパティの値を変更するインスタンスを右クリックして、**[名前の変更]** をクリックします。
5. 新しい値を入力します。

1 つ以上のインスタンスの製品コードを更新する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

各インスタンスは一意の製品コードを持たなくてはなりません。このため、[リリース] ビューで製品構成にインスタンスを追加したとき、ProductCode プロパティが新しい GUID 値と共に自動的に右のペインに追加されます。

InstallShield を使って、プロジェクト内の製品構成で定義さている 1 つ以上のインスタンスの ProductCode プロパティの GUID 値を簡単に変更できます。

タスク ***特定のインスタンスの製品コードを更新するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、インスタンスを含む製品構成をクリックします。
3. **[複数インスタンス]** タブをクリックします。
4. **インスタンス]** エクスプローラーで、製品コードを変更するインスタンスをクリックククします。

5. 右側のペインにあるグリッドで、アップグレードする ProductCode 設定の GUID を右クリックしてから、**[新しい GUID の生成]** をクリックします。

InstallShield が GUID を変更します。

タスク ***製品構成にある [複数インスタンス] タブで定義されている各インスタンスの製品コードをアップデートするには、次の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、インスタンスを含む製品構成をクリックします。
3. **[複数インスタンス]** タブをクリックします。
4. **[インスタンス]** エクスプローラーを右クリックしてから、[各インスタンスの ProductCode を変更する] をクリックします。

InstallShield が、製品構成で定義されている各インスタンスの GUID を変更します。

インスタンスのプロパティを設定します

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

各インスタンスは一意の製品コードを持たなくてはなりません。このため、[リリース] ビューで製品構成にインスタンスを追加したとき、ProductCode プロパティが新しい GUID 値と共に自動的に右のペインに追加されます。

また、各インスタンスについて、ProductName プロパティを異なる名前に設定します。異なる名前を設定することで、[プログラムの追加と削除] で各インスタンスを簡単に識別し易くなります。

インスタンスに追加のプロパティを設定することができます。たとえば、コンポーネントの条件で使用するインスタンス固有のプロパティを定義する必要がある場合があります。また、別の例として、各インスタンスのプロパティを設定して、特定のファイルとレジストリ エントリでインスタンス固有の場所を定義する必要がある場合があります。

個別のインスタンスに適用できるメジャー アップグレードを作成したい場合は、各インスタンスで UpgradeCode プロパティを異なる GUID に設定します。[アップグレード] ビューを使ってプロジェクトにメジャー アップグレード アイテムを追加するとき、メジャー アップグレード アイテムの [共通] タブにある **[アップグレードコードを共有している製品]** を確実に選択してください。

ヒント: *InstallShield は、製品構成の各インスタンスについて、InstanceId プロパティの値に異なる識別子を自動的に設定します。詳細については、「インスタンスの名前付け」を参照してください。*

タスク ***インスタンスのプロパティを設定するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、インスタンスを含む製品構成をクリックします。
3. **[複数インスタンス]** タブをクリックします。
4. **[インスタンス]** エクスプローラーで、インスタンスをクリックします。
5. 右のペイン内のグリッドで、[プロパティ] 列の最後の行にあるフィールドをクリックし、設定するプロパティの名前を入力します。
6. **[値]** 列で、現在のインスタンスに関連付けるプロパティの値を入力します。

ベースのインスタンス (ベースのインストール パッケージによってインストールされるインスタンス) のプロパティ値を設定するには、[プロパティ マネージャー] ビューでプロパティとそれに対応する値を入力します。詳細については、「Windows Installer ベースのプロジェクトにおけるプロパティの作成」を参照してください。

製品構成からインスタンスを削除する

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

マシン コンテキストおよび各ユーザー コンテキストで許可されている製品インスタンスの数を減らす場合、プロジェクト内の製品構成からインスタンスを削除することができます。

タスク ***製品構成からインスタンスを削除するには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、削除するインスタンスを含む製品構成をクリックします。
3. **[複数インスタンス]** タブをクリックします。
4. **[インスタンス]** エクスプローラーで、削除するインスタンスを右クリックして **[削除]** をクリックします。

インスタンスおよびそれに関連付けられているすべてのプロパティがプロジェクトから削除されます。

複数インスタンス サポートに関する特別考慮

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

注意 - *エンド ユーザーが製品の複数インスタンスを同じマシン上に同じコンテキストでインストールできるインストールを作成するには、インストール開発者の洗練されたオーサリング能力と献身的な取り組みが必要です。この機能の使用は、上級インストール開発者にのみお勧めします。*

複数のインスタンスが同じコンテキストで同時に存在できるように、製品の複数インスタンスのインストールを可能にするサポートについては、綿密な事前の計画が必要です。インストールを計画する際、次の点に留意してください:

- 製品のあるインスタンスをアンインストールまたはアップグレードしたとき、他のインスタンスに影響が出ないようにするために、各インスタンスについてターゲット システムにインストールされているファイルおよび非ファイル データの一部またはすべて分離しなくてはならない場合があります。
- 一部のファイルを分離の対象からはずしたい場合があります。たとえば、インストールに 1 つまたは複数の COM サーバーが含まれているとき、COM サーバーを含む各コンポーネントを構成して、インスタンスごとに変わることがない場所にインストールするようにしたい場合があります。これらのコンポーネントについては、Windows Installer が DLL 参照カウントを使用するように "共有" 設定を [はい] に設定します。
- インストールがサポートする各インスタンスについて、別々のコンポーネントのセットを作成する必要がある場合があります。このシナリオでは、各コンポーネントについて [コンポーネント] ビューまたは [セットアップのデザイン] ビューで条件を追加して、各セットが適切なインスタンスにインストールされるようにできます。たとえば、インストール時にインストールされるベースのインスタンスの InstanceId のデフォルト値は 0 であるため、ベースのインスタンスのデータを含むコンポーネントに次の条件を使用することができます:

  InstanceId=0

  インストールされる次のインスタンスのデフォルト InstanceId 値は 1 です。従って、このインスタンスのデータを含むコンポーネントには、次の条件を使用できます:

  InstanceId=1

詳しい情報は、Windows Installer ライブラリの「Authoring Multiple Instances with Instance Transforms」を参照してください。

Installing Multiple Instances of Products and Patches (Windows Installer ヘルプ ライブラリ)
Organizing Applications into Components (Windows Installer ヘルプ ライブラリ)

複数インスタンスのサポートを含むリリースの構成とビルド

プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

リリースの構成

リリースに複数インスタンスのサポートを含めるには、リリースを構成して、Setup.exe セットアップランチャーを含めます。この構成により、インスタンスの選択ダイアログが適宜表示されます。Setup.exe セットアップランチャーを含める方法については、「セットアップランチャーの作成」をご覧ください。

リリースのビルド

複数インスタンスのサポートを含むリリースのビルドは、このサポートを含まないリリースのビルドと若干異なります。複数インスタンスをサポートするために、InstallShield は追加のファイル (インスタンス トランスフォームとインスタンス パッケージ) をビルドします。

インスタンス トランスフォーム

ビルド時に、各インスタンスのインスタンス トランスフォームが生成されます。インスタンス トランスフォームは Disk1 フォルダーに作成された .msi にストリームされます。リリースの構成に従って、.msi ファイルを Setup.exe ファイルに圧縮することもできますし、非圧縮のままにしておくこともできます。

インスタンス パッケージ

InstallShield では、ビルド時に、各インスタンスの .msi ファイルが生成されます。各ファイルには **InstanceIdN.msi** という名前が付いています (**N** は、各インスタンスの InstanceId プロパティの値が入ります)。.msi ファイルは、リリースの場所の下にある *Instances* というフォルダーに格納されます。

Installing Multiple Instances of Products and Patches (Windows Installer ヘルプ ライブラリ)

複数インスタンスのサポートを含むメジャー アップグレードの構成とビルド

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

リリースの構成

リリースに複数インスタンスのサポートを含めるには、リリースを構成して、Setup.exe セットアップランチャーを含めます。この構成により、インスタンスの選択ダイアログが適宜表示されます。Setup.exe セットアップランチャーを含める方法については、「セットアップランチャーの作成」をご覧ください。

アップグレードの構成

複数インスタンスでメジャー アップグレードをサポートするには、次の要件が満たされている必要があります:

- プロジェクト内の各インスタンスには、それぞれに値を持つ UpgradeCode プロパティが必要です。プロジェクトのインスタンスに UpgradeCode プロパティを定義する方法については、「インスタンスのプロパティを設定します」を参照してください。
- より新しいバージョンの各インスタンスには、関連バージョンの同じインスタンスとは異なる ProductCode プロパティの値が必要です。
- より新しいバージョンの各インスタンスには、関連バージョンの同じインスタンスとは異なる、より大きい製品バージョン番号が必要です。
- [アップグレード] ビューのメジャー アップグレードで、**[アップグレード コードを共有している製品]** オプションが選択されている必要があります。

リリースのビルド

複数インスタンスのサポートを含むリリースのビルドは、このサポートを含まないリリースのビルドと若干異なります。複数インスタンスをサポートするために、InstallShield は追加のファイル (インスタンス トランスフォームとインスタンス パッケージ) をビルドします。

インスタンス トランスフォーム

ビルド時に、各インスタンスのインスタンス トランスフォームが生成されます。インスタンス トランスフォームは Disk1 フォルダーに作成された .msi にストリームされます。リリースの構成に従って、.msi ファイルを Setup.exe ファイルに圧縮することもできますし、非圧縮のままにしておくこともできます。

インスタンス パッケージ

InstallShield では、ビルド時に、各インスタンスの .msi ファイルが生成されます。各ファイルには **InstanceIdN.msi** という名前が付いています (**N** は、各インスタンスの InstanceId プロパティの値が入ります)。.msi ファイルは、リリースの場所の下にある *Instances* というフォルダーに格納されます。

Installing Multiple Instances of Products and Patches (Windows Installer ヘルプ ライブラリ)

製品の複数インスタンスをインストールするためのサポート

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

製品の複数インスタンスを更新するパッチを作成する場合、まずマイナー アップグレードまたはスモール アップデートを作成し、それから [パッチのデザイン] ビューで、パッチを構成します。以前のセットアップの場合、アップグレードするバージョンのインストールを行う以前のインストールの Setup.exe ファイルまたは .msi ファイルを選択します。最新のセットアップの場合、アップグレード リリースの Setup.exe ファイルまたは .msi ファイルを選択します。

ヒント: *アップグレードとパッチの作成に関する詳細は、「アプリケーションのアップデート」を参照してください。*

パッチのビルド時に、[パッチのデザイン] ビューで行ったパッチの構成に従って、Update.exe ファイルまたは .msi ファイルが作成されます。

複数インスタンスのサポートを含むパッチのランタイム動作についての概要は、「製品のインスタンスを複数インストールするときの実行時の動作」を参照してください。

Installing Multiple Instances with Instance Transforms (Windows Installer ヘルプ ライブラリ)

製品のインスタンスを複数インストールするときの実行時の動作

プロジェクト: この情報は、基本の MSI プロジェクトに適用します。

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

Setup.exe ファイルを含むインストールを初回で実行する

エンドユーザーが複数インスタンスをサポートするインストールの Setup.exe ファイルを初回で実行したとき、Windows Installer によって、ベースのインストール パッケージに対応するベースのインスタンスが、インスタンス トランスフォームなしに、インストールされます。デフォルトで、標準ダイアログが、インストールで複数インスタンスがサポートされているいないに関わらず表示されます。

製品のインストールが成功したあと、エンドユーザーがインストールを再度実行したとき、インストールするインスタンスを選択できるダイアログがセットアップランチャーによって表示されます。インスタンスの選択ダイアログは、言語の選択ダイアログ (インストールに含まれている場合) と InstallWelcome ダイアログの間に表示されます。ダイアログには、2 つのラジオ ボタンが含まれています:

- **新しいインスタンスをインストールする** – エンド ユーザーは、製品の新しいインスタンスをインストールできます。エンドユーザーがこのオプションを選択すると、新しい製品のインスタンスがインストールされます。
- **既存のインスタンスを保守またはアップグレードする** – エンドユーザーはこのオプションを利用して、既にインストールされているインスタンスの一覧から選択したインスタンスを保守またはアップグレードすることができます。エンドユーザーがこのオプションを選択し、変更するインスタンスを選択してから、[次へ] をクリックすると、インストールで選択されたインスタンスについて PreparingToInstall ダイアログ、MaintenanceWelcome ダイアログ、その後に MaintenanceType ダイアログが表示されます。

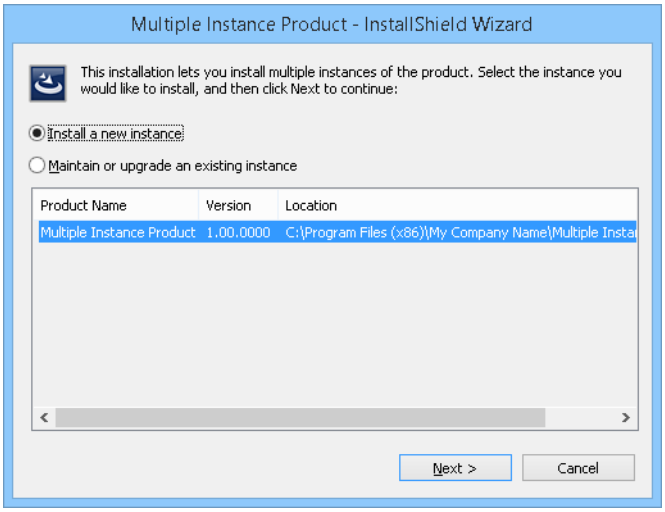


図 1: 製品の 2 回目のインストール時に表示されるインスタンスの選択ダイアログ

エンドユーザーがサポートされている最後のインスタンスをインストールしてから、インストールを再度実行したとき、同じダイアログが表示されますが、**[新しいインスタンスをインストールする]** オプションは無効になっています。**[既存のインスタンスを保守またはアップグレードする]** オプションが選択されると、エンドユーザーは保守またはアップグレードするインスタンスを選択することができます。

ヒント: *インスタンスの選択ダイアログを抑制するには、/instance コマンドライン パラメーターを利用します。*

Setup.exe ファイルを含む完全インストールとしてパッケージされたアップグレードを実行する

複数インスタンスのインストールが Setup.exe ファイルを含む完全インストールとしてパッケージされたアップグレードの場合、エンド ユーザーが Setup.exe ファイルを起動して新しいインスタンスをインストールしたとき、そのアップグレードは初回インストールとして動作します。エンドユーザーが既にインストール済みのインスタンスに **[既存のインスタンスを保守またはアップグレードする]** オプションを選択すると、エンドユーザーは、初回インストールと同じく、インストール時にそのインスタンスをアップグレードまたは保守することができます。

ヒント: *インスタンスの選択ダイアログを抑制するには、/instance コマンドライン パラメーターを利用します。*

初回インストールまたは Setup.exe ファイルを含まない完全インストールとしてパッケージされたアップグレードをコマンドラインから実行する

Setup.exe セットアップランチャーを複数インスタンス インストールに含めなかった場合、エンドユーザーは、インスタンス トランスフォームを使わずに、.msi ファイルを実行して、ベースのインストール パッケージに対応するベース インスタンスをインストールすることができます。

パッケージ内にあるインスタンス トランスフォームの 1 つに対応するインスタンスをインストールするとき、エンドユーザーはMsiExec.exe の適切なコマンドライン パラメーターを含める必要があります。

たとえば、次のコマンドラインは、MSINEWINSTANCE プロパティが 1 に設定されているため、新しいインスタンスをインストールします。Windows Installer は、新しいインスタンスのインストール中に InstanceId1.mst インスタンス トランスフォームを適用します。InstallShield はインスタンス トランスフォームを .msi ファイルに埋め込むため、インスタンス トランスフォームの名前の前にコロンが必要になります。

```
msiexec /i MyPackage.msi MSINEWINSTANCE=1 TRANSFORMS=:InstanceId1.mst
```

InstallShield がインスタンス トランスフォーム ファイルに名前を付けるとき、まずプロパティ名 *InstanceId*、次に InstanceId プロパティのインスタンスの値、そして最後に *.mst* という順番で名前を構成します。従って、InstanceId プロパティが 5 に設定されているインスタンスの場合、トランスフォームの名前は InstanceId5.mst になります。

MSINEWINSTANCE プロパティは、エンドユーザーが指定されたインスタンスを初回でインストールときのみ 1 に設定します。それ以外の場合、エラーが表示されます。

インスタンスの保守またはアップグレードを行うとき、/n パラメーターを利用して、インスタンスの製品コードを渡します。たとえば、次のコマンドラインは MaintenanceType ダイアログを表示します。このダイアログで、エンドユーザーは、指定された製品コードを持つ製品のインスタンスについて、アップグレード/保守/削除のいずれかの操作を指定できます。

```
msiexec /i MyPackage.msi /n {00000001-0002-0000-0000-624474736554}
```

既存のインスタンスをアンインストールする場合、コマンドラインで次の形式を使用します:

```
msiexec /i MyPackage.msi /n {00000001-0002-0000-0000-624474736554} /x
```

Update.exe ファイルがあるパッチとしてパッケージされたアップグレードを実行する

アップグレードをパッチとしてパッケージし、Update.exe ファイルを含めるように指定した場合、インストール時に、パッチ バージョンのインスタンスの選択ダイアログが表示されます。ダイアログには、2 つのラジオ ボタンが含まれています:

- **既存インスタンスの全てにパッチを適用する** – エンド ユーザーは、インストール済みの製品のすべてのインスタンスにパッチを適用できます。エンドユーザーがこのオプションを選択すると、Windows Installer は、すべてのインスタンスが更新されるまで、各インスタンスにパッチを別々に適用します。インスタンスは、InstanceId プロパティの値が小さいものから順番に更新されます。
- **既存インスタンスにパッチを適用する** – エンド ユーザーは、既にインストールされているインスタンスの一覧から選択したインスタンスにパッチを適用することができます。エンドユーザーがこのオプションを選択し、変更するインスタンスを選択してから、[次へ] をクリックすると、インストールで PreparingToInstall ダイアログと PatchWelcome ダイアログが順番に表示されます。

ヒント: *インスタンスの選択ダイアログのパッチ バージョンを抑制するには、/instance コマンドライン パラメーターを使用します。*

Update.exe ファイルのないパッチとしてパッケージされたアップグレードを実行する

パッチに Update.exe ファイルがない .msp ファイルを使用したとき、エンドユーザーが .msp ファイルをコマンドライン パラメーターを渡さずに実行した場合、Windows Installer はエンドユーザーがパッチ バージョンのインスタンスの選択ダイアログで **[すべての既存のインスタンスをパッチする]** オプションを選択したかのように動作します。Windows Installer は、すべてのインスタンスが更新されるまで、各インスタンスに対してパッチを別々に適用します。インスタンスは、InstanceId プロパティの値が小さいものから順番に更新されます。

.msp パッチを特定のインスタンスに適用するには、/p オプションおよび /n オプションを渡します。/n オプションは、パッチが適用されるインストール済みインスタンスの製品コードを指定する必要があります。例:

```
msiexec /p mypatch.msp /n {00000001-0002-0000-0000-624474736554}
```

Installing Multiple Instances of Products and Patches (Windows Installer ヘルプ ライブラリ)
Command-Line Options (Windows Installer ヘルプ ライブラリ)

ターゲット システムの条件を検出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *スイート/アドバンスト UI*
- *トランスフォーム*

インストールを作成するとき、ターゲット システム上に特定の条件が存在する場合があります。たとえば、製品が特定のオペレーティング システムを必要とする場合に、インストールがターゲット システムをチェックして、この要件が満たされていることを確認する場合があります。要件が満たされていない場合、インストールはその要件についてエンド ユーザーに通知するためのエラーを表示します。

InstallShield には、ターゲット システムの様々な条件を検出するためのサポートが含まれています。このセクションには、次のトピックが含まれています:

- 条件ステートメントのビルド
- 管理者権限と昇格された権限を検出する
- 初回インストール、メンテナンス モード、およびアンインストールを検出する
- 初回インストール時にのみ動作をトリガーする
- エンド ユーザーが特定の機能を選択したかどうかを検出する
- エンド ユーザーが特定のオペレーティング システムを実行しているかどうか検出する
- インストールが仮想マシン上で実行されているかどうかを検出する
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

条件ステートメントのビルド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

条件ステートメントは、Windows Installer が True または False に評価できる式です。条件ステートメントは通常、プロパティの値または有無に基づいて特定のアクションを実行したり、コンポーネントのインストールを有効にする目的で使用されます。

Visual Basic と同様の演算子を使用して、比較および論理演算を行うこともできます。詳細については、「条件ステートメントの構文」を参照してください。

ヒント: *InstallShield の [コンポーネント]、[ダイアログ]、[カスタム アクションとシーケンス]、および [カスタム アクション] ビューで、条件プロパティの 省略ボタンをクリックすると、条件ステートメントのビルドが簡単に行える [条件ビルダー] ダイアログ ボックスにアクセスすることができます。*

重要: *このダイアログ ボックスの [OK] ボタンをクリックすると、InstallShield によって基本的な条件検証が行われますが、条件ステートメントが予定通りの結果に評価されるかどうか、確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントの構文」を参照してください。*

例

次のテーブルには、条件ステートメントの例と共にその使い方が説明されています。Windows Installer プロパティおよび条件式の演算子 についての詳細は、Windows Installer ヘルプライブラリを参照してください。

テーブル 1 - 条件ステートメントの例

目的	構文
クライアント バージョンの Windows 7 以降が搭載されているマシンにのみコンポーネントをインストールするには、"条件" 設定にこの式を指定します。	VersionNT>=601 And MsiNTProductType=1
Windows Server 2008 R2 以降のドメイン コントローラーおよびサーバー上にのみコンポーネントをインストールするには、"条件" 設定にこの式を指定します。	VersionNT>=601 And MsiNTProductType>1
Windows XP にのみコンポーネントをインストールするには、"条件" 設定にこの式を指定します。	VersionNT=501 ***メモ:*** *ServicePackLevel および WindowsBuild のプロパティを使用して、異なるオペレーティング システムを識別することもできます。値の一覧表は、Windows Installer ヘルプ ライブラリの「オペレーティング システム プロパティの値」を参照してください。*
Windows 7 以降を実行しているコンピューターにのみコンポーネントをインストールするには、"条件" 設定にこの式を指定します。	VersionNT>=601
製品がインストールされていない場合のみカスタム アクションを実行するには、この条件の Installed プロパティの値を確認します。	インストールされていません
ターゲット システムで米国英語がデフォルト言語として使用されている場合のみコンポーネントをインストールするには、このコンポーネント条件を使用して、SystemLanguageID プロパティの値を米国英語の言語 ID と比較します。	SystemLanguageID=1033 ***メモ:*** *関連プロパティは、*UserLanguageID *(現在のユーザーのデフォルト言語) と* ProductLanguage *(セットアップ プログラムのユーザー インターフェイスで使用されている言語) です。*
画面の最小解像度が 1024 X 768 の場合のみ製品をインストール可能にするには、この条件を [一般情報] ビューの "インストール条件" 設定に追加します。	ScreenX >= 1024 AND ScreenY >= 768
ターゲット システムの RAM が 2 GB 以上の場合のみ製品をインストール可能にするには、この条件を [一般情報] ビューの "インストール条件" 設定に追加します。	PhysicalMemory >= 2048

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

条件ステートメントの構文

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

条件ステートメントでは、プロパティ、Windows Installer のテーブル キー、リテラル、および評価演算子を使用する厳密な構文が使用されます。

機能

条件ステートメントは、条件式が True に評価されるかどうか、つまり数値 1 を持つかどうかに基づいて、カスタム アクションの起動やコンポーネントのインストールなどのアクションをインストールで実行します。False に評価されると、数値 0 が戻されます。プロパティ名の指定は最も単純な式は、プロパティが定義されている限り True になります。たとえば、インストールが Windows 95 または Windows 98 で実行中の場合、単純な条件ステートメント Version9X は True と評価されます。

以下の表で説明されている演算子と値を使用して、より複雑な条件ステートメントをビルドできます。

値

次の表に、条件ステートメントにおける値の使用方法を説明します。環境変数以外のすべての値では、大文字と小文字が区別されます。

テーブル 2 • 値の説明

値	説明
Windows Installer プロパティ	プロパティの名前。存在しないプロパティは空の文字列として評価されます。
整数	-32,767 から +32,767 の範囲の任意の整数値。浮動小数点値は使用できません。
文字列リテラル	テキストは引用符で囲んでください。例: **"InstallShield"**
環境変数	変数名は、先頭にパーセント記号を付けます。例: **%TEMP**

演算子

以下の表に、条件ステートメント演算子を各種類ごとに示します。次の事項に注意してください。

- 演算子は Visual Basic の場合と同じ優先順位で処理されます。
- 丸かっこを使用すると、優先順位を無効にできます。
- 演算子では、大文字と小文字は区別されません。
- Visual Basic と違い、算術演算子はありません。
- 比較で大文字小文字の違いを無視させたい場合は、演算子の先頭にチルド記号 (~) を付けます。
- <> 演算子を使用した場合を除き、文字列と整数を比較すると、結果は常に False になります。

テーブル 3・論理演算子の説明

論理演算子	説明
いいえ	論理否定を実行します。結果は、値が False の場合に True、値が True の場合に False になります。
And	両方の値が True の場合、結果が True になります。
Or	どちらかの値が True の場合、結果が True になります。
Xor	いずれかの値が True の場合、結果が True になります。
Eqv	両方の値が True の場合、または両方の値が False の場合に結果が True になります。
Imp	論理含意を実行します。第 1 条件が False、または第 2 条件が True の場合に、結果が True になります。

テーブル 4・比較演算子の説明

比較演算子	説明
=	1 番目の値と 2 番目の値が等しい場合に結果が True になります。
<>	1 番目の値と 2 番目の値が等しくない場合に結果が True になります。
>	1 番目の値が 2 番目の値を超える場合に結果が True になります。
>=	1 番目の値が 2 番目の値以上の場合に結果が True になります。
<	1 番目の値が 2 番目の値未満の場合に結果が True になります。
<=	1 番目の値が 2 番目の値未満の場合に結果が True になります。

テーブル 5・サブストリング演算子の説明

サブストリング演算子	説明
><	1 番目の文字列に 2 番目の文字列が含まれる場合に結果が True になります。
<<	1 番目の文字列が 2 番目の文字列で始まる場合に結果が True になります。
>>	1 番目の文字列が 2 番目の文字列で終わる場合に結果が True になります。

テーブル 6 - ビット演算子の説明

ビット演算子	説明
><	両方の整数に共通のビットがある場合、結果が True になります。
<<	1 番目の整数の上位 16 ビットが 2 番目の整数と等しい場合に結果が True になります。
>>	1 番目の整数の下位 16 ビットが 2 番目の整数と等しい場合に結果が True になります。

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

管理者権限と昇格された権限を検出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト

Windows XP 以前および Windows Server 2003 以前のシステム上で、ユーザー権限を検出するための 2 つのプロパティは AdminUser と Privileged です。AdminUser プロパティは、エンド ユーザーが管理者権限を持つ場合に設定されます。Privileged プロパティは、インストールが昇格された権限で実行されている場合に設定されます (つまり、エンド ユーザーが管理者権限を持つ場合、システム管理者がインストールを割り当てた場合、またはユーザーとコンピューター ポリシーの両方の AlwaysInstallElevated ポリシーが True に設定されている場合)。ほとんどの場合、Privileged プロパティがより適しています。

Windows Vista 以降と Windows Server 2008 以降では、AdminUser には、デフォルトで Privileged と同じ値が割り当てられます。これらのシステム上で AdminUser と Privileged の区別をとり戻すためには、[プロパティ マネージャー] ビューで MSIUSEREALADMINDETECTION プロパティを 1 に設定します。

Windows Vista 以降と Windows Server 2008 以降では、AdminUser と Privileged は常にユーザー インターフェイス シーケンスで設定されるため、ユーザー インターフェイス シーケンス中に、インストールが実際に昇格された権限で実行されているかどうかを検出することはできませんので、ご注意ください。ただし、遅延システム コンテキストで実行中のカスタム アクションは、Privileged に正しい値を持ちます (MSIUSEREALADMINDETECTION も設定されている場合は AdminUser にも正しい値を持ちます)。遅延システム コンテキストで実行中のアクションのみがシステムを更新するため、インストールが昇格された権限または昇格されていない権限のどちらであるかの区別は、この種類のアクションに対してのみ大きな意味を持ちます。この動作の結果として、AdminUser と Privileged を Windows Vista 以降または Windows Server 2008 以降をターゲットとするプロジェクトのインストール条件に使用できません。

InstallScript および InstallScript MSI プロジェクト

次の InstallScript 式は、Windows Vista 以降と Windows Server 2008 以降のシステムの一部の状況を除いて、エンド ユーザーが管理者権限を有するときに、TRUE を返します。

Is(USER_ADMINISTRATOR, "");

Windows Vista 以降と Windows Server 2008 以降のシステムでは、SE_GROUP_USE_FOR_DENY_ONLY セキュリティ識別子 (SID) 属性がグループに設定されていない場合、**Is** は TRUE を返します。つまり、現在のユーザーが管理者グループに属しているにもかかわらず、Windows Vista 上でインストールを標準アクセス トークンを使ってインストールを実行している場合、**Is** は FALSE を返します。

初回インストール、メンテナンス モード、およびアンインストールを検出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

インストールは、それがターゲット システムで初めて実行されたかどうかを判断できます。

Windows Installer ベースのプロジェクト

Windows Installer シーケンスでは、次のような条件で特定のインストールの種類を検出します。

- 初回インストール: Not Installed
- メンテナンス: Installed
- アンインストール: REMOVE="ALL" (InstallValidate アクションの後)

基本の MSI または InstallScript MSI プロジェクトのすべての "条件" 設定でこれらの条件を使用できます。

InstallScript ベースのプロジェクト

InstallScript の MAINTENANCE 変数は、初回インストールの場合は FALSE となり、メンテナンス モードあるいはアンインストールの場合は TRUE になります。これを利用して、初回インストール時のみに実行させたいコードを作成するには、次の例のような if ステートメント を使用します。

```
if (!MAINTENANCE) then
    // 初回インストール時のみに実行するコード
endif;
```

この種類のコードは、InstallScript および InstallScript MSI プロジェクトのイベント ドリブン型 InstallScript コードで使用できます。

初回インストール時にのみ動作をトリガーする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

アンインストール中は除いて、初回インストール中にのみ特定の動作がトリガされる必要がある場合があります。

基本の MSI プロジェクト

基本の MSI プロジェクトの実行時、初回インストールとメンテナンス インストール (アンインストールを含む) で同じシーケンスが使用されます。個別のアンインストール シーケンスはありません。そのためデフォルトでは、[インストール] シーケンスでスケジュールするカスタム アクションはすべてインストールとアンインストールの両方で実行されます。

アクションを初回インストール時のみ実行するには、Not Installed 条件を利用します。Not Installed (未インストール) 条件は、たとえばインストールされたアプリケーションの Readme ファイルを起動するカスタム アクションに利用できます。

InstallScript および InstallScript MSI プロジェクト

InstallScript と InstallScript MSI インストール プロジェクトでは、MAINTENANCE システム変数を利用してインストールがメンテナンス モードで実行中なのか、初回インストールなのかを判断することができます。次のコード サンプルはスクリプト内での利用法を説明します。

```
if (!MAINTENANCE) then
    // 初回インストール
endif;
```

そして、再インストール/変更/修復の場合、

```
if (MAINTENANCE) then
    // ...初回インストールではない
endif;
```

エンド ユーザーが特定の機能を選択したかどうかを検出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

インストールは、エンド ユーザーが特定の機能のインストールを選択したかどうかを判断できます。使用するプロジェクトの種類によって、その方法は異なります。

基本の MSI プロジェクト

Windows Installer では、&FeatureName=*n* という条件式で、機能の動作状態を検出します。たとえば、ProgramFiles という機能がローカルにインストールするよう選択され、まだインストールされていないことを確認するには、&ProgramFiles=3 という条件式を使用します。

InstallScript および InstallScript MSI プロジェクト

InstallScript の **FeatureIsItemSelected** 関数は、機能選択ダイアログ (**SdFeatureTree** など) で機能が現在選択された状態にある場合、TRUE を戻します。

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

エンド ユーザーが特定のオペレーティング システムを実行しているかどうか検出する

インストールが、ターゲット システム上で実行中のオペレーティング システムを判断できます。

Windows Installer プロパティを使って、オペレーティング システムを検出する

Windows Installer プロパティ VersionNT、Version9X、ServicePackLevel、および WindowsBuild は、ターゲット オペレーティング システムを示します。

プロジェクト: *これらの Windows Installer プロパティは、以下のプロジェクト タイプで、機能、コンポーネント、およびカスタム アクションなどの条件に使用できます。*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

メモ: *プロパティ名は大文字と小文字を区別するため、*Version9X *と* Version9x *は異なるプロパティと見なされます。*

InstallScript 構文変数を使って、オペレーティング システムを検出する

InstallScript 構造 SYSINFO は自動的に初期化され、ターゲット システム上で実行されているオペレーティング システムを示します。InstallScript コードでこの構造を使って、定められたプラットフォームで特定の動作をトリガーできます。この値については、「SYSINFO」を参照してください。

プロジェクト: *以下のプロジェクト タイプでは、イベント ドリブン型の InstallScript コードで SYSINFO 構造を使用できます:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

以下のプロジェクト タイプでは、この構造を InstallScript カスタム アクションでも使用できます:

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

インストールが仮想マシン上で実行されているかどうかを検出する

InstallShield を使って、インストールが以下の種類の仮想マシンで実行されているかどうかを検出できます:

- Microsoft Hyper-V
- VMware Player、VMware Workstation、または VMware Server などの VMware 製品
- Microsoft Virtual PC
- Xen
- Amazon EC 2 インスタンス
- QEMU
- Parallels Desktop
- Google Compute
- Microsoft Azure
- EverRun

仮想マシンをチェックするには、Windows Installer プロパティを使う方法と、InstallScript 構造または関数を使用する方法があります。

メモ: *Bochs エミュレータは検出されません。*

Windows Installer プロパティを使って、仮想マシンを検出する

プロジェクト: *以下のプロジェクト タイプでは、機能、コンポーネント、およびカスタム アクションなどの条件に Windows Installer プロパティを使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

Windows Installer プロパティを使って仮想マシンの存在を検出および、その種類を判別するには、まず SetAllUsers.dll ファイルで ISDetectVM 関数を呼び出すカスタムアクションを作成する必要があります。

タスク ***仮想マシンをチェックするカスタム アクションを追加するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI の場合) または **[カスタム アクション]** (DIM、マージ モジュール プロジェクトの場合) をクリックします。
2. 中央のペインで、**[カスタム アクション]** エクスプローラーを右クリックして、**[新しい MSI DLL]** をポイントしてから、**[Binary テーブルに保存]** をクリックします。新しいカスタム アクションが追加されます。
3. アクションの名前を、たとえば **ISDetectVM** のように変更します。
4. 右側のペインでカスタム アクションの設定を構成します:
   - **"DLL ファイル名"** 設定で、SetAllUsers.dll ファイルを選択します。通常のパスは以下のとおりです:

     <ISRedistPlatformDependentFolder>¥SetAllUsers.dll
   - **"関数名"** 設定で、次の名前を入力します:

     ISDetectVM
   - **"スクリプト内実行"** 設定で、**[即時実行]** を選択します。
   - 1 つまたは複数のシーケンス設定で適切なオプションを選択して、必要に応じて、カスタム アクションをスケジュールします。

カスタム アクションは、実行時に以下の Windows Installer プロパティを設定します:

テーブル 7 ・ 仮想マシンを検出する Windows Installer プロパティ

プロパティ名	説明
IS_VM_DETECTED	このプロパティの値が 1 の場合、インストールは次の仮想マシン環境のうちの 1 つで実行しています: • Microsoft Hyper-V • VMware Player、VMware Workstation、または VMware Server などの VMware 製品 • Microsoft Virtual PC • Xen • Amazon EC 2 インスタンス • QEMU • Parallels Desktop • Google Compute • Microsoft Azure • EverRun この値が設定されない場合、仮想マシンは検出されませんでした。

テーブル 7 • 仮想マシンを検出する Windows Installer プロパティ

プロパティ名	説明
IS_VM_TYPE	このプロパティは、インストールを実行中の仮想マシンの種類を示します。次の値が可能です: • 未定義—仮想マシンは検出されませんでした。 • 1 — VMware Player、VMware Workstation、または VMware Server などの VMware 製品 • 2—インストールは Microsoft Hyper-V マシン上で実行中です。 • 3—インストールは、Microsoft Virtual PC マシン上で実行中です。 • 4—仮想マシンの種類は不明です。 • 5—インストールは Xen マシン上で実行中です。 • 6—インストールは Amazon EC2 インスタンス マシン上で実行中です。 • 8—インストールは QEMU マシン上で実行中です。 • 9時—インストールは Parallels Desktop マシン上で実行中です。 • 10—インストールは Google Compute マシン上で実行中です。 • 11—インストールは Microsoft Azure マシン上で実行中です。 • 12—インストールは EverRun マシン上で実行中です。

前述のプロパティを条件ステートメントに使用できます。たとえば、仮想マシン上で製品の使用が不可能な場合、**[一般情報]** ビューの **"インストール条件"** 設定に次の条件ステートメントを入力します。

```
Not IS_VM_DETECTED
```

VMware イメージ上にのみインストールするコンポーネントの場合、[コンポーネント] ビューの "条件" 設定に次の条件ステートメントを入力します。

```
IS_VM_TYPE=1
```

InstallScript 構造と変数を使って仮想マシンを検出する

プロジェクト: *以下のプロジェクト タイプでは、イベント ドリブン型の InstallScript コードで InstallScript 構造と変数を使用できます:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

以下のプロジェクト タイプでは、この構造と変数を InstallScript カスタム アクションでも使用できます:

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

InstallScript コードで SYSINFO.bIsVirtualMachine 構造を使って、インストールが仮想マシンで実行中かどうかを判別できます。さらに、**GetSystemInfo** 関数と共に VIRTUAL_MACHINE_TYPE 定数を使って、存在する仮想マシンの種類を判別することができます。

たとえば、OnFirstUIBefore などの InstallScript イベント ハンドラー、または InstallScript カスタム アクションでは、以下のようなコードを使用できます:

```
if (SYSINFO.bIsVirtualMachine) then
    SprintfBox (0, "VM", "これは、仮想マシンです。種類: %d",
        GetSystemInfo (VIRTUAL_MACHINE_TYPE, nNumber, szString));
else
    MessageBox ("これは、仮想マシンではありません。", 0);
endif;
```

実行時に、メッセージ ボックスが表示されます。ターゲットシステムが、Microsoft Hyper – V イメージ、VMware イメージ、Microsoft Virtual PC イメージ、Xen イメージ、Amazon EC 2 インスタンスイメージ、QEMU イメージ、Parallels Desktop イメージ、Google Compute イメージ、Microsoft Azure イメージ、または EverRun イメージである場合、メッセージボックスは、「これは、仮想マシンです」と表示して、その仮想マシンの種類も一覧表示します。ターゲット システムがこれらの仮想マシンの 1 つではない場合、メッセージ ボックスは「これは、仮想マシンではありません。」と表示します。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールにおける、異なる条件の使い方

InstallShield のアドバンスト UI およびスイート/アドバンスト UI プロジェクト タイプでは、ランタイムの異なる領域を制御する条件ステートメントのビルドがサポートされています:

- **終了条件**—[一般情報] ビューの "終了条件" 設定を使って、アドバンスト UI またはスイート/アドバンスト UI インストールで、インストールの終了の前に、様々な条件に応じて表示される終了エラー メッセージを指定できます。エンドユーザーがアドバンスト UI またはスイート/アドバンスト UI インストールを起動したとき、定義された条件が True となった場合、エラー メッセージが表示されます。エンドユーザーがエラー メッセージ ボックスを閉じると、インストールは終了します。

- **検出条件**—[パッケージ] ビューのパッケージの "検出条件" 設定を使って、アドバンスト UI またはスイート/アドバンスト UI インストールがターゲット システム上にパッケージが既にインストールされているかどうかを評価するときに使用する条件を指定できます。実行時にこれが false から true (または true から false) に変更されない場合、アドバンスト UI またはスイート/アドバンスト UI インストールはパッケージがペイロードをインストール (または削除) に失敗したものとみなします。また、その後に実行されるメンテナンスと削除操作 (またはその後のメンテナンスおよびインストール操作) も、期待通りに動作しない可能性もあります。
- **対象条件**—[パッケージ] ビューのパッケージの "対象条件" 設定を使って、ターゲット システムがパッケージを実行するのに必要な要件を満たしているかどうかをアドバンスト UI またはスイート/アドバンスト UI インストールが判断するのに使用する条件を指定できます。たとえば、パッケージが 64 ビット システム上でのみ実行する場合、条件に x64 プラットフォーム要件を設定できます。これによって、アドバンスト UI またはスイート/アドバンスト UI インストールは 64 ビット システム上でのみパッケージを起動します。また、エンド ユーザーが現在のパッケージ バージョンを使って、将来リリースされる新しいバージョンを上書きインストールすることを防ぐために、対象条件を設定することもできます。
- **機能条件**—[機能] ビューの "条件" 設定を使って、InstallationFeatures ウィザード ページで機能のインストールをデフォルトで選択するかどうかを評価する条件ステートメントをビルドします。
- **アクションの条件**—[イベント] ビューでイベントにスケジュールされているアクションの "条件" 設定を使って、インストールがアクションを実行するかどうかを評価する条件ステートメントをビルドできます。

  また、[パッケージ] ビューでパッケージの [イベント] 領域にある設定を使って、インストールが選択されたパッケージでアクションを実行するかどうかを評価する条件ステートメントをビルドできます。

  ***プロジェクト・**アクションの条件は、スイート/アドバンスト UI プロジェクトで使用できます。*
- **ウィザード インターフェイスの条件**—ウィザード ページの条件設定および [ウィザード インターフェイス] ビューのコントロールを使って、インストールがアクションを起動するか、エンド ユーザーの応答を検証するか、コントロールを有効化するか、ページまたはコントロールを表示するかどうかなど、動作を評価する条件ステートメントをビルドできます。
- **モード条件**—InstallShield では、2 つの種類のモード条件 ([インストール] モード条件、[メンテナンス] モード条件) が自動的に作成されます。モード条件は、プロジェクト ファイル (.issuite) でのみ表示されます。この種類の条件は、InstallShield 内で編集はできません。これらのモードは、アドバンスト UI またはスイート/アドバンスト UI インストールを初回インストール モードで実行するか、メンテナンス モードで実行するかを判別します。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする」を参照してください。

条件演算子

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで定義された各条件は、次の演算子のいずれかで始まります:

- **Any**—Any 条件グループは、論理演算子 OR と同じ要領で動作します。Any グループに含まれる任意の条件が True 評価された場合、条件グループ全体が True 評価されます。Any グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が False 評価されます。
- **All**—All 条件グループは、論理演算子 AND と同じ要領で動作します。All グループが True 評価されるためには、All グループに含まれる条件のすべてが True 評価される必要があります。
- **None**—None 条件グループは、論理演算子 NOR と同じ要領で動作します。None グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が True 評価されます。None 条件グループに含まれる

任意の条件が True 評価された場合、条件グループ全体が False 評価されます。None 条件グループが 1 つの条件ステートメントで構成される場合、これは論理演算子 NOT と同じ要領で動作します。

条件チェックの種類

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な異なる種類から選択できます。たとえば、オペレーティング システムのバージョン、ファイルまたはレジストリ エントリの存在、およびインストール済み製品の存在など。

サポートされている条件チェックの各種類、および条件の下に表示される各サブ設定についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類」を参照してください。

ビルド条件の構文

条件ステートメントをビルドするとき、1 つ以上の条件設定を使って演算子または条件チェックを追加します。

条件ステートメントを条件設定に追加するには、設定の [新しい条件] ボタンをクリックします:

[新しい条件] ボタンをクリックすると、条件設定の下に演算子の行 (デフォルトの演算子は Any) が追加されます。条件の演算子を変更するには、演算子設定のドロップダウン矢印をクリックしてから、適切なオプションをクリックします。

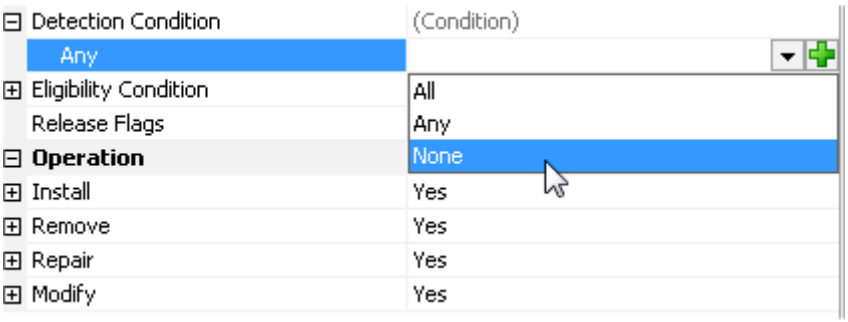


図 1: デフォルト演算子の変更

演算子を追加したあと、演算子設定の [新しい条件] ボタンをクリックしてから、適切な条件チェックの種類をクリックします。ここで、InstallShield によって演算子設定の下にサブ設定 (選択されたチェックの種類によっては、さらに多くのサブ設定) が追加されます。

条件ステートメントが無効である場合 (たとえば、設定を入力する必要がある場合に空白のままであるとき)、その設定に警告アイコンが表示されます。条件が無効である理由を調べるには、ポインターをアイコンの上に配置すると、問題を解説するツールヒントが表示されます。

条件内部に別の条件をネストするには、演算子設定の [新しい条件] ボタンをクリックしてから、適切な演算子をクリックして、ネストされた条件を設定します。Any/All/None メタ条件の 1 つは、*条件グループ*と呼ばれます。ネスト条件グループおよび条件設定の下にある条件のすべては、*条件ツリー*を構成します。

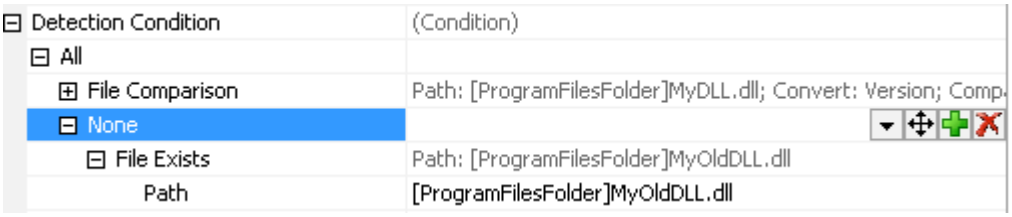


図 2: 条件内部に条件をネストする

条件ツリー内で条件または条件グループを移動させるには、アイテムの設定にある [条件を移動] ボタンをポイントしてから、適切な移動方向 (上に移動、下に移動、左に移動、または右に移動) をクリックします。

構成中のパッケージが .msi パッケージである限り、MSI パッケージの対象条件および検出条件を定義する際に、現在のパッケージの製品コード、パッケージ コード、および製品バージョンのプレースホルダーとしてアスタリスクを使用することができます。

図 3: .msi パッケージのデフォルトの対象条件

また、構成中のパッケージが .msi パッケージである限り、MSI アップグレードの対象条件および検出条件を定義する際に、現在の製品のアップグレード コードおよび最小バージョン番号/最大バージョン番号のいずれかのプレースホルダーとしてアスタリスクを使用することができます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、条件チェックの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な異なる種類から選択できます。

各種類の条件には、1 つ以上の関連サブ設定があります。以下の種類の条件を使用できます:

テーブル 8 • 条件チェックの種類

条件チェックの種類	説明
プラットフォーム	ターゲット システムをチェックして、オペレーティング システムのバージョンやアーキテクチャ、その他のプラットフォーム情報など、1 つまたは複数の特性を確認します。 この種類の条件チェックに表示される各設定の説明は、「"プラットフォーム" 条件の設定」を参照してください。
ファイルの存在	ターゲット システムをチェックして、特定のファイルの有無を確認します。 この種類の条件チェックにおける、この設定の説明は、「"ファイルの存在" 条件の設定」を参照してください。
フィルの比較	ターゲット システムをチェックして、特定のファイルの特定の情報 (日付、バージョン番号、またはコンテンツ) を確認します。 この種類の条件チェックに表示される各設定の説明は、「"ファイルの比較" 条件の設定」を参照してください。
レジストリの存在	ターゲット システムをチェックして、特定のレジストリ キーの存在、およびオプションで特定の値名の存在を確認します。 この種類の条件チェックに表示される各設定の説明は、「"レジストリの存在" 条件の設定」を参照してください。
レジストリの比較	ターゲット システムをチェックして、特定のレジストリ エントリの特定の情報 (DWORD、QWORD、文字列、複数文字列、ファイル バージョン、または製品バージョン) を確認します。値名を指定しなかった場合、指定したキーのデフォルト値が比較時に使用されます。 この種類の条件チェックに表示される各設定の説明は、「"レジストリの比較" 条件の設定」を参照してください。
プロパティの比較	特定のビルトイン アドバンスト UI またはスイート/アドバンスト UI プロパティ、あるいは [プロパティ マネージャー] ビューで定義されたプロパティの値をチェックできます。 この種類の条件チェックに表示される各設定の説明は、「"プロパティの比較" 条件の設定」を参照してください。
MSI パッケージ	ターゲット システムをチェックして、特定の .msi パッケージによってインストールされた製品の存在を確認します。 この種類の条件チェックに表示される各設定の説明は、「"MSI パッケージ" 条件の設定」を参照してください。

テーブル 8 • 条件チェックの種類 (続き.)

条件チェックの種類	説明
MSI アップグレード	ターゲット システムをチェックして、アップデートする製品のバージョンが存在するかどうか、またはダウングレードしてはならない製品のより新しいバージョンが存在するかどうかを確認します。いずれの場合も、その存在を確認する製品は .msi パッケージによってインストールされます。 この種類の条件チェックに表示される各設定の説明は、「"MSI アップグレード" 条件の設定」を参照してください。
有効なパッケージ	アドバンスト UI またはスイート/アドバンスト UI インストール内の別のパッケージまたは機能がターゲット システム上でインストールの対象になるかどうかを確認します。たとえば、ベース パッケージのインストールが可能である場合のみパッチ パッケージのインストールが可能である場合、[有効なパッケージ] 条件を作成して、パッチ パッケージのインストールをベース パッケージのインストールが可能であるかどうかに関連付けます。この場合、[パッケージ] ビューでパッチ パッケージを選択してから、条件を作成し、その条件内でベース パッケージのパッケージ GUID を指定します。 この種類の条件チェックにおける、この設定の説明は、「"対象パッケージ" 条件の設定」を参照してください。
InstallScript パッケージ	ターゲット システムをチェックして、特定の InstallScript パッケージによってインストールされた製品の存在を確認します。 この種類の条件チェックに表示される各設定の説明は、「"InstallScript パッケージ" 条件の設定」を参照してください。
ロケール	ターゲット システムで、1 つまたは複数のロケール関連の設定に一致するものを検索します。 この種類の条件チェックに表示される各設定の説明は、「"ロケール" 条件の設定」を参照してください。
インストール済みスイート	アドバンスト UI またはスイート/アドバンスト UI インストールの特定のバージョンがターゲット システム上で既にインストールされているかどうかを確認します。 この種類の条件チェックに表示される各設定の説明は、「"インストール済み" 条件の設定」を参照してください。
Windows アプリ パッケージ	ターゲット システムをチェックして、特定の Windows アプリの存在を確認します。 この種類の条件チェックに表示される各設定の説明は、「「Windows アプリ パッケージの条件" の設定」を参照してください。

テーブル 8 • 条件チェックの種類 (続き.)

条件チェックの種類	説明
Windows アプリ パッケージの対象	ターゲット システムで Windows アプリ パッケージの実行-時依存ファイルの存在を確認して、これをサポートしない Windows または Windows Server バージョンに Windows アプリ パッケージがインストールされることのないように防ぎます。 この条件は、Windows アプリ パッケージの対象条件のみで使用可能です。別のパッケージ タイプで使用した場合、正しく機能しません。 ***ヒント:*** *Windows アプリ パッケージを正しくサイドロードするためには、サイドローディングを有効化する必要があります。サイドローディングは Windows 10 以降で使用できます。サイドローディングに関する詳細は、MSDN 記事「Enable your device for deployment 」を参照してください。*
UWP タイプが存在する	ターゲット システムをチェックして、UWP 機能の存在を確認します。たとえば、デスクトップ ブリッジの存在を確認する条件ステートメントを作成するには、Windows.ApplicationModel.FullTrustProcessLauncher タイプを確認します。これは、条件付でインストールをブロックする、または .msi および Windows アプリ パッケージ (.appx) のどちらをインストールするかを選択する場合に使用できます。 この種類の条件チェックに表示される各設定の説明は、「"UWP タイプの存在" 条件の設定」を参照してください。
パッケージの操作	***プロジェクト:*** *この種類の条件は、スイート/アドバンスト UI プロジェクトで使用できます。* スイート/アドバンスト UI インストール内の特定のパッケージのターゲット状態をチェックします。この種類の条件は、[イベント] ビューまたは [パッケージ] ビューのイベントに関連付けられたアクションでのみ使用できます。 この種類の条件チェックに表示される各設定の説明は、「"パッケージの操作" 条件の設定」を参照してください。
機能の操作	***プロジェクト:*** *この種類の条件は、スイート/アドバンスト UI プロジェクトで使用できます。* スイート/アドバンスト UI インストール内の特定の機能のターゲット状態をチェックします。この種類の条件は、[イベント] ビューまたは [パッケージ] ビューのイベントに関連付けられたアクションでのみ使用できます。 この種類の条件チェックに表示される各設定の説明は、「"機能の操作" 条件の設定」を参照してください。

テーブル 8 • 条件チェックの種類 (続き.)

条件チェックの種類	説明
拡張条件 DLL	作成した C/C++ DLL を使って、ターゲット システム上で、他の種類の条件チェックによって評価されない要因をチェックする自作のカスタム条件を実装することができます。 この種類の条件チェックに表示される各設定の説明は、「"拡張条件" の設定」を参照してください。
モード	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のモードをチェックします。 この種類の条件チェックは、[ウィザード インターフェイス] ビューの様々なウィザード インターフェイス要素で使用できます。 この種類の条件チェックに表示される各設定の説明は、「"モード条件" の設定」を参照してください。

"プラットフォーム" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、"プラットフォーム" 条件を作成できます。

"プラットフォーム" 設定は、[プラットフォーム] サブ設定で構成されたプラットフォーム関連の条件を表示します。ターゲット システム上のオペレーティング システム バージョン、アーキテクチャ、およびその他のプラットフォームの詳細を評価する条件を定義するには、必要に応じて 1 つ以上のプラットフォーム関連のサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 9 • "プラットフォーム" 条件の設定

設定	説明
OS バージョン	ターゲット システム上の Windows バージョンを評価する条件ステートメントを作成するには、以下のいずれかを実行します: • 完全一致を確認するには、Windows のバージョン番号を *major.minor* 形式で入力します。 • 最小および最大バージョン番号を指定するには、ダッシュを使ってその範囲を入力します。たとえば、Windows の バージョン 5.0 からバージョン 6.1 をテストするには、以下を入力します: **5.0–6.1** • 最小バージョン番号を確認するには、最小バージョン番号の後にダッシュを入力します。たとえば、バージョン 5.0 以降をテストするには、以下を入力します: **5.0–** • 最大バージョン番号を確認するには、ダッシュの後に最大バージョン番号を入力します。たとえば、バージョン 6.0 以前をテストするには、以下を入力します: **–6.0** 有効なバージョンリストは、MSDN Web サイトのドキュメントに記載されている OSVERSIONINFOEX 構造 の dwMajorVersion および dwMinorVersion メンバーを参照してください。 ***重要*** *• "OS バージョン" 設定でオペレーティング システムの範囲、ならびに "サービス パック" 設定でサービス パックの範囲を指定する場合、実行時に条件が予定通りに評価する事を確認してください。*

テーブル 9 • "プラットフォーム" 条件の設定 (続き.)

設定	説明
サービス パック	ターゲット システム上の Windows サービス パック番号を評価する条件ステートメントを作成するには、以下のいずれかを実行します: • 完全一致を確認するには、特定の Windows サービス パック番号を入力します。たとえば、SP2 をテストするには、数字 **2** を入力します。 • 最小および最大サービス パック番号を指定するには、ダッシュを使ってその範囲を入力します。たとえば、Windows の バージョン SP1 から SP3 をテストするには、以下を入力します: **1-3** • 最小サービス パック番号を確認するには、最小サービス パック番号の後にダッシュを入力します。たとえば、SP2 以降をテストするには、以下を入力します: **2-** • 最大サービス パック番号を確認するには、ダッシュの後に最大サービス パック番号を入力します。たとえば、SP2 以前をテストするには、以下を入力します: **-2** 有効なサービス パック番号のリストは、MSDN Web サイトのドキュメントに記載されている OSVERSIONINFOEX 構造 の wServicePackMajor および wServicePackMinor メンバーを参照してください。 ***重要*** *• "OS バージョン" 設定でオペレーティング システムの範囲、ならびに "サービス パック" 設定でサービス パックの範囲を指定する場合、実行時に条件が予定通りに評価する事を確認してください。*
CSD バージョン	ターゲット システム上の Windows CSD バージョンを評価する条件ステートメントを作成するには、特定の CSD バージョンを入力します。**Service Pack 2** は CSD バージョンの一例です。
ビルド番号	ターゲット システム上の Windows 最小ビルド番号を評価する条件ステートメントを作成するには、特定の ビルド番号を入力します。 有効なビルド番号の一覧は、MSDN Web サイトのドキュメントに記載されている「OSVERSIONINFOEX structure」の dwBuildNumber メンバーを参照してください。
製品の種類	ターゲット システムがドメイン コントローラー、サーバー、またはワークステーションかどうかを評価する条件ステートメントを作成するには、適切なオプションを選択します。 有効な種類のリストは、MSDN Web サイトのドキュメントに記載されている OSVERSIONINFOEX 構造 の wProductType メンバーを参照してください。

テーブル 9 • "プラットフォーム" 条件の設定 (続き.)

設定	説明
アーキテクチャ	ターゲット システムのアーキテクチャ (x86、x64、IA64、Armまたは Arm64) を評価する条件ステートメントを作成するには、適切なオプションを選択します。

"ファイルの存在" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、"ファイルの存在" 条件を作成できます。

"ファイルの存在" 設定は、ターゲット システムで特定のファイルが存在するかどうかをチェックするように構成された条件を表示します。この種の条件を定義するには、必要に応じて "ファイルの存在" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 10 • ″ファイルの存在″ 条件の設定

設定	説明
パス	特定のファイルの存在を確認する条件ステートメントを作成するには、次の操作の 1 つを行います: • ファイルの名前とパスを入力します。パスに定義済みのフォルダーを含めるには、一覧で適切なプロパティを選択します。64 ビット フォルダーの場所の 1 つを選択すると、インストールは 64 ビット ターゲット システム上の 64 フォルダーの場所をチェックし、32 ビット ターゲット システム上の 32 ビット フォルダーの場所をチェックします。 • ″パス″ 設定にファイル名のみを入力します。″検索パス″ 設定を使って、実行時にインストールがチェックする 1 つ以上のパスを指定します。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

テーブル 10 • "ファイルの存在" 条件の設定 (続き.)

設定	説明
検索パス	"パス" 設定にパスとファイル名を指定する場合、"検索パス" 設定は空白のまま残します。 "パス" 設定にファイル名のみを指定する場合、"検索パス" 設定を使って、ファイルをチェックするターゲット システム上の場所へのパス一覧を指定します。複数のパスはセミコロン (;) で区切ります。パスに定義済みのフォルダーを含めるには、適切なプロパティを使用します。次の "検索パス" 設定のサンプル値は、ディレクトリ プロパティを使って 2 つの異なるパスを検索します: **[CommonFilesFolder]Microsoft Shared¥Office15;[CommonFilesFolder]Microsoft Shared¥Office14** できる限り検索パスを限定することが、強く推奨されます。一般的に、検索パスが広いほど、インストーラーがターゲット システム上でチェックを行う時間が長引きます。 この設定に入力した任意のパスに [System64Folder] が含まれている場合、インストールは任意の指定されたパスをシステム フォルダーでチェックするとき、64 ビット ターゲット システム上で、ファイル システムのリダイレクトを無効にします。この設定のパスに [System64Folder] を入力しなかった場合、インストールはシステムのリダイレクトを有効な状態に保ちます。 つまり、この設定に **[System64Folder];[SystemFolder]** の値を入力すると、インストールは 32 ビット および 64 ビット両方のターゲット システム上で C:¥Windows¥System32 をチェックします。**[SystemFolder]** の値を入力すると、インストールは 32 ビット ターゲット システム上では C:¥Windows¥System32 を、64 ビット システム上では C:¥Windows¥SysWOW64 をチェックします。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 たとえば、ターゲット システム上で PATH 環境変数に定義されているすべてのディレクトリの "パス" 設定で指定されているファイルを検索するには、"検索パス" 設定に次の値を入力します: [%PATH]

"ファイルの比較" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、"ファイルの比較" 条件を作成できます。

"ファイルの比較" 設定は、ターゲットシステムで特定のファイルの特定の情報 (日付、バージョン番号、またはコンテンツ) をチェックするように構成された条件を表示します。この種の条件を定義するには、必要に応じて "ファイルの比較" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 11 • "ファイルの比較" 条件の設定

設定	説明
パス	特定のファイルについて特定の情報を確認する条件ステートメントを作成するには、次の操作の 1 つを行います: • ファイルの名前とパスを入力します。パスに定義済みのフォルダーを含めるには、一覧で適切なプロパティを選択します。64 ビット フォルダーの場所の 1 つを選択すると、インストールは 64 ビット ターゲット システム上の 64 フォルダーの場所をチェックし、32 ビット ターゲット システム上の 32 ビット フォルダーの場所をチェックします。 • "パス" 設定にファイル名のみを入力します。"検索パス" 設定を使って、実行時にインストールがチェックする 1 つ以上のパスを指定します。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

テーブル 11 • "ファイルの比較" 条件の設定 (続き.)

設定	説明
検索パス	"パス" 設定にパスとファイル名を指定する場合、"検索パス" 設定は空白のまま残します。 "パス" 設定にファイル名のみを指定する場合、"検索パス" 設定を使って、ファイルをチェックするターゲット システム上の場所へのパス一覧を指定します。複数のパスはセミコロン (;) で区切ります。パスに定義済みのフォルダーを含めるには、適切なプロパティを使用します。次の "検索パス" 設定のサンプル値は、ディレクトリ プロパティを使って 2 つの異なるパスを検索します: **[CommonFilesFolder]Microsoft Shared¥Office15;[CommonFilesFolder]Microsoft Shared¥Office14** できる限り検索パスを限定することが、強く推奨されます。一般的に、検索パスが広いほど、インストーラーがターゲット システム上でチェックを行う時間が長引きます。 この設定に入力した任意のパスに [System64Folder] が含まれている場合、インストールは任意の指定されたパスをシステム フォルダーでチェックするとき、64 ビット ターゲット システム上で、ファイル システムのリダイレクトを無効にします。この設定のパスに [System64Folder] を入力しなかった場合、インストールはシステムのリダイレクトを有効な状態に保ちます。 つまり、この設定に **[System64Folder];[SystemFolder]** の値を入力すると、インストールは 32 ビット および 64 ビット両方のターゲット システム上で C:¥Windows¥System32 をチェックします。**[SystemFolder]** の値を入力すると、インストールは 32 ビット ターゲット システム上では C:¥Windows¥System32 を、64 ビット システム上では C:¥Windows¥SysWOW64 をチェックします。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 たとえば、ターゲット システム上で PATH 環境変数に定義されているすべてのディレクトリの "パス" 設定で指定されているファイルを検索するには、"検索パス" 設定に次の値を入力します: [%PATH]

テーブル 11 • ″ファイルの比較″ 条件の設定 (続き.)

設定	説明
変換	比較するデータの種類を選択します。選択可能なオプションは以下のとおりです: • **コンテンツ**—比較するデータは文字列です。 • **バージョン**—比較するデータは *nn.nn.nn.nn* 形式のファイル バージョンです。 • **ローカル日付とローカル時刻**—比較するデータはアドバンスト UI またはスイート/アドバンスト UI インストールを実行するターゲット システム上のローカル日付とローカル時刻です(例、**4/26/2004 3:57:46 PM**)。
比較	ターゲット システム上の値と ″比較対象″ 設定で指定した値とをどのように比較するかを説明するオプションを選択します。たとえば、″比較″ 設定の値が [以下] で、″比較対象″ 設定が [4.1] であるファイル バージョン条件を考慮します。実行時に、この条件はターゲット システム上でファイル バージョンが 4.0 または 4.1 の場合、True 評価します。ターゲット システム上でファイル バージョンが 4.2 の場合、False 評価します。
比較対象	ターゲット システム上にある値を比較する値を指定します。入力する値は、″変換″ 設定で選択したオプションによって異なります。 • ″変換″ 設定で [コンテンツ] が選択されている場合、″比較対象″ 設定に比較するファイル文字列を入力します。 • ″変換″ 設定で [バージョン] が選択されている場合、ファイル バージョンを *nn.nn.nn.nn* 形式で入力します。 • ″変換″ 設定で [ローカル日付とローカル時刻] が選択されている場合、アドバンスト UI またはスイート/アドバンスト UI インストールが実行されるターゲットシステム上のローカル日付とローカル時刻を入力します(例、4/26/2004 **3:57:46 PM**)。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 ″比較″ 設定では、ターゲット システム上の値と ″比較対象″ 設定で指定した値との比較方法を指定します。たとえば、″比較″ 設定の値が [以下] で、″比較対象″ 設定が [4.1] であるファイル バージョン条件を考慮します。実行時に、この条件はターゲット システム上でファイル バージョンが 4.0 または 4.1 の場合、True 評価します。ターゲット システム上でファイル バージョンが 4.2 の場合、False 評価します。

"レジストリの存在" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、"レジストリの存在" 条件を作成できます。

"レジストリの存在" 設定は、ターゲット システムで特定のレジストリ キー、およびオプションで特定の値名が存在するかどうかをチェックするように構成された条件を表示します。この種の条件を定義するには、必要に応じて "レジストリの存在" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 12 · "レジストリの存在" 条件の設定

設定	説明
レジストリ キー	特定のレジストリ キーの存在をチェックする条件ステートメントを作成するには、この設定で適切なルート キーを選択してから、残りのレジストリ パスを入力します。使用可能なルート キーは次の通りです: • HKLM—HKEY_LOCAL_MACHINE • HKCU—HKEY_CURRENT_USER • HKCR—HKEY_CLASSES_ROOT • HKU—HKEY_USERS この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

テーブル 12 - "レジストリの存在" 条件の設定 (続き.)

設定	説明
値名	指定したレジストリ キーの特定のレジストリ値の存在をチェックする条件ステートメントを作成するには、適切なレジストリ値を入力します。特定のレジストリ値の存在をチェックしたくない場合、つまり、レジストリ キーの存在のみをチェックしたい場合は、この設定を空白のままにします。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
64 ビット キー	アドバンスト UI またはスイート/アドバンスト UI インストールが 64 ビット システム上のレジストリの 64 ビット部分をチェックするかどうかを指定します。True を選択すると、アドバンスト UI またはスイート/アドバンスト UI インストールは 32 ビット ターゲット システム上では、この条件ステートメントを無視します。

"レジストリの比較" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、"レジストリの比較" 条件を作成できます。

"レジストリの比較" 設定は、ターゲットシステムで特定のレジストリ エントリの特定の情報 (DWORD、QWORD、文字列、複数文字列、ファイル バージョン、または製品バージョン) をチェックするように構成された条件を表示します。値名を指定しなかった場合、指定したキーのデフォルト値が比較時に使用されます。この種の条件を定義するには、必要に応じて "レジストリの比較" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 13 • "レジストリの比較" 条件の設定

設定	説明
レジストリ キー	レジストリ キーの特定の値名について特定の情報をチェックする条件ステートメントを作成するには、この設定で適切なルート キーを選択してから、残りのレジストリ パスを入力します。使用可能なルート キーは次の通りです: • **HKLM**—HKEY_LOCAL_MACHINE • **HKCU**—HKEY_CURRENT_USER • **HKCR**—HKEY_CLASSES_ROOT • **HKU**—HKEY_USERS
値名	レジストリ比較条件でチェックするレジストリ値を入力します。レジストリ キーのデフォルト値をチェックする場合、この設定を空白のままに残します。
64 ビット キー	アドバンスト UI またはスイート/アドバンスト UI インストールが 64 ビット システム上のレジストリの 64 ビット部分をチェックするかどうかを指定します。True を選択すると、アドバンスト UI またはスイート/アドバンスト UI インストールは 32 ビット ターゲット システム上では、この条件ステートメントを無視します。
変換	比較する値の種類を選択します。選択可能なオプションは以下のとおりです: • **DWORD**—比較する値は DWORD (32 ビット) 値です。 • **QWORD**—比較する値は QWORD (64 ビット) 値です。 • **文字列**—比較する値は文字列値です。 • **複数文字列**—比較する値は複数文字列値です。 • **ファイル バージョン**—比較する値は *nn.nn.nn.nn* 形式のファイル バージョンです。 • **製品バージョン**—比較する値は製品バージョン値です。バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。

テーブル 13 • ″レジストリの比較″ 条件の設定 (続き.)

設定	説明
比較	ターゲット システム上の値と ″比較対象″ 設定で指定した値とをどのように比較するかを説明するオプションを選択します。たとえば、″比較″ 設定の値が [以下] で、″比較対象″ 設定が [4.1] であるレジストリ条件を考慮します。実行時に、この条件はターゲット システム上で指定されたレジストリ値データの数値が 4.0 または 4.1 の場合、True 評価します。ターゲット システム上でファイル バージョンが 4.2 の場合、False 評価します。
比較対象	ターゲット システム上にある値を比較する値を指定します。入力する値は、″変換″ 設定で選択したオプションによって異なります。 • ″変換″ 設定で [DWORD] が選択されている場合、″比較対象″ 設定に比較する DWORD 値を入力します。 • ″変換″ 設定で [QWORD] が選択されている場合、″比較対象″ 設定に比較する QWORD 値を入力します。 • ″変換″ 設定で [文字列] が選択されている場合、″比較対象″ 設定に比較する文字列値を入力します。 • ″変換″ 設定で [複数文字列] が選択されている場合、″比較対象″ 設定に比較する複数文字列値を入力します。 • ″変換″ 設定で [ファイル バージョン] が選択されている場合、″比較対象″ 設定に比較するファイルバージョン値を *nn.nn.nn.nn* 形式で入力します。

テーブル 13 • ″レジストリの比較″ 条件の設定 (続き.)

設定	説明
ファイルとの比較 (続き)	• ″変換″ 設定で [製品バージョン] が選択されている場合、″比較対象″ 設定に比較する製品バージョン値を入力します。バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 ″比較″ 設定では、ターゲット システム上の値と ″比較対象″ 設定で指定した値との比較方法を指定します。たとえば、″比較″ 設定の値が [以下] で、″比較対象″ 設定が [4.1] であるレジストリ条件を考慮します。実行時に、この条件はターゲット システム上で指定されたレジストリ値データの数値が 4.0 または 4.1 の場合、True 評価します。ターゲット システム上でファイル バージョンが 4.2 の場合、False 評価します。

″プロパティの比較″ 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、機能、またはウィザード インターフェイス条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、″プロパティの比較″ 条件を作成できます。″プロパティの比較″ 設定は、特定のビルトイン アドバンスト UI またはスイート/アドバンスト UI プロパティ、あるいは [プロパティ マネージャー] ビューで定義されたプロパティの値をチェックするために構成された条件を表示します。この種の条件を定義するには、必要に応じて ″プロパティの比較″ 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 14 • ″プロパティの比較″ 条件の設定

設定	説明
名前	特定のアドバンスト UI またはスイート/アドバンスト UI プロパティの値をチェックする条件ステートメントを作成するには、この設定にプロパティの名前を入力します。 ビルトイン条件のリストは、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。
変換	比較する値の種類を選択します。選択可能なオプションは以下のとおりです: • **DWORD**—比較する値は DWORD (32 ビット) 値です。 • **QWORD**—比較する値は QWORD (64 ビット) 値です。 • **文字列**—比較する値は文字列値です。 • **ファイル バージョン**—比較する値は *nn.nn.nn.nn* 形式のファイル バージョンです。 • **製品バージョン**—比較する値は製品バージョン値です。バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。 • **ローカル日時**—比較する値は日付または日付と時刻です。
比較	ターゲット システム上で実行時のプロパティの値と ″比較対象″ 設定で指定した値とをどのように比較するかを説明するオプションを選択します。たとえば、″変換″ 設定の値が [文字列]、″比較″ 設定の値が [等しい]、″比較対象″ 設定の値が MyValue であるプロパティの比較条件を考慮します。実行時、プロパティの値が MyValue のとき、この条件は True 評価されます。プロパティの値が設定されていない、またはその他の値のとき、この条件は False 評価されます。

テーブル 14 • ″プロパティの比較″ 条件の設定 (続き.)

設定	説明
比較対象	実行時にターゲット システムでアドバンスト UI またはスイート/アドバンスト UI プロパティの値と比較する値を指定します。入力する値は、″変換″ 設定で選択したオプションによって異なります。 • ″変換″ 設定で [DWORD] が選択されている場合、″比較対象″ 設定に比較する DWORD 値を入力します。 • ″変換″ 設定で [QWORD] が選択されている場合、″比較対象″ 設定に比較する QWORD 値を入力します。 • ″変換″ 設定で [文字列] が選択されている場合、″比較対象″ 設定に比較する文字列値を入力します。 • ″変換″ 設定で [ファイル バージョン] が選択されている場合、″比較対象″ 設定に比較するファイルバージョン値を *nn.nn.nn.nn* 形式で入力します。
ファイルとの比較 (続き)	• ″変換″ 設定で [製品バージョン] が選択されている場合、″比較対象″ 設定に比較する製品バージョン値を入力します。バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。 • ″変換″ 設定で [ローカル日時] が選択されている場合、″比較対象″ 設定に比較する日付と時刻を入力します (例、**4/26/2004 3:57:46 PM**)。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 ″比較″ 設定では、ターゲット システム上で実行時のプロパティの値と ″比較対象″ 設定で指定した値との比較方法を指定します。たとえば、″変換″ 設定の値が [文字列]、″比較″ 設定の値が [等しい]、″比較対象″ 設定の値が MyValue であるプロパティの比較条件を考慮します。実行時、プロパティの値が MyValue のとき、この条件は True 評価されます。プロパティの値が設定されていない、またはその他の値のとき、この条件は False 評価されます。

″MSI パッケージ″ 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、″MSI パッケージ″ 条件を作成できます。

″MSI パッケージ″ 設定は、ターゲット システムで特定の .msi パッケージによってインストールされた製品が存在するかどうかをチェックするように構成された条件を表示します。この種の条件を定義するには、必要に応じて ″MSI パッケージ″ 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 15 ・ ″MSI パッケージ″ 条件の設定

設定	説明
製品コード	特定の製品コードを使って、.msi パッケージでインストールされた製品の有無をチェックする条件ステートメントを作成する場合、この設定の省略記号ボタン (...) をクリックして、パッケージを参照すると、製品コードが自動設定されます。別の方法として、製品コードを手入力することもできます。たとえば、{5D607F6A-AF48-4003-AFA8-69E019A4496F} と入力します。 ***ヒント:*** *構成中のパッケージが .msi パッケージである限り、MSI パッケージの対象条件および検出条件を定義する際に、現在のパッケージの製品コード、パッケージ コード、および製品バージョンのプレースホルダーとしてアスタリスクを使用することができます。* ″スイート″ プロパティを使って、このフィールドを構成することもできます。GUID の代わりに、角括弧でプロパティを囲みます。例: [MSIProductCode]

テーブル 15 • ″MSI パッケージ″ 条件の設定 (続き.)

設定	説明
パッケージ コード	特定のパッケージ コードを使って、.msi パッケージでインストールされた製品の有無をチェックする条件ステートメントを作成するには、この設定の省略記号ボタン (...) をクリックして、パッケージを参照すると、パッケージ コードが自動設定されます。別の方法として、パッケージ コードを手入力することもできます。。たとえば、{5D607F6A-AF48-4003-AFA8-69E019A4496F} と入力します。 ***ヒント:*** *構成中のパッケージが .msi パッケージである限り、MSI パッケージの対象条件および検出条件を定義する際に、現在のパッケージの製品コード、パッケージ コード、および製品バージョンのプレースホルダーとしてアスタリスクを使用することができます。* ”スイート” プロパティを使って、このフィールドを構成することもできます。GUID の代わりに、角括弧でプロパティを囲みます。例: [MSIPackageCode]
製品バージョン	特定の製品バージョンを使って、.msi パッケージでインストールされた製品の有無をチェックする条件ステートメントを作成するには、この設定の省略記号ボタン (...) をクリックして、パッケージを参照すると、製品バージョンが自動設定されます。別の方法として、製品バージョンを手入力することもできます。。次に、″比較″ 設定を使って製品バージョンが指定するバージョンより大きい、小さい、または等しいかを指定します。 バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。aaa と bbb の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。 ***ヒント:*** *構成中のパッケージが .msi パッケージである限り、MSI パッケージの対象条件および検出条件を定義する際に、現在のパッケージの製品コード、パッケージ コード、および製品バージョンのプレースホルダーとしてアスタリスクを使用することができます。* ”スイート” プロパティを使って、このフィールドを構成することもできます。バージョン の代わりに、角括弧でプロパティを囲みます。例: [MSIVersion]
比較	″製品バージョン″ 設定でバージョン番号を指定する場合、ターゲット システム上のバージョン番号と入力したバージョン番号とを比較する方法について、適切なオプションを選択します。たとえば、ターゲット システム上のバージョンが 4.0.0 以下である場合に True 評価する条件の場合、″製品バージョン″ 設定に 4.0.0 と入力して、″比較″ 設定で [以下] を選択します。

テーブル 15 • ″MSI パッケージ″ 条件の設定 (続き.)

設定	説明
パッチ コード	.msi パッケージでインストールされた特定の製品コード用の特定のパッチ コードを持つ製品が存在するかどうかをチェックする条件ステートメントを作成するには、この設定にパッチ コードを入力してから、″製品コード″ 設定にその製品コードを入力します。たとえば、{5D607F6A-AF48-4003-AFA8-69E019A4496F} と入力します。 この設定に値を入力すると、″パッケージ コード″ 設定および ″製品バージョン″ 設定に入力された値はすべて無視されます。 ”スイート” プロパティを使って、このフィールドを構成することもできます。GUID の代わりに、角括弧でプロパティを囲みます。例: [PatchCode]

″MSI アップグレード″ 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、”MSI アップグレード” 条件を作成できます。

″MSI アップグレード″ 設定は、アップデートする製品のバージョンがターゲット システム上に存在するかどうか、またはダウングレードしてはならない製品のより新しいバージョンが存在するかどうかを確認するために構成された条件を表示します。いずれの場合も、その存在を確認する製品は .msi パッケージによってインストールされます。この種の条件を定義するには、必要に応じて ″MSI アップグレード″ 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 16 • ″MSI アップグレード″ 条件の設定

設定	説明
アップグレード コード	特定のアップグレード コードを使って、.msi パッケージでインストールされた製品の有無をチェックする条件ステートメントを作成する場合、この設定の省略記号ボタン (...) をクリックして、パッケージを参照すると、アップグレード コードが自動設定されます。別の方法として、アップグレード コードを手入力することもできます。。たとえば、{5D607F6A-AF48-4003-AFA8-69E019A4496F} と入力します。 ***ヒント:*** *構成中のパッケージが .msi パッケージである限り、MSI アップグレードの対象条件および検出条件を定義する際に、現在の製品のアップグレード コードおよび最小バージョン番号/最大バージョン番号のいずれかのプレースホルダーとしてアスタリスク (*) を使用することができます。* ″スイート″ プロパティを使って、このフィールドを構成することもできます。GUID の代わりに、角括弧でプロパティを囲みます。例: [UpgradeCode]
最小バージョン	アップグレード パッケージを実行するかどうかが判別されるときに条件ステートメントで評価する最小製品バージョンを入力するか、この設定の省略記号ボタン (...) をクリックして、パッケージを参照すると、最小バージョンが自動設定されます。バージョン番号は xxxx.xxxx.xxxx 形式です。 ***ヒント:*** *構成中のパッケージが .msi パッケージである限り、MSI アップグレードの対象条件および検出条件を定義する際に、現在の製品のアップグレード コードおよび最小バージョン番号/最大バージョン番号のいずれかのプレースホルダーとしてアスタリスク (*) を使用することができます。* ″スイート″ プロパティを使って、このフィールドを構成することもできます。バージョン の代わりに、角括弧でプロパティを囲みます。例: [MinimumVersion]
最小バージョンを含める	条件ステートメントに含める製品のバージョン範囲に最小バージョン値を含めるかどうかを示します。この設定を使って、アップグレード パッケージがサポートする、またはサポートしない製品のバージョン範囲に最小値を指定するかどうかを設定できます。

テーブル 16 • ″MSI アップグレード″ 条件の設定 (続き.)

設定	説明
最大バージョン	アップグレード パッケージを実行するかどうかが判別されるときに条件ステートメントで評価する最大製品バージョンを入力するか、この設定の省略記号ボタン (...) をクリックして、パッケージを参照すると、最大バージョンが自動設定されます。バージョン番号は xxxx.xxxx.xxxx 形式です。 ***ヒント•*** *構成中のパッケージが .msi パッケージである限り、MSI アップグレードの対象条件および検出条件を定義する際に、現在の製品のアップグレードコードおよび最小バージョン番号/最大バージョン番号のいずれかのプレースホルダーとしてアスタリスク (*) を使用することができます。* ”スイート” プロパティを使って、このフィールドを構成することもできます。バージョン の代わりに、角括弧でプロパティを囲みます。例: [MaximumVersion]
最大バージョンを含める	条件ステートメントに含める製品のバージョン範囲に最大バージョン値を含めるかどうかを示します。この設定を使って、アップグレード パッケージがサポートする、またはサポートしない製品のバージョン範囲に最大値を指定するかどうかを設定できます。

″対象パッケージ″ 条件の設定

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、”対象パッケージ” 条件を作成できます。

″対象パッケージ″ 設定は、アドバンスト UI またはスイート/アドバンスト UI インストール内の異なるパッケージまたは機能がターゲット システム上でインストール可能であるかどうかをチェックするために構成された条件を表示します。たとえば、ベース パッケージのインストールが可能である場合のみパッチ パッケージのインストールが可能である場合、[有効なパッケージ] 条件を作成して、パッチ パッケージのインストールをベース パッケージのインストールが可能であるかどうかに関連付けます。この場合、[パッケージ] ビューでパッチ パッケージを選択してから、条件を作成し、その条件内でベース パッケージのパッケージ GUID を指定します。

この種の条件を定義するには、必要に応じて ″有効なパッケージ″ 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 17 • "対象パッケージ" 条件の設定

設定	説明
パッケージ GUID	アドバンスト UI またはスイート/アドバンスト UI インストールの別のパッケージがターゲット システム上でインストール可能かどうかに基づく条件ステートメントを作成する場合、この設定にあるリストで、その片方の必須パッケージを選択します。リストには、プロジェクトのすべてのパッケージが含まれています。別の方法として、もう片方の必須パッケージのパッケージ GUID を入力することもできます。たとえば、{5D607F6A-AF48-4003-AFA8-69E019A4496F} と入力します。 たとえば、ベース製品のインストールが可能である場合のみパッチ パッケージのインストールが可能である場合、[有効なパッケージ] 条件を作成して、パッチ パッケージのインストールをベース製品のインストールが可能であるかどうかに関連付けます。このシナリオでは、[パッケージ] ビューでパッチ パッケージを選択してから、条件を作成し、その条件内でベース パッケージを選択します。 パッケージ GUID は、[パッケージ] ビューの "パッケージ GUID" 設定で識別されます。

"InstallScript パッケージ" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、"InstallScript パッケージ" 条件を作成できます。

"InstallScript パッケージ" 設定は、ターゲット システムで特定の InstallScript パッケージによってインストールされた製品が存在するかどうかをチェックするように構成された条件を表示します。この種の条件を定義するには、必要に応じて "InstallScript パッケージ" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 18 • "InstallScript パッケージ" 条件の設定

設定	説明
製品コード	特定の製品コードを使って、特定の InstallScript パッケージによってインストールされた製品の有無をチェックする条件ステートメントを作成する場合、この設定の省略記号ボタン (...) をクリックして、非圧縮 InstallScript パッケージの data1.hdr ファイルを参照すると、製品コードが自動設定されます。別の方法として、製品コードを手入力することもできます。たとえば、{5D607F6A-AF48-4003-AFA8-69E019A4496F} と入力します。
製品バージョン	InstallScript パッケージでインストールされた特定の製品バージョンの存在をチェックする条件ステートメントを作成するには、製品バージョンを入力します。次に、"比較" 設定を使って製品バージョンが指定するバージョンより大きい、小さい、または等しいかを指定します。 バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。aaa と bbb の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。
ログの存在を確認	アドバンスト UI またはスイート/アドバンスト UI インストールで、製品に登録されているアンインストールのログ ファイルがターゲット システムに存在していることを確認するかどうかを示すオプションを選択します。この設定を空白のままにしておくと、アドバンスト UI またはスイート/アドバンスト UI インストールは、この設定で [はい] が選択されたときと同じように動作します。
ユーザーごとの登録を確認	アドバンスト UI またはスイート/アドバンスト UI インストールで、ターゲット システムの HKEY_CURRENT_USER をチェックしてアンインストール情報があることを確認するかどうかを示すオプションを選択します。選択可能なオプションは以下のとおりです: • **はい**—HKEY_CURRENT_USER と HKEY_LOCAL_MACHINE の両方をチェックしてアンインストール情報を確認します。 • **いいえ**—HKEY_LOCAL_MACHINE のみをチェックしてアンインストール情報を確認します。 この設定を空白のままにしておくと、アドバンスト UI またはスイート/アドバンスト UI インストールは、この設定で [はい] が選択されたときと同じように動作します。

テーブル 18 • "InstallScript パッケージ" 条件の設定 (続き.)

設定	説明
比較	"製品バージョン" 設定でバージョン番号を指定する場合、ターゲット システム上のバージョン番号と入力したバージョン番号とを比較する方法について、適切なオプションを選択します。たとえば、ターゲット システム上のバージョンが 4.0.0 以下である場合に True 評価する条件の場合、"製品バージョン" 設定に 4.0.0 と入力して、"比較" 設定で [以下] を選択します。

"ロケール" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、"ロケール" 条件を作成できます。

"ロケール" 設定では、ターゲット システムでロケール関連の設定に一致するものを検索するために構成された条件が表示されています。この種の条件を定義するには、必要に応じて "ロケール" 設定の下にある 1 つまたは複数のサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 19 • ロケール条件の設定

設定	説明
ユーザー インターフェイス	ターゲット システムでユーザー インターフェイスのロケールを確認する条件ステートメントを作成するには、次のいずれかのフォーマットを使って、ロケール ID (LCID) を入力します。 • 十進 LCID; 例: **1033** • 16 進数 LCID (先頭に 0x を付加します); 例: **0x409** • ロケール名。Windows Vista 以降のシステムでのみ確認され、それ以前のシステムでは無視されます; 例: **en-us**、**en**、**zh-CN** LCID リストについては、MSDN の「Locale IDs Assigned by Microsoft」を参照してください。 この設定での値入力に関する詳しい情報は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する」を参照してください。
オペレーティング システム	ターゲット システムでオペレーティング システムの言語を確認する条件ステートメントを作成するには、次のいずれかのフォーマットを使って、ロケール ID (LCID) を入力します。 • 十進 LCID; 例: **1033** • 16 進数 LCID (先頭に 0x を付加します); 例: **0x409** LCID リストについては、MSDN の「Locale IDs Assigned by Microsoft」を参照してください。 この設定での値入力に関する詳しい情報は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する」を参照してください。
ANSI コード ページ	ターゲット システムに ANSI コードのページがあるかどうかを確認する条件ステートメントを作成するには、コード ページ ID を入力します。 コード ページ ID のリストは、MSDN の「Code Page Identifiers」を参照してください。 この設定での値入力に関する詳しい情報は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する」を参照してください。

"インストール済み" 条件の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、"インストール済みスイート" 条件を作成できます。

"インストール済みスイート" 設定では、アドバンスト UI またはスイート/アドバンスト UI インストールの特定のバージョンがターゲット システムにインストールされているかどうかを確認するために構成された条件が表示されます。この種の条件を定義するには、必要に応じて "インストール済みスイート" 設定の下にある 1 つまたは複数のサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 20 ・ インストール済み条件の設定

設定	説明
スイート GUID	アドバンスト UI またはスイート/アドバンスト UI インストールの特定のバージョンがターゲット システムにインストールされているかどうかを確認するための条件ステートメントを作成するには、そのインストールのスイート GUID を入力します。 ***ヒント:*** *アドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成した時、そのインストールに対して、自動的に、"インストール済みスイート" の終了条件が作成され、条件の "スイート GUID" と "バージョン" 設定でアスタリスク (*) がプロジェクト自身のスイート GUID とバージョンのプレースホルダーとして使用されます。このビルトイン条件は、アドバンスト UI またはスイート/アドバンスト UI インストールによって、より新しいバージョンのインストールが、上書きインストールされるの防ぐために追加されます。通常、デフォルト動作の変更が必要な特殊な状況を除いて、このビルトイン条件をオーバーライドすることはありません。* *詳細については、「特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」を参照してください。*

テーブル 20・インストール済み条件の設定 (続き.)

設定	説明
Version	アドバンスト UI またはスイート/アドバンスト UI インストールの特定のバージョンがターゲット システムにインストールされているかどうかを確認するための条件ステートメントを作成するには、そのバージョン番号を入力します。次に、"比較" 設定を使って製品バージョンが指定するバージョンより大きい、小さい、または等しいかを指定します。 バージョン番号の一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。aaa と bbb の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。製品のバージョンを指定するときに 4 番目のフィールド (*ddddd*) を含めることもできますが、アドバンスト UI またはスイート/アドバンスト UI インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。 ***ヒント:*** *アドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成した時、そのインストールに対して、自動的に、"インストール済みスイート" の終了条件が作成され、条件の "スイート GUID" と "バージョン" 設定でアスタリスク (*) がプロジェクト自身のスイート GUID とバージョンのプレースホルダーとして使用されます。このビルトイン条件は、アドバンスト UI またはスイート/アドバンスト UI インストールによって、より新しいバージョンのインストールが、上書きインストールされるの防ぐために追加されます。通常、デフォルト動作の変更が必要な特殊な状況を除いて、このビルトイン条件をオーバーライドすることはありません。* *詳細については、「特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」を参照してください。*

テーブル 20・インストール済み条件の設定 (続き.)

設定	説明
比較	適切なオプションを選択して、ターゲット システム上のバージョン番号と入力した番号をどう比較するかを指定します。たとえば、ターゲット システム上のバージョンが 4.0.0 以下である場合に True 評価する条件の場合、"製品バージョン" 設定に 4.0.0 と入力して、"比較" 設定で [以下] を選択します。 ***ヒント:*** *アドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成した時、そのインストールに対して、自動的に、"インストール済みスイート" の終了条件が作成され、条件の "スイート GUID" と "バージョン" 設定でアスタリスク (*) がプロジェクト自身のスイート GUID とバージョンのプレースホルダーとして使用されます。このビルトイン条件は、アドバンスト UI またはスイート/アドバンスト UI インストールによって、より新しいバージョンのインストールが、上書きインストールされるの防ぐために追加されます。通常、デフォルト動作の変更が必要な特殊な状況を除いて、このビルトイン条件をオーバーライドすることはありません。* *詳細については、「特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する」を参照してください。*

「Windows アプリ パッケージの条件" の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、スイート/アドバンスト UI プロジェクトでアクション条件に条件ステートメントをビルドするとき、"Windows アプリ パッケージ" 条件を作成できます。

「Windows アプリ パッケージ" 設定は、ターゲット システムで特定の Windows アプリが存在するかどうかをチェックするように構成された条件を表示します。この種類の条件を定義するには、"Windows アプリ パッケージ" 設定で省略記号ボタン (...) をクリックしてパッケージを参照すると、InstallShield によってサブ設定に適切な情報が入力されます。別の方法として、"Windows アプリ パッケージ" 設定の下のサブ設定を必要に応じて手入力で構成できます。

メモ: "Windows アプリ パッケージ" 設定で省略記号ボタンを使って Windows アプリ パッケージ (.appx) を参照するには、InstallShield または Standalone Build が搭載されているマシン上で Windows Vista または Windows Server 2008以降が必要です。

この条件には、以下のサブ設定があります。

テーブル 21 - 「Windows アプリ パッケージの条件」 の設定

設定	説明
名前	特定の Windows アプリの存在を確認する条件ステートメントを作成するには、その Windows アプリを識別する名前をオペレーティング システムに入力します。 適切な Windows アプリ パッケージ (.appx) を参照して、InstallShield を使ってこの設定に値を入力するには、"Windows アプリ パッケージ" 設定で省略記号ボタン (...) をクリックします。
発行者	この設定はパッケージの署名に使用された証明書の "件名" フィールドに使用される発行者を指定します。 適切な Windows アプリ パッケージ (.appx) を参照して、InstallShield を使ってこの設定に値を入力するには、"Windows アプリ パッケージ" 設定で省略記号ボタン (...) をクリックします。
Version	この設定は、パッケージのバージョン番号を示します。 適切な Windows アプリ パッケージ (.appx) を参照して、InstallShield を使ってこの設定に値を入力するには、"Windows アプリ パッケージ" 設定で省略記号ボタン (...) をクリックします。
比較	"バージョン" 設定でバージョン番号を指定する場合、ターゲット システム上のバージョン番号と入力したバージョン番号とを比較する方法について、適切なオプションを選択します。たとえば、ターゲット システム上のバージョンが 4.0.0 以下である場合に True 評価する条件の場合、"バージョン" 設定に 4.0.0 と入力して、"比較" 設定で [以下] を選択します。
プロセッサ アーキテクチャ	この設定は、このパッケージがターゲット システムで必要なアーキテクチャ (x86、x64、またはニュートラル) を識別します。 適切な Windows アプリ パッケージ (.appx) を参照して、InstallShield を使ってこの設定に値を入力するには、"Windows アプリ パッケージ" 設定で省略記号ボタン (...) をクリックします。

"UWP タイプの存在" 条件の設定

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

重要: *この条件は、Windows 10 以前のオペレーティング システムで常に False 評価されます。"Windows.ApplicationModel.FullTrustProcessLauncher " タイプ名サブ設定を使用するには、Windows 10 Anniversary Update 以降が必要です。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了または対象条件の条件ステートメントをビルドするとき、"UWP タイプの存在" 条件を作成できます。

"UWP タイプの存在" 設定は、ターゲット システムで UWP 機能が存在するかどうかをチェックするように構成された条件を表示します。たとえば、デスクトップ ブリッジの存在を確認する条件ステートメントを作成するには、Windows.ApplicationModel.FullTrustProcessLauncher タイプを確認します。これは、条件付でインストールをブロックする、または .msi および Windows アプリ パッケージ (.appx) のどちらをインストールするかを選択する場合に使用できます。

テーブル 22 ・ "UWP タイプの存在" 条件の設定

設定	説明
タイプ名	特定の UWP タイプの存在を確認する条件ステートメントを作成するには、ドロップダウン リストから Windows.ApplicationModel.FullTrustProcessLauncher を選択するか、オペレーティング システムが UWP タイプを識別するのに使用する名前を入力します。

"パッケージの操作" 条件の設定

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、"パッケージの操作" 条件を作成できます。

"パッケージの操作" 設定は、スイート/アドバンスト UI インストールの特定のパッケージのターゲット状態を確認するために構成された条件を表示します。この種類の条件は、[イベント] ビューまたは [パッケージ] ビューのイベントに関連付けられたアクションでのみ使用できます。アクションを [イベント] ビューのイベントと関連付けるとき、アクションをスケジュールできる最初のイベントは OnStaging です。

この種の条件を定義するには、必要に応じて "パッケージの操作" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 23 • "パッケージの操作" 条件の設定

設定	説明
パッケージ	スイート/アドバンスト UI インストールの特定のパッケージのターゲット操作をチェックする条件ステートメントを作成するには、この設定でパッケージの名前を選択します。
演算子	パッケージ操作の条件をチェックするための操作の種類を選択します。選択可能なオプションは以下のとおりです: • **何もしない** – パッケージは、インストール、削除、修復、変更、または修復の対象として設定されていません。 • **インストール** – パッケージは、インストールの対象として設定されています。 • **削除** – パッケージは、削除の対象として設定されています。 • **変更** – パッケージは、変更の対象として設定されています。 • **修復** – パッケージは、修復の対象として設定されています。

"機能の操作" 条件の設定

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、"機能の操作" 条件を作成できます。

"機能の操作" 設定は、スイート/アドバンスト UI インストールの特定の機能のターゲット状態を確認するために構成された条件を表示します。この種類の条件は、[イベント] ビューまたは [パッケージ] ビューのイベントに関連付けられたアクションでのみ使用できます。アクションを [イベント] ビューのイベントと関連付けるとき、アクションをスケジュールできる最初のイベントは OnStaging です。OnStaging イベントの前に機能の状態は使用できません。

この種の条件を定義するには、必要に応じて "機能の操作" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 24 • "機能の操作" 条件の設定

設定	説明
機能	スイート/アドバンスト UI インストールの特定の機能のターゲット操作をチェックする条件ステートメントを作成するには、この設定で機能の名前を選択します。
演算子	機能の操作の条件をチェックするための操作の種類を選択します。選択可能なオプションは以下のとおりです: • **何もしない** — 機能は、インストールまたは削除用に設定されません。 • **インストール** — 機能は、インストールの対象として設定されています。 • **削除** — 機能は、削除の対象として設定されています。

"拡張条件" の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、独自のカスタム条件を実装するために作成した C/C++ DLL を呼び出す拡張条件を作成できます。カスタム条件は、ターゲット システムをチェックして、ビルトイン タイプの条件チェックで評価できない要因を確認できます。DLL を作成して、プロジェクトに追加する方法については、「アドバンスト UI またはスイート/アドバンスト UI インストールのカスタム条件を作成する」を参照してください。

拡張条件 DLL を追加すると、その条件に対して新しい行が追加されます。行には、2 つのフィールドがあります。この行の左のフィールドでは、次の構文を使って条件を表記します:

拡張DLL名:条件

ここで、*拡張DLL名*は、DLL ファイルの名前ですが、.dll の部分は除きます。

この行の右のフィールドでは、条件に対して定義したパラメーターと値がペアで表示されます (読み取り専用)。この設定ではまた、パラメーターとその値を定義できる **[パラメーターの追加]** ボタンのほか、拡張条件を削除できる **[この条件を削除]** ボタンも表示されています。

パラメーターと値を追加するには、拡張条件行の右フィールドにある **[パラメーターの追加]** ボタンをクリックします。**[拡張条件の構成]** ダイアログ ボックスが開き、パラメーターの名前を入力できます。パラメーターに対して新しい行が追加されます。右側のフィールドに表示されているパラメーターの値を構成します。

"モード条件" の設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトでウィザード インターフェイス条件の条件ステートメントをビルドするとき、"モード" 条件を作成できます。"モード" 設定は、アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のモードをチェックするために構成された条件を示します。この種の条件を定義するには、必要に応じて "モード" 設定の下にあるサブ設定を構成します。

この条件には、以下のサブ設定があります。

テーブル 25 • "モード条件" の設定

設定	説明
値	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のモードをチェックする条件ステートメントを作成するには、この設定に適切なモードを選択します。選択可能なオプションは以下のとおりです: • **インストール** – インストールは初回インストール モードで実行しています。 • **ステージング** – インストールがステージングされています。つまり、インストールは圧縮解除され、特定の場所にコピーされました。インストール内のパッケージは、いずれもこのモードで実行しません。 • **削除** – 製品が削除中です。 • **変更** – 製品が変更中です。 • **修復** – 製品が修復中です。 • **メンテナンス (削除、変更、修復)** – インストールはメンテナンス モード (変更、削除、または修復) で実行中です。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで条件を定義するときのガイドライン

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

以下は、アドバンスト UI またはスイート/アドバンスト UI インストールで条件を定義するときの、いくつかのガイドラインです。

各 .exe パッケージの対象条件を定義して、ダウングレードを防ぐ

InstallShield は空白の対象条件を、パッケージのより新しいバージョンがターゲット システム上に既に存在しないことを確認するための要求として自動的に処理します。たとえば、サポートされていないプラットフォームへのインストールを阻止する [プラットフォーム] 条件など、自動の動作を保持したままでこの拡張を行うことができます。

.exe パッケージの場合、以前のバージョンがより新しいバージョンを上書きインストールできるように許可して、アドバンスト UI またはスイート/アドバンスト UI インストールが製品をダウングレードしないようにする適切な対象条件を作成します。

製品がサイドバイサイド バージョン (たとえば同じシステム上でバージョン 3 とバージョン 4) をサポートする場合、条件がこれをサポートすることを確認してください。製品がこれらのバージョンを同じシステム上でサポートしない場合、条件がこのシナリオを阻止するようにします。

要件および依存関係をチェックする各パッケージの対象条件の定義を考慮する

要件を満たさないターゲット システム上 (たとえば、最小オペレーティング システム バージョン) で、または .NET Framework、Microsoft SQL Server、あるいはサードパーティ ライブラリなどの依存関係が不足している場合、パッケージのインストールが失敗、または製品が期待通りに実行しない場合があります。アドバンスト UI またはスイート/アドバンスト UI パッケージのインストールが失敗する可能性があるターゲット システム上で、パッケージのダウンロード (適切な場合) および起動が行われないようにするには、要件および依存関係をチェックする対象条件を定義します。1 つ以上の起動条件を含むパッケージがある場合、パッケージに対応する対象条件を必ず追加してください。

.msp パッケージには、そのターゲット .msi パッケージが存在しなくてはならないため、ターゲット システム上でアップデートする製品の存在をチェックする対象条件を作成します。

デフォルトの終了条件の関連性を確認する

デフォルトで、新しいアドバンスト UI またはスイート/アドバンスト UI パッケージを作成すると、インストール対象のプライマリ パッケージがない場合にアドバンスト UI またはスイート/アドバンスト UI インストールを中止する終了条件が InstallShield によって自動的に作成されます。終了条件には [対象パッケージ] タイプの条件チェックが含まれていて、条件のパッケージ ID 設定で各プライマリ パッケージの ID にプレースホルダーとしてアスタリスク (*) が使用されます。

各パッケージの対象条件を変更することで、このデフォルト終了条件を間接的に制御することができます。

デフォルトの終了メッセージは、対象条件がダウングレードを防ぐことのみを想定しています。プロジェクトに含まれるプライマリ パッケージのいずれかに、その他の要素をチェックする対象条件が含まれている場合、デフォルト メッセージを編集してアドバンスト UI またはスイート/アドバンスト UI インストールが中止される理由を適切に反映させる必要があります。

たとえば、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに 1 つの 64 ビット プライマリ パッケージが含まれる場合、パッケージには x64 アーキテクチャをチェックする対象条件が含まれています。この場合、終了メッセージと条件を編集して、パッケージが 32 ビットのターゲット システムを対象としない事実を考慮に入れる必要があります。ただし、アドバンスト UI またはスイート/アドバンスト UI パッケージに 1 つの 64 ビット プライマリ パッケージと 1 つの 32 ビット プライマリ パッケージが含まれていて、対応するアーキテクチャが搭載されているターゲット システム上で適切なパッケージが起動されるように対象条件を構成した場合、デフォルトの終了メッセージと条件を変更する必要はないかもしれません。

各 .exe パッケージの検出条件を定義する

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる各 .exe パッケージには、少なくとも 1 つ以上の定義済み検出条件が必要です。検出条件は、パッケージがターゲット システム上に既にインストールされているかどうかを評価します。ビルド時に、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のすべてのプライマリ パッケージからの検出条件が組み合わされて、アドバンスト UI またはスイート/アドバンスト UI インストールのモード条件の一部が形成されます。

アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルは、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれる .msi パッケージと InstallScript パッケージがターゲット システム上に既にインストールされているかどうかを判断できます。また、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれる .msp パッケージが製品の以前のバージョンに適用済みかどうかも判断します。同様に、スイート/アドバンスト UI インストールの Windows アプリ パッケージ (.appx) がターゲット システム上に既にインストール済みかどうかを判断します。したがって、これらの 3 つの種類のパッケージに検出条件を手動で定義する必要はありません。

明示的であること

すべての可能な結果に対する条件を作成してください。そうすることで、適切な動作を完全に制御することができます。たとえば、ファイルまたは特定のレジストリ値の特定のバージョンをチェックする場合、ファイルまたはレジストリ エントリの存在もチェックしてください。不足しているファイルに問題がなくてもファイルの指定されたバージョン番号に問題がある場合、不足しているファイルに問題がある場合があります。

テスト、さらにテスト、さらにもう一度テスト

様々なターゲット システム上で異なる状況下で条件をテストして、アドバンスト UI またはスイート/アドバンスト UI インストールが予定通りに動作することを確認してください。クリーン マシン、製品の以前のバージョンがインストールされたマシン、および現在のバージョンがインストールされたマシンを使用します。また、製品の未来のバージョンをビルドして、現在のインストールによって未来のバージョンがダウングレードされないこともテストしてください。

特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、特定のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされているかどうかを判別するためのサポートが含まれています。この種類の条件チェックは、*インストール済みスイート条件*と呼ばれ、ターゲット システムのレジストリをチェックして、アドバンスト UI またはスイート/アドバンスト UI インストールの Uninstall キーの有無を確認します。

アドバンスト UI またはスイート/アドバンスト UI インストールで、エンドユーザーが製品のインストールに成功すると、レジストリの次の 32 ビットの場所に Uninstall レジストリ キーが作成されます:

SOFTWARE¥Microsoft¥Windows¥現在のバージョン¥Uninstall¥*{スイート GUID}*

Uninstall キーには、インストールのアンインストール情報が含まれています。レジストリ キーでは、アドバンスト UI またはスイート/アドバンスト UI インストールのスイート GUID という名前が使われています。このレジストリ キーには、DisplayVersion 値がレジストリ値の 1 つとして含まれています。この値のデータは、アドバンスト UI またはスイート/アドバンスト UI インストールのバージョン番号です。スイート GUID もバージョン番号も、プロジェクトの [一般情報] ビューで定義されます。

デフォルトで、アドバンスト UI またはスイート/アドバンスト UI インストールには、それぞれ、2 つのインストール済みスイート条件が含まれています。また、必要に応じて、プロジェクトに自作のインストール済みスイート条件を追加することもできます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、デフォルトのインストール済みスイート条件

アドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成したとき、デフォルトで、プロジェクトに 2 つのインストール済みスイート条件が作成されます。

- **終了条件**—このデフォルトの終了条件は、プロジェクトの [一般情報] ビューで表示されています。インストール済みスイートの終了条件は、エンドユーザーが、アドバンスト UI またはスイート/アドバンスト UI インストールの現在のバージョンによって、同じアドバンスト UI またはスイート/アドバンスト UI インストールの将来のバージョンが上書きインストールされるのを防ぎます。

  つまり、インストールによって、ターゲット システムの Uninstall キーの DisplayVersion 値に格納されているバージョン番号が、プロジェクトで構成された値と比較されます。ターゲット システムのレジストリ内のバージョンが、プロジェクトで構成されたバージョンよりも大きい場合、エラー メッセージが表示されます。

エンドユーザーがメッセージを閉じると、インストールが終了します。このデフォルトのインストール済みスイートの終了条件によって、アドバンスト UI またはスイート/アドバンスト UI インストールのダウングレードが防がれます。

- **インストール モード条件**—モード条件は、プロジェクト ファイル (.issuite) でのみ表示されます。この種類の条件は、InstallShield 内で編集はできません。"インストール済みスイート" の終了条件によって、エンドユーザーが、新しいバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールを、以前のバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールの上から上書きインストールしたとき、アドバンスト UI またはスイート/アドバンスト UI インストールは初回インストール モードで実行されます。また、同じバージョンのアドバンスト UI またはスイート/アドバンスト UI インストールが既にインストールされている場合、"インストール済みスイート" のモード条件によって、インストールが初回インストール モードで実行されるのが禁止されることがあります。

  デフォルトのインストール モード条件によって、ターゲット システムの Uninstall キーの DisplayVersion 値がプロジェクトで構成された値と比較されます。これらの値が等しくなかった場合、インストール モード条件の [インストール済みスイート] 部分は True として評価されます。

  インストール モード条件によって、インストール内のすべてのプライマリ パッケージの検出条件も評価されますので注意してください。したがって、インストール モード条件の [インストール済みスイート] の箇所は False と評価されたように見えても (つまり、ターゲット システムに、プロジェクトで構成されたバージョンと同じバージョンが存在するように見える場合も)、アドバンスト UI またはスイート/アドバンスト UI インストールが、初回インストール モードで実行されることがあります。これは、インストールに、True に評価されたプライマリ パッケージの検出条件がなかった場合に発生する可能性があります。

通常、デフォルト動作の変更が必要な特殊な状況を除いて、このビルトインのインストール済みスイート条件をオーバーライドすることはありません。

InstallShield では、デフォルトのインストール済みスイート条件の "スイート GUID" と "バージョン" 設定で、アスタリスク (*) が、プロジェクト自身のスイート GUID とバージョンのプレースホルダーとして使用されます。ビルド時に、アスタリスクが、作成されたビルドで、適切な値で置き換えられます。

作成した条件で、インストール済みスイートの条件チェックを使用する

アドバンスト UI またはスイート/アドバンスト UI インストールで、ターゲット システムをチェックして、インストールの特定のバージョンの有無を確認する場合、終了条件、検出条件などの条件用に、自分で定義したインストール済みスイート条件チェックを追加することができます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでロケール条件を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ロケール タイプの条件には、いくつかのサブ設定があります。次のテーブルは、各サブ設定で言語を指定するときに使用できるフォーマットの説明です。

テーブル 26 • ロケール条件の設定の下にあるサブ設定

設定	この設定の値のフォーマット	注
ユーザー インターフェイス	次のいずれかのフォーマットでロケール ID (LCID) を入力します: • 十進 LCID; 例: **1033** • 16 進数 LCID (先頭に 0x を付加します); 例: **0x409** • ロケール名。Windows Vista 以降のシステムでのみ確認され、それ以前のシステムでは無視されます; 例: **en-us**、**en**、**zh-CN** オプションで、LCID または名前の前にチルダ (~) を付加して、国や地域を無視し、言語の一致のみ確認することもできます。 たとえば、**~1033** (英語-米国) と入力した場合、オーストラリア英語、イギリス英語、米語、または他の地域の英語を使用するターゲット システムで一致の有無が確認されます。	インストールの実行時に、次の関数によって、ターゲット システムのユーザー インターフェイスが確認されます: • GetUserDefaultLCID • GetSystemDefaultLCID • GetUserDefaultLocaleName (only on Windows Vista 以降のシステムのみ) • GetSystemDefaultLocaleName (only on Windows Vista 以降のシステムのみ) LCID リストについては、MSDN の「Locale IDs Assigned by Microsoft」を参照してください。
オペレーティング システム	次のいずれかのフォーマットで、LCID を入力します: • 十進 LCID; 例: **1033** • 16 進数 LCID (先頭に 0x を付加します); 例: **0x409** オプションで、LCID の前にチルダを付加して、国や地域を無視し、言語の一致のみ確認することもできます。 たとえば、**~1033** (英語-米国) と入力した場合、オーストラリア英語、イギリス英語、米語、または他の地域の英語を使用するターゲット システムで一致の有無が確認されます。	この条件が役に立つ 1 つの例として、Windows Server 2003 の .NET Framework 1.1 サービス パックが挙げられます。これらのサービス パックは、特定のオペレーティング システム言語をターゲットします。 インストールの実行時、関数 GetSystemDefaultUILanguage によって、ターゲット システムで使用されているオペレーティング システムの言語が確認されます。 LCID リストについては、MSDN の「Locale IDs Assigned by Microsoft」を参照してください。

テーブル 26 • ロケール条件の設定の下にあるサブ設定 (続き.)

設定	この設定の値のフォーマット	注
ANSI コード ページ	コード ページ ID; 例: **932**	インストールの実行時、関数 GetACP によって、ターゲット システムの ANSI コード ページが確認されます。 このエラーの情報についてナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。コード ページ ID のリストは、MSDN の「Code Page Identifiers」を参照してください。

ターゲット システムのユーザー インターフェイス、オペレーティング システム、または ANSI コード ページをチェックして、2 つ以上の言語が使用されているかどうかを確認する場合、複数の値をカンマ (,) で区切って指定できます。設定に含まれている空白は、すべて無視されます。たとえば、英語 (米語) または日本語のユーザー インターフェイスをチェックする場合、"ユーザー インターフェイス" 設定で次の値を入力します:

0x409, 0x411

このシナリオでは、ユーザー インターフェイス条件は、ターゲット システムの UI が米語または日本語の場合に満たされます。

2 つ以上のロケール設定で値を指定した場合、条件が満たされるには、それぞれのロケール設定で、一致が検出される必要があります。たとえば、"ユーザー インターフェイス" 設定と "ANSI コード ページ" 設定で言語を指定した場合、条件は、ユーザー インターフェイス言語の 1 つが一致として検出され、ANSI コード ページの言語の 1 つが一致として検出されたとき、常に True と評価されます。

アドバンスト UI またはスイート/アドバンスト UI インストールのカスタム条件を作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで終了、検出、対象、または機能条件の条件ステートメントをビルドするとき、またはスイート/アドバンスト UI プロジェクトでアクション条件の条件ステートメントをビルドするとき、ターゲット システム上で評価するチェックを様々な異なる種類から選択できます。たとえば、オペレーティング システムのバージョン、ファイルまたはレジストリ エントリの存在、およびインストール済み製品の存在など。

ビルトイン タイプの条件チェックによって、アドバンスト UI またはスイート/アドバンスト UI インストールの実行時に、ターゲット システムで評価する必要がある状況の中が、すべてカバーされないことがあります。このため、InstallShield では、*拡張条件*と呼ばれる自作のカスタム条件を作成することができるようになっています。拡張条件によって、作成された C/C++ DLL が呼び出されます。

たとえば、IIS Web サイトの有無を確認したい場合、ターゲット システムをチェックする DLL を作成して、それをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加できます。

拡張条件 DLL の作成に関するガイダンスついては、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する」をご覧ください。

拡張条件 DLL をプロジェクトに組み込む手順ついては、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する」をご覧ください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

メモ: *Revenera では、Windows プログラミングや DLL デバッグに関する技術的なサポートは提供していません。DLL 関数は、製作者が正しく作成する責任があります。カスタム DLL 関数のプロトタイプは、以下のように作成します。戻り値の型が異なると、拡張条件 DLL が失敗する原因になります。*

拡張条件を作成する最初のステップでは、まず DLL を作成します。拡張条件 DLL は、Visual C++ の最近のバージョン、または、COM および DLL のエクスポートをサポートしているすべてのツールや言語を使って作成できます。

アドバンスト UI およびスイート/アドバンスト UI エンジンには、条件ごとに 2 つのエントリ ポイント (条件を検証するエントリ ポイントと条件を評価するエントリ ポイント) が必要です。エントリ ポイントは両方とも、拡張条件の名前がベースになっている必要があります。たとえば、**MyCondition** のいう名前の条件の場合、条件をテストする DLLによって、次のエントリ ポイントをエクスポートする必要があります:

- **MyCondition_Validate**
- **MyCondition_Evaluate**

これらの関数は、次のように定義します:

```
HRESULT _stdcall MyCondition_Validate(IDispatch *pCondition);
HRESULT _stdcall MyCondition_Evaluate(IDispatch *pCondition, VARIANT_BOOL *pbResult);
```

これらの関数をアドバンスト UI またはスイート/アドバンスト UI インストールから呼び出せるようにするには、関数を適切にエクスポートする DLL をビルドするときに、定義ファイル (.def) を含める必要があります。次は、サンプル .def ファイルのコンテンツです。LIBRARY の後の名前は、DLL に使用した名前です。

```
LIBRARY "MyConditionLibrary"
EXPORTS
    MyCondition_Validate
    MyCondition_Evaluate
```

アドバンスト UI またはスイート/アドバンスト UI インストールで、検証エントリ ポイントは、プロジェクトで定義された条件が解析されているときに呼び出されます。これにより、拡張条件で、プロジェクトで定義された条件のパラメーターが、予期された方法で評価を行う条件に適しているかどうかを確認できるようになります。このエントリ ポイントは、この種類の拡張条件が、終了条件、検出条件、またはその他の条件で見つかったとき、常に呼び出されます。アドバンスト UI またはスイート/アドバンスト UI インストールで、このエントリ ポイントから S_OK を返すと、条件が有効であるという意味に解釈されます。アドバンスト UI またはスイート/アドバンスト UI インストールで、失敗した HRESULT を返すと、最後に検証された条件がエラーとして中止されます。

アドバンスト UI またはスイート/アドバンスト UI インストールで、評価エントリ ポイントは、この種類の拡張条件がプロジェクトで構成された条件によってされたときに呼び出されます。評価では、条件が True または False であるかを評価するために必要なアクションを実行することができます。結果は、pbEvalResult パラメーターで、VARIANT_BOOL フォーマットで返されます。VARIANT_TRUE の結果は、True として扱われ、VARIANT_FALSE の結果は、False として扱われます。成功を示すステータス (S_OK) は、評価関数から返します。失敗を示すステータスによって、現時評価中の条件ブロックの評価が中止されますが、これによって、インストールが中止されることはありません。

アドバンスト UI またはスイート/アドバンスト UI インストールでは、アクションを実行することができますが、拡張条件は、拡張条件を呼び出したアドバンスト UI またはスイート/アドバンスト UI インストールの権限で実行されますので注意してください。一部のアクションでは、管理者権限が必要な場合があります (IIS 7.x 構成データの読み取りなど)。このようなケースでは、条件で必要なデータに適切にアクセスできるように、場合によって、インストールを管理者権限で実行するか、インストールに管理者マニフェストを含める必要があります。

検証および評価エントリ ポイントに渡される IDispatch インターフェイスは、ISuiteExtension インターフェイスを実装します。ISuiteExtension へのポイントを取得するには、IDispatch インターフェイスで QueryInterface メソッドを呼び出します。ISuiteExtension インターフェイスでは、条件関数を使って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトで拡張定義に定義された属性パラメーターにアクセスすることができます。プロジェクト内の条件は、それぞれの条件インスタンスに固有の異なるインターフェイス ポインターが渡されます。したがって、エントリ ポイント関数に渡されたインターフェイス ポイントを常に使用し、それを他の拡張 .dll の呼び出し用に保存しないようにしてください。

インターフェイスは、次のように定義されています。

```
interface ISuiteExtension : IDispatch {
  [propget, id(1), helpstring("Attribute")]
    HRESULT Attribute([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  [id(2), helpstring("method LogInfo")]
    HRESULT LogInfo([in]BSTR bstr);

  [propget, id(3), helpstring("Property")]
    HRESULT Property([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  [propput, id(3), helpstring("Property")]
    HRESULT Property([in]BSTR bstrName, [in]BSTR bstrValue);

  [id(4), helpstring("FormatProperty")]
```

```
    HRESULT FormatProperty([in]BSTR bstrValue, [out, retval]BSTR *bstrReturnValue);

  [id(5), helpstring("ResolveString")]
    HRESULT ResolveString([in]BSTR bstrStringId, [out, retval]BSTR *bstrReturnValue);
};
```

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、関数プロトタイプを使って、ISuiteExtension COM インターフェイス ポインターが拡張条件 DLL に渡されます。これにより、次の関数にアクセスできるようになります:

- **get_Attribute**

  HRESULT get_Attribute([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  このメソッドにより、現在の条件インスタンスの属性値が取得されます。属性名は、bstrName で渡され、対応する値は、bstrValue で返されます。

- **LogInfo**

  HRESULT LogInfo([in]BSTR bstr);

  このメソッドにより、アドバンスト UI またはスイート/アドバンスト UI デバッグ ログに必要な情報を書き込みし、それらをデバッグやその他の情報提供目的に使用できるようにします。bstr パラメーターには、ログに書きこまれる文字列が含まれています。

- **get_Property**

  HRESULT get_Property([in]BSTR bstrName, [out, retval]BSTR *bstrValue);

  このメソッドによって、現在実行中のアドバンスト UI またはスイート/アドバンスト UI インストールで定義されているプロパティの値が取得されます。定義されていないプロパティは、空の値を返します。bstrName パラメーターによって、値が取得されるプロパティの名前が指定されます。プロパティの値は、bstrValue パラメーターで返されます。

- **put_Property**

  HRESULT put_Property([in]BSTR bstrName, [in]BSTR bstrValue);

  このメソッドを使って、現在実行中のアドバンスト UI またはスイート/アドバンスト UI インストールで、新しいプロパティの値を設定したり、既存のプロパティの値を変更したりできます。空の値を渡すと、その結果として、プロパティが削除されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。bstrName パラメーターによって、設定するプロパティの名前が指定されます。

- **FormatProperty**

  HRESULT FormatProperty([in]BSTR bstrValue, [out, retval]BSTR *bstrReturnValue);

  この方法を使うと、bstrValue パラメーターで提供された文字列内に組み込まれた形式化された式を解決することができます。フォーマットされた値は、bstrReturnValue パラメーターで戻されます。形式化された式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

- **ResolveString**

  HRESULT ResolveString([in]BSTR bstrStringId, [out, retval]BSTR *bstrReturnValue);

  このメソッドは、アドバンスト UI またはスイート/アドバンスト UI の文字列 ID を、現在選択されている UI 言語で実行中されている アドバンスト UI またはスイート/アドバンスト UI インストールの対応する文字列値に解決します。bstrStringId パラメーターによって、解決する文字列 ID が指定され、解決された文字列は bstrReturnValue で返されます。指定された文字列 ID が存在しない場合、文字列は空の状態で戻されます。

作成した DLL から ISuiteExtension インターフェイスにアクセスするには、#import を使って、InstallShield と共にインストールされている SetupSuite.exe ファイルからタイプ ライブラリ情報を組み込むことができます。次は、そのパスです:

InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386¥SetupSuite.exe"

たとえば、VC++ プロジェクト内のタイプ ライブラリ (例、stdafx.h) をインポートするには、次のステートメントを使用します:

```
#import "C:¥Program Files¥InstallShield¥2021¥Redist¥Language Independent¥i386¥SetupSuite.exe" no_namespace raw_interfaces_only named_guids
```

InstallShield が別の場所にインストールされている場合、#import ステートメントのパスを、それに従って変更する必要があります。

一旦拡張条件 DLL が作成されると、それをプロジェクトに追加できます。詳しくは、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する」をご覧ください。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

一旦拡張条件 DLL が作成されると、それをプロジェクトに追加できます。

タスク ***アドバンスト UI またはスイート/アドバンスト UI プロジェクトに拡張条件 DLL を追加するには、以下の手順に従います:***

1. プロジェクト内で、作成した拡張条件 DLL を使用する検出条件、対照条件、またはその他の条件の設定を見つけます。

   これらの条件の設定についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。

2. 必要に応じて、条件グループに、**Any**、**All**、または **None** 演算子を追加します。
3. **"Any"**、**"All"**、または **"None"** 設定で、**[新しい条件]** ボタンをクリックして、**[拡張条件 DLL を参照する...]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
4. 作成した拡張条件 DLL を選択して、**[開く]** をクリックします。**[拡張条件の構成]** ダイアログ ボックスが開きます。

DLL は必ず、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、拡張条件 DLL を作成する」内の手順に従って作成してください。

5. 使用する条件をクリックします。InstallShield によって DLL がサポート ファイルとしてプロジェクトに追加されます。条件に対して新しい行が追加されます。行には、2 つのフィールドがあります。この行の左のフィールドでは、次の構文を使って条件を表記します:

   拡張DLL名:条件

   ここで、*拡張DLL名*は、DLL ファイルの名前ですが、.dll の部分は除きます。

   この行の右のフィールドでは、条件に対して定義したパラメーターと値がペアで表示されます (読み取り専用)。この設定ではまた、パラメーターとその値を定義できる **[パラメーターの追加]** ボタンのほか、拡張条件を削除できる **[この条件を削除]** ボタンも表示されています。

6. パラメーターと値を追加するには、拡張条件行の右フィールドにある **[パラメーターの追加]** ボタンをクリックします。**[拡張条件の構成]** ダイアログ ボックスが開きます。

7. パラメーターの名前を入力してから、**[OK]** をクリックします。パラメーターに対して新しい行が追加されます。右側のフィールドに表示されているパラメーターの値を構成します。

インストールされたデータの検索

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield は、実行時にターゲット システムにインストールされているデータを検索する Windows Installer の機能をサポートしています。検索結果によって、インストールはプロパティ値を格納したり、またインストール条件でプロパティを利用したりすることもできます。インストールが処理することのできるシステム検索の例には、次のものが含まれます。

- 実行時にターゲット システム上でファイルまたはフォルダーを検索し、それらが見つかった場合、そのファイルまたはフォルダーへの完全パスを格納します。これには XML ファイルのサポートも含まれます。
- レジストリ、またはターゲット システムの WindowsFolder ディレクトリにある .ini ファイルからデータを読み出し、プロパティにそのデータを格納します。そしてインストール条件でそのプロパティを利用します。

InstallShield には、プロジェクトへ追加することが可能ないくつかの定義済みシステム検索が含まれています。さらに、あるプロジェクトで定義され、リポジトリへパブリッシュされたシステム検索は、別のプロジェクトで利用することもできます。InstallShield に含まれる検索、またはレポジトリに格納されている検索のどちらの場合でも、[システム検索] ビューを利用して、定義済みシステム検索をプロジェクトに追加します。また、システム検索ビューを利用して定義済み検索をカスタマイズしたり、インストールのシステム検索を独自に定義したりすることもできます。

このセクションには、次のトピックが含まれています:

- 定義済みシステム検索の追加
- システム検索の追加
- システム検索の変更
- 実行時にレジストリを検索する場合の特別考慮
- システム検索をリポジトリにパブリッシュする
- システム検索を削除する
- XML データの検索

定義済みシステム検索の追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield に含まれる検索、またはレポジトリに格納されている検索のどちらの場合でも、[システム検索] ビューを利用して、定義済みシステム検索をプロジェクトに追加します。

タスク ***定義済みシステム検索を追加するには、次の手順を実行します。***

1. **[システム検索]** ビューを開きます。
2. グリッドを右クリックしてから、**[定義済み検索の追加]** をクリックします。[定義済み検索の追加] ダイアログ ボックス が開きます。
3. レポジトリにパブリッシュされているシステム検索を表示するには、**"レポジトリの検索を表示"** チェックボックスを選択します。
4. 追加するシステム検索の値を選択します。
5. **[OK]** をクリックします。

InstallShield が [システム検索] ビューのグリッドへ検索を追加します。定義済み検索をプロジェクトへ追加した後に変更する必要がある場合、その手順についてシステム検索の変更を参考にして下さい。

システム検索の追加

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

インストールには、ターゲット システムにインストールされた異なるデータ型に対するシステム検索を含めることが可能です。プロジェクトへ InstallShield で利用可能な定義済みシステム検索のひとつ、またはリポジトリに格納されているシステム検索を追加するには、 定義済みシステム検索の追加を参照してください。独自のカスタムシステム検索を定義するには、次の手順を実行します。

タスク ***システム検索を追加するには、次の手順を実行します。***

1. **[システム検索]** ビューを開きます。
2. グリッドを右クリックして、**[追加]** をクリックします。**システム検索ウィザード**が開きます。
3. システム検索ウィザードのパネルを完成します。

InstallShield が [**システム検索**] ビューのグリッドへ検索を追加します。

システム検索の変更

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***システム検索を変更するには、次の手順を実行します。***

1. **[システム検索]** ビューを開きます。
2. 変更するシステム検索のグリッドエントリを右クリックしてから、**[変更]** をクリックします。**システム検索ウィザード**が開きます。
3. システム検索ウィザードのパネルを完成します。

実行時にレジストリを検索する場合の特別考慮

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム検索ウィザードと [システム検索] ビューを使って、次の種類のレジストリ関連システム検索を構成することができます:

- ファイル パス(レジストリ エントリで指定)
- フォルダー パス(レジストリ エントリで指定)
- レジストリ エントリ

これらの種類のシステム検索について以下の点にご注意ください:

レジストリ キーの存在を検索する

Windows Installer には、特定のレジストリ キーの存在をターゲット システム上で検索するビルトイン サポートが含まれていません。ただし、特定のレジストリ キーのデフォルト値を検索するシステム検索を作成することはできます。デフォルト値が存在する場合、Windows Installer がその値をシステム検索に関連付けられたプロパティに書き込みます。システム検索がデフォルト値を検出できなかった場合、Windows Installer はプロパティを未定義のままに残します。

メモ: *システム検索の適切な機能を確保するためには、特定のレジストリ キーの既定値に、空でない／空白でない値を入力することが必須です。*

システム検索のプロパティにレジストリ エントリの種類を保管する

実行時、Windows Installer はレジストリ関連のシステム検索に指定したプロパティ値をターゲット システム上で見つかった場所に設定します。さらに、Windows Installer はプロパティ値にプレフィックスを追加して、レジストリ エントリの種類を示します (たとえば、REG_DWORD または REG_BINARY)。その後プロパティを使ってレジストリを構成するか、このプロパティを読み取る条件を作成する場合、Windows Installer がプロパティを設定するために使用する構文について把握しておく必要があります。

次の表は、レジストリ エントリの各種類、および Windows Installer がシステム検索プロパティを設定するときに使用する関連プレフィックスについて説明します。

テーブル 1 • Windows Installer がシステム検索プロパティの値に使用するプレフィックス、またはその他の表記

プレフィックス/表記	意味
<プレフィックスなし>	プロパティ値には文字列値 (REG_SZ) が含まれます。
# **#+** **#–**	プロパティ値には整数値 (REG_DWORD) が含まれます。
#x	プロパティ値には 16 進数値 (REG_BINARY) が含まれます。 Windows Installer は、各16 進数の値をこのプロパティ値の ASCII 文字として変換します。
#%	プロパティ値には拡張可能な文字列値 (REG_EXPAND_SZ) が含まれます。
<ヌル>	プロパティ値には、文字列リスト (REG_MULTI_SZ) が含まれます。この値は、ヌル文字で開始および終了します。

Searching for Existing Applications, Files, Registry Entries or .ini File Entries (Windows Installer ヘルプ ライブラリ)

システム検索をリポジトリにパブリッシュする

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクトに別のプロジェクトで再利用、または他のユーザーと共有したいシステム検索がある場合、リポジトリにそれをパブリッシュすることができます。

タスク ***システム検索をリポジトリにパブリッシュするには、次の手順を実行します。***

1. **[システム検索]** ビューを開きます。
2. グリッドでシステム検索を右クリックしてから、**[パブリッシュ ウィザード]** をクリックします。**パブリッシュ ウィザード**が開きます。
3. パブリッシュ ウィザードのパネルを完成します。

リポジトリからプロジェクトへシステム検索をインポートした後、現在のシステム検索と既存のリポジトリ システム検索とは関連性を持ちません。システム検索を変更してリポジトリへ再パブリッシュしても、インポート先プロジェクト内のシステム検索には影響しません。但し、リポジトリからプロジェクトへシステム検索を再インポートすることができます。

システム検索を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

タスク ***システム検索を削除するには、次の手順を実行します。***

1. **[システム検索]** ビューを開きます。
2. 削除するシステム検索のグリッドエントリを右クリックしてから、**[削除]** をクリックします。

削除されたシステム検索は、今後インストールには含まれません。

XML データの検索

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

アプリケーションと構成の設定は、XML ファイルに格納されることがあります。[システム検索] ビューでは、実行時にターゲット システムに存在する可能性がある XML ファイル内のデータの検索を定義します。特定の属性、特定の要素、または特定の要素コンテンツ文字列を検索することができます。結果はプロパティに格納されます。インストール条件にあるこのプロパティを、インストールで使用するコンポーネントのインストール先として、またはインストールの他の箇所で使用することができます。

例

このセクションでは、実行時にターゲット システムで XML データの検索をどう行うかが説明されています。この例では、**MyProduct** という名前の製品に、**MyProduct.exe** という名前の実行可能ファイルがあります。この .exe ファイルへのパスは、実行時に Configuration.xml という名前のファイルに書き込まれます。

ヒント: *実行時にプロパティを利用して、XML ファイルを変更するときのその方法については、「Windows Installer のプロパティを使用して、XML ファイルを動的に変更する」をご覧ください。*

次は、Configuration.xml ファイルからのサンプル コードです。

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <startup>
    <requiredRuntime version="1.0.0" safemode="true" />
  </startup>
  <appSettings>
    <add value="MyProduct" path="C:¥Program Files¥MyCompany¥MyProduct¥MyProduct.exe"/>
  </appSettings>
</configuration>
```

上記 XML コードの **path** 属性の値は、エンドユーザー が **MyProduct** をどの場所にインストールするかによって異なります。

この例の主な目的は、**path** 属性の値を検索して、その値をプロパティに格納することです。

タスク ***XML データのシステム検索を構成するには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー リストで、**[システム検索]** をクリックします。
2. グリッドを右クリックして、**[追加]** をクリックします。システム検索ウィザード が開きます。
3. 次のウィザード パネルを完成します。
    a. **[検索する対象を指定してください]** パネル: リストで、**[XML ファイルの値]** を選択します。
    b. **[ファイルの詳細の指定]** パネル:
        - **[プロパティ名]** ボックスで、次のように入力します。

          Configuration.xml
        - **[検索場所]** 領域で、**[完全パス]** を選択して、次のパスを入力します。

          [ProgramFilesFolder]MyCompany¥MyProduct
    c. **[データの指定]** パネル:
        - **[XML 要素への XPath]** ボックスで、次の XPath 式を入力します。

          /configuration/appSettings/add[@value='MyProduct']
        - **[検索]** 領域で、**[この要素の属性値]** を選択して、**[属性名]** ボックスで次を入力します。

          パス
    d. **[この値の処理方法を指定してください]** パネル:

- **[値を保存するプロパティ]** ボックスで、次を入力します。

  PATH_IN_XML_FILE

- **[追加オプション]** 領域で、**[プロパティに値を保存する]** を選択します。

検索プロパティの値は、プロジェクトの他の領域で使用することができます。たとえば、インストールのダイアログでプロパティの値を表示するには、テキスト コントロールをダイアログに追加するか、またはコントロールの Text 値を次のように設定します。

PATH_IN_XML_FILE = [PATH_IN_XML_FILE]

XML ファイルの path 属性の値が **C:¥Program Files¥MyCompany¥MyProduct¥MyProduct.exe** の場合、次のテキストがダイアログで表示されます。

PATH_IN_XML_FILE = C:¥Program Files¥MyCompany¥MyProduct¥MyProduct.exe

インストールおよびプロジェクトのテーブルを編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

重要: *InstallShield の専用ビューを使わずに、インストールとプロジェクト テーブルを直接編集する方法は、Windows Installer データベース形式について詳しい上級ユーザー向けです。*

InstallShield でビルドされる Windows Installer データベースは、テーブルで構成されます。InstallShield プロジェクト ファイル (.ism ファイル) は、Windows Installer テーブル形式を使用します。InstallShield では、Windows Installer および .ism テーブルのコンテンツで作業を行うために用意されているグラフィック ユーザー インターフェイスに加え、[ダイレクトエディター] ビューを使用してテーブルを直接編集できます。

ダイレクト エディターは、異なる種類のテーブルをサポートします:

- 標準テーブル。Windows Installer によって定義されます。
- InstallShieldテーブル。インストールに機能性を追加するために InstallShieldによって作成されます。
- カスタム テーブル。ユーザーが作成し、カスタム アクションで読み込み/書き込みを行うことができます。

ダイレクト エディターは 2 つの異なるモードで実行できます:

- **プロジェクト編集モード** – プロジェクト ファイル (.ism) に含まれるテーブルを編集できるモード。ダイレクト エディターで追加された変更は、InstallShield がビルド時に作成する Windows Installer パッケージに組み込まれます。

  Windows Installer ベースのプロジェクト タイプである、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、プロジェクト編集モードを使用します。

- **ダイレクト編集モード** – Windows Installer データベース (.msi、.msm、または .mst ファイル) のテーブルを編集できるモード。ダイレクト エディターで追加した変更を保存するとき、InstallShield が Windows Installer データベースを更新します。

  Windows Installer ベースのプロジェクト タイプである、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、ダイレクト編集モードを使用します。

このセクションには、次のトピックが含まれています:

- プロジェクト テーブルを編集する
- ダイレクト エディターで新しいテーブルを追加する

- テーブルへレコードを追加する
- ダイレクト エディターでの検索と置換
- テーブル レコードを編集する
- テーブルのフィールド編集
- ダイレクト エディターからテーブルをエクスポートする
- ダイレクト エディターを使ってテーブルをインポートする
- テーブルからレコードを削除する
- ダイレクト エディターでテーブルを削除する
- Windows Installer データベースとプロジェクト ファイルのインプレース差分比較
- テーブルへバイナリデータを入力する
- 親のないディレクトリ エントリ

プロジェクト テーブルを編集する

プロジェクト: *次のプロジェクト タイプのダイレクト エディターでは、プロジェクト ファイル (.ism) 内のテーブルを参照および編集できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *QuickPatch*

InstallScript および InstallScript オブジェクト プロジェクトでは、ダイレクト エディターで、.ism ファイルに含まれるすべてのテーブルを参照できますが、これらのプロジェクト タイプでは、InstallShield のその他のビューを使って変更を行うことが推奨されます。

InstallShield プロジェクト ファイル (.ism ファイル) は、Windows Installer テーブル形式を使用します。ダイレクト エディターは InstallShield のビューで、プロジェクト ファイル (.ism) 内の全てのテーブルを参照できます。このビューでは、Windows Installer データベース フォーマットに詳しい上級開発者向けに、高度な機能も提供されています。たとえば、ダイレクト エディターを使って以下のタスクを行います:

- プロジェクト ファイル (.ism) 内のすべてのテーブルを参照する。
- テーブルにレコードを追加/削除する。
- レコードまたはフィールドを切り取り、コピー、および貼り付ける。
- テーブル内の個別のフィールドを編集する。
- カスタム テーブルを追加する。
- 表示されているテーブル レコードをフィルターして、特定の文字列を含むプロパティを非表示にする。
- 1 つのテーブル、またはすべてのテーブルで特定の文字列を検索して、必要に応じてそれを置換する。

- テーブル内の列のサイズを変更、およびその順序を変更する。

ダイレクト エディターで新しいテーブルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

InstallShield を使って、プロジェクト ファイル (.ism)、または Windows Installer データベース (.msi、.msm、または .mst) にテーブルを追加できます。

タスク ***ダイレクト エディターを使って新しいテーブルを追加するには、以下の手順を実行します:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーを右クリックして、**[テーブルの追加]** をクリックします。**テーブル作成ウィザード**が開きます。.
3. ウィザードのすべてのパネルを完成します。

Windows ロゴ: *Windows ロゴ プログラムでは、カスタム テーブルの名前に MSI (大文字、小文字、または両方の組み合わせ) を使用しないことを要求しています。「MSI」で始まる名前は、今後の新しい標準プロパティおよびテーブル用に予約されています。*

テーブルへレコードを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*

- *トランスフォーム*

[ダイレクト エディター] ビューでは、プロジェクトまたはデータベースのテーブルにレコード (または行) を追加できます。

タスク ***レコード (行) をテーブルに追加するには、以下の手順を実行します:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. レコードを追加するテーブルを選択します。
3. **[新しいレコード]** ボタンをクリックします。[**テーブルにレコードを追加**] ダイアログ ボックスが開きます。
4. 各フィールドを必要に応じて完成させます。
5. **[OK]** をクリックします。

ダイレクト エディターでの検索と置換

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ダイレクト エディターは、標準の検索/置換機能をサポートします。この機能は、プロジェクトまたはデータベース内で特定のデータまたは文字列を検索して、改訂済みのデータまたは文字列と置換するときに便利です。

ダイレクト エディターでデータを検索する

タスク ***ダイレクト エディターでデータを検索するには、以下の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、検索を行うテーブルの 1 つを選択します。
3. **[文字列の検索]** ボタンをクリックします。**[検索]** ダイアログ ボックスが開きます。
4. **[検索する文字列]** ボックスで、検索する文字列を入力します。
5. 必要に応じて、その他のオプションを選択します。

   選択したテーブルのみで検索を行うには、**[現在のテーブルのみを検索]** チェック ボックスを選択します。ダイレクト エディターの全てのテーブルで検索を行うには、このチェック ボックスをクリアします。

6. **[次を検索]** をクリックします。

InstallShield が、指定された文字列の最初のインスタンスを検出します。

ダイレクト エディターでデータを検索/置換する

タスク ***ダイレクト エディターでデータを検索/置換するには、以下の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、データを検索/置換するテーブルを選択します。
3. **[検索/置換]** ボタンをクリックします。**[置換]** ダイアログ ボックスが開きます。
4. **[検索する文字列]** ボックスで、置換するテキストを入力します。
5. **[置換後の文字列]** ボックスで、新しい文字列を入力します。
6. 必要に応じて、その他のオプションを選択します。
7. **[次を検索]**、**[置換]**、または **[すべて置換]** をクリックします。

InstallShield が必要に応じて文字列を置換します。

テーブル レコードを編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

[ダイレクト エディター] ビューでは、プロジェクトまたはデータベースのテーブルでレコードまたは行を編集できます。

タスク ***ダイレクト エディターでテーブルのレコードを編集するには、以下の手順を実行します:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、編集するレコードを含むテーブルを選択してから、行を選択します。
3. **[選択したレコードを編集]** ボタンをクリックします。**[テーブル内のレコードを編集]** ダイアログ ボックスが表示されます。このダイアログ ボックスで、選択した行の任意のデータを編集できます。
4. 必要に応じてレコードを編集します。

5. [OK] をクリックします。

メモ- ダイレクト エディターは、不適切なデータ タイプの使用を避けるために、フィールド レベルでの検証を行います。

*ダイレクトエディターでのテーブル編集中に参照の整合性を保つには、[オプション] ダイアログの [プリファレンス] タブにある **[参照の整合性を保つ]** を選択します。[オプション] ダイアログ ボックスにアクセスするには、[ツール] メニューから [オプション] を選択します。このチェック ボックスはデフォルトで選択されます。*

テーブルのフィールド編集

***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

[ダイレクト エディター] ビューには、プロジェクトまたはデータベースで文字列その他のデータを編集できる、スプレッドシート形式のテーブルが含まれています。

タスク** **ダイレクト エディターでテーブルのフィールドを編集するには、以下の手順を実行します:

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. 編集するフィールドが含まれたテーブルを選択します。
3. 以下のいずれかを実行します:
   - テーブル セル内のテキストをすべて上書きするには、テーブル セルをクリックしてから、新しい値を入力します。
   - テーブル セル内の特定の位置にカーソルを配置するには、その場所をダブルクリックします。次に、変更を入力します。
   - フィールドが他のテーブルへの外部キーを持つ場合、セル内のリストから適切な外部キーを選択できます。

***ヒント-** 各列のヘッダーに括弧付きで挿入されている表示は、列に入力可能なデータの種類とサイズを示します。たとえば、S255 は文字列の制限文字数が 255 文字であることを示し、I2 は、2 バイトの整数を示します。*

メモ: *ダイレクト エディターは、不適切なデータ タイプの使用を避けるために、フィールド レベルでの検証を行います。*

ダイレクトエディターでのテーブル編集中に参照の整合性を保つには、[オプション] ダイアログの [プリファレンス] タブにある ***[参照の整合性を保つ]*** *を選択します。[オプション] ダイアログ ボックスにアクセスするには、[ツール] メニューから [オプション] を選択します。このチェック ボックスはデフォルトで選択されます。*

ダイレクト エディターからテーブルをエクスポートする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ダイレクト エディターを使って、1 または複数のテーブルを .idt ファイルとしてエクスポートできます。その後、ダイレクト エディターを通して .idt ファイルを別のプロジェクトまたはデータベースにインポートできます。

タスク ***ダイレクト エディターから 1 つまた複数のテーブルをエクスポートするには、以下の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、エクスポートを行うテーブルの 1 つを右クリックしてから、**[テーブルのエクスポート]** をクリックします。**[テーブルのエクスポート]** ダイアログ ボックスが開きます。
3. **[出力ディレクトリ]** ボックスに、.idt ファイルを保存するディレクトリを指定するか、**[参照]** ボタンをクリックして、保存場所を指定します。
4. **[テーブル]** ボックスに、エクスポートするテーブルのチェック ボックスを選択します。
5. **[OK]** をクリックします。

エクスポートが選択された各テーブルごとに、個別の .idt ファイルが作成されます。

ダイレクト エディターを使ってテーブルをインポートする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ダイレクト エディターを使って、テーブル (.idt ファイル) を InstallShield プロジェクト ファイル (.ism) または Windows Installer データベース (.msi、.msm、または .mst) にインポートできます。

タスク ***ダイレクト エディターを使ってテーブル (.idt ファイル) をインポートするには、以下の手順を実行します:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーを右クリックして、**[テーブルのインポート]** を選択します。
3. インポートするテーブルを選択します。
4. **[開く]** をクリックします。

プロジェクトまたはデータベースにテーブルがインポートされます。

テーブルからレコードを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ダイレクト エディターを使って、InstallShield プロジェクト ファイル (.ism) または Windows Installer データベース (.msi、.msm、または .mst) からレコードを削除できます。

タスク ***テーブルからレコード (行) を削除するには、以下の手順を実行します:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、削除するレコードを含むテーブルを選択します。
3. テーブル グリッドで、削除するレコードを選択します。

4. **[選択したレコードを削除する]** ボタンをクリックするか、または DELETE キーを押します。

ダイレクト エディターでテーブルを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ダイレクト エディターを使って、InstallShield プロジェクト ファイル (.ism) または Windows Installer データベース (.msi、.msm、または .mst) からカスタム テーブルを削除できます。

タスク ***カスタム テーブルを削除するには、以下の手順に従います:***

1. **[追加ツール]** の下のビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーで、削除するカスタム テーブルを右クリックしてから、**[テーブルの削除]** をクリックします。

メモ: *削除できるのはユーザー定義のテーブルだけです。InstallShield テーブルおよび標準テーブルは削除できません。*

Windows Installer データベースとプロジェクト ファイルのインプレース差分比較

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

InstallShield には、比較を行うために、ダイレクト エディターで 2 つの .msi パッケージを開くことができる .msi 差分比較機能があります。トランスフォーム (.mst) を編集している場合、基本になるパッケージと適用されたトランスフォームの間の差分がダイレクト エディターで強調表示されます。

ベースパッケージは InstallShield インターフェイスで開いた現在のファイルです。比較されたファイルはシステムにあり、ベースパッケージと比較するものです。従って、Direct Editorのテーブルの前に赤い X印がある場合は、テーブルがベースファイルにあり、比較ファイルには存在ないことを示します。

インプレース差分比較機能は、.msi パッケージとトランスフォームに限定されていません。.msm、.ism、および .pcp ファイルといった、別のファイルを比較することもできます。

ダイレクトエディターでは差分を元に戻したり、受け入れることができます。ダイレクトエディター テーブルの最初の列のアイコンは、その行が追加または削除されたか、または変更が加えられているかを表します。比較ファイルは、テーブルで追加または削除された列も表示します。

テーブル 1・ダイレクト エディター テーブルのアイコン

アイコン	Description
	行に変更が加えられた
	行が追加された
	行が削除された

加えられた特定の変更を表示するには、列の上にマウスを置くと差分を説明するツールヒントが表示されます。

タスク

2 つのプロジェクトまたはデータベースを比較するには、以下の手順に従います:

1. ベース プロジェクト ファイル (.ism) またはデータベース (.msi、.msm、.mst、または .pcp) を開きます。
2. **[ツール]** メニューで、**[差分]** をポイントし、**[ファイルとの比較]** をクリックします。**[比較するファイルの選択]** ダイアログ ボックスが開きます。
3. **[ファイルの種類]** ボックスで、ベース ファイルと比較するファイルの種類を選択してから、それを参照および選択します。
4. **[開く]** をクリックします。

ダイレクト エディターに、2 つのファイル間の差分が表示されます。

タスク

比較セッションを終了するには、以下の手順に従います:

[ツール] メニューで、**[差分]** をポイントし、**[比較の終了]** をクリックします。

ダイレクト エディターからインプレース差分が全て削除されます。

テーブルへバイナリデータを入力する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

Windows Installer ベースのプロジェクト タイプである、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、プロジェクト編集モードを使用します。プロジェクト編集モードの場合、Binary、Icon、および Patch といったテーブルは、バイナリ データを格納しません。これらのテーブルは、ビルド ソース パスへのリンクを格納します。このため、ダイレクト エディターを使ってこれらのテーブルを編集するときは、必ず完全修飾パスとファイル名を含めるようにしてください。そうでない場合、InstallShield がビルドするリリースにそのファイルが含まれません。

Windows Installer ベースのプロジェクト タイプである、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、ダイレクト編集モードを使用します。これらのプロジェクト タイプの場合、Binary、Icon、および Patch といったテーブルは、バイナリ データを格納します。ダイレクト エディターを使ってこれらのテーブルを編集する場合、テーブルにストリームするファイルを参照および選択することができます。

親のないディレクトリ エントリ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ディレクトリ エントリは、主にダイレクト エディターで、不適切な変更が行われた場合に親を失います。これは Directory テーブルでパスが壊れたときに発生します。

たとえば、INSTALLDIR は次のように 3 つのレコードが組み合わさって完全パスを構成しています。

ProgramFilesFolder¥ISYourCompanyDir¥ISYourProductDir

これは通常、次のように解決されます:

C:¥Program Files¥Your Company Name¥Your Product Name

ProgramFilesFolder¥ISYourCompanyDir¥ISYourProductDir のうちいずれかのレコードが不適切に変更されると、INSTALLDIR やこの不正なレコードを含んでいるディレクトリパスは、正しく解決されなくなります。この例では、Directory レコードを (Directory の列で) ISYourCompanyDir から MyCompany に変更し、ISYourProductDir レコードの Directory_Parent を MyCompany に更新しなかった場合、INSTALLDIR パスは壊れます。

InstallShield ツールの使用

InstallShield には、Windows Installer パッケージおよび InstallScript インストールにおける複雑な問題をトラブルシュートするのに役立つ様々なツールが搭載されています。

このセクションには、次のトピックが含まれています:

- InstallShield MSI ツール
- InstallShield キャビネット & ログ ファイル ビューアー
- InstallShield MSIX バンドル ユーティリティ

InstallShield MSI ツール

エディション: *これらの InstallShield MSI ツールは、InstallShield の Premier および Professional Edition で提供されています。Premier Edition には、別のマシンに InstallShield 以外のツールのみをインストールできる、個別のインストールおよび追加ライセンスが含まれています。詳しい使用条件については、InstallShield MSI ツールの使用許諾契約書を参照してください。*

InstallShield には、Windows Installer パッケージにまつわる複雑な問題を解決するのに役立つツールが含まれています:

- InstallShield MSI Diff を使って、2 つの .msi、.msm、または .pcp データベース ファイルを素早く比較できます。これを使って、1 つの .msi ファイルに対して 1 つまたは複数の .msp と .mst ファイルを適用し、その結果となる .msi データベースの変更を確認することができます。また、このツールを使って、バイナリ形式で保存された 2 つの InstallShield プロジェクト ファイル (.ism または .ise) を比較することができます。このツールでは、追加、変更、削除、スキーマの違いを示すためにカラー コードが使用されます。InstallShield MSI Diff は、ほとんどのソース コード管理システムに簡単に統合できます。
- InstallShield MSI Query を使って、SQL ステートメントをビルド スクリプトで実行する前に、SQL の Windows Installer バージョンを用いてこれをテストします。SQL ステートメントが正しくフォーマットされているかを素早く確認でき、また生成される結果を参照できます。
- InstallShield MSI Sleuth は、ターゲット システムの現在のインストール済み状態を参照できる診断ツールです。InstallShield MSI Sleuth はインストール済みのすべての .msi パッケージの一覧を表示します。リストの中から任意の .msi パッケージをクリックすると、データベース内のテーブルやバイナリ ストリームだけでなく、その機能とコンポーネントの状態および既知のソースの場所を参照できます。このツールは、特定のコンポーネント コードを持つパッケージを含むインストール済みの製品 (複数可) を識別するのにも役立ちます。
- InstallShield MSI Grep は、.msi と .msm パッケージのコレクションの中から特定のテキストを検索します。特定のテーブルまたは列に関する結果のみを表示するための詳細検索も可能です。

これらすべてのツールは、Windows [スタート] メニューの [InstallShield ツール] サブフォルダーから起動できます。

メモ: *これらのツールでは、すべて .NET Framework 2.0 がインストールされている必要があります。*

InstallShield MSI Diff

InstallShield MSI Diff を使って、2 つの .msi、.msm、または .pcp ファイルを素早く比較できます。これを使って、1 つの .msi ファイルに対して 1 つまたは複数の .msp と .mst ファイルを適用し、その結果となる .msi データベースの変更を確認することができます。また、このツールを使って、バイナリ形式で保存された 2 つの InstallShield プロジェクト ファイル (.ism または .ise) を比較することができます。このツールは、ほとんどのソース コード管理システムに簡単に統合できます。

このツールでは、追加、変更、削除、スキーマの違いを示すためにカラー コードが使用されます。InstallShield MSI Diff の右下にあるカラー コードは、それぞれの違いが何色で示されているかを示す凡例です。

InstallShield MSI Diff は、Windows [スタート] メニューの InstallShield Tools サブフォルダーから起動します。このツールは、[ツール] メニューで [差分] をポイントして InstallShield MSI Diff をクリックして、InstallShield 内からも起動できます。

ヒント: *各データベース テーブルの変更されていない行を表示または非表示にするには、[表示] メニューで、[変更されていない行] をクリックします。変更されていない行を表示するとき、このメニュー コマンドにチェック マークが付いています。変更されていない行を表示しないときは、チェック マークがありません。*

変更されていないデータベース テーブルを表示または非表示にするには、[表示] メニューで、[変更されていないテーブル] をクリックします。変更されていないテーブルを表示するとき、このメニュー コマンドにチェック マークが付いています。変更されていないテーブルを表示しないときは、チェック マークがありません。

トランスフォーム (.mst ファイル) によって行われる変更を判別する

InstallShield MSI Diff を利用して、トランスフォーム (.mst ファイル) が適用されたときに .msi パッケージがどう変更されるかを確認できます。

タスク ***トランスフォーム (.mst ファイル) によって行われる変更を判別するには、以下の手順に従います:***

1. InstallShield MSI Diff を開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. .msi パッケージを選択して、**[開く]** をクリックします。InstallShield MSI Diff で、.msi データベースが開きます。
4. **[ファイル]** メニューで **[トランスフォーム]** をポイントし、**[トランスフォームの適用]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. .mst パッケージを選択して、**[開く]** をクリックします。トランスフォームが、開いている .msi データベースに適用されます。
6. 変更を確認します。

一連のトランスフォームを .msi パッケージに適用する場合、それぞれのトランスフォームに対して、ステップ 4 と 5 を繰り返します。

ヒント: *InstallShield MSI Diff では、一連の .msp ファイルと .mst ファイルを .msi ファイルに適用し、結果の .msi データベースに適用された変更を確認することもできます。*

パッチを .msi パッケージに適用する方法については、「パッチによる実行時の影響を判別する」をご覧ください。

パッチが .msi パッケージに対して有効であるかどうかを判別する

InstallShield MSI Diff を利用して、特定の .msi パッケージにパッチが有効であるかどうかを判別することができます。InstallShield MSI Diff は、パッチに含まれている内部トランスフォームを分析します。

タスク ***パッチが .msi パッケージに対して有効であるかどうかを判別するには、以下の手順に従います:***

1. InstallShield MSI Diff を開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. .msi パッケージを選択して、**[開く]** をクリックします。InstallShield MSI Diff で、.msi データベースが開きます。
4. **[ファイル]** メニューで **[パッチ]** をポイントし、**[パッチの適用]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. パッチ (.msp ファイル) を選択して、**[開く]** をクリックします。

パッチが選択した .msi データベースに適用されるとき、パッチは開いているデータベースに適用されます。パッチの適用が不可能な場合、パッチが選択されたデータベースに無効であることを通知するメッセージ ボックスが表示されます。

パッチによる実行時の影響を判別する

InstallShield MSI Diff を利用して、Windows Installer が指定されたパッチ ファイルをどう処理するか、および、それによってターゲット システムにキャッシュされている .msi パッケージにどう影響されるかを判別することができます。

タスク ***パッチによる実行時の影響を判別するには、以下の手順に従います:***

1. InstallShield MSI Diff を開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. .msi パッケージを選択して、**[開く]** をクリックします。InstallShield MSI Diff で、.msi データベースが開きます。
4. **[ファイル]** メニューで **[パッチ]** をポイントし、**[パッチの適用]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. パッチ (.msp ファイル) を選択して、**[開く]** をクリックします。パッチが、開いている .msi データベースに適用されます。
6. 変更を確認します。

一連のパッチを .msi パッケージに適用する場合、それぞれのパッチに対して、ステップ 4 と 5 を繰り返します。

ヒント: InstallShield MSI Diff では、一連の .msp ファイルと .mst ファイルを .msi ファイルに適用し、結果の .msi データベースに適用された変更を確認することもできます。

パッチを .msi パッケージに適用する方法については、「トランスフォーム (.mst ファイル) によって行われる変更を判別する」をご覧ください。

2 つの .msi パッケージ間の違いを判別する

InstallShield MSI Diff を利用して、2 つの .msi パッケージ間の違いを判別することができます。

タスク ***2 つの .msi パッケージ間の違いを判別するには、以下の手順に従います:***

1. InstallShield MSI Diff を開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. ベースの .msi パッケージを選択して、**[開く]** をクリックします。InstallShield MSI Diff で、.msi データベースが開きます。
4. **[ファイル]** メニューで、**[比較先]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 2 つ目の .msi パッケージを選択して、**[開く]** をクリックします。

2 つ目の .msi パッケージからのデータがベースの .msi パッケージ内のデータと比較され、結果が表示されます。

InstallShield MSI Diff をコマンドラインから実行して、差分のログ ファイルを生成する

InstallShield MSI Diff をコマンドラインから実行して、2 つの .msi、.msm、または .pcp ファイル間の差分を確認したり、トランスフォーム (.mst) またはパッチ ファイル (.msp) によって Windows Installer データベースに適用される変更点を表示するログ ファイルを生成できます。また、このツールをコマンドラインから使用して、バイナリ形式で保存されている 2 つの InstallShield プロジェクト ファイル (.ism または .ise) 間の違いを識別するログ ファイルを生成することもできます。

InstallShield MSI Diff は、次の場所にインストールされます:

InstallShield Program Files フォルダー¥System¥MsiDiff.exe

差分を表示するログ ファイルを生成するには、InstallShield MSI Diff をコマンドラインから起動するときに、次の構文を使用します:

msidiff.exe <BaseFile> <ComparisonFile> /out <diff.xml>

コマンドラインのサンプルを次に示します:

"C:¥Program Files¥InstallShield¥2021¥System¥MsiDiff.exe" "C:¥InstallShield 2021 Projects¥MyProject1.ism" "C:¥InstallShield 2021 Projects¥MyProject2.ism" /out "C:¥Log File.xml"

2 つの InstallShieldプロジェクト間の違いを判別する

InstallShield MSI Diff を利用して、バイナリ形式で保存されている 2 つの InstallShield プロジェクト (.ism または .ise ファイル) 間の違いを判別することができます。

タスク ***2 つの InstallShield プロジェクト間の違いを判別するには、以下の手順に従います:***

1. InstallShield MSI Diff を開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. InstallShield プロジェクト ファイルの 1 つを選択して、**[開く]** をクリックします。InstallShield MSI Diff で、. プロジェクト ファイルが開きます。
4. **[ファイル]** メニューで、**[比較先]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
5. 2 つ目の InstallShield プロジェクト ファイルを選択して、**[開く]** をクリックします。

2 つ目のプロジェクトからの最初のプロジェクト内のデータと比較され、結果が表示されます。

InstallShield MSI Query

InstallShield MSI Query は、SQL ステートメントをビルド スクリプトで実行する前に、SQL の Windows Installer バージョンを使ってテストを行います。SQL ステートメントが正しくフォーマットされているかを素早く確認でき、また生成される結果を参照できます。

InstallShield MSI Query は、Windows [スタート] メニューの InstallShield Tools サブフォルダーから起動します。

Windows Installer における有効な SQL クエリについては、Windows Installer ヘルプ ライブラリの「SQL Syntax」を参照してください。

特定の Windows Installer ベースに対して SQL Query を実行する

InstallShield MSI Query を使って、SQL ステートメントをビルド スクリプトで実行する前に、SQL の Windows Installer バージョンを用いてステートメントをテストすることができます。SQL ステートメントが正しくフォーマットされているかを素早く確認でき、また生成される結果を参照できます。

タスク ***クエリを実行するには、以下の手順を実行します。***

1. InstallShield MSI Query を開きます。
2. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
3. テストに使用するデータベース (.msi または .msm ファイル) を選択して、**[OK]** をクリックします。InstallShield MSI Query のタイトル バージョンに、ファイルの名前が表示されます。
4. **[クエリ]** ボックスで、実行するクエリを入力します。

   InstallShield MSI Query では、1 つ以上のパラメーターの値を疑問符 (?) として入力することができます。クエリに疑問符を含めた場合、[クエリ] ボックスの下に [引数] 行が表示されます。各疑問符は異なる [引数] ボックスに対応しています。[引数] ボックスを使用して、置換を値を指定します。

   Windows Installer における有効な SQL クエリについては、Windows Installer ヘルプ ライブラリの「SQL Syntax」を参照してください。
5. **[実行]** ボタンをクリックします。

選択したデータベースに対して指定されたクエリが実行されます。クエリの結果として発生した変更を保存するには、[ファイル] メニューにある [保存] をクリックします。

InstallShield MSI Sleuth

InstallShield MSI Sleuth は、ターゲット システムの現在のインストール済み状態を参照できる診断ツールです。InstallShield MSI Sleuth はインストール済みのすべての .msi パッケージの一覧を表示します。リストに挙げられている任意の .msi パッケージをクリックすると、データベースに含まれるテーブルと、データベースに埋め込まれているストリームされたファイルのすべてを見ることができます。このツールは、特定のコンポーネント コードを持つパッケージを含むインストール済みの製品 (複数可) を識別するのにも役立ちます。

InstallShield MSI Sleuth は、Windows [スタート] メニューの InstallShield Tools サブフォルダーから起動します。

ターゲット システムで実行された .msi パッケージについての詳細を表示する

InstallShield MSI Sleuth を利用して、ターゲット システムで実行された .msi パッケージについての詳細を表示することができます。

タスク ***.msi パッケージに関する詳細を表示するには、以下の手順に従います:***

1. InstallShield MSI Sleuth を開きます。
2. **[ファイル]** メニューで、**[パッケージを開く]** をクリックします。**[インストール済みのパッケージ]** ダイアログ ボックスが開きます。ターゲット システムにインストールされたパッケージの全一覧が表示されます。
3. 分析するパッケージを選択して、**[OK]** をクリックします。

選択した .msi パッケージの詳細が表示されます。

ヒント: *詳細は .sleuth.xml に保存することができます。.sleuth.xml は MSI Sleuth 内から再度開くことができます。詳細を保存するには、以下の手順に従います: 詳細を保存するには、[ファイル] メニューで、[パッケージの詳細を保存] をクリックしてから、ファイル名を指定します。*

.sleuth.xml ファイルをあとで開くには、[ファイル] メニューで、[パッケージの概要を開く] をクリックします。開くファイルを選択します。

特定のコンポーネントを含むすべてのインストール済みのパッケージを検索する

InstallShield MSI Sleuth を利用して、あるコンポーネントを、指定されたアンインストールされないファイルに関する問題を解決するためのコンポーネント コードと一緒にインストールしたすべてのパッケージを識別することができます。また、コンポーネントが複数の製品によってインストールされたかどうかを判別することもできます。

タスク ***特定のコンポーネントを含むすべてのインストール済みのパッケージを検索するには、以下の手順に従います:***

1. InstallShield MSI Sleuth を開きます。
2. **[ファイル]** メニューで、**[コンポーネント コードで開く]**、**[コンポーネント コード]** の順にポイントします。このメニュー コマンドが編集フィールドに変わります。
3. コンポーネント コードを入力し、Enter キーを押します。クリップボードに保存されたコンポーネント コードがある場合、CTRL+V を押して、それをメニュー コマンド フィールドに貼り付けます。**[インストール済みのパッケージ]** ダイアログ ボックスが開きます。

指定されたコンポーネント コードを含むインストール済みパッケージの全一覧が表示されます。パッケージを表示するには、それを選択して、[OK] をクリックします。

InstallShield MSI Grep

InstallShield MSI Grep は、.msi と .msm パッケージのコレクションの中から特定のテキストを検索します。特定のテーブルまたは列に関する結果のみを表示するための詳細検索も可能です。

InstallShield MSI Grep は、Windows [スタート] メニューの InstallShield Tools サブフォルダーから起動します。

.msi パッケージで特定の文字列を検索する

InstallShield MSI Grep を利用して、ひとまとまりの .msi と .msm パッケージのコレクションから特定のテキストを検索することができます。特定のテーブルまたは列に関する結果のみを表示するための詳細検索も可能です。

タスク ***1 つまたは複数の .msi と .msm パッケージで特定のテキストを検索するには、以下の手順に従います:***

1. InstallShield MSI Grep を開きます。
2. **[検索]** ボックスで、検索する文字列を入力します。
3. 検索条件を狭めるには、次から該当するものを実行します:
   - **[テーブル]** ボックスで、検索するデータベース テーブルの名前を指定します。
   - **[列]** ボックスで、検索するデータベース列の名前を指定します。
   - InstallShield MSI Grep で検索するフォルダーの一覧を変更するには、省略ボタンをクリックして、1 つまたは複数のフォルダーを選択します。
   - 指定したディレクトリの全サブフォルダーで検索を行うには、**[再帰]** チェック ボックスを選択します。
   - 検索を特定のデータベースの種類またはファイル名に制限する場合、**[ファイル]** ボックスでファイル名を入力します。アスタリスク (*) をワイルドカード文字として使用します。複数のファイル名またはパターンを指定する場合、セミコロンでそれぞれ区切ります。
4. **[検索]** ボタンをクリックします。

検索の結果が表示されます。検索結果をクリックしてから、検索結果からのテーブルをクリックすると、詳細が表示されます。

InstallShield キャビネット & ログ ファイル ビューアー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield キャビネット & ログ ファイル ビューアーを使って、以下を行います:

- InstallScript キャビネット ファイル (.cab) または InstallScript ヘッダー ファイル (.hdr)、およびそれらの圧縮ファイル、レジストリ エントリ、コンポーネント、機能、その他のデータ を参照する。このツールを使って、.cab ファイルからファイルを抽出することもできます。この関数は、InstallScript および InstallScript オブジェクト プロジェクトに適用します。
- InstallScript インストールが作成した InstallScript ログ ファイル (.ilg) を参照する。このツールを使って、インストールがログ ファイルに記録した内容を参照できます。ログ ファイルには、重要なアンインストール情報がバイナリ形式で保存されます。この機能は、InstallScript プロジェクトおよび InstallScript MSI プロジェクトに適用します。

タスク

InstallShield 内から InstallShield キャビネット & ログ ファイル ビューアーを起動するには、以下の手順に従います:

[ツール] メニューで **InstallScript** をポイントして **[キャビネット & ログ ファイル ビューアー]** をクリックします。

ヒント: *このツールは、Windows [スタート] メニューの [InstallShield ツール] サブフォルダーから起動することもできます。*

詳細については、ドキュメントの該当セクションを参照してください:

- InstallScript キャビネット ファイル (.cab) およびヘッダー ファイル (.hdr) を表示する
- InstallScript ログ ファイル (.ilg) を参照する

InstallScript キャビネット ファイル (.cab) およびヘッダー ファイル (.hdr) を表示する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

InstallShield キャビネット & ログ ファイル ビューアーを使って、data1.cab (InstallScript キャビネット ファイル) および data1.hdr (InstallShield が InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトに作成する InstallScript ヘッダー ファイル) を開くことができます。InstallShield キャビネット & ログ ファイル ビューアーを使って、これらのファイルからファイルを抽出することもできます。

data1.cab または data1.hdr ファイルを InstallShield キャビネット & ログ ファイル ビューアーで開くと、そのインターフェイスは 2 つのペインに分かれています。左側のペインには data1.cab ファイルまたは data1.hdr ファイルが Windows Explorer に似たディレクトリ構造として表示され、右側のペインには左側のペインで選択されたアイテムと関連付けられた情報が表示されます。

右側のペインに表示されるフォルダは、インストールによって使用される情報を示します。ほとんどの場合、フォルダにはファイルが含まれていません。これらの種類のフォルダを選択すると、右側のペインに対応する設定のグリッドが表示されます。フォルダをクリックすると、そのフォルダと関連付けられた設定が表示されます。

InstallShield キャビネット & ログ ファイル ビューアーを使って、data1.cab ファイルおよび data1.hdr ファイルを変更または編集することはできません。ただし、data1.cab ファイルまたは data1.hdr ファイルについての情報をテキスト ファイルとして保存および編集することができます。詳細については、「InstallScript .cab または .hdr ファイルからファイルについてのレポートを生成する」を参照してください。

InstallShield キャビネット & ログ ファイル ビューアーではエキスパート モードを有効または無効に切り替えることで、data1.cab ファイルまたは data1.hdrファイルについて表示される情報量を調節できます。

- エキスパート モードを無効にすると、ビューアーはインストール プロジェクトに追加されたファイル、レジストリ エントリ、およびその他の情報のみを表示します。
- エキスパート モードを有効にすると、ビューアーはインストール プロジェクトに追加されたファイル、レジストリ エントリ、およびその他の情報すべてを表示します。ビューアーはまた、 が data1.cab ファイルに追加したファイルも表示します。これには、InstallScript エンジンおよびコンパイルされたスクリプト ファイル (Setup.inx) なども含まれます。さらに、ビューアーはインストール プロジェクトに含まれている InstallScript オブジェクトも表示します。

タスク ***InstallShield キャビネット & ログ ファイル ビューアーでエキスパート モードを有効または無効にするには、以下の手順に従います:***

[ビュー] メニューで、**[エキスパート モード]** をクリックします。

[エキスパート モード] コマンドにチェック マークが付いている場合、エキスパート モードは有効になっています。チェック マークが付いてない場合、エキスパート モードは無効です。

InstallScript .cab および .hdr ファイルの概要

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

InstallScript インストールをビルドすると、InstallShield は data1.cab という名前の InstallScript キャビネット ファイル (.cab) と、必要な場合は追加の .cab ファイルを作成します。InstallShield はまた、data1.hdrという名前の InstallScript ヘッダー ファイルを作成します。プロジェクトのリリースの設定に基づいて、これらのファイルは非圧縮の状態か、Setup.exe ファイルにストリームされます。

InstallScript .cab ファイルは、インストールがユーザーのシステムにインストールするファイルを含む圧縮ファイルです。キャビネット ファイルには、InstallShield またはスクリプトを通して構成された、インストールされるファイルが格納されています。キャビネット ファイルにはまた、言語およびシステム依存関係に関する情報とファイルも含まれています。さらに、InstallShield がビルド時にインストールに追加する InstallScript エンジンなどのその他のファイルも格納します。

data1.hdr ファイルは InstallScript ヘッダー ファイル (.hdr) で、対応する data1.cab ファイルを参照します。このファイルには、インストール プロジェクトに入力された全般情報が含まれています。また、インストールに含まれる各ファイル、コンポーネント、機能、その他のデータに関する情報も含まれています。

重要: *InstallShield キャビネット & ログ ファイル ビューアーを使って、*data1.cab *ファイルおよび* data1.hdr *ファイルを開くことができます。InstallShield は、インストールに* data2.cab *などの別の .cab ファイルを作成する場合がありますが、ビューアーは* data1.cab *ファイルおよび* data1.hdr *ファイルのみを開くことができます。InstallScript インストールの一部である* data1.cab *ファイル、*data1.hdr *ファイル、およびその他すべての .cab ファイルは、InstallShield キャビネット & ログ ファイル ビューアーを使用するときに同じフォルダに配置されていなくてはなりません。*

InstallScript .cab および .hdr ファイルを開く

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

重要: *InstallShield キャビネット & ログ ファイル ビューアーを使って、*data1.cab *ファイルおよび* data1.hdr *ファイルを開くことができます。InstallShield は、インストールに* data2.cab *などの別の .cab ファイルを作成する場合がありますが、ビューアーは* data1.cab *ファイルおよび* data1.hdr *ファイルのみを開くことができます。InstallScript インストールの一部である* data1.cab *ファイル、*data1.hdr *ファイル、およびその他すべての .cab ファイルは、InstallShield キャビネット & ログ ファイル ビューアーを使用するときに同じフォルダに配置されていなくてはなりません。*

タスク ***InstallShield キャビネット & ログ ファイル ビューアーで data1.cab または data1.hdr ファイルを開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. 開くファイルが含まれている場所を参照して、そのファイルを選択します。
3. **[開く]** ボタンをクリックします。

InstallShield キャビネット & ログ ファイル ビューアーは、data1.hdr または data1.cab ファイルについて以下の情報を表示します:

- **機能** – インストールに含まれる各機能に対して構成された設定を表示します。各機能に関連付けられているコンポーネントも表示します。
- **コンポーネント** – インストールに含まれる各コンポーネントに対して構成された設定を表示します。コンポーネントを展開して、そのコンポーネントに含まれているファイルを参照できます。任意のファイルを抽出することができます。

- **セットアップの種類** – インストールに含まれる各セットアップの種類に対して構成された設定を表示します。ビューアーのこの領域には、各セットアップの種類に関連付けられた機能も表示されます。
- **エクスプローラー シェル** – InstallShield でプロジェクトに追加された各シェル オブジェクトをリスト表示します。たとえば、インストールにショートカットまたはプログラム フォルダが含まれている場合、ビューアーのこの領域にそれらが表示されます。
- **レジストリ エントリ** – ターゲット システムのレジストリにインストールされるすべてのエントリをリスト表示します。
- **メディア 設定** – [一般情報] ビューで構成されたプロジェクトの設定を表示します。そのファイルを作成した InstallShield のバージョン情報も表示されます。

data1.hdr ファイルまたは data1.cab ファイルの任意の情報を参照するには、InstallShield キャビネット & ログ ファイル ビューアーの左側のペインでノードを展開してから、任意のノードまたはサブノードをクリックします。

InstallScript .cab または .hdr ファイルからファイルを抽出する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

data1.cab ファイルまたは data1.hdr ファイルが InstallShield キャビネット & ログ ファイル ビューアーで開いて、インストールの cab ファイルに圧縮されている任意のファイルを抽出して、別の場所に保存することができます。

タスク ***ファイルを抽出するには、以下の手順に従います:***

1. 左側のペインで、**[コンポーネント]** ノードを展開してから、抽出したいファイルを含むコンポーネントを展開し、そのコンポーネントの下にある **[ファイル]** ノードをクリックします。
2. 右側のペインで、抽出したいファイルのレコードを選択します。
3. **[ツール]** メニューで、**[ファイルの抽出]** をクリックします。**[保存]** ダイアログ ボックスが開きます。
4. InstallShield キャビネット & ログ ファイル ビューアーを使って抽出したファイルを保存する場所を参照してから、**[保存]** ボタンをクリックします。

メモ: *メディア ファイルとコンポーネントは、パスワードで保護されている場合があります。メディア全体がパスワードで保護されている場合、そのメディアに関連付けられているキャビネット ファイルを開くためには、正しいパスワードの入力が必要です。個別のコンポーネントがパスワードで保護されている場合、そのコンポーネントに関連付けられている圧縮ファイルを抽出するためには、正しいパスワードの入力が必要です。*

InstallScript .cab または .hdr ファイルからファイルについてのレポートを生成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

InstallShield キャビネット & ログ ファイル ビューアーを使って、現在開いている data1.cab または data1.hdr ファイルに含まれるファイル、コンポーネント、機能、その他のデータに関する概要情報を含むテキスト形式のレポートを生成することができます。

タスク ***開いている data1.cab または data1.hdr ファイルに関するレポートを生成するには、以下の手順に従います:***

[ツール] メニューで、**[レポート]** をクリックします。

InstallShield キャビネット & ログ ファイル ビューアーが、Notepad などのテキストエディタでレポート (.txt ファイル) を生成します。

InstallScript ログ ファイル (.ilg) を参照する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield キャビネット & ログ ファイル ビューアーを使って、InstallScript または InstallScript MSI インストールによって作成されたログ ファイルを参照することができます。このビューアーを使って、インストールがログ ファイルに記録した内容を参照できます。ログ ファイルには、重要なアンインストール情報がバイナリ形式で保存されます。

InstallShield キャビネット & ログ ファイル ビューアーで InstallScript ログ ファイル (.ilg) が開かれているとき、ビューアーのインターフェイスは 2 つのペインに分かれています。左側のペインには、ログ ファイルのエントリがフォルダに分かれて表示されます。左側のペインでアイテムを選択すると、右側のペインにそのアイテムに関する情報が表示されます。たとえば、左側のペインで機能を選択すると、右側のペインに "状態" プロパティが表示されます。このプロパティには、機能がインストールされているかどうかが示されます。左側のペインで [ファイル操作] アイテムを選択すると、右側のペインにターゲット システムに転送されたファイルおよびそれらのファイルがパーマネントとしてマークされているかどうかなどの情報が表示されます。

<Support> フォルダーとそのサブアイテムは、セットアップ ファイルの操作に関する情報を提供します。<Data> フォルダーとそのサブアイテムは、アプリケーション ファイルの操作に関する情報を提供します。Disk<1> フォルダとそのサブアイテムは、メンテナンス モードとアンインストールを有効にするインストール ファイルについての情報を提供します。

InstallScript ログ ファイルの概要

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

通常、InstallScript ログ ファイルは、製品がアンインストールされたとき、またはインストールが中止されたときにのみ使用されます。ログ ファイルは、インストールのログ記録が有効な場合に、それが行った操作を記録します。これらの操作には、以下が含まれます:

- 製品名など、インストールされる製品の一般情報。
- インストールされる製品のレジストリ キーの場所とその値。
- 転送されたファイル、作成されたフォルダ、および作成されたショートカットまたはアイコンをはじめとする、システム ファイルの構造に行われた変更。
- InstallShield プロジェクトに追加されたファイルに関する情報。

ログ ファイルの名前は、setup.ilg です。ログ ファイルは、インストールがターゲットシステムで実行されるとき、通常のファイル転送中に自動的に作成されます。ログ ファイルのデフォルトの場所は、次の通りです:

<PROGRAMFILES>¥InstallShield Installation Information¥{Product Code}

ただし、Move Data イベントの前に DISK1TARGET システム変数の値を変更して、この場所を変更することができます。

InstallScript ログ ファイルはバイナリ ファイルです。InstallShield キャビネット & ログ ファイル ビューアーを使って、ログ ファイルの内容を参照できます。

ヒント: *デフォルトで、ログ記録は有効となっています。ログ記録は、InstallScript コード内で Disable(LOGGING) および Enable(LOGGING) 関数を呼び出して有効または無効にできます。詳細については、「Disable」および「Enable」を参照してください。*

InstallScript ログ ファイルを開く

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield キャビネット & ログ ファイル ビューアーを使って、InstallScript ログ ファイル (.ilg) の内容を参照できます。

タスク ***InstallShield キャビネット & ログ ファイル ビューアーで InstallScript ログ ファイルを開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. 開くファイルが含まれている場所を参照して、そのファイルを選択します。
3. **[開く]** ボタンをクリックします。

InstallScript ログ ファイルを検索する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield キャビネット & ログ ファイル ビューアーでは、InstallScript ログ ファイル (.ilg) 内のテキストを検索することができます。

タスク ***ログ ファイル内でテキストを検索するには、以下の手順に従います:***

1. **[検索]** メニューで [検索] を**クリック**します。**[検索]** ダイアログ ボックスが開きます。
2. **[検索する文字列]** ボックスで、検索するテキストを入力します。
3. **[次を検索]** をクリックします。検索条件に一致するログ エントリがある場合、最初のログ エントリが右側のペインで選択されます。
4. 条件に一致する次のアイテムがある場合にそれを検索するには、F3 キーを押します。このステップを必要に応じて繰り返します。

InstallScript ログ ファイルをテキスト ファイルに変換する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

InstallShield キャビネット & ログ ファイル ビューアーを使って、InstallScript ログ ファイル (.ilg) のテキスト ファイル (.txt) バージョンを作成できます。

タスク ***ログ ファイルの .txt ファイル バージョンを作成するには、以下の手順に従います:***

[ツール] メニューで、**[レポート]** をクリックします。

InstallShield キャビネット & ログ ファイル ビューアーが、ログ ファイルの .txt ファイル バージョンを作成して、Notepad などのテキストエディタで開きます。

InstallShield MSIX バンドル ユーティリティ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

エディション: *InstallShield MSIX バンドル ユーティリティは、InstallShield の Premier および Professional Edition で提供されています。*

メモ: *MSIX バンドルをビルドするには、InstallShield が搭載されているマシン上に Windows 10 または Windows Server 2019 以降が必要です。*

InstallShield MSIX バンドル ユーティリティを使って、MSIX バンドル パッケージを作成することができます。グラフィック インターフェイスでペイロード MSIX パッケージおよび署名情報を指定し、構成済みの MSIX パッケージから MSIX バンドル パッケージを作成できます。

MSIX バンドル パッケージとは、異なるシステム アーキテクチャ、言語固有の資産、様々なイメージスケール資産、または特定のデバイスに適用するリソースをターゲットとする MSIX パッケージのバンドルです。MSIX バンドルを使って、異なるアーキテクチャ向けの MSIX パッケージ、および関連資産パッケージを単一のエンティティにパッケージすることができます。.Msixbundle のインストール中、デバイスのアーキテクチャに適用するパッケージのみがインストールされます。

InstallShield では、InstallShield MSIX プロジェクト、MSI からMSIX への変換ウィザード、または リリースの [Appx/MSIX] タブ (基本の MSI プロジェクト) を使って MSIX パッケージをビルドすることができます。これらの MSIX パッケージは、MSIX バンドル ユーティリティを使って MSIX バンドルをビルドして構成します。

MSIX バンドルをビルドするためには、MSIX パッケージをユニバーサル拡張またはデスクトップ拡張と共にビルドし、Windows 10 バージョンを 10.0.17763.0 に設定する必要があります。

異なるプラットフォームをターゲットとする MSIX パッケージパッケージから MSIX バンドルを作成できます。Windows [スタート] メニューにある [InstallShield] フォルダーから InstallShield MSIX バンドル ユーティリティを起動するか、InstallShield ツール メニューまたはリボンから **[MSIX バンドル ユーティリティ]** オプションをクリックして、このツールを起動することもできます。

MSIX バンドルの構成

タスク ***MSIX バンドルを構成するには、次の手順に従ってください:***

1. **[追加]** ボタンをクリックして MSIX ペイロード パッケージを追加します。
2. (必要な場合) 署名証明書情報を入力して、MSIX バンドル パッケージに署名を行います。
3. 生成される MSIX バンドル パッケージの出力ファイル名を入力します。
4. **[次へ]** ボタンをクリックして、MSIX バンドル パッケージをビルドします。
   ビルド進行状況および生成されるバンドル情報が **[進行状況]** ウィンドウに表示されます。

MSIX パッケージおよび署名情報が構成されると、MSIX バンドル ユーティリティが MSIX パッケージの MSIX バンドル要件を検証します。次に、MSIX バンドル パッケージが作成され、提供された署名情報に基づいて MSIX バンドルに署名が行われます。

InstallShield には、 MSIX バンドルを、アドバンスト UI またはスイート/アドバンスト UI プロジェクトにパッケージとして含めることができるサポートも提供されています。

MSIX バンドルのビルド

MSIX パッケージおよび署名情報を構成してから **[次へ]** ボタンをクリックすると、MSIX バンドルのビルドが開始します。ビルド中、MSIX パッケージでパッケージ ID の一貫性 (MSIX パッケージを通して名前、発行元、バージョンが統一されているかどうか) が検証されます。MSIX バンドルのビルド中に情報の不一致が検出されると、適切なエラー情報と共にビルドが終了します。

パッケージ ID の他にも、次のパッケージ アーキテクチャが検証されます:

- 単一の MSIX パッケージが単一のアーキテクチャに所属するかどうか、または
- **[ニュートラル]** アーキテクチャを持つ MSIX パッケージが他のアーキテクチャを持つパッケージと共に構成されているかどうか。**[ニュートラル]** パッケージが構成されている場合、別のアーキテクチャを持つパッケージと共に MSIX パッケージをバンドルすることはできません。競合がある場合、適切なエラー情報と共に MSIX バンドルのビルドが終了します。

検証が完了すると、構成済み MSIX パッケージ ペイロードに基づいて MSIX バンドルが作成されます。ペイロード パッケージがターゲット デバイス ファイミーで古いプラットフォームをターゲットとする場合、MSIX バンドルの作成が失敗する場合があります。MSIX パッケージが適切なデバイス ファミリーのバージョンをターゲットにするバンドルのビルドを構成するようにしてください。MSIX バンドルが正しくビルドされた後、進行状況ウィンドウにバンドルの詳細とパスが表示されます。

MSIX バンドルは入力された署名情報を使ってデジタル署名され、署名のステータスが進行状況ウィンドウに更新されます。MSIX バンドルに正しく署名を行うため、ペイロード MSIX パッケージに構成されたものと同じ署名情報を提供してください。

MSIX バンドルが正しくビルドされた後、進行状況ウィンドウに MSIX バンドル パスが表示されます。

Windows Installer プロパティおよびアドバンスト UI またはスイート/アドバンスト UI プロパティの使い方

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

Windows Installer ベースのプロジェクトのプロパティとアドバンスト UI およびスイート/アドバンスト UI プロジェクトのプロパティには、いくつかの異なる点があります。これらの違いについては、適宜、説明が記載されています。

Windows Installer、アドバンスト UI、およびスイート/アドバンスト UI エンジンは、プロパティを使ってグローバル情報を保持します。[プロパティ マネージャー] ビューには、Windows Installer エンジン、または、アドバンスト UI またはスイート/アドバンスト UI エンジンが実行時に使用するインストール プロパティの一覧が表示されます。[プロパティ マネージャー] ビューでインストーラー プロパティを作成、変更、または削除できます。ビルド時に、InstallShield は、[プロパティ マネージャー] ビューのプロパティを、作成するインストールに書き込みます。

Windows Installer ベースのプロジェクトにおけるプロパティ サポートについての背景情報は、以下を参照してください:

- Windows Installer プロパティの概要
- Windows Installer プロパティ リファレンス

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでサポートされているプロパティに関する背景情報は、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」をご覧ください。

このセクションには、次のトピックが含まれています:

- Windows Installer ベースのプロジェクトにおけるプロパティの作成
- アドバンスト UI およびスイート/アドバンスト UI プロジェクトでプロパティを作成する
- 既存プロパティを変更する
- ローカライズ可能なプロパティを作成する
- 既存プロパティをローカライズ可能にする
- プロパティ値が Windows Installer ログ ファイルに記録されないように防ぐ

- アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ
- パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する
- InstallScript で Windows Installer プロパティを取得または設定する方法
- プロパティから値を削除する
- プロパティを削除する

Windows Installer プロパティの概要

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ビルトイン InstallShield プロパティの概要は、「Windows Installer プロパティ リファレンス」を参照してください。

プロパティの種類

Windows Installer のプロパティには 4 つの一般的な種類があります。

- パブリック
- プライベート
- 制限付きパブリック
- 必要

メモ: *これらのカテゴリの一部は重複しています。たとえば* ProductCode *プロパティは、必須のプライベート プロパティです。*

パブリック プロパティ

パブリック プロパティの名前には、大文字だけが使用されます。たとえば INSTALLDIR はパブリック プロパティです。パブリック プロパティはコマンドラインで指定することが可能で、インストールの起動に利用したり、ユーザー インターフェイスを利用して選択するのに利用できます。独自のプロパティを作成している場合、エンドユーザーがこれらのプロパティにアクセスできるようにするには、すべて大文字を使って作成してください。

エンドユーザーまたは管理者がユーザー インターフェイスまたはコマンドラインからインストール先を変更するには、コンポーネントのインストール先のディレクトリ識別子をパブリック プロパティにする必要があります。

メモ: インストールのユーザー インターフェイスからの値を、インストールでターゲット システムを変更する時点まで保持するのはパブリック プロパティだけです。エンドユーザーに表示されるダイアログ ボックスのプロパティの値を設定する場合、その値をファイルまたはレジストリに書き込むようにするには、パブリック プロパティを使用するようにしてください (例 MY_PUBLIC_PROPERTY)。

プライベート プロパティ

プライベート プロパティには、その名前に最低小文字が 1 つ使用され、ユーザー インターフェイスから変更することはできません。たとえば ProgramFilesFolder はプライベート プロパティです。プライベートプロパティの値は、コマンドラインから設定できないので、エンドユーザーが制御することはできません。

制限付きパブリック プロパティ

制限付きパブリックプロパティでは、ネットワーク管理者がシステム管理者のみが変更できるパブリックプロパティを定義することができます。これにより管理者は、ネットワークの他のユーザーがセットアップを変更することを心配せずに、即座に設定を変更できます。詳細については、「パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する」を参照してください。

必須プロパティ

Windows Installer サービスは、すべての Windows Installer セットアップで必要な 5 つのプロパティに依存しています。デフォルトで、これらのプロパティは InstallShield を使用して作成したインストールすべてに含まれます。プロジェクトから以下のプロパティのうちの 1 つでも削除した場合、インストールを正常に機能させるには、削除したプロパティを再度追加する必要があります。

- ProductCode
- ProductLanguage
- Manufacturer
- ProductVersion
- ProductName

条件

多くのプロパティは、インストールが起動されるまで設定されません。こうしたプロパティには、ターゲット システムからの情報が与えられます。たとえば、VersionNT プロパティは、インストールが起動されるまで設定されません。オペレーティング システムが Windows NT ベース システムの場合、このプロパティは、ターゲット マシンで実行されている Windows のバージョンに設定されます。たとえば、Windows Vista システムで VersionNT の値は 600 です。

実行時に設定したプロパティを、条件付きインストールの作成に使用できます。Windows Vista のみに製品をインストールする場合は、条件付き論理を使ってエンド ユーザーのシステムを確認し、すべての条件を満たした場合に製品をインストールするように設定することができます。

Windows Installer プロパティ リファレンス

***プロジェクト:** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

多くのプロパティをインストール中に設定できます。次の表には、ほとんどのプロパティの機能並びにそれを実行するのに必要な構文がリストされています。プロパティの多くは InstallShield 内で設定できます。その他のプロパティは、実行時に Windows Installer サービスによって初期化されます。

メモ: *デフォルトで、Windows Installer はプロジェクトに含まれる各 Windows Installer プロパティの最終値を /L 引数 を使って MsiExec を起動して生成されたログ ファイルに書き込みます。Windows Installer バージョン 2.0 からは、特定のプロパティ (パスワードを含むものなど) がログ ファイルに書き込まれるのを防ぐことができます。これを行うには、プロパティ マネージャーで* MsiHiddenProperties *というプロパティを作成し、その値にセミコロンでそれぞれ区切ったプロパティ名の一覧を設定します。このプロパティについての詳細は、「MsiHiddenProperties」を参照してください。*

すべて大文字で表示されているプロパティは、いずれもコマンドラインで設定できます。すべて大文字で表されているプロパティは、*パブリック プロパティ* と呼ばれます。以下のリストは、プロパティの機能に従って整理されています。各表では、含まれるプロパティの種類が簡単に説明されています。

Windows Installer プロパティの次のカテゴリはこのトピックに説明されています。リンクをクリックすると直接カテゴリに移動できます。

- 特別なフォルダーとファイルのプロパティ
- 機能のインストールに関するプロパティ
- その他の構成可能なプロパティ
- ユーザーが提供する情報
- 定義済みユーザー アカウントを作成するためのプロパティ
- 製品固有のプロパティ
- インストーラーが設定するシステム フォルダー
- インストーラーにより設定されるオペレーティング システムのプロパティ
- インストーラーにより設定されるハードウェア プロパティ
- PowerShell のプロパティ
- 仮想マシンのプロパティ
- インストーラーにより更新されるステータス プロパティ
- 日付と時刻のプロパティ
- InstallScript エンジン関連のプロパティ
- SQL 関連のプロパティ

- MDAC プロパティ

メモ: インストール用のプロパティとパス変数の違いに注意してください。パス変数は山かっこ (<>) で囲みます。どちらもディレクトリを表しますが、Windows Installer プロパティは実行時に使用できるのに対し、パス変数は、セットアップを設計、作成するときにソースファイルを指すためにだけ使用できます。

特別なフォルダーとファイルのプロパティ

特別なフォルダーのプロパティは、ターゲット システムに格納されたり、インストールされるファイルの場所を定義します。ファイルのプロパティは特定のファイルを参照します。

インストールでフォルダー プロパティを使用するには、プロパティを角かっこ ([]) で囲みます。たとえばコンポーネントを、デフォルトのインストール先フォルダーの下にある Bin フォルダーにインストールさせるには、コンポーネントの "インストール先フォルダー" 設定に **[INSTALLDIR]Bin** と入力します。

テーブル 1 ・ フォルダーのプロパティ

プロパティ名	説明
INSTALLDIR	このプロパティには、機能とコンポーネントのファイルのデフォルトのインストール先が含まれます。詳細については、「デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定」を参照してください。 INSTALLDIR は、InstallScript コードで変数として直接使用できます。
ISDEBUGLOG	インストールで機能の前提条件のログファイルを生成するには、このプロパティをログ ファイルの完全パスとファイル名に設定します。パスにはディレクトリ プロパティおよび環境変数を使用できます。 このプロパティは、機能前提条件のトラブルシュートに使用され、通常 /debuglog コマンドライン パラメータが使用されるときに、コマンドラインを通して以下のように設定されます: Setup.exe /debuglog"C:¥Path¥setup.log" /v"ISDEBUGLOG=prereq.log"
ISPREREQDIR	このプロパティは、実行中の InstallShield 前提条件へのパスを識別します。このパスは、末尾のスラッシュを含みます。このプロパティを使って、InstallShield 前提条件内のファイルを参照することができます。例: [ISPREREQDIR]MyConfigFile.ini 詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。
SETUPEXEDIR	このプロパティは Setup.exe へのパスを格納します。たとえば、Setup.exe へのパスが C:¥MySetups¥MyApp¥Setup.exe とすると、SETUPEXEDIR の値は C:¥MySetups¥MyApp になります。 詳細については、「SETUPEXEDIR」を参照してください。

テーブル 1・フォルダーのプロパティ（続き.）

プロパティ名	説明
SETUPEXENAME	プロジェクトがビルドされるときに作成される、セットアップランチャー ファイルの名前を識別します。セットアップランチャー ファイルの名前が変更された場合、インストールは実行時に、このプロパティ値を更新します。以下のパスは、このファイルへの完全パスを示します: [SETUPEXEDIR]¥[SETUPEXENAME]
SourceDir	このプロパティには、実行中の .msi ファイルが置かれているフォルダーが格納されます。
TARGETDIR	TARGETDIR プロパティは、管理インストール時に Windows Installer パッケージ（非圧縮データ ファイル）がコピーされる場所を表します。 InstallScript MSI プロジェクトでは、この値は InstallScript の *MSI_TARGETDIR* 変数によって指定されます。

機能のインストールに関するプロパティ

以下のセクションでは、機能のインストールに関するプロパティが説明されています。これらのプロパティを利用して、エンドユーザーは機能をどのようにインストールするかを指定することができます。

テーブル 2・ 機能のインストールに関するプロパティ

プロパティ名	説明
ADDDEFAULT	ADDDEFAULT プロパティは、デフォルトの構成でインストールされる機能のリストをコンマで区切って格納します。このプロパティの値を ALL に設定すると、ユーザーはすべての機能をデフォルトの設定でインストールできます。
ADDLOCAL	このプロパティは、ローカルにインストールされる機能のリストをカンマで区切って格納します。各機能には、インストール用に選択された機能がローカルにインストールされるか、ソース メディアから実行されるかを決めるための Remote Installation プロパティがあります。
ADDSOURCE	このプロパティは、ソース メディアから実行される機能のリストをカンマで区切って格納します。このプロパティが ALL に設定される場合、すべての機能がソース メディアから実行されます。
ADVERTISE	このプロパティは、アドバタイズされる機能のリストをカンマで区切って格納します。
REINSTALL	このプロパティは、再インストールされる機能のリストをカンマで区切って格納します。REINSTALL が ALL に設定されていると、ユーザーのシステムに既にインストールされているすべての機能が再インストールされます。

テーブル 2 - (続き.)機能のインストールに関するプロパティ

プロパティ名	説明
REINSTALLMODE	このプロパティには、再インストールの種類を指定する文字列が含まれます。これらのオプションは、Msiexec.exe /f コマンドラインパラメーターで使用可能な値に対応します。このプロパティの詳細は、Windows Installer ヘルプ ライブラリの「REINSTALLMODE Property」プロパティ トピックを参照してください。 REINSTALLMODE は、常に REINSTALL と一緒に設定されます。
ReinstallModeText	このプロパティには、ユーザーが MaintenanceType ダイアログで [修復] を選択したときに設定される再インストール オプションが含まれます。ReinstallModeText の値はコントロール イベント ReinstallMode に引数として渡されます。このコントロール イベントは、REINSTALLMODE プロパティで使用可能なオプションと同じオプションを受け付けます。 ReinstallModeText は定義済みの Windows Installer プロパティではありません。しかし、上述のコントロール イベントで使用するために、新しいプロジェクトすべてに対してプロパティ マネージャーで指定されます。デフォルト値は *omus* です。
REMOVE	このプロパティは、削除される機能のリストをコンマで区切って格納します。REMOVE が ALL に設定されていると、すべての機能が削除されます。
COMPADDLOCAL	このプロパティは、ローカルにインストールされるコンポーネント ID のリストをカンマで区切って格納します。最小のディスク容量を使用するコンポーネントの機能がインストールされます。
COMPADDSOURCE	このプロパティは、ソース メディアから実行されるコンポーネント ID のリストをカンマで区切って格納します。最小のディスク容量を使用するコンポーネントの機能がインストールされます。
PATCH	パッチをインストールするとき、このプロパティにはパッチ パッケージへの完全パスが含まれます。

その他の構成可能なプロパティ

以下のセクションでは、その他各種の構成可能なプロパティについて説明します。

テーブル 3 - 追加の構成可能プロパティ

プロパティ名	説明
ACTION	このプロパティは、実行するシーケンス ([インストール]、[アドバタイズ] または [管理]) を指定します。使用できる値は INSTALL、ADVERTISE および ADMIN です。ACTION プロパティは、インストーラー起動に使用されたコマンドラインに基づいて、自動的に設定されます。

テーブル 3 • 追加の構成可能プロパティ (続き.)

プロパティ名	説明
ALLUSERS	Windows Installer が、マシンごとまたはユーザーごとのインストールを試みるかどうかを指定します。 ALLUSERS の値が 1 に設定されている場合、Windows Installer はマシンごとのインストールを試みます。マシンごとのインストールの場合、ショートカット、レジストリ エントリなどの構成情報は All Users のプロファイルに格納されます。 • Windows Vista 以降のシステムでは、[ユーザー アカウント制御] が有効にされていて、ユーザーが管理者権限を持たないとき、製品をインストールするためには管理資格情報が必要です。 • 他のシステムでは、ユーザーが管理者権限を持たないとき、インストールはエラー メッセージを表示して終了します。 ***プロジェクト:*** *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、デフォルトで、このプロパティは 1 に設定されています。詳細については、「ユーザーごとのインストールとマシンごとのインストールの違い」を参照してください。* ALLUSERS の値が設定されていなく、値が空の文字列 (“”) の場合、Windows Installer はユーザーごとのインストールを実行し、構成情報はユーザーの個人プロファイルに格納されます。 ALLUSERS の値が 2 に設定されている場合、Windows Vista 以降のシステムで、Windows Installer はマシンごとのインストールを試みます。以前のプラットフォームでは、ユーザーに権限がある場合、Windows Installer は、マシンごとのインストールを試みます。それ以外の場合、Windows Installer はユーザーごとのインストールを実行します。 ***プロジェクト:*** *インストールがサイレントで実行される場合に起こる* ALLUSERS *関連の問題を回避するため、このプロパティ セットは InstallScript MSI プロジェクトでは 1 に設定されています。InstallScript MSI プロジェクトでは、このプロパティを* **SdCustomerInformation** *または* **SdCustomerInformationEx** *ダイアログ関数でオーバーライドすることができます。* ***メモ:*** ALLUSERS *は、InstallScript 変数 では、FOLDER_PROGRAMS のような効果をもちません。*ALLUSERS *は、[実行] シーケンスが実行されるまでは影響がありません。*

テーブル 3・追加の構成可能プロパティ (続き.)

プロパティ名	説明
ARPAUTHORIZEDCDFPREFIX	このプロパティは、アプリケーションの更新チャネルの URL を格納します。
ARPCOMMENTS	このプロパティには、[プログラムの追加と削除] で表示されるこの製品に関するコメントが含まれます。 この値は、[一般情報] ビューの "ARP コメント" 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。
ARPCONTACT	このプロパティには、電子メール アドレスや電話番号などのサポート連絡先情報が含まれます。 この値は、[一般情報] ビューの "サポート連絡先" 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。
ARPINSTALLLOCATION	このプロパティには、アプリケーションのプライマリ フォルダーへの完全修飾パスが格納されます。ARPINSTALLLOCATION は、"プロパティの設定" タイプのカスタム アクションを使って INSTALLDIR の値に設定することができます。
ARPNOREPAIR	このプロパティが 1 に設定されている場合、プログラム ウィザードに [修復] ボタンは表示されません。
ARPREADME	製品の Readme ファイルの完全修飾パスまたは URL を保持します。 この値は、[一般情報] ビューの Readme 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。
ARPSIZE	このプロパティは、アプリケーションの予測サイズをキロバイト単位で格納します。
ARPSYSTEMCOMPONENT	このプロパティを 1 に設定すると、[プログラムの追加と削除] パネルにプログラムが表示されなくなります。
ARPURLINFOABOUT	このプロパティは、アプリケーションのホームページまたは作成者のホームページの URL を格納します。 この値は、[一般情報] ビューの "発行元/製品 URL" 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。

テーブル 3・追加の構成可能プロパティ (続き.)

プロパティ名	説明
ARPURLUPDATEINFO	このプロパティは、アプリケーションの更新情報の URL を格納します。 この値は、[一般情報] ビューの "製品アップデート URL" 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。
ARPNOMODIFY	このプロパティを設定して、製品が [プログラムの追加と削除] を通して変更されることを防ぎます。
ARPNOREMOVE	このプロパティを設定して、製品が [プログラムの追加と削除] を通して削除されることを防ぎます。
AVAILABLEFREEREG	このプロパティは、アプリケーションに必要な追加の空きレジストリ領域の大きさをキロバイト単位で設定可能にします。
CCP_DRIVE	このプロパティは、競争力のあるアップグレード行う特定の製品のインストールディスクのルートパスを保持します。
DISABLEADVTSHORTCUTS	このプロパティは、アドバタイズされたショートカットの作成を無効にします。
DISABLEMEDIA	この設定は、インストーラーがソース リストにメディア情報を追加できないようにします。
DISABLEROLLBACK	このプロパティを 1 に設定すると、インストーラーがインストールプロセス中に変更または削除されたファイルのコピーを保存するロールバック スクリプトを作成できないようになります。
EXECUTEACTION	このプロパティは、ExecuteAction アクションにより開始されたトップレベルのアクションを設定します。
EXECUTEMODE	このプロパティは、インストーラーの実行モードを設定します。[None] という値は、システムが変更されていないことを示します。デフォルト値である [Script] は、システムへの変更はすべてスクリプトを通じて実行されることを意味します。
INSTALLLEVEL	このプロパティは、機能の値に対応する値を維持します。インストールされる機能のレベルが INSTALLLEVEL プロパティと同じであるかそれより小さい場合、機能がインストールされます。これは、[標準] または [カスタム] などの異なるセットアップの種類に使用されます。
LOGACTION	ログされるアクション名のリストはセミコロンで区切ります。

テーブル 3・追加の構成可能プロパティ (続き.)

プロパティ名	説明
MSIINSTALLPERUSER	このプロパティは、Windows Installer によって、パッケージが現在のユーザーに対してのみインストールされることを示します。 • ALLUSERS プロパティが 2 に設定されていて、MSIINSTALLPERUSER に空の文字列 (""") が設定されている場合、Windows Installer は、マシンごとにインストールを実行します。 • ALLUSERS プロパティが 1 に設定されていて、MSIINSTALLPERUSER に 1 が設定されている場合、Windows Installer は、ユーザーごとにインストールを実行します。 このプロパティは、Windows Installer 5 および Windows 7、または Windows Server 2008 R2 で使用できます。以前のバージョンの Windows Installer と Windows は、このプロパティを無視します。 詳細は、次を参照してください: • ユーザーごとのインストールとマシンごとのインストールの違い • MSIINSTALLPERUSER プロパティ (MSDN Web サイト)
Privileged	このプロパティは、ユーザーが管理者である場合、またはアプリケーションが管理者が割り当てたアプリケーションである場合に、程度の高い権限でインストールを実行します。
PROMPTROLLBACKCOST	このプロパティは、ディスク容量不足のためにインストールが続行できなくなった場合の処理を指定します。ユーザー インターフェイス レベルに応じて、ユーザーからの入力なしで自動的にロールバックを行うか、あるいはロールバックを無効にして続行することをユーザーにたずねるか指定できます。
PRIMARYFOLDER	このプロパティで指定したフォルダーが、インストールの「プライマリ」フォルダーになります。このフォルダーに対するパスは、PrimaryVolumePath プロパティ、PrimaryVolumeSpaceAvailable プロパティ、PrimaryVolumeSpaceRequired プロパティ、および PrimaryVolumeSpaceRemaining プロパティの値を決定するために使用されます。
REBOOT	このプロパティは、インストールが完了した後で再起動を強制または抑止することができます。使用できる値は次のとおりです。 • F — インストールが完了したときに再起動を強制します。 • S — ForceReboot アクションによって発生する再起動を除くすべての再起動を抑制します。 • R — Windows Installer アクションによって発生するすべての再起動を抑制します。

テーブル 3 - 追加の構成可能プロパティ (続き.)

プロパティ名	説明
ROOTDRIVE	このプロパティは、管理モードでは、最初に見つかった書き込み可能なネットワーク ドライブにデフォルトのドライブを設定します。その他のモードでは、使用可能な最大量のディスク容量を持つ書き込み可能ローカル ドライブに、デフォルトのドライブを設定します。
SCRIPTFILE	このプロパティは、インストール中に実行されるすべての操作を含むスクリプト ファイルの場所を定義します。
SEQUENCE	このプロパティは、テーブルでアクションが実行される順序を示す .msi データベース テーブルを指定します。
SHORTFILENAMES	管理者モードで、このプロパティに、短いファイル名のみが使用されることを示すように設定できます。
TRANSFORMS	このプロパティは、MSI データベースに適用される トランスフォームのリストを格納します。これらは、インストールとアドバタイズ モードでのみ設定できます。変換を 2 つのテーブルに適用している場合、このプロパティの構文は、c:¥transforms¥trans1;;trans2 のようになります。トランスフォームは文字列に表示される順序で適用されるので注意してください。
TRANSFORMSATSOURCE	このプロパティを 1 に設定すると、インストーラーはインストール ソースでトランスフォームを検索します。
LIMITUI	このプロパティを設定すると、ユーザー インターフェイス レベルが基本に制限されます。カスタム ユーザー インターフェイスをインストーラーに組み込まれた UI と対話させない場合に便利です。
DefaultUIFont	このプロパティは、デフォルトのフォントを指定するために、TextStyle テーブルで見つかったあらかじめ定義されたスタイルの1つに設定します。このプロパティが設定されない場合、インストーラーはシステム フォントを使用します。そのため、フォーマットが壊れることがあります。

ユーザーが提供する情報

以下のセクションでは、エンドユーザーが入力する情報について説明します。エンド ユーザーが入力する情報には、ユーザー名、会社名、または言語などが含まれます。

テーブル 4 - ユーザーが提供する情報

プロパティ名	説明
AdminProperties	AdminProperties は、管理インストール中にプロパティ セットのリストを保持します。これらのプロパティは外部 (ユーザー名) または内部 (このページの他のプロパティ) のいずれかになります。

テーブル 4・ユーザーが提供する情報 (続き.)

プロパティ名	説明
COMPANYNAME	このプロパティは、インストールを実行するエンドユーザーの組織名を格納します。この情報は、**[顧客情報]** ダイアログ ボックス (基本の MSI プロジェクト)、または SdCustomerInformation か SdCustomerInformationEx ダイアログ (InstallScript MSI プロジェクト) から取ったものです。
ISX_SERIALNUM	このプロパティは、エンド ユーザーが CustomerInformation ダイアログの "シリアル番号" フィールドに入力したシリアル番号を格納します。
UserLanguageID	このプロパティは、エンドユーザーのデフォルトの言語 ID を保持します。
USERNAME	このプロパティは、インストールを実行しているエンドユーザー名を保存します。この情報は、**[顧客情報]** ダイアログ ボックス (基本の MSI プロジェクト)、または **SdCustomerInformation** か **SdCustomerInformationEx** ダイアログ (InstallScript MSI プロジェクト) から取ったものです。
ProductLanguage	このプロパティは、数値で表されたこの製品の言語 ID を格納します。

定義済みユーザー アカウントを作成するためのプロパティ

以下のテーブルは、ログオン ダイアログを使わずに、1 つまたは複数の Windows ユーザー アカウントを作成できるプロパティの説明です。

テーブル 5・定義済みユーザー アカウントを作成するためのプロパティ

プロパティ名	説明
ISNetApiLogonUsername	このプロパティの値を、インストール時に作成するユーザー アカウントに設定します。次のいずれかのフォーマットを使用します: ・ *マシン名¥ユーザー名* ・ *ドメイン名¥ユーザー名* 詳細については、「実行時に既定のユーザー アカウントを作成する」を参照してください。
ISNetApiLogonGroup	このプロパティの値を、ユーザー アカウントが属するグループに設定します。 詳細については、「実行時に既定のユーザー アカウントを作成する」を参照してください。

テーブル 5 • 定義済みユーザー アカウントを作成するためのプロパティ (続き.)

プロパティ名	説明
ISNetApiLogonPassword	ISNetApiLogonPassword—このプロパティの値を、ユーザー アカウントに使用されるパスワードに設定します。 詳細については、「実行時に既定のユーザー アカウントを作成する」を参照してください。

製品固有のプロパティ

Property テーブルで設定できる製品固有のプロパティに関する情報は以下のとおりです。これらの種類のプロパティの例としては、テクニカル サポートの電話番号、製品名、シリアル番号があります。

テーブル 6 • 製品固有のプロパティ

プロパティ名	説明
ARPHELPLINK	このプロパティは、テクニカル サポートのインターネット アドレスを保持します。 この値は、[一般情報] ビューの "サポート URL" 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。
ARPHELPTELEPHONE	このプロパティは、テクニカル サポートの電話番号を保持します。 この値は、[一般情報] ビューの "サポート電話番号" 設定で設定されます。インストールをグローバル化するためには、文字列エントリを指定する必要があります。
ProductCode	ProductCode は製品の特定のバージョンの GUID です。言語バージョンやリリース バージョンが異なれば、この ID も異なります。このプロパティは、[一般情報] ビューで設定されます。
ProductName	このプロパティは、製品名を格納します (例、InstallShield)。このプロパティは、[一般情報] ビューで設定されます。
ProductState	インストーラーは製品のインストールされた状態にこのプロパティを設定します。このプロパティは、以下の 4 つの値のいずれか 1 つを保持することができます。 -1 – 製品はインストールおよびアドバタイズされていません。 1 – 製品はアドバタイズされていますが、インストールされていません。 2 – 製品は別のユーザー用にインストールされています。 5 – 製品はインストールされており、現在のユーザーが使用できます。

テーブル 6 • 製品固有のプロパティ (続き.)

プロパティ名	説明
ProductVersion	ProductVersion プロパティは、メジャー、マイナー、およびビルド バージョン番号を *AA.BB.CCCC* の形式で格納します。このプロパティは、[一般情報] ビューで設定されます。
Manufacturer	製品メーカーの名前を格納します。 この値は、[一般情報] ビューの "発行元" 設定で設定されます。プロジェクトをグローバル化するためには、文字列エントリを指定する必要があります。
DiskPrompt	このプロパティは、ディスクを入力するように指示するメッセージ ボックスに表示される文字列を保持します。"Disk 1" のように、ディスクのラベルに印刷されている追加情報に空のテキストを含めることも必要です。
DiskSerial	DiskSerial プロパティはこのリリースの内部シリアル番号に設定する必要があります。
ComponentDownload	このプロパティは、文字列識別子 (GUID) により製品をダウンロードする URL を保持します。
LeftUnit	このプロパティは、数字の左側に単位を表示します。この構造が要求される言語に必要です。
UpgradeCode	これは、すでにインストールされている製品の関連セットを検索するために使用される GUID です。
IsAdminPackage	このプロパティは、現行のインストール パッケージが管理インストールを通じて作成されている場合、1 に設定されます。このプロパティは、事後管理インストールを検出するために使用できます。

インストーラーが設定するシステム フォルダー

以下のプロパティは、エンド ユーザーのシステムの多くのフォルダーへの完全修飾パスを保持します。これらのプロパティの多くは、MsiGetProperty を呼び出すことなしに直接スクリプトで使用できます。

テーブル 7 • システム フォルダーのプロパティ

プロパティ名	説明
AppDataFolder	このプロパティは、現在のユーザーのアプリケーション データ フォルダーへの完全パスを保持します。
CommonAppDataFolder	このプロパティは、All Users アプリケーション データ フォルダーへの完全修飾パスを保持します。

テーブル 7 • システム フォルダーのプロパティ (続き.)

プロパティ名	説明
CommonFilesFolder	このプロパティの値は、32 ビット Common Files フォルダーへの完全修飾パスです。
CommonFiles64Folder	このプロパティの値は、64 ビット Common Files フォルダーへの完全修飾パスです。このプロパティには、Windows Installer バージョン 2.0 が必要です。
DesktopFolder	このプロパティは、現在のユーザーの [デスクトップ] フォルダーへの完全パスを保持するために使用します。ALLUSERS プロパティが設定される場合、DesktopFolder プロパティはすべてのユーザーのデスクトップ フォルダーへの完全パスを保持しなくてはなりません。
FavoritesFolder	FavoritesFolder プロパティは現在のユーザーの Favorites フォルダーへの完全パスを保持します。
FontsFolder	このプロパティは、Fonts フォルダーへの完全パスを保持します。
PersonalFolder	このプロパティは、現在のユーザーの個人フォルダーへの完全パスを保持します。
ProgramFilesFolder	このプロパティは、現在のユーザーの Program Files フォルダーへの完全パスを保持します。
ProgramFiles64Folder	このプロパティは、現在のユーザーの 64 ビット Program Files フォルダーへの完全パスを保持します。このプロパティには、Windows Installer バージョン 2.0 が必要です。
ProgramMenuFolder	このプロパティは、現在のユーザーの [**プログラム**] メニューへの完全パスを保持するために使用されます。ALLUSERS プロパティが設定される場合、ProgramMenuFolder プロパティはすべての ユーザーの [プログラム] メニューへの完全パスを保持しなくてはなりません。
SendToFolder	このプロパティは、現在のユーザーの SendTo フォルダーへの完全パスを保持します。
StartMenuFolder	このプロパティは、現在のユーザーの [スタート] メニュー フォルダーまでの完全パスを保持するために使用されます。ALLUSERS プロパティが設定される場合、StartMenuFolder プロパティはすべての ユーザーの [スタート メニュー] フォルダーへの完全パスを保持しなくてはなりません。

テーブル 7 • システム フォルダーのプロパティ (続き.)

プロパティ名	説明
StartupFolder	このプロパティは、現在のユーザーの [スタートアップ] フォルダーへの完全修飾パスを保持するために使用されます。ALLUSERS プロパティが設定される場合、StartupFolder プロパティはすべてのユーザーの [Startup] メニューへの完全パスを保持しなくてはなりません。
SystemFolder	このプロパティは、32-bit System フォルダーまでの完全パスを保持します。
System64Folder	このプロパティは、64-bit System フォルダーまでの完全パスを保持します。このプロパティには、Windows Installer バージョン 2.0 が必要です。
TempFolder	このプロパティは、Temp フォルダーまでの完全パスを保持します。
TemplateFolder	このプロパティは、現在のユーザーの Template フォルダーへの完全パスを保持します。
WindowsFolder	このプロパティは、ユーザーの Windows フォルダーへの完全パスを保持します。
WindowsVolume	このプロパティは Windows がインストールされているドライブに設定されます。

インストーラーにより設定されるオペレーティング システムのプロパティ

以下のプロパティは、インストーラーにより実行時に設定されます。ターゲット システム上の環境変数を参照します。

テーブル 8 • オペレーティング システムのプロパティ

プロパティ名	説明
AdminUser	このプロパティは、ユーザーが管理者権限を持っている場合のみ、インストール時にインストーラによって設定されます。
ComputerName	このプロパティは、インストールが実行されているコンピューターの名前を格納します。このプロパティは、インストーラーの初期設定時に Windows API の **GetComputerName** の呼び出しによって設定されます。
LogonUser	このプロパティは、インストールを実行するユーザーの名前を格納します。このプロパティは、Windows API の **GetUserName** の呼び出しによって設定されます。
OLEAdvtSupport	このプロパティはインストーラによって初期化中に設定されます。

テーブル 8 • オペレーティング システムのプロパティ (続き.)

プロパティ名	説明
ServicePackLevel	オペレーティング システムのサービスパックがインストールされている場合、このプロパティは、そのアップデートの数値を格納します。
SharedWindows	このプロパティは、ターゲット システムで共有ウィンドウが使用されているときに設定されます。
ShellAdvtSupport	このプロパティは、ターゲット システムで機能のアドバタイズをサポートしている場合、初期化中にインストーラーによって設定されます。このプロパティは、Windows 98 または Windows 98 以降で、また Internet Explorer 4.01 がインストールされている場合はそれ以前のシステムでも自動的に設定されます。
SystemLanguageID	このプロパティは、ターゲット システムのデフォルトの言語 ID を保持します。この値は、初期化時に GetSystemDefaultLangID を呼び出すことによって、インストーラーが定義します。
TerminalServer	このプロパティは、ターゲット システムが Windows Terminal Server を装備したサーバーの場合、初期化時にインストーラーによって設定されます。
TTCSupport	このプロパティは、ターゲット システムが True Type のフォントコレクション (TTC) をサポートする場合、初期化時にインストーラーによって設定されます。JPN – 932、Taiwan – 950、China – 936、Korea – 949、Hong Kong – 950 システムは、TTC をサポートします。
Version9X	このプロパティは、Windows 95 と 98 オペレーティング システムのバージョン番号を整数として格納します。以下の数式が整数を判別するために使用されます: (MajorVersion * 100) + MinorVersionWindows 95 では Version9X は 400 に、Windows 98 では 410 に、Windows ME では 490 に設定されます。Version9X は Windows NT ベースのシステムでは設定されません。
VersionDatabase	このプロパティは、インストール中に使用されるデータベースのバージョン番号を格納します。
VersionNT	このプロパティは、Windows NT ベースのオペレーティング システムのバージョン番号を整数として格納します。以下の数式が整数を判別するために使用されます: (MajorVersion * 100) + MinorVersion特定のオペレーティング システムの VersionNT プロパティについては、Windows Installer ヘルプ ライブラリを参照してください。

テーブル 8 • オペレーティング システムのプロパティ (続き.)

プロパティ名	説明
VersionNT64	このプロパティは、Windows NT ベースのオペレーティング システムのバージョン番号を 64-bit システムでのみ、整数として格納します。以下の数式が整数を判別するために使用されます: (MajorVersion * 100) + MinorVersionこのプロパティには、Windows Installer バージョン 2.0 が必要です。
WindowsBuild	このプロパティは、実行されているオペレーティング システムのビルド番号を格納します。
MsiNTProductType	このプロパティは、ターゲットマシンで実行されている NT オペレーティング システムの種類を格納します。このプロパティには、Windows Installer バージョン 2.0 が必要です。
MsiNTSuiteBackOffice	このプロパティは Microsoft BackOffice コンポーネントがインストールされている場合に 1 に設定されます。他の場合、このプロパティは設定されていません。このプロパティには、Windows Installer バージョン 2.0 が必要です。
MsiNTSuiteDataCenter	このプロパティは、Windows 2000 Datacenter Server がインストールされている場合に 1 に設定されます。他の場合、このプロパティは設定されていません。
MsiNTSuiteEnterprise	このプロパティは、Windows 2000 Advanced Serverがインストールされている場合に 1 に設定されます。他の場合、このプロパティは設定されていません。
MsiNTSuiteEnterprise	このプロパティは、Windows 2000 Advanced Serverがインストールされている場合に 1 に設定されます。他の場合、このプロパティは設定されていません。
MsiNTSuiteSmallBusiness	このプロパティは Microsoft Small Business Server がインストールされている場合に 1 に設定されます。他の場合、このプロパティは設定されていません。
MsiNTSuiteSmallBusinessRestricted	このプロパティは、制限クライアントライセンス付き Microsoft Small Business Server がインストールされている場合に 1 に設定されます。他の場合、このプロパティは設定されていません。
MsiNTSuitePersonal	このプロパティは オペレーティング システムが Workstation Personal の場合に 1 に設定されます。他の場合、このプロパティは設定されていません。
MsiNetAssemblySupport	このプロパティは、オペレーティング システムで .NET Framework アセンブリをサポートするときに設定されます。他の場合、このプロパティは設定されていません。

テーブル 8 • オペレーティング システムのプロパティ (続き.)

プロパティ名	説明
MsiWin32AssemblySupport	このプロパティは、オペレーティング システムで Win32 アセンブリをサポートするときに設定されます。他の場合、このプロパティは設定されていません。

インストーラーにより設定されるハードウェア プロパティ

以下のプロパティは、実行時にインストーラーによって設定され、エンド ユーザーのシステムの特定のハードウェア プロファイルの設定を格納します。

テーブル 9 • ハードウェアのプロパティ

プロパティ名	説明
Alpha	このプロパティは、プロセッサ レベルの数値を格納し、Alpha プロセッサでセットアップが実行される場合にのみ定義されます。(このプロパティは、Windows Installer バージョン 1.0 でのみサポートされています。)
BorderSide	このプロパティは、ウィンドウの横の境界線の幅をピクセル単位で指定します。
BorderTop	このプロパティは、ウィンドウの上の境界線の幅をピクセル単位で指定します。
CaptionHeight	このプロパティは、キャプション領域の高さをピクセル単位で指定します。
ColorBits	このプロパティは、各ピクセルの隣接カラービットの数 (つまり、ユーザーのモニターのカラー深度) を格納します。たとえば、ユーザーのモニターが 256 色を使用している場合、ColorBits は 8 に設定されます。
Intel	このプロパティは、プロセッサ レベルの数値を格納し、Intel 32-bit プロセッサでセットアップが実行される場合にのみ定義されます。
Intel64	このプロパティは、プロセッサ レベルの数値を格納し、Intel 64-bit プロセッサでセットアップが実行される場合にのみ定義されます。このプロパティには、Windows Installer バージョン 2.0 が必要です。
PhysicalMemory	このプロパティは、インストールされている物理メモリの大きさをメガバイト単位で格納します。
ScreenX	このプロパティは、画面の幅をピクセル単位で定義します。
ScreenY	このプロパティは、画面の高さをピクセル単位で定義します。

テーブル 9・ハードウェアのプロパティ (続き.)

プロパティ名	説明
TextHeight	このプロパティは、テキスト文字の高さを設定します。
VirtualMemory	このプロパティには、使用可能なページ ファイル領域の大きさがメガバイト単位で格納されます。

PowerShell のプロパティ

次の PowerShell 関連のプロパティが実行時に使用できます。

テーブル 10 ・ PowerShell のプロパティ

プロパティ名	説明
IS_CLR_VERSION	このプロパティは、カスタム アクションがマネージコードおよび PowerShell スクリプトを実行するためにロードする .NET Framework バージョンの一覧をセミコロン区切りで指定します。ほとんどのシナリオで、このプロパティはインストール パッケージで設定されていません。コマンドラインで設定されます。バージョン 1.1 が必要であることを指定する場合、次のコマンドライン パラメーターを使用します: IS_CLR_VERSION=v1.1.4322 プロパティの値には、.NET Framework の完全バージョン番号を指定する必要があることに注意してください。複数のバージョンが使用可能な場合、バージョンの一覧をセミコロンで区切って指定できます。ロードが可能であるバージョンの中から最初のものが使用されます。次のコマンドライン パラメーターの例では、アクションがバージョン 2.0 をロードしようと試みます。このバージョンが存在しない場合、バージョン 1.1 のロードが試みられます。バージョン 1.1 が見つからなかった場合、アクションは失敗します。 IS_CLR_VERSION=v2.0.50727;v1.1.4322 指定したバージョンがインストールされなかったとき、アクションでインストールされている .NET Framework の中から一番新しいバージョンをロードする試みを行うように指定する場合、下の例のようにセミコロンをプロパティ値の最後に付加します: IS_CLR_VERSION=v2.0.50727;v1.1.4322; また、プロパティ値の最後に付加されたセミコロンは、指定されたバージョンが存在しないが、あるバージョンの .NET Framework が既にロードされている場合、それが既にインストールされている最新バージョンではなくても、アクションが現在ロードされているバージョンを使用することを示します。 .NET Framework のバージョンの不一致が原因でアクションの問題が発生した場合、インストールの実行時に、エンドユーザーにコマンドラインで IS_CLR_VERSION プロパティを設定するように指示することも考えられます。 詳しくは、次を参照してください: • PowerShell カスタム アクションの呼び出し • マネージコード カスタム アクションの実行時要件
POWERSHELLVERSION	PowerShell がインストールされると、定義済みの PowerShell システム検索によって、このプロパティの値が設定されます。

仮想マシンのプロパティ

Windows Installer プロパティを使って仮想マシンの存在を検出および、その種類を判別するには、まず SetAllUsers.dll ファイルで ISDetectVM 関数を呼び出すカスタムアクションを作成する必要があります。手順については、「インストールが仮想マシン上で実行されているかどうかを検出する」を参照してください。

カスタム アクションは、実行時に以下の Windows Installer プロパティを設定します:

テーブル 11 • 仮想マシンのプロパティ

プロパティ名	説明
IS_VM_DETECTED	このプロパティの値が 1 の場合、次の仮想マシン環境のうちの 1 つが存在します: • Microsoft Hyper-V • VMware Player、VMware Workstation、または VMware Server などの VMware 製品 • Microsoft Virtual PC この値が設定されない場合、仮想マシンは検出されませんでした。
IS_VM_TYPE	このプロパティは、ターゲット システムに存在する仮想マシンの種類を示します。次の値が可能です: • 0 – 仮想マシンは検出されませんでした。 • 1 – VMware Player、VMware Workstation、または VMware Server などの VMware 製品 • 2–インストールは Microsoft Hyper-V マシン上で実行中です。 • 3–インストールは、Microsoft Virtual PC マシン上で実行中です。 • 4–仮想マシンの種類は不明です。

インストーラーにより更新されるステータス プロパティ

以下のプロパティは、インストーラーにより実行時に設定されます。これらのプロパティは、インストールの状態を示します。

テーブル 12 • ステータスのプロパティ

プロパティ名	説明
AFTERREBOOT	このプロパティは、ForceReboot アクションにより再起動された直後にインストーラーによって 1 に設定されます。
CostingComplete	このプロパティは、コスティングが始まるとすぐに 1 に設定され、コスティングが完了すると 0 に設定されます。
RollbackDisabled	インストーラーは、RollbackDisabled プロパティをロールバックが無効になるたびに設定します。デフォルトでは、このプロパティは設定されていません。

テーブル 12 • ステータスのプロパティ (続き.)

プロパティ名	説明
インストール済み	このプロパティは、製品がすでにインストールされているかどうかを判別します。
OutOfDiskSpace	このプロパティは、インストールのターゲットになるドライブに十分なディスク容量がない場合に True に設定されます。それ以外の場合は False に設定されます。
OutOfNoRbDiskSpace	このプロパティは、インストール時のターゲット ディスクに十分な空きディスク容量がない場合、およびロールバック機能がオフになっている場合に True に設定されます。すべてのターゲット ディスクに十分な空き容量がある場合は False に設定されます。
Preselected	このプロパティは、機能が事前に選択されているかどうかを判別し、選択されていない場合は選択ダイアログを表示しません。
PrimaryVolumePath	インストーラーは、このプロパティに PRIMARYFOLDER プロパティで指定されたパスを設定します。
PrimaryVolumeSpaceAvailable	インストーラーは、このプロパティに PRIMARYFOLDER プロパティで指定されたボリューム上で使用可能なバイト総数 (512 バイト単位) を表す文字列を設定します。
PrimaryVolumeSpaceRequired	このプロパティは、現在選択している機能に必要なディスク容量の合計をバイト数で表した文字列 (512 バイト単位で表示) を格納します。
PrimaryVolumeSpaceRemaining	インストーラーは、選択されている機能がすべてインストールされた場合に、システムで使用可能な残りのバイト数を 512 バイト単位で表した文字列に、このプロパティを設定します。
Resume	このプロパティは、保留されたセットアップからインストールを続行するときにユーザーに表示されるテキスト文字列を格納します。
UpdateStarted	このプロパティは、システムへの変更がインストール処理の結果として行われる場合に設定されます。
ReplacedInUseFiles	このプロパティは、現在使用中のファイルが上書きされる場合に設定されます。再起動が必要かどうかをチェックするためのカスタム アクションがこのプロパティを使用します。
NOUSERNAME	このプロパティを 1 に設定すると、インストーラーは USERNAME プロパティを設定しません。デフォルトでは、このプロパティは設定されず、USERNAME プロパティはレジストリから設定されます。

テーブル 12 • ステータスのプロパティ (続き.)

プロパティ名	説明
NOCOMPANYNAME	このプロパティを 1 に設定すると、インストーラーは COMPANYNAME プロパティを設定しません。デフォルトでは、このプロパティは設定されず、COMPANYNAME プロパティはレジストリから設定されます。

日付と時刻のプロパティ

テーブル 13 • 日付と時刻のプロパティ

プロパティ名	説明
Date	このプロパティは、現在の日を保持します。
Time	このプロパティは、現在の時間を保持します。

InstallScript エンジン関連のプロパティ

これらのプロパティには、InstallScript エンジンに関連する情報が含まれています。

テーブル 14 • InstallScript エンジン関連のプロパティ

プロパティ名	説明
STANDARD_USE_SETUPEXE	 ***プロジェクト・** このプロパティは、InstallScript MSI プロジェクトに適用されます。* InstallScript MSI プロジェクトは、Setup.exe を起動して実行します。このローカライズ可能プロパティには、Setup.exe を起動しないで、.msi データベースを直接起動したエンド ユーザーに表示されるメッセージが入っています。

SQL 関連のプロパティ

これらのプロパティは、[SQL スクリプト] ビューで構成された SQL 接続および SQL スクリプトの関連データを含みます。

テーブル 15 • InstallScript エンジン関連のプロパティ

プロパティ名	説明
IS_SQLSERVER_ALIAS_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールで SQL Server エイリアスのみ表示するよう指定します。 詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。

テーブル 15 • InstallScript エンジン関連のプロパティ (続き.)

プロパティ名	説明
IS_SQLSERVER_AUTHENTICATION	このプロパティは、指定されたカタログの接続に使用する認証の種類を識別します。デフォルトのプロパティは IS_SQLSERVER_AUTHENTICATION です。プロパティ値には、次の数値が有効です: • 0 – 現在のユーザーの Windows 認証情報 • 1 – サーバー認証 詳細については、「SQL ログインの設定に Windows Installer プロパティを使用する」を参照してください。
IS_SQLSERVER_CONNECTIONS_TO_VALIDATE	SQLLogin ダイアログの [次へ] ボタンをクリックしたときテストされる接続をオーバーライドします。 詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。
IS_SQLSERVER_CXNS_ABSENT_FROM_INSTALL	インストールまたはアンインストール中にスキップする 1 つまたは複数の SQL 接続を指定します。 詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。
IS_SQLSERVER_DATABASE	このプロパティは、インストール中にそのプログラムへの接続を作成する SQL カタログの名前を識別します。 詳細については、「SQL ログインの設定に Windows Installer プロパティを使用する」を参照してください。
IS_SQLSERVER_DO_NOT_USE_REG	レジストリに格納されている SQL Server ログイン情報を使用しないように指定します。 詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。
IS_SQLSERVER_LOCAL_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールでローカル SQL Server のみ表示するよう指定します。 詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。
IS_SQLSERVER_PASSWORD	このプロパティは、サーバー認証に使用するパスワードを識別します。 詳細については、「SQL ログインの設定に Windows Installer プロパティを使用する」を参照してください。

テーブル 15・InstallScript エンジン関連のプロパティ (続き.)

プロパティ名	説明
IS_SQLSERVER_REMOTE_ONLY	SQL Server 参照コンボ ボックスおよびリスト ボックス コントロールでリモート SQL Server のみ表示するよう指定します。 詳細については、「デフォルトの SQL ランタイム動作をオーバーライドする」を参照してください。
IS_SQLSERVER_SERVER	このプロパティは、(Microsoft SQL Server および MySQL の場合) ターゲット サーバーのインスタンス名、または (Oracle の場合) URL 接続文字列またはローカル ネット サービス名を識別します。 詳細については、「SQL ログインの設定に Windows Installer プロパティを使用する」を参照してください。
IS_SQLSERVER_USERNAME	このプロパティは、サーバー認証に使用するログイン ID を識別します。 詳細については、「SQL ログインの設定に Windows Installer プロパティを使用する」を参照してください。

MDAC プロパティ

これらのプロパティは、MDAC のバージョンチェックに適用されます。

テーブル 16・MDAC プロパティ

プロパティ名	説明
ISINSTALL_MDAC_BYVERSION	InstallShield でデフォルトのバージョン チェックを行う場合、このプロパティを作成してプロパティを [はい] に設定します。
ISINSTALL_MDAC_SKIP	独自でバージョンチェックを行っていて、InstallShield による MDAC のインストールを希望しない場合、プロジェクト マネージャーでこのプロパティを作成し、設定します。

SETUPEXEDIR

プロジェクト・ *SETUPEXEDIR プロパティをサポートするプロジェクト タイプは次のとおりです。*

- *基本の MSI*
- *InstallScript MSI*

InstallScript インストールが実行された場所を判別するには、SRCDIR または PACKAGE_LOCATION を使用してください。

SETUPEXEDIR プロパティには、Setup.exe へのパスが含まれています。たとえば、Setup.exe へのパスが C:¥MySetups¥MyApp¥Setup.exe とすると、SETUPEXEDIR の値は C:¥MySetups¥MyApp になります。

SETUPEXEDIR の使用

SETUPEXEDIR は、ディレクトリ識別子 SourceDir の代わりに使用できます。SourceDir を使用したとき、実行中の .msi パッケージの場所がポイントされるという問題が起こることがあります。圧縮インストールの場合、.msi パッケージは一時的な場所にストリームされて、そこから実行されます。このため、SourceDir の値はエンド ユーザーのマシン上の一時的な場所になり、その値が期待していた値とは異なる場合もあります。

SETUPEXEDIR の制限

SETUPEXEDIR の使用には 2 つの制限があります。

- SETUPEXEDIR は Setup.exe によって設定されます。エンドユーザーが .msi パッケージを直接実行した場合、SETUPEXEDIR は設定されません。これに対処するため、SETUPEXEDIR を使うインストールと SourceDir を使うインストールの 2 つの実装をインストールに含めることができます。SETUPEXEDIR の存在をテストし、存在しない場合は条件付きで SourceDir 実装を使用することができます。
- SETUPEXEDIR はアンインストール時に設定されない場合があります。エンドユーザーが Setup.exe を実行することによってアンインストールをトリガーすると、SETUPEXEDIR は設定されます。アンインストールが [プログラムの追加と削除] から実行されると、SETUPEXEDIR は設定されません。

メモ: *非圧縮インストールでは、*SourceDir *と* SETUPEXEDIR *は同じ値を持ちます。*

アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールの実行中、多数のプロパティが設定されます。InstallShield 内部からプロパティの一部の値を設定することができます。その他のプロパティは、実行時にアドバンスト UI またはスイート/アドバンスト UI エンジンによって初期化されます。

基本の MSI プロジェクトと違って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトはパブリック プロパティとプライベート プロパティとを区別しません。ただし、大文字と小文字の区別はあります。アドバンスト UI またはスイート/アドバンスト UI のプロパティの名前に大文字と小文字を組み合わせる場合、プロジェクト内でそのプロパティを参照する際には常に同じ大文字と小文字の組み合わせで使用します。

[プロパティ マネージャー] ビューで定義したプロパティが、2 つ目のプロパティを参照する値に設定されていて、かつ、2 つ目のプロパティを最初のプロパティの値に解決されるようにする場合、場合によって、最初のプロパティに対して、"フォーマット済み" チェックボックスを設定する必要があります。詳細については、「[プロパティ マネージャー] ビュー」を参照してください。

以下は、アドバンスト UI およびスイート/アドバンスト UI インストールで使用できるビルトイン プロパティのカテゴリです:

- アドバンスト UI およびスイート/アドバンスト UI の特殊のプロパティ
- アドバンスト UI およびスイート/アドバンスト UI のフォルダー プロパティ
- システム フォルダーのプロパティ

メモ: アドバンスト UI およびスイート/アドバンスト UI プロパティとパス変数の違いに注意してください。パス変数は山かっこ (<>) で囲みます。どちらもディレクトリを表しますが、アドバンスト UI およびスイート/アドバンスト UI のプロパティは実行時に評価されるのに対し、パス変数は、インストールを設計およびビルドするときにソースファイルを指すためにだけ使用できます。

アドバンスト UI およびスイート/アドバンスト UI の特殊のプロパティ

以下のプロパティによって、アドバンスト UI およびスイート/アドバンスト UI インストールにおける、いろいろな情報が示されます。

テーブル 17 ・ アドバンスト UI およびスイート/アドバンスト UI の特殊のプロパティ

プロパティ	説明
IS_CLR_VERSION	このプロパティは、スイート/アドバンスト UI アクションがマネージコードおよび PowerShell スクリプトを実行するためにロードする .NET Framework バージョンの一覧をセミコロン区切りで指定します。ほとんどのシナリオで、このプロパティはインストール パッケージで設定されていません。コマンドラインで設定されます。バージョン 1.1 が必要であることを指定する場合、次のコマンドライン パラメーターを使用します: IS_CLR_VERSION=v1.1.4322 プロパティの値には、.NET Framework の完全バージョン番号を指定する必要があることに注意してください。複数のバージョンが使用可能な場合、バージョンの一覧をセミコロンで区切って指定できます。ロードが可能であるバージョンの中から最初のものが使用されます。次のコマンドライン パラメーターの例では、アクションがバージョン 2.0 をロードしようと試みます。このバージョンが存在しない場合、バージョン 1.1 のロードが試みられます。バージョン 1.1 が見つからなかった場合、アクションは失敗します。 IS_CLR_VERSION=v2.0.50727;v1.1.4322 指定したバージョンがインストールされなかったとき、アクションでインストールされている .NET Framework の中から一番新しいバージョンをロードする試みを行うように指定する場合、下の例のようにセミコロンをプロパティ値の最後に付加します: IS_CLR_VERSION=v2.0.50727;v1.1.4322; また、プロパティ値の最後に付加されたセミコロンは、指定されたバージョンが存在しないが、あるバージョンの .NET Framework が既にロードされている場合、それが既にインストールされている最新バージョンではなくても、アクションが現在ロードされているバージョンを使用することを示します。 .NET Framework のバージョンの不一致が原因でアクションの問題が発生した場合、インストールの実行時に、エンドユーザーにコマンドラインで IS_CLR_VERSION プロパティを設定するように指示することも考えられます。
ISFeatureInstall	このプロパティは、インストールすることを選択するコマンドラインによる機能選択用のアドバンスト UI またはスイート/アドバンスト UI の Setup.xml ファイルで定義された機能名のリストをコンマ区切りで格納します。
ISFeatureRemove	このプロパティは、削除することを選択するコマンドラインによる機能選択用のアドバンスト UI またはスイート/アドバンスト UI の Setup.xml ファイルで定義された機能名のリストをコンマ区切りで格納します。

テーブル 17・アドバンスト UI およびスイート/アドバンスト UI の特殊のプロパティ（続き.）

プロパティ	説明
ISHiddenProperties	このプロパティは、大文字と小文字を区別するプロパティ名のリストをセミコロン区切りで格納し、それらの値をデバッグ ログ ファイルに書き込まないようにできます。このプロパティを使って、パスワードその他の機密情報を含むプロパティのログ記録を防ぐことができます。 ISHiddenProperties は、次の状況下で値がログ記録されるのを防ぐのに役立ちます: ・ プロパティの値が変更される場合。 ・ アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動したときに、エンド ユーザーが、コマンドラインを使ってプロパティの値を設定する場合。 ・ アドバンスト UI またはスイート/アドバンスト UI インストールがパッケージに渡すコマンドラインを使ってプロパティが構成される場合。これは [パッケージ] ビューの [共通] タブ、[操作] 領域で構成できます。 ISuiteExtension::LogInfo を使ってプロパティをログ記録する場合、スイートエンジンでログ記録を防ぐことはできません。したがって、ログ ファイルにプロパティ値を書き込むために作成するすべてのコードは、ISHiddenProperties を読み込んで、そのプロパティ値がログ記録されるべきかを判別する必要があります。
ISInstallMode	この読み取り専用プロパティは、アドバンスト UI またはスイート/アドバンスト UI インストールを実行中のモードを示す値を格納します。使用できる値は次のとおりです。 ・ 0 – 初回インストール ・ 1 – メンテナンス/UI メンテナンス モードの選択 ・ 2 – メンテナンス/変更 ・ 3 – メンテナンス/削除 ・ 4 – メンテナンス/修復 ・ 5 – ステージングのみ（インストールは、アドバンスト UI またはスイート/アドバンスト UI インストールがステージングされる、または圧縮解除および特定の場所にコピーされる、ステージ モードで実行中。インストール内のいずれのパッケージも、このモードでは実行しません。）
ISInstallProgress	このプロパティには、全アドバンスト UI またはスイート/アドバンスト UI インストールの何パーセントが完了したかが格納されます。この情報は、一般的に InstallationProgress ウィザード ページのプログレス サークル コントロールに使用されます。
ISInstallStatus	このプロパティには、アドバンスト UI またはスイート/アドバンスト UI インストールからの進行状況メッセージが格納されます。

テーブル 17 • アドバンスト UI およびスイート/アドバンスト UI の特殊のプロパティ (続き.)

プロパティ	説明
ISOnRebooted	起動の後にアドバンスト UI またはスイート/アドバンスト UI インストールが再開する場合、このプロパティは True に設定します。
ISParcelProgress	このプロパティには、現在実行中のパッケージの何パーセントが完了したかが格納されます。この情報は一般的に、プログレス サークル コントロールの ″内部プロパティ' 設定に使用されます。
ISPassword	アドバンスト UI またはスイート/アドバンスト UI リリースをパスワードで保護するように構成した場合、アドバンスト UI またはスイート/アドバンスト UI のユーザー インターフェイスで、*ISPassword* プロパティが設定されます。アドバンスト UI またはスイート/アドバンスト UI の Setup.xml ファイルでパスワードを設定して値を入力すると、このプロパティの値は、Setup.xml に含まれるパスワードに対して検証されます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する」を参照してください。
ISPerformDelayedReboot	このプロパティは、アドバンスト UI またはスイート/アドバンスト UI インストールが終了したときターゲット システムを再起動する必要がある場合、アドバンスト UI またはスイート/アドバンスト UI のユーザー インターフェイスによって True に設定されます。
ISParcelStatus	このプロパティは、現在実行中のパッケージからの進行状況メッセージを格納します。
ISRootStagePath	このプロパティは、インストールがステージ モードで実行される場合に使用されます。このプロパティは、アドバンスト UI またはスイート/アドバンスト UI インストールがステージング (または圧縮解除および特定の場所にコピー) されるの場所へのパスを格納します。
ISSelectedLanguage	このプロパティには、インストールで現在表示されている言語の十進 ID を含む文字列が格納されています。InstallationLanguage ウィザード ページなどで、実行時に、このプロパティを変更すると、アドバンスト UI またはスイート/アドバンスト UI インストールの現在の言語が変更されます。このプロパティを使って、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの言語を設定できます。
ISSilentInstall	起動の後にアドバンスト UI またはスイート/アドバンスト UI インストールがサイレントで実行する場合 (ユーザー インターフェイスなし)、このプロパティは True に設定します。

テーブル 17 • アドバンスト UI およびスイート/アドバンスト UI の特殊のプロパティ (続き.)

プロパティ	説明
ISUpdateAvailable	このプロパティは、現在実行中のアドバンスト UI およびスイート/アドバンスト UI インストール用のアップグレードのダウンロードが可能かどうかを示します。アップデートが利用可能な場合、このプロパティは 1 に設定されます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。
ISUpdateVersion	このプロパティは、ダウンロードが利用可能なアドバンスト UI およびスイート/アドバンスト UI インストールのバージョンを示します。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI のフォルダー プロパティ

次のプロパティは、ターゲット システム上でファイルを格納、またはインストールする場所を定義します。

テーブル 18 • アドバンスト UI およびスイート/アドバンスト UI のフォルダー プロパティ

プロパティ	説明
ISLogDir	アドバンスト UI またはスイート/アドバンスト UI インストールに有効化され構成済みのパッケージが 1 つ以上あり、エンドユーザーがアドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインを /log パラメーターを使って起動する場合、このプロパティには、パッケージ ログ ファイルを含むディレクトリのパスが格納されます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。
ISPREREQDIR	このプロパティは、パッケージがステージングされるとき、または実行時に一時的にターゲット システムにコピーされるとき、アドバンスト UI またはスイート/アドバンスト UI パッケージのファイルへのパスを格納します。
ORIGINALSETUPEXEDIR	このプロパティには、アドバンスト UI またはスイート/アドバンスト UI インストールの Setup.exe ファイルへのパスが格納されます。たとえば、Setup.exe へのパスが C:¥MySetups¥MyApp¥Setup.exe とすると、*ORIGINALSETUPEXEDIR* の値は C:¥MySetups¥MyApp になります。

テーブル 18 • アドバンスト UI およびスイート/アドバンスト UI のフォルダー プロパティ (続き.)

プロパティ	説明
ORIGINALSETUPEXENAME	プロジェクトがビルドされるときに作成される、セットアップランチャー ファイルの名前を識別します。セットアップランチャー ファイルの名前が変更された場合、インストールは実行時に、このプロパティ値を更新します。以下のパスは、このファイルへの完全パスを示します: [ORIGINALSETUPEXEDIR]¥[ORIGINALSETUPEXENAME]
SETUPEXEDIR	このプロパティには、アドバンスト UI またはスイート/アドバンスト UI インストールの Setup.exe ファイルへのパスが格納されます。通常、一時保管場所です。
SETUPEXENAME	セットアップランチャー ファイルの名前が変更された場合、インストールは実行時に、このプロパティ値を更新します。以下のパスは、このファイルへの完全パスを示します: [SETUPEXEDIR]¥[SETUPEXENAME]
SETUPSUPPORTDIR	このプロパティは、実行時にアドバンスト UI またはスイート/アドバンスト UI インストールが一時的に使用した後に削除されるサポート フォルダーおよびファイルを含むフォルダーへのパスを格納します。この場所は、*デフォルト ステージング パス* と呼ばれることもあります。[サポート ファイル] ビューの [言語非依存] ノードにあるファイルを参照する場合、次の構文を使用します: [SETUPSUPPORTDIR]¥MyLanguageIndependentFile.txt [サポート ファイル] ビューの言語固有ノードの 1 つにあるファイルを参照する場合、パスに 言語 ID を含めます。たとえば、英語の言語 ID は 1033 です: [SETUPSUPPORTDIR]¥1033¥MyEnglishFile.txt
TempFolder	このプロパティには、アドバンスト UI またはスイート/アドバンスト UI インストールで実行中のユーザー環境における使用中の一時フォルダーへのパスが格納されます。

システム フォルダーのプロパティ

次のプロパティは、エンド ユーザーのシステム上にある様々なフォルダーへのパスを定義します。

テーブル 19 • システム フォルダーのプロパティ

プロパティ	説明
AdminToolsFolder	このプロパティの値は、管理ツールを含むエンド ユーザーのフォルダーへの完全パスです。
AppDataFolder	このプロパティの値は、エンド ユーザーの Application Data フォルダーへの完全パスです。

テーブル 19 • システム フォルダーのプロパティ (続き.)

プロパティ	説明
CommonAppDataFolder	このプロパティの値は、All Users の Application Data フォルダーへの完全パスです。
CommonDocuments	このプロパティの値は、すべてのユーザーに共通のドキュメントを含むフォルダーへの完全パスです。
CommonFilesFolder	このプロパティの値は、32 ビット Common Files フォルダーへの完全修飾パスです。
CommonFiles64Folder	このプロパティの値は、64 ビット システム上の 64 ビット Common Files フォルダーへの完全修飾パスです。このプロパティは、32 ビット システム上では定義されません。
LocalAppDataFolder	このプロパティの値は、ローカル (非ローミング) アプリケーション用のデータ ファイルを含むフォルダーへの完全パスです。
PersonalFolder	このプロパティの値は、エンド ユーザーの個人用フォルダーへの完全パスです。
ProgramFilesFolder	このプロパティの値は、エンド ユーザーの Program Files フォルダーへの完全パスです。64 ビット システム上で、このプロパティは 32 ビットの場所を参照します。
ProgramFiles64Folder	このプロパティの値は、エンド ユーザーの 64 ビット Program Files フォルダーへの完全パスです。このプロパティは、32 ビット システム上では定義されません。
SystemFolder	このプロパティの値は、32 ビット システム フォルダーへの完全パスです。
System64Folder	このプロパティの値は、64 ビット システム上の 64 ビット ビルトイン システム フォルダーへの完全パスです。このプロパティは、32 ビット システム上では定義されません。
WindowsFolder	このプロパティの値は、エンド ユーザーの Windows フォルダーへの完全パスです。

Windows Installer ベースのプロジェクトにおけるプロパティの作成

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでプロパティを作成する方法については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトでプロパティを作成する」をご参照してください。

プロパティマネージャーを使用して、プロジェクト全体で使用される独自のプロパティを作成できます。これらのプロパティでは、一度設定した値はプロジェクト全体で使用できます。

タスク ***プロパティを作成するには、以下の手順に従ってください。***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンをクリックします。ビューの下に新しい行が追加されます。
3. **[名前]** 列に、新しいプロパティの名前を入力します。
4. **[値]** 列に、新しいプロパティの値を入力します。
5. **[コメント]** 列に、オプションでプロパティについてのコメントを入力します。

ヒント: *[プロパティ マネージャー] ビューに値を入力する際は、次のガイドラインに従ってください:*

- *テーブル セル内のテキストをすべて上書きするには、テーブル セルをクリックしてから、新しいプロパティ名、値、またはコメントを入力します。*
- *テーブル セル内の特定の位置にカーソルを配置するには、その場所をダブルクリックします。次に、変更を入力します。*

Windows Installer ベースのプロジェクトで Windows Installer プロパティを作成する時に役立つヒント。

コマンドラインから変更できるプロパティを作成する場合は、プロパティ名をすべて大文字で指定してください。たとえば INSTALLDIR は、コマンドラインから設定や変更が可能なプロパティです。詳細については、「Windows Installer プロパティの概要」を参照してください。

Windows ロゴ *- [プロパティ マネージャー] ビューに入力された情報の有効性は、自動的に検証されません。たとえば、ARPHELPLINK プロパティの値を* **http://www.mycompany.com** *ではなく、***MyCompany** *と変更した場合、エンドユーザーがそのリンクをクリックすると失敗しますが、[プロパティ マネージャー] にデータを入力したとき、またはビルド時に InstallShield によってエラーメッセージが表示されることはありません。ビルトイン Windows Installer プロパティに入力した情報および構文が正しいかどうかを確認するには、「Windows Installer プロパティ リファレンス」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでプロパティを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

プロパティマネージャーを使用して、プロジェクト全体で使用される独自のプロパティを作成できます。これらのプロパティでは、一度設定した値はプロジェクト全体で使用できます。

タスク ***プロパティを作成するには、以下の手順に従ってください。***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンをクリックします。ビューの下に新しい行が追加されます。
3. **[名前]** 列に、新しいプロパティの名前を入力します。
4. **[値]** 列に、新しいプロパティの値を入力します。
5. **[フォーマット済み]** 列でチェックボックスを選択またはクリアして、**[値]** 列に入力したテキストがフォーマット済みかどうかを示します。
   - プロパティ値 (角かっことそのコンテンツを含む) をそのままにしておく場合、**[フォーマット済み]** チェック ボックスをクリアします。
   - [値] 列の値に、実行時に解決させるプロパティ名、環境変数の参照、その他の特殊な文字列を含む 1 つ以上の形式化された式が含まれている場合、**[フォーマット済み]** チェック ボックスを選択します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

ヒント: *[プロパティ マネージャー] ビューに値を入力する際は、次のガイドラインに従ってください:*

- *テーブル セル内のテキストをすべて上書きするには、テーブル セルをクリックしてから、新しいプロパティ名、値、またはコメントを入力します。*
- *テーブル セル内の特定の位置にカーソルを配置するには、その場所をダブルクリックします。次に、変更を入力します。*

既存プロパティを変更する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

タスク ***プロジェクトに含まれる既存プロパティを変更するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. 以下のいずれかを実行します。
   - テーブル セル内のテキストをすべて上書きするには、テーブル セルをクリックしてから、新しいプロパティ名、値、またはコメントを入力します。
   - テーブル セル内の特定の位置にカーソルを配置するには、その場所をダブルクリックします。次に、変更を入力します。

ローカライズ可能なプロパティを作成する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

インストールが使用する言語に基づいて、プロパティが異なる値を使用するには、ローカライズ可能なプロパティを作成できます。

タスク ***ローカライズ可能なプロパティを作成するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[新しいプロパティ]** ボタンの隣にある矢印をクリックして、**[ローカライズ可能なプロパティ]** をクリックします。ビューの下に新しい行が追加されます。そのプロパティの [値] 列に、新しい文字列 ID がリストされます。

3. **[名前]** 列に、新しいプロパティの名前を入力します。

ヒント: *[文字列エディター] ビューを使って、プロジェクトがサポートする各言語に対して異なるプロパティ値を設定できます。*

プロジェクト内における文字列 ID と値の使用方法についての詳細は、「InstallShield で文字列エントリを使用する」を参照してください。

既存プロパティをローカライズ可能にする

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

インストールが使用する言語に基づいて、プロパティが異なる値を使用するには、そのプロパティをローカライズ可能に変更できます。

タスク ***プロパティをローカライズ可能に変更するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. ローカライズを行うプロパティを選択します(複数選択可)。

   連続する複数のプロパティを選択するには、最初のプロパティを選択してから SHIFT キーを押しながら最後のプロパティを選択します。連続しない複数のプロパティを選択するには、最初のプロパティを選択してから CTRL キーを押しながらその他の各プロパティを選択します。
3. **[選択したプロパティをローカライズ可能にする]** ボタンをクリックします。

そのプロパティの [値] 列に新しい文字列 ID が追加されます。

ヒント: *[文字列エディター] ビューを使って、プロジェクトがサポートする各言語に対して異なるプロパティ値を設定できます。*

プロジェクト内における文字列 ID と値の使用方法についての詳細は、「InstallShield で文字列エントリを使用する」を参照してください。

プロパティ値が Windows Installer ログ ファイルに記録されないように防ぐ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

デフォルトで、Windows Installer はインストールに含まれる各 Windows Installer プロパティの最終値を /L 引数 を使って MsiExec を起動して生成されたログ ファイルに書き込みます。Windows Installer バージョン 2.0 以降、特定のプロパティ (たとえば、パスワードを含むプロパティなど) がログ ファイルに書き込まれないように防ぐことができます。

タスク ***プロパティがログ ファイルに書き込まれないように防ぐには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[名前]** 列で、まず MsiHiddenProperties プロパティを見つけます。

   このプロパティがリストされていない場合、**[新しいプロパティ]** ボタンをクリックして、このプロパティを作成してから、**[名前]** 列に **MsiHiddenProperties** と入力します。
3. **[値]** 列に、非表示にするプロパティの名前を入力します。複数のプロパティをリストするには、それぞれをセミコロン (;) で区切ります。

このプロパティについての詳細は、「MsiHiddenProperties」を参照してください。

アドバンスト UI およびスイート/アドバンスト UI デバッグ ログ ファイルへのプロパティ値の書き込みを防ぐ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

/debuglog パラメーターを使って アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動すると、スイート エンジンがデバッグ ファイルを生成します。デフォルトで、デバッグ ログ ファイルには、アドバンスト UI またはスイート/アドバンスト UI プロパティの値が含まれます。

場合によって、スイート エンジンが特定のプロパティの値をデバッグ ログ ファイルに書き込むことを防ぐ必要があります。たとえば、パスワードその他の機密情報を含むプロパティのログ記録を防ぐ必要があります。

タスク ***プロパティがログ ファイルに書き込まれないように防ぐには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[名前]** 列で、ISHiddenProperties プロパティを見つけます。

   このプロパティがリストされていない場合、**[新しいプロパティ]** ボタンをクリックして、このプロパティを作成してから、**[名前]** 列に **ISHiddenProperties** と入力します。
3. [値] 列で、その値をログ記録しないプロパティの名前を入力します。大文字と小文字を正しく区別してください。複数のプロパティをリストするには、それぞれをセミコロン (;) で区切ります。

ISHiddenProperties は、次の状況下で値がログ記録されるのを防ぐのに役立ちます:

- プロパティの値が変更される場合。
- アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動したときに、エンド ユーザーが、コマンドラインを使ってプロパティの値を設定する場合。
- アドバンスト UI またはスイート/アドバンスト UI インストールがパッケージに渡すコマンドラインを使ってプロパティが構成される場合。これは [パッケージ] ビューの [共通] タブ、[操作] 領域で構成できます。

ISuiteExtension::LogInfo を使ってプロパティをログ記録する場合、スイート エンジンでログ記録を防ぐことはできません。したがって、ログ ファイルにプロパティ値を書き込むために作成するすべてのコードは、ISHiddenProperties を読み込んで、そのプロパティ値がログ記録されるべきかを判別する必要があります。

パブリック プロパティが制限付きパブリック プロパティである必要があることを指定する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

制限付きパブリック プロパティを利用すると、ネットワーク管理者は、システム管理者または昇格された権限を持つ個人によってのみ変更が可能なパブリック プロパティを定義することができます。これにより管理者は、ネットワークで承認されていないユーザーがインストールを改ざんする可能性があることを心配することなく、即座に設定を変更できます。

Windows Installer は、いくつかのパブリック プロパティを制限付きパブリック プロパティとして見なします。制限付きパブリック プロパティの全一覧については、Windows Installer ヘルプ ライブラリの「Restricted Public Properties」を参照してください。

追加のパブリック プロパティを含める場合、それらを SecureCustomProperties プロパティに追加します。

以下のようなタスクを実行すると、InstallShield は自動的に該当するプロパティを SecureCustomProperties プロパティに追加します:

- システム検索でパブリック プロパティを使用したとき、InstallShield はそのパブリック プロパティを SecureCustomProperties プロパティに追加します。
- ダイアログを基本の MSI またはマージ モジュール プロジェクトに追加またはインポートして、そのダイアログにプロパティで設定されたコントロールが含まれている場合、InstallShield はそのプロパティを SecureCustomProperties プロパティに追加します。
- プロジェクトにメジャー アップグレード アイテムを追加したとき、InstallShield は [詳細] タブの "検出プロパティ" 設定でそのアップグレード アイテムに設定された値を SecureCustomProperties プロパティに追加します。

これにより、カスタム パブリック プロパティを [ユーザー インターフェイス] シーケンスで設定してから、[実行] シーケンスに渡すことができるようになります。

タスク ***パブリック プロパティが制限付きパブリック プロパティである必要があることを手動で指定するには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. **[名前]** 列で、**SecureCustomProperties** プロパティを見つけます。

   このプロパティがリストされていない場合、**[新しいプロパティ]** ボタンをクリックして、このプロパティを作成してから、**[名前]** 列に **SecureCustomProperties** と入力します。
3. **Value** 列で、制限するパブリック プロパティを入力します。複数のエントリはセミコロン (;) で区切ります。

InstallScript で Windows Installer プロパティを取得または設定する方法

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

MsiGetProperty および **MsiSetProperty** 関数を使うと、Windows Installer プロパティの取得および設定が行えます。例については、「プロパティの取得と設定」を参照してください。

スクリプトで Windows Installer API を利用するには、#include "iswi.h" または #include "ifx.h" ステートメントを含む必要があります。

プロパティから値を削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*

- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、ローカライズ可能なプロパティはサポートされていません。

プロパティ マネージャーからプロパティを削除せずにプロパティの値をクリアできます。この機能は、ローカライズ可能プロパティとローカライズ不可能プロパティで使用できます。たとえば、ARPCOMMENTS のようなローカライズ可能なプロパティをクリアしたい場合があります。

タスク ***プロパティ値をクリアするには、以下の手順に従います:***

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. クリアしたいプロパティを選択します(複数選択可)。

   連続する複数のプロパティを選択するには、最初のプロパティを選択してから SHIFT キーを押しながら最後のプロパティを選択します。連続しない複数のプロパティを選択するには、最初のプロパティを選択してから CTRL キーを押しながらその他の各プロパティを選択します。
3. **[選択したプロパティをクリア]** ボタンをクリックします。

メモ: *プロパティをクリアすると、文字列 ID がないためプロパティがローカライズ不可能になります。*

プロパティを削除する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

プロジェクトで不要になったプロパティがある場合、[プロパティ マネージャー] ビューで、それを削除できます。

タスク　　プロジェクトからプロパティを削除するには、以下の手順に従います:

1. ビュー リストの **[動作とロジック]** の下にある **[プロパティ マネージャー]** をクリックします。
2. 削除するプロパティを選択します(複数選択可)。

   連続する複数のプロパティを選択するには、最初のプロパティを選択してから SHIFT キーを押しながら最後のプロパティを選択します。連続しない複数のプロパティを選択するには、最初のプロパティを選択してから CTRL キーを押しながらその他の各プロパティを選択します。
3. **[選択したプロパティを削除]** ボタンをクリックします。

選択したプロパティのみを削除するのか、選択したプロパティおよび関連付けられた文字列エントリも削除するのかを指定できるメッセージ ボックスが表示されます。

.msi および .msm データベースを直接編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *MSI データベース*
- *MSM データベース*

MSI データベースおよび MSM データベース プロジェクトを使用すると、中間プロジェクト フォーマット (.ism ファイル) で作業せずに、Windows Installer インストール パッケージおよびマージ モジュール データベースを編集することができます。これらのプロジェクトタイプは、ダイレクト エディター機能を拡張して、標準インストール プロジェクトでサポートされている異なるビューを含めます。これらのプロジェクトの種類でサポートされているビューは、 .ism プロジェクトで機能するのと同じように機能するように設計されています。

メモ: *MSI データベースおよび MSM データベース プロジェクトでは、文字列エントリもパス変数もサポートされていません。*

既存の .msi ファイルまたは .msm ファイルを直接編集したり、新規の MSI データベースまたは MSM データベースを開いて新規のデータベースを直接作成することもできます。

このセクションには、次のトピックが含まれています:

- Windows Installer パッケージを開く
- ダイレクト編集モードで .msi/.msm データベースを編集する
- ダイレクト編集モードでファイルを追加する
- ダイレクト編集モードでマージ モジュールを追加する

Windows Installer パッケージを開く

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *MSI データベース*

InstallShield には、ダイレクト編集モードを使って .msi パッケージを直接編集するか、InstallShield プロジェクト (.ism) に変換してから InstallShield で基本の MSI プロジェクトとして編集するオプションがあります。

メモ: *.msi パッケージをダイレクト編集モードで編集する場合、一部の機能に制限があります。*

タスク ***.msi パッケージを InstallShield のダイレクト編集モードで開くには、以下の手順を実行します:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. **[ファイルの種類]** リストで **[Windows Installer パッケージ (*.msi)]** を選択します。

3. 開く .msi パッケージを選択します。
4. **[形式を指定して開く]** ボックスで、**[自動]** が選択されていることを確認します。
5. **[開く]** をクリックします。

メモ: *デフォルトでは、InstallShield はダイレクト編集モードで .msi パッケージを開きます。*

MSI/MSM オープン ウィザードを使用すると、既存の Windows Installer (.msi) パッケージを InstallShield プロジェクト (.ism) ファイルに変換し、新規プロジェクトを開いて InstallShield で変更できます。

メモ: *InstallShield で、パッチ作成プロジェクト ファイル (.pcp ファイル) を開いて、ダイレクト エディターで編集することもできます。*

ダイレクト編集モードで .msi/.msm データベースを編集する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *MSI データベース*
- *MSM データベース*

Windows Installer パッケージ (.msi ファイル) またはマージ モジュール (.msm ファイル) を直接編集する場合、InstallShield のダイレクト モードでファイルを開いてから、必要な変更を行うことができます。

タスク ***.msi ファイルや .msm ファイルを開くには、次の手順を実行します。***

1. **[ファイル]** メニューで、**[開く]** をクリックします。
2. **[ファイルの種類]** リストで **[Windows Installer パッケージ (*.msi)]** または **Windows Installer モジュール (*.msm)** を選択します。
3. 開く Windows Installer パッケージまたはマージ モジュールを参照します。
4. **[開く]** をクリックします。

選択されたパッケージまたはマージ モジュールが、InstallShield インターフェイスの限られたビューで開きます。ダイレクト編集モードで行った変更は、保存したときに直接 .msi または .msm ファイルに追加されます。

ダイレクト編集モードでファイルを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *MSI データベース*

- *MSM データベース*

ダイレクト編集モードでファイルを追加するとき、メディアへファイルを追加するオプションには、msi フォーマット (MSI パッケージまたは MSI データベース) に非圧縮、圧縮、または圧縮してストリームされた状態があります。これらのオプションは、ダイレクト編集モードの [ファイルとフォルダー] ビューに新しいファイルをドラッグ アンド ドロップした時に [メディアの場所選択] ダイアログ ボックスに表示されます。そのダイアログ ボックスには、COM 情報抽出オプションも含まれます。但し、このオプションはファイルをフォルダーにドラッグした場合、または単一ファイルをその他の既存ファイルを持たないコンポーネントに追加した場合のみ有効です。その結果、COM ファイルがその特定のコンポーネントに含まれる唯一のファイルとなります。

ファイルを削除すると、File テーブルからエントリが削除されます。メディアに含まれるファイルは削除されず、メディアに追加される予定のファイルは、保存するまで追加されません。

メモ: *ダイレクト編集モードでは、File と Media テーブルは保存処理中にアップデートされます。*

ファイルを追加すると、デフォルトのシーケンス値と共にレコードが追加されます。このとき Media テーブル エントリは作成されません。メディアまたはプロジェクトが保存されるとき、新たに追加されたファイルのシーケンスはメディアを適切に反映するようアップデートされ、Media テーブルにメディア エントリが作成されます。

ダイレクト編集モードでマージ モジュールを追加する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *MSI データベース*
- *MSM データベース*

ダイレクト編集モードで [再配布可能ファイル] ビューを使って .msi データベースに構成可能なマージ モジュールまたはオブジェクトを 追加すると、[マージ モジュールのプロパティ] ダイアログ ボックスが開きます。このダイアログ ボックスを通して、マージ モジュールのプロパティを参照または設定することができます。ただし、マージ モジュールが最初にインストール データベースにマージされている場合のみ、この設定が可能です。つまり、ダイアログ ボックスが最初に開いたときです。また、ダイレクト編集モード でマージ モジュールを選択すると、直ちにマージされることに注意してください。

メモ: *依存するマージ モジュールのインストール先を設定または構成することはできません。*

マージ モジュールカタログにリストされていないモジュールをマージ解除することはできません。たとえば、Modules¥i386 フォルダーでファイルが見つからない場合。それぞれのマージ モジュールの隣にあるチェック ボックスは選択されていますが、無効となっています。

マージ モジュールを追加、構成、およびマージした後、データベースを InstallShield で保存する時にマージ モジュールに含まれるファイルがすべてソース メディアにコピーされます。

10

InstallShield と外部アプリケーションの統合

InstallShield の重要な側面のひとつは、Microsoft Source Control Interface に準拠するソース管理ソフトウェアとの共存および統合だけでなく、Visual Studio(TM) .NET の様な他のソフトウェア開発ツールとどのように共存、および統合するかという点です。

このセクションにあるヘルプ トピックで、InstallShield での外部アプリケーション サポートの範囲について各詳細を参照してください。このセクションには、次のトピックが含まれています:

- ソース コード管理を使用する
- Microsoft Visual Studio との統合
- Microsoft Visual Studio Team Foundation Server との統合

ソース コード管理を使用する

InstallShieldでは、ソース コード管理ソフトウェアでプロジェクト ファイルの異なるバージョンを管理することができます。InstallShield は、マイクロソフトのソース管理インターフェイスに準拠するあらゆるソース管理システムとインタラクトでき、開発システム上のデフォルトのプログラムを使用します。プログラムがインストールされていない場合は、ソース管理オプションは使用できません。

ヒント: *InstallShield はまた、Team Foundation ソース管理との統合もサポートします。詳細については、「Microsoft Visual Studio Team Foundation Server との統合」を参照してください。*

インストール プロジェクトをソース管理に追加すると、InstallShield はプロジェクト ファイル形式をテキスト形式（XML 形式）に自動的に変換して、デフォルトでソース管理システムに保持します。XML 形式でチェックインしない方法を選択した場合、ファイルはバイナリ ファイルとしてチェックインされます。

メモ: *プロジェクト ファイル フォーマットを XML またはバイナリ形式に変換しても、プロジェクト ファイルの拡張子は .ism のまま変更されません。*

スクリプト ファイル サポート

InstallShield では、任意の InstallScript ファイルおよび SQL スクリプトを挿入、インポート、または新しいファイルとして追加することもできます。また、これらのスクリプト ファイルの保存場所用に [パス変数] ビューで定義されたすべてのパス変数の種類をサポートします。ただし、ソース コード管理 ソフトウェアがパスを解決できるよう、対応するフォルダーの構成がソース コード データベースに確実に存在していることが重要です。

このセクションには、次のトピックが含まれています:

- ソース コード管理統合を使用する
- プロジェクトをソース管理へ追加する
- ソース管理からプロジェクトをチェックアウトする
- ソース管理にプロジェクトをチェックインする
- ソース管理からのプロジェクトのリンクの解除

ソース コード管理統合を使用する

タスク ***InstallShield のソース コード管理の統合を使用した一般的なシナリオを以下に示します。***

1. InstallShield インストール プロジェクトを作成する。
2. ソース管理プログラムにセットアップ プロジェクトを追加する。
3. ソース管理からプロジェクトをチェックアウトする。
4. プロジェクトを編集する。
5. プロジェクトを保存する。
6. ソース管理に再びプロジェクトをチェックインする。

ヒント: *上記手順は、InstallShield で新規プロジェクトがソース管理に自動的に追加されるように、あるいは編集したプロジェクトをチェックアウトするように設定するなどして、合理化することができます。詳細については、[オプション] ダイアログ ボックスの [ソース管理] タブを参照してください。このダイアログ ボックスは、[ツール] メニューで [オプション] をクリックすると表示されます。*

プロジェクトをソース管理へ追加する

InstallShield を使って、InstallShield プロジェクトを作成した後、それをソース管理システムに追加することができます。

ヒント: *[オプション] ダイアログ ボックスの [ソース管理] タブで* ***[ソース管理に新規プロジェクトを追加する]*** *チェック ボックスが選択されている場合、ステップ 1 と 2 を省略することができます。[ツール] メニューの [オプション] をクリックすると、ダイアログ ボックスが表示されます。プロジェクトを作成すると、InstallShield によって即時そのプロジェクトがソース コード管理システムに追加されます。*

タスク ***プロジェクトを作成した後、それをソース管理システムに追加するには、以下の手順に従います:***

1. InstallShield でプロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[ソース管理]** をポイントして **[ソース管理へ追加]** を選択します。[ソース管理に追加] ダイアログ ボックスが開きます。
3. 該当する場合はコメントを追加し、プロジェクトをチェックアウトしたままにするかどうか指定します。
4. **[OK]** をクリックします。組織内のソース管理システムへのログイン ダイアログ ボックスが開きます。
5. 自分のログイン情報を入力します。
6. プロジェクト ファイルを追加するソース管理データベースの場所を参照します。

メモ: *リンクされたファイルはソース管理に追加されません。*

ファイルをソース管理に追加すると、ほとんどのプログラムでローカル コピーは読み取り専用になります。ソース管理プログラムがこのような場合、変更を加える前にプロジェクト ファイルをチェックアウトする必要があります。

ソース管理からプロジェクトをチェックアウトする

InstallShield を使って、ソース コード管理システムから InstallShield プロジェクトをチェックアウトできます。

ヒント: *[オプション] ダイアログ ボックスの [ソース管理] タブで* ***[編集時にプロジェクトをチェックアウトする]*** *チェック ボックスが選択されている場合、ステップ 1 と 2 を省略することができます。[ツール] メニューの [オプション] をクリックすると、ダイアログ ボックスが表示されます。InstallShield でプロジェクトを変更する (ツールバーの [上書き保存] ボタンが有効になる操作を行う) と、そのプロジェクトがソース管理プログラムに追加されます。*

タスク ***ソース コード管理プログラムからプロジェクトをチェックアウトするには、以下の手順を実行します。***

1. InstallShield でプロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[ソース管理]** をポイントして **[チェックアウト]** を選択します。
3. **[オプション]** ダイアログ ボックスの **[ソース管理]** タブにある **[チェックアウトにダイアログを使用する]** を選択した場合、コメントを入力するための **[チェックアウト]** ダイアログ ボックスが表示されます。
4. 必要に応じてコメントを入力して、**[OK]** をクリックします。

ここでプロジェクトはソース管理プログラムからチェックアウトされます。このプロジェクトをソース管理プログラムに戻す前に、そのローカルコピーを編集することができます。

メモ: *InstallShield のプロジェクトを閉じたときに、プロジェクトはソース管理に自動的にチェックインされません。閉じる前に InstallShield 内部からプロジェクトをチェックインしてください。*

ソース管理にプロジェクトをチェックインする

タスク ***InstallShield 内部からソース管理プログラムにプロジェクトをチェックインするには、以下の手順に従います:***

1. InstallShield でプロジェクトを開きます。
2. **[プロジェクト]** メニューで、**[ソース管理]** をポイントして **[チェックイン]** を選択します。
3. **[チェックイン]**ダイアログ ボックスに、ファイルをチェックインするためのオプションを入力するよう指示するメッセージが表示されます。**[OK]** をクリックします。

ここでプロジェクトはソース管理プログラムにチェックインされます。これは一般に、プロジェクトファイルがチェックアウトされるまで読み取り専用に設定されることを意味します。

メモ: *InstallShield のプロジェクトを閉じたときに、プロジェクトはソース管理に自動的にチェックインされません。閉じる前に InstallShield 内部からプロジェクトをチェックインしてください。*

ソース管理からのプロジェクトのリンクの解除

タスク ***ローカル マシンにあるプロジェクトとソース管理 アプリケーションにあるプロジェクトの間のリンクを解除するには、以下の手順を実行します。***

[プロジェクト] メニューで、**[ソース管理]** をポイントして **[ソース管理からリンクを解除]** を選択します。

Microsoft Visual Studio との統合

InstallShieldでは、Microsoft Visual Studio でインストール プロジェクトを直接作成することができます。InstallShield では、Microsoft Visual Studio 内部からインストールの作成、変更、ビルドを実行することができます。

統合機能

InstallShield は Visual Studio シェル内部に完全統合されます。統合のユニークな機能には以下のようなものがあります。

- InstallShield のすべてのナビゲーションは Solution Explorer 内に表示されます。
- 各 InstallShield ビューは個別のウィンドウに表示されるのでスクロールする必要は無く、並列表示オプションも用意されています。
- InstallShield を Visual Studio の外部で実行することができます。
- InstallShield では、インストール プロジェクトは、他の Visual Studio プロジェクト出力に動的にリンクされるので、ソリューションをビルドするたびに最新のソース ファイルと共に自動更新されます。Microsoft Visual Studio で InstallShield プロジェクトを作成する
- InstallShield デバッガーによって識別されたアイテムを、Visual Studio タスク リストへ直接移動させることができます。

統合の利点

InstallShield の統合インストール オーサリング ソリューションを利用するその他の利点は、以下の通りです:

- Visual Studio ユーザー インターフェイスから離れることなく、使い慣れたナビゲーション、レイアウト オプションを使用しながら、インストールの作成やカスタマイズが可能です。
- インストールはソリューションが作成される度にソース ファイルが更新され、自動的に最新の状態が保たれます。
- インストールは、たとえばデバッグ、リリース、ビルド ディレクトリから自動的に含まれたソース ファイルといったソリューションの [ビルド構成] を反映します。
- .NET 属性と依存関係は自動的にスキャンして、インストールに含めることができます。

メモ: *Visual Studio 内部から直接 InstallShield プロジェクトを作成、編集、およびビルドするには、Visual Studio 2005 以降が必要です。InstallShield を Visual Studio 2003 以前と統合することはできません。*

Visual Studio の統合は 1 度に InstallShield の 1 バージョンとのみ可能です。システムで最後にインストールまたは修復された InstallShield のバージョンが Visual Studio の統合に使用されます。

このセクションには、次のトピックが含まれています:

- Microsoft Visual Studio で InstallShield プロジェクトを作成する
- Microsoft Visual Studio で InstallShield プロジェクトを開く
- Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する
- Visual Studio ソリューションにリファレンスを追加する
- ツールボックスを使ってダイアログ コントロールを追加する
- InstallShield ツールバーまたはコマンドを Visual Studio ツールバーに追加する
- Microsoft Visual Studio でリリースをビルドする
- .NET アセンブリをインストール プロジェクトに追加する
- Web サービスまたは Web アプリケーション プロジェクトからプロジェクト出力を追加する

Microsoft Visual Studio で InstallShield プロジェクトを作成する

InstallShield は Microsoft Visual Studio と統合されています。ソリューションとして Visual Studio ワークスペース内から、InstallShield インストールを作成することができます。

タスク ***Microsoft Visual Studio 内で、InstallShield プロジェクトを作成するには、以下の手順を実行します。***

1. **[ファイル]** メニューで、**[新規]** をポイントして **[プロジェクト]** をクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. *Visual Studio 2010 以降の場合:* **[インストールされたテンプレート]** ボックスで **[InstallShield プロジェクト]** をクリックします。次に、適切なプロジェクトの種類を選択します。

以前のバージョンの Visual Studio の場合: **[プロジェクトの種類]** ボックスで **[InstallShield プロジェクト]** をクリックします。次に、**[テンプレート]** ボックスで適切なプロジェクトの種類を選択します。

3. 必要に応じて名前とプロジェクトの場所の設定を構成します。
4. **[OK]** をクリックします。

Microsoft Visual Studio で InstallShield プロジェクトを開く

InstallShield は Microsoft Visual Studio と統合されています。ソリューションとして Visual Studio ワークスペース内から、InstallShield インストールを開くことができます。

タスク ***Microsoft Visual Studio 内で、InstallShield プロジェクトを開くには、以下の手順を実行します。***

1. **[ファイル]** メニューで **[開く]** をポイントして **[プロジェクト]** を選択して、**[プロジェクトを開く]** ダイアログを表示します。
2. InstallShield ファイルを参照して開き、選択します。
3. **[開く]** をクリックします。

メモ: *Microsoft Visual Studio 内からは、InstallShield X 以降、InstallShield DevStudio、InstallShield Dveloper、InstallShield–Windows Installer Edition、InstallShield Express バージョン 3.x 以降で作成した .ism ファイルのみを開くことができます。InstallShield Professional または InstallShield Express 3.x バージョン以前を使って作成したファイルは開けません。*

Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *スイート/アドバンスト UI*

ハイレベルなベース ディレクトリを参照する、VSSolutionFolder と呼ばれる新しい定義済みパス変数をプロジェクトで使用できます。このサポートを使うと、InstallShield プロジェクトで Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルへのスタティック リンクを含めることができます。異なるマシン上のプロジェクトで作業を行う場合、VSSolutionFolder パス変数を使用するスタティック リンクは、姉妹プロジェクトのファイルへの正しいパスを参照することができます。

VSSolutionFolder パス変数は、InstallShield が Visual Studio ソリューション内で開かれたときに自動的に定義されます。また、MSBuild を使って InstallShieldプロジェクトを含むソリューションをビルドするときにも、自動的に定義されます。ただし、Visual Studio ソリューションなしで InstallShieldプロジェクトを開いた場合、VSSolutionFolder が自動的に定義されることはありません。たとえば、InstallShieldプロジェクトを、Visual Studio を開かずに InstallShieldインターフェイスで直接開いた場合、VSSolutionFolder は定義されません。同様に、コマンドライン ツール IsCmdBld.exe や、MSBuild で .isproj ファイルを使用する場合、VSSolutionFolder は定義されません。IsCmdBld.exe を使って InstallShield プロジェクトのリリースをビルドするには、-L コマンドライン パラメーターを使って、VSSolutionFolder の値を設定します。MSBuild を使う場合は、PathVariables パラメーターを使って VSSolutionFolder の値を設定します。このプロパティは、デフォルトのターゲット ファイルが使用されたとき、ItemGroup InstallShieldPathVariableOverrides として公開されます。

InstallShieldプロジェクトで VSSolutionFolder パス変数を含むパスを持つソース ファイルを含み、それを VSSolutionFolder パス変数がサポートされていない環境でビルドすると、次のようなビルド エラーが発生する可能性があります:

- -6103: ファイル <VSSolutionFolder>¥MyFile.exe が見つかりません
- -6271: ファイル <VSSolutionFolder>¥MyFile.exe が見つかりませんでした。このファイルの MsiFileHash テーブルをビルド中にエラーが発生しました。指定した場所にファイルが存在することを確認します。

Visual Studio ソリューションにリファレンスを追加する

[ファイルとフォルダー] ビューを使って、インストール プロジェクトに Visual Studio のリファレンスを追加します。

タスク ***リファレンスを追加するには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にある **[ソリューション エクスプローラー]** で、**[ファイルとフォルダー]** をダブルクリックします。
2. **[ソース コンピューターのフォルダー]** ペインにある **Visual Studio Solution** ノードには、現在のソリューションに含まれているすべてのプロジェクトのサブノードがあります。プロジェクトをクリックしてそのプロジェクトの出力グループを表示します。出力グループは **[ソース コンピューターのファイル]** ペインに表示されます。
3. インストール プロジェクトへ出力リファレンスを追加するには、**[インストール先コンピューターのフォルダー]** ペインのターゲット フォルダーへ出力をドラッグ アンド ドロップします。

メモ: *プロジェクトのリファレンスを参照するには、プロジェクトを右クリックして [プロジェクト出力] を選択します。出力グループがどのファイルに解決するかを示す [出力] ダイアログ が表示されます。*

ツールボックスを使ってダイアログ コントロールを追加する

Microsoft Visual Studioでセットアップ プロジェクトを開いている際にダイアログ コントロールを追加するには、[ツールボックス] を使用します。

Toolbox へのアクセス

タスク ***InstallShield Toolbox へアクセスするには、以下の手順を実行します。***

1. 以下のいずれかを実行します。
   - **[ビュー]** メニューで、**[ツールボックス]** をクリックします。
   - **CTRL+ALT+X** を押します。
2. 閉じたままの状態であれば、**[InstallShield ダイアログ]** タブをクリックします。

コントロールを追加する

タスク ***ダイアログにコントロールを追加するには、以下の手順を実行します。***

1. ダイアログに追加するコントロールの種類をクリックします。
2. ダイアログ上でクリックしてマウスボタンを押したまま、コントロールのサイズまでドラッグします。高さと幅のプロパティを設定して、コントロールのサイズを微調整することができます。

   マウスボタンを離すとカーソルが「ツールの選択」に変わるので、そこで新しいコントロールの大きさを変更することができます。
3. コントロールのプロパティの設定。詳細については、「基本の MSI プロジェクトでダイアログ レイアウトを編集する」または「InstallScript または InstallScript MSI プロジェクトのダイアログ レイアウトを編集する」を参照してください。

固定状態モードを使用する

さらに toolbox コントロールには、*固定状態* モードがあり、Toolbox へ戻らずに追加コントロールを作成することが可能です。固定状態モードを有効にするには、CTRL キーを押しながらカーソルを toolbox からダイアログへドラッグします。

自動非表示を利用する

ダイアログ Toolbox には自動非表示機能があり、利用していない時に toolbox を非表示にすることができます。[自動非表示] を有効にするには、toolbox の右上にある [自動非表示] ボタンをクリックします。有効になっている場合、toolbox は画面の端に隠れます。

InstallShield ツールバーまたはコマンドを Visual Studio ツールバーに追加する

InstallShield ツールバーおよび個別のツールバー コマンドボタンを Microsoft Visual Studio ワークスペースに追加することができます。

InstallShield ツールバーを追加する

タスク ***InstallShield ツールバーを追加するには、以下の手順を実行します。***

1. ツールバーの任意の場所を右クリックしてツールバーオプションを表示します。
2. ツールバーを選択して Visual Studio ワークスペースの上部に追加します。InstallShield で提供されているツールバーは、InstallShield、InstallScript、InstallShield Layout、および、MSI Debugger です。

InstallShield ツールバー コマンド ボタンを追加する

タスク ***個別のコマンドボタンをツールバーへ追加するには、以下の手順を実行します。***

1. ツールバーの任意の場所を右クリックしてツールバーオプションを表示します。
2. リストの下から [**カスタマイズ**] を選択します。[**カスタマイズ**] ダイアログが表示されます。
3. **[コマンド]** タブをクリックします。
4. **[カテゴリ]** リストからカテゴリを選択して、特定のカテゴリ内で利用可能なコマンドを表示します。
5. **[コマンド]** リストからコマンドボタンをクリックして、それをツールバーへドラッグします。

メモ: *InstallShield Layout および MSI Debugger ツールバーは、個別のツールバー ボタンとしては提供されていません。*

Microsoft Visual Studio でリリースをビルドする

プロジェクト: *以下の情報の一部はプロジェクト固有の情報で、適切な場合には、それが表示されています。*

Visual Studio 内での InstallShield プロジェクトのリリースのビルドは、InstallShield でのリリースのビルドと異なります。Visual Studio 内でリリースをビルドするとき、インストール プロジェクトを含むソリューション全体をビルドするオプションと、インストール プロジェクトのみをビルドするオプションがあります。

Visual Studio 内で InstallShield プロジェクトを作成するとき、プロジェクトにはデフォルトで 2 つのリリース (デバッグとリリース) が含まれます。これらのデフォルト リリースを使用するか、新しいリリースを作成できます。リリースをビルドする前に、リリースをソリューション構成と関連付ける必要があります。Visual Studio の構成マネージャーを使って、リリースをソリューション構成と関連付けます。

プロジェクト: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、各リリースは 1 つの製品構成に属します。製品構成により、製品名、製品コード、パッケージコードなど、類似した設定を共有するリリースを一緒にグループ化することができます。これらのプロジェクトの種類でリリースをビルドするとき、常にデフォルトの製品構成が使用されます。異なる製品構成でリリースをビルドする場合は、その製品構成をデフォルトに設定します。*

デフォルトの製品構成を選択する (基本の MSI、InstallScript MSI、および マージ モジュール プロジェクトのみ)

基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトでリリースをビルドする場合、そのリリースがデフォルトまたは *アクティブ* な製品構成でなくてはなりません。これは、[リリース] ビューのチェック ボックス アイコンで示されます:

タスク ***特定の製品構成を、InstallShield プロジェクトのデフォルトの構成として指定するには、以下の手順に従います:***

1. **[メディア]** の下にある **[ソリューション エクスプローラー]** で、**[リリース]** をダブルクリックします。
2. **[リリース]** エクスプローラーで、ビルドするリリースを含む製品構成を右クリックしてから、**[デフォルトの製品構成]** をクリックします。

ヒント・ *新しい製品構成を作成するには、**[リリース]** エクスプローラーを右クリックしてから、**[新しい製品構成]** をクリックします。製品構成を作成した後に、それをデフォルトの製品構成として設定できます。*

リリースのビルド

Visual Studio では、インストール プロジェクトを含むソリューション全体をビルドするか、インストール プロジェクトのみをビルドすることができます。

タスク ***Visual Studio 内部でリリースをビルドするには、以下の手順に従います:***

1. Visual Studio の構成マネージャーを使ってプロジェクト構成を適切なソリューション構成にマップします。
   a. **[ビルド]** メニューで **[構成マネージャー]** をクリックします。

      別の方法として、**[標準]** ツールバーで **[ソリューション構成]** リストの中から **[構成マネージャー]** を選択します。

   b. **[アクティブ ソリューション構成]** リストから構成を選択します。
   c. **[プロジェクト コンテキスト]** ボックスで、プロジェクト構成を **[構成]** 列にある適切なリリースにマップします。

      メモ・ *[ビルド] チェック ボックスの選択がクリアされているときにソリューションをビルドすると、選択解除されたプロジェクト構成はビルドされません。*

   d. **[閉じる]** ボタンをクリックします。

2. **[ビルド]** メニューで、該当するコマンドを選択します:

テーブル 1 · [ビルド] メニューのコマンド

コマンド	説明
ビルド ソリューション	構成マネージャーで、ソリューションに含まれている各プロジェクトに対して何がマップされているかに従って全ソリューションをビルドします。
ビルド *[インストール プロジェクト名]*	構成マネージャーの [構成] 列で指定されているリリースに従って、インストール プロジェクトのみをビルドします。このオプションは、[ソリューション エクスプローラー] でインストール プロジェクトが選択されているときに有効です。

ヒント· *ソリューション全体をビルドするには、CTRL+SHIFT+B も使用できます。*

.NET アセンブリをインストール プロジェクトに追加する

InstallShield では、.NET アセンブリ ファイルをコンポーネントに追加することによって、インストール プロジェクトに .NET アセンブリを追加することができます。

タスク ***.NET アセンブリをインストール プロジェクトに追加する方法としては、次の方法がお勧めです。***

1. **[セットアップのデザイン]** ビューで、.NET アセンブリを格納するコンポーネントを作成します。
2. (Windows Installer ベースのプロジェクトのみ) ファイルを右クリックして、コンテキスト メニューから **[キー ファイルの設定]** を選択することで、ファイルをコンポーネントのキー ファイルにします。
3. 次のいずれかを行って、プロジェクトをスキャンして依存関係の有無を確認します。
   - **"ビルド時に .NET をスキャン"**コンポーネント プロパティを、[依存関係とプロパティ] に設定します。この場合、.NET アセンブリのマニフェスト(Manifest)、アプリケーション、および関連エントリはプロジェクトファイルに入力されていません。その代わり、これらの情報はビルド時に .msi ファイルへ追加されます。
   - スタティック スキャン ウィザード を起動します。この場合、.NET アセンブリの値はスキャン中に追加されます。コンポーネントの **[詳細設定]** にある **[アセンブリ]** ノードでこれらの値を編集することができます。

Web サービスまたは Web アプリケーション プロジェクトからプロジェクト出力を追加する

InstallShield には、拡張 Web サービスサポートが搭載されています。Web サービスまたは Web アプリケーション プロジェクトからプロジェクト出力を追加すると、プロジェクトを Web サービスとして追加するよう求めるメッセージが InstallShield によって表示されます。[いいえ] を選択すると、プロジェクト出力は通常通り追加されます。[はい] を選択すると、InstallShield は次の操作を行います。

1. IISROOTFORDER というインストール先フォルダーを作成。
2. 次の Visual Studio プロジェクト出力を IISROOTFOLDER に配布。
   [コンテンツ ファイル] は [IISROOTFOLDER]{VSIPProjectName} に移動
   [Primary Output] は [IISROOTFOLDER]{VSIPProjectName}¥bin″ に移動
3. ターゲットに [IISROOTFOLDER]{VSIPProjectName} を持つ IISVirtualDirectory を作成。

Microsoft Visual Studio Team Foundation Server との統合

Microsoft Visual Studio Team Foundation Server (TFS) は、製品のチーム開発においてタスクを共同作業で進めることができるツールとテクノロジのセットです。 は、Team Foundation Serve との統合をサポートします。
InstallShield では、Team Foundation Server との統合をサポートします。統合による主要な機能は以下のとおりです:

- **ソース管理**—ソース管理エクスプローラーを使って、InstallShield プロジェクトを Team Foundation のバージョン管理と統合することで、InstallShield プロジェクトと Visual Studio ソリューションへの変更を管理できます。
- **自動ビルド**—Team Foundation ビルドを使って、InstallShield プロジェクトと Visual Studio ソリューションを定期的にコンパイル、テスト、およびデプロイできます。インストールはソリューションが作成される度にソース ファイルが更新され、自動的に最新の状態が保たれます。
- **プロジェクト管理**— InstallShield プロジェクトと Visual Studio ソリューションのバグ、タスク、およびプロジェクト ドキュメントなどの作業項目を管理します。プロジェクト ステータスは、チーム全体で Team System Web Access 内部、および Team Explorer 内部から使用できます。

統合の要件

InstallShield を Team Foundation Server と統合するには、InstallShield プロジェクトの作成、アップデート、またはビルドを行う各マシンに InstallShield をインストールします。つまり、InstallShield プロジェクトの作成およびアップデートを行う各マシンに InstallShield のインストールが必要です。Team Foundation Server に格納されている InstallShield プロジェクトのビルド エージェントとして指定されているマシン上にも をインストールする必要があります。InstallShield のライセンスの詳細については、InstallShield 使用許諾契約書 (EULA) を参照してください。

メモ: *Standalone Build は、InstallShield の完全版をビルド マシンにインストールしなくても InstallShield プロジェクトをビルドすることができるビルドエンジンです。Standalone Build をお持ちの場合、Team Foundation Server のビルド エージェントとして指定されているマシン上に、それをインストールできます。*

一部の種類のプロジェクトおよびソリューションをビルド エージェントがビルドするために、ビルド マシンに追加のソフトウェアをインストールすることが必要な場合もあります。たとえば、C++ プロジェクトをビルドする場合、C++ コンパイラやその他の依存関係も必要なため、ビルドマシンに Visual Studio をインストールする必要があります。

複数のビルド エージェントを使って Team Foundation Server プロジェクトをビルドする場合、特定のビルド タグを InstallShield が搭載されているマシン上の任意のエージェントに割り当てることができます。また、InstallShield プロジェクト用に作成された各ビルド定義に特殊なビルド タグを適用することもできます。こうすることで、InstallShield がインストールされているビルド マシンによってのみ InstallShield インストールのビルドに使用されます。ビルド タグの作成およびエージェント、ならびにビルド定義への割り当てについての詳細は、Visual Studio Team Foundation Server ドキュメントを参照してください。

64 ビットのビルド マシン上でビルドをキューに配置する場合、InstallShield.Tasks.dll ファイル (32 ビット ファイル) をロードするのに 32 ビット バージョンの MSBuild が使用されるように InstallShield プロジェクトのビルド定義を構成してしてください。そうしなければ、InstallShield.Tasks.dll ファイルをロードできなかったことを通知するビルド エラーが発生します。32 ビット バージョンの MSBuild を選択するには、Team Explorer でビルド定義の [プロセス] タブをクリックします。次に、[詳細] ノードの下にある "MSBuild プラットフォーム" 設定で [x86] を選択します。32 ビットのビルド マシンを使用している場合、"MSBuild プラットフォーム" 設定では [自動] または [x86] のどちらかを選択できます。

InstallShield と Visual Stuio が搭載された同じマシン上に Team Explorer をインストールした場合、Visual Studio で開かれている InstallShield プロジェクト内から Team Explorer を使用できます。これで、次のようなタスクを行うことができます:

- InstallShield プロジェクトでの作業中にソース管理エクスプローラーを使用。
- InstallShield プロジェクトと Visual Studio ソリューションのビルドを構成。
- 新しいビルドをキューに配置。

InstallShield プロジェクトを含むソリューションのビルドをキューに配置すると、ビルドされたインストールはドロップ フォルダー内の Install サブフォルダーにコピーされます。InstallShield ビルドが Team Foundation ビルドの下で実行されていることを検出すると、インストールはソリューションの最終出力の場所 (OutDir) にコピーされます。つまり、これはバイナリ ディレクトリで、Team Foundation ビルド プロセスの終わりに、ドロップ フォルダーにコピーされます。

Visual Studio と統合しないで InstallShield 内部で InstallShield プロジェクトの作業を行うときに、Team Foundation ソース管理を使用することも可能です。この機能を使うには、MSDN Web サイトの Visual Studio Gallery から取得できる Team Foundation Server MSSCCI Provider のインストールが必要です。Team Foundation Server MSSCCI Provider をインストールした後、InstallShield 内部において InstallShield プロジェクトを Team Foundation ソース管理に追加、InstallShield プロジェクトを Team Foundation ソース管理からチェックアウト、およびプロジェクトを Team Foundation ソース管理にチェックインすることができます。InstallShield 内部におけるソース管理の統合についての詳細は、「ソース コード管理を使用する」を参照してください。

InstallShield プロジェクトを Team Explorer に追加する

InstallShield と Visual Studio が搭載されている同じマシン上に Team Explorer をインストールした場合、Team Explorer を使って、InstallShield プロジェクト (.ism と .isproj ファイル、および任意の対応する InstallScript ファイルと SQL スクリプト) を Team Foundation Server に追加できます。

Visual Studio 内から InstallShield プロジェクトを作成したときに、プロジェクトをソース管理エクスプローラーに追加するには、[新規プロジェクト] ダイアログ ボックスで **[ソース管理に追加]** チェック ボックスを選択します。この作業を行った場合、その他のファイルを追加するのと同じ方法で、後で InstallScript ファイルと SQL スクリプト ファイルをソース管理に追加する必要がある可能性もあります。

既存の InstallShield プロジェクトをソース管理エクスプローラーに追加するには、その他のプロジェクトを追加するのと同じ要領でプロジェクトを追加します。(ソリューション エクスプローラーで、InstallShield プロジェクトを右クリックしてから、[ソリューションをソース管理に追加] をクリックします。)

11

インストールの開発およびビルド プロセスの自動化

オートメーション インターフェイスとは、大まかに定義すると、プログラミング用インターフェイスの 1 つで、COM をサポートする任意のプログラミング言語からアクセスすることができます。最も重要な特徴として、VBScript や JScript などの遅滞バインディング スクリプト言語から利用できる点があげられます。InstallShield でオートメーション インターフェイスは、InstallShield プロジェクト ファイルへのオートメーション インターフェイスを意味します。

InstallShield では、InstallShield インターフェイスを直接開いて異なるビューで変更を加えることなく、オートメーション インターフェイスを利用してビルド プロセスを自動化することができます。オートメーション インターフェイスを通して、メソッドの呼び出し、プロパティの設定、コレクションへのアクセスを行い、InstallShield インターフェイスの場合と同様にプロジェクトを開き、その機能やコンポーネント データを変更することができます。さらに、InstallShield オートメーション インターフェイスの Build メソッドを使用してリリースをビルドすることもできます。

この章の次のセクションでは、オートメーション インターフェイスについて、およびビルド処理の自動化におけるその他の側面についての詳細が説明されています:

- VBScript を使用したサブフォルダーの機能へのリンク
- オートメーション インターフェイスを使用した文字列エントリのインポートとエクスポート
- オートメーション インターフェイスを使用したプロジェクト レポートの実行
- コマンドラインでソース コード管理を使用する
- ビルドのステータス イベント
- 64 ビット システム上でオートメーション インターフェイスを使用する

VBScript を使用したサブフォルダーの機能へのリンク

この VBScript の例は、サブフォルダーへのダイナミック リンクをシミュレートします。.ini ファイルを編集すると、リンクするフォルダーやサブフォルダー、ファイルのコンポーネントの作成方法、および新規コンポーネントと関連付ける機能などをスクリプトに指示できます。

以下は、VBScript の例を使用するときの要件です。

- .vbs ファイルを実行するためには、Windows Scripting Host が必要です。
- オートメーション インターフェイスおよび .ini ファイル リーダーを使用する前に、システムに InstallShield のインストールが必要です。
- スクリプトには 2 つのコマンドライン引数が求められます。
  - 既存のセットアップ プロジェクトへの完全修飾パス (.ism ファイル)。
  - 指定のフォルダーのファイルやサブフォルダーで追加する機能の説明がある .ini ファイルへの完全修飾パス (この例では Links.ini)。さらに、各機能は、各フォルダーおよびサブフォルダー内のファイル用に作成されるコンポーネントのプロパティを説明する異なる .ini ファイル (Template.ini) をポイントすることができます。

この例のソース ファイルは、以下のディレクトリにあります。

InstallShield Program Files フォルダー¥Samples¥WindowsInstaller¥Automation Interface¥Add Files and Components

各ファイルを以下に説明します。

LinkToFeature.vbs

LinkToFeature.vbs ファイルのスクリプトは、関連付けられた Links.ini ファイルを読み取り、指定されたフォルダーおよびサブフォルダーから指定された機能へそのファイルを追加するように設計されています。このスクリプトにより、Template.ini に構築したモデルに基づいて、機能がコンポーネント内に編成されます。スクリプトを開始するには、セットアップ プロジェクトへのパスと Link.ini へのパスにそれぞれ以下のコマンドラインでスクリプトを渡します。

LinkToFeature.vbs "C:¥MySetups¥Malaprop.ism" "C:¥Automation Interface¥Links.ini"

LinkToFeature.vbs の該当するセクションを調べて、その動作をはっきり確認することができます。コマンドラインで指定した .ism ファイルを開いた後、スクリプトにより、各機能のコレクションが 53 行目で始まるプロジェクトで作成されます。このコレクションは、Links.ini にリストされる機能がプロジェクトに存在するかどうか確認するために後で使用されます。

```
' Scripting Dictionary は Scrrun.dll で定義されます
Dim pFeatCol
Set pFeatCol = CreateObject("Scripting.Dictionary")

' プロジェクト内にあるすべての機能のコレクションを作成します
For Each ISWIFeature In ISWIProject.ISWIFeatures
    pFeatCol.Add ISWIFeature, ISWIFeature.Name
    DoSubFeature ISWIFeature, pFeatCol
Next
```

次に、LinkToFeature.vbs が IniReader を使用して InstallShield Links.ini を開きます。70 行目にあるコメントにあるように、必要な場合にはこのユーティリティを使用できますが、そうでない場合は、既存のものを使用します。Links.ini が開いた後、残りのスクリプトは 80 行目で始まり 202 行目で終わる For ループによりコントロールされます。

For Each pLinksSection In LinksIniFile.Sections

このループの目的は、Links.ini の各セクションを読み取ることです。各セクションには、機能の名前が含まれており、そこに新しいコンポーネントを追加したり、そのファイルやサブフォルダーがあるフォルダーを追加できます。各キーの値を取得した後に、ネストした For ループが 102 行目から 201 行目にわたり存在し、コレクション pFeatCol における各機能をループして、Links.ini の機能の名前と比較します。

```
For Each ISWIFeature In pFeatCol
    ' Links.ini にある機能とセットアップ プロジェクトにある機能とが一致するか確認します。
    If (ISWIFeature.Name = linksFeature) Then
```

このループは、スクリプトの残りのほとんどの部分に使用されているため、実行するすべての処理は現在の機能に作用すると考えることができます。次に、ループにより、この Links.ini セクションの Action キーが Delete に設定されているか、またその場合に機能のコンポーネントすべてを削除するかどうか確認します。

```
If (linksAction = "Delete") Then
    For Each ISWIComponent In ISWIFeature.ISWiComponents
        ISWIProject.DeleteComponent ISWIComponent
    Next
End If
```

27 行目で、スクリプトにより指定されたフォルダーと各サブフォルダーのコレクションが作成されます。

```
Set pFolders = CreateObject("Scripting.Dictionary")
pFolders.Add folder, folder.Path
GetSubFolders pFolders, folder
```

136 行目から 163 行目で、スクリプトは現在の機能に追加されるコンポーネントすべてのテンプレートを含む Template.ini ファイルを開きます。

```
Set iniFile = CreateObject("IniReader.IniFile")
iniFile.Load (linksTemplateFile)
```

LinkToFeature.vbs は、行番号 169 で始まって 199 で終わる For loop で実際の処理を行います:

```
' ルート フォルダーとそのすべてのサブフォルダーににコンポーネントを作成します。
For Each folder In pFolders
```

Template.ini の記述によって、コンポーネント名が形成され、プロジェクトに追加されるほか、現在の機能と関連付けられ、コンポーネントのプロパティが設定されてから、現在のフォルダーにあるファイルをコンポーネントに追加します。

最後に、その .vbs ファイルによりセットアップ プロジェクトが保存され、ファイルが閉じます。

Links.ini

この .ini ファイルの各セクションには、以下のキーが含まれています。LinkToFeatue.vbs はそれらのキーを読み取り、プロジェクトの各機能に追加するフォルダーやサブフォルダーを決定をします。

テーブル 1・サンプル ファイルの .ini ファイル キー

Key	説明
[LinkN]	; 一意のセクション名
Feature=FeatureName	; プロジェクトに存在する機能の名前
Source=C:¥Data Files¥Executables	; ルート フォルダーへのパス
TemplateFile=Template.ini	; コンポーネント テンプレート .ini ファイルへのパス
ComponentTemplate=Template1	; この機能のコンポーネントに対する Template.ini 内のセクション名

テーブル 1・サンプル ファイルの .ini ファイル キー (続き.)

Key	説明
Action=Delete	; Delete は LinkToFeature.vbs に対し、この機能のコンポーネントすべてを再作成される前に永久に削除するよう指示します。コンポーネントを破棄したくない場合は、この値を空白にしてください。

Template.ini

Template.ini の各セクションには、機能と関連するすべてのコンポーネントを作成するための情報が含まれています。Links.ini の各セクションは特定のコンポーネント テンプレート ファイルおよびセクションを指示する必要があります。テンプレート ファイルおよびセクションを以下に説明します。

テーブル 2・Template.ini ファイル セクション

セクション	説明
[TemplateName]	; 特定のテンプレートを識別する一意のセクション名
Name=DefaultName	; この値は、DLS_DefaultName_*N* における形式で、コンポーネントの名前のビルドに使用されます。
Destination=[ProgramFilesFolder]	; これはコンポーネントのルート インストール先フォルダーです。サブフォルダーの名前はルートに追加されると、ファイルが正しいフォルダーにインストールされます。
Languages=1033	; コンポーネントの言語すべての 10 進数の値を一覧表示し、カンマで区切ります。
Condition=VersionNT	; 各コンポーネントに添付する条件を作成します。

オートメーション インターフェイスを使用した文字列エントリのインポートとエクスポート

InstallShield は、文字列エントリを プロジェクト ファイル (.ism) の ISString テーブルに格納します。InstallShield では、言語の文字列エントリをテキスト ファイルにエクスポートして、翻訳者に渡すことができます。文字列を翻訳した後、テキスト ファイルを InstallShield プロジェクト ファイルにインポートできます。

InstallShield の [文字列エディター] ビューでは、特定の言語用の文字列エントリを手動でエクスポートおよびインポートできます。この作業を自動化するために、InstallShield オートメーション インターフェイスが用意されています。

オートメーション インターフェイスの機能の 1 つは、プロジェクトから文字列エントリをインポートおよびエクスポートができるということです。以下は、具体的な実行例です。

サンプル VBScript ファイルの実行

このサンプルを実行するには、まず下記のサンプル コードを ImportExportStrings.vbs という名前のファイルにコピーする必要があります。そのあと、コマンドラインを使って、文字列エントリのエクスポートまたはインポートを実行します。

言語の文字列エントリをエクスポートする

サンプル コードを使用して文字列エントリをエクスポートするには、コマンドラインから以下のコマンドを入力します。

```
wscript.exe ImportExportStrings.vbs "C:¥MyProject.ism" "C:¥1033.txt" "1033" "E"
```

言語の文字列エントリをインポートする

サンプル コードを使用して文字列エントリをインポートするには、コマンドラインから以下のコマンドを入力します。

```
wscript.exe ImportExportStrings.vbs "C:¥MyProject.ism" "C:¥1033.txt" "1033" "I"
```

サンプル ImportExportStrings.vbs コード

```
'//////////////////////////////////////////////////////////////////////////
'BEGIN ImportExportStrings.vbs
'
このユーティリティは文字列エントリのインポートとエクスポートを支援するために設計されました。
'
'
' ファイル名: ImportExportStrings.vbs
'
' 説明: サンプル VBScript ファイルが文字列エントリをテキスト ファイルにインポートまたはエクスポートします
'
' 使用法: オブジェクトをこのスクリプトで利用するには、次のアイテムがシステム上に
'       存在することを確認してください:
'       1.InstallShield が、エンドユーザー オートメーションが利用できるように
'       インストールされていなければなりません。
'       2.Windows Scripting Host (Wscript.exe) がインストールされていなくてはなりません。
'       3.スクリプトには次のコマンドライン引数がこの順序で
'       必要です:
'         a. 既存 .ism ファイルへの完全修飾パス。
'         b. 文字列エントリを含むテキスト ファイルへの完全修飾パス
'         c. 10 進法言語 ID
'         d. Import または Export
'
'//////////////////////////////////////////////////////////////////////////


If Wscript.Arguments.Count < 1 Then
    Wscript.Echo "InstallShield Subfolder Utility" & _
        vbNewLine & "1 番目の引数は、.ism ファイルへの完全パス" & _
        vbNewLine & "2 番目の引数は、文字列テキスト ファイルへの完全パス" & _
        vbNewLine & "3 番目の引数は、10 進数言語 ID" & _
        vbNewLine & "4 番目の引数は、エキスポートの E、またはインポートの I"
    Wscript.Quit 1
End If

' エンドユーザー オートメーション オブジェクトを作成します
```

```
Dim ISWIProject
Set ISWIProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject"): CheckError

' コマンドラインで指定したプロジェクトを開きます
ISWIProject.OpenProject Wscript.Arguments(0): CheckError

if(Wscript.Arguments(3)="E")then
    StringsExport ISWIProject,Wscript.Arguments(1),Wscript.Arguments(2)
elseif(Wscript.Arguments(3)="I")then
    StringsImport ISWIProject,Wscript.Arguments(1),Wscript.Arguments(2)
end if

' プロジェクトを保存して閉じます
ISWIProject.SaveProject: CheckError
ISWIProject.CloseProject: CheckError

'//////////////////////////////////////////////////////////////////////////
' 言語の文字列エントリをエクスポートします
Sub StringsExport(byref p,byval path, byval language)
    p.ExportStrings path, Date , language
    Wscript.Echo "言語のエキスポート文字列エントリ " & language & " エキスポート先 " & path
End Sub

'//////////////////////////////////////////////////////////////////////////
' 文字列エントリをインポートします
Sub StringsImport(byref p,byval path, byval language)
    p.ImportStrings path, language
    Wscript.Echo "言語のインポート文字列エントリ " & language & " インポート元 " & path
End Sub

'//////////////////////////////////////////////////////////////////////////
Sub CheckError()
    Dim message, errRec
    If Err = 0 Then Exit Sub
    message = Err.Source & " " & Hex(Err) & ": " & Err.Description
    Wscript.Echo message
    Wscript.Quit 2
End Sub

'END ImportExportStrings.vbs
```

オートメーション インターフェイスを使用したプロジェクト レポートの実行

次の VBScript 例は、セットアップ内のオブジェクトにアクセスして、機能、コンポーネント、ファイル、および様々な種類のファイルの合計数についてのレポートを表示します。出力は、メッセージ ボックスに表示されますが、この情報をテキスト ファイルに書き込むこともできます。

要件

次の要件が満たされなくてはなりません。

- .vbs ファイルを実行するためには、Windows Scripting Host が必要です。(このスクリプトを変更して、簡単に Visual Basic プログラムで実行することができます。)
- InstallShieldオートメーション インターフェイスを使用する前に、InstallShield のインストールが必要です。
- スクリプトを実行する前に、文字列変数 sProj に、既存のセットアップ プロジェクト (.ism ファイル) の完全修飾パスを割り当てます。

スクリプト例

```
Option Explicit

' オートメーション オブジェクトを作成します
Dim pProj
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")

' セットアップ プロジェクトを開きます。IDE で既に開かれていないことを確認してください
Dim sProj
sProj = "C:¥MySetups¥TestProject.ism"
pProj.OpenProject sProj

' 機能、コンポーネントおよびファイルを表示する文字列をビルドします
Dim sItems
sItems = "プロジェクトレポート" & vbNewLine
sItems = sItems & "プロジェクト: " & sProj & vbNewLine
sItems = sItems & "機能の数: " & pProj.ISWiFeatures.Count & vbNewLine
sItems = sItems & "コンポーネントの数: " & pProj.ISWiFeatures.Count & vbNewLine

' ファイルとファイルの種類のカウンターを指定および初期化します
Dim i, nEXE, nDLL, nOCX
i = 0
nEXE = 0: nDLL = 0: nOCX = 0

' コンポーネントをループしてファイル数と各ファイル名を取得します
Dim sFileName
Dim pComp, pFile
For Each pComp In pProj.ISWiComponents
  For Each pFile In pComp.ISWiFiles
    i = i + 1
    sFileName = pFile.DisplayName
    If Instr (1, sFileName, ".exe", vbTextCompare) > 0 Then
      nEXE = nEXE + 1
    ElseIf Instr (1, sFileName, ".dll", vbTextCompare) > 0 Then
      nDLL = nDLL + 1
    ElseIf Instr (1, sFileName, ".ocx", vbTextCompare) > 0 Then
      nOCX = nOCX + 1
    End If
  Next
Next

sItems = sItems & "ファイルの数: " & i & vbNewLine
sItems = sItems & vbTab & "EXEs: " & nEXE & vbNewLine
sItems = sItems & vbTab & "DLLs: " & nDLL & vbNewLine
sItems = sItems & vbTab & "OCXs: " & nOCX & vbNewLine

' レポートを表示します
```

```
Wscript.Echo sItems

‘ プロジェクトを閉じます
pProj.CloseProject
```

コマンドラインでソース コード管理を使用する

ソース管理システムに格納されているインストール プロジェクトへのアクセスは、ソース管理アプリケーションへのコマンドラインの呼び出しと、InstallShield の多数の機能へコマンドラインのようにアクセスできる InstallShieldオートメーション インターフェイス の利用を制限することで可能です。ソース管理のコマンドライン サポートがあると、プロジェクトの最新バージョンのチェック イン、チェック アウト、および取得を行うことができます。

以下に、この機能の使用例を 2 つ示します。いずれの例も、Microsoft Visual SourceSafe をソース管理アプリケーションとして使用し、VBScript を使用して書かれます。

いずれかの例をコピーして .vbs ファイルに貼り付け、環境に有効になるようにパスを変更して、実行することもできます。

メモ: *InstallShield 製品の以前のバージョンでは、ソース管理 (.isv) ファイルを InstallShield プロジェクト (.ism) ファイルに変換してからまた .isv ファイルに戻す必要がありました。InstallShield 製品の最新のバージョンでは、.ism ファイルをソース管理アプリケーション用にテキスト ファイルとして格納できるため、これを行う必要はなくなりました。*

最新バージョンの取得とインストールのビルド

```
    Const VSSFLAG_USERRONO   = 1
    Const VSSFLAG_TIMEMOD    = 8
    Const VSSFLAG_REPREPLACE = 128

    Const PROJECT_SCC_INI_LOC = "¥¥Server¥srcsafe.ini"
    Const PROJECT_SCC_FOLDER = "$/MyFiles/"
    Const PROJECT_SCC_BASE_NAME = "MyProject"
    Const PROJECT_SCC_LOCAL_FOLDER = "C:¥Project"

    ' Microsoft Visual SourceSafe への参照を作成
    Set VSS = CreateObject("SourceSafe")
    ' VSS データベースをポイント
    VSS.open PROJECT_SCC_INI_LOC

    ' プロジェクト ファイルを取得
    Set VSSISVFile = VSS.VSSItem (PROJECT_SCC_FOLDER + PROJECT_SCC_BASE_NAME + ".ism")
    VSSISVFile.Get PROJECT_SCC_LOCAL_FOLDER + PROJECT_SCC_BASE_NAME + ".ism", SSFLAG_TIMEMOD +
VSSFLAG_USERRONO + VSSFLAG_REPREPLACE

    'すべての残りのファイルを取得
    Set VSSIDTFolder = VSS.VSSItem (PROJECT_SCC_FOLDER + PROJECT_SCC_BASE_NAME)
    VSSIDTFolder.LocalSpec = PROJECT_SCC_LOCAL_FOLDER + PROJECT_SCC_BASE_NAME
    For Each VSSObj In VSSIDTFolder.Items(False)
        VSSObj.Get , VSSFLAG_TIMEMOD + VSSFLAG_USERRONO + VSSFLAG_REPREPLACE
    Next
```

```
strFileBasePath = PROJECT_SCC_LOCAL_FOLDER + PROJECT_SCC_BASE_NAME + ".ism"
strCmdLine = "ISCmdBld.exe –p """ + strFileBasePath + ".ism"""
' インストールをビルド
Set wshshell = CreateObject("Wscript.Shell")
RunCmdLine = wshshell.Run(strCmdLine, 1, True)
```

チェック アウト、変更、およびチェック イン

```
Const VSSFLAG_USERRONO = 1
Const VSSITEM_FILE     = 1

Const PROJECT_SCC_INI_LOC = "¥¥Server¥srcsafe.ini"
Const PROJECT_SCC_FOLDER = "$/MyFiles/"
Const PROJECT_SCC_BASE_NAME = "MyProject"
Const PROJECT_SCC_LOCAL_FOLDER = "C:¥Project"

' Microsoft Visual SourceSafe への参照を作成
Set VSS = CreateObject("SourceSafe")
' VSS データベースをポイント
VSS.open PROJECT_SCC_INI_LOC

' プロジェクト ファイルをチェックアウト
Set VSSISVFile = VSS.VSSItem (PROJECT_SCC_FOLDER + PROJECT_SCC_BASE_NAME + ".ism")
VSSISVFile.CheckOut , PROJECT_SCC_LOCAL_FOLDER + PROJECT_SCC_BASE_NAME + ".ism", VSSFLAG_USERRONO

'残りのすべてのファイルをチェック アウト
Set VSSIDTFolder = VSS.VSSItem (PROJECT_SCC_FOLDER + PROJECT_SCC_BASE_NAME)
VSSIDTFolder.LocalSpec = PROJECT_SCC_LOCAL_FOLDER + PROJECT_SCC_BASE_NAME
For Each VSSObj In VSSIDTFolder.Items(False)
   If VSSObj.Type = VSSITEM_FILE Then
      VSSObj.CheckOut , , VSSFLAG_USERRONO
   End If
Next

' InstallShield オートメーション インターフェイスへの参照を作成
Set m_ISWiProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
strFileBasePath = PROJECT_SCC_LOCAL_FOLDER + PROJECT_SCC_BASE_NAME + ".ism"
' プロジェクトを開く
m_ISWiProject.OpenProject strFileBasePath
' 機能を追加
m_ISWiProject.AddFeature "Robofeature1"
' プロジェクトを保存
m_ISWiProject.SaveProject
' プロジェクトを閉じる
m_ISWiProject.CloseProject

' プロジェクト ファイルをチェックイン
Set VSSISVFile = VSS.VSSItem (PROJECT_SCC_FOLDER + PROJECT_SCC_BASE_NAME + ".ism")
VSSISVFile.CheckIn

'残りのすべてのファイルをチェックイン
Set VSSIDTFolder = VSS.VSSItem (PROJECT_SCC_FOLDER + PROJECT_SCC_BASE_NAME)
For Each VSSObj In VSSIDTFolder.Items(False)
   If VSSObj.Type = VSSITEM_FILE Then
      VSSObj.CheckIn "チェックイン コメント"
   End If
```

Next

ビルドのステータス イベント

オートメーション インターフェイスはビルド ステータス イベント (ProgressIncrement、ProgressMax、および StatusMessage) を含みます。これらのイベントを使ってビルドの進行状況を表示し、ステータスの更新状況を確認できます。

基本の MSI プロジェクト、InstallScript プロジェクトおよび InstallScript MSI プロジェクトの例

次の Visual Basic 6 コードは、これらのイベントを基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用する方法をデモンストレーションします。

```
Public WithEvents pISWiRelease As ISWiAutoAutomation Interface Version.ISWiRelease

Private Sub Foo()
    Dim pISWiProject As IswiAutoAutomation Interface Version.ISWiProject
    Set pISWiProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
    pISWiProject.OpenProject "C:¥InstallShield 2021 Projects¥My Project Name-1.ism", False
    Set pISWiRelease = pISWiProject.ISWiProductConfigs("Product Configuration 1").ISWiReleases("Release 1")
    pISWiRelease.Build
    pISWiProject.CloseProject
    Set pISWiRelease = Nothing
    Set pISWiProject = Nothing
End Sub

Private Sub pISWiRelease_ProgressIncrement(ByVal lIncrement As Long, pbCancel As Boolean)
    ' ここにコードを配置
End Sub

Private Sub pISWiRelease_ProgressMax(ByVal lMax As Long, pbCancel As Boolean)
    ' ここにコードを配置
End Sub

Private Sub pISWiRelease_StatusMessage(ByVal sMessage As String, pbCancel As Boolean)
    ' ここにコードを配置
End Sub
```

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの例

次の Visual Basic 6 コードは、これらのイベントをアドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用する方法をデモンストレーションします。

```
Public WithEvents pISWiSuiteRelease As ISWiAutoSuiteAutomation Interface Version.ISWiSuiteRelease

Private Sub Foo()
    Dim pISWiProject As ISWiAutoSuiteAutomation Interface Version.ISWiProject
    Set pISWiProject = CreateObject("ISWiAutoSuiteAutomation Interface Version.ISWiProject")
    pISWiProject.OpenProject "C:¥InstallShield 2021 Projects¥My Project Name-1.issuite", False
    Set pISWiRelease = pISWiProject.ISWiSuiteReleases("Release 1")
    pISWiSuiteRelease.Build
    pISWiProject.CloseProject
    Set pISWiSuiteRelease = Nothing
```

```
    Set pISWiProject = Nothing
End Sub

Private Sub pISWiSuiteRelease_ProgressIncrement(ByVal lIncrement As Long, pbCancel As Boolean)
    ' ここにコードを配置
End Sub

Private Sub pISWiSuiteRelease_ProgressMax(ByVal lMax As Long, pbCancel As Boolean)
    ' ここにコードを配置
End Sub

Private Sub pISWiSuiteRelease_StatusMessage(ByVal sMessage As String, pbCancel As Boolean)
    ' ここにコードを配置
End Sub
```

64 ビット システム上でオートメーション インターフェイスを使用する

オートメーション インターフェイスは 32 ビット インターフェイスを持つため、32 ビット プロセスからロードしなくてはなりません。オートメーション インターフェイスを 64 ビット マシンで使用する場合、これを 32 ビット実行可能ファイルを使ってロードする必要があります。

たとえば、オートメーション インターフェイスで VBScript を使用する場合、cscript.exe を 32 ビット システム フォルダー (SysWow64) から起動する必要があるかもしれません。そうしなければ、64 ビット スクリプト ホストでオートメーション オブジェクト作成時に次のようなエラーが発生する可能性があります:

Microsoft VBScript ランタイム エラー: ActiveX コンポーネントはオブジェクトを作成できません: 'ISWiAutoAutomation Interface Version.ISWiProject'

12

InstallShield プロジェクトを DevOps でビルドする

InstallShield では、DevOps における InstallShield プロジェクトのビルドがサポートされています。InstallShield プロジェクトをビルドするための、次の機能が提供されています:

- Azure DevOps ビルド拡張
- InstallShield StandAlone Build と Docker

Azure DevOps ビルド拡張

InstallShield では、Azure DevOps ビルド拡張がサポートされています。この拡張を使って、Azure DevOps Pipelines で InstallShield プロジェクトをビルドすることができます。

タスク ***Azure DevOps Pipelines で InstallShield プロジェクトのインストールをビルドするには、次の手順に従います:***

1. InstallShield ビルド拡張 を Azure DevOps 組織に追加およびインストールします。

   Install InstallShield SAB タスクおよび**InstallShield Build タスク**は Build Pipeline タスクで使用できます。

2. タスクを構成します。
3. リポジトリに格納されている InstallShield プロジェクトをビルドします。

Install InstallShield SAB タスク

Install InstallShield SAB タスクは、タスクで構成されている InstallShield バージョンに基づいて InstallShield Standalone Build セットアップを Microsoft がホストするマシン上にインストールして、タスクで提供されているライセンス サーバーの詳細を使ってライセンス サーバーを構成します。

InstallShield SAB タスクを構成するには、「InstallShield Azure DevOps Build Extension (InstallShield Azure DevOpsビルド拡張)」サポート技術情報の記事の**「タスク# 1: Install InstallShield SAB Task (Install InstallShield SABタスク)」**を参照してください。

InstallShield Build タスク

InstallShield Build タスクは、InstallShield SAB がエージェント上にインストール済みかどうかをチェックし、インストールされていない場合は、InstallShield SAB をインストールして InstallShield プロジェクトのビルドをトリガします。

InstallShieldビルドタスクを構成するには、「**InstallShield Azure DevOps ビルド拡張」サポート技術情報の記事の「タスク**# 2: InstallShieldビルドタスク」を参照してください。

InstallShield StandAlone Build と Docker

InstallShield プロジェクトは、Docker コンテナを使ってビルドすることもできます。Docker コンテナを使って InstallShield プロジェクトをビルドするには、Docker イメージをダウンロードしてから Docker イメージをビルドします。詳細については、「InstallShield StandAlone Build with Docker (Docker を使った InstallShield StandAlone のビルド)」を参照してください。

13

リファレンス

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *QuickPatch*
- *スイート/アドバンスト UI*
- *トランスフォーム*

InstallShield のリファレンス情報は次のセクションに分類されます。

テーブル 1 ・ リファレンス セクション

セクション	説明
メニュー、ツールバー、および ウィンドウのリファレンス	メニュー、ツールバー、およびウィンドウを含む InstallShield ユーザー インターフェイスの様々なコンポーネントについて説明します。
ダイアログ ボックス リファレンス	InstallShield で表示される各ダイアログ ボックスについてのリファレンス情報を含みます。
ウィザード リファレンス	InstallShield で利用できる各ウィザードの詳しい情報を提供します。

テーブル 1 • リファレンス セクション (続き.)

セクション	説明
ビュー リファレンス	InstallShield で表示される各ビューについて説明します。
InstallShield 前提条件エディター リファレンス	InstallShield で提供されている、InstallShield 前提条件エディターを紹介します。このツールを使用して、プロジェクトに含める InstallShield 前提条件を作成したり、既存の前提条件を編集したりできます。
エラーと警告	インストールの作成、コンパイル、ビルド、および実行する際に起こりうるエラー コードおよび警告についての情報を提供します。このセクションには、プロジェクトを InstallShield 製品の以前のバージョンから最新バージョンに移行するときに起こりうるエラーと警告についてのリファレンス情報も含まれています。
オートメーション インターフェイス	オートメーション インターフェイスでプロジェクトを作成、変更、およびビルドするのに使用されるオブジェクト、メソッド、プロパティ、および、コレクションについて説明するリファレンス資料が含まれています。
アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス	アドバンスト UI またはスイート/アドバンスト UI プロジェクトの様々な設定にオブジェクト式を埋め込むのに使用できる各オブジェクトについて説明します。オブジェクト式を使って、ターゲットシステムについてファイル、レジストリ エントリ、その他の項目についての情報を検索できます。実行時、インストールは式の値を拡張して、ターゲット システムを検索します。
InstallShield カスタム アクション リファレンス	InstallShield で使用できるカスタム アクションそれぞれについて説明します。
コマンドライン ツール	リリースのビルドおよびインストールの実行などのタスクを実行するためにコマンドラインから使用できるツールを紹介します。
InstallScript 言語リファレンス	関数、データ型、イベント ハンドラー、演算子、および InstallScript の他の面についての詳しい情報を提供します。InstallScript は、インストールを設計するときに利用できるシンプルでありながら強力なプログラム言語です。このセクションには、ユーザーがスクリプト ファイルで使用できるサンプル コードも含まれています。

メニュー、ツールバー、および ウィンドウのリファレンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

このセクションは、メニュー、ツールバー、およびウィンドウを含む InstallShield ユーザー インターフェイスの様々なコンポーネントについて説明します。

- メニュー
- ツールバー
- 出力ウィンドウ

メニュー

プロジェクト: *[インストール情報] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

InstallShield のメニューは、InstallShield ユーザー インターフェイスの上部のメニューバーにあります。メニューには、それぞれ、コマンドのリストがあります。一部のコマンドはその隣にアイコンを持っており、コマンドとアイコンを手早く関連付けることができます。

InstallShield の各メニューは、このセクションで説明されています。

- [ファイル] メニュー
- [ホーム] メニュー
- [編集] メニュー
- [表示] メニュー
- [移動] メニュー
- [プロジェクト] メニュー
- [ビルド] メニュー
- [ツール] メニュー
- [ウィンドウ] メニュー
- [ヘルプ] メニュー

[ファイル] メニュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

次の表は、[ファイル] メニューのコマンド一覧と関連キーボード ショートカットおよびアイコンの一覧です。

テーブル 1 ・ ファイル メニュー コマンド

コマンド	ショートカット	アイコン	説明
新規作成	Ctrl+N		新規インストール プロジェクトを作成します。

テーブル 1 • ファイル メニュー コマンド (続き.)

コマンド	ショートカット	アイコン	説明
開く	Ctrl+O		既存のインストール プロジェクトを開きます。
閉じる			現在のプロジェクトを閉じます。
保存	Ctrl+S		現在のプロジェクトを保存します。
名前を付けて保存			現在のプロジェクトファイルに新しい名前を付けて、新しい場所に保存することができます。
印刷	Ctrl+P		アクティブなスクリプト ウィンドウに表示されているスクリプト ファイルを印刷します。
最近使ったプロジェクト			最後にアクセスしたプロジェクトの 1 つを開きます。
終了			現在のプロジェクトを閉じて InstallShield を終了します。

[ホーム] メニュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

次の表は、[ホーム] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 2 ・ [ホーム] メニューのコマンド

コマンド	ショートカット	アイコン	説明
新規作成	Ctrl+N		新規インストール プロジェクトを作成します。
開く	Ctrl+O		既存のインストール プロジェクトを開きます。
保存	Ctrl+S		現在のプロジェクトを保存します。
[トラディショナル] ビュー			[トラディショナル] ビューに切り替えます。
コンパイル	Ctrl+F7		InstallScript をコンパイルします。
ビルド	F7		アクティブ リリースをビルドします。
ビルドの停止	Ctrl+Break		アクティブ リリースのビルドを停止します。
ユーザー インターフェイスのテスト	Ctrl+T		アクティブな Windows Installer ベースのリリースのユーザー インターフェイスをテストします。
実行	Ctrl+F5		MSIX リリース – アクティブ リリースを実行または登録します。
アンインストール			MSIX リリース のアンインストール – アクティブな Windows インストーラー ベースのリリースをアンインストールします。
デバッグ	F5		アクティブ リリースをデバッグします。

テーブル 2 - [ホーム] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
リリース フォルダーを開く			Windows Explorer でリリース フォルダーを開きます。
MSIX のビルド			
MSIX の実行			
MSIX のアンインストール			
前のビュー	ALT+ ↑		ビュー リストに表示されている現在のビューの 1 つ上のビューに移動します。
次のビュー	ALT+ ↓		ビュー リストで表示されている現在のビューの 1 つ下のビューに移動します。
戻る	Alt+←		ビューを選択する履歴から最後に訪れたビューへ移行します。ビュー履歴に複数のエントリがある場合、これは何回でも使用することが可能です。
進む	ALT+→		ビューを選択する履歴から最後に訪れたビューへ移行します。最初に [戻る] をクリックしたビューに到達するまで続けられます。
ヘルプ			[ヘルプ] ビューを表示します。

[編集] メニュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

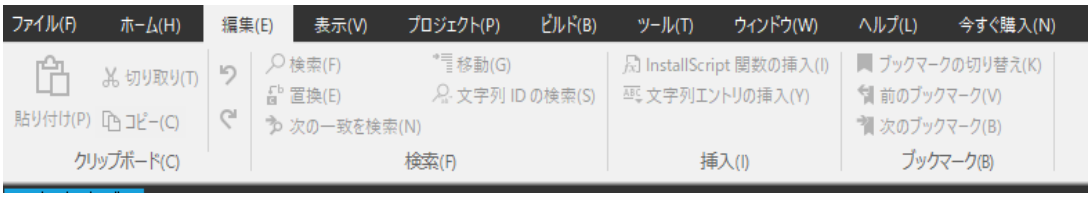
次の表は、[編集] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 3 ・ [編集] メニュー コマンド

コマンド	ショートカット	アイコン	説明
元に戻す	Ctrl+Z		最後に実行されたアクションを元に戻します。
やり直し	Ctrl+Y		[元に戻す] コマンドで実行した最後のアクションを無効にします。
切り取り	Ctrl+X		選択中のテキストを削除して、クリップボードにおきます。
コピー	Ctrl+C		選択中のテキストをクリップボードにコピーします。
貼り付け	Ctrl+V		クリップボードの内容を挿入ポインターの位置に挿入してから、選択したテキストを置換します。
検索	Ctrl+F		指定したテキスト、ファイルまたはフォルダーを検索します。
置換	Ctrl+H		指定したテキストを検索し置換します。
次を検索	F3		指定したテキストの次の位置を検索します。
移動	Ctrl+G		アクティブなスクリプト ウィンドウで指定した行番号へ挿入ポイントを移動します。
文字列 ID の検索			指定した文字列エントリがある場所をインストール プロジェクト全体から検索します。
InstallScript 関数の挿入	Ctrl+I		関数ウィザードを起動してスクリプトに InstallScript 関数を追加します。
文字列 エントリの挿入			スクリプトに文字列エントリを挿入します。

テーブル 3 · [編集] メニュー コマンド (続き.)

コマンド	ショートカット	アイコン	説明
ブックマークの切り替え	Alt+K		この行にブックマークを切り替えます。
前のブックマーク	Ctrl+Shift+K		前のブックマークに移動します。
次のブックマーク	Ctrl+K		次のブックマークに移動します。



[表示] メニュー

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

次の表は、[表示] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 4 · [表示] メニュー コマンド

コマンド	ショートカット	アイコン	説明
ビュー リスト	F4		ビュー リストを表示または非表示にします。

テーブル 4 · [表示] メニュー コマンド (続き.)

コマンド	ショートカット	アイコン	説明
ビュー バー			ビューバーを表示または非表示にします。
ヘッダー バー			[スタート ページ]、[プロジェクト アシスタント] および [インストール デザイナー] のタブがあるヘッダー バーを表示または非表示にします。
ステータス バー			ステータス バーを表示または非表示にします。
変数			InstallScript デバッガーの変数ウィンドウを表示または非表示にします。
MSI デバッガー			MSI デバッガーツールバーを表示または非表示にします。
レイアウト			レイアウト ツールバーを表示または非表示にします。
コントロール			コントロール ツールバーを表示または非表示にします。
プロジェクト アシスタント			プロジェクト アシスタントを表示します。
Microsoft App-V			[Microsoft App-V] タブを表示します。
出力ウィンドウ			[出力] を表示または非表示にします。

[移動] メニュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

メモ: *[移動] メニューは、[モダン] ビューでは使用できません。*

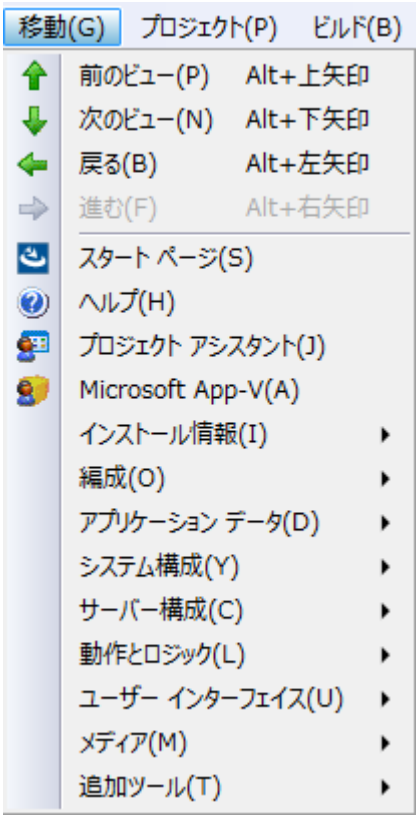
次の表は、[移動] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。一部のビューに関するコマンドは、InstallShield で開いたプロジェクトによっては、[移動] メニューからは利用できません。

テーブル 5 · [移動] メニュー コマンド

コマンド	ショートカット	アイコン	説明
前のビュー	ALT+↑		ビュー リストに表示されている現在のビューの 1 つ上のビューに移動します。
次のビュー	ALT+↓		ビュー リストで表示されている現在のビューの 1 つ下のビューに移動します。
戻る	Alt+←		ビューを選択する履歴から最後に訪れたビューへ移行します。ビュー履歴に複数のエントリがある場合、これは何回でも使用することが可能です。
進む	ALT+→		ビューを選択する履歴から最後に訪れたビューへ移行します。最初に [戻る] をクリックしたビューに到達するまで続けられます。
スタート ページ			スタートページに移動します。
ヘルプ			[ヘルプ] ビューに移動します。
プロジェクト アシスタント			プロジェクト アシスタントを表示します。
Microsoft App-V			[Microsoft App-V] タブを表示します。
VMware ThinApp			[VMware ThinApp] タブを表示します。
インストール情報			[一般情報] ビューを開くことができます。
編成			[セットアップのデザイン]、[機能]、[コンポーネント] および [セットアップの種類] を表示することができます。

テーブル 5 • [移動] メニュー コマンド (続き.)

コマンド	ショートカット	アイコン	説明
アプリケーション データ			[ファイルとフォルダー]、[再配布可能ファイル (InstallScript プロジェクトでは [オブジェクト])]、および [前提条件] ビューを表示することができます。
システム構成			[ショートカット]、[レジストリ]、[ODBC リソース]、[INI ファイルの変更]、[環境変数]、[XML ファイルの変更]、および [テキスト ファイルの変更] ビューを表示できます。
サーバー構成			[IIS 構成]、[コンポーネント サービス] および [SQL スクリプト] ビューを表示することができます。
動作とロジック			InstallScript、[カスタム アクションとシーケンス] (または [カスタム アクション])、[サポート ファイル] (InstallScript と InstallScript MSI プロジェクトの場合 [サポート ファイル/ビルボードのサポート])、[システム検索]、および、[プロパティ マネージャー] ビューを表示することができます。
ユーザー インターフェイス			[ダイアログ]、[ビルボード]、および [文字列エディター] ビューを表示できます。
メディア			[パス変数]、[アップグレード]、[リリース]、[パッチのデザイン] ビューを表示することができます。
追加ツール			[依存関係スキャナー]、[MSI デバッガー]、および [ダイレクト エディター] ビューを表示できます。
パッチの設定			[一般情報]、[ファイル]、[レジストリ] および [パッチ変数] ビューを表示できます。



[プロジェクト] メニュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

メモ: *MSIX プロジェクト タイプで [プロジェクト] メニューは使用できません。*

次の表は、[プロジェクト] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 6 · [プロジェクト] メニュー コマンド

コマンド	アイコン	説明
Visual Studio デプロイメント プロジェクト インポート ウィザード		Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield プロジェクトにインポートできるウィザードを起動します。
コンポーネントのエクスポート ウィザード...		コンポーネントのエクスポートウィザードを開きます。このウィザードで、ファイルのコンポーネントを既存または新しいプロジェクトへエクスポートする処理が簡単になります。
デバイス ドライバー ウィザード		スマート ドライバー ウィザードを開きます。 スマート ドライバー ウィザードの案内に従って、デバイスドライバーをインストールに簡単に追加することができます。
Reg-Free COM ウィザード...		Reg-Free COM ウィザードを開きます。 Reg-Free COM ウィザードの案内に従って、効率よく COM マニフェスト ファイルをインストールに追加できます。また、既存のマニフェスト ファイルを構成することもできます。
スタティック スキャン		スタティック スキャン ウィザードを開きます。このウィザードは、プロジェクトに既に追加されているファイルをスキャンして必要な依存関係があるかどうかをチェックします。
ダイナミック スキャン		ダイナミック スキャン ウィザードを開きます。このウィザードは、実行可能ファイルをモニターして、すべての .dll および .ocx 依存関係をチェックし、それらを自動的にプロジェクトに追加します。
ソース パス コンバーター		ソース パス変換ウィザード を開きます。ソース パス変換ウィザードを使用すると、ハードコードされた既存のパスをパス変数で変換できるため、インストール プロジェクトの移植性を向上させることができます。
プロジェクト コンバーター		いくつかのコマンドを提供します: • **InstallScript MSI プロジェクトに変換する** – 基本の MSI プロジェクトを InstallScript MSI プロジェクトに変換します。詳細については、「基本の MSI プロジェクトを InstallScript MSI プロジェクトへ変換する」を参照してください。 • **InstallScript プロジェクトに変換する** – [InstallScript MSI の変換] ダイアログ ボックス を開いて、InstallScript MSI プロジェクトを InstallScript プロジェクトに変換します。 • **Compact プロジェクトを基本の MSI プロジェクトに変換する** – コンパクト プロジェクトを基本の MSI プロジェクトに変換します。

テーブル 6 ・[プロジェクト] メニュー コマンド (続き.)

コマンド	アイコン	説明
ソース管理		いくつかのコマンドを提供します: ・ **最新バージョンの取得** – 現在のプロジェクトの最新バージョンをプロジェクト フォルダーにコピーします。プロジェクト フォルダーに置かれているプロジェクトは読み取り専用です。 ・ **チェックアウト** – プロジェクトを変更できるよう、書き込み可能なコピーをプロジェクト フォルダーに配置します。 ・ **チェックイン** – ソース管理アプリケーション内の現在のプロジェクトに変更を保存します。 ・ **チェックアウトを元に戻す** – ソース管理アプリケーションおよびプロジェクト フォルダー両方の変更をキャンセルし、チェックアウトを元に戻します。プロジェクトはチェックアウト前の状態も戻ります。 ・ **ソース管理に追加** – 現在のプロジェクトをソース管理アプリケーションに追加します。 ・ **ソース管理からリンクを解除する** – 現在のプロジェクトとソース管理アプリケーションの間のリンクを解除します。 ・ **履歴の表示** – 変更レコードを、ソース管理アプリケーションへ初回で追加された時点から表示します。履歴を表示して、履歴されているプロジェクトのいつの時点に戻って、その時点で存在していたファイルを回復することができます。 ・ **差分を表示** – 2 つのファイルを比較してから、差分があれば、それを表示します。 ・ **ステータスのリフレッシュ** – ソース管理ステータスをリフレッシュします。 ・ **ソース管理プログラムの起動** – ソース コード管理アプリケーションを開きます。
プロジェクトの設定		[プロジェクトの設定] ダイアログ ボックス が開きます。

[ビルド] メニュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

次の表は、[ビルド] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 7 • [ビルド] メニューのコマンド

コマンド	ショートカット	アイコン	説明
リリース ウィザード			[リリース ウィザード] を開きます。このウィザードを使って、リリースの設定を指定し、ビルドすることができます。
コンパイル	Ctrl+F7		スクリプト ファイルをコンパイルします。
ビルド	F7		リリースをデフォルトの設定でビルドするか、既にリリースをビルド済みの場合は、最後に保存した設定でリリースを再ビルドします。
テーブルのみをビルドする	SHIFT+F7		インストールの .msi ファイルテーブルのみ再ビルドします。これにより。完全再ビルドに比べて変更点を素早く確認することができます。
テーブルのビルドとファイルのリフレッシュ	ALT+F7		 ***プロジェクト:*** *このコマンドは、Windows Installer ベースのプロジェクトで使用できます。* .msi ファイルのテーブルのみを再ビルドして、インストールの変更があったファイルのみをアップデートします。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワークイメージである場合だけ作動します。

テーブル 7 • [ビルド] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
リフレッシュ ビルド	ALT+F7		 ***プロジェクト•*** *このコマンドは、InstallScript プロジェクトで使用できます。* アクティブ メディアを再ビルドします。このコマンドは、メディアが最後にビルドされたとき以後変更を加えたアイテムのみを再生成します。
バッチ ビルド			構成が異なる複数のリリースを、同時にビルドすることができます。
ビルドの停止	Ctrl+Break		現在のビルドプロセスをキャンセルします。
ユーザー インターフェイスのテスト	Ctrl+T		システムに何の変更も加えることなく、インストールのユーザー インターフェイス部分を実行できます。すべてのカスタム アクションが実行されます。
デバッグ	F5		デバッグを開始します。
実行	Ctrl+F5		InstallShield を終了せずに、完成したインストールを実行することができます。
アンインストール			前回実行されたリリースをアンインストールします。 ***プロジェクト•*** *このコマンドは、次のプロジェクト タイプで使用できます:* • *基本の MSI* • *InstallScript MSI* • *MSIX*
パッケージの実行			インストール パッケージを実行することができます。このコマンドは、ネットワークイメージ メディアの種類を使用し、すべてのファイルが .exe ファイルに圧縮されている InstallScript プロジェクトにのみ利用可能です。
パッケージのデバッグ			インストール パッケージをデバッグすることができます。このコマンドは、ネットワークイメージ メディアの種類を使用し、すべてのファイルが .exe ファイルに圧縮されている InstallScript プロジェクトにのみ利用可能です。
Web からの実行			One-Click Install Web ページを表示します。このページから、InstallShield を終了せずに完成したインストールを実行することができます。

テーブル 7 • [ビルド] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
MSI デバッガー			いくつかのコマンドを提供します: • **MSI デバッグの開始** — [MSI デバッガー] ビューで、デバッグを開始します。 • **ブレークポイントの切り替え** — F9 または 挿入ポイントがある行でブレークポイントをセットまたは削除します。 • **すべてのブレークポイントの削除** — SHIFT+F9 または — デバッガーでセットされたブレークポイントをすべて削除します。 • **ステップ オーバー** — F10 または — デバッガーでカスタム アクションまたはダイアログをステップスルーし、実行を継続します。 • **ステップイン** — F11 または — デバッガーで、現在のカスタム アクションをステップインします。 • **MSI デバッグの停止** — SHIFT+5 または — [MSI デバッガー] ビューでのデバッグを停止します。

テーブル 7 • [ビルド] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
検証			いくつかのコマンドを提供します: • **アップグレード検証ウィザード** — アップグレード検証ウィザードを開きます。このウィザードは一連のテストを実行して、基本の MSI インストールまたは InstallScript MSI インストールが以前のバージョンを適切にアップグレードするかどうかを判別します。 • **新しい検証 (.cub) モジュールの参照** — 新規またはカスタム検証 (.cub) ファイルを参照選択することができます。 • **InstallShield 検証スイート – Windows 8** — このスイートは、インストール時に潜在的に Windows 8 システムで予期しない動作が発生する原因になる問題を識別する多数の InstallShield 内部整合性評価プログラム (ISICE) から構成されています。このスイートは、完全 MSI 検証スイートでは必ずしも発見できるとは限らない問題を確認します。 • **InstallShield マージ モジュール検証スイート – Windows 8** または **InstallShield マージ モジュール検証スイート – Windows 7** — これらのスイートは、Windows 8 (または Windows 7) システム上でマージ モジュールの予期しない動作が発生する原因になる問題を識別する多数の InstallShield 内部整合性評価プログラム (ISICE) から構成されています。このスイートは、マージ モジュール検証スイートでは必ずしも発見できるとは限らない問題を確認します。 • **InstallShield Certified for Windows Vista 検証スイート** — このスイートは、インストール時に潜在的に Windows Vista システムで予期しない動作が発生する原因になる問題を識別する多数の InstallShield 内部整合性評価プログラム (ISICE) から構成されています。このスイートは、完全 MSI 検証スイートでは必ずしも発見できるとは限らない問題を確認します。 • **InstallShield Certified for Windows Vista マージ モジュール検証スイート** — このスイートは、マージ モジュールの実行時に潜在的に Windows Vista システムで予期しない動作が発生する原因になる問題を識別する多数の InstallShield 内部整合性評価プログラム (ISICE) から構成されています。このスイートは、マージ モジュール検証スイートでは必ずしも発見できるとは限らない問題を確認します。

テーブル 7 • [ビルド] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
検証 (続き)			• **InstallShield ベスト プラクティス スイート** – 基本の MSI インストールまたは InstallScript MSI インストールをスキャンして、ベスト プラクティス ガイドラインに適合しているかどうかを判別します。 ***エディション:*** *InstallShield ベスト プラクティス スイートは、InstallShield の Premier Edition で提供されています。* • **完全 MSI 検証スイート** – 基本の MSI インストールまたは InstallScript MSI インストールをスキャンして、インストールが有効かどうかを判別します。ビルドした .msi パッケージは、ICE 規則に照合されます。 • **Windows 2000 ロゴ検証スイート** – 基本の MSI インストールまたは InstallScript MSI インストールをスキャンして、Designed for Windows 2000 ロゴ要件を満たしているかどうかを判別します。 • **Windows XP ロゴ検証スイート** – 基本の MSI インストールまたは InstallScript MSI インストールをスキャンして、Designed for Windows XP ロゴ要件を満たしているかどうかを判別します。 • **Windows Vista ロゴ検証スイート** – 基本の MSI インストールまたは InstallScript MSI インストールをスキャンして、Certified for Windows Vista ロゴ要件を満たしているかどうかを判別します。 • **マージ モジュール検証スイート** – マージ モジュール パッケージをスキャンします。マージ モジュール検証ツールは完全 MSI 検証スイートのマージ モジュールバージョンです。ビルドされたマージ モジュールは正常に機能するかを判別するための ICE 規則と比較されます。 • **InstallShield 仮想化適合性スイート**– .msi パッケージをスキャンして、仮想化適合性に関連するすべてのテストを実行します。問題が見つかったすべてのテストがレポートに一覧表示され、各問題に対して、レポート内の関連列に既知の仮想化フォーマット (App-V v5、Server App-V、App-V-v4、ThinApp、または XenApp) に対する適合性が表示されます。 • **InstallShield MSIX/UWP アプリ適合性スイート**–.msi パッケージ内で Windows アプリ パッケージ (.appx) 形式に適さないアイテムの存在をスキャンします。問題が見つかったすべてのテストがレポートに一覧表示され、各問題に対して、レポート内の関連列に既知の Windows アプリ バージョン (UWA、デスクトップ、ストア、WSA、または Nano) に対する適合性が表示されます。 • **最近の検証モジュール** – 最近使用された検証 (.cub) ファイルのリストを表示します。 詳細については、「プロジェクトを検証する」および「検証結果を表示する」を参照してください。

テーブル 7 · [ビルド] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
設定	ALT+F6		[設定] ダイアログ ボックスを開きます。このダイアログ ボックスを使用すると、InstallShield がプロジェクトをビルドするときの環境設定を指定することができます。

[ツール] メニュー

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

次の表は、[ツール] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 8 · [ツール] メニュー コマンド

コマンド	アイコン	説明
リリース フォルダーを開く		Windows Explorer を起動し、現在のリリースの DISK1フォルダーで開きます。リリースがない、または、リリースばまだビルドされていない場合、Windows Explorer がデフォルトのプロジェクトの場所に開きます。
新しい言語ウィザードの追加		新しい言語ウィザードを開きます。このウィザードを利用して、ほとんどの言語のサポートを 1 つまたは複数のプロジェクトに追加することができます。

テーブル 8 • [ツール] メニュー コマンド (続き.)

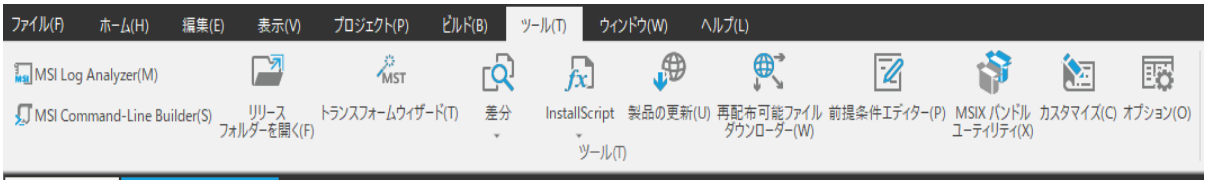
コマンド	アイコン	説明
トランスフォーム ウィザード		トランスフォームウィザードを開きます。このウィザードを利用すると、ウィザードの案内に従って、トランスフォームを作成しインストール プロジェクトに適用することができます。
MSI Log Analyzer		InstallShield MSI Log Analyzer を起動します。このアナライザーは、Windows Installer ログ ファイルの分析に基づいてレポートを作成することができるユーティリティです。
MSI コマンドライン ビルダー		InstallShield MSI コマンドライン ビルダーを起動します。このツールを使用して、Windows Installer 関連のタスクを実行する複雑なコマンドライン文字列を素早くビルドすることができます。
差分		いくつかのコマンドを提供します: • **ファイルとの比較** – 開いているプロジェクトと比較するプロジェクトを指定できるダイアログ ボックスを開きます。 • **比較の終了** – 前回開始した比較セッションを終了します。 • **InstallShield MSI Diff** – InstallShield MSI Diff を開きます。InstallShield MSI Diff を使用して、2 つの異なる Windows Installer インストール データベースの Windows Installer テーブルを比較することができます。また、Windows Installer パッケージにトランスフォームやパッチを適用した後に、データベース テーブルにどのように影響されているかを確認することもできます。詳細については、「InstallShield MSI Diff」を参照してください。

テーブル 8 • [ツール] メニュー コマンド (続き.)

コマンド	アイコン	説明
InstallScript		いくつかのコマンドを提供します: • **標準ダイアログ サンプラー** – 標準ダイアログ サンプラーを開きます。サンプラーを利用して、エンドユーザーダイアログのサンプルを見ることができます。サンプルダイアログは、スキンなしで表示されます。 • **スキン ダイアログ サンプラー** – スキン ダイアログ サンプラーを開きます。サンプラーを利用して、エンド ユーザー ダイアログのサンプルを見ることができます。サンプル ダイアログは、[ダイアログ] ビューで選択したスキンと共に表示されます。 • **[キャビネット ファイル] ビューアー** – キャビネット ファイル ビューアーを開きます。このツールで、キャビネット (.cab) ファイルを選択すると、その圧縮ファイル、ファイル グループ、コンポーネント、セットアップの種類を表示できます。これらのプロパティを確認することもできます。キャビネット ファイルからファイルを抽出することもできます。 • **[ログ ファイル] ビューアー** – ログ ファイル ビューアーを開きます。このツールを利用して、インストールのログ ファイルを表示することができます。ログファイルは、インストールが最初に実行されるときに作成され、メンテナンス モードでインストールが実行されているときにメンテナンスされます。ログファイルを使用して、インストールで各コンポーネントの現在の状態を判断することができます。
アップデートの確認		サービスパックおよび InstallShield への他のアップデートを確認します。アップデーが可能な場合、ダウンロードおよびインストールを選択できます。
再配布可能ファイル ダウンローダー		再配布可能ファイルダウンローダーウィザードを起動して、サードパーティ再配布可能ファイル、マージ モジュール、前提条件、その他のファイルをローカルコンピューターに素早くダウンロードすることができます。
前提条件エディター		InstallShield 前提条件ファイルエディター を開きます。これを利用して、インストール プロジェクトに含むことが可能な InstallShield 前提条件ファイルの作成および変更を行うことができます。
MSIX バンドル ユーティリティ		MSIX バンドル パッケージを作成することができます。グラフィック インターフェイスでペイロード MSIX パッケージ、署名情報を指定し、構成済みの MSIX パッケージから MSIX バンドル パッケージを作成できます。
カスタマイズ		[カスタマイズ] ダイアログ ボックスを表示します。このダイアログ ボックスで、InstallShieldユーザー インターフェイスで表示されるツールバーをカスタマイズすることができます。

テーブル 8 · [ツール] メニュー コマンド (続き.)

コマンド	アイコン	説明
オプション		[オプション] ダイアログ ボックスを開きます。このダイアログ ボックスで、InstallShield でプロジェクトの作成および作業を行うときの環境設定を指定することができます。



[ウィンドウ] メニュー

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

***メモ·** MSIX プロジェクト タイプで [ウィンドウ] メニューは使用できません。*

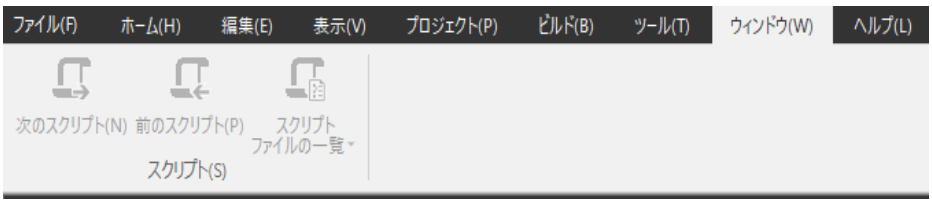
次の表は、[ウィンドウ] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 9 · [ウィンドウ] メニューのコマンド

コマンド	ショートカット	アイコン	説明
次のスクリプト	Ctrl + Tab		InstallScript ビューを開き、ウィンドウ メニューの下で表示されるようにプロジェクトで次のスクリプト ファイルを表示します。
前のスクリプト	Ctrl + Shift + Tab		InstallScript ビューを開き、ウィンドウ メニューの下で表示されるようにプロジェクトで次のスクリプト ファイルを表示します。

テーブル 9 - [ウィンドウ] メニューのコマンド (続き.)

コマンド	ショートカット	アイコン	説明
スクリプト ファイル リスト			InstallScript ビューを開き、プロジェクトに含まれているスクリプト ファイル一覧を表示します。



[ヘルプ] メニュー

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

次の表は、[ヘルプ] メニューのコマンド一覧と関連キーボードショートカットおよびアイコンの一覧です。

テーブル 10 - [ヘルプ] メニュー コマンド

コマンド	アイコン	説明
サポート セントラル		Web のサポート セントラルを表示します。
コミュニティ		Web の [コミュニティ] を表示します。
目次		オンライン ヘルプ ライブラリの目次タブを表示します。

テーブル 10 • [ヘルプ] メニュー コマンド (続き.)

コマンド	アイコン	説明
リリース ノート		InstallShield リリースノートを表示します。
InstallShield Web サイト		InstallShield の Web サイトに接続します。
InstallShield について		バージョン情報を表示して InstallShield を登録できる [バージョン情報] ダイアログ ボックスを表示します。

ツールバー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

InstallShield ユーザー インターフェイスには、使用頻度の高いメニュー項目に簡単にアクセスできるツール バー、並びにダイアログ エディターで使用できるグラフィック ツールが用意されています。

ビルトイン ツールバー

InstallShield インターフェイスでは、次のツールバーを使用できます。InstallShield 開くと、ユーザー インターフェイスの上部に [標準] ツールバーのみが表示されます。[コントロール] ツールバーおよび [ダイアログ レイアウト] ツールバーは、[ダイアログ] ビューで作業中に表示され、このビューから移動すると表示されなくなります。

すべてのツールバーはサイズや配置を変更できるほか、簡単に取り付けたり取り外したりできます。

- 標準ツールバー
- [コントロール] ツールバー
- レイアウト ツールバー
- [MSI デバッグ] ツールバー

標準ツールバー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

標準ツールバーのすべてのボタンの説明は次の通りです。

テーブル 11 • 標準ツールバーのボタン

ボタン	名前	説明
	新しいプロジェクト	プロジェクトタイプを選択して新しいプロジェクトを始める [新規プロジェクト] ダイアログ ボックスを起動します。
	開く	プロジェクトファイルを開いたり、インストールパッケージ (.msi) ファイルを InstallShield インストール プロジェクト (.ism) ファイルに変換することができます。
	保存	プロジェクトファイルへ変更を保存します。
	元に戻す	ダイアログ エディターまたはスクリプト エディターの [元に戻す] ボタンをクリックして、最後に行った変更を元に戻します。
	やり直し	ダイアログ エディターまたはスクリプト エディターの [やり直し] ボタンをクリックし、[元に戻す] ボタンをクリックして取りやめた最後の動作を復元します。

テーブル 11 • 標準ツールバーのボタン (続き.)

ボタン	名前	説明
	ビューバー	インターフェイスの左側に表示されるビューバーを表示、または非表示します。
	ビュー リスト	InstallShield IDE で使用できるすべてのビューを表示するビュー リストを表示または非表示にします。 ***メモ:*** *また、F4 を押して、ビューリストを表示または非表示にすることもできます。*
	前のビュー	ビュー リストで表示されている通り、現在のビューの 1 つ上のビューを表示します。
	次のビュー	ビュー リストで表示されている通り、現在のビューの 1 つ下のビューを表示します。
	戻る	ビューを選択した履歴から、最後に訪れたビューを表示します。ビュー履歴に複数のエントリがある場合、このボタンは何回でもクリックすることが可能です。
	進む	ビューを選択した履歴から最後に訪れたビューを表示します。初めに [戻る] ボタンをクリックしたビューに画面が変わるまで、このボタンをクリックし続けることができます。
	InstallScript 関数の挿入	ビルトイン InstallScript 関数の記述プロセスを簡単にする、関数ウィザードを起動します。
	リリース ウィザード	プロジェクトを出荷可能な Windows Installer セットアップパッケージ、または完全なマージ モジュールに仕上げるリリース ウィザードを起動します。ウィザードで選択するオプションにより、リリースのレイアウトとコンテンツを決定します。
	コンパイル	セットアップ全体をビルドすることなく、プロジェクトに含まれる InstallScript ファイルをコンパイルします。
	ビルド	リリースをデフォルトの設定でビルドするか、既にリリースをビルド済みの場合は、最後に保存した設定でリリースを再ビルドします。
	ビルドの停止	現在のビルドプロセスをキャンセルします。

テーブル 11 • 標準ツールバーのボタン (続き.)

ボタン	名前	説明
	ユーザー インターフェイスのテスト	システムに何の変更も加えることなく、インストール プロジェクトのユーザー インターフェイスの部分を実行できます。すべてのカスタム アクションが実行されます。 ***プロジェクト•*** *[テスト] ボタンは、次のプロジェクトの種類で使用可能です:* • 基本の MSI • InstallScript MSI
	実行	完成したインストール プロジェクトを実行します。 ***プロジェクト•*** *基本の MSI および InstallScript MSI プロジェクト — **[インストールまたはデバッグの前に自動的に製品をアンインストール]** チェック ボックスを選択した場合、InstallShield がインストールまたはデバッグを開始する前に製品をアンインストールします。このチェック ボックスは、[オプション] ダイアログ ボックスの [プリファレンス] タブから使用できます。*
	アンインストール	前回実行されたリリースをアンインストールします。 ***プロジェクト•*** *このボタンは、次のプロジェクト タイプで使用できます:* • *基本の MSI* • *InstallScript MSI*
	デバッグ	作成しているインストールの種類に従って適切なデバッガーを実行します。プロジェクトが InstallScript または InstallScript MSI の場合、[デバッグ] をクリックすると InstallShield Debugger が起動します。プロジェクトが基本の MSI またはトランスフォームの場合、[デバッグ] をクリックすると MSI デバッガーが起動します。 ***プロジェクト•*** *基本の MSI および InstallScript MSI プロジェクト — **[インストールまたはデバッグの前に自動的に製品をアンインストール]** チェック ボックスを選択した場合、InstallShield がインストールまたはデバッグを開始する前に製品をアンインストールします。このチェック ボックスは、[オプション] ダイアログ ボックスの [プリファレンス] タブから使用できます。*
	リリース フォルダーを開く	Windows Explorer を起動し、現在のリリースの DISK1フォルダーで開きます。リリースがない、または、リリースはまだビルドされていない場合、Windows Explorer がデフォルトのプロジェクトの場所に開きます。

テーブル 11 • 標準ツールバーのボタン (続き.)

ボタン	名前	説明
	[ヘルプ] ビュー	InstallShield について多くの質問の答えが見つかる IDE ヘルプビューを表示します。

[コントロール] ツールバー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[コントロール] ツールバーは、ランタイム ダイアログで利用可能なすべての標準コントロールを提供します。このツールバーは、ダイアログ エディターでダイアログのレイアウトを編集するときに表示されます。

タスク

ダイアログにコントロールを書き込むには、以下の手順を実行します。

コントロールのボタンをクリックし、ダイアログ上の任意の場所をクリックします。次に、マウスのボタンを押しながらマウスポインターをドラッグし、コントロールが適切な大きさになったらボタンを離します。

テーブル 12 • [コントロール] ツールバーのボタン

ボタン	名前	説明
	Select Tool	ダイアログ上のコントロールがフォーカスされます。この状態で、コントロールの移動、サイズ変更、削除、または配置を行うことができます。
	チェック ボックス	ダイアログ上にチェック ボックス コントロールを配置します。
	Push Button	ダイアログ上にプッシュ ボタン コントロールを配置します。
	編集フィールド	ダイアログ上に編集フィールド コントロールを配置します。

テーブル 12 • [コントロール] ツールバーのボタン (続き.)

ボタン	名前	説明
	コンボ ボックス	ダイアログ上にコンボ ボックス コントロールを配置します。
	テキスト領域	ダイアログ上にテキスト領域コントロールを配置します。
	リスト ボックス	ダイアログ上にリスト ボックス コントロールを配置します。
	ラジオ ボタン	ダイアログ上にラジオ ボタン コントロールを配置します。ラジオ ボタンは、ラジオ ボタン グループ コントロールに追加する必要があります。
	ビットマップ	ダイアログ上にビットマップ コントロールを配置します。
	グループ ボックス	ダイアログ上にグループ ボックス コントロールを配置します。
	Billboard	ダイアログ上にビルボード コントロールを配置します。
	線	ダイアログ上に行コントロールを配置します。
	ラジオ ボタン グループ	ダイアログ上にラジオ ボタン グループ コントロールを配置します。
	選択ツリー	ダイアログ上に選択ツリー コントロールを配置します。
	進行状況バー	ダイアログ上に進行状況バー コントロールを配置します。
	リスト ビュー	ダイアログ上にリスト ビュー コントロールを配置します。
	スクロール可能テキスト	ダイアログ上にスクロール可能テキスト コントロールを配置します。
	アイコン	ダイアログ上にアイコン コントロールを配置します。
	ディレクトリ リスト	ダイアログ上にディレクトリ リスト コントロールを配置します。
	ディレクトリ コンボ	ダイアログ上にディレクトリ コンボ コントロールを配置します。
	ボリューム コスト リスト	ダイアログ上にボリューム コスト リスト コントロールを配置します。
	ボリューム選択コンボ	ダイアログ上にボリューム選択コンボ コントロールを配置します。

テーブル 12 • [コントロール] ツールバーのボタン (続き.)

ボタン	名前	説明
	マスク編集	ダイアログ上にマスク編集コントロールを配置します。
	パス編集	ダイアログ上にパス編集コントロールを配置します。
	ハイパーリンク	ハイパーリンク コントロール を配置します。このコントロールはアドレスへの HTML リンクを表示します。エンド ユーザーがリンクをクリックすると、デフォルト ブラウザでそのページが開きます。

レイアウト ツールバー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

レイアウト ツールバーは、ダイアログ内のコントロールを調整したり、サイズを変更したりするいくつかのツールを提供します。このツールバーは、ダイアログ エディターでダイアログのレイアウトを編集するときに表示されます。

このツールバーにある大半のボタンは、複数のコントロールを選択している時に有効です。複数のコントロールを選択するには、ひとつを選んでシフトキーを押して他のものを選択するか、あるいはマウスボタンを押したままカーソルをドラッグして数個のコントロールを選択します。レイアウト ツールバーには、次のボタンが含まれています。

テーブル 13 • レイアウト ツールバーのボタン

ボタン	名前	説明
	左端を揃える	2個以上のコントロールを選択し、[左端を揃える] ボタンをクリックして、選択したコントロールをすべて、一番左端にあるコントロールの左側に揃えます。
	右端を揃える	2個以上のコントロールを選択し、[右端を揃える] ボタンをクリックして、選択したコントロールをすべて、一番右端にあるコントロールの右側に揃えます。

テーブル 13 • レイアウト ツールバーのボタン (続き.)

ボタン	名前	説明
	上を揃える	2個以上のコントロールを選択し、[上を揃える] ボタンをクリックして、選択したコントロールをすべて、一番上にあるコントロールの上端に揃えます。
	下を揃える	2個以上のコントロールを選択し、[下を揃える] ボタンをクリックして、選択したコントロールをすべて、一番下にあるコントロールの下側に揃えます。
	上下中央配置	コントロールをひとつ選択し、[上下中央配置] ボタンをクリックしてダイアログの垂直方向(x 軸)の中央に再配置します。
	左右中央配置	コントロールをひとつ選択し、[左右中央配置] ボタンをクリックしてダイアログの水平方向(y 軸)の中央に再配置します。
	左右均等配置	ダイアログ上の3個以上のコントロールを選択し、[左右均等配置] ボタンをクリックして、ダイアログ上に水平均等に配置します。
	上下均等配置	ダイアログ上の3個以上のコントロールを選択し、[上下均等配置] ボタンをクリックして、ダイアログ上に垂直均等に配置します。
	幅を揃える	2個以上のコントロールを選択し、[幅を揃える] ボタンをクリックし、コントロールの幅を一番大きなコントロールの幅に揃えます。
	高さを揃える	2個以上のコントロールを選択し、[高さを揃える] ボタンをクリックし、コントロールの高さを一番大きなコントロールの高さに揃えます。
	同じサイズに設定	2個以上のコントロールを選択し、[同じサイズに設定] ボタンをクリックし、コントロールのサイズを一番大きなコントロールのサイズに揃えます。
	最前面へ移動	コントロールをひとつ選択し、[最前面へ移動] ボタンをクリックして重なり合った複数のコントロールのトップに配置します。以下の注を参照してください。
	最背面へ移動	コントロールをひとつ選択し、[最背面へ移動] ボタンをクリックして重なり合った複数のコントロールの背後に配置します。以下の注を参照してください。
	グリッド表示	[グリッド表示] ボタンをクリックし、ダイアログに作図用グリッド線を表示、または非表示にします。

メモ: *[最前面へ移動]、および [最背面へ移動] アクションは、プロジェクトを閉じてから再び開いたときに保存されません。これらのアクションは、設計中に使用するためのものです。たとえば、条件に基づいて異なるコントロールを表示するダイアログがあるとします。ダイアログを設計中に、[最前面へ移動] および [最背面へ移動] アクションを使用して、これらの異なるコントロールで作業することができます。*

[MSI デバッグ] ツールバー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

以下は、[MSI デバッグ] ツールバーのすべてのボタンの説明です。

テーブル 14 • [MSI デバッグ] ツールバーのボタン

ボタン	名前	説明
	ブレークポイントの切り替え	アクションまたはダイアログにブレイクポイントを設定するためにこのボタンをクリックしてください。
	MSI デバッグ	デバッグを開始するとき、またはデバッグ中にセットアップを続行するためにこのボタンをクリックしてください。
	ステップ オーバー	現在のアクションをステップオーバーするときに、このボタンをクリックしてください。
	ステップイン	現在のアクションにステップインして登録してあるデバッガーを起動する場合、このボタンをクリックします。
	すべてのブレークポイントの削除	MSI デバッグで設定したブレークポイントをすべて削除するときに、このボタンをクリックしてください。
	MSI デバッガーの停止	デバッグを停止するときにこのボタンをクリックしてください。

出力ウィンドウ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSIX*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

プロジェクトのリリースをビルドすると、[出力] ウィンドウが開き、ビルド情報が表示されます。その他のタスク別情報も表示されます。たとえば、プロジェクトを検証すると、[出力] ウィンドウに検証結果が表示されます。以前のバージョンの InstallShield で作成されたプロジェクトを開くと、[出力] ウィンドウには、アップグレード情報が表示されます。

出力ウィンドウには次のようなタブがあります。

テーブル 15 • 出力ウィンドウのタブ

タブ	説明
ビルド	配布出力情報を保存し、ビルド出力を表示します。これには、テキストファイルとして保存された出力ファイルへのリンクが含まれます。
検証	検証結果を表示します。
結果	プロジェクトの規則情報と XML ファイルの変更のテストの詳細を表示します。
コンパイル	[ビルド] メニューの [コンパイル] をクリックしたときに InstallScript 情報を表示します。
タスク	プロジェクトをビルド、コンパイルまたは検証したときのエラーおよび警告コードの説明を表示します。表示されるエラーおよび警告コード はそれぞれ、ナレッジベース記事にリンクしています。 ***ヒント:*** *エラーまたは警告のコードを右クリックして、[ダイレクト エディター] を選択します。エラーに関連する影響のあった箇所は、ダイレクト エディターでハイライト表示されます。*
配布	リリースを仮想マシンに配布する方法についての情報が表示されます。詳細については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

ダイアログ ボックス リファレンス

このセクションでは、InstallShield ユーザー インターフェイスで使用できる各ダイアログ ボックスについて説明します。

- [.NET インストーラー クラス引数] ダイアログ ボックス
- [引数/値ペアの追加] ダイアログ ボックス
- [既存のメディアを追加] ダイアログ ボックス
- [このパッケージのファイルを追加する] ダイアログ ボックス
- [MIME の種類を追加] ダイアログ ボックス
- [新しいメソッドの追加] ダイアログ ボックス
- [新しいプロパティの追加] ダイアログ ボックス
- [定義済み検索の追加] ダイアログ ボックス
- [ソース管理に追加] ダイアログ ボックス
- [アプリケーション拡張子マッピング] ダイアログ ボックス
- [アプリケーションのマッピング] ダイアログ ボックス
- [コンポーネントの関連付け] ダイアログ ボックス
- [DIM の関連付け] ダイアログ ボックス
- [バッチ ビルド] ダイアログ ボックス
- [ディレクトリの参照] ダイアログ ボックス
- ディレクトリ/ショートカットのターゲットを参照するダイアログ ボックス
- [ファイルの参照] ダイアログ ボックス
- [実行するファイルの参照] ダイアログ ボックス
- [ショートカット アイコンの参照] ダイアログ ボックス
- [タイトル ターゲットの参照] ダイアログ ボックス
- [証明書の選択] ダイアログ ボックス
- [チェックイン] ダイアログ ボックス
- [チェックアウト] ダイアログ ボックス
- [コンポーネントのプロパティ] ダイアログ ボックス
- [条件ビルダー] ダイアログ ボックス
- [コンテンツ ソース パス] ダイアログ ボックス
- [InstallScript MSI の変換] ダイアログ ボックス
- [新しいコンポーネントの作成] ダイアログ ボックス
- [新しい機能の作成] ダイアログ ボックス

- [カスタム エラー処理] ダイアログ ボックス
- [カスタム エラー] ダイアログ ボックス
- [検証設定のカスタマイズ] ダイアログ ボックス
- [データ言語] ダイアログ ボックス
- [プロパティの削除] ダイアログ ボックス
- [依存関係] ダイアログ ボックス
- [ダイアログのイメージ] ダイアログ ボックス
- [セットアップのデジタル署名] ダイアログ ボックス
- [差分] ダイアログ ボックス
- [ダイナミック ファイル リンクの設定] ダイアログ ボックス
- [データの編集] ダイアログ ボックス
- [オプション リストの編集] ダイアログ ボックス
- [レジストリ データの編集] ダイアログ ボックス
- [仮想マシンの構成を編集] ダイアログ ボックス
- [エラー マッピングのプロパティ] ダイアログ ボックス
- [言語の除外] ダイアログ ボックス
- [テーブルのエクスポート] ダイアログ ボックス
- [機能条件ビルダー] ダイアログ ボックス
- [ファイルの詳細] ダイアログ ボックス
- [ファイルのプロパティ] ダイアログ ボックス
- [ファイルのプロパティ] ダイアログ ボックス (InstallScript インストール ロジェクト)
- ファイル 削除の [プロパティ] ダイアログ ボックス
- [検索/置換] ダイアログ ボックス
- [プロジェクト内の文字列 ID を検索する] ダイアログ ボックス
- [フォルダーのプロパティ] ダイアログ ボックス
- [関数シグネチャ] ダイアログ ボックス
- [一般オプション - 詳細] ダイアログ ボックス
- [一般オプション - 別のディスク ファイル] ダイアログ ボックス
- [履歴] ダイアログ ボックス
- [ダイアログのインポート] ダイアログ ボックス
- [InstallScript ファイルのインポート] ダイアログ ボックス
- [SQL スクリプト ファイルのインポート] ダイアログ ボックス
- [アクションの挿入] ダイアログ ボックス

- [InstallShield前提条件のプロパティ] ダイアログ ボックス
- [言語] ダイアログ ボックス
- [リンク タイプ] ダイアログ ボックス
- [Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックス
- 小文字のコンポーネントディレクトリ – 警告
- [メディア ファイルのプロパティ] ダイアログ ボックス
- [マージ モジュールの構成可能な値] ダイアログ ボックス
- [マージ モジュールのプロパティ] ダイアログ ボックス
- [メソッドの参照] ダイアログ ボックス
- [メソッド シグネチャ] ダイアログ ボックス
- [MIME の種類] ダイアログ ボックス
- [ダイナミック リンクの変更] ダイアログ ボックス
- [プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル
- [モジュール依存関係] ダイアログ ボックス
- [除外モジュール] ダイアログ ボックス
- [MSI ログファイル] タブ
- [MSI 値] ダイアログ ボックス
- [MST SIS の設定] ダイアログ ボックス
- [複数行文字列値] ダイアログ ボックス
- [新規依存関係] ダイアログ ボックス
- [新規ファイル] ダイアログ ボックス
- [新規プロジェクト] ダイアログ ボックス
- [オペレーティング システム] ダイアログ ボックス
- [オプション] ダイアログ ボックス
- その他の IIS プロパティ ダイアログボックス
- [その他のウィンドウ スタイル] ダイアログ ボックス
- [出力] ダイアログ ボックス
- [上書き] ダイアログ ボックス
- [パッチ シーケンス] ダイアログ ボックス
- [パス変数のオーバーライド] ダイアログ ボックス
- [パス変数の推奨] ダイアログ ボックス
- ファイルとディレクトリの [アクセス許可] ダイアログ ボックス
- レジストリ キーの [アクセス許可] ダイアログ ボックス

- [プラットフォーム スイート] ダイアログ ボックス
- [プラットフォーム] ダイアログ ボックス
- [ビルド後のイベント] ダイアログ ボックス
- [ビルド前のイベント] ダイアログ ボックス
- [圧縮前のイベント] ダイアログ ボックス
- [ファイルの準備イベント] ダイアログ ボックス
- [前提条件設定] ダイアログ ボックス
- [特権] ダイアログ ボックス
- [製品条件ビルダー] ダイアログ ボックス
- [プロジェクト設定] ダイアログ ボックス
- [キー名の変更] ダイアログ ボックス
- [必要な機能] ダイアログ ボックス
- [競合の解決] ダイアログ ボックス
- [名前を付けて保存] ダイアログ ボックス
- [スクリプト変換] ダイアログ ボックス
- [スクリプト エディターのプロパティ] ダイアログ ボックス
- [SCM シャットダウンの遅延] ダイアログ ボックス
- [セキュリティ記述子] ダイアログ ボックス
- [ファイルの選択] ダイアログ ボックス
- [メディアの場所選択] ダイアログ ボックス
- [文字列の選択] ダイアログ ボックス
- シリアル番号違反ダイアログ ボックス
- [サーバーの場所] ダイアログ ボックス
- 自動開始サービスの遅延に関する [サービスのオプション] ダイアログ ボックス
- 回復操作の実行のための [サービスのオプション] ダイアログ ボックス
- [サービス SID] ダイアログ ボックス
- [ファイルの URL の設定] ダイアログ ボックス
- [設定] ダイアログ ボックス
- [セットアップ言語] ダイアログ ボックス
- [SQL Server 要件] ダイアログ ボックス
- [優先] ダイアログ ボックス
- [UI 言語] ダイアログ ボックス
- [マージ モジュール検索パスのアップデート] ダイアログ ボックス

[.NET インストーラー クラス引数] ダイアログ ボックス

[.NET Installer クラス引数] ダイアログ ボックスには、.NET Installer クラスに渡す引数が表示されます。ダイアログ ボックスを起動するには、".NET Installer クラス引数" コンポーネント設定の省略記号ボタン (...) をクリックします。追加する引数の下の [引数の追加] ボタンをクリックして、[引数/値ペアの追加] ダイアログ ボックスを起動します。

インストール中にさまざまな箇所で呼び出される Installer クラスには 4 つのメソッド (Install、Commit、Uninstall、および Rollback) があります。このダイアログを使用して、各実行コンテキストの Installer クラスの引数リストを指定します。

メモ: *Install カスタム アクションは、Installer クラス カスタム アクションを含むコンポーネントがインストールされている時に実行されます。Uninstall カスタム アクションは、Installer クラスカスタム アクションを含むコンポーネントがアンインストールされている時に実行されます。この 2 つのタイプは、StartServices の直後と RegisterUser の前に [実行] シーケンスで実行されます。*

引数の使用

すべてのメソッドのデフォルト引数は */LogFile=* です。これらのメソッドが実行されると、ログ記録なしの結果となります。この引数は、Windows Installer プロパティを [かっこ] に入れて使用して完成させることができます。引数を追加するには、適切な編集フィールドに入力するか、[引数の追加] ボタンを使用します。

構文規則

OK をクリックしたとき、InstallShield は編集フィールドの内容を検証し、構文に誤りがある場合は警告を表示します。構文規則には次のようなものがあります。

- 常に正しく引用符を用いなくてはなりません。
- エスケープ文字が利用できます。引数で *¥"* を使用する場合、引用の一致をチェックするときにはカウントされません。
- スペースで区切られたすべての引数トークンは、スラッシュを名前の前に付けて、名前の後には等号(=)を付けます。等号の右側の値はオプションです。
- 引数の中のスペースは、引数トークンに何を含めるか決定している場合にはカウントされません。たとえば */arg="hi there"* という引数では、*"hi there"* は */arg*で始まる引数トークンのすべての部分を表します。引数 */arg=hi there* では、*there* は個別の引数トークンであり、無効です。

[引数/値ペアの追加] ダイアログ ボックス

このダイアログは [.NET Installer クラス引数] ダイアログ ボックス の [引数の追加] ボタンをクリックすると表示されます。引数の作成が終わったら、[OK] をクリックすると引数が追加されます。

ダイアログ オプション

引数名

編集フィールドに引数名を入力します。

引数値

オプション ボタンのリストから値の 1 つを選択して引数を作成します。カスタム引数を作成するには [カスタム] オプションを選択します。

プロパティ

メニューからプロパティを選択して、インストール プロジェクトに存在するプロパティを参照します。

Directory

ドロップダウンメニューを利用して、インストール プロジェクトにあるディレクトリにリファレンスを追加することができます。

カスタム

カスタム引数を作成するには、編集フィールドに引数値を入力します。

[既存のメディアを追加] ダイアログ ボックス

[既存のメディアを追加] ダイアログ ボックスは、リリース ウィザードの [アップデート] パネルで [追加] ボタンをクリックすると開きます。このダイアログ ボックスではリリースのリストから現在のプロジェクトにあるリリースを選択できます(現在のリリースを除く)。

[このパッケージのファイルを追加する] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[このパッケージのファイルを追加する] ダイアログ ボックスは、アドバンスト UI またはスイート/アドバンスト UI インストールの [パッケージ] ビューで、パッケージ (.msi パッケージや .exe パッチなど) を追加するときに開きます。

このダイアログ ボックスで、追加するパッケージ ファイルに、追加するパッケージの近くにあるその他のファイルとフォルダーが必要であるかどうかを指定できます。たとえば、非圧縮の .msi パッケージをプロジェクトに追加する場合、場合によって、msi パッケージによってターゲット システムにインストールされるファイルを含むフォルダーやその他のファイルを含める必要があります。これらのフォルダーおよびその他のファイルは通常、.msi パッケージが含まれている同じフォルダーに保存されます。

[このパッケージのファイルを追加する] ダイアログ ボックスでは、追加ファイルのダイナミック リンクを使用するかどうかも指定することができます。

以下は、[このパッケージのファイルを追加する] ダイアログ ボックスで使用できるオプションです:

テーブル 1 ・ [このパッケージのファイルを追加する] ダイアログ ボックスの設定

設定	説明
追加なし	プロジェクトには、パッケージ ファイルのみ含められます。プロジェクトに追加するパッケージに必要な追加ファイルが存在しない場合、このオプションを選択します。 このオプションは、パッケージが圧縮されている場合、選択できます。
追加ファイルを追加する	パッケージ ファイルと同じフォルダー内にあるファイルが含められます。 パッケージ ファイルと同じフォルダーの中のサブフォルダーや、そのサブフォルダーの中のファイルは含められません。
サブフォルダー内のファイルを追加する	追加するパッケージ ファイルを含むフォルダーのサブフォルダー内のファイルが含められます。 パッケージ ファイルと同じフォルダー内にあるファイルは含められません。
隣接するファイルとサブフォルダー内のファイルを追加する	パッケージ ファイルと同じフォルダー内にあるファイルが含められます。また、追加するパッケージ ファイルを含むフォルダーのサブフォルダー内のファイルも含められます。

テーブル 1 • [このパッケージのファイルを追加する] ダイアログ ボックスの設定 (続き.)

設定	説明
ビルド時にダイナミックに見つかったファイルを追加する	追加ファイルをダイナミック ファイル リンクとして含める場合、このチェック ボックスを選択します。追加ファイルをスタティック ファイル リンクとして含める場合、このチェック ボックスをクリアします。デフォルトで、このチェック ボックスがクリアされます。 パッケージをプロジェクトに追加するとき、すべての追加ファイルの名前が分かっている場合、任意の追加ファイルに対してスタティック ファイル リンクを使うことができます。パッケージの追加ファイルが時間と共に変わる可能性がある場合、任意の追加ファイルに対してダイナミック ファイル リンクを使用することができます。 ***メモ•*** *InstallShield では、パッケージ ファイル (.ms、.msp、InstallScript パッケージ、または .exe) に対して、常にスタティック リンクが含まれます。従って、パッケージ ファイルの名前は、ビルドが実行される度に変わることはありません。* *[追加なし] オプションがクリックされた場合、このチェック ボックスを選択しても、効果はありません。* スタティックおよびダイナミック ファイル リンクについての詳細は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージにおけるスタティック ファイルとダイナミック ファイルの違い」を参照してください。

[MIME の種類を追加] ダイアログ ボックス

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[MIME の種類を追加] ダイアログ ボックスを使って、ファイル拡張子名とプログラム、又はそれらのファイルを処理するインタプリタ間のマッピングを追加又は変更します。このダイアログ ボックスは、[MIME の種類] ダイアログ ボックスの [追加] または [編集] ボタンをクリックすると開きます。

テーブル 2 - [MIME の種類を追加] ダイアログ ボックスの設定

設定	説明
ファイル名拡張子	ファイル名拡張子を入力します(例、**.abc**)。これはスタティック ファイル名拡張子です。 実行可能ファイルにワイルドカードのアプリケーション マッピングを使用するには、アスタリスク (*) を入力します。
MIME タイプ	MIME の種類を入力します (例、**application/octet-stream**)。

[新しいメソッドの追加] ダイアログ ボックス

プロジェクト:

このダイアログ ボックスを使用して、新しいメソッドの名前、戻り値の型、およびパラメーターの種類を指定し、対応するコードをオブジェクトスクリプトに自動貼り付けることができます。このダイアログ ボックスは、InstallScript ビューの [メソッド] フォルダーを右クリックし、[新しいメソッドを追加する] を選択したときに開きます。

Properties

テーブル 3 - メソッドのプロパティ

プロパティ	説明
名前	メソッドの名前を入力します。
戻り値の型	このメソッドによって返されるデータ型を選択します。文字列、数値、ブール値、および配列の中から選択できます。
パラメーター リスト	メソッドに渡すことのできるパラメーターの種類を入力します。

[新しいプロパティの追加] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

[新しいプロパティの追加] ダイアログ ボックスでは、新しいプロパティの名前と属性を指定し、対応するコードをスクリプトに自動的に貼り付けることができます。このダイアログ ボックスは、InstallScript ビューの [プロパティ] フォルダーを右クリックしてから、[新しいプロパティの追加] をクリックすると開きます。

テーブル 4 - [新しいプロパティの追加] ダイアログ ボックスの設定

設定	説明
プロパティ名	これは、このプロパティのグローバル名です。そのため、スクリプトからこのプロジェクトの読み取りまたは書き込みする場合は、この名前を使用する必要があります。このボックスに入力した名前は、ローカル変数名プロパティに追加されます。
データ型	プロパティに必要なデータ型によって、[文字列]、[数値]、または [ブール値] を選択します。ローカル変数名は、ここでの選択によって変わります。
アクセスのメソッド	プロパティの適切なアクセス レベルを選択します。選択可能なオプションは [読み取り/書き込み]、[読み取り専用]、および [書き込み専用] です。
配列	プロパティが配列の場合に、このチェック ボックスを選択します。 ***メモ-*** *オブジェクトの設計中のインターフェイスに InstallShield ストック ウィザードを使用する場合は、配列プロパティは使用できません。配列プロパティを使用できるカスタム ウィザードを作成するか、配列を使用せずにプロパティを再設計します。*
ローカル変数名	デフォルトの名前を受け入れるか、独自の変数名を作成します。この変数名は、スクリプト内から変数を呼び出す際に使用します。
デフォルト値	プロパティの開始値を入力します。必ず、選択したデータ型と値が一致するようにします。文字列値を指定した場合は、スクリプト内と同じように入力します。つまり、文字列値を引用符で囲み、特殊文字には適切なエスケープシーケンスを使用します (たとえば、「¥¥」のように円記号を入力します)。文字列 ID やシステム変数を InstallScript オブジェクト プロジェクトに入力しないでください。プロパティの値の初期設定は、オブジェクトがプロジェクトに追加される際に行われ、この時点では文字列 ID およびシステム変数は初期設定されていません。

[定義済み検索の追加] ダイアログ ボックス

[デフォルト検索の追加] ダイアログ ボックスは、[システム検索] ビュー内でグリッドを右クリックして [デフォルト検索の追加] をクリックすると開きます。

このボックス内にあるリストビューからデフォルト値を選択して OK をクリックすると、システム検索ビューのグリッドに検索が追加されます。

レポジトリにパブリッシュされているシステム検索を表示するには、レポジトリの検索を表示チェック ボックスを選択します。

[ソース管理に追加] ダイアログ ボックス

このダイアログは、InstallShield を使ってソース管理プログラムにプロジェクトを追加すると表示されます。

ダイアログ オプション

コメント

ファイルについて記述するコメントを入力します。これらのコメントは、ソース管理プログラムに格納されます。

チェックアウト状態を保持

.ism ファイル(テキスト形式)を追加して作業ディレクトリに取り出したままにする場合、このオプションを選択します。

XML に変換する

このオプションを使うと、.ism ファイルはソース管理により適した XML を使用します。バイナリ形式でプロジェクトをソース管理に追加すると、このオプションが使用できるようになり、デフォルトで選択されます。このオプションを選択したままにしておくことをお勧めします。

*メモ・ プロジェクトファイルフォーマットを XML またはバイナリ形式に変換しても、プロジェクトファイルの拡張子は *.ism のまま変更されません。*

[アプリケーション拡張子マッピング] ダイアログ ボックス

[アプリケーション拡張子マッピング] ダイアログ ボックスを使って、ファイル拡張子名とプログラム、又はファイルを処理するインタプリタ間のマッピングを追加又は変更します。このダイアログ ボックスは[アプリケーション マッピング] ダイアログ ボックスで [追加] をクリックした時に表示されます。

テーブル 5 · [アプリケーション拡張子マッピング] ダイアログ ボックスの設定

設定	説明
拡張子	アプリケーション (例、*.abc*) に関連付けられたファイル名の拡張子を入力します。 実行可能ファイルにワイルドカードのアプリケーション マッピングを使用するには、アスタリスク (*) を入力します。
実行可能ファイル	パスを入力するか、[参照] をクリックして [実行可能ファイルの選択] ダイアログ ボックスを立ち上げます。ここでマップするプロジェクトの実行可能ファイルを指定します。実行可能ファイルの名前 (.exe or .dll) を入力、又は [参照] ボタンを使ってファイルを検索します。実行可能ファイルは Web サーバーのローカル ハード ドライブに配置します。

テーブル 5 ・[アプリケーション拡張子マッピング] ダイアログ ボックスの設定 (続き.)

設定	説明
動詞	[動詞] セクションには、どの HTTP 動詞をアプリケーションにパスすべきかを指定できます。 ・ **すべての動詞** — すべての動詞を含める場合、このオプションを選択します。すべてのリクエストをアプリケーションに渡します。 ・ **最大数** — このオプションを選択して、アプリケーションに渡す HTTP 動詞を指定することができます。動詞をカンマで分けます。
スクリプト エンジン (IIS 6 以前のみ)	アプリケーションを実行許可なしにディレクトリ内で実行させたい時、このチェック ボックスを選択します。基本的にこの設定はインタープリタへマップされた ASP や IDC といったスクリプトベースのアプリケーション用のものです。 スクリプト マップされたアプリケーションを実行するためには、実行許可プロパティで "スクリプトのみ" 又は "スクリプトと実行可能ファイル" オプションを選択します。スクリプトマップされたアプリケーションのみを実行する場合は [スクリプトのみ] オプションを選択します。スクリプト マップされたアプリケーションと実行可能ファイル (.exe and .dll) を実行する場合は [スクリプトと実行可能ファイル] を選択します。 この設定は IIS 6 以前に適用します。IIS 7 は、この設定を無視します。
そのファイルの存在を確認 (IIS 6 以前のみ)	Web サーバーに対して、要求されたスクリプト ファイルが存在していることと、要求しているユーザーがそのスクリプト ファイルへのアクセス許可を持っていることを確認するよう指示するには、このチェック ボックスを選択します。 スクリプト ファイルが存在しない場合、もしくはエンドユーザーがアクセス権を持っていない場合はブラウザーに適切な警告メッセージが表示され、スクリプト エンジンは呼び出されません。このオプションは、たとえば Perl インタープリタのような、スクリプトがアクセス不能の場合に CGI レスポンスを送らない CGI 以外の実行可能ファイルにマップされたスクリプトに使うと便利です。 ***メモ・** スクリプトがサーバーとスクリプトエンジンによって 2 度開かれているので、このチェック ボックスが選択されているときにはパフォーマンスが低下します。* この設定は IIS 6 以前に適用します。IIS 7 は、この設定を無視します。

[アプリケーションのマッピング] ダイアログ ボックス

[アプリケーションのマッピング] ダイアログ ボックスを使って、ファイル拡張子名とそのファイルを処理するアプリケーションとの間のマッピングを編集または削除することができます。

[アプリケーションのマッピング] ダイアログ ボックスは、[IIS 構成] ビュー内から利用することができます。このダイアログ ボックスを開くには、エクスプローラーで Web サイト、アプリケーション、または仮想ディレクトリをクリックします。次に、"アプリケーションのマッピング" 設定で省略記号ボタン (...) をクリックします。

*メモ・ [動詞]列にアスタリスク(*)が表示されたときは、指定した拡張子にすべての動詞が使用されます。*

テーブル 6 ・ [アプリケーションのマッピング] ダイアログ ボックスの設定

設定	説明
追加	ファイル名拡張子とプログラム、またはこれらのファイルを処理するインタープリタ間とのマッピングを追加するには、このボタンをクリックします。そうすると、[アプリケーション拡張子マッピング] ダイアログ ボックス が開きます。
編集	既存するアプリケーション マッピングを編集するには、マッピングを選択して、このボタンをクリックします。
削除	既存するアプリケーション マッピングを削除するには、拡張子を選択して、このボタンをクリックします。

[コンポーネントの関連付け] ダイアログ ボックス

[コンポーネントの関連付け] ダイアログ ボックスで、コンポーネントと [セットアップのデザイン] ビューの機能を関連付けることができます。現在の機能と関連付けるコンポーネントを 1 つまたは複数選択して、[OK] をクリックします。

メモ・ 機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。

コンポーネントと機能の関係については、「新しいコンポーネントを機能に関連付ける」を参照してください。

[DIM の関連付け] ダイアログ ボックス

[DIM の関連付け] ダイアログ ボックスで、DIM と [セットアップのデザイン] ビューの機能を関連付けることができます。現在の DIM と関連付ける DIM を 1 つまたは複数選択して、[OK] をクリックします。

メモ・ 機能に関連付けられていない DIM は、親のないコンポーネントのアイコンで表示されます。

DIM と機能の関係については、「DIM リファレンスを機能に関連付ける」を参照してください。

[バッチ ビルド] ダイアログ ボックス

このダイアログには、現在のプロジェクトで定義したすべての製品構成とリリースを含むツリー コントロールが表示されます。バッチ ビルドに含めるリリースを選択します。

ダイアログ オプション

ツリー コントロール

該当するボックスをクリックして、ビルドするリリースを選択します。製品構成アイコンを選択すると、その製品構成に関連したすべてのリリースが選択されます。

すべて選択

ツリーの中のすべてのリリースを選択する場合には、このボタンをクリックします。

選択の解除

ツリーの中のすべてのリリースを選択解除する場合には、このボタンをクリックします。

ビルド

選択したすべてのリリースをビルドするには、このボタンをクリックします。

[ディレクトリの参照] ダイアログ ボックス

[ディレクトリの参照] ダイアログ ボックスを使用して、ディレクトリの参照、新規作成、名前の変更、または削除を行うことができます。

ダイアログ ボックス オプション

インストール先ディレクトリ

このフィールドには、現在使用できるすべてのインストール先ディレクトリが一覧表示されます。このフィールドで、ディレクトリの選択、 作成 、 名前の変更 、または 削除 を行うことができます。

ディレクトリの選択

タスク

ディレクトリを選択するには、次の操作を実行します。

1. 選択するディレクトリをクリックします。
2. [OK] をクリックします。

ディレクトリの新規作成

タスク

新しいディレクトリを作成するには、次の操作を実行します。

1. ディレクトリまたはインストール先コンピューターを選択して **Insert** キーを押すか、右クリックしてコンテキスト メニューで **[新規作成]** を選択します。これにより、選択したフォルダーまたはインストール先コンピューターの下にディレクトリが作成されます。
2. ディレクトリ名を入力します。
3. 必要であれば ディレクトリ識別子 を入力します。

ディレクトリ名の変更

タスク ***ディレクトリの名前を変更するには、次の操作を実行します。***

1. ディレクトリまたはインストール先コンピューターを選択して F2 キーを押すか、右クリックしてコンテキストメニューで **[名前の変更]** を選択します。
2. 新しいディレクトリ名を入力します。定義済みのディレクトリ名は変更できないことにご注意ください。
3. 必要に応じて、ディレクトリ識別子を変更して、ディレクトリの新しい名前と統一させます。

ディレクトリの削除

タスク ***ディレクトリを選択するには、次の操作を実行します。***

ディレクトリを選択して **Delete** キーを押すか、右クリックしてコンテキスト メニューで **[削除]** を選択します。

定義済みディレクトリは削除できないことにご注意ください。

メモ: *ディレクトリを削除すると、選択したディレクトリの下にあるすべてのサブディレクトリも一緒に削除されます。*

ディレクトリ識別子

[ディレクトリ識別子] フィールドを使用して、使いやすい名前を付けることができます。たとえば、ProgramFilesFolder¥MyProgram¥Graphics¥Jpg というディレクトリがあった場合、ディレクトリ パスを識別するために、JPG という名前を使用することができます。IDE の **"コンポーネントインストール先"** フィールドには、このパスは {JPG} [ProgramFilesFolder]MyProgram¥Graphics¥Jpg と表示されます。

メモ: *ディレクトリ識別子は、有効な Windows Installer 識別子でなくてはなりません。機能の場合、ディレクトリ識別子はすべて大文字である必要があります。*

ディレクトリ/ショートカットのターゲットを参照するダイアログ ボックス

[ディレクトリの参照] ダイアログ ボックスを使用して、ディレクトリの参照、新規作成、名前の変更、または削除を行うことができます。[ショートカット ターゲットの参照] ダイアログ ボックスを使用して、ディレクトリまたはファイルの参照、新規作成、名前の変更、または削除を行うことができます。

ダイアログ ボックス オプション

インストール先ディレクトリ

このフィールドには、現在使用できるすべてのインストール先ディレクトリが一覧表示されます。このフィールドで、ディレクトリの選択、 作成 、 名前の変更 、または 削除 を行うことができます。

ディレクトリの選択

タスク ***ディレクトリを選択するには、次の操作を実行します。***

1. 選択するディレクトリをクリックします。
2. **OK**をクリックします。

ディレクトリの新規作成

タスク ***新しいディレクトリを作成するには、次の操作を実行します。***

1. ディレクトリまたはインストール先コンピューターを選択して Insert キーを押すか、右クリックしてコンテキスト メニューで **"新規作成"** を選択します。これにより、選択したフォルダーまたはインストール先コンピューターの下にディレクトリが作成されます。
2. ディレクトリ名を入力します。
3. 必要であれば ディレクトリ識別子 を入力します。

ディレクトリ名の変更

タスク ***ディレクトリの名前を変更するには、次の操作を実行します。***

1. ディレクトリまたはインストール先コンピューターを選択して F2 キーを押すか、右クリックしてコンテキストメニューで **[名前の変更]** を選択します。
2. 新しいディレクトリ名を入力します。定義済みのディレクトリ名は変更できないことにご注意ください。
3. 必要に応じて、ディレクトリ識別子を変更して、ディレクトリの新しい名前と統一させます。

ディレクトリの削除

タスク ***ディレクトリを選択するには、次の操作を実行します。***

ディレクトリを選択して **Delete** キーを押すか、右クリックしてコンテキスト メニューで **[削除]** を選択します。定義済みディレクトリは削除できないことにご注意ください。

メモ: *ディレクトリを削除すると、選択したディレクトリの下にあるすべてのサブディレクトリも一緒に削除されます。*

ショートカットのハードコード化

タスク ***C:¥Winnt¥Notepad.exe などのパスへのショートカットをハードコード化するには、以下の手順に従います:***

まず、"C:" と "Winnt" という 2 つのフォルダーを作成します。そして **[ディレクトリの参照]** ダイアログ内のファイル名フィールド に「Notepad.exe」を入力、またはルート ノードを選択してファイル名フィールドに「C:¥Winnt¥Notepad.exe」を入力します。

ディレクトリ識別子

[ディレクトリ識別子] フィールドを使用して、使いやすい名前を付けることができます。たとえば、ProgramFilesFolder¥MyProgram¥Graphics¥Jpg というディレクトリがあった場合、ディレクトリ パスを識別するために、JPG という名前を使用することができます。InstallShield インターフェイスの **"コンポーネントのインストール先"** フィールドには、このパスは {JPG} [ProgramFilesFolder]MyProgram¥Graphics¥Jpg と表示されます。

メモ: *ディレクトリ識別子は、有効な MSI 識別子 である必要があります。機能の場合、ディレクトリ識別子はすべて大文字である必要があります。*

[**ファイル名**の編集] ボックスにファイル名を入力することができます。この場合、ショートカットのターゲットとして選択された directory¥ にファイル名を入力することになります。

[ファイルの参照] ダイアログ ボックス

[ファイルを開く] ダイアログ ボックスは、[リリース] ビューで、リリースの [署名] タブにある "含めるパターンとファイル" 設定または "除外するパターンとファイル" 設定で省略記号ボタン (...) をクリックすると表示されます。[ファイルを参照] ダイアログ ボックスで、プロジェクトのスタティック ファイルに署名を行うかどうかを指定できます。また、ワイルドカード文字としてアスタリスク (*) を使用することもできます。ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイル含め、特定のパターンに一致するすべてのファイルに署名を行う場合、特に便利です。

テーブル 7 · [ファイルの参照] ダイアログ ボックスの設定

設定	説明
署名するファイルを選択	このボックスは、"含めるパターンとファイル" 設定で省略記号ボタン (...) をクリックすると表示されます。 このダイアログ ボックスでは、**[ファイルの種類を表示する]** 一覧で選択したファイルの種類に一致するプロジェクトにあるすべての静的に含められたファイルが一覧表示されます。いくつかのデフォルト ファイル パターン (*.dll など) も一覧表示されます。 ビルド時に InstallShield で署名するプロジェクト内のファイルの種類に対応するファイルとファイル パターンのチェック ボックスを選択します。

テーブル 7 • [ファイルの参照] ダイアログ ボックスの設定 (続き.)

設定	説明
署名をスキップするファイルを選択	このボックスは、"除外するパターンとファイル" 設定で省略記号ボタン (...) をクリックすると表示されます。 このダイアログ ボックスでは、**[ファイルの種類を表示する]** 一覧で選択したファイルの種類に一致するプロジェクトにあるすべての静的に含められたファイルが一覧表示されます。いくつかのデフォルト ファイル パターン (*.dll など) も一覧表示されます。 ビルド時に、あるファイルとファイル パターンが InstallShield によって署名されるのを避ける場合、該当するファイルとファイル パターンのチェック ボックスを選択します。
ファイルの種類を表示する	この一覧を使用して、**[署名するファイルを選択]** ボックスまたは **[署名をスキップするファイルを選択]** ボックスで表示されるファイルの種類をフィルターします。

Windows ロゴ • *インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

このダイアログ ボックスで [OK] をクリックすると、"含めるパターンとファイル" 設定の下に新しい "含める" 設定、または "除外するパターンとファイル" 設定の下に新しい "除外する" 設定が追加されます。

署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、"含める" 設定および "除外する" 設定に *.exe を指定すると、InstallShield は .exe ファイルに署名を行いません。

[実行するファイルの参照] ダイアログ ボックス

[実行するファイルの参照] ダイアログ ボックスは、[スケジュール タスク] ビューの "ターゲット" 設定内の省略記号 (...) ボタンをクリックすると開きます。このダイアログ ボックスでは、スケジュール タスクで実行するファイルに対して、ターゲット システム上でのファイル名と場所を指定できます。

テーブル 8 • [実行するファイルの参照] ダイアログ ボックスの設定

設定	説明
インストール先ディレクトリ	スケジュール タスクとして実行するファイルを含むターゲット システム上のディレクトリを選択します。 新しいディレクトリを指定する場合、新しいディレクトリを含める場所を右クリックして、[新しいディレクトリ] をクリックします。 このボックスで空の定義済みディレクトリを表示したり非表示にしたりするには、このボックスを右クリックして、[空の定義済みディレクトリを表示] をクリックします。

テーブル 8 - [実行するファイルの参照] ダイアログ ボックスの設定 (続き.)

設定	説明
ディレクトリ識別子	[インストール先ディレクトリ] ボックスで作成したインストール先ディレクトリを選択した場合、この設定を使って、ディレクトリに分かりやすい名前を付けることができます。たとえば、[ProgramFilesFolder]¥MyProgram¥Graphics¥JPG というディレクトリがあった場合、ディレクトリ パスを識別するために、JPG という名前を使用することができます。InstallShield インターフェイスの "コンポーネントのインストール先" フィールドには、このパスは **{JPG} [ProgramFilesFolder]MyProgram¥Graphics¥JPG** と表示されます。 [インストール先ディレクトリ] ボックスで定義済みディレクトリを選択した場合、この設定は読み取り専用になっています。
ファイル名	スケジュール タスクで起動するファイルの名前を入力します。

[ショートカット アイコンの参照] ダイアログ ボックス

プロジェクト- *これは、InstallScript プロジェクトに適用します。*

[ショートカット アイコンの参照] ダイアログ ボックスを使って、ターゲット システムでのアイコン ファイル (.ico) の場所を指定したり、アイコン リソースを含む .exe または .dll ファイルを指定したりできます。このダイアログ ボックスは、[ショートカット] ビューにある "アイコン" 設定内の省略記号 (...) ボタンをクリックすると開きます。

テーブル 9 - [ショートカット アイコンの参照] ダイアログ ボックスの設定

設定	説明
インストール先ディレクトリ	アイコン ファイルを含むターゲット システム上にあるディレクトリを選択します。
現在の値	この読み取り専用の設定には、選択されたアイコン ファイルを含むディレクトリが表示されます。
ファイル名	アイコン ファイル (.ico) の名前、または、アイコン リソースを含む .exe または .dll ファイルの名前を入力します。

ヒント- *ハードコード化されたファイル パスを使用する場合、[ショートカット アイコンの参照] ダイアログ ボックスを使って、ディレクトリのシステム変数を使用する代わりに、"アイコン ファイル" 設定にパスを手入力で入力します。"アイコン ファイル" 設定は、[ショートカット] ビューの "アイコン" 設定の下にあります。*

[タイトル ターゲットの参照] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallShield は、スタート画面上のデスクトップ アプリのタイルの外観を構成することができます。[タイル ターゲットの参照] ダイアログ ボックスを使って、タイルの外観を構成するアプリケーションの .exe ファイルを参照して選択します。

テーブル 10 • [タイル ターゲットの参照] ダイアログ ボックスの設定

設定	説明
検索先	外観を構成するアプリの .exe ファイルを含むディレクトリを参照します。このリストからフォルダーを選択すると、プライマリ ペインにそのフォルダーのコンテンツが表示されます。プライマリ ペインで .exe ファイルをクリックして選択します。
ファイル名	プライマリ ペインで .exe ファイルを選択すると、ここにそのファイル名が表示されます。

[証明書の選択] ダイアログ ボックス

エディション: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*

リリース、パッチ、または QuickPatch パッケージのデジタル署名情報を構成するとき、[証明書の選択] ダイアログ ボックスを使って、ファイルに署名を行うために使用する証明書を指定します。InstallShield では、次のオプションから選択できます。

- 使用中のコンピューターにある、署名に使用する .pfx 証明書ファイル (EV がエクスポートした .cer ファイル) を指定できます。

- 署名に使用する証明書を含む証明書ストアを参照できます。

[証明書の選択] ダイアログ ボックスへのアクセス

[証明書の選択] ダイアログ ボックスへのアクセス方法は、リリース、パッチ、または QuickPatch パッケージのいずれの証明書情報を指定するかによって異なります。

タスク ***リリースの [証明書の選択] ダイアログ ボックスにアクセスするには、次の手順に従います:***

1. **[メディア]** の下にあるビュー バー/ビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[署名]** タブをクリックします。
4. **"デジタル証明書情報"** 設定で、省略記号ボタン (...) をクリックします。

タスク ***パッチの [証明書の選択] ダイアログ ボックスにアクセスするには、次の手順に従います:***

1. **[メディア]** の下にあるビュー バー/ビュー リストで、**[パッチのデザイン]** をクリックします。
2. **[パッチのデザイン]** エクスプローラーで、変更するパッチ構成を選択します。
3. **[デジタル署名]** タブをクリックします。
4. **"デジタル証明書情報"** 設定の横にある **[参照]** ボタンをクリックします。

タスク ***QuickPatch プロジェクトで [証明書の選択] ダイアログ ボックスにアクセスするには、次の手順に従います:***

1. **[パッチの設定]** の下にあるビュー バー/ビュー リストで、**[一般情報]** をクリックします。
2. **[一般情報]** エクスプローラーで、**[ビルドの設定]** を選択します。
3. **[デジタル署名]** タブをクリックします。
4. **"デジタル証明書情報"** 設定の横にある **[参照]** ボタンをクリックします。

[証明書の選択] ダイアログ ボックスの設定

テーブル 11 • [証明書の選択] ダイアログ ボックスの設定

設定	説明
ファイル (.pfx) を使用	.pfx ファイルまたは証明書ファイル (EV がエクスポートした .cer ファイル) を使ってビルド時にリリースに署名を行うには、このオプションを選択します。そのあと、.pfx または証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定します。ファイルへのパスを入力するか、省略記号ボタン (...) を使ってファイルの場所を参照します。
証明書ストアを使用	ビルド時にリリースにデジタル署名を行うために使用する証明書を含む証明書ストアを参照する場合、このオプションを選択してから、このオプションの下にあるサブ設定に値を入力します。
証明書ストア名	使用する証明書を含む証明書ストアの名前を選択します。選択可能なオプションは以下のとおりです: • Personal • 信頼されたルート証明機関 • エンタープライズの信頼 • 中間証明機関 この設定は、**[証明書ストアを使用]** オプションを選択すると有効になります。
証明書ストアの場所	使用する証明書を含む証明書ストアの場所を選択します。選択可能なオプションは以下のとおりです: • ユーザー • マシン この設定は、**[証明書ストアを使用]** オプションを選択すると有効になります。
証明書サブジェクト	使用する証明書のサブジェクトを入力するか、マシン上で使用可能な証明書のリストから選択します。 この設定は、**[証明書ストアを使用]** オプションを選択すると有効になります。
詳細表示	証明書の全般的な情報や、セキュリティ、証明書パスなどの情報が **[詳細表示]** オプションに一覧表示されます。
署名ダイジェスト	署名ダイジェストのハッシュ アルゴリズムを選択する、または証明書のハッシュに基づいて InstallShield が自動的に選択します。選択可能なオプションは以下のとおりです: • 証明書のハッシュに従う • SHA-1 • SHA-256 • デュアル (SHA-1 & SHA-256)

[チェックイン] ダイアログ ボックス

InstallShield を通して InstallShield プロジェクトをソース コード管理アプリケーションにチェックインするとき、[チェックイン] ダイアログ ボックスが開きます。

テーブル 12 - [チェックイン] ダイアログ ボックスの設定

設定	説明
Comments	ファイルに行なった変更を説明するコメントを入力します。これらのコメントは、ソース管理アプリケーションに格納されます。
チェックアウト状態を保持	変更された .isv ファイルをチェックインして、作業ディレクトリに取り出したままにする場合、このオプションを選択します。
差分	チェックインする .ism ファイル (テキスト フォーマット) と、以前のバージョンのファイルとを 1 行ごとに比較して表示するには、このボタンをクリックします。

[チェックアウト] ダイアログ ボックス

[オプション] ダイアログ ボックスの [ソース管理] タブで、**[チェックアウトにダイアログを使用する]** オプションを選択した場合、InstallShield を通してソース コード管理アプリケーションからプロジェクトをチェックアウトするとき、[チェックアウト] ダイアログ ボックスが開きます。

テーブル 13 - [チェックアウト] ダイアログ ボックスのオプション

オプション	説明
Comments	.ism ファイルをチェックアウトする理由を入力します。
アドバンスト UI	ソース コード管理アプリケーションの設定をすべて構成してプロジェクト ファイルをチェックアウトするには、このボタンをクリックします。

[コンポーネントのプロパティ] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

[コンポーネントのプロパティ] ダイアログ ボックスを使うと、ダイナミック ファイル リンクの設定や関連付けられた機能など、コンポーネントのプロパティを [ファイルとフォルダー] ビューから設定できます。

タスク ***ダイアログ ボックスを起動するには、次の手順を実行します。***

1. **[アプリケーション データ]** の下にあるビュー バー/ビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. コンポーネントが、**[インストール先コンピューターのフォルダー]** ペインに表示されていることを確認してください。

   コンポーネントが表示されない場合: [**インストール先コンピューターのフォルダー**] ペインで、**[インストール先コンピューター]**を右クリックし、[**コンポーネントの表示**] (Windows Installer ベースのプロジェクトの場合) または、[**コンポーネントとサブフォルダーを表示**] (InstallScript プロジェクトの場合) をクリックします。
3. コンポーネントを右クリックして、**[プロパティ]** を選択します。**[コンポーネントのプロパティ]** ダイアログ ボックスが開きます。

このダイアログ ボックスでは次のタブが利用できます。

- [全般] タブ
- [ファイルのリンク] タブ
- [機能] タブ

[全般] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネントのプロパティ] ダイアログ ボックス内の [全般] タブでは、選択したコンポーネントのプロパティを設定します。

テーブル 14 • [コンポーネントのプロパティ] ダイアログ ボックスの [全般] タブにある設定

設定	プロジェクトの種類	説明
共有	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このコンポーネントに含まれるファイルが共有ファイルの場合、このチェック ボックスを選択します。 詳細については、「共有ファイルの参照カウントを管理する」を参照してください。
上書きしない	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ターゲット マシンに既に存在する同ファイルの既存バージョンを置換しない場合、このチェック ボックスを選択します。 詳細については、「インストール前にファイルのバージョンを確認する」を参照してください。
パーマネント	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このコンポーネントのファイルを、アンインストール時にシステムから削除しない場合、このチェック ボックスを選択します。たとえば、検証規則 ICE09 では、インストール先が [SystemFolder] であるコンポーネントはすべてパーマネントでなければならないと定めています。 詳細については、「アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する」を参照してください。
Uninstall	InstallScript、InstallScript オブジェクト	このコンポーネントのファイルを、アプリケーションのアンインストール時に、削除しない場合、このチェック ボックスを選択します。 詳細については、「アンインストール時にコンポーネントのファイルと他の関連データをアンインストールするかどうかを指定する」を参照してください。

テーブル 14 • [コンポーネントのプロパティ] ダイアログ ボックスの [全般] タブにある設定 (続き.)

設定	プロジェクトの種類	説明
64 ビット	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このコンポーネントを 64 ビットとしてマークする場合、このチェック ボックスを選択します。64-bit コンポーネントがインストールに含まれていると、インストールを 32-bit のマシンで実行することはできません。
ビルド時に COM 情報を抽出	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	InstallShield でリリースをビルドするとき、常に COM 情報を抽出する場合、このチェック ボックスを選択します。COM インターフェイスが変更されやすい場合には、静的に取得または提供されたデータを COM 登録詳細設定に保持しておくよりも、このオプションをお勧めします。 コンポーネントの COM サーバー ファイルを自動登録ファイルとしてマークしてある場合、このチェック ボックスを選択しないでください。
自己登録	InstallScript、InstallScript オブジェクト	コンポーネントのファイルが自己登録ファイルの場合、このチェック ボックスを選択します。自己登録ファイルは、他の OLE アプリケーションが認識できるようにレジストリにそれ自体の情報を入れることができる OLE サーバーです。このようなファイルは、アンインストール時にこの情報をレジストリから削除することもできます。 ***プロジェクト:*** *InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトは、.dll ファイル、.exe ファイル、タイプ ライブラリ (.tlb および .olb ファイル) の自己登録をサポートします。Windows Installer ベースのプロジェクトの場合、自己登録は、コンポーネント レベルではなく、ファイル レベルで設定されます。*

[ファイルのリンク] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[コンポーネントのプロパティ] ダイアログ ボックスの [ファイルのリンク] タブでは、ダイナミック リンクのすべての設定が表示されます。

テーブル 15 · [コンポーネントのプロパティ] ダイアログ ボックスの [ファイルのリンク] タブにある設定

設定	説明
ソース	この列では、ソース ファイルが格納されている場所が表示されます。通常、パス変数が使用されます。
サブフォルダーを含む	この列は、各サブフォルダー内のファイルにダイナミック リンクを付加するかどうかを示します。 InstallShield で、サブフォルダー内にあるダイナミック リンクを持つファイルにコンポーネントがどう作成されるかについては、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
自己登録	この列は、InstallShield で、ダイナミック リンクにあるすべてのファイルを自己登録にするかどうかを示します。 ファイルが 64 ビット コンポーネントの一部で、ダイナミック リンクについて Self-Register 列が Yes になっている場合、インストール時にターゲット マシンで 64 ビット自己登録が実行されます。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。
コンポーネントの作成	この列は、InstallShield で、ビルド時にダイナミック ファイルがリリースに追加されるとき、コンポーネントがどう作成されるかを示します。有効オプションは、以下のとおりです: • **ベスト プラクティス** – ダイナミック リンクがあるファイルのコンポーネントを作成するとき、InstallShield はベスト プラクティスを順守します。 • **ディレクトリごと** – 各フォルダーとサブフォルダーについてコンポーネントが 1 つ作成されます。 これらの 2 つのメソッドについては、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
詳細	この列は、ダイナミック リンクからのファイルを選択および除外するときのフィルター基準を示します。

ヒント· *[ファイルのリンク] タブを使用するとき、次のガイドラインを参考にしてください:*

- *新しいダイナミック リンクを追加する場合、[新しいリンク] ボタンをクリックします。[ダイナミック ファイル リンクの設定] ダイアログ ボックスが開きます。*
- *既存のリンクの設定を変更する場合、リンクを選択してから、[変更] ボタンをクリックします。[ダイナミック ファイル リンクの設定] ダイアログ ボックスが開きます。*
- *リンクとそれに関連するダイナミック リンクがあるファイルを削除する場合、リンクを選択してから、[削除] ボタンをクリックします。*

[機能] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

[コンポーネントのプロパティ] ダイアログ ボックス内の [機能] タブでは、コンポーネントと関連付ける機能を選択します。コンポーネントと関連付ける機能 (複数可) の隣にあるチェック ボックスを選択します。

[条件ビルダー] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[条件ビルダー] ダイアログ ボックスは、有効な Windows Installer プロパティと条件式演算子を提供して、コンポーネント、ダイアログ、カスタム アクション、SQL スクリプト、システム検索、その他のインストール要素に対する条件の作成プロセスを簡素化します。

[条件ビルダー] ダイアログ ボックスには、次の設定があります:

テーブル 16 - [条件ビルダー] ダイアログ ボックスの設定

設定	説明
条件	このボックスを使って、コンポーネント、ダイアログ、カスタム アクション、または他のインストール要素の条件を定義できます。このボックスに条件を直接入力するか、[プロパティ] リスト、[演算子] リスト、および [追加] ボタンを使って、条件式をビルドすることができます ***重要-** このダイアログ ボックスの [OK] ボタンをクリックすると、InstallShield によって基本的な条件検証が行われますが、条件ステートメントが予定通りの結果に評価されるかどうか、確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*
Properties	このリストには、[プロパティ マネージャー] ビューで追加されたすべてのプロパティだけでなく、一般的な Windows Installer プロパティが一覧されます。評価するプロパティを選択してから、このリストの隣にある [**追加**] ボタンをクリックします。 リストに評価を行うプロパティが含まれていない場合、適切なプロパティ名を [条件] ボックスに入力できます。
演算子	このリストには、Windows Installer が認識する有効な演算子すべてが含まれています。 条件ステートメントに演算子を含める場合、使用する演算子を選択してから **[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。演算子を追加した場合、その後に適切な値を入力します。

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

[コンテンツ ソース パス] ダイアログ ボックス

[コンテンツ ソース パス] ダイアログ ボックスは、[IIS 構成] ビューで Web サイトの "コンテンツ ソース パス (ローカル または UNC)" 設定にある UNC ボタンをクリックすると開きます。このダイアログ ボックスには、以下の設定が含まれます:

テーブル 17 - [コンテンツ ソース パス] ダイアログ ボックスの設定

設定	説明
Universal Naming Convention コンテンツ ソース パス	IIS Web サイトに含まれるファイルの UNC パスを指定します。例: **¥¥server¥share**

[InstallScript MSI の変換] ダイアログ ボックス

[InstallScript MSI の変換] ダイアログ ボックスを使うと、InstallScript MSI プロジェクトを InstallScript プロジェクトに変換できます。InstallScript プロジェクトは、Windows Installer の制限やオーバーヘッド無しで、ダイアログ ボックスのスキンや柔軟性に優れたエンド ユーザー インターフェイス、強力なスクリプト モデルなどの InstallScript MSI の特徴を活用することができます。

このダイアログ ボックスは、[プロジェクト] メニューの InstallScript プロジェクトに変換 コマンドを選択すると表示されます。

はいをクリックすると、ほとんどのプロジェクト要素が変更無しに変換されます。次の変更が起こります(変換プロセスに際し、状況を知らせる多くのメッセージが表示されます)。

- INSTALLDIR は次の IDE の要素で TARGETDIR に変換されます。
  - コンポーネントのインストール先
  - [ファイルとフォルダー] ビューのパス指定
  - [ショートカット] ビューのターゲット指定

    TARGETDIR の値は、OnFirstUIBefore イベント ハンドラー関数の中で次のデフォルトのコードで設定されるようにスクリプトで設定する必要があります。

    ```
    if ( ALLUSERS ) then
       TARGETDIR = PROGRAMFILES ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
    else
       TARGETDIR = FOLDER_APPDATA ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
    endif;
    ```
- 新しい製品コードが生成されます。
- レジストリデータが関連付けられている各コンポーネントのレジストリセットが作成されます。
- 複数のダイナミック ファイル リンクを持つ各コンポーネントは、それぞれ 1 つのダイナミック ファイル リンクを持つ複数のコンポーネントに分割されます。
- スクリプト ファイルの Setup.rul は OldSetup.rul に変更され、InstallScript ビューに表示されます。このスクリプト ファイルに変更は加えられません。
- サポートされていない次のアイテムには警告が発せられます。
  - [ファイルとフォルダー] ビューのパス指定、または [ショートカット] ビューのインストール先かターゲット指定でサポートされていないフォルダー指定子
  - コンポーネントと関連付けられた Win32 アセンブリ
  - コンポーネントと関連付けられたグローバル アセンブリ キャッシュ
  - ハイフン (-) またはアスタリスク (*) を含むレジストリ値の名前
  - アドバタイズ ショートカット
  - [サポート ファイル] ビューの Disk1 フォルダーの下にあるフォルダー
  - マージ モジュール ([オブジェクト] ビューから変換済みプロジェクトへこれらを再び含めることができます。)
  - データを含む、サポートされていない MSI テーブル

リリースおよびスクリプトは変換されません。最低でも上記のようにスクリプトを修正して TARGETDIR に設定し、すべての INSTALLDIR 参照を TARGETDIR に変更して、Windows Installer API 関数(名前が Msi で始まる関数)への呼び出しを削除する必要があります。また、使用されなくなった文字列エントリの PRODUCT_NAME は変換されません。かわりに IFX_PRODUCT_NAME を使用できます。

OnFirstUIBefore の SHELL_OBJECT_FOLDER を設定する必要はありません。また、デフォルトのスクリプト行である SHELL_OBJECT_FOLDER = @PRODUCT_NAME; は削除する必要があります。

次の Windows Installer フォルダー指定子は InstallScript プロジェクトではサポートされていません。

- ALLUSERSPROFILE
- AdminToolsFolder
- AppDataFolder
- CommonAppDataFolder
- CommonFiles64Folder
- FavoritesFolder
- FontsFolder
- GlobalAssemblyCache
- LocalAppDataFolder
- MyPicturesFolder
- PersonalFolder
- PrimaryVolumePath
- ProgramFiles64Folder
- SendToFolder
- System16Folder
- System64Folder
- TempFolder
- TemplateFolder
- USERPROFILE
- WindowsVolume

さらに次のフォルダー指定子はショートカットのインストール先にはサポートされていません。

- ProgramFilesFolder
- CommonFilesFolder
- WindowsFolder
- SystemFolder

[新しいコンポーネントの作成] ダイアログ ボックス

インストール プロジェクトには、必ずコンポーネントと関連する側面があります。たとえば、ショートカットの作成や、.ini ファイルの変更の際に、コンポーネントが存在しない場合は、InstallShield により新しいコンポーネントの作成を促すプロンプトが表示されます。

ダイアログ オプション

新規コンポーネント名

InstallShield はコンポーネントにデフォルトの名前を付けます。このフィールドに、わかりやすい名前を入力してください。この情報は参照のためだけに使用され、エンドユーザーに表示されることはありません。

このコンポーネントを関連付ける機能を選択する

プロジェクト: *このプロパティは次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

新しいコンポーネントに関連付ける機能をリストから選択します。このフィールドに何も機能が表示されない場合は、[新しい機能] をクリックすると、機能を追加できます。新しいコンポーネントへの機能の関連付けは必須ではありません。ただし、関連付けを行わないと、このコンポーネントはインストールされません。

新しい機能

プロジェクト: *このプロパティは次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*

[新しい機能] をクリックすると、セットアッププロジェクトに機能を追加できます。機能を作成すると、機能名の次にあるボックスをクリックすることにより、新規コンポーネントと関連付けることができます。

今後、このダイアログを表示しない

今後、InstallShield が必要なコンポーネントを自動的に作成する場合、このオプションを選択します。

[新しい機能の作成] ダイアログ ボックス

インストール プロジェクトには、必ず機能と関連する側面があります。たとえばマージ モジュールの追加や ODBC リソースの設定の際に、何の機能も存在しない場合は、InstallShield により機能の作成を促すプロンプトが表示されます。

ダイアログ オプション

新しい機能名

InstallShield によりデフォルトの機能名が作成されます。機能に対して、わかりやすい名前を指定してください。この情報は参照のためだけに使用され、エンドユーザーに表示されることはありません。

今後、このダイアログを表示しない

必要な場合に応じて InstallShield がコンポーネントや機能を自動的に作成する場合は、このボックスをクリックします。

[カスタム エラー処理] ダイアログ ボックス

この [カスタム エラー処理] ダイアログ ボックスで、特定の SQL エラーの処理方法を指定します。定義するエラーがスクリプトのデフォルトのエラー処理よりも優先されます。プロジェクトに含まれるすべてのスクリプトに対して、またはこのスクリプトのみに対してエラーの処理方法を定義することができます。

エラー番号

エラー処理をカスタマイズする SQL エラー番号を入力します。

動作

次のオプションから 1 つを選択すると、デフォルトの動作が上書きされます。

- エラー時に、インストールを中止する
- エラー時に、次のスクリプトへ移動する
- エラー時に、次のステートメントへ移動する

プロジェクト ワイド

プロジェクト内のすべてのスクリプトにカスタム エラー処理を適用するには、[はい] を選択します。このスクリプトのみに適用するには、[いいえ] を選択します。

[カスタム エラー] ダイアログ ボックス

[カスタム エラー] ダイアログ ボックスは、Web サイト、アプリケーション、または IIS 仮想ディレクトリ用にカスタマイズ可能な HTTP エラーをすべてを一覧表示します。カスタム エラーは、URLまたはサーバー上のファイルへのポインターのいずれかです。

タスク ***エラーメッセージを構成するには、以下の手順を実行してください。***

1. **[カスタム エラー]** ダイアログ ボックスで 1 つまたは複数のエラーを選択します。
2. **[編集]** をクリックします。**[エラー マッピングのプロパティ]** ダイアログ ボックスが開きます。
3. メッセージの種類を選択し、必要に応じて、ファイルまたは URL を指定します。
4. **[OK]** をクリックします。

[デフォルト値に設定] ボタンをクリックして、選択したエラーをデフォルト設定に戻すことができます。

[検証設定のカスタマイズ] ダイアログ ボックス

[検証設定のカスタマイズ] ダイアログ ボックスでは、特定の検証スイートでどの内部整合性評価プログラム (ICE) を使用するのかを指定することができます。このダイアログ ボックスは、[オプション] ダイアログ ボックスの [検証] タブで [カスタマイズ] をクリックすると開きます。

特定の検証タイプで実行される ICE のリストをカスタマイズするには、CUB ファイルリストで検証スイートを選択します。実行されるそれぞれの ICE のチェック ボックスを選択するか、実行されないそれぞれの ICE のチェック ボックスをクリアします。

特定の ICE については、Windows Installer ヘルプ ライブラリの「ICE Reference」を参照してください。

InstallShield 検証スイートに追加された個別の InstallShield ICE に関する情報は、「ISICE」を参照してください。

[データ言語] ダイアログ ボックス

エディション: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

[データ言語] ダイアログ ボックスを使用して、各コンポーネントで選択された言語に基づいて特定のコンポーネントを含めたり、その他のコンポーネントを除外したりできます。コンポーネントに指定された言語が、このダイアログ ボックスで選択された言語のどれにも一致しない場合、そのコンポーネントはリリースに含まれません。

[データ言語] ダイアログ ボックスを開くには、[リリース] ビューを開いてリリースを選択します。次に、[ビルド] タブで "データ言語" 設定にある省略記号ボタン (...) をクリックします。

デフォルトでは、リリースは言語に依存しません。つまり、プロジェクトのコンポーネントはすべてリリースに含まれます。

[プロパティの削除] ダイアログ ボックス

セットアップ プロジェクトでプロパティに関連付けられたシステム検索を削除すると、[プロパティの削除] ダイアログ ボックスが表示されます。このダイアログ ボックスは、検索と一緒にそのプロパティを削除するかどうか指定するのが目的です。

メモ: *プロパティを削除すると、セットアップの他の部分に影響を与えることがあります。*

そのプロパティがなくてもプロジェクトが影響を受けないことが確実にわかっている場合は、[はい] を選択します。インストールに影響があるかどうか確かでない場合は、[いいえ] を選択します。

[依存関係] ダイアログ ボックス

[依存関係] ダイアログ ボックスは、選択されたファイルの依存関係を一覧表示します。このダイアログ ボックスにアクセスするには、[ファイルとフォルダー] ビューの **[インストール先コンピューターのファイル]** ペインのファイルを右クリックして、**[ビルド時に依存関係をスキャン]** をクリックします。

ダイアログ ボックスはアセンブリ DLL の結果を表示します。Microsoft Visual Studio からダイアログを起動した場合、ダイアログ ボックスはプロジェクト出力の結果を表示します。Visual Studio の外で InstallShield から起動した場合、[ビルド時に依存関係をスキャン] はプロジェクト出力で無効になります。

メモ: *キーファイルとビルド時の .NET スキャンプロパティが [依存関係とプロパティ] に設定されている場合のみ有効です。探している依存関係、またはその内の一つが見つからなかった場合は赤いアイコンが表示されます。*

ダイアログ ボックス オプション

依存関係

このセクションにはすべての依存関係が一覧で表示され、各ファイルの横にはチェック ボックスがあります。ビルドから依存関係を除外する場合、ファイル横のチェック ボックスをクリアします。[OK] をクリックしてダイアログを閉じます。

メモ: *ビルド時に検出された新たな依存関係 ([依存関係] ダイアログ ボックスを閉じた後で追加されたファイル) はビルドに追加されます。*

[ダイアログのイメージ] ダイアログ ボックス

[ダイアログ イメージ] ダイアログ ボックスを使用して、インストールのダイアログに表示するイメージ (.bmp、.gif、.jpg、または .ibd) を追加します。

全画面イメージ

*外部*ダイアログの全画面背景になるグラフィック ファイルを参照します。外部ダイアログは、インストールの最初または最後に表示されるダイアログで、InstallWelcome および SetupCompleteSuccess(インストール成功時に最後に表示されるダイアログ)などがあります。全画面のイメージ サイズは、499x312 ピクセルです。

バナーイメージ

内部ダイアログの上部で実行されるグラフィックファイルを参照します。内部ダイアログはインストールダイアログの最初と最後の間に表示され、LicenseAgreement ダイアログと CustomSetup ダイアログがあります。バナーのイメージ サイズは 499x58 ピクセルです。

[セットアップのデジタル署名] ダイアログ ボックス

[セットアップのデジタル署名] ダイアログ ボックスは、プロジェクト アシスタントの [インストールのビルド] ページにある [セットアップにデジタル署名する] をクリックすると表示されます。このダイアログ ボックスで、オブジェクト内のコードが発行以来変更または破損されていないことをエンドユーザーに対して保証できます。

テーブル 18 • [セットアップのデジタル署名] ダイアログ ボックスの設定

設定	説明
セットアップにデジタル署名する	インストールにデジタル署名を行う場合、このチェック ボックスを選択します。 このチェック ボックスを選択すると、このダイアログの他の設定が有効にされます。
証明書 URL	完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。

テーブル 18 • [セットアップのデジタル署名] ダイアログ ボックスの設定 (続き.)

設定	説明
デジタル証明書情報	リリースに署名を行うために使用するデジタル証明書を指定するには、この設定の横にある省略記号 (...) ボタンをクリックします。[証明書の選択] ダイアログ ボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。 InstallShield では、パブリック キー証明書 (.cer) ファイルを使って EV トークン パスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。.cer ファイルは通常、USB eToken プロバイダー (たとえば、SafeNet 認証クライアント) と関連付けられた EV 認証クライアント ツールからパブリック キーをエクスポートして作成されます。.cer ファイルが指定された場合、InstallShield は Extended Validation 証明書 (EV) プロパティを構成するための追加オプションを表示します。詳細については、次を参照してください。 • InstallShield における拡張検証 (EV) 証明書情報の構成 • [証明書の選択] ダイアログ ボックス .pfx ファイルを指定するか、テスト ストアから証明書を選択すると、次が表示されます: • **証明書の拇印** – この読み取り専用設定は、証明書の拇印を表示します。 • **発行元** – この読み取り専用設定は、証明書の発行元情報を表示します。 • **有効期限日** – この読み取り専用設定は、証明書の有効期限日を表示します。 証明書ファイル (EV がエクスポートした .cer ファイル) を指定すると、次の設定が表示されます: • **暗号化サービス プロバイダー** – この設定を使って、暗号化サービス プロバイダー (CSP) を指定します。 • **コンテナ名** – この設定を使って、暗号化サービス プロバイダー (CSP) に関連付けられた秘密キーのコンテナ名を指定します。 • **トークン パスワード** – この設定を使って、暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。 ***メモ:*** *証明書ファイル (EV がエクスポートした .cer ファイル) を指定した後に表示される設定を使用する際は、以下の情報に注意してください:* • *暗号化サービス プロバイダー (CSP) 名とコンテナ名の両方は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*

テーブル 18 • [セットアップのデジタル署名] ダイアログ ボックスの設定 (続き.)

設定	説明
パスワード	使用する .pfx にパスワードがある場合、それを入力します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。

ヒント• *[リリース] ビューの [署名] タブでは、ビルド時に、インストールのどの部分に対してデジタル署名を行うかを指定できます。InstallShield は、作業中のプロジェクトの種類に応じて、リリースに含まれる次の任意およびすべてのファイルに署名することができます。*

- *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの Windows Installer パッケージ (.msi ファイル)*
- *マージ モジュール プロジェクトのマージ モジュール パッケージ (.msm ファイル)*
- *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの* Setup.exe *ファイル*
- *InstallScript プロジェクトのメディア ヘッダー ファイル*
- *InstallScript プロジェクトのパッケージ (自己展開型実行可能ファイル)*
- *リリースの任意のファイル (アプリケーション ファイルを含む)*

詳細については、「ビルド時にリリースとそのファイルにデジタル署名を行う」を参照してください。

Windows ロゴ • *インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

[差分] ダイアログ ボックス

[差分] ダイアログは、[プロジェクト] メニューの [ソース管理] メニューにある [差分を表示] コマンドをクリックすると表示されます。このダイアログには、インストール プロジェクト内のすべての Windows Installer テーブルを含むリストがあります。

タスク ***ソース管理プログラムのテーブルの差分を表示するには、以下の手順に従います:***

リストからテーブルを選択し、**[OK]** をクリックします。

[ダイナミック ファイル リンクの設定] ダイアログ ボックス

[ダイナミック ファイル リンクの設定] ダイアログ ボックスでは、ダイナミック リンクのソース フォルダーを指定することができます。続いてそのフォルダー内の全ファイル、または指定のファイルの種類のみを含めるよう選択できます。

[ダイナミック リンクの設定] ダイアログ ボックスは、[コンポーネントのプロパティ] ダイアログ ボックスの [ファイルのリンク] タブおよび [ダイナミック リンクの変更] ダイアログ ボックスで提供されています。

テーブル 19 • [ダイナミック ファイル リンクの設定] ダイアログ ボックスの設定

設定	説明
ソース フォルダー	動的にリンクするフォルダーへのフルパスを入力するか、[参照] ボタンをクリックしてファイルを指定します。ソース フォルダーは、次のいずれかの方法で識別できます: • C:¥Build¥MySourceFolder¥Bitmaps のように、フォルダーへの絶対パスを入力する。 • パス変数を入力する。パス変数の後に円記号を入れてサブフォルダーを指定します。例、<CommonFilesFolder>¥InstallShield¥IScript。 • [参照] ボタンをクリックして、フォルダーに移動する。
サブフォルダーを含める	各サブフォルダー内のファイルにダイナミック リンクを持たせるには、このチェック ボックスを選択します。 InstallShield で、サブフォルダー内にあるダイナミック リンクを持つファイルにコンポーネントがどう作成されるかについては、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
すべてのファイルの自己登録	ダイナミック リンクのすべてのファイルを自己登録する場合、このチェック ボックスを選択します。 ファイルが 64 ビット コンポーネントの一部のとき、このチェック ボックスを選択すると、インストールはターゲット マシンで 64 ビット自己登録を実行します。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

テーブル 19 • [ダイナミック ファイル リンクの設定] ダイアログ ボックスの設定 (続き.)

設定	説明
ベスト プラクティス コンポーネントを作成する	ダイナミック リンクがあるファイルのコンポーネントを作成するとき、ベスト プラクティスを順守することを指定する場合、このチェック ボックスを選択します。コンポーネント作成のベスト プラクティスに従うと、選択と除外のフィルター基準を満たすすべてのファイルに対して、次のタスクがビルド時に実行されます: • ダイナミック リンクがあるフォルダーにある各ポータブル実行可能 (PE) ファイルについてコンポーネントが別々に作成されます。各 PE ファイルは、そのコンポーネントのキー ファイルです。 • ダイナミック リンクのルート レベルにあるすべての非 PE ファイルがリンクを含むコンポーネントに追加されます。 • ダイナミック リンクにサブフォルダーが含まれている場合、サブフォルダー内にあるすべての非 PE ファイルに新しいコンポーネントが作成されます。ダイナミック リンクに複数のサブフォルダーが含まれている場合、各サブフォルダー内のすべての非 PE ファイルにコンポーネントが 1 つずつ別々に作成されます。 ダイナミック リンクがあるファイルのコンポーネントを作成するとき、ベスト プラクティスを順守しないことを指定する場合、このチェック ボックスをクリアします。コンポーネント作成メソッドでは、選択と除外のフィルター基準を満たすすべてのファイルに対して、次のタスクがビルド時に実行されます: • ファイルの種類に関わらず、ダイナミック リンクがあるソース フォルダーのルート レベルにあるすべてのファイルにコンポーネントが 1 つ作成されます。 • ダイナミック リンクに 1 つまたは複数のサブフォルダーが含まれている場合、ファイルの種類に関わらず、各サブフォルダーのすべてのファイルにコンポーネントが 1 つずつ作成されます。サブフォルダーのコンポーネント内にある最初のダイナミック リンクが付いたファイルが、そのコンポーネントのキー ファイルです。 すべての新しいダイナミック リンクについて、このチェック ボックスがデフォルトで選択されています。 詳細については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
すべてのファイルを含める	インストール内のリンクがあるディレクトリの全コンテンツを含める場合、このオプションを選択します。

テーブル 19 • [ダイナミック ファイル リンクの設定] ダイアログ ボックスの設定 (続き.)

設定	説明
次のワイルド カード パターンを基にファイルを選択/除外する	ファイルの種類を選択または除外する場合、このオプションを選択します。先頭にアスタリスク (*) をつけて、拡張子を "選択" または "除外" フィールドに入力します。複数のエントリはカンマで区切ります。 たとえば、すべての画像ファイルがサウンド ファイルと共に 1 つのフォルダーの中にあるとき、画像ファイルのみダイナミック リンクを付加する場合、ダイナミック リンクがあるフォルダーに .bmp ファイルと .ico ファイルのみを含めるように指定することができます。これを行うには、以下の例のように、選択パターンにアスタリスク (*) を使用します: ***.bmp、*.ico** 特定のファイルを選択または除外する場合、選択また除外のパターン ボックスに完全なファイル名を入力します。

[データの編集] ダイアログ ボックス

[データの編集] ダイアログは、プロジェクト アシスタントの [アプリケーション レジストリ] パネルから使用できます。ダイアログにアクセスするには、[インストール先コンピューターのレジストリ データ] ペインのレジストリ値をダブルクリックします。複数文字列値用に、[複数行文字列値の編集] ダイアログ ボックスが表示されます。

ダイアログ オプション

値名

レジストリ値の名前が、[インストール先コンピューターのレジストリ データ] ペインに表示されます。値の名前を変更するには、値を右クリックして [名前の変更] を選択します。

値データ

レジストリ値に保管されている情報が入っています。データを変更または編集するには編集フィールドに入力します。

[IIS Metabase 値の編集] ダイアログ ボックス

[その他の IIS プロパティ] ダイアログボックスで [値の変更] ボタンをクリックすると、[IIS Metabase 値の編集] ダイアログ ボックスが表示されます。[その他の IIS プロパティ] ダイアログ ボックスは、[IIS 構成] ビューで選択された Web サイト、アプリケーション、または仮想ディレクトリの "その他の IIS プロパティ" 設定で省略記号ボタン (...) をクリックすると表示されます。

[IIS Metabase 値の編集] ダイアログ ボックスでは、[IIS 構成] ビューの他のタブで表示されない IIS 設置の値を指定することができます。

詳細プロパティのデフォルト値を変更すると、そのプロパティと、指定した値、および追加の必須情報が ISIISProperty テーブルに追加されます。

テーブル 20 - [IIS Metabase 値の編集] ダイアログ ボックスの設定

設定	説明
プロパティ名	この読み取り専用設定は、その構成を選択したプロパティを表示します。
値	選択した IIS プロパティに使用する値を選択します。

[オプション リストの編集] ダイアログ ボックス

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション- *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[オプション リストの編集] ダイアログ ボックスを使って、インストールのユーザー インターフェイスでコンボ ボックス コントロールまたはリストボックス コントロールに表示するオプションのリストを定義します。詳細については、「ウィザード インターフェイスのコンボ ボックスおよびリスト ボックス コントロールを設定する」を参照してください。

タスク

[オプション リストの編集] ダイアログ ボックスにアクセスするには、以下の手順に従います:

1. **[ユーザー インターフェイス]** の下にあるビュー バー/ビュー リストで、**[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、構成したいコンボ ボックス コントロールまたはリスト ボックス コントロールをを含むウィザード ページまたは 2 番目のウィンドウを展開します。
3. ***"コンテンツ プロパティ"*** 設定で、省略記号ボタン (...) をクリックします。

テーブル 21 - [オプション リストの編集] ダイアログ ボックスの設定

設定	説明
新規作成	新しい行をテーブルに追加するには、このボタンをクリックします。
削除	1 つ以上の選択された行を削除するには、このボタンをクリックします。

テーブル 21 - [オプション リストの編集] ダイアログ ボックスの設定 (続き.)

設定	説明
上	1 つ以上の選択された行の上に移動するするには、このボタンをクリックします。 インストールでは、このダイアログ ボックスでリストされている順番で、リストオプションが表示されます。 連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。連続しない複数の行を選択するには、選択する行の 1 つをクリックしてから CTRL を押しながら追加する各行をクリックします。
下	1 つ以上の選択された行の下に移動するするには、このボタンをクリックします。 インストールでは、このダイアログ ボックスでリストされている順番で、リストオプションが表示されます。 連続する複数の行を選択するには、最初の行をクリックしてから SHIFT を押しながら最後の行をクリックします。連続しない複数の行を選択するには、選択する行の 1 つをクリックしてから CTRL を押しながら追加する各行をクリックします。
テキスト	この列には、コントロールで表示するオプションのリストが含まれています。 この列に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
値	この列の各行には、選択されたウィザード インターフェイス コントロールのリスト オプションに対応する値が含まれています。これが、選択されたコントロールの "プロパティ" 設定で指定されたプロパティに格納されるプロパティ値です。

[レジストリ データの編集] ダイアログ ボックス

[レジストリの編集] ダイアログ ボックスでは、プロジェクト内のレジストリ データを編集できます。このダイアログ ボックスを起動するには、[レジストリ] ビューの値を右クリックをして [変更] を選択します。

テーブル 22 - [レジストリ データの編集] ダイアログ ボックスの設定

設定	説明
値名	この読み取り専用の設定には、レジストリ値の名前が表示されます。 値の名前を変更する場合、[レジストリ データの編集] ダイアログ ボックスを閉じて、名前を変更する値を選択し、F2 キーを押してから新しい名前を入力します。

テーブル 22 • [レジストリ データの編集] ダイアログ ボックスの設定 (続き.)

設定	説明
値データ	ターゲット システムで表示するのと同様に、このレジストリ値に対してデータを入力します。 ***ヒント•*** *レジストリに、PropertyName という Windows Installer プロパティの値を書き込む場合、値データでの表記は [PropertyName] となります。*

[仮想マシンの構成を編集] ダイアログ ボックス

エディション• *InstallShield Premier Edition では、仮想マシンにリリースを配布する機能がサポートされています。*

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

[仮想マシンの構成を編集] ダイアログ ボックスを使って、ビルド時またはオンデマンドで、正しくビルドされたリリースを配布するときに使用する仮想マシン (VM) 設定を構成できます。

ヒント• *[仮想マシンの構成を編集] ダイアログ ボックスにある "VM 構成" 設定を指定すると、InstallShield によって 指定されたデータが* VMConfigurations.xml *という名前のファイルに書き込まれます。この* VMConfigurations.xml *ファイルは、マシン全体に適用するファイルで、一度構成を行うと他の InstallShield プロジェクトでも使用できます。また、他のチームメンバーと共有することも可能です。また、このファイルをビルド マシンで使用することも可能です。詳細については、「リリースの配布用の仮想マシン設定を共有する」を参照してください。*

[リリース] ビューの [イベント] タブで、"構成" 設定の下にあるサブ設定を使って VM の詳細を構成すると、[仮想マシンの構成を編集] ダイアログ ボックスで入力済みの、マシン全体に適用される値が上書きされる場合があります。

タスク ***[仮想マシンの構成を編集] ダイアログ ボックスにアクセスするには、以下の手順に従います:***

1. **[メディア]** の下にあるビュー バー/ビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを選択します。
3. **[イベント]** タブをクリックしてから、**[仮想マシン]** 領域を探します。
4. **"構成"** 設定で、省略記号ボタン (...) をクリックします。

[仮想マシンの構成を編集] ダイアログ ボックスが開きます。このダイアログ ボックスの左側を使って、選択肢として表示する VM 構成のリストを管理できます。新しい構成を追加するには、[新規] ボタンをクリックします。既存の構成を削除するには、それを選択して、[削除] ボタンをクリックします。

[構成] ボックスで特定の構成を選択して、右側でその設定を構成します。設定は、次のメイン カテゴリに分かれています:

- 全般
- サーバー
- 仮想マシン

[全般] の設定

[全般] 領域を使って、選択された VM 構成の名前を指定します。

テーブル 23 • [仮想マシンの構成を編集] ダイアログ ボックスにある全般設定

設定	説明
名前	選択されたリリースを VM に配布するときに使用される VM 構成設定グループの名前を入力します。

サーバーの設定

[サーバー] 領域を使って、リリースを配布する VM をホストするサーバーについての詳細を指定します。

テーブル 24 • [仮想マシンの構成を編集] ダイアログ ボックスにあるサーバー設定

設定	説明
サーバーの種類	リリースを配布する VM をホストするサーバーの種類を指定します。選択可能なオプションは以下のとおりです: • Microsoft Hyper-V Server • VMware ESX or ESXi Server • VMware Workstation
サーバー名	この設定は、サーバーの種類 Microsoft Hyper-V Server および VMware ESX または ESXi Server.で使用できます。 リリースを配布する VM をホストするサーバーの名前を指定します。
Authentication	この設定は、サーバーの種類 Microsoft Hyper-V Server および VMware ESX または ESXi Server.で使用できます。 リリースを配布する VM をホストするサーバーに接続するために使用する認証の種類を選択します。

テーブル 24 - [仮想マシンの構成を編集] ダイアログ ボックスにあるサーバー設定 (続き.)

設定	説明
ユーザー名	この設定は、サーバーの種類 Microsoft Hyper-V Server および VMware ESX または ESXi Server で使用できます。サーバー認証で使用できます。 リリースを配布する VM をホストするサーバーに接続するために使用するアカウントのユーザー名を入力します。
パスワード	この設定は、サーバーの種類 Microsoft Hyper-V Server および VMware ESX または ESXi Server で使用できます。サーバー認証で使用できます。 リリースを配布する VM をホストするサーバーに接続するために使用するアカウントのパスワードを入力します。

仮想マシンの設定

[仮想マシン] 領域を使って、リリースを配布する VM をホストするサーバーについての詳細を指定します。

テーブル 25 - [仮想マシンの構成を編集] ダイアログ ボックスにある仮想マシン設定

設定	説明
マシン名	リリースを配布する VM の名前を指定します。
スナップショット	オプションで、リリースの配布に使用するスナップショットの名前を指定することもできます。個別のリリース用のスナップショットをオーバーライドすることができます。 VM 構成にスナップショットが指定されていない場合、InstallShield は特定のスナップショットには戻りません。InstallShield は特定のスナップショットに戻さずに VM の電源をオンにして、VM にインストールをコピーします。
Authentication	オプションで、リリースを配布する VM をホストするサーバーに接続するために使用する認証の種類を選択します。個別のリリース用の認証をオーバーライドすることができます。
ユーザー名	この設定は、サーバー認証に適用します。 オプションで、リリースを配布する VM に接続するために使用するアカウントのユーザー名を入力します。個別のリリース用のユーザー名をオーバーライドすることができます。
パスワード	この設定は、サーバー認証に適用します。 オプションで、リリースを配布する VM に接続するために使用するアカウントのパスワードを入力します。個別のリリース用のパスワードをオーバーライドすることができます。

テーブル 25 • [仮想マシンの構成を編集] ダイアログ ボックスにある仮想マシン設定 (続き.)

設定	説明
インストール先パス	オプションで、ビルドされたリリースを InstallShield がステージングする VM 上のパスを指定できます。パスの最後のサブフォルダーとして、InstallShield が VM 上に作成するパスを使用できます。その他のパスは既存していなくてはなりません。 個別のリリース用のインストール先パスをオーバーライドすることができます。

[エラー マッピングのプロパティ] ダイアログ ボックス

[エラー マッピングのプロパティ] ダイアログ ボックスは、IIS エラーコードとそのデフォルト プロパティを表示します。[メッセージの種類] リストを利用して メッセージに URL またはポインターにサーバー上のファイルを設定することができます。メッセージに URL を設定する場合、URLを完全な形で指定します。エラーメッセージにファイルポインターを設定する場合、エラーメッセージとして使用するファイルを参照、または、そのファイルの完全パスを入力します。これには、システムには既に存在していて、作成中のプロジェクトには存在しないファイルを含みます。

[言語の除外] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *マージ モジュール*
- *MSM データベース*

[一般情報] ビューで除外マージ モジュールの "言語" 設定にある省略記号ボタン (...) をクリックすると、[言語の除外] ダイアログ ボックスが開きます。このダイアログ ボックスを使って、作成中のマージ モジュールと互換性を持たないマージ モジュール用の言語要件を指定できます。

除外マージ モジュールの言語要件を指定するには、以下のいずれかを行います:

- 特定言語のマージ モジュールを除外するには、言語リストからその言語を選択してから、**[一致する]** オプションを選択します。
- 特定言語以外のマージ モジュールを除外するには、言語リストからその言語を選択してから、**[一致しない]** オプションを選択します。
- 言語によるマージ モジュールの除外を行わない場合、言語リストから [言語非依存] を選択します。

[テーブルのエクスポート] ダイアログ ボックス

[テーブルのエクスポート] ダイアログ ボックスを使って、[ダイレクト エディター] を通して 1 つまたは複数のテーブルを .idt ファイルとしてエクスポートできます。

テーブル 26 • [テーブルのエクスポート] ダイアログ ボックスの設定

設定	説明
出力先ディレクトリ	テーブルのエクスポート先ディレクトリを指定します。[参照] をクリックしてディレクトリに移動することもできます。
Tables	このボックスは、プロジェクトに含まれるすべてのテーブルのチェック ボックスを一覧表示します。エクスポートするテーブルのチェック ボックスを選択します。
すべて選択	すべてのテーブルを選択してエクスポートします。
すべてをクリア	すべてのテーブルを選択解除します。
選択の切り替え	選択したテーブルをすべて選択解除すると同時に、選択解除されていたすべてのテーブルを選択します。

[機能条件ビルダー] ダイアログ ボックス

[機能条件ビルダー] ダイアログ ボックスには、有効な Windows Installer プロパティと条件式の演算子が提供されていて、機能条件を簡単に作成することができます。

1 つの機能に対して、必要な条件を無制限に作成できます。条件の作成に関する詳細については、「機能の条件を設定する」を参照してください。

[機能条件ビルダー] ダイアログ ボックスには、次の設定があります。

テーブル 27 - [機能条件ビルダー] ダイアログ ボックスの設定

設定	説明
条件	このボックスには、次の列を含むグリッドが表示されます: • **レベル** – この列には、条件が True 評価された場合に機能に使用するインストール レベルを割り当てます。 • **条件** – この列に条件を定義します。この列に条件を直接入力するか、[プロパティ] リスト、[演算子] リスト、および [追加] ボタンを使って、条件式をビルドすることができます このボックスに新しい条件を追加するには、[新しい条件] ボタンをクリックしてから、[レベル] と [条件] 列に適切な情報を入力します。 このボックスの条件を削除するには、条件を選択してから、[条件の削除] ボタンをクリックします。 ***重要** - このダイアログ ボックスの [OK] ボタンをクリックすると、InstallShield によって基本的な条件検証が行われますが、条件ステートメントが予定通りの結果に評価されるかどうか、確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。*
Properties	このリストには、[プロパティ マネージャー] ビューで追加されたすべてのプロパティだけでなく、一般的な Windows Installer プロパティが一覧されます。評価するプロパティを選択してから、このリストの隣にある [**追加**] ボタンをクリックします。 リストに評価を行うプロパティが含まれていない場合、適切なプロパティ名を [条件] 列に入力できます。
演算子	このリストには、Windows Installer が認識する有効な演算子すべてが含まれています。 条件ステートメントに演算子を含める場合、使用する演算子を選択してから **[追加]** ボタンをクリックします。InstallShield によって、条件ステートメントに演算子が追加されます。

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

[ファイルの詳細] ダイアログ ボックス

システム検索ウィザード の [詳細] ボタンをクリックして、このダイアログにアクセスします。このボタンは、検索方法を定義して検索するファイルを指定するとアクティブになります。このダイアログでは次の詳細を指定することにより、検索を拡張できます。

テーブル 28 · [デバイス ファイル] ダイアログ ボックスの設定

設定	説明
最小バージョン	ファイルがターゲット システムに存在し、バージョンが入力した値と等しいか、または高ければ検索は成功です。
最大バージョン	ファイルがターゲット システムに存在し、バージョンが入力した値と等しいか、または低ければ検索は成功です。
最小の日付	チェック ボックスを選択して、最小日で検索します。ファイルがターゲット システムに存在し、日付が入力した値と等しいか、または大きければ検索は成功です。
最大の日付	チェック ボックスを選択して、最大日で検索します。ファイルがターゲット システムに存在し、日付が入力した値と等しいか、または小さければ、検索は成功です。
最小サイズ	ファイルがターゲット システムに存在し、サイズが (バイトで)指定した値と等しいか、または大きければ、検索は成功です。
最大サイズ	ファイルがターゲット システムに存在し、サイズが (バイトで)指定した値と等しいか、または小さければ、検索は成功です。
Languages	[参照 (…)] ボタンをクリックして [言語] ダイアログを表示します。検索の条件として複数言語を選択することができます。リストされた言語から最低 1 つマッチすれば、検索は成功です。

***メモ·** 編集フィールドに入力する情報はオプションです。フィールドは空白で残すこともできます。*

[ファイルのプロパティ] ダイアログ ボックス

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *MSM データベース*
- *トランスフォーム*

[プロパティ] ダイアログ ボックスを使って、ファイルがターゲット システムにインストールされるときの、様々なプロパティをオーバーライドできます。

タスク ***[ファイルのプロパティ] ダイアログ ボックスを開くには、以下の手順を実行してください。***

1. **[アプリケーション データ]** の下にあるビュー バー/ビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. ファイルを右クリックして、**[プロパティ]** を選択します。

重要: *ダイナミック リンク ファイルのファイルのプロパティは設定できません。*

テーブル 29 - [プロパティ] ダイアログ ボックスの設定

設定	説明
場所	この読み取り専用フィールドは、このファイルが含まれるコンポーネントの保存先に従って、ターゲット システムにこのファイルをインストールするディレクトリを表示します。 保存場所を変更する場合、このファイルが含まれるコンポーネントのインストール先を変更します。
フォントタイトル	フォントをインストールしている場合、ここでフォント タイトルを **FontTitle (FontType)** の形式で指定できます (例、**Roman (All res)**)。フォントがシステムに登録されている場合、InstallShield がフォント名を提供します。 .ttf または .ttc フォントに対してフォント タイトルを指定しないでください。Windows Installer では、埋め込みフォント名を読み取って登録を行います。また、InstallShield では、.ttf または .ttc ファイル用にこのフィールドに **[ファイルから読み込むタイトル]** とマークします。 さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「ICE07」を参照してください。
自己登録	インストール中にファイルを自己登録する場合、このチェック ボックスを選択します。"自己登録" 設定は、.dll、.ocx、.exe、.tlb および .olb ファイルに対して有効です。 ファイルが 64 ビット コンポーネントの一部の場合、インストールはターゲット マシンで 64 ビット自己登録を実行します。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。 ***メモ:*** *自己登録を使わずに、自己登録ファイルから COM 情報を抽出することをお勧めします。詳細については、「COM サーバーの登録」を参照してください。*

テーブル 29 • [プロパティ] ダイアログ ボックスの設定 (続き.)

設定	説明
システム属性のオーバーライド	このファイルを、開発システムでこのファイルに現在設定されている同じシステムのプロパティでインストールするには、このチェック ボックスをクリアします。 ファイルの 1 つ以上のプロパティをオーバーライドするには、**[システム属性のオーバーライド]** チェック ボックスを選択してから、以下のチェック ボックスから 1 つ以上を選択します。 • **読み取り専用**—Windows Installer がファイルをインストールするときに、これを読み取り専用にする場合、このチェック ボックスを選択します。 • **隠しファイル** — Windows Installer がファイルをインストールするときに、これを隠しファイルにする場合、このチェック ボックスを選択します。 • **ファイル ハッシュの使用** — このオプションは、バージョン指定されていないファイルにのみ使用できます。Windows Installer はファイルのハッシュを使用してバージョン指定されていないファイルの不要なコピーを避けたり、検出したりできます。Windows Installer が既存ファイルをアップグレードするかどうか判断するときに、インストールに含まれるファイル ハッシュとターゲット システム上の対応するファイルのファイル ハッシュとを比較する場合、このチェック ボックスを選択します。 • **システム** — Windows Installer がファイルをシステム ファイルとしてインストールする場合は、このチェック ボックスを選択します。 • **重要** — このファイルがコンポーネントの操作に必須であることを示す場合、このチェック ボックスを選択します。何らかの理由によって重要ファイルがインストールされないと、コンポーネントもインストールされません。必須ファイルがインストールされない場合、通常表示される [中止]、[再試行]、および [無視] ボタンではなく、[再試行] および [キャンセル] オプションを含むエラーメッセージがエンドユーザーに対して表示されます。(これによって、エンド ユーザーはそのファイルをインストールしないでインストール処理を完了できます。) ***メモ:*** *[リリース] ビューでリリースの [ビルド] タブにある "ファイルのハッシュ値の生成" 設定に [いいえ] を選択した場合、個別のファイルの [ファイル ハッシュの使用] チェック ボックスが選択されているかどうかに関わらず、どのファイルにもファイル ハッシュ エントリは作成されません。*
システムのサイズを変更する	"サイズ" 設定は、**"システム サイズのオーバーライド"** 設定がクリアされたとき、ファイルのサイズを表示します。 Windows Installer がインストールの必要ディスク容量を計算するときに、選択されたファイルの実際のサイズを無視して、指定された値をそのファイルのサイズとして判断するように設定するには、このチェック ボックスを選択してから "サイズ" 設定で適切な値をバイト単位で入力します。

テーブル 29 • [プロパティ] ダイアログ ボックスの設定 (続き.)

設定	説明
常に上書き	Windows Installer が選択されたファイルの実際のバージョン番号を無視して、その代わりにターゲット システムにそのファイルが既存する場合、常に上書きを試みるように設定するには、このチェック ボックスを選択します。実行時、このファイルがターゲット システム上にあるファイルと同じ名前および同じターゲット場所を持つ場合、Windows Installer がターゲット システムのファイルを現在のインストールに含まれているバージョンでアップデートするか、ファイルをそのままに残すかを決定するとき、インストールに含まれているファイルのバージョンを 65535.0.0.0 と判断します。 Windows Installer が既存ファイルを上書きするかどうかを判断する方法についての詳細は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。 ファイルの最大バージョン番号は、65535.65535.65535.65535 です。
システムのバージョンを変更する	ファイルのバージョンが指定されている場合、**"システムバージョンの上書き"** 設定がクリアされたときに、"バージョン" 設定にはファイルのバージョンが表示されます。 Windows Installer が選択されたファイルの実際のバージョン番号を無視して、その代わりに指定された番号をバージョン番号と判断するように設定するには、このチェック ボックスを選択してから、"バージョン" 設定に適切なバージョン番号を入力します。実行時、このファイルがターゲット システム上にあるファイルと同じ名前および同じターゲット場所を持つ場合、Windows Installer がターゲット システムのファイルを現在のインストールに含まれているバージョンでアップデートするか、ファイルをそのままに残すかを決定するとき、指定されたバージョン番号を使用します。 たとえば、プロジェクトに含まれるファイルのバージョンが 2.0.0.0 で、上書きバージョンを 3.0.0.0 と入力した場合、Windows Installer は、ターゲット システム上のファイル バージョンが 3.0.0.1 以降の場合は置換しますが、3.0.0.0 以前の場合は置換しません。 Windows Installer が既存ファイルを上書きするかどうかを判断する方法についての詳細は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。 ファイルの最大バージョン番号は、65535.65535.65535.65535 です。

テーブル 29 • [プロパティ] ダイアログ ボックスの設定 (続き.)

設定	説明
システムの言語を変更する	Windows Installer が選択されたファイルの言語を無視して、その代わりに指定された言語を使用するように設定する場合は、このチェック ボックスを選択してから [言語] ボックスの言語識別コードに 10 進数値を入力します。実行時、このファイルがターゲット システム上にあるファイルと同じ名前および同じターゲット場所を持つ場合、Windows Installer がファイルを現在のインストールに含まれているバージョンでアップデートするか、ファイルをそのままに残すかを決定するとき、インストールに含まれているファイルの言語が指定された言語であると判断します。 Windows Installer が既存ファイルを上書きするかどうかを判断する方法についての詳細は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。 フォントは埋め込み言語 ID リソースを持たないため、Font ファイルは言語 ID と共に作成しません。フォントファイルには、このエントリを空白で残します。
アクセス許可	ファイルのアクセス許可を設定するには、このボタンをクリックします。

[ファイルのプロパティ] ダイアログ ボックス (InstallScript インストール ロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

このダイアログ ボックスは、ファイルがターゲット システムにインストールされるときにそのファイル用に設定されるプロパティを表示します。

タスク

[ファイルのプロパティ] ダイアログ ボックスを開くには、以下の手順を実行してください。

ファイルを右クリックして、**[プロパティ]** を選択します。このオプションは、"リンク タイプ" プロパティに **[ダイナミック]** が設定されているコンポーネント内のファイルには無効です。

InstallScript プロジェクトでは、[ファイルのプロパティ] ダイアログ ボックスは選択されたファイルについての情報を表示します。このダイアログ ボックス内の情報は、ファイル システムから直接取得され、次のコントロール以外はダイアログ ボックスは読み取り専用です。

フォントファイルの登録

このボックスがチェックされると、グローバル フォント登録を有効にしている場合、ファイルはターゲット マシンに登録されるよう内部的にマークされます。このボックスがチェックされていない場合、ファイルはターゲット マシーンに登録されません。

フォントタイトル

.fon ファイルの場合、このテキストはインストールがフォントに登録するフォントのタイトルを指定します。デフォルトでは、InstallShield ユーザー インターフェイスはソースシステムのレジストリからフォントタイトルを読み込み、それを編集ボックスに表示します。TrueType ファイル (.ttf または .ttc ファイル)の場合、デフォルトでは、編集ボックスにはテキストが表示されず、インストールは実行時にそのファイルからフォントタイトルを読み込みます。これら3つにすべてのファイルにおいて、編集ボックスに非デフォルトのテキストを入力した場合、インストールはそのテキストをフォントタイトルとして登録します。

ファイル 削除の [プロパティ] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ファイル 削除の [プロパティ] ダイアログ ボックスには、ターゲット システムから削除するように選択されたファイルまたはフォルダーの情報が表示されます。このファイルとフォルダーの削除機能は、アプリケーションによって作成されるファイルの削除など、インストールが追跡を行わない処理に使用すると便利です。

ファイル削除の [プロパティ] ダイアログ ボックスには、[ファイルとフォルダー] ビュー内から、あるいは [セットアップのデザインデザイン] ビュー (インストール プロジェクト) または [コンポーネント] ビュー内からアクセスが可能です。

タスク ***[ファイルとフォルダー] ビュー内からファイル削除の [プロパティ] ダイアログ ボックスにアクセスするには、以下の手順に従います:***

1. **[アプリケーション データ]** の下にあるビュー バー/ビュー リストで、**[ファイルとフォルダー]** をクリックします。
2. **[インストール先コンピューターのフォルダー]** ペインで、構成したいファイル削除項目を含むフォルダーをクリックします。
3. **[インストール先コンピューターのファイル]** ペインで、構成するファイル削除項目を右クリックしてから、**[プロパティ]** をクリックします。

   InstallShield では、削除するように構成されたファイルまたはフォルダーを識別するために赤い X 印のアイコンが使用されます。

タスク

ファイル削除の [プロパティ] ダイアログ ボックスには、[セットアップのデザイン] ビュー (インストール プロジェクト) または [コンポーネント] ビュー内からアクセスするには、以下の手順に従います:

1. **[編成]** の下にあるビュー リストで、**[セットアップのデザイン]** (基本の MSI、InstallScript MSI、または MSI データベース プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. エクスプローラーで、構成するファイル削除の項目を含むコンポーネントを展開してから、その下の**ファイル**ノードをクリックします。右側の **[ファイル]** ペインに、選択されたコンポーネントと関連付けられたファイルと削除項目が表示されます。
3. **[ファイル]** ペインで、構成するファイル削除の項目を右クリックしてから、**[プロパティ]** をクリックします。

   InstallShield では、削除するように構成されたファイルまたはフォルダーを識別するために赤い X 印のアイコンが使用されます。

ファイルとフォルダーの削除項目をプロジェクトに追加する方法については、「ターゲット システムからファイルとフォルダーを削除する」を参照してください。

テーブル 30 ・ [プロパティ] ダイアログ ボックスの設定

設定	説明
場所	この設定は、削除するように構成されている選択済みの項目の場所を示します。
フォルダーが空の場合は削除する	選択された削除項目が、ターゲット システム上で空白の場合に削除するフォルダーである場合、このオプションを選択します。
ファイルをフォルダーから削除する	選択された削除項目が "場所設定"識別されるフォルダー内にあるファイルで、これをターゲット システムから削除する場合は、このオプションを選択します。
ファイル名	この設定は、**[ファイルをフォルダーから削除する]** オプションを選択すると使用できます。 変更するファイルの名前を指定します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 30 • [プロパティ] ダイアログ ボックスの設定 (続き.)

設定	説明
削除のスケジューリング	ターゲット システムから選択したファイルまたはフォルダーを削除するタイミングを指定します。ファイルまたはフォルダーに関連付けられたコンポーネントが、次の 1 つの方法で処理されるときに削除処理が発生します: • **コンポーネントのインストール** – 選択された項目のコンポーネントがインストールされるとき、フォルダー削除項目の場合は、そのフォルダーが空の場合に、選択された項目が削除されます。 • **コンポーネントのアンインストール** – 選択された項目のコンポーネントがアンインストールされるときに、フォルダー削除項目の場合は、そのフォルダーが空の場合に、選択された項目が削除されます。 • **コンポーネントのインストールおよびアンインストール** – 選択された項目のコンポーネントがインストール、またはアンインストールされるときに、フォルダー削除項目の場合は、そのフォルダーが空の場合に、選択された項目が削除されます。

[検索/置換] ダイアログ ボックス

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[検索/置換] ダイアログ ボックスを使って、実行時に SQL スクリプト ファイル内の文字列を検索して、それらを異なる文字列で置換することができます。[検索/置換] ダイアログ ボックスは、[SQL スクリプト] ビューで選択された SQL スクリプトの [テキスト置換] タブにある [追加] ボタンをクリックすると開きます。

基本の MSI、DIM、および InstallScript MSI プロジェクトでは、文字列を Windows Installer プロパティで置換することができます。その場合、実行時に Windows Installer が SQL スクリプト ファイルにプロパティの値を書き込みます。

InstallScript プロジェクトでは、テキスト置換を使って文字列変数を適切な値で置換できます。

***注意 •** 置き換えるテキストは、スクリプト構文エラーが起きないように、一意でなければなりません。*

[検索/置換] ダイアログ ボックスでは、以下の設定を使用できます:

テーブル 31 • [検索/置換] ダイアログ ボックスの設定

設定	説明
検索する文字列	実行時に SQL スクリプトで置換する文字列を入力します。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI プロジェクトの場合、Windows Installer パブリック プロパティを使って、置換文字列を指定できます。詳細については、「Windows Installer プロパティを使って SQL スクリプトの文字列を動的に置換する」を参照してください。* *InstallScript プロジェクトの場合、InstallScript テキスト置換を使って、置換文字列を指定できます。詳細については、「InstallScript テキスト置換を使って SQL スクリプトの文字列を動的に置換する」を参照してください。* *Windows Installer プロパティまたは InstallScript テキスト置換を指定すると、製品の SQL スクリプトが変更された場合に、エンド ユーザーがダイアログに入力したデータ、または実行時に決定するその他の構成情報を使用することができます。*
置換後の文字列	実行時に SQL スクリプトで検出された既存の文字列と置換する新しい文字列を入力します。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI プロジェクトの場合、Windows Installer パブリック プロパティを使って、置換文字列を指定できます。詳細については、「Windows Installer プロパティを使って SQL スクリプトの文字列を動的に置換する」を参照してください。* *InstallScript プロジェクトの場合、InstallScript テキスト置換を使って、置換文字列を指定できます。詳細については、「InstallScript テキスト置換を使って SQL スクリプトの文字列を動的に置換する」を参照してください。* *Windows Installer プロパティまたは InstallScript テキスト置換を指定すると、製品の SQL スクリプトが変更された場合に、エンド ユーザーがダイアログに入力したデータ、または実行時に決定するその他の構成情報を使用することができます。*
大文字と小文字を区別する	このチェック ボックスを使って、"検索文字列" 設定で使用している大文字と子文字の区別をそのまま利用して文字列を検索するように制限するかどうかを指定します。 たとえば、このチェック ボックスを選択した場合で、"検索文字列" 設定に **install** を入力すると、インストールはその文字列のすべて小文字を使ったインスタンスのみを検索します。このチェック ボックスをクリアした場合、インストールは *install* だけでなく *INSTALL*、*Install*、その他の大文字と小文字を組み合わせたインスタンスも検索します。

テーブル 31 • [検索/置換] ダイアログ ボックスの設定 (続き.)

設定	説明
単語単位で探す	このチェック ボックスを使って、"検索する文字列" 設定に入力した値が単語単位で一致するインスタンスのみを検索するかどうかを指定します。 たとえば、このチェック ボックスを選択して、"検索文字列" 設定に **install** を入力すると、インストールは *install* のインスタンスのみを検索し、同じ単語の別の形式、たとえば *installs*、*installation*、または *uninstall* は検索しません。 その他の例として、SQL スクリプトで以下のような行が考えられます: PASSWORD = N'%PASSWORD%' このチェック ボックスを選択して、"検索する文字列" 設定に **%PASSWORD%** と入力してから "置換後の文字列" 設定に **1234** と入力すると、インストールが文字列を置換します。結果の SQL スクリプトでは、次が表示されます: PASSWORD = N'1234'
大文字と小文字の区別を保持する	このチェック ボックスを使って、"検索文字列" 設定で使用している大文字と子文字の区別を置換文字列にも使用するかどうかを指定します。 たとえば、このボックスを選択して "検索文字列" 設定に **databasedir** および "置換後の文字列" 設定に **mydatabasedir** と入力すると、インストールは *Databasedir* を *Mydatabasedir* で置換します。また、*databasedir* を *mydatabasedir* で置換します。
一度だけ置換	このチェック ボックスを使って、インストールが、最初に見つかった検索文字列のみを置換するかどうかを指定します。 このチェック ボックスをクリアした場合、インストールは検索文字列のすべてのインスタンスを置換します。

[プロジェクト内の文字列 ID を検索する] ダイアログ ボックス

[プロジェクト内の文字列 ID を検索する] ダイアログ ボックスを使うと、特定の文字列エントリをプロジェクト全体から検索できます。検索する文字列エントリをリストから選択して [OK] をクリックします。検索の結果は、スクリーンの下の出力ウィンドウに表示されます。

[フォルダーのプロパティ] ダイアログ ボックス

フォルダーの [プロパティ] ダイアログ ボックスでは、ファイルがターゲット システムにインストールされるときにフォルダー用に設定されるプロパティを決定します。このダイアログ ボックスにアクセスするには、[ファイルとフォルダー] ビルド内にあるフォルダーを右クリックして、[プロパティ] を選択します。

[プロパティ] ダイアログ ボックスは、2 つのタブに分かれています:

- 全般
- 共有

[全般] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

フォルダーの [プロパティ] ダイアログ ボックスの [全般] タブでは、フォルダーに関する基本的なプロパティを構成できます。

テーブル 32 • [全般] タブの設定

設定	説明
ディレクトリ識別子	この設定では、選択したフォルダーの現在のディレクトリ ID が識別されます。
ソースの場所	この設定は、非圧縮リリースのビルド時に、ディスク イメージ フォルダーにコピーされるファイルの元の保存場所を識別します。
ターゲット	この設定は、ターゲット システム上で、フォルダーがインストールされる場所を識別します。

[アクセス許可] ボタンを使って、フォルダーのアクセス許可を構成できます。詳細については、「ファイルとディレクトリの [アクセス許可] ダイアログ ボックス」を参照してください。

[共有] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

フォルダーの [プロパティ] ダイアログ ボックスの [全般] タブでは、フォルダーに関する基本的なプロパティを構成できます。

テーブル 33 • [全般] タブの設定

設定	説明
このフォルダーをネットワークで共有する	このフォルダーをネットワークで共有する場合、このチェック ボックスを選択します。
共有名	このフォルダーの共有を有効化するには、共有フォルダーに接続するためにユーザーが必要な名前を指定します。
同時ユーザーの数を次の数に制限する:	このフォルダーの共有を有効化するには、同時に共有フォルダーに接続可能なユーザーの最大数を指定します。 ***メモ:*** *Windows では、同時ユーザーの最大数は制限されています。これらの制限は、Windows のバージョンとエディションによって異なります。インストールでは、この Windows によって課されている制限をオーバーライドすることはできません。したがって、ターゲット システムでサポートされている数を超える人数を指定した場合、インストール時に、ターゲット システムのオペレーティング システムで許可されている最大数に制限が設定されます。* *Windows の特定のバージョンおよびエディションの制限を判別するrには、その Windows のバージョンおよびエディションで次のコマンドライン ステートメントを使用します:* Net Config Server *[コマンド プロンプト] ウィンドウで表示された最大ユーザー数の値が、オペレーティング システムでサポートされている最大数です。* *Windows の使用許諾契の「デバイスの接続」セクションでも最大数が明示されています。*
ネットワーク ユーザーが自分のファイルを変更するのを許可する	このフォルダーの共有を有効化して、ネットワーク ユーザーにこのフォルダー内のファイルの変更を許可する場合、このチェック ボックスを選択します。
コンポーネント	このネットワーク共有を含めるこのフォルダー内のコンポーネントを選択します。

[関数シグネチャ] ダイアログ ボックス

[関数シグネチャ] ダイアログ ボックスで、標準 .dll ファイル中のエントリ ポイント関数のパラメーターを指定します。

タスク ***[関数シグネチャ] ダイアログ ボックスにアクセスするには、以下の手順に従います:***

1. **[動作とロジック]** の下のビュー バー/ビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーで、パラメーターを指定するカスタム アクションを選択します。
3. "関数シグネチャ" プロパティで、省略記号 (...) ボタンをクリックします。

ダイアログ ボックス オプション

名前

呼び出す関数の名前を指定します。.dll ファイルは、各カスタム アクションに対して シングル エントリ ポイントを持つことができます。

引数

引数のリストを関数に渡す順序でビルドします。

まず、[引数] リストの最後の行をクリックして新規引数を追加します。次に、各引数の Type、Source および Value 列を完成させます。

種類

このリストで、パラメーターのデータ型を選択します。引数には 2 つの特殊な種類があります。これらを以下の方法で識別します。

タスク ***ポインターを NULL に初期化するには次の操作を実行します。***

1. [タイプ] リストから **POINTER** を選択します。
2. **[ソース]** リストから、**[定数]** を選択します。
3. **[Value]** リストから **[NULL]** を選択します。**NULL** の代わりに 0 (ゼロ) は入力しないでください。これは、「0 番をポイントするポインター」として解釈されます。

タスク ***ハンドルに .msi データベースまたは Windows Installer ウィンドウを設定するには、次の手順を実行します。***

1. **[タイプ]** リストから **[HANDLE]** を選択します。
2. **[ソース]** リストから、**[定数]** を選択します。
3. **[値]** リストで、ハンドルに MSI データベースを設定するか、MSI ウィンドウを設定するかによって、**MsiHandle** または **MsiWindowHandle** を選択します。

ソース

データを関数に渡す方法を示します。以下のオプションは次のリストで利用可能です。

テーブル 34 - [関数シグネチャ] ダイアログ ボックスのオプション

オプション	説明
定数	渡す値はリテラルとしてインストール プロジェクト中に保存されます。引数のソースに **[定数]** を選択した後、その定義した値を [値] 列に入力します。 この定数には、名前や ID は必要ありません。文字列リテラルは引用符で囲まないでください。
in プロパティ	ソースで **[in プロパティ]** を指定して、値が関数に渡される Windows Installer プロパティの名前を指定します。このタイプの引数は ByVal パラメーターに適しています。
inout プロパティ	このソースの種類は ByRef パラメーターに似ています。ソースで **[inout プロパティ]** を指定して、値が関数に渡され、関数で変更することが可能な Windows Installer プロパティの名前を指定します。
out プロパティ	**[out プロパティ]** は、関数で設定することができる値のプレースホルダーとして機能します。関数に値は渡されませんが、関数が値を変更できるプロパティの名前は必要です。

引数のソースのプロパティを選択するたびに、[値] 列にプロパティの名前を指定する必要があります。

値

値は以下の 2 種類のうちの 1 つです。

- 数字または文字列リテラル 引数のソースが定数の場合
- Windows Installer プロパティ 引数のソースがプロパティの場合

メモ- *[タイプ] リストを HANDLE に設定し、[ソース] リストを [定数] に設定すると、Value 列に 2 つのオプション（MsiHandle MsiWindowHandle）が含まれます。これらの定数を使用して、.msi データベースまたは Windows Installer ウィンドウへのハンドルを取得できます。*

ソースがプロパティの場合、[値] リストでプロパティ マネージャー中のすべてのプロパティの名前が表示されます。既存のプロパティを選択するか、または新しい名前を入力できます。後者の場合、プロパティ マネージャーに新規プロパティが追加されます。

プロパティは単なる値のコンテナであることに留意してください。パラメーターがその値を[in プロパティ] または [inout プロパティ] に保存する場合は、プロパティマネージャーでそのプロパティの値を必ず指定してください。

(コンテキスト メニュー)

コンテキスト メニューで、引数を使用するためのオプションが表示されます。コンテキスト メニューにアクセスするには、[引数] グリッドで引数を右クリックします。コンテキスト メニュー オプションを以下に説明します。

テーブル 35 - コンテキストメニューのオプション

オプション	説明
追加	[追加] をクリックして、[引数] グリッドに引数を追加します。
削除	引数を削除するには、[削除] をクリックします。
上へ移動	引数は、関数がそれを受け取るのとまったく同じ順序でなければなりません。引数をリストの上に移動するには、[上へ移動] をクリックします。
下へ移動	引数をリストの下に移動するには、[下へ移動] をクリックします。

戻り値の型

関数の戻り値のデータ型を選択します。戻り値を確認する必要がない場合は、void を受け入れることができます。

戻り値プロパティ

値が関数の戻り値に設定されるプロパティ名を選択します。インストールの他の場所で戻り値をチェックする場合、戻りプロパティの値を初期化することも考えられます。

メモ- アクションが即時実行に構成されていない限り、関数の戻り値にプロパティを設定することはできません。したがって、アクションが遅延、ロールバック、またはコミット実行としてスケジュールされている場合、実行時に "戻り値プロパティ" 設定は無視されます。

[一般オプション - 詳細] ダイアログ ボックス

リリース ウィザードの [一般オプション] パネル で [詳細] をクリックすると、[一般オプション - 詳細] ダイアログ ボックスが開きます。このダイアログ ボックスを使うと、ビルドしたファイルのカスタム保存場所の指定(インストール プロジェクトの場合のみ)、ディスクイメージフォルダー内に確保してあるスペースの指定、およびインストールが MD5 をチェックを行なうかどうか、また(オブジェクトプロジェクトの場合のみ)オブジェクトのビルド中にオブジェクトプロジェクトに含まれるサブジェクトの InstallShield による処理方法を指定できます。

パネルのオプション

カスタムフォルダーにメディアをビルド

選択を解除した場合、リリースの Disk Images、Log Files、および Report Files フォルダーは、次のデフォルトの場所に作成されます:

<プロジェクト フォルダー>¥メディア¥<リリース名>

オンにした場合、別の場所をコンボ ボックスに入力するか、パス変数に対して定義されたリスト項目の 1 つを選択して完成、または参照ボタンをクリックして Web ファイルを選択することにより指定します。

予約されたスペース

プロジェクト- *このオプションは、インストール プロジェクトで使用することができます。オブジェクトプロジェクトを構築している場合、この設定は表示されません。*

現在のリリースのメディアの種類が、ネットワーク イメージに該当する単一のディスクイメージを作成する場合は無効になります。特定のディスク イメージ フォルダーに、指定したディスク容量を確保 (空のままに) します。[ディスク] リストボックスでディスクイメージフォルダーの番号を選択し (たとえば、値 1 はディスクイメージフォルダー Disk1 を指定)、[予約されたスペース] 編集ボックスに、そのフォルダー内に確保する容量をキロバイトで入力します。

MD5 シグネチャを検証

プロジェクト- *このオプションは、インストール プロジェクトで使用することができます。オブジェクトプロジェクトを構築している場合、この設定は表示されません。*

このチェック ボックスがチェックされている場合、インストールされている各 MD5 のハッシュ値が、.cab ファイルに保存されている値と比較されます。この設定は、Setup.ini ファイルの [Startup] セクションの CheckMD5 キーに保存されます。

ヒント- *MD5 確認は破損したファイルを検出します。これは インターネットでのインストール中に便利です。MD5 確認を行わなかった場合、ファイル転送処理速度は速くなります。*

オブジェクトのインクルード-埋込み

プロジェクト- *このオプションは、オブジェクト プロジェクトで使用することができます。*

オブジェクトプロジェクトにサブオブジェクトが含まれる場合、リリースにはサブオブジェクトへの参照ではなく、サブオブジェクトそのものが含まれます。このオプションを使ってビルドしたリリースは、システムでサブオブジェクトが登録されていない InstallShield ユーザーが使用することができます。

オブジェクトのインクルード-参照

プロジェクト- *このオプションは、オブジェクト プロジェクトで使用することができます。*

オブジェクトプロジェクトにサブオブジェクトが含まれる場合、リリースにはサブオブジェクトそのものではなく、サブオブジェクトへの参照が含まれます。このオプションを使ってビルドしたリリースは、システムでサブオブジェクトが登録されていない InstallShield ユーザーは使用することができません。

[一般オプション – 別のディスク ファイル] ダイアログ ボックス

リリース ウィザードの [一般オプション] パネルで [別のディスク ファイル] をクリックすると、**[一般オプション – 別のディスク ファイル]** ダイアログ ボックスが開きます。このダイアログ ボックスで、サポート ファイル/ビルボード 内の 拡張ファイル の下にある その他 (Other) フォルダーにある各ファイルに対して、ディスクイメージフォルダー内の場所を指定することができます。

パネルのオプション

ファイル名

ディスクイメージフォルダー番号を指定するファイルを選択します。

Disk

選択したファイル用の希望のディスク イメージ フォルダーを選択します (たとえば、3 を選択すると、選択したファイルはディスク イメージ フォルダー Disk3 に配置されます)。

[履歴] ダイアログ ボックス

[履歴] ダイアログは、**[プロジェクト/ソース管理] メニューで [履歴の表示]** を選択すると表示されます。このダイアログには、セットアッププロジェクトのすべての Windows Installer テーブルを含むドロップダウンリスト ボックスが表示されます。

タスク ***ソース管理プログラムのテーブルの履歴を表示するには、以下の手順に従います:***

リストからテーブルを選択し、**[OK]** をクリックします。

[ダイアログのインポート] ダイアログ ボックス

[ダイアログのインポート] ダイアログ ボックスでは、インストール プロジェクトにインポートするダイアログを選択することができます。ダイアログのファイル (.isd ファイル) はリポジトリに格納するか、その他の別の場所に格納することができます。

タスク ***[ダイアログのインポート] ダイアログ ボックスにアクセスするには、以下の手順を実行します。***

1. **[ダイアログ]** ビューを開きます。
2. [ダイアログ] エクスプローラーで **[すべてのダイアログ]** を右クリックしてから、**[ダイアログのインポート]** を選択します。**[ダイアログのインポート]** ダイアログ ボックスが開きます。

ダイアログ ボックス オプション

リポジトリ項目

このボックスは、リポジトリにある利用可能なダイアログすべてを一覧表示します。プロジェクトに追加するダイアログを選択します。

インポートするダイアログ ボックス (.isd ファイル) のファイルがリポジトリに格納されていない場合、参照ボタンをクリックしてファイルを選択します。

[InstallScript ファイルのインポート] ダイアログ ボックス

[InstallScript ファイルのインポート] ダイアログ ボックスでは、インストール プロジェクトにインポートする複数の InstallScript (.rul) または InstallScript ヘッダー(.h) ファイルを選択することができます。ファイルはリポジトリ、またはその他の場所に格納することができます。

タスク ***[InstallScript ファイルのインポート] ダイアログ ボックスにアクセスするには、以下の手順を実行します。***

1. **InstallScript** ビューを開きます。
2. InstallScript エクスプローラーで、**[ファイル]** を右クリックして、**[スクリプト ファイルのインポート]** を選択します。**[InstallScript ファイルのインポート]** ダイアログ ボックスが開きます。

ダイアログ ボックス オプション

リポジトリ項目

このボックスは、リポジトリにある利用可能な InstallScript ファイルすべてを一覧表示します。プロジェクトに追加する InstallScript ファイル (複数可) を選択します。

インポートしたい InstallScript ファイルがリポジトリに格納されていない場合、[参照] ボタンをクリックしてファイルを選択します。

[SQL スクリプト ファイルのインポート] ダイアログ ボックス

[SQL スクリプト ファイルのインポート] ダイアログ ボックスでは、インストール プロジェクトにインポートする SQL スクリプト ファイル (複数可) を選択することができます。SQL スクリプト (.isd) ファイルはリポジトリに格納するか、その他の別の場所に格納することができます。

タスク ***[SQL スクリプト ファイルのインポート] ダイアログ ボックスにアクセスするには、以下の手順を実行します。***

1. **[SQL スクリプト]** ビューを開きます。
2. SQL スクリプト エクスプローラーで、[ファイル] を右クリックして、**[スクリプト ファイルのインポート]** をクリックします。**[SQL スクリプト ファイルのインポート]** ダイアログ ボックスが開きます。

ダイアログ ボックス オプション

リポジトリ項目

このボックスは、リポジトリにある利用可能な SQL スクリプト ファイルすべてを一覧表示します。プロジェクトに追加する SQL スクリプト ファイル (複数可) を選択します。

インポートしたい SQL スクリプト ファイルがリポジトリに格納されていない場合、参照ボタンをクリックしてファイルを選択します。

[アクションの挿入] ダイアログ ボックス

プロジェクト: *[アクションの挿入] ダイアログ ボックスは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アクションの挿入] ダイアログ ボックスを使って、標準アクション、カスタム アクション、またはダイアログをプロジェクトのシーケンスの 1 つに挿入できます。

タスク ***[アクションの挿入] ダイアログ ボックスを起動するには、以下の手順に従います:***

1. **[動作とロジック]** の下にあるビュー バー/ビュー リストで、**[カスタム アクションとシーケンス]** をクリックします。
2. シーケンス、ダイアログ、カスタム アクションまたは標準アクションを右クリックして、**[挿入]** をクリックします。

ヒント: *基本の MSI プロジェクトでは、このダイアログ ボックスを [DIM リファレンス] ビューからも起動できます。DIM リファレンスをクリックし、[シーケンス] タブをクリックします。シーケンス、ダイアログ、カスタム アクションまたは標準アクションを右クリックして、[挿入] をクリックします。*

ダイアログ オプション

テーブル 36 • [アクションの挿入] ダイアログ ボックスのオプション

オプション	説明
アクションの種類	挿入するアクションのタイプを選択します。 • **カスタム アクション** – カスタム アクションは、カスタム アクション ウィザードを使って作成できます。 • **ダイアログ** – [ユーザー インターフェイス] シーケンスに、ダイアログを挿入できます。カスタム ダイアログを作成するには、[ダイアログ] ビューを利用します。 • **DIM カスタム アクション** – これらのカスタム アクションはインストール プロジェクトに追加した 1 つまたは複数の DIM の中に格納されます。この DIM の GUID は、カスタム アクションの名前に追加されます。たとえば、DIM に LaunchNotepad という LaunchUtility アクションが格納されている場合、LaunchNotepad.xxxxxxxx_xxxx_xxxx_xxxx_xxxxxxxx のように表示されます (x が DIM の GUID)。 • **DIM ダイアログ** – これらのダイアログはインストール プロジェクトに追加した 1 つまたは複数の DIM リファレンスの中に格納されます。この DIM の GUID は、ダイアログの名前に追加されます。たとえば、DIM に CustomerInfo というダイアログが格納されている場合、CustomerInfo.xxxxxxxx_xxxx_xxxx_xxxx_xxxxxxxx のように表示されます (x がこの DIM の GUID)。 • **マージ モジュール カスタム アクション** – これらのカスタム アクションはインストール プロジェクトに関連付けた 1 つまたは複数のマージ モジュールの中に格納されます。このマージ モジュールの GUID は、カスタム アクションの名前に追加されます。たとえば、マージ モジュールに LaunchNotepad というカスタム アクションが格納されている場合、LaunchNotepad.xxxxxxxx_xxxx_xxxx_xxxx_xxxxxxxx のように表示されます (x がこのモジュールの GUID)。 • **マージ モジュール ダイアログ** – これらのダイアログはインストール プロジェクトに関連付けた 1 つまたは複数のマージ モジュールの中に格納されます。このマージ モジュールの GUID は、ダイアログの名前に追加されます。たとえば、マージ モジュールに CustomerInfo というダイアログが格納されている場合、CustomerInfo.xxxxxxxx_xxxx_xxxx_xxxx_xxxxxxxx のように表示されます (x がこのモジュールの GUID)。 • **標準アクション** – これらのアクションは Windows Installer に組み込まれています。 ***メモ:*** *シーケンスの 1 つに挿入されているカスタム アクションやダイアログが含まれているプロジェクトからマージ モジュールを削除すると、そのシーケンスから、これらのアクションやダイアログが自動的に削除されます。同じことが、DIM リファレンスでも発生します。*

テーブル 36 • [アクションの挿入] ダイアログ ボックスのオプション (続き.)

オプション	説明
アクションの選択	アクションのタイプを選択してから、特定のアクションを選択します。選択するには、挿入するアクションをハイライトします。
条件	アクションに対して条件を設定できます。条件が真と評価されない場合、アクションは実行されません。条件の詳細については、「条件ステートメントの構文」を参照してください。
Comments	このフィールドにコメントを入力します。入力したコメントは内部で使用するためのもので、エンドユーザーに表示されることはありません。

[InstallShield前提条件のプロパティ] ダイアログ ボックス

[InstallShield 前提条件プロパティ] ダイアログ ボックスは、[再配布可能ファイル] ビューまたは [前提条件] ビューで選択された InstallShield 前提条件を右クリックしてから [プロパティ] をクリックすると開きます。このダイアログ ボックスでは、選択された InstallShield 前提条件の場所を指定することができます。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。

プロジェクト: *基本の MSI プロジェクトと InstallScript MSI プロジェクトでは、[InstallShield 前提条件のプロパティ] ダイアログ ボックスを使って、特定のビルドから除外する InstallShield 前提条件のリリース フラグを設定することもできます。たとえば、前提条件を必要とする特別なアドオンを含んだ製品の特別なエディションにの*

み含める InstallShield 前提条件がある場合、その前提条件にフラグをつけることで、必要な場合にのみそれがビルドに含まれるようにします。

テーブル 37 • InstallShield[前提条件] ダイアログ ボックスの設定

設定	説明
ビルドの場所	InstallShield 前提条件を含むインストールをパッケージする場合、次のどれか 1 つの方法を使ってエンドユーザーに対して前提条件ファイルを提供することができます。 • **Web からダウンロードする** – InstallShield 前提条件ファイルを、必要に応じて、この InstallShield 前提条件に対して指定された URL からダウンロードします。 このオプションは、インストールがインターネットからダウンロードされるパッケージサイズとダウンロード時間を最小化したい場合に推奨されます。InstallShield 前提条件は、適切なバージョンが既にターゲット マシンにある場合はダウンロードされません。 このオプションは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。 • **Setup.exe から抽出する** – InstallShield 前提条件ファイルを Setup.exe に圧縮して、必要に応じて実行時に抽出します。 このオプションは、インストール全体が Setup.exe に完全に含まれていなければならない場合に選択します。[Web からダウンロードする] オプションを選択すると、インストールが小さくなりダウンロード時間も短縮されますが、[Setup.exe から抽出する] オプションは完全に独立したインストールを提供します。 このオプションは、基本の MSI および InstallScript MSI プロジェクトで使用できます。 • **[ソース メディアからコピーする]**– InstallShield 前提条件ファイルをソース メディアに格納します。 このオプションは、CD、DVD またはローカル ネットワークなどの固定メディアからインストールが非圧縮で実行される場合に使用します。 このオプションは、基本の MSI および InstallScript MSI プロジェクトで使用できます。 • **[メディアに含める]** – InstallShield 前提条件ファイルをソース メディアに格納します。 このオプションは、InstallScript プロジェクトで使用できます。 "ビルドの場所" 設定で選択するオプションは、[リリース] ビューでオーバーライドされることがありますので、ご注意ください。詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。 ***メモ•*** *InstallShield 前提条件が別の前提条件の依存ファイルとしてプロジェクトに追加される場合、前提条件依存ファイルの場所は、それを必要とする前提条件の場所設定に従います。*

テーブル 37 • InstallShield[前提条件] ダイアログ ボックスの設定 (続き.)

設定	説明
リリース フラグ	 ***プロジェクト:*** *この設定は、基本の MSI および InstallScript MSI プロジェクトで使用できます。* 文字列を入力します。文字列には、文字または数字を使用できます。複数のフラグを InstallShield 前提条件追加する場合は、コンマを使ってフラグを区切ります。

[言語] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

[コンポーネント] ビューの "言語" 設定で省略記号ボタン (...) をクリックすると、[言語] ダイアログ ボックスが開きます。このダイアログ ボックスを使って、ビルド時のフィルタリングに使用する、コンポーネントが依存する言語を指定できます。デフォルトでは、コンポーネントは言語に依存しません。つまり、コンポーネントのデータ (ファイル、レジストリ エントリなど) はどれも特定の言語に固有ではありません。

コンポーネントが特定の言語のみに依存する場合は、適切な言語のチェック ボックスを選択します。

プロジェクト: *InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは、プロジェクトの [一般情報] ビューで言語が選択されていない場合、その言語はプロジェクトのコンポーネント用に利用可能な言語の一覧には表示されません。*

[リンク タイプ] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

[リンク タイプ] ダイアログ ボックスは、[コンポーネント] ビューおよび [セットアップのデザイン] ビューで "リンク タイプ" 設定で参照ボタンをクリックしたとき開きます。また、[ファイルとフォルダー] ビューの **[インストール先コンピューターのフォルダー]** ペインにあるコンポーネントを右クリックして、[ファイルのリンク] をクリックしても開きます。

[リンク タイプ] ダイアログ ボックスを使うと、コンポーネントのファイル リンクがスタティックかダイナミックか、および、ダイナミックの場合、メディアのビルド時にリンクがどう判別されるかを指定できます。

ダイアログ ボックスの設定

スタティック リンク

コンポーネントには、明示的な完全修飾ファイル名によって指定されたファイルへのファイルリンクが含まれます。

ダイナミック リンク

コンポーネントには、ファイル名をワイルドカード (たとえば *.* や *.exe) で指定できる名前を持つファイルや、明示的なパスまたはビルド変数で名前を指定できるフォルダーへのファイルリンクが含まれます。

***メモ・** 次の設定は、[ダイナミック リンク] オプションが選択されたときのみ有効です。*

含めるファイルが入ったフォルダーを指定する

この設定は、リンクがあるファイルを含むフォルダーを指定します。明示的なパスを入力するか、ビルド変数を選択または入力するか、または定義済みのパス変数に選択したパスを追加します。

***注意・** 相対パス (例　¥MyFolder) を入力すると予期せぬ結果が生じます。*

参照

このボタンをクリックすると、明示的パスを指定できる [フォルダーの参照] ダイアログ ボックスを開きます。

サブフォルダーを含める

このチェック ボックスを選択すると、InstallShield は、**"含めるファイルを格納するフォルダーを指定する"** 設定で識別したフォルダー内のいずれかのサブフォルダーに含まれているファイルにダイナミック リンクを付加します。

***ヒント・** InstallScript エンジンでは、サブコンポーネントという概念をサポートされていません。ダイナミック ファイル リンクにサブフォルダーを含めると、各サブフォルダーに別々のコンポーネントが作成されます。*

ワイルドカード

ファイルのリストは、最初に [インクルード] コンボボックスに入力した内容と一致するすべてのファイルを含め、次にそれらのファイルから [エクスクルード] コンボボックスに入力した内容に一致するすべてのファイルを除外することによって作成されます。

インクルード

コンポーネントにファイルリンクが含まれるファイルの名前を指定します。パス変数を入力するか選択することによって単一のファイル名を入力するか、このファイル名指定に「*.*」または「*.exe」などのワイルドカードを使用することができます。複数のワイルドカードを指定するには、例のようにセミコロンで区切り、スペースを入れないようにします。

.txt;.doc

エクスクルード

コンポーネントにファイルリンクが含まれないファイルの名前を指定します。パス変数を入力するか選択することによって単一のファイル名を入力するか、このファイル名指定に「*.*」または「*.exe」などのワイルドカードを使用することができます。複数のワイルドカードを指定するには、例のようにセミコロンで区切り、スペースを入れないようにします。

.exe;.dll

上の例では、.exe ファイルも DLL ファイルもコンポーネントに含まれません。

[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックス

[一般情報] ビューにある [MSI ログの作成] 設定で [参照 (...)] ボタンをクリックすると、[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックスが表示されます。このダイアログ ボックスを使うと、Windows Installer 4.0 がインストールをログ記録するかどうかについて、コマンド ラインの使用やレジストリでのログ パラメーターの構成の必要なしに、プロジェクト全体を通して指定することができます。このダイアログ ボックスを使用して、ログ記録されるメッセージの種類をカスタマイズすることもできます。

テーブル 38 • [Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックス

オプション	説明
いいえ	インストールはログ記録されません。これがデフォルトの値です。
はい (MsiLoggingは voicewarmup のデフォルト値に設定されます)	InstallShield によって **voicewarmup** のデフォルト値が MsiLogging プロパティに挿入されます。 インストールが、Windows Installer 4.0 以降を搭載するターゲット システム、あるいは Windows Vista または Windows Server 2008 以降で実行された場合、以下の処理が行われます: • voicewarmup のデフォルト ログ モードに従って、インストーラーがログ ファイルを作成します。 • インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。 • SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。 この設定は、Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で Windows Installer 4.0 以降を使って実行するインストールに適用します。Windows Installer の古いバージョンを実行する以前のシステム上では、実行時ダイアログで **[Windows Installer ログを表示する]** チェック ボックスは表示されません。

テーブル 38 • [Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックス (続き.)

オプション	説明
カスタム MsiLogging 値	InstallShield は、このボックスに指定された値を MsiLogging プロパティに挿入します。 インストールが、Windows Installer 4.0 以降を搭載するターゲット システム、あるいは Windows Vista または Windows Server 2008 以降で実行された場合、以下の処理が行われます: • このボックスに指定されたカスタム値に基づいて、インストーラーがログ ファイルを作成します。 • インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。 • SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。 この設定は、Windows Vista 以降のシステム、および Windows Server 2008 以降のシステム上で Windows Installer 4.0 以降を使って実行するインストールに適用します。Windows Installer の古いバージョンを実行する以前のシステム上では、実行時ダイアログで **[Windows Installer ログを表示する]** チェック ボックスは表示されません。

重要• MsiLogFileLocation *プロパティは、読み取り専用のため、ログ ファイルの場所の設定および変更には使用できません。*

小文字のコンポーネントディレクトリ – 警告

このダイアログは、コンポーネントのインストール先設定時にディレクトリ識別子を小文字で指定したり、大文字と小文字を合わせて指定した場合に表示されます。

ディレクトリ識別子を大文字で指定すると、その値がユーザー インターフェイスから設定できる パブリックプロパティになります。大文字と小文字を混ぜたり、小文字でディレクトリ識別子を指定すると、ディレクトリエントリが プライベートプロパティ として定義されます。プライベートプロパティは、ユーザー インターフェイスから変更できません。

メモ• *この警告を表示しないようにするには、"**今後、このダイアログを表示しない**" オプションを選択します。*

[メディア ファイルのプロパティ] ダイアログ ボックス

リリース ウィザードの [アップデート] パネルで [インポート] または [変更] ボタンをクリックすると開きます。現在のプロジェクトにないリリースと、そのバージョン情報を決定する方法を指定できます。

メディア ヘッダー ファイル

希望のリリースのヘッダーファイルの完全修飾名を入力するか選択します(リリースのヘッダーファイルには、.hdr というファイル拡張子が付き、通常は Data1.hdr という名前が付いています)。または参照ボタンをクリックしてヘッダーファイルを参照します。

バージョン情報を読む...

バージョン情報をリリースのヘッダーファイルから自動的に読み取るように指定します。

以下でバージョン情報を指定する

バージョン情報がコンボボックスにタイプ入力または選択したものであることを指定します。

[マージ モジュールの構成可能な値] ダイアログ ボックス

構成可能な再配布可能ファイルは、**ModuleSubstitution** テーブル内の少なくとも 1 行によって参照される ModuleConfiguration テーブル内の少なくとも 1 行を含むマージ モジュールまたはオブジェクトです。これによって再配布可能ファイルの値を変更することができます。

[マージ モジュールの構成可能な値] ダイアログ ボックスの表示

[マージ モジュールの構成可能な値] ダイアログ ボックスは、[再配布可能ファイル] ビュー内でチェック ボックスが選択されている構成可能なマージ モジュールまたはオブジェクトを選択すると表示されます。選択された再配布可能ファイルがオブジェクトの場合、[マージ モジュールの構成可能な値] ダイアログ ボックスは、オブジェクト ウィザードが終了したとき表示されます。

構成可能マージ モジュールまたはオブジェクトを右クリックして、[マージ モジュールの構成] を選択することもできます。

ダイアログ ボックスの設定

[マージ モジュールの構成可能な値] ダイアログ ボックスには、構成可能値を変更できるグリッドが含まれています。左の列には構成可能値の名前が含まれています。右の列には、値のオプションが含まれています。ドロップダウンメニューから新しい値を選択することができます。値に関する情報については、再配布可能ファイル ベンダーのマニュアルを参照してください。

使用する値を指定したあと、OK をクリックして新しい値を保存します。これらの値は、プロジェクトをビルドするとき、.msi パッケージをビルドするために使用されます。

デフォルトに戻す

このボタンをクリックして、再配布可能ファイルのデフォルトの設定を復元します。再配布可能ファイル内の構成可能な値はすべてデフォルトに戻されます。

[マージ モジュールのプロパティ] ダイアログ ボックス

[マージ モジュールのプロパティ] ダイアログは、[再配布可能ファイル] ビューで選択されたマージ モジュールまたは再配布可能ファイルを右クリックし、[プロパティ] をクリックすると表示されます。次のタブがこのダイアログ ボックスと関連付けられています。

- メディア
- インストール先
- 構成可能値

[メディア] タブ

ファイルの場所

[メディア] タブを使って、出力の形式と保存先を指定することができます。

ソースメディア上 (非圧縮)

ファイルを非圧縮でメディアにコピーする場合はこのオプションを選択します。

新規 .CAB ファイル内

このオプションを選択すると、ソース メディアが .msi パッケージと同じ場所に配置されます。ただし .cab を .msi パッケージにストリームするように選択できます。

新規 .CAB ファイルを Windows Installer Package にストリーム入力する

このオプションを選択すると、.cab が .msi パッケージにストリームされます。

メモ: トランスフォーム (.mst) プロジェクトの種類では、.cab を .msi パッケージにストリームするオプションが無効になっています。これは、トランスフォーム プロジェクトが .cab ファイルを内部に格納することができないためです。非圧縮ファイルまたは外部 cab に格納されたファイルを追加するだけです。*

[インストール先] タブ

[マージ モジュールのプロパティ] ダイアログ ボックスにある [移動先] タブでは、移動先情報を構成できます。

設定

GUID

マージ モジュールの一意の GUID を表示します。

作成者

マージ モジュールの作成者を表示します。

バージョン

マージ モジュールのバージョンを表示します。

Destination

マージ モジュールのファイルの保存先を指定します。デフォルト設定 (マージ モジュールのデフォルトのインストール先を使用します) の利用をお勧めします。

[メソッドの参照] ダイアログ ボックス

[メソッドの参照] ダイアログ ボックスで、カスタム アクションが呼び出すマネージ アセンブリ内にあるクラスとメソッドを指定できます。[マネージ メソッドの定義] パネルで [参照] ボタンをクリックしたとき、カスタム アクション ウィザードでこのダイアログ ボックスが表示されます。また、このダイアログ ボックスは、[メソッド シグネチャ] ダイアログ ボックスで [参照] ボタンをクリックしたときも表示されます。

メモ: *パブリック クラスのパブリック メソッドのみが、選択肢として提供されています。また、オーバーロードしているメソッドを選択することはできません。*

参照機能は .NET Framework がインストールされているときのみ使用できます。また、この機能は、マネージ アセンブリの場所が Binary *テーブルに設定されていて、製品と共にインストールされているときのみ使用可能です。アセンブリのパスがプロパティ、または* Directory *テーブルのディレクトリを参照する場合は使用できません。*

また、参照機能は 32 ビット アセンブリおよび Microsoft intermediate language (MSIL) ファイルでのみ使用できます。64 ビット アセンブリでは使用できません。

[メソッド シグネチャ] ダイアログ ボックス

[メソッド シグネチャ] ダイアログ ボックスは、[カスタム アクションとシーケンス] ビュー (または [カスタム アクション] ビュー) でマネージ カスタム アクションの "メソッド シグネチャ" 設定で参照ボタン (...) をクリックしたとき表示されます。このダイアログ ボックスで、カスタム アクションが呼び出すパブリック クラス メソッドを指定できます。

[メソッド シグネチャ] ダイアログ ボックスには、次の設定があります。

テーブル 39 · [メソッド シグネチャ] パネルの設定

設定	説明
クラス	メソッドが関連付けられているパブリック クラスの名前 (完全名前空間を含む) を入力します。次のフォーマットを使用してください: **MyNamespace.MyClass** また、[参照] ボタンをクリックして、[メソッドの参照] ダイアログ ボックスを指定することもできます。このダイアログ ボックスを使って、[アクションのパラメーター] パネルで選択したマネージ アセンブリにあるパブリック クラスの一覧からパブリック メソッドを選択することができます。 ***メモ:*** *参照機能は .NET Framework がインストールされているときのみ使用できます。また、この機能は、マネージ アセンブリの場所が* Binary *テーブルに設定されていて、製品と共にインストールされているときのみ使用可能です。アセンブリのパスがプロパティ、または* Directory *テーブルのディレクトリを参照する場合は使用できません。* *また、参照機能は 32 ビット アセンブリおよび Microsoft intermediate language (MSIL) ファイルでのみ使用できます。64 ビット アセンブリでは使用できません。*

テーブル 39 • [メソッド シグネチャ] パネルの設定 (続き.)

設定	説明
メソッド	インストールで呼び出すパブリック メソッドを入力します。また、[参照] ボタンをクリックして、[メソッドの参照] ダイアログ ボックスを指定することもできます。このダイアログ ボックスを使って、[アクションのパラメーター] パネルで選択したマネージ アセンブリにあるパブリック クラスの一覧からパブリック メソッドを選択することができます。 ***メモ•*** *参照機能は .NET Framework がインストールされているときのみ使用できます。また、この機能は、マネージ アセンブリの場所が* Binary *テーブルに設定されていて、製品と共にインストールされているときのみ使用可能です。アセンブリのパスがプロパティ、または* Directory *テーブルのディレクトリを参照する場合は使用できません。* *また、参照機能は 32 ビット アセンブリおよび Microsoft intermediate language (MSIL) ファイルでのみ使用できます。64 ビット アセンブリでは使用できません。*
カスタム メソッド シグネチャを使用する	デフォルトのメソッド シグネチャを使用する場合、このチェック ボックスをクリアします。デフォルトのメソッド シグネチャを利用すると、シグネチャに 1 つまたは複数の無効なパラメーターまたは単一 MsiHandle パラメーターがあるとき、インストールで、メソッドが MsiHandle パラメーターが使われて呼び出されます。 カスタム メソッド シグネチャを使用する場合、このチェック ボックスを選択します。そのあと、必要に応じて、引数と戻り値プロパティを指定します。 [参照] ボタンをクリックしてクラスとメソッドを指定すると、関連付けられたパラメーターが自動的に追加されます。メソッドに渡す各パラメーターの値を指定します。 メソッドのシグネチャの指定については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。

テーブル 39 • [メソッド シグネチャ] パネルの設定 (続き.)

設定	説明
引数	カスタム メソッド シグネチャを使用する場合、選択したメソッドに渡す引数の一覧を指定します。 [参照] ボタンをクリックしてクラスとメソッドを指定すると、関連付けられたパラメーターがこの領域にあるグリッドに自動的に追加されます。 プロパティを引数として使用する場合、パラメーターの値フィールドをクリックして、一覧からプロパティを選択します。一覧の中にないプロパティを使用するには、そのプロパティの名前を入力します。プロパティは実行時に定義される必要があります。 コンテキスト メニューで、引数を使用するためのオプションが表示されます。コンテキスト メニューにアクセスするには、[引数] グリッドで引数を右クリックします。以下は、提供されるコンテキスト メニューのコマンドです: • **[追加]** — グリッドに引数を追加します。 • **[削除]** — グリッドから引数を削除します。 このグリッドは、**[カスタム メソッド シグネチャを使用する] チェック ボックス**が選択されたときのみ使用できます。 引数の指定については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。
戻り値プロパティ	値にメソッドの戻り値を設定するプロパティの名前を選択または入力します。 この一覧は、**[カスタム メソッド シグネチャを使用する] チェック ボックス**が選択されたときのみ使用できます。 戻り値プロパティの指定については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。

[MIME の種類] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[MIME の種類] ダイアログ ボックスを使って、ファイル名拡張子と、Web サーバーからブラウザまたはメール クライアントにスタティック ファイルとして提供される対応コンテンツの種類とのマッピングを追加、編集、または削除します。

[MIME の種類] ダイアログ ボックスは、[IIS 構成] ビュー内から利用することができます。このダイアログ ボックスを開くには、エクスプローラーで Web サイト、アプリケーション、または仮想ディレクトリをクリックします。次に、"MIME の種類" 設定で省略記号ボタン (...) をクリックします。

テーブル 40 • [MIME の種類] ダイアログ ボックスの設定

設定	説明
追加	ファイル名拡張子と、Web サーバーからブラウザまたはメール クライアントにスタティック ファイルとして提供される対応コンテンツの種類とのマッピングのためのエントリを追加します。これによって、[MIME の種類を追加] ダイアログ ボックス が開きます。
編集	既存する MIME の種類を編集するには、MIME の種類を選択して、このボタンをクリックします。
Delete	既存する MIME の種類を削除するには、MIME の種類を選択して、このボタンをクリックします。

[ダイナミック リンクの変更] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[ダイナミック リンクの変更] ダイアログ ボックスでは、ダイナミック リンクのすべての設定が表示されます。

タスク

ダイアログ ボックスを起動するには、次の手順を実行します。

1. **[コンポーネント]** ビュー、または **[セットアップのデザイン]** ビューを開きます。
2. エクスプローラーで、変更するダイナミックリンクを含むコンポーネントを開き、ファイルをクリックします。
3. [ファイル] ペインで右クリックして **[ダイナミック ファイル リンク]** をクリックします。**[ダイナミック リンクの変更]** ダイアログ ボックスが開きます。

テーブル 41 • [コンポーネントのプロパティ] ダイアログ ボックスの [ファイルのリンク] タブにある設定

設定	説明
ソース	この列では、ソース ファイルが格納されている場所が表示されます。通常、パス変数が使用されます。

テーブル 41 • [コンポーネントのプロパティ] ダイアログ ボックスの [ファイルのリンク] タブにある設定 (続き

設定	説明
サブフォルダーを含める	この列は、各サブフォルダー内のファイルにダイナミック リンクを付加するかどうかを示します。 InstallShield で、サブフォルダー内にあるダイナミック リンクを持つファイルにコンポーネントがどう作成されるかについては、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
自己登録	この列は、InstallShield で、ダイナミック リンクにあるすべてのファイルを自己登録にするかどうかを示します。 ファイルが 64 ビット コンポーネントの一部で、ダイナミック リンクについて Self-Register 列が Yes になっている場合、インストール時にターゲット マシンで 64 ビット自己登録が実行されます。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。
コンポーネントの作成	この列は、InstallShield で、ビルド時にダイナミック ファイルがリリースに追加されるとき、コンポーネントがどう作成されるかを示します。有効オプションは、以下のとおりです: • **ベスト プラクティス** – ダイナミック リンクがあるファイルのコンポーネントを作成するとき、InstallShield はベスト プラクティスを順守します。 • **ディレクトリごと** – 各フォルダーとサブフォルダーについてコンポーネントが 1 つ作成されます。 これらの 2 つのメソッドについては、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
詳細	この列は、ダイナミック リンクからのファイルを選択および除外するときのフィルター基準を示します。

ヒント• *[ファイルのリンク] タブを使用するとき、次のガイドラインを参考にしてください:*

- *新しいダイナミック リンクを追加する場合、[新しいリンク] ボタンをクリックします。[ダイナミック ファイル リンクの設定] ダイアログ ボックスが開きます。*
- *既存のリンクの設定を変更する場合、リンクを選択してから、[変更] ボタンをクリックします。[ダイナミック ファイル リンクの設定] ダイアログ ボックスが開きます。*
- *リンクとそれに関連するダイナミック リンクがあるファイルを削除する場合、リンクを選択してから、[削除] ボタンをクリックします。*

[プロパティの変更] ダイアログ ボックス / リリース ウィザード – [プラットフォーム] パネル

プロジェクト• *次のトピックは、InstallScript プロジェクトに適用します。*

リリース ウィザードの [プロパティの変更] ダイアログ ボックスと [プラットフォーム] パネルで、リリースが 1 つまたは複数のプラットフォームに固有かどうかを指定することができます。コンポーネントに指定されたプラットフォームが、このリリースに選択されたプラットフォームのどれにも一致しない場合、そのコンポーネントはリリースに含まれません。

タスク ***[プロパティの変更] ダイアログ ボックスにアクセスするには、***

1. **[メディア]** の下にあるビュー バー/ビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースをクリックします。
3. **[ビルド]** タブをクリックします。
4. **"プラットフォーム"** フィールドの値をクリックして、参照 (...) ボタンをクリックします。

タスク ***リリース ウィザードの [プラットフォーム] パネルにアクセスするには、***

1. **[メディア]** の下にあるビュー バー/ビュー リストで、**[リリース]** をクリックします。
2. **[リリース]** エクスプローラーで、構成するリリースを右クリックして、**リリース ウィザード**をクリックします。
3. ウィザード内の各パネルを完成していくと、**[プラットフォーム]** パネルに到達します。

次のテーブルは、リリース ウィザードの [プロパティの変更] ダイアログ ボックスおよび [プラットフォーム] パネルで表示される設定です。

テーブル 42 ・ リリース ウィザードの [プロパティの変更] ダイアログ ボックスおよび [プラットフォーム] パネルで表示される設定

設定	説明
"プラットフォーム フィルター" 設定で指定されたプラットフォームを使用する	選択されたリリースが [一般情報] ビューの "プラットフォーム フィルター" 設定で指定されたプラットフォームをすべてサポートする場合、このオプションを選択します。このオプションを選択すると、ビルド時にコンポーネントはリリースから除外されません。

テーブル 42 • リリース ウィザードの [プロパティの変更] ダイアログ ボックスおよび [プラットフォーム] パネルで表示される設定 (続き.)

設定	説明
以下でサポートするプラットフォームを指定	選択されたリリースが 1 つまたは複数のオペレーティング システムに固有の場合、この設定を選択して、該当するオペレーティング システムを選択します。 コンポーネントに指定されたプラットフォームが、この設定で選択されたプラットフォームのどれにも一致しない場合、そのコンポーネントはリリースに含まれません。 ***ヒント•*** *連続する複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択したあと SHIFT キーを押しながら最後のオペレーティング システムを選択します。連続しない複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択してから CTRL キーを押しながらその他の各オペレーティング システムを選択します。* ***プロジェクト•*** *InstallScript プロジェクトの場合、プロジェクト レベルでチェック ボックスがクリアになっているオペレーティング システムは、このダイアログ ボックスで表示されるオペレーティング システムの一覧からすべて除外されます。*

[モジュール依存関係] ダイアログ ボックス

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *マージ モジュール*
- *MSM データベース*

[一般情報] ビューの "モジュール依存関係" 設定にある省略記号ボタン (...) をクリックすると、[モジュール依存関係] ダイアログ ボックスが開きます。このダイアログ ボックスでは、作成中のマージ モジュールが必要とするマージ モジュールを選択できます。

ヒント• *ダイアログ ボックスには、[オプション] ダイアログ ボックス の [マージ モジュール] タブで指定されたディレクトリで検出されたマージ モジュールがリストされます。マシンに存在しない依存関係を追加するには、[一般情報] ビューの "モジュール依存関係" 設定で* ***[新しいモジュール依存関係を追加する]*** *ボタンをクリックします。*

[除外モジュール] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *マージ モジュール*
- *MSM データベース*

一部のマージ モジュールは他のマージ モジュールがあると、正常に作動しません。古いバージョンのモジュールが新しいモジュールと互換性を持たないために、このような状況が発生する場合があります。このため、新しいモジュールを含むインストールから、互換性を持たないモジュールを除外しなくてはなりません。

[一般情報] ビューの "除外モジュール" 設定にある省略記号ボタン (...) をクリックすると、[除外モジュール] ダイアログ ボックスが開きます。このダイアログ ボックスでは、作成中のマージ モジュールと互換性を持つマージ モジュールを選択できます。

インストール プロジェクトに除外を行うマージ モジュールを追加した場合で、これらの除外モジュールがインストール プロジェクトに既に追加されていた場合、InstallShield がプロジェクトから除外モジュールへの参照をすべて内部的に削除します。

注意: *除外を行うモジュールの追加後にインストールに追加された除外モジュールはいずれも削除されず、またそれらに互換性がないことを示す警告が表示されることもありません。*

ヒント: *ダイアログ ボックスには、[オプション] ダイアログ ボックス の [マージ モジュール] タブで指定されたディレクトリで検出されたマージ モジュールがリストされます。マシンに存在しない除外モジュールを追加するには、[一般情報] ビューの "除外モジュール" 設定で **[新しいモジュール依存関係を追加する]** ボタンをクリックします。*

[MSI 値] ダイアログ ボックス

[MSI 値] ダイアログ ボックスで、表示された Windows Intaller パッケージの Registry テーブルにあるプライマリ キーを作成または変更できます。詳細については、「Registry のプライマリ キーを指定する」を参照してください。

[MST SIS の設定] ダイアログ ボックス

このダイアログを使うと、エラーを抑制し、トランスフォームおよび基本の MSI パッケージの検証オプションを指定できます。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「Database.CreateTransformSummaryInfo」を参照してください。

ダイアログ オプション

エラー抑制オプション

抑制するエラー条件の横のチェック ボックスを選択します。

テーブル 43 • エラー抑制オプション

条件	説明
既存の行を追加	既に存在する行を追加します。
既存のテーブルを追加	既に存在するテーブルを追加します。
見つからない行を削除	存在しない行を削除します。
見つからないテーブルを削除	存在しないテーブルを削除します。
見つからない行を変更	存在しない行を更新または変更します。
コードページの変更	トランスフォームページと .msi コードページが一致せず、いずれのコードページも中立ではありません。

検証オプション

このセクションでは、基本の MSI パッケージでトランスフォームを適用する前にトランスフォームパッケージが検証する情報を指定できます。

テーブル 44 • 検証オプション

検証オプション	説明
言語の一致	基本の MSI とトランスフォームの両方の言語を検証します。言語が一致しないとトランスフォームは適用されません。
アップグレード コードの一致	基本の MSI とトランスフォームの両方のアップグレードコードを検証します。アップグレードコードが一致しないとトランスフォームは適用されません。
製品コードの一致	基本の MSI とトランスフォームの両方の製品コードを検証します。製品コードが一致しないとトランスフォームは適用されません。

バージョン比較

比較する種類を選択します。

バージョンレベル

メジャーバージョンだけを比較するかマイナーバージョンも比較するか選択します。

[複数行文字列値] ダイアログ ボックス

[複数行文字列値の編集] ダイアログ ボックスでは、複数行のレジストリ行を入力できます。このダイアログ ボックスは、[レジストリ] ビルドでレジストリ キーの複数行文字列値を追加または編集する時に開きます。

(複数行編集コントロール)

複数行編集フィールドで、各ヌル区切り文字列を入力または変更します。グリッドを右クリックしてコンテキスト メニューを表示するか、次のアクションに関連した以下のグリッドのキーを押します。

テーブル 45 ・ 複数行編集フィールドの操作

アクション	ショートカット キー
文字列の追加	INSERT
文字列の名前変更	F2
文字列の削除	DELETE
上へ移動	UP ARROW
下へ移動	DOWN ARROW

各ヌル区切り文字列の行を入力したら、[OK] をクリックします。文字列は自動的に連結されます。

メモ・ *スペースだけが含まれた文字列は使用できますが、文字列を空白にしたり、[~] にすることはできません。これはプロジェクトに保存される文字列の区切り文字として [インストール先コンピューターのレジストリデータ] ペインに表示されます。この区切り文字は、レジストリ値がターゲット システムに作成されたときには含まれません。*

レジストリ値の変更方法を選択します。

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

Windows Installer ベースのプロジェクトでは、次のオプションから値の追加場所を選択できます。

テーブル 46 • レジストリ値の変更に関するオプション

オプション	説明
後に追加する	既存のレジストリ値文字列の末尾に追加します。
前に追加	既存のレジストリ値文字列の前に追加します。
Replace	既存のレジストリ値文字列を置き換えます。

[新規依存関係] ダイアログ ボックス

[新規依存関係] ダイアログ ボックスは、InstallShield 前提条件エディターの [依存関係] タブにある [追加] ボタンまたは [変更] ボタンをクリックすると開きます。このダイアログ ボックスでは、この前提条件が依存するその他の InstallShield 前提条件 (.prq ファイル) を指定します。

テーブル 1 - [新規依存関係] ダイアログ ボックスのオプション

ダイアログ オプション	説明
File	現在の InstallShield 前提条件に必要な .prq ファイルの名前とパスを入力します。.prq ファイルを参照して見つけるには、省略記号 (...) ボタンをクリックします。InstallShield 前提条件ファイル (.prq) は、デフォルトで次の場所に格納されています: *InstallShield Program Files フォルダー*¥SetupPrerequisites InstallShield 前提条件ファイルが保存されている場所についての詳細は、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。

[新規ファイル] ダイアログ ボックス

[新規ファイル] ダイアログ ボックスは、InstallShield 前提条件エディターの [含めるファイル] タブにある [追加] ボタンをクリックすると開きます。ダイアログ ボックスでは、InstallShield 前提条件とするファイルを指定することができます。

テーブル 2 - [新規ファイル] ダイアログ ボックスのオプション

オプション	説明
File	ファイルのパスと名前を入力します。ファイルを参照して見つけるには、省略記号 (...) ボタンをクリックします。
ファイルへの URL	エンドユーザーが Web からファイルをダウンロードすることが可能な場合、このボックスにその URL を指定します。この URL は、インストール作成者が [再配布可能ファイル] ビューを利用してインターネットからローカル マシンへ InstallShield 前提条件をダウンロードする時に InstallShield が利用する URL と同じです。 たとえば、そのファイルの URL が http://www.mywebsite.com/Folder1/Notepad.exe の場合、このボックスに http://www.mywebsite.com/Folder1 を入力します。

[新規プロジェクト] ダイアログ ボックス

このダイアログ ボックスは、InstallShield で新しいプロジェクトを作成すると表示されます。このダイアログ ボックスでは、プロジェクトの種類の選択やプロジェクトの名前、プロジェクト ファイルの格納場所の指定などを行うことができます。プロジェクトタイプを選択してから OK をクリックすると、InstallShield でプロジェクトが開きます。

プロジェクト: *InstallScript、Basic MSI、MSIX、または InstallScript MSI などのインストール プロジェクトを選択すると、プロジェクト作成を支援するプロジェクト アシスタントが起動します。*

ダイアログ ボックスのタブ

テーブル 3 - [新規プロジェクト] ダイアログ ボックスのタブ

タブ	説明
共通	このタブには、最もよく使用されるプロジェクトの種類が表示されます。
InstallScript	このタブには、InstallScript インストール プロジェクトをはじめ、InstallShield インストール エンジンで使われるすべてのプロジェクトタイプが表示されます。以前 InstallShield Professional を使用したことがあるユーザーは、これらのプロジェクトの種類にすぐ慣れることができます。
Windows Installer	このタブには、基本の MSI インストール プロジェクトをはじめ、Microsoft Windows Installer エンジンを利用するプロジェクトが含まれています。
MSIX	このタブは、MSIX は Windows アプリ パッケージ形式で、Windows アプリにモダンなパッケージ エクスペリエンスを提供するプロジェクトを含みます。
すべてのタイプ	このタブには、InstallShield で使用できるすべてのプロジェクトの種類が表示されます。ここには、リポジトリにあるテンプレートだけでなく、以前に保存したプロジェクトテンプレートも含まれます。

ダイアログ ボックス オプション

テーブル 4 - [新規プロジェクト] ダイアログ ボックスのオプション

オプション	説明
プロジェクト名	このボックスにプロジェクト名を入力します。 ***プロジェクト-*** *新しいマージ モジュール プロジェクトを作成するとき、このダイアログ ボックスで指定する名前は 35 文字以下でなくてはなりません。これは、入力した名前がオブジェクトまたはモジュールファイルの* ModuleSignature *テーブルにある ModuleID フィールドで GUID と共に利用されるため、また ModuleID フィールドの名前部分の最大文字数が 35 文字のためです。* *新しい DIM を作成するとき、このダイアログ ボックスで指定する名前は 35 文字以下でなくてはなりません。*
場所	場所を入力するか、[参照] をクリックしてプロジェクトの場所に移動します。表示されているデフォルトのプロジェクトの場所を変更するには、[オプション] ダイアログ ボックスの [ファイルの場所] タブで指定された [プロジェクトの場所] パスを変更します。
[プロジェクト名] サブフォルダーにプロジェクトを作成	このチェック ボックスは、InstallShield で、[プロジェクト] の場所にプロジェクトの名前を持つサブフォルダーを作成する場合に選択します。

[オペレーティング システム] ダイアログ ボックス

このダイアログでは、(InstallScript MSI プロジェクトの) コンポーネントを関連付けるオペレーティング システムを指定することができます。

ダイアログ オプション

オペレーティング システム

コンポーネントをインストールするオペレーティング システムを選択します。オペレーティング システムを一切選択しないと、ターゲットのオペレーティング システムに関係なくコンポーネントがインストールされます。

[オプション] ダイアログ ボックス

[オプション] ダイアログ ボックスを使って、InstallShield でプロジェクトの作成および作業を行うときのプリファレンスを指定できます。

タスク ***[オプション] ダイアログ ボックスにアクセスするには、以下の手順を実行します。***

[ツール] メニューから **[オプション]** ボタンをクリックします。

[オプション] ダイアログ ボックスは、複数のタスク関連タブで構成されています:

- [全般] タブ
- [ファイルの場所] タブ
- [パス変数] タブ
- [ユーザー インターフェイス] タブ
- [プリファレンス] タブ
- [品質] タブ
- [.NET] タブ
- [ファイル ビュー] タブ
- [ファイルの拡張子] タブ
- [前提条件] タブ
- [ソース管理] タブ
- [ディレクトリ] タブ
- [リソース] タブ
- [検証] タブ
- [リポジトリ] タブ
- [SQL スクリプト] タブ
- [マージ モジュール] タブ

[全般] タブ

[オプション] ダイアログ ボックスの **[全般]** タブでは、プロジェクトの全般オプションを設定します。

テーブル 5 · [全般] タブの設定

設定	プロジェクトの種類	説明
セットアップ ベスト プラクティスを強制する	基本の MSI、InstallScript MSI	インストール デザインが セットアップ ベスト プラクティス に準拠するように InstallShield による監視を行うには、このオプションを選択します。
ヘルプ ウィンドウを常に手前に表示する	All	ヘルプウィンドウが InstallShield インターフェイスの上に留まるようにするには、このチェック ボックスを選択します。InstallShield が強調されている時にヘルプウィンドウを背景表示する場合は、このチェック ボックスを選択解除します。
最初のエラーの発生時にビルド プロセスを停止	基本の MSI、InstallScript、InstallScript MSI	ビルドエラーが発生したときにビルドプロセスを停止する場合、このチェック ボックスを選択します。
ISSetAllUsers アクションを自動作成する	基本の MSI、InstallScript MSI	プロジェクトの Upgrade テーブルに 1 つ以上のレコードが含まれる場合、ISSetAllUsers アクションをシーケンスに追加するにはこのチェック ボックスを選択します。

[ファイルの場所] タブ

[オプション] ダイアログ ボックスの **[ファイルの場所]** タブでは、プロジェクトファイルを保存する場所、並びにすべてのマージ モジュールを保存する場所のデフォルト ディレクトリを設定することができます。

テーブル 6 · [ファイルの場所] タブの設定

設定	プロジェクトの種類	説明
プロジェクトの場所	All	プロジェクトの位置は、新しいインストール プロジェクトの [お気に入り] フォルダーになります。プロジェクト (.ism) ファイル、インストールパッケージ (.msi ファイル)、およびディスクイメージファイルなどのソースファイルやリリースファイルはすべて、[お気に入り] のサブフォルダーに保管され、現在は [不明] です。完全なパスを入力するか、または [参照] ボタンをクリックして、このフォルダーへ移動できます。 デフォルトで、プロジェクトは次の場所に格納されています。 C:¥InstallShield 2021 Projects ***ヒント:*** *プロジェクトの保存場所を変更しても、以前のプロジェクトの場所にある既存のフォルダーとファイルは移動しません。リリースを新しい名前でビルドするか、フォルダーを手動で移動させて、保管する場所を変更する必要があります。*
ダイアログの場所	基本の MSI	ダイアログウィザード がダイアログを検索するパスを入力します。この場所にあるダイアログは、ダイアログウィザードのダイアログテンプレートパネル のリストに表示されます。追加のパスはカンマで区切ります。

[パス変数] タブ

パス変数タブでは、ユーザーレベルパス変数オプションを設定することができます。

テーブル 7 · [パス変数] タブの設定

設定	プロジェクトの種類	説明
常にパス変数を推奨する	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、QuickPatch、スイート/アドバンスト UI	InstallShield で既存のパス変数を自動的にファイルに割り当てる場合、このオプションを選択します。たとえば、Program Files フォルダーに格納されているインストール プロジェクトにファイルを追加する場合、InstallShield では、直接入力したパスではなく、<ProgramFilesFolder> が使用されるようになります。 InstallShield でパスを自動作するには、このオプションの下のチェック ボックスを選択します。新しい変数を作成すると、<MYVAR#> (# は連続する番号) というタイトルが付けられます。パス変数エディターを使用すると、この変数の名前を変更できます。
パス変数を推奨しない	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、QuickPatch、スイート/アドバンスト UI	すべてのファイルのリンクで直接入力したパスを使用する場合は、このオプションを選択します。このオプションが選択されていると、InstallShield はパス変数を表示しません。
常にパス変数の推奨ダイアログを表示する	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、QuickPatch、スイート/アドバンスト UI	ファイルをインストール プロジェクトと関連付けるたびに [パス変数の推奨] ダイアログ ボックスを表示する場合は、このオプションを選択します。

[ユーザー インターフェイス] タブ

[オプション] ダイアログ ボックスの **[ユーザー インターフェイス]** タブでは、ユーザー インターフェイスの環境を設定します。これらの設定は、このシステムで作成するすべてのプロジェクトで保守されます。

テーブル 8 · [ユーザー インターフェイス] タブの設定

設定	プロジェクトの種類	説明
新しいコンポーネントおよび機能を常に作成する	基本の MSI、InstallScript、InstallScript MSI	必要な場合に応じて コンポーネントや機能を自動的に作成する場合は、このチェック ボックスを選択します。このチェック ボックスを選択しなかった場合、コンポーネントと機能の作成が必要な場合にメッセージが表示されます。

テーブル 8 • [ユーザー インターフェイス] タブの設定 (続き.)

設定	プロジェクトの種類	説明
機能、コンポーネント、およびリリースの削除を常に確認する	基本の MSI、InstallScript、InstallScript MSI	機能、コンポーネントまたはリリースを削除した時、常に確認ダイアログ ボックスが表示されるようにする場合、このチェック ボックスを選択します。
スクリプト用語の変換を常に確認する	基本の MSI、InstallScript、InstallScript MSI	InstallShield Professional で作成したプロジェクトを開いたとき、確認ダイアログ ボックスが表示されるようにする場合、このチェック ボックスを選択します。ダイアログ ボックスは、スクリプトを変換して InstallShield InstallScript 用語を使うかどうかを聞いてきます。
最適な機能またはコンポーネントを自動的に決定する	基本の MSI、InstallScript、InstallScript MSI	[レジストリ] ビルドで構成したレジストリ データを含む機能またはコンポーネントを自動判別する場合、このチェック ボックスを選択します。このチェック ボックスがクリアにされると、ビュー フィルターで [すべてのアプリケーション]データ オプションまたは [機能] オプションのレジストリ データを作成する時に、適切な機能またはコンポーネントの名前を指定できるダイアログ ボックスが表示されます。
IDE ウィザードで [ようこそ] パネルを表示する	基本の MSI、InstallScript、InstallScript MSI	ウィザードを実行したとき、常に [ようこそ] パネルが表示されるようにする場合、このチェック ボックスを選択します。
Web サイトの削除を警告する	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	[IIS の構成] ビューで Web サイトを削除する時に、Web サイトに 1 つ以上の IIS アプリケーションまたは仮想ディレクトリが含まれている時に確認ダイアログ ボックスを表示する場合、このチェック ボックスを選択します。確認ダイアログ ボックスは、Web サイトを削除する選択をした時、Web サイトのアプリケーションおよび仮想ディレクトリがプロジェクトから削除されることを警告します。
InstallScript プロジェクト タイプを表示する	InstallScript	このチェック ボックスは、AdminStudio に含まれている InstallShield で使用可能です。デフォルトで、このチェック ボックスは選択されていません。 [新規] ダイアログ ボックスで InstallScript プロジェクトと InstallScript MSI プロジェクトを非表示にする場合、このチェック ボックスを選択します。

[プリファレンス] タブ

[オプション] ダイアログ ボックスの **[プリファレンス]** タブでは、プログラムの環境を設定します。これらの設定は、このシステムで作成するすべてのインストール プロジェクトで保守されます。

テーブル 9 - [プリファレンス] タブの設定

設定	プロジェクトの種類	説明
自己登録	基本の MSI、InstallScript MSI	このセクションでは、ファイルを自己登録にしたときに InstallShield が使用する自己登録方法を選択できます。この設定を変更すると以降の自己登録設定に適用され、これまでに自己登録に指定されたファイルは影響を受けません。 • **Windows Installer 自己登録テーブル (SelfReg)** – 自己登録にするすべてのファイルに Windows Installer 自己登録テーブルを使用する場合、このオプションを選択します。 • **Windows Installer 自己登録テーブル (SelfReg)** – 自己登録にするすべてのファイルに Windows Installer 自己登録テーブルを使用する場合、このオプションを選択します。 このセクションでは、自己登録ファイルの COM 抽出のデフォルトのタイミングを選択できます。これは、自己登録ファイルを [ファイルとフォルダー] ビューまたはコンポーネント ウィザードから追加した場合に使用するデフォルト設定です。 • **ビルド時に COM 抽出が行われます** – ビルド中に抽出するには、このオプションを選択します。ビルドログファイルに COM 情報が表示されます。 • **ファイルを追加すると直ちに COM 抽出が発生します** – COM ファイルを追加した直後に COM データの抽出を行うには、このオプションを選択します。このオプションを選択すると、InstallShield はコンポーネントの [COM 登録詳細設定] ビューを作成します。
[実行] コマンド	基本の MSI、InstallScript MSI	インストールのデバッグまたはインストールを [ビルド] メニューから再起動して再度実行する前に製品を自動的にアンインストールするには、**[インストールまたはデバッグの前に自動的に製品をアンインストール]** チェック ボックスを選択します。
参照の整合性	基本の MSI、InstallScript、InstallScript MSI	ダイレクト エディターで作業中に InstallShield が自動的に参照の整合性を保つようにするには、**[参照の整合性を保つ]** チェック ボックスを選択します。たとえば Control テーブルには Dialog_ 列があり、この列が Dialog テーブルの外部キーであることを意味します。Dialog テーブルでキー名を変更すると、Control テーブルのキーの名前も変わります。またダイアログ全体を削除すると、すべてのコントロールも削除されます。これは、このような種類の外部キー関係を持つすべてのテーブルに当てはまります。

テーブル 9 • [プリファレンス] タブの設定 (続き.)

設定	プロジェクトの種類	説明
プロジェクトの再ロード	All	InstallShield を起動したときに、最後に開いたプロジェクトを自動的に再ロードするには、**[最後に開いたプロジェクトを起動時に再ロードする]** チェック ボックスを選択します。

[品質] タブ

***プロジェクト・[品質]** タブは、次のプロジェクトの種類で使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[オプション] ダイアログボックスの **[品質]** タブでは、InstallShieldの**カスタマー エクスペリエンス向上**プログラムへの参加を構成することができます。このプログラムは、匿名使用データの収集を通じて、Revenera が InstallShield の信頼性、パフォーマンス、およびユーザー エクスペリエンスを向上させるのに役立つように設計されています。

カスタマーエクスペリエンス向上プログラムへの参加を選択または設定すると、InstallShield ユーザー インターフェイスの使用に関する簡単な 5 分間のアンケートを完了するよう招待される場合があります。アンケート完了後、**カスタマーエクスペリエンス向上プログラム**を通じて収集されたすべての使用状況データは、Flexera Software LLC の部門である Revenera へ安全に送信され、処理されます。

参加することの利点

事前同意することで、Revenera を次のように支援することができます。

- 一般的なクラッシュシナリオとパフォーマンスのボトルネックを特定することで、InstallShield の安定性を強化します。
- さまざまな環境で InstallShield の機能とワークフローがどのように使用されているかを理解することで、使いやすさを向上させます。

- 集約されたテレメトリ データにより、Revenera チームは問題を事前に検出して対処できるため、より優れたサポート性を確保します。

参加することで、InstallShield を実際の使用状況とお客様のニーズに基づいて進化させることができます。**カスタマーエクスペリエンス向上**プログラムへの参加は、次のオプションから設定できます:

テーブル 10 • [品質] タブのオプション

設定	説明
はい、参加します（推奨）	**カスタマーエクスペリエンス向上**プログラムに参加する場合は、このオプションを選択します。デフォルトで、このオプションが選択されています。
いいえ、参加しません。	**カスタマーエクスペリエンス向上**プログラムに参加しない場合は、このオプションを選択します。

重要• *ユーザーは、**[ツール]** > **[オプション]** > **[品質]**からいつでも**カスタマーエクスペリエンス向上**プログラムへの参加を変更できます。選択したオプション（上記）を有効にするには、InstallShield IDEを再起動する必要があります。*

[.NET] タブ

[オプション] ダイアログ ボックスの **[.NET]** タブでは、.NET プロジェクトの環境設定を行います。また、ここでは .NET Framework と共に含まれるユーティリティ Regasm.exe および InstallUtilLib.dll ファイルの場所も指定します。これらのユーティリティは COM interop と .NET カスタム アクションで利用されます。

テーブル 11 • [.NET] タブの設定

設定	プロジェクトの種類	説明
コンポーネント設定ビルド時のデフォルトの .NET スキャン	基本の MSI、InstallScript MSI	このリストでは、"ビルド時に .NET をスキャン" 設定を新しいコンポーネントにどのように設定するかを選択します。これは、ファイルをプロジェクトに追加したときに 自動的に作成されるコンポーネントに適用されます。

テーブル 11 • [.NET] タブの設定 (続き.)

設定	プロジェクトの種類	説明
.NET Framework ファイルの場所	基本の MSI、InstallScript MSI	Regasm.exe と InstallUtilLib.dll は .NET Framework の各バージョンに含まれるユーティリティです。.NET インストーラー クラスおよび COM Interop を含むリリースのビルド時に使用する、これらのファイルのバージョンを含むディレクトリのパスを指定します。 • **32 ビットの場所** — Regasm.exe および InstallUtilLib.dll のパスを入力またはその場所を参照します。 • **64 ビットの場所** — InstallShield を 32 ビット システム上で使用している場合、このオプションは無効です。InstallShield を 64 ビット システムで使用している場合、Regasm.exe および InstallUtilLib.dll の場所を参照するか、そのパスを入力します。 ***メモ:*** *これらのオプションを構成するためには、InstallShield を管理者権限で実行する必要があります。管理者権限を持っていない場合、これらのオプションは無効です。詳細については、「InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い」を参照してください。*

[ファイル ビュー] タブ

[オプション] ダイアログ ボックスの **[ファイル ビュー]** タブで、InstallShield のビューのいくつかの領域でどの列を表示するかなど、ユーザー設定を指定することができます。

このタブで変更を行うとき、プロジェクトを一度閉じてから開くと変更点が有効になります。

テーブル 12 • [ファイル ビュー] タブの設定

設定	プロジェクトの種類	説明
サイズ	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	InstallShield の次の領域に [サイズ] 列を含めるには、このチェックボックスを選択します。 • [ファイルとフォルダー] ビュー • [コンポーネント] ビューの [ファイル] サブビュー • プロジェクト アシスタントの [アプリケーション ファイル] ページ

テーブル 12 • [ファイル ビュー] タブの設定 (続き.)

設定	プロジェクトの種類	説明
リンク先	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	InstallShield の次の領域に [リンク先] 列を含めるには、このチェックボックスを選択します。 • [ファイルとフォルダー] ビュー • [コンポーネント] ビューの [ファイル] サブビュー • プロジェクト アシスタントの [アプリケーション ファイル] ページ
更新日時	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	InstallShield の次の領域に [変更日] 列を含めるには、このチェックボックスを選択します。 • [ファイルとフォルダー] ビュー • [コンポーネント] ビューの [ファイル] サブビュー • プロジェクト アシスタントの [アプリケーション ファイル] ページ
Destination	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	プロジェクト アシスタントの [アプリケーション ファイル] ページに [インストール先] 列を含めるには、このチェックボックスを選択します。
バージョン	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	InstallShield の次の領域に [バージョン] 列を含めるには、このチェックボックスを選択します。 • [ファイルとフォルダー] ビュー • [コンポーネント] ビューの [ファイル] サブビュー ***重要:*** *[バージョン] チェック ボックスを選択すると、前述のビューのパフォーマンスが低下します。*
ファイル キー	基本の MSI、DIM、InstallScript MSI, マージ モジュール	[コンポーネント] ビューの [ファイル] サブビューに [キー] 列を含めるには、このチェックボックスを選択します。

テーブル 12 • [ファイル ビュー] タブの設定 (続き.)

設定	プロジェクトの種類	説明
表示するファイルの最大数	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	InstallShield の次の領域でダイナミック ファイル リンクに表示するファイルの最大数を指定します。 • [ファイルとフォルダー] ビュー • [コンポーネント] ビューの [ファイル] サブビュー • プロジェクト アシスタントの [アプリケーション ファイル] ページ デフォルト値は 1000 です。ダイナミック ファイル リンクに指定された数以上のファイルが含まれている場合、ファイル リストの先頭項目に ***リストが切り捨てられました*** と表示されます。
未使用のコンポーネントをクリーンアップする	基本の MSI、DIM、InstallScript、InstallScript MSI, マージ モジュール	使用されていないコンポーネントをプロジェクトから自動的に削除する場合は、このチェック ボックスを選択します。 このチェックボックスが選択されているときに、コンポーネントのすべてのファイルを削除して、そのコンポーネントが別の領域で必要でない場合、そのコンポーネントは自動的に削除されます。 コンポーネントが引き続き必要で、プロジェクトから自動的に削除しない状況の例: • コンポーネントが [ファイルとフォルダー] ビューの **[インストール先コンピューターのフォルダー]** ペインで選択されている。 • コンポーネントが削除されると、[ファイルとフォルダー] ビューでフォルダーは表示されません。 • 製品の別の領域でアイテムにコンポーネントが選択されている (例、[XML ファイルの変更] ビューで構成された XML ファイルの変更)。 • InstallScript プロジェクトで、[アプリケーション ターゲット フォルダー] の下の DefaultComponent が自動的に削除されることはありません。

[ファイルの拡張子] タブ

プロジェクト・[ファイルの拡張子] *タブは、次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*

[オプション] ダイアログ ボックスの **[ファイルの拡張子]** タブでは、[ポータブル実行可能ファイル] (PE ファイル) のプリファレンスを指定しますInstallShield は、いくつかの異なるケースでこの PE ファイル一覧を参照します。例:

- [ファイルとフォルダー] ビューの **[インストール先コンピューターのフォルダー]** ペインで PE ファイルをフォルダーに追加すると、新しいコンポーネントが作成され、そのコンポーネントのキー ファイルとして設定されます。
- コンポーネントを作成するときにコンポーネント ウィザードのベスト プラクティス オプションを利用すると、PE ファイルについて新しいコンポーネントが作成され、各 PE ファイルがそのコンポーネントのキー ファイルとして設定されます。
- ダイナミック ファイル リンクにコンポーネントを作成するときベスト プラクティス メソッドを使用すると、ビルド時に、ダイナミック リンクがあるフォルダー内にある各 PE ファイルについてコンポーネントが別々に作成されます。各 PE ファイルは、そのコンポーネントのキー ファイルです。

以下は、このタブで提供されている設定です:

テーブル 13 • [ファイルの拡張子] タブの設定

設定	説明
ポータブル実行可能ファイルの拡張子	このボックスには、InstallShield に PE ファイルとして認識させるファイル拡張子を入力します。拡張子をコンマで分けます。以下は、このボックスのデフォルトのエントリです: **EXE,DLL,OCX,VXD,CHM,HLP,TLB, AX**

[前提条件] タブ

[オプション] ダイアログ ボックスの **[前提条件]** タブで、InstallShield 前提条件に関するプリファレンスを設定することができます。

テーブル 14 · [前提条件] タブの設定

設定	プロジェクトの種類	説明
前提条件ファイルの場所 (現在のユーザー) と前提条件ファイルの場所 (すべてのユーザー)	基本の MSI、InstallScript、InstallScript MSI	InstallShield 前提条件ファイル (.prq ファイル) を格納するフォルダのパスを指定します。 複数の場所を指定する場合、次の例のように、各パスをカンマで区切ります: C:¥Prerequisites,C:¥My Files¥Prerequisites 以下の例のように、パスにはパス変数を使用できます: <ISProductFolder>¥SetupPrerequisites,<ISProjectFolder>¥MyCustomPrerequisites All Users オプションは、ユーザー設定を簡単に更新できないシステム アカウントでコマンドラインのビルドを実行している場合に使用できます。 InstallShield では、InstallShield 前提条件ファイル ファイルを含むフォルダーを指定するその他の方法も提供されています。詳細については、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。 ***メモ·** [すべてのユーザー] オプションを構成するためには、InstallShield を管理者権限で実行する必要があります。管理者権限を持っていない場合、このオプションは無効です。詳細については、「InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い」を参照してください。*

[ソース管理] タブ

[オプション] ダイアログ ボックスの **[ソース管理]** にある設定を構成して、InstallShield とソースコード管理プログラムとの対話を制御します。このタブの設定はすべて、現在のユーザーがこのシステムで開いているプロジェクトすべてに適用されます。

***プロジェクト·** 次のオプションは、例外指定がない限り Windows Installer ベースのプロジェクトと InstallScript ベースのプロジェクトの両方に適用されます。*

- ***チェックアウトにダイアログを使用する** — このチェック ボックスが選択されている場合、InstallShield を使ってプロジェクトをチェックアウトすると、チェックアウト ダイアログ ボックスが表示されます。このチェック ボックスがクリアになっている場合、**[編集時にプロジェクトをチェックアウト]** チェック ボック*

スを選択したとき、または [プロジェクト] メニューで [ソース管理] をポイントしてから [チェックアウト] をクリックした場合、InstallShield がコメント無しでファイルを自動的にチェックアウトします。

- ***プロジェクトを開くときに最新バージョンを取得する*** *– このチェック ボックスは、InstallShieldでプロジェクトを開くときにソース管理プログラムからプロジェクトの最新バージョンをチェック アウトする場合に選択します。最新バージョンが取得できない場合は、作業ディレクトリの旧バージョンに編集を加えることもできます。その場合、他の誰かがソース管理にあるプロジェクト ファイルに暫定的な変更をしていないか確認してください。*
- ***編集時にプロジェクトをチェック アウトする*** *– このチェック ボックスを選択すると、InstallShieldでプロジェクトファイルを変更する時はいつでも、ソース管理プログラムからプロジェクトファイルが自動的にチェック アウトされます。プロジェクトファイルがチェック アウトできない場合は、読み取り専用であることが多く、その場合、プロジェクト ファイルをチェック アウトするまで、変更を上書き保存できません。*
- ***ソース管理に新規プロジェクトを追加する*** *– このチェック ボックスは、プロジェクトを InstallShield で作成したときに自動的にソース管理に 新規プロジェクトを追加する場合に選択します。*
- ***新しい Windows Installer ベースのプロジェクトを作成するときに XML 形式を使用する****–このチェック ボックスを選択すると、[新規プロジェクト] ダイアログ ボックスを使って作成した Windows Installer プロジェクトのデフォルト ファイル形式に XML が使用されます。このチェック ボックスが選択解除された場合、Windows Installer プロジェクトのデフォルトのファイル形式はバイナリとなります。プロジェクトのファイル形式は、****[一般情報]*** *ビューの "プロジェクト ファイル形式" 設定の値を変更して、いつでも変更することができます。*

メモ: *プロジェクト ファイル フォーマットを XML またはバイナリ形式に変換しても、プロジェクト ファイルの拡張子は .ism のまま変更されません。*

プロジェクト: ***[新規 Windows Installer ベースのプロジェクトを作成する際に XML フォーマットを利用する]*** *オプションは、Windows Installer ベースのプロジェクトにのみ適用します。*

ユーザー名

使用するユーザー名を指定すると、ソース管理プロジェクトにログオンできます。

アドバンスト UI

このボタンをクリックし、ソース管理プログラムのプロパティダイアログ ボックスを表示して、詳細を設定することができます。

[ディレクトリ] タブ

[オプション] ダイアログ ボックスの **[ディレクトリ]** タブでは、Windows Installer および InstallScript MSI プロジェクトの様々なデザイン時のディレクトリ オプションを設定することができます。

テーブル 15 • [ディレクトリ] タブの設定

設定	プロジェクトの種類	説明
INSTALLDIR を表示する	基本の MSI、InstallScript MSI	このチェック ボックスを選択すると、定義済みのフォルダーとして INSTALLDIR がフォルダー階層の最上層に表示されます。チェック ボックスが選択解除されていない場合、INSTALLDIR はその他のフォルダーに相対した場所に表示されます。[ファイルとフォルダー] ビューでは、ProgramFilesFolder と [会社名] フォルダーの下にあるサブフォルダーとして表示されます。
異なる ID を個別に持つディレクトリノードを常に表示する	基本の MSI、InstallScript MSI	[ファイルとフォルダー] ビューで、同じパスを持つ 2 つのディレクトリ ID は、デフォルトで 1 つのディレクトリとして表示されます。同じパスを持つ異なるディレクトリ ID を 2 つの異なるディレクトリとして ファイルとフォルダー ビューに表示するにはこのチェック ボックスを選択します。
未使用のディレクトリをクリーンアップする	基本の MSI、InstallScript MSI	このチェック ボックスを選択すると、インストール プロジェクトから未使用のユーザー定義ディレクトリが削除されます。たとえば、コンポーネントのインストール先を [MYDIR][INSTALLDIR]MYDIR1 に設定してから、[INSTALLDIR] に変更した場合、MYDIR1 がプロジェクトの他の場所で使用されていなければ、このディレクトリはプロジェクトから自動的に削除されます。

[リソース] タブ

[オプション] ダイアログ ボックスの **[リソース]** タブで、システム上のリソースコンパイラおよびリソース リンカ プログラムの場所を指定することができます。これらのプログラムは、InstallScript MSI インストール プロジェクトで修正されたダイアログを表示するのに必要です。

これらのフィールドには、各プログラムのデフォルトのファイル保存場所とコマンドラインオプションが自動的に入力されますが、必要であれば編集することもできます。

テーブル 16 • [リソース] タブの設定

設定	プロジェクトの種類	説明
リソース コンパイラの設定	基本の MSI、InstallScript、InstallScript MSI	このセクションでは、リソースコンパイラの場所を指定して、コマンドラインオプションを指定できます。 • **リソース コンパイラの場所** – デフォルトのリソース コンパイラ (rc.exe) の場所が、このフィールドに表示されます。リソースコンパイラの場所を編集するには、現在システムにインストールされているリソースコンパイラの場所を入力または参照します。 • **リソース コンパイラのコマンドライン オプション** – リソース コンパイラのコマンドライン オプションを編集するには、フィールドの既存のコマンドライン オプションの上に新しいコマンドライン オプションを入力します。デフォルトのコマンドラインオプション文字列は /r "%1" です。
リソース リンカの設定	基本の MSI、InstallScript、InstallScript MSI	このセクションでは、リソースリンカの場所を指定して、コマンドラインオプションを指定できます。 • **リソース リンカの場所** – デフォルトのリソース リンカ (link.exe) の場所が、このフィールドに表示されます。リソースリンカの場所を編集するには、現在システムにインストールされているリソースリンカの場所を入力または参照します。 • **リソース リンカのコマンドライン オプション** – リソース リンカのコマンドライン オプションを編集するには、フィールドの既存のコマンドライン オプションの上に新しいコマンドライン オプションを入力します。デフォルトのコマンドラインオプション文字列は "%1" /DLL /NOENTRY /NODEFAULTLIB /MACHINE:iX86 /OUT:"%2" です。

[検証] タブ

[オプション] ダイアログ ボックスの **[検証]** タブでは、ビルドが成功した後に InstallShield で実行する検証の種類を指定します。

テーブル 17 • [検証] タブの設定

設定	プロジェクトの種類	説明
パッチおよびアップグレード検証の実行する	基本の MSI、InstallScript MSI	パッチとアップグレードの両方について検証を実行する場合, このチェック ボックスを選択します。

テーブル 17 • [検証] タブの設定 (続き.)

設定	プロジェクトの種類	説明
次を使用して検証を実行する:	基本の MSI、InstallScript MSI	インストール プロジェクトをビルドしたあと毎回検証を実行する場合、このチェック ボックスを選択して、使用する検証スイートのチェック ボックスを選択します (複数可)。新しい検証モジュール (.cub ファイル) を参照することもできます。
次を使用してマージモジュールを検証する:	マージ モジュール	マージ モジュールをビルドしたあと毎回検証を実行する場合、このチェック ボックスを選択して、使用する検証スイートのチェック ボックスを選択します (複数可)。新しい検証モジュール (.cub ファイル) を参照することもできます。

特定の検証スイートでどの内部整合性評価プログラム (ICE) を使用するのかを指定するには、[カスタマイズ] ボタンをクリックします。詳細については、「検証中に実行する ICE、ISICE、ISVICE および ISBP を指定する」を参照してください。

[リポジトリ] タブ

***エディション:** [リポジトリ] タブにあるボックスは、InstallShield Premier Edition でのみ利用可能です。*

ネットワーク リポジトリの設定、またはその場所、名前、そして説明の変更には、**[オプション]** ダイアログ ボックスにある **[リポジトリ]** タブを使用します。

テーブル 18 • [リポジトリ] タブの設定

設定	説明
ネットワークリポジトリ XML ファイル	これは、InstallShield がネットワークリポジトリ用に作成する .xml ファイルの名前とパスです。InstallShield はまた、.xml ファイルと同じディレクトリに自動的にサブフォルダーも作成します。これらのサブフォルダーはネットワークリポジトリに公開される項目を含みます。
名前	これがネットワークリポジトリの名前です。
説明	これがネットワークリポジトリの説明です。

[SQL スクリプト] タブ

***プロジェクト:** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

[オプション] ダイアログ ボックスの **[SQL スクリプト]** タブで、SQL スクリプト サポートについてオプションを 1 つ設定できます。

テーブル 19 • [SQL スクリプト] タブの設定

設定	説明
新しい接続に一意の Windows Installer プロパティを生成する	プロジェクトに追加したすべての新しいデータベース接続で Windows Installer プロパティを共有する場合、このチェック ボックスをクリアします。 追加した新しい接続で異なる Windows Installer プロパティを使用する場合、このチェック ボックスを選択します。 このチェック ボックスはデフォルトでクリアになっています。 詳細については、「新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する」を参照してください。

[マージ モジュール] タブ

[オプション] ダイアログの **[マージ モジュール]** タブは、マージ モジュールおよびマージ モジュール プロジェクトの環境設定を行います。

テーブル 20 - [マージ モジュール] タブの設定

設定	プロジェクトの種類	説明
マージ モジュールの場所 (現在のユーザー) とマージ モジュールの場所 (All Users)	基本の MSI、InstallScript、InstallScript MSI	マージ モジュール (.msm ファイル) を格納する場所へのパスを入力します。パスを追加するには、次の例のようにカンマで区切ります。 C:¥MergeModules,C:¥My Files¥MergeModules 以下の例のように、パスにはパス変数を使用できます: <ISProductFolder>¥Modules¥i386,<ISProjectFolder>¥MyCustomModules パスには環境変数を使用できます。 All Users オプションは、ユーザー設定を簡単に更新できないシステム アカウントでコマンドラインのビルドを実行している場合に使用できます。 リストした最初のパスは、マージ モジュールをビルドした後にそのコピーを格納する場所です。ファイルは、リリース ウィザードの [マージ モジュール オプション] パネルで **[Modules フォルダーにコピー]** が選択されている場合、[リリース] ビューの [イベント] タブにある "マージ モジュールのパブリッシュ" 設定で **[Merge Modules フォルダーにコピー]** オプションが選択されている場合、または ISCmdBld.exe が -e コマンドライン オプションで実行された場合のみコピーされます。このフォルダーが存在しない場合は、InstallShield によって作成されます。 InstallShield では、マージ モジュールを含むフォルダーを指定するその他の方法も提供されています。詳細については、「マージ モジュールを含むディレクトリを指定する」を参照してください。 ***メモ:*** *[すべてのユーザー] オプションを構成するためには、InstallShield を管理者権限で実行する必要があります。管理者権限を持っていない場合、このオプションは無効です。詳細については、「InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い」を参照してください。*

テーブル 20 • [マージ モジュール] タブの設定 (続き.)

設定	プロジェクトの種類	説明
マージモジュールのファイル検索の動作	基本の MSI、InstallScript MSI	ベスト プラクティス コンポーネント ウィザードまたは (ベスト プラクティスをアクティブにして) [ファイル] ビューを使用してファイルをプロジェクトに追加する際、InstallShield はマージ モジュールを検索し、同じファイルを含むモジュールがないかどうかを確認します。一致するファイルがあった場合はそれを通知します。 このセクションで、InstallShield がファイルふぁ一致するかどうか判別するために使用するオプションを選択することができます。

その他の IIS プロパティ ダイアログボックス

[IIS 構成] ビューで Web サイト、アプリケーション、仮想ディレクトリを選択すると、多くの設定が含まれるグリッドが表示されます。[その他の IIS プロパティ] ダイアログ ボックスは、グリッド内の [詳細設定] 領域にある "その他の IIS プロパティ" 設定で省略記号ボタン (...) をクリックすると開きます。このダイアログ ボックスでは、このビューの他の領域では表示されない IIS 設定の値を指定できます。

プロパティの値をカスタマイズするには、プロパティ名をクリックしてから、[値の変更] ボタンをクリックします。[ISIISMetaData 値] ダイアログ ボックス が開き、ここで必要に応じて値を設定できます。

メモ: *その他の IIS プロパティの設定は IIS 6 以前に適用します。IIS 7 は、これらの設定を無視します。*

特定の設定のヘルプについては、MSDN Web サイトの「メタベース プロパティ リファレンス」を参照してください。

[その他のウィンドウ スタイル] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

[その他のウィンドウ スタイル] ダイアログ ボックスには、[ダイアログ] ビューにあるダイアログまたはコントロール用のその他のウィンドウ スタイル オプションが提供されています。このダイアログ ボックスは、"その他のウィンドウ スタイル" 設定で省略記号ボタン (...) をクリックすると表示されます。"その他のウィンドウ スタイル" 設定は、[ダイアログ] ビューでダイアログまたは特定の種類のコントロールをクリックすると表示されます。

使用可能なオプションは、[その他のウィンドウ スタイル] ダイアログ ボックスを開いたときにアクティブなダイアログ コントロールの種類によって異なります。

- ダイアログ
- ビットマップ、アイコン、およびテキスト領域コントロール
- ボタン コントロール (チェック ボックス、ラジオ ボタン グループ、プッシュ ボタン、およびグループ ボックス)
- チェック ボックス コントロール
- リスト ビュー コントロール
- 編集フィールドとリスト ボックス コントロール
- 行コントロール
- [進行状況バー] コントロール
- [選択ツリー] コントロール

この値の詳細については MSDN Library を参照してください。

ダイアログのその他のウィンドウ スタイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ダイアログでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 21 ・ ダイアログのウィンドウスタイル

値	説明
WS_POPUP	ポップアップ ウィンドウを作成します。
DS_NOIDLEMSG	ダイアログ ボックスが表示される時、システムはダイアログの所有者に WM_ENTERIDLE メッセージを送信しますが、このメッセージの送信を抑制します。
DS_CONTROL	他のダイアログ ボックスの子ウィンドウとして動くダイアログ ボックスを作成します。子ダイアログ ボックスのコントロールウィンドウ間のタブ切り換えをユーザーに許可し、ユーザーがアクセラレータキーを使用できるようにします。
DS_SYSMODAL	このスタイルは 16 ビット版 Windows との互換性を取るために用意されているだけで、通常は使用しません。このスタイルを指定すると、WS_EX_TOPMOST スタイルのダイアログが作成されます。 このスタイルは、ユーザーによるデスクトップ上の他のウィンドウへのアクセスを妨げません。DS_CONTROL スタイルとは併用しないでください。

テーブル 21・ダイアログのウィンドウスタイル（続き）

値	説明
DS_3DLOOK	ダイアログ ボックス中のテキストをフォントに太字をかけず表示し、ダイアログ ボックス内のコントロールウィンドウの周囲に 3 次元輪郭線を描画します。
DS_CENTER	ダイアログ ボックスを、エンド ユーザーの画面の作業領域の中央に表示します。
DS_LOCALEDIT	ダイアログ ボックス中の編集コントロールに対して、アプリケーションのデータセクション内にあるメモリを使用させます。デフォルトでは、ダイアログ ボックス中のすべての編集コントロールは、アプリケーションのデータセクション外のメモリを使用します。
DS_FIXEDSYS	ダイアログ ボックスに対して、デフォルトの SYSTEM_FONT の代わりに、SYSTEM_FIXED_FONT を使用させます。これはモノスペース フォントで、Windows 3.0 以前の 16 ビットのオペレーティング システムで使われていたシステム フォントと互換性があります。
DS_CENTERMOUSE	カーソルをダイアログ ボックスの中央に配置します。
DS_SETFONT	ダイアログ ボックステンプレートのヘッダーに追加データを記入して、ダイアログ ボックスのクライアント領域とコントロールで使用するテキストを指定します。フォントデータは、タイトル配列の WORD 境界から始めます。ここには、16 ビットのポイントサイズ値と、Unicode フォント名の文字列を指定します。 このスタイルが指定されない場合、ダイアログ ボックステンプレートにフォントデータは含まれません。
DS_ABSALIGN	ダイアログ ボックスの座標として、画面の座標を使用します。
DS_CONTEXTHELP	ダイアログ ボックスのタイトルバーに疑問符を表示させます。エンドユーザーがこの疑問符をクリックすると、カーソルがポインター付きの疑問符に変わります。その状態でダイアログ ボックス内のコントロールをクリックすると、このコントロールは WM_HELP メッセージを受け取ります。このメッセージは、コントロールからダイアログ ボックスプロシージャに渡されます。ヘルプ アプリケーションが、コントロールに関するヘルプなどが納められたポップアップ ウィンドウを表示します。 このDS_CONTEXTHELP は単なるプレースホルダーです。ダイアログ ボックスが作成されると、システムは DS_CONTEXTHELP をチェックして、WS_EX_CONTEXTHELP があれば、これをもとにダイアログ ボックスの拡張スタイルを追加します。この WS_EX_CONTEXTHELP は、WS_MAXIMIZEBOX または WS_MINIMIZEBOX スタイルと併用はできません。
DS_MODALFRAME	ダイアログ ボックスにモーダルダイアログ ボックスフレームを付けて作成します。このフレームは、WS_CAPTION および WS_SYSMENU を指定することで、タイトルバーおよびウィンドウメニューと組み合わせることができます。

テーブル 21・ダイアログのウィンドウスタイル (続き.)

値	説明
DS_NOFAILCREATE	たとえば、子ウィンドウが作成できない場合や、編集コントロール用の特別なデータセグメントが作成できない場合など、エラーが発生してもダイアログの作成を続行させます。
DS_SETFOREGROUND	システムが **SetForegroundWindow** 関数を使用し、ダイアログ ボックスが手前に表示されます。

ビットマップ、アイコン、およびテキスト領域コントロールのその他のウィンドウ スタイル

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ビットマップ、アイコン、およびテキスト領域コントロールでは、以下のウィンドウ スタイル オプションが使用できます: 詳細については、MSDN ライブラリを参照してください。

テーブル 22・ビットマップ、アイコン、およびテキスト領域コントロールのウィンドウ スタイル オプション

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
SS_CENTERIMAGE	これを指定すると、スタティックコントロールのクライアント領域よりも小さいビットマップやアイコンの場合、クライアント領域の残りの部分を、ビットマップやアイコンの左上端のピクセル色で塗りつぶします。またスタティックコントロールに含まれるのが 1 行のテキストの場合、このテキストは、コントロールのクライアント領域の縦方向の中央に表示されます。
SS_NOTIFY	コントロールがユーザーからのクリックやダブルクリックを受けた時に、親ウィンドウに対して STN_CLICKED、STN_DBLCLK、STN_DISABLE および STN_ENABLE 通知メッセージを送信させます。
SS_RIGHTJUST	SS_BITMAP または SS_ICON スタイルを持つスタティックコントロールに対して、サイズ変更された時にコントロールの右下端を固定するように指定します。新規のビットマップやアイコンに適合するよう調整をする場合、変更できるのは上端および左端だけとなります。
WS_VSCROLL	垂直スクロールバーの付いたウィンドウを作成します。

テーブル 22 • ビットマップ、アイコン、およびテキスト領域コントロールのウィンドウ スタイル オプション (続

値	説明
SS_REALSIZECONTROL	スタティックコントロールのサイズに合うようにビットマップを調整します。ターゲット マシン上の特定のディスプレイ設定によっては、システム フォントや DPI スケールといったイメージの表示に使用されるコントロール領域のサイズに影響が及びます。このスタイルは、ビットマップをコントロールのサイズに合わせて自動調整します。これは、高 DPI 設定のマシンをサポートするために、SS_REALSIZEIMAGE の代わりに推奨されます。これを使用しなかった場合、一部のマシン上で実行時にイメージの端が切り取られたり、コントロール全体に表示されない場合があります。
SS_REALSIZEIMAGE	スタティックなアイコンおよびビットマップコントロール (つまり SS_ICON または SS_BITMAP スタイルのスタティックコントロール) に対して、ロード時や描画時のサイズ変更ができないようにします。配置される領域よりもアイコンやビットマップが大きい場合は、イメージがクリップされます。
SS_ENDELLIPSIS	文字列の末尾が枠内に収まり切らない場合、はみ出す部分を切り詰めて、省略記号を表示します。枠にかかる単語で行末ではないものは、省略記号を付けずに切り詰められます。SS_PATHELLIPSIS および SS_WORDELLIPSIS との違いに注意してください。 ***メモ:*** *SS_ENDELLIPSIS、SS_PATHELLIPSIS および SS_WORDELLIPSIS は、いずれも同時に指定することはできません。*
SS_PATHELLIPSIS	文字列の中央部を省略記号に置き換えて、行末が枠内に納まるよう調整します。円記号 (¥) 記号を含む文字列の場合、SS_PATHELLIPSIS は最後の円記号以降の文字をできるだけ多く表示するようにします。SS_ENDELLIPSIS および SS_WORDELLIPSIS との違いに注意してください。 ***メモ:*** *SS_ENDELLIPSIS、SS_PATHELLIPSIS および SS_WORDELLIPSIS は、いずれも同時に指定することはできません。*
SS_WORDELLIPSIS	枠内に収まらない単語はすべて切り詰めて、省略記号で置き換えます。SS_ENDELLIPSIS および SS_PATHELLIPSIS との違いに注意してください。 ***メモ:*** *SS_ENDELLIPSIS、SS_PATHELLIPSIS および SS_WORDELLIPSIS は、いずれも同時に指定することはできません。*

ボタンコントロールのその他のウィンドウスタイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ボタン コントロール (チェック ボックス、ラジオ ボタン グループ、プッシュ ボタン、およびグループ ボックス コントロール) では、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 23 ・ ボタンコントロールのウィンドウスタイル

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
BS_LEFTTEXT	ラジオボタンまたはチェック ボックスのスタイルと併用した場合、これらのコントロールの左側にテキストを配置します。
BS_CENTER	ボタン枠内の横方向の中央にテキストを配置します。
BS_BOTTOM	ボタン枠の下端にテキストを配置します。
WS_VSCROLL	垂直スクロールバーの付いたウィンドウを作成します。
BS_VCENTER	ボタン枠の縦方向の中央にテキストを配置します。
BS_PUSHLIKE	チェック ボックス、三段階チェック ボックス、ラジオボタンのような、プッシュボタンの外観と機能を持つボタンを作成します。このボタンは、押されていない選択状態では、浮き上がっているよう表示され、押されるとチェック状態となり、押し込まれたように表示されます。
BS_FLAT	ボタンを 2 次元表示にします。この場合、デフォルトで 3 次元イメージに付けられるシェードはかけられません。
BS_NOTIFY	ボタンから親ウィンドウに対して BN_KILLFOCUS および BN_SETFOCUS 通知メッセージを送信させます。 このスタイルが適用されているかにかかわらず、ボタンは BN_CLICKED 通知メッセージを送信します。BN_DBLCLK 通知メッセージを送信するには、ボタンに BS_RADIOBUTTON または BS_OWNERDRAW スタイルが適用されている必要があります。
BS_MULTILINE	ボタンのテキスト文字列が長くて、1 行ではボタン枠に収まりきらない場合、テキストを複数の行に折り返します。

プッシュボタンコントロールの、その他のウィンドウスタイル

次の値は、プッシュボタンコントロールにのみ適用されます。この指定を同時に複数適用することはできません。

テーブル 24 - プッシュボタンコントロールのウィンドウスタイル

値	説明
BS_OWNERDRAW	所有者描画ボタンを作成します。ボタンの外見が変更されると、所有者ウィンドウが WM_DRAWITEM メッセージを受信します。この BS_OWNERDRAW スタイルは、他のボタンスタイルと組み合わせて使用しないでください。
BS_USERBUTTON	16 ビット版 Windows との互換性を取るために用意されているだけで、通常は使用しません。Win32対応ソフトウェアでは BS_OWNERDRAW を使用してください。

チェック ボックスコントロールの、その他のウィンドウスタイル

次の値は、チェック ボックスコントロールにのみ適用されます。この指定を同時に複数適用することはできません。

テーブル 25 - チェック ボックスコントロールのウィンドウスタイル

値	説明
BS_AUTOCHECKBOX	ボタンをチェック ボックスにしますが、通常のものと違い、チェック ボックスが選択されると、自動的にチェック状態と解除状態が交互に切り替えられます。
BS_3STATE	ボタンをチェック ボックスにしますが、通常のものと違い、チェック状態と解除状態に加えて、灰色表示もされるようになります。この灰色表示は、チェック ボックスの選択が未決定であることを示すのに使えます。
BS_AUTO3STATE	ボタンを三段階チェック ボックスにしますが、通常のものと違い、選択されるとボタンの状態が変化します。ボタンの状態は、チェック、灰色表示、解除が交互に切り替わります。

コンボ ボックスコントロールのその他のウィンドウ スタイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

コンボ ボックス コントロールでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 26 • コンボ ボックスコントロールのウィンドウ スタイル

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
WS_VSCROLL	垂直スクロールバーの付いたウィンドウを作成します。
CBS_SIMPLE	常にリストボックスを表示させます。リストボックス中の選択項目は、編集コントロールに表示されます。
CBS_DROPDOWN	これは CBS_SIMPLE と同様ですが、編集コントロール横のアイコンをユーザーが選択するまで、リストボックスは表示されません。
CBS_OEMCONVERT	コンボボックスの編集コントロールに入力されたテキストを、Windows キャラクタセットから、OEM キャラクタセットに変換し、その後 Windows のセットに戻します。これにより、アプリケーションが **CharToOem** 関数を呼び出して、コンボボックス内の Windows 文字列を OEM 文字に変換させるときに、適切な変換を行うことができます。このスタイルは、データとしてファイル名を持つコンボボックスに対して使うと有用で、 CBS_SIMPLE または CBS_DROPDOWN スタイルのコンボボックスにのみ適用されます。
CBS_HASSTRINGS	オーナ描画コンボボックスのデータ項目が、文字列から構成されていることを指定します。コンボボックスは、これらの文字列に対するメモリとアドレスを保持するので、アプリケーションは CB_GETLBTEXT メッセージを使用して、特定の項目のテキストを取り出すことが可能となります。
CBS_UPPERCASE	選択フィールドおよびリスト中のすべてのテキストを、大文字に変換します。
CBS_AUTOHSCROLL	ユーザーの行末文字の入力に合わせて、編集コントロール中のテキストを自動的に右にスクロールさせます。このスタイルを指定しない場合、フィールド枠に収まる分のテキストしか入力できません。
CBS_DISABLENOSCROLL	リストボックスの項目数が少なくスクロールが不要の場合、垂直スクロールバーを無効にして表示します。このスタイルを指定しないと、リストボックスの項目数が少ない場合、スクロールバーが隠された状態で表示されます。
CBS_NOINTEGRALHEIGHT	コンボボックスのサイズを、コンボボックス作成時にアプリケーションが指定したサイズと正確に一致させます。通常は、項目の一部しか表示されることがないように、システムがコンボボックスをサイズ変更します。

テーブル 26 • コンボ ボックスコントロールのウィンドウ スタイル (続き.)

値	説明
CBS_OWNERDRAWFIXED	リストボックスの所有者が内容の描画の責任を負うようにし、またリストボックス中の項目は、すべて同じ高さであることを指定します。所有者ウィンドウは、コンボボックスの作成時に WM_MEASUREITEM メッセージを受信し、コンボボックスの外見が変更された場合は WM_DRAWITEM メッセージを受信します。
CBS_OWNERDRAWVARIABLE	リストボックスのオーナが内容の描画の責任を負うようにし、またリストボックス中の項目は、高さが一定でないことを指定します。所有者ウィンドウは、コンボボックスの作成時に、コンボボックス内の各項目に対する WM_MEASUREITEM メッセージを受信し、コンボボックスの外見が変更された場合は WM_DRAWITEM メッセージを受信します。

リスト ビュー コントロールのその他のウィンドウ スタイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

リスト ビュー コントロールでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 27 • リスト ビュー コントロールのウィンドウ スタイル

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
LVS_SINGLESEL	一度に選択できる項目を 1 つに制限します。デフォルトでは、複数の項目が選択できるようになっています。
LVS_NOLABELWRAP	アイコンビュー時の項目名を、1 行表示にします。デフォルトでは、アイコンビュー時の項目名は、折り返して表示されます。
LVS_AUTOARRANGE	アイコンビューおよび小アイコンビュー時に、アイコンを自動的に整列させます。
LVS_SHOWSELALWAYS	コントロールがアクティブでなくても、選択項目を常にハイライト表示させます。
LVS_SORTASCENDING	項目名に基づき、項目を昇順で並べ替えます。

テーブル 27・リスト ビュー コントロールのウィンドウ スタイル (続き.)

値	説明
LVS_SORTDESCENDING	項目名に基づき、項目を降順で並べ替えます。
LVS_EDITLABELS	項目名を直接編集できるようにします。この場合、親ウィンドウは LVN_ENDLABELEDIT 通知メッセージを処理する必要があります。
LVS_OWNERDATA	仮想リストビューコントロールを作成します。
LVS_NOSCROLL	スクロールを無効にするので、この場合はすべての項目をクライアント領域内に納める必要があります。
LVS_NOSORTHEADER	列の見出しをボタンとして機能しないようにします。このスタイルは、レポートビューで列の見出しをクリックしても並べ替えなどの動作をさせたくない場合に有用です。
LVS_OWNERDRAWFIXED	所有者ウィンドウに、レポートビューでの項目のペイントを許可します。リストビューコントロールは、ペイントする各項目に対して WM_DRAWITEM メッセージを送信しますが、サブ項目に対して個別にメッセージは送信しません。DRAWITEMSTRUCT 構造の itemData メンバーは、指定されたリストビュー項目のデータを持ちます。
LVS_NOCOLUMNHEADER	デフォルトのビューであるレポートビューで、列の見出しを表示しないようにします。
LVS_SHAREIMAGELISTS	コントロールが破棄される際に、コントロールに割り当てられたイメージ リストが破棄されないようにします。このスタイルを指定することにより、1 つのイメージリストを複数のリストビューで使用することが可能になります。

種類

メモ: 次の値は、同時に複数適用することはできません。

テーブル 28・タイプの値

値	説明
LVS_ICON	アイコンビューを指定。
LVS_REPORT	レポートビューを指定。
LVS_LIST	リストビューを指定。
LVS_SMALLICON	小アイコンビューを指定。

配置

メモ: 次の値は、同時に複数適用することはできません。

テーブル 29 • 配置の値

値	説明
LVS_ALIGNTOP	アイコンビューおよび小アイコンビューで、リストビューコントロールの上側に項目を整列させます。
LVS_ALIGNLEFT	アイコンビューおよび小アイコンビューで、項目を左側に整列させます。

編集フィールドとリスト ボックス コントロールのその他のウィンドウ スタイル

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

編集フィールドとリスト ボックス コントロールでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 30 • 編集フィールドとリスト ボックス コントロールのウィンドウ スタイル

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
WS_VSCROLL	垂直スクロールバーの付いたウィンドウを作成します。
LBS_STANDARD	縁取りされたリストボックスを作成します。
LBS_NOTIFY	エンドユーザーが文字列をクリックまたはダブルクリックした時、親ウィンドウがインプットメッセージを受信するようにします。

テーブル 30・編集フィールドとリスト ボックス コントロールのウィンドウ スタイル (続き.)

値	説明
LBS_NODATA	ノーデータリストボックスを指定します。リストボックス中の項目のカウントが 1000 以上になる場合、このスタイルを指定してください。ノーデータリストボックスを使う場合、LBS_OWNERDRAWFIXED スタイルも指定する必要がありますが、LBS_SORT および LBS_HASSTRINGS スタイルを指定してはいけません。 ノーデータリストボックスは所有者描画リストボックスに似ていますが、項目として文字列やビットマップデータを持たない点が違います。項目の追加、挿入、削除を行うようなコマンドは、指定されたすべての項目データを常に無視するので、リストボックス中の文字列を検索する要求を出しても、必ず失敗します。項目の描画が必要になると、システムは WM_DRAWITEM メッセージを親ウィンドウに送信します。 WM_DRAWITEM メッセージを渡されたDRAWITEMSTRUCT 構造の itemID メンバーは、描画する項目の行番号を指定します。ノーデータリストボックスは、WM_DELETEITEM メッセージを送信しません。
LBS_NOINTEGRALHEIGHT	リストボックス作成時にアプリケーションが指定したサイズと正確に一致するよう、リストボックスを作成します。
LBS_NOREDRAW	変更を加えられた際に、リストボックスを更新しないようにします。
LBS_MULTIPLESEL	エンドユーザーが文字列をクリックまたはダブルクリックした時、選択文字列を交互に切り換えます。
LBS_HASSTRINGS	リストボックスのデータ項目が、文字列から構成されていることを指定します。リストボックスは、これらの文字列に対するメモリとアドレスを保持するので、アプリケーションは LB_GETTEXT メッセージを使用して特定の項目のテキストを取り出すことが可能となります。デフォルトでは、所有者描画リストボックス以外のすべてのリストボックスにはこのスタイルが適用されます。所有者描画リストボックスを作成する際に、このスタイルの有無は任意に指定できます。
LBS_USETABSTOPS	リストボックスにタブ記号を認識させ、文字列の描画時にタブを効かせるようにします。
LBS_MULTICOLUMN	水平方向にスクロール可能な複数列リストボックスを作成します。
LBS_EXTENDEDSEL	SHIFT キーとマウス (またはキー操作) の組み合わせによる、複数項目の選択をエンドユーザーに許可します。
LBS_DISABLENOSCROLL	リストボックスの項目数が少なくスクロールが不要の場合、垂直スクロールバーを無効にして表示します。

テーブル 30・編集フィールドとリスト ボックス コントロールのウィンドウ スタイル (続き.)

値	説明
LBS_WANTKEYBOARDINPUT	リストボックスにフォーカスを持つ状態でエンドユーザーがキーを押した時に、WM_VKEYTOITEM または WM_CHARTOITEM メッセージを渡します。これを利用して、キーボードからのインプットに応じた特別な処理をアプリケーションにさせることができます。
LBS_OWNERDRAWFIXED	リスト ボックスの所有者がリストボックスの内容の描画の責任を負うようにします。ただしこの場合は、リストボックス中の項目はすべて同じ高さになります。
LBS_OWNERDRAWVARIABLE	リスト ボックスの所有者がリストボックスの内容の描画の責任を負うようにします。ただしこの場合は、リストボックス中の項目の高さは一定ではありません。

行コントロールのその他のウィンドウ スタイル

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

行コントロールでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 31・行コントロールのウィンドウ スタイル

値	説明
SS_SIMPLE	通常の四角形の中に、テキストを左寄せで 1 行表示させます。テキスト行を縮めたりするなど、変更はいっさいできません。また、コントロールが使用不可能状態でも、テキストは灰色表示されません。
SS_ETCHEDVERT	スタティック コントロールの左右の境界線を、EDGE_ETCHED エッジ スタイルを使って描画します。詳細については MSDN Library の **DrawEdge Windows** API 関数を参照してください。
SS_ETCHEDFRAME	スタティックコントロールのフレームを、EDGE_ETCHED エッジスタイルを使って描画します。詳細については MSDN Library の「DrawEdge Windows API 関数」を参照してください。
SS_ETCHEDHORZ	スタティックコントロールの上下の境界線を、EDGE_ETCHED エッジスタイルを使って描画します。詳細については MSDN Library の「DrawEdge Windows API 関数」を参照してください。

テーブル 31・行コントロールのウィンドウ スタイル (続き.)

値	説明
SS_BLACKRECT	四角形を、現在のウィンドウのフレーム色で塗りつぶして描画させます。デフォルトの色は黒です。
SS_BLACKFRAME	四角形のフレームを、現在のウィンドウのフレーム色で描画させます。デフォルトの色は黒です。
SS_WHITERECT	四角形を、現在のウィンドウの背景色で塗りつぶして描画させます。デフォルトの色は白です。
SS_WHITEFRAME	四角形のフレームを、現在のウィンドウの背景色で描画させます。デフォルトの色は白です。
SS_GRAYRECT	四角形を、現在の画面の背景色で塗りつぶして描画させます。デフォルトの色は灰色です。
SS_GRAYFRAME	四角形のフレームを、現在の画面 (デスクトップ) の背景色で描画させます。デフォルトの色は灰色です。
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
WS_VSCROLL	垂直スクロールバーの付いたウィンドウを作成します。
SS_NOTIFY	コントロールがユーザーからのクリックやダブルクリックを受けた時に、親ウィンドウに対して STN_CLICKED、STN_DBLCLK、STN_DISABLE および STN_ENABLE 通知メッセージを送信させます。
SS_REALSIZECONTROL	スタティックコントロールのサイズに合うようにビットマップを調整します。ターゲット マシン上の特定のディスプレイ設定によっては、システム フォントや DPI スケールといったイメージの表示に使用されるコントロール領域のサイズに影響が及びます。このスタイルは、ビットマップをコントロールのサイズに合わせて自動調整します。これは、高 DPI 設定のマシンをサポートするために、SS_REALSIZEIMAGE の代わりに推奨されます。これを使用しなかった場合、一部のマシン上で実行時にイメージの端が切り取られたり、コントロール全体に表示されない場合があります。
SS_RIGHTJUST	SS_BITMAP または SS_ICON スタイルを持つスタティックコントロールに対して、サイズ変更された時にコントロールの右下端を固定するように指定します。新規のビットマップやアイコンに適合するよう調整をする場合、変更できるのは上端および左端だけとなります。

テーブル 31・行コントロールのウィンドウ スタイル (続き.)

値	説明
SS_CENTERIMAGE	これを指定すると、スタティックコントロールのクライアント領域よりも小さいビットマップやアイコンの場合、クライアント領域の残りの部分を、ビットマップやアイコンの左上端のピクセル色で塗りつぶします。またスタティックコントロールに含まれるのが 1 行のテキストの場合、このテキストは、コントロールのクライアント領域の縦方向の中央に表示されます。
SS_REALSIZEIMAGE	スタティックなアイコンおよびビットマップコントロール (つまり SS_ICON または SS_BITMAP スタイルのスタティックコントロール) に対して、ロード時や描画時のサイズ変更ができないようにします。配置される領域よりもアイコンやビットマップが大きい場合は、イメージがクリップされます。
SS_ENDELLIPSIS	文字列の末尾が枠内に収まり切らない場合、はみ出す部分を切り詰めて、省略記号を表示します。枠にかかる単語で行末ではないものは、省略記号を付けずに切り詰められます。SS_PATHELLIPSIS および SS_WORDELLIPSIS との違いに注意してください。 ***メモ・*** *SS_ENDELLIPSIS、SS_PATHELLIPSIS および SS_WORDELLIPSIS は、いずれも同時に指定することはできません。*
SS_PATHELLIPSIS	文字列の中央部を省略記号に置き換えて、行末が枠内に納まるよう調整します。円記号 (¥) 記号を含む文字列の場合、SS_PATHELLIPSIS は最後の円記号以降の文字をできるだけ多く表示するようにします。SS_ENDELLIPSIS および SS_WORDELLIPSIS との違いに注意してください。 ***メモ・*** *SS_ENDELLIPSIS、SS_PATHELLIPSIS および SS_WORDELLIPSIS は、いずれも同時に指定することはできません。*
SS_WORDELLIPSIS	枠内に収まらない単語はすべて切り詰めて、省略記号で置き換えます。SS_ENDELLIPSIS および SS_PATHELLIPSIS との違いに注意してください。 ***メモ・*** *SS_ENDELLIPSIS、SS_PATHELLIPSIS および SS_WORDELLIPSIS は、いずれも同時に指定することはできません。*

進行状況バー コントロールのその他のウィンドウ スタイル

プロジェクト・ *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

進行状況バー コントロールでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 32 ・ 進行状況バー コントロールのウィンドウ スタイル

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
PBS_VERTICAL	進行状況バーの進行を、下から上に向かう垂直表示にします

選択ツリー コントロールのその他のウィンドウスタイル

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

選択ツリーコントロールでは、以下のウィンドウ スタイル オプションが使用できます。詳細については、MSDN ライブラリを参照してください。

テーブル 33 ・ 選択ツリーコントロールのウィンドウ スタイル

値	説明
WS_GROUP	コントロールのグループで最初のコントロールを指定します。このスタイルを指定して定義されたコントロールで、最初のコントロールの次にくるものは、すべて同じグループに属します。
TVS_EDITLABELS	ツリービューで項目ラベルの編集を、エンドユーザーに許可します。
TVS_HASBUTTONS	親となる項目に、プラス記号 (+) およびマイナス記号 (−) ボタンを付けて表示します。ユーザーがこのボタンをクリックすると、子アイテムの親アイテム一覧の展開および折りたたみができます。ツリービューのルートの項目をボタンに含めるには、TVS_LINESATROOT も指定してください。
TVS_DISABLEDRAGDROP	ツリービューコントロールの TVN_BEGINDRAG 通知メッセージの送信を抑制します。
TVS_SHOWSELALWAYS	選択項目の描画に、システムのハイライト色を使用させます。
TVS_FULLROWSELECT	ツリービューの全行選択を許可します。選択項目の行全体がハイライト表示され、項目行の任意の場所をクリックするとその行が選択されます。このスタイルは TVS_HASLINES スタイルと併用はできません。

テーブル 33 • 選択ツリーコントロールのウィンドウ スタイル (続き.)

値	説明
TVS_HASLINES	項目の階層構造を線を使って表示します。
TVS_LINESATROOT	ツリービューコントロールのルートにある項目を線で結んで表示します。この指定は、TVS_HASLINES が指定されていない場合、無視されます。
TVS_RTLREADING	テキスト表示を右から左に表示させます (RTL)。Windows の通常の表示では、テキストは左から右に表示されます (LTR)。ただし Windows は、ヘブライ語やアラブ語のような RTL 式表示の言語に対して、表示を反転できるようになっています。通常、ツリービューのテキストは、親ウィンドウのテキスト表示と同じ方向で表示されます。TVS_RTLREADING を指定すると、ツリービューのテキストは、親ウィンドウのテキスト表示の逆方向で表示されるようになります。
TVS_NOTOOLTIPS	ツールヒントを使用不可能にします。
TVS_CHECKBOXES	ツリービューコントロール中の項目を、チェック ボックスとして表示させます。このスタイルは、項目の状態イメージを利用して、チェック ボックスとして使用します。
TVS_TRACKSELECT	ツリービューコントロールでホットトラッキングを使用可能にします。
TVS_SINGLEEXPAND	ツリービュー内での選択操作に応じて、選択された項目を展開し、非選択の項目を折りたたませます。閉じている選択項目をマウスでシングルクリックすると、項目が展開されます。Ctrl キーを押しながら項目を選択すると、非選択項目の折りたたみは行われません。
TVS_INFOTIP	TVN_GETINFOTIP メッセージを送信して、ToolTip 情報を収集します。
TVS_NOSCROLL	コントロールの水平方向スクロールを使用不可能にします。これを指定したコントロールには、水平方向スクロールバーはまったく表示されなくなります。
TVS_NONEVENHEIGHT	TVM_SETITEMHEIGHT メッセージで、項目の高さを奇数値にします。デフォルトでは、項目の高さは偶数値にされています。

[出力] ダイアログ ボックス

[出力] ダイアログ ボックスは、ファイル システム エディターでプロジェクト出力グループについての情報を表示します。このダイアログは、Microsoft Visual Studio で作成された InstallShield プロジェクトで使用できます。

タスク ***[出力] ダイアログにアクセスするには、以下の手順を実行します。***

1. プロジェクト出力グループが**ファイル システム エディター**で選択されたとき、**[プロパティ]** ウィンドウで **"出力"** プロパティを選択します。
2. **[ファイルとフォルダー]** ビューの **[ソース コンピューターのファイル]** ペインでアイテムを右クリックし、**[プロジェクト出力の解決]** を選択します。

3. **[ファイルとフォルダー]** ビューの **[インストール先コンピューターのファイル]** ペインでアイテムを右クリックし、**[プロジェクト出力の解決]** を選択します。

メモ: 複数のプロジェクト出力グループが選択された場合、情報は最初の選択されたグループに対してのみ表示されます。

ダイアログ オプション

ターゲット名

選択されたプロジェクト出力グループのファイル名をターゲット コンピューターで表示されるとおりに表示します。このフィールドは読み取り専用です。

ソース パス

開発コンピューター上でのプロジェクト出力グループ ファイルへのパスを表示します。このフィールドは読み取り専用です。

[上書き] ダイアログ ボックス

このダイアログ ボックスを使うと、アクティブなコンポーネント ([セットアップのデザイン] ビューまたは [コンポーネント] ビューで選択されたコンポーネント) でターゲット システムにある既存のバージョンを常に上書きするか、上書きしないか、または日時スタンプやバージョン番号などに基づいて条件的に上書きするかを指定できます。

ダイアログ オプション

リストボックス

ターゲット システムのファイルを常に上書きするか、上書きしないか、または日時スタンプやバージョン番号に基づいて条件付きで上書きするかを選択します。

次のコントロールは、*リストボックス*で "バージョンでファイルを上書き" または "バージョンと日付でファイルを上書き (必要な場合)" が選択されている場合に有効になります。

メモ: ターゲット システムのファイルにバージョン番号があり、配布メディアのファイルにバージョン番号がない場合や、逆にターゲット システムのファイルにバージョン番号がなくて配布メディアのファイルにバージョン番号がある場合、バージョン番号のないファイルはバージョン番号が低いファイルとして処理されます。

バージョン: 新しいバージョン

配布メディアのファイルが番号が高いバージョン番号の場合、ターゲット システムのファイルは上書きされます。

配布メディアのファイルとターゲットファイルのファイルのバージョン番号が同じか、いずれのファイルにもバージョン番号がなくて バージョン、次に日付でファイルを上書き (必要な場合)がリストボックスで選択されている場合、ターゲット システムのファイルが上書きされるかどうかは、選択した日時のオプションによって決定されます。

そして、双方のファイルにバージョン番号があり、[バージョン番号によって上書き] を選択した場合、ターゲット システムのファイルには上書きされません。

バージョン: 新しいか同じ

配布メディアのファイルが番号が同じか高いバージョン番号の場合、ターゲット システムのファイルは上書きされます。

いずれのファイルにもバージョン番号がなく、リストボックスから「バージョン、次に日付でファイルを上書き(必要な場合)」が選択されている場合、ターゲット システムのファイルが上書きされるかどうかは、選択した日時のオプションによって決定されます。

そして、双方のファイルにバージョン番号があり、[バージョン番号によって上書き] を選択した場合、ターゲット システムのファイルには上書きされません。

バージョン: 古いバージョン

配布メディアのファイルが番号が低いバージョン番号の場合、ターゲット システムのファイルは上書きされます。

配布メディアのファイルとターゲットファイルのファイルのバージョン番号が同じか、いずれのファイルにもバージョン番号がなくて バージョン、次に日付でファイルを上書き (必要な場合)がリストボックスで選択されている場合、ターゲット システムのファイルが上書きされるかどうかは、選択した日時のオプションによって決定されます。

そして、双方のファイルにバージョン番号があり、[バージョン番号によって上書き] を選択した場合、ターゲット システムのファイルには上書きされません。

次のコントロールは、*リストボックス*で ″日付でファイルを上書き″ または ″バージョンと日付でファイルを上書き (必要な場合)″ が選択されている場合に有効になります。

日付/時間: 新しい日付

配布メディアのファイルがより新しい日付の場合、ターゲット システムのファイルは上書きされます。

日付/時間: 新しい日付か同じ

配布メディアのファイルがより新しいか同じ日付の場合、ターゲット システムのファイルは上書きされます。

日付/時間: 古い日付

配布メディアのファイルがより古い日付の場合、ターゲット システムのファイルは上書きされます。

[パッチ シーケンス] ダイアログ ボックス

[パッチ シーケンス] ダイアログ ボックスは、[パッチのデザイン] ビューでパッチシーケンスを作成すると開きます。

テーブル 34 · [パッチ シーケンス] ダイアログ ボックスのオプション

オプション	説明
ファミリー名	このパッチが所属するパッチファミリーの名前を指定します。Windows Installer 3.0 以降はパッチファミリーを利用して、同じファミリー内でスモールアップデートパッチとその他すべてのパッチを比較し、各パッチをターゲットマシンに適用する順番を決定します。 パッチ構成が複数のファミリーに所属することを示すには、各パッチファミリーの [シーケンス] タブに別の行を作成します。たとえば、パッチが複数の製品をアップデート可能な場合に、パッチを複数ファミリーに所属させる例が考えられます。
ターゲット	パッチ構成を特定の製品と関連付ける場合、このボックスに製品コード GUID を入力します。このボックスへの入力は必須ではありません。 パッチ構成が複数の GUID をターゲットにすることを示すには、各 GUID およびファミリーの組み合わせの[シーケンス] タブに別の行を作成します。
シーケンス番号	パッチ構成のシーケンス番号を指定します。この値は、同じパッチファイミリーの同じターゲット GUID (示されている場合) に所属するパッチのシーケンス内にあるパッチ構成の配置を示します。
以前のバージョンに優先する	同じパッチ ファミリーに属し、このパッチよりも低いシーケンス値を持つすべてのパッチの代わりにこのパッチを利用する場合、このチェック ボックスを選択します。以前のパッチが複数のパッチファミリーに所属している場合があるため、このチェック ボックスを選択しても、常にすべての以前のパッチと入れ替わることを保証するわけではありません。このチェック ボックスが選択されているかどうかに関わらず、スモール アップデート パッチがマイナーアップグレードパッチ、またはメジャー アップグレードパッチと入れ替わることはできません。

[パス変数のオーバーライド] ダイアログ ボックス

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[パス変数のオーバーライド] ダイアログ ボックスは、"パス変数のオーバーライド" 設定で省略記号ボタン (...) をクリックすると表示されます。"パス変数のオーバーライド" 設定は、[リリース] ビューにあるリリースの [ビルド] タブにあります。この設定を使って、選択されたリリースのビルド時にプロジェクト内の 1 つ以上のパス変数をオーバーライドできます。

[パス変数のオーバーライド] ダイアログ ボックスには、[パス変数] ビューで構成されたパス変数、環境変数、およびレジストリ変数のチェック ボックスが表示されます。定義済みのパス変数、または [リリース] ビュー内の同じリリースで既にオーバーライドされているパス変数のチェック ボックスは含まれていません。

オーバーライドする各パス変数のチェック ボックスを選択します。次に [OK] ボタンをクリックします。"パス変数のオーバーライド" 設定の下に、選択された各パス変数に新しい設定が追加されます。これらの設定を使って、パス変数に新しい値を指定できます。

ビルド時に、[パス変数] ビューで設定された値が、[リリース] ビューにあるリリースに構成された値でオーバーライドされます。

[パス変数の推奨] ダイアログ ボックス

[オプション] ダイアログ ボックスの [パス変数] タブで、**[常に [パス変数の推奨] ダイアログを表示]** オプションを選択した場合、新しいソース フォルダーをセットアップ プロジェクトに関連付けするたびに、[パス変数の推奨] ダイアログが表示されます。

このダイアログ ボックスは、[オプション] ダイアログ ボックスの [パス変数] タブで **[常にパス変数を推奨する]** オプションを選択し、つぎのどれかが True の場合にも表示されます。

- InstallShield が特定のソースフォルダーに対して複数のパス変数を推奨することが可能な場合。
- InstallShield が、既存のパス変数の基づいてパス変数を推奨できない場合、および [パス変数] タブの自動作成チェック ボックスを選択していない場合。

ダイアログ オプション

次のパス変数ベースのフォルダー表現をソースフォルダーに使用する

このオプションを指定すると、絶対パスの代わりに、推奨されるパス変数またはパス変数の組み合わせが自動的に入力されます。たとえば、C:¥Work¥Files をポイントする <MyFiles> と呼ばれるパス変数があって、C:¥Work¥Files¥Images からインストールにファイルを追加する場合、完全ハードコードされたパス C:¥Work¥Files¥Images ではなく <MyFiles>¥Images を使用することが推奨されます。

このオプションは、InstallShield が特定のソース フォルダーに対してパス変数を推奨できる場合のみ使用できます。

新規パス変数を作成する

ファイルが配置されるソース フォルダーのパス変数を作成する場合、[新しいパス変数を作成] オプションを選択します。このオプションを指定すると、新規ファイルが存在するフォルダーにマップされた標準パス変数が作成されます。[パス変数名] ボックスで新規の変数パス名を入力します。

***注意・**パス変数名は山かっこで囲まないでください。[OK] をクリックすると、括弧が自動的に付加されます。*

次の絶対パスを使用する

特定のリンクを表すパス変数を使用しない場合、**"次の絶対パスを使用"** オプションを選択します。このボタンの下にソースフォルダーへの絶対パスが表示されます。

ファイルとディレクトリの [アクセス許可] ダイアログ ボックス

***プロジェクト・**この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[アクセス許可] ダイアログ ボックス

[アクセス許可] ダイアログ ボックスで、ロックダウンされた環境で製品を実行するエンドユーザーのために、ファイルとフォルダーを保護するための設定を構成することができます。ファイルまたはフォルダーのアクセス許可を特定のグループとユーザーに割り当てることができます。たとえば、管理者グループに特定のファイルについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。

プロジェクトの [一般情報] ビューにある "ロックダウンの設定方法" 設定の選択に従って、InstallShield は ISLockPermissions テーブルまたは LockPermissions テーブルのどちらかにアクセス許可データを追加します。詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

次のテーブルは、[アクセス許可] ダイアログ ボックスの各領域についての説明です。

テーブル 35 • [アクセス許可] ダイアログ ボックスの各領域

領域	説明
名前	このグリッドで、ドメイン名およびユーザー名を任意に組み合わせて入力できます。エントリを追加するには、グリッド上で右クリックして [新規] をクリックします。同じコンテキストメニューを利用して、エントリを変更または削除することができます。 現在のユーザーのドメインを指定するには、"ドメイン" フィールドに **[%USERDOMAIN]** を選択します。現在インストールを実行中のユーザーを指定するには、"ユーザー" フィールドに **[LogonUser]** を選択します。ローカル システム上におけるユーザー アカウントのアクセス許可を設定するには、"ドメイン" フィールドを空白のままにします。"ドメイン" または "ユーザー" フィールドには、実行時に設定される任意の Windows Installer プロパティを入力できます (例、**[MYPROPERTY]**)。 [一般情報] ビューの "ロックダウンの設定方法" 設定で [カスタム InstallShield 処理] オプションを選択した場合、"ユーザー" フィールドには、よく使用されるセキュリティ識別子 (SID) の一覧が表示されます。ほとんどの SID は、[従来型の Windows Installer 処理] オプションではサポートされていません。 [カスタム InstallShield 処理] オプションは、"ユーザー" フィールドに一覧されているすべての SID の翻訳された名前をサポートします。従来型のオプションで、非英語システム上で翻訳された名前を使ってアクセス許可を設定すると、インストールが失敗する可能性があります。 ***ヒント:*** *カスタム InstallShield 処理オプションと、従来型の Windows Installer 処理オプションについての詳細は、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。*
アクセス許可	[名前] 領域で名前を選択し、[アクセス許可] ボックスにあるチェックボックスを選択またはクリアして、対応するファイルまたはフォルダーのアクセス許可を構成します。アクセス許可を選択すると、[詳細] ボタンをクリックして、他の関連するアクセス許可と詳細設定を指定することができます。

テーブル 35 • [アクセス許可] ダイアログ ボックスの各領域 (続き.)

領域	説明
アクセス拒否	[アクセス許可] ボックスで選択するアクセス許可を明示的に拒否するには、このチェック ボックスを選択します。 このチェック ボックスは、[一般情報] ビューにある "ロックダウンの設定方法" 設定で [カスタム InstallShield 処理] オプションが選択されている場合のみ利用できます。[従来型の Windows Installer 処理] オプションは、この動作をサポートしません。 ***ヒント:*** *カスタム InstallShield 処理オプションと、従来型の Windows Installer 処理オプションについての詳細は、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。*

[特殊なアクセス許可] ダイアログ ボックス

[アクセス許可] ダイアログ ボックスで [詳細設定] ボタンをクリックすると、[特殊なアクセス許可] ダイアログ ボックスが開きます。次のテーブルは、[特殊なアクセス許可] ダイアログ ボックスの各領域についての説明です。

テーブル 36 • [特殊なアクセス許可] ダイアログ ボックスの各領域

領域	説明
特殊なアクセス許可	このボックスで、設定するアクセス許可のチェック ボックスを選択します。
これらのアクセス許可を子オブジェクトに適用する	フォルダーのアクセス許可を構成するとき、すべてのフォルダーのサブフォルダーとファイルにアクセス許可を適用するには、このチェック ボックスを選択します。 このチェック ボックスは、[一般情報] ビューにある "ロックダウンの設定方法" 設定で [カスタム InstallShield 処理] オプションが選択されている場合のみ利用できます。[従来型の Windows Installer 処理] オプションは、この動作をサポートしません。 ***ヒント:*** *カスタム InstallShield 処理オプションと、従来型の Windows Installer 処理オプションについての詳細は、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。*

レジストリ キーの [アクセス許可] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[アクセス許可] ダイアログ ボックス

[アクセス許可] ダイアログ ボックスで、ロックダウンされた環境で製品を実行するエンドユーザーのために、レジストリ キーを保護するための設定を構成することができます。レジストリ キーのアクセス許可を特定のグループとユーザーに割り当てることができます。たとえば、管理者グループに特定のレジストリ キーについての [読み取り]、[書き込み]、および [削除] アクセス許可を割り当てることができますが、別のグループのすべてにユーザーについては [読み取り] 許可のみ割り当てることができます。

プロジェクトの [一般情報] ビューにある "ロックダウンの設定方法" 設定の選択に従って、InstallShield は ISLockPermissions テーブルまたは LockPermissions テーブルのどちらかにアクセス許可データを追加します。詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

次のテーブルは、[アクセス許可] ダイアログ ボックスの各領域についての説明です。

テーブル 37 • [アクセス許可] ダイアログ ボックスの各領域

領域	説明
名前	このグリッドで、ドメイン名およびユーザー名を任意に組み合わせて入力できます。エントリを追加するには、グリッド上で右クリックして [新規] をクリックします。同じコンテキストメニューを利用して、エントリを変更または削除することができます。 現在のユーザーのドメインを指定するには、"ドメイン" フィールドに **[%USERDOMAIN]** を選択します。現在インストールを実行中のユーザーを指定するには、"ユーザー" フィールドに **[LogonUser]** を選択します。ローカル システム上におけるユーザー アカウントのアクセス許可を設定するには、"ドメイン" フィールドを空白のままにします。"ドメイン" または "ユーザー" フィールドには、実行時に設定される任意の Windows Installer プロパティを入力できます (例、**[MYPROPERTY]**)。 [一般情報] ビューの "ロックダウンの設定方法" 設定で [カスタム InstallShield 処理] オプションを選択した場合、"ユーザー" フィールドには、よく使用されるセキュリティ識別子 (SID) の一覧が表示されます。ほとんどの SID は、[従来型の Windows Installer 処理] オプションではサポートされていません。 [カスタム InstallShield 処理] オプションは、"ユーザー" フィールドに一覧されているすべての SID の翻訳された名前をサポートします。従来型のオプションで、非英語システム上で翻訳された名前を使ってアクセス許可を設定すると、インストールが失敗する可能性があります。 ***ヒント:*** *カスタム InstallShield 処理オプションと、従来型の Windows Installer 処理オプションについての詳細は、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。*
アクセス許可	[名前] 領域で名前を選択し、[アクセス許可] ボックスにあるチェックボックスを選択またはクリアして、対応するレジストリ キーのアクセス許可を構成します。アクセス許可を選択すると、[詳細] ボタンをクリックして、他の関連するアクセス許可と詳細設定を指定することができます。

テーブル 37 - [アクセス許可] ダイアログ ボックスの各領域 (続き.)

領域	説明
アクセス拒否	[アクセス許可] ボックスで選択するアクセス許可を明示的に拒否するには、このチェック ボックスを選択します。 このチェック ボックスは、[一般情報] ビューにある "ロックダウンの設定方法" 設定で [カスタム InstallShield 処理] オプションが選択されている場合のみ利用できます。[従来型の Windows Installer 処理] オプションは、この動作をサポートしません。 ***ヒント:*** *カスタム InstallShield 処理オプションと、従来型の Windows Installer 処理オプションについての詳細は、「ロックダウン環境における ファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。*

[特殊なアクセス許可] ダイアログ ボックス

[アクセス許可] ダイアログ ボックスで [詳細設定] ボタンをクリックすると、[特殊なアクセス許可] ダイアログ ボックスが開きます。次のテーブルは、[特殊なアクセス許可] ダイアログ ボックスの各領域についての説明です。

テーブル 38 - [特殊なアクセス許可] ダイアログ ボックスの各領域

領域	説明
特殊なアクセス許可	このボックスで、設定するアクセス許可のチェック ボックスを選択します。
これらのアクセス許可を子オブジェクトに適用する	レジストリ キーのアクセス許可を構成するとき、すべてのキーのサブキーにアクセス許可を適用するには、このチェック ボックスを選択します。 このチェック ボックスは、[一般情報] ビューにある "ロックダウンの設定方法" 設定で [カスタム InstallShield 処理] オプションが選択されている場合のみ利用できます。[従来型の Windows Installer 処理] オプションは、この動作をサポートしません。 ***ヒント:*** *カスタム InstallShield 処理オプションと、従来型の Windows Installer 処理オプションについての詳細は、「ロックダウン環境における ファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。*

[プラットフォーム スイート] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

[プラットフォーム スイート] ダイアログ ボックスでは、選択されたコンポーネントを関連付けるプラットフォーム スイートを指定できます。このダイアログ ボックスにアクセスするには、[コンポーネント] ビューまたは [セットアップのデザイン] ビューにあるコンポーネントの プラットフォーム スイート" 設定で省略記号ボタン (...) をクリックします。

テーブル 39 • [プラットフォーム スイート] ダイアログ ボックスの設定

設定	説明
ターゲット システムのプラットフォーム スイートの検証を行わずにインストールする	コンポーネントのファイルのインストールは、ターゲット システムのスイートに依存しません。
ターゲット システムに複数のプラットフォーム スイートが指定されている場合にインストールする	インストールは、選択されたスイートの 1 つ以上がターゲット システムに存在する場合のみ、コンポーネントのファイルをインストールします。 このオプションを選択する場合は、適切なプラットフォーム スイートのチェック ボックスを選択します。
ターゲット システムに指定されたすべてのプラットフォームスイートがある場合にのみインストールする	インストールは、選択されたスイートすべてがターゲット システムに存在する場合のみ、コンポーネントのファイルをインストールします。 このオプションを選択する場合は、適切なプラットフォーム スイートのチェック ボックスを選択します。
すべてのプラットフォーム スイートを表示	利用可能なすべてのスイートを参照するには、このチェック ボックスを選択します。完全なリストは、Windows API の OSVERSIONINFOEX データ構造 (MSDN ドキュメントを参照) で指定することができるものです。 最も一般的なスイートのみを表示するには、このチェック ボックスを選択解除します。

[プラットフォーム] ダイアログ ボックス

プロジェクト: *一部の情報は InstallScript プロジェクトに適用し、他の一部の情報は InstallScript MSI プロジェクトに適用します。*

[プラットフォーム] ダイアログ ボックスの起動方法に応じて、次のいずれかを指定することができます:

- プロジェクトでコンポーネントまたはリリースにその後オペレーティング システムを選択するときの選択肢に加えるプラットフォーム。これは、InstallScript プロジェクトで使用することができます。

- コンポーネントのプラットフォーム要件。これは InstallScript プロジェクトおよび InstallScript MSI プロジェクトで使用できます。

プロジェクト レベルのプラットフォーム サポート (InstallScript プロジェクト)

プロジェクト レベルの [プラットフォーム] ダイアログ ボックスにアクセスすると、プロジェクトでコンポーネントまたはリリースにオペレーティング システムを選択するときの選択肢に加えるプラットフォームを指定することができます。

メモ: *プロジェクト レベルでプラットフォームを指定しても、インストールを実行するターゲット は作成されません。InstallScript プロジェクトでターゲット を作成する場合、ターゲットのオペレーティング プラットフォームを識別する SYSINFO 構造を使用することができます。*

タスク ***プロジェクト レベルの [プラットフォーム] ダイアログ ボックスにアクセスするには、以下の手順に従います:***

1. **[インストール情報]** の下のビュー リストにある **[一般情報]** をクリックします。
2. **"プラットフォーム フィルター"** 設定で、省略記号ボタン (...) をクリックします。

次のテーブルは、プロジェクト レベルで [プラットフォーム] ダイアログ ボックスにアクセスしたときに表示される設定です。

テーブル 40 ・ プロジェクトのための [プラットフォーム] ダイアログ ボックスの設定

設定	説明
プロジェクトは、InstallShield がサポートするすべてのプラットフォームをサポートする	InstallShield の使用中、次の設定でサポートされている実行時のプラットフォームをすべて一覧表示する場合、このオプションを選択します: • コンポーネント レベルの"オペレーティング システム" 設定 (InstallScript プロジェクトと InstallScript MSI プロジェクト) • リリース レベルの "プラットフォーム" サポート (InstallScript プロジェクト) このオプションを選択すると、プロジェクト内の特定のコンポーネントまたはリリースがいずれかのサポートされているプラットフォームでターゲットされるように指定できます。

テーブル 40 • プロジェクトのための [プラットフォーム] ダイアログ ボックスの設定 (続き.)

設定	説明
プロジェクトは、以下で選択されたプラットフォームのみをサポートする	InstallShield の使用中、次の設定で表示されているプラットフォームの一部のみを一覧表示する場合、このオプションを選択します: • コンポーネント レベルの"オペレーティング システム" 設定 (InstallScript プロジェクトと InstallScript MSI プロジェクト) • リリース レベルの "プラットフォーム" サポート (InstallScript プロジェクト) 表示するダイアログを選択します。 ***ヒント:*** *連続する複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択したあと SHIFT キーを押しながら最後のオペレーティング システムを選択します。連続しない複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択してから CTRL キーを押しながらその他の各オペレーティング システムを選択します。* 一般的に、プロジェクト レベルのこの設定で表示されないプラットフォームがあるとき、プロジェクト内にある特定のコンポーネントまたはリリースをこのプラットフォームにターゲットするという指定ができなくなります。

コンポーネント レベルのプラットフォーム サポート (InstallScript プロジェクトと InstallScript MSI プロジェクト)

コンポーネント レベルで [プラットフォーム] ダイアログ ボックスにアクセスすると、選択されたコンポーネントにサポートされるプラットフォームを指定できます。

タスク

コンポーネント レベルの [プラットフォーム] ダイアログ ボックスにアクセスするには、以下の手順に従います:

1. **[編成]** の下のビュー リストにある **[コンポーネント]** をクリックします。
2. **[コンポーネント]** エクスプローラーで、プラットフォームの要件を構成するコンポーネントをクリックします。
3. **"オペレーティング システム"** 設定ドの値をクリックして、参照 (...) ボタンをクリックします。

次のテーブルは、コンポーネント レベルで [プラットフォーム] ダイアログ ボックスにアクセスしたときに表示される設定です。

テーブル 41 ・ コンポーネントのための [プラットフォーム] ダイアログ ボックスの設定

設定	説明
セットアップが実行されているプラットフォームにかかわらず、このコンポーネントのファイルをインストールする	選択されたコンポーネントがオペレーティング システム非依存 (コンポーネントに特定のオペレーティング システム固有のデータがない) の場合、このオプションを選択します。
以下のチェックされたプラットフォームでセットアップが実行されているときにのみ、このコンポーネントのファイルをインストールする	選択されたコンポーネントが 1 つまたは複数のオペレーティング システムに固有の場合、この設定を選択して、該当するオペレーティング システムを選択します。ターゲット マシンのオペレーティング システムがこの設定に指定されたオペレーティング システムの中にない場合、コンポーネントはインストールされません。 ***ヒント:*** *連続する複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択したあと SHIFT キーを押しながら最後のオペレーティング システムを選択します。連続しない複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択してから CTRL キーを押しながらその他の各オペレーティング システムを選択します。* ***プロジェクト:*** *InstallScript プロジェクトの場合、プロジェクト レベルでチェック ボックスがクリアになっているオペレーティング システムは、このダイアログ ボックスで表示されるオペレーティング システムの一覧からすべて除外されます。* *選択されたコンポーネントがリリースでサポートされていないプラットフォームをサポートしている場合、このコンポーネントはビルド時にリリースに含まれません。*

[ビルド後のイベント] ダイアログ ボックス

エディション: *この情報は、InstallShield Premier Edition に適用します。*

[ビルド後のイベント] ダイアログ ボックスは、"ビルド後のイベント" 設定で省略記号ボタン (...) をクリックすると表示されます。"ビルド後のイベント" 設定は、[リリース] ビューにあるリリースの [イベント] タブにあります。

[ビルド後のイベント] ダイアログ ボックスを使って、InstallShield がリリースをビルドおよび署名した後に実行する 1 つ以上のコマンドを指定できます。

1 つ以上のコマンドを指定するには、1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。

コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。

詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

[ビルド前のイベント] ダイアログ ボックス

***エディション:** この情報は、InstallShield Premier Edition に適用します。*

[ビルド前のイベント] ダイアログ ボックスは、"ビルド前のイベント" 設定で省略記号ボタン (...) をクリックすると表示されます。"ビルド前のイベント" 設定は、[リリース] ビューにあるリリースの [イベント] タブにあります。

[ビルド前のイベント] ダイアログ ボックスを使って、InstallShield がリリースをビルドし始める前に実行する 1 つ以上のコマンドを指定できます。このイベントは InstallShield がリリース フォルダとログ ファイルを作成した後、リリースのビルドを開始する前に実行します。

1 つ以上のコマンドを指定するには、1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。

コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。

詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

[圧縮前のイベント] ダイアログ ボックス

***エディション:** この情報は、InstallShield Premier Edition に適用します。*

[圧縮前のイベント] ダイアログ ボックスは、"圧縮前のイベント" 設定で省略記号ボタン (...) をクリックすると表示されます。"圧縮前のイベント" 設定は、[リリース] ビューにあるリリースの [イベント] タブにあります。

[圧縮前のイベント] ダイアログ ボックスを使って、製品のデータ ファイルを .cab ファイルに格納する場合、InstallShield が .msi パッケージと .cab ファイルをビルドした後に実行する 1 つ以上のコマンドを指定します。このイベントは .cab ファイルが .msi パッケージにストリームされた後、.msi パッケージにデジタル署名が行われて Setup.exe ファイルにストリームされる前に発生します。

1 つ以上のコマンドを指定するには、1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。

コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。

詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

[ファイルの準備イベント] ダイアログ ボックス

エディション・この情報は、InstallShield Premier Edition に適用します。

[ファイルの準備イベント] ダイアログ ボックスは、**"ファイルの準備イベント"** 設定で省略記号ボタン (...) をクリックすると表示されます。**"ファイルの準備イベント"** 設定は、**[リリース]** ビューにあるリリースの **[イベント]** タブにあります。

[ファイルの準備イベント] ダイアログ ボックスを使って、InstallShield のビルドの中に実行する 1 つ以上のコマンドを指定できます。このイベントは、InstallShield がリリース フォルダーおよびログ ファイルを作成した後、InstallShield ビルドによって生成された DLL が .msi パッケージまたは Setup.exe ファイルにストリームされる前に実行されます。

1 つ以上のコマンドを指定するには、1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。

コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。

詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

[前提条件設定] ダイアログ ボックス

[前提条件設定] ダイアログ ボックスは、[InstallShield 前提条件] エディターの [条件] タブにある [追加] ボタンをクリックすると開きます。ダイアログは、このタブで既存の条件を選択し、[変更] ボタンをクリックしたときも開きます。

[前提条件設定] ダイアログ ボックスでは、InstallShield 前提条件がターゲットマシンにインストール済みかどうかを決定するインストール条件の定義を行います。これを行わなかった場合、ターゲット システムは前提条件が適切にインストールされなかったと見なして作動するため、問題が発生します。また、オペレーティング システム、レジストリ、またはファイル要件を指定するインストール条件を作成することも可能です。

テーブル 42 ・ [前提条件設定] ダイアログ ボックスのオプション

オプション	説明
レジストリキーの存在の有無	このオプションを選択する場合、レジストリ キーの名前を入力し、64 ビット システム上でチェックを行う適切なレジストリの場所 (**32 ビット** または **64 ビット**.) を選択します。**[デフォルト]** ラジオボタンは、選択が行われていないことを示し、32 ビットが使用されます。 このレジストリ キーを持つターゲットマシンがこのインストール条件を満たす場合、**[指定したレジストリ キーが存在する場合]** を選択します。このレジストリ キーを持たないターゲット マシンがこのインストール条件を満たす必要がある場合、**[指定したレジストリ キーが存在しない場合]** を選択します。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
レジストリ エントリが指定の値を含む	このオプションを選択した場合、レジストリ キーの名前、値名、および値データを入力します。また、64 ビット システム上でチェックを行う適切なレジストリの場所 (**32 ビット** または **64 ビット**) を選択します。**[デフォルト]** ラジオボタンは、選択が行われていないことを示し、32 ビットが使用されます。 さらに、このダイアログ ボックスで指定したレジストリ エントリとターゲットマシン上のレジストリ エントリとを比較するのに利用するオプションを選択します。 たとえば、[値データ] ボックスで **1.4.2** を指定し、**[より大きい]** を選択した場合、このレジストリ エントリに 1.4.2 より大きい値を持つターゲット マシンはインストール条件を満たすことになります。
レジストリ エントリが指定のバージョン値を含む	このオプションを選択した場合、レジストリ キーの名前、値名、および値データを入力します。値データは、バージョン番号です。 バージョン番号の各フィールドの最初に付いているゼロは無視されます。たとえば、3.5.3.0010 は 3.5.3.009 よりも大きく、3.5.3.009 は 3.5.3.01 よりも大きいです。 さらに、このダイアログ ボックスで指定したレジストリ エントリとターゲットマシン上のレジストリ エントリとを比較するのに利用するオプションを選択します。 最後に、64 ビット システム上でチェックを行う適切なレジストリの場所 (**32 ビット** または **64 ビット**) を選択します。**[デフォルト]** ラジオボタンは、選択が行われていないことを示し、32 ビットが使用されます。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
レジストリ エントリがワイルドカード パターンと一致し、指定のバージョン値を含む	このオプションを選択する場合、ワイルドカード文字 (%)、レジストリ値名、および値データとともに、レジストリ サブキー名の予測可能部分を入力します。値データは、バージョン番号です。 予測可能なレジストリ サブキー名パターンを指定する場合、レジストリ サブキー名と共にワイルドカード文字 (%) を使用する必要があります。 例: **HKEY_LOCAL_MACHINE¥SOFTWARE¥WOW6432Node¥Microsoft¥Updates¥.NET Core¥%Microsoft .NET Core%** この場合、InstallShieldは、Microsoft .NET Core を含む名前を持つターゲット システム上の .NET Core ノードの下にあるすべてのレジストリ サブキーを検索し、一致する各レジストリ サブ キー内で入力または指定されたレジストリ値と値データを評価します。 ***重要•*** *ワイルドカード文字 (%) は、非常に柔軟かつ正確な検索であるために、レジストリ サブキー名内で複数回使用することができます。たとえば、"%Microsoft% %.NET% %Core%" のようなパターンにより、InstallShield は幅広い関連レジストリ サブキーを識別することができます。* バージョン番号の各フィールドの最初に付いているゼロは無視されます。たとえば、3.5.3.0010 は 3.5.3.009 よりも大きく、3.5.3.009 は 3.5.3.01 よりも大きいです。 このダイアログ ボックスで指定したレジストリ エントリとターゲット マシン上のレジストリ エントリとを比較するのに使用する適切な条件を選択します。 また、64 ビット システム上でチェックを行う適切なレジストリの場所 (**32 ビット** または **64 ビット**) を選択します。**[デフォルト]** ラジオボタンは、選択が行われていないことを示し、32 ビット レジストリの場所がチェックされます。。 ***メモ•*** *このオプションは、次のような場合に有効です。* • レジストリキー部分のみが予測可能である場合。 • 指定されたレジストリキー名の一部と一致するか、または含むレジストリ サブキー内の、入力または指定されたレジストリ値と値データを、InstallShieldが検出および比較するようにしたい場合。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
ファイルの存在の有無	このオプションを選択した場合、次のどれかを行います。 • ファイルの名前とパスを指定します。定義済みのフォルダーを指定するには、一覧で適切なプロパティを選択します。使用可能なプロパティは、[CommonFiles64Folder]、[CommonFilesFolder]、[ProgramFiles64Folder]、[ProgramFilesFolder]、[System64Folder]、[SystemFolder]、および [WindowsFolder] です。 64 ビット フォルダーの場所の 1 つを選択すると、インストールは 64 ビット ターゲット システム上の 64 フォルダーの場所をチェックし、32 ビット ターゲット システム上の 32 フォルダーの場所をチェックします。 • パスまたはプロパティなしで、ファイル名のみを指定します。ターゲットマシン上の PATH 環境変数として定義されたすべてのディレクトリ内で、インストールは指定されたファイルを検索します。 • レジストリ キーをプロパティとして指定します。この場合、インストールはレジストリ値を使ってレジストリキーを解決します。レジストリ パスの最後の要素は、値名として処理されますので注意してください。パスの最後の文字が円記号の場合、そのレジストリ キーのデフォルト レジストリ エントリの値が利用されます。 例えば次のように値を入力した場合、インストールは HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥ORACLE_HOME レジストリ エントリの値として指定されたパスに oci.dll という名前のファイルを持つ bin ディレクトリが含まれるかどうかを確認します。 **[HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥ORACLE_HOME]bin¥oci.dll** このファイルを持つターゲット マシンがこのインストール条件を満たす場合、**[指定したファイルが上記で指定した場所で見つかった場合]** を選択します。このファイルを持たないターゲット マシンがこのインストール条件を満たす場合、**[指定したファイルが上記で指定した場所で見つからなかった場合]** を選択します。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
特定の日付をもつファイルが存在する	このオプションを選択した場合、次のどれかを行います。 • ファイルの名前とパスを指定します。定義済みのフォルダーを指定するには、一覧で適切なプロパティを選択します。使用可能なプロパティは、[CommonFiles64Folder]、[CommonFilesFolder]、[ProgramFiles64Folder]、[ProgramFilesFolder]、[System64Folder]、[SystemFolder]、および [WindowsFolder] です。 64 ビット フォルダーの場所の 1 つを選択すると、インストールは 64 ビット ターゲット システム上の 64 フォルダーの場所をチェックし、32 ビット ターゲット システム上の 32 フォルダーの場所をチェックします。 • パスまたはプロパティなしで、ファイル名のみを指定します。ターゲットマシン上の PATH 環境変数として定義されたすべてのディレクトリ内で、インストールは指定されたファイルを検索します。 • レジストリ キーをプロパティとして指定します。この場合、インストールはレジストリ値を使ってレジストリキーを解決します。レジストリ パスの最後の要素は、値名として処理されますので注意してください。パスの最後の文字が円記号の場合、そのレジストリ キーのデフォルト レジストリ エントリの値が利用されます。 例えば次のように値を入力した場合、インストールは HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥ORACLE_HOME レジストリ エントリの値として指定されたパスに oci.dll という名前のファイルを持つ bin ディレクトリが含まれるかどうかを確認します。 **[HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥ORACLE_HOME]bin¥oci.dll** さらに、日付、および必要に応じて時間を入力します。例: **4/26/2004 3:57:46 PM** 適切な一致オプションを選択して、InstallShield 前提条件を実行するタイミングを指定します。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
特定のバージョンをもつファイルが存在する	このオプションを選択した場合、次のどれかを行います。 • ファイルの名前とパスを指定します。定義済みのフォルダーを指定するには、一覧で適切なプロパティを選択します。使用可能なプロパティは、[CommonFiles64Folder]、[CommonFilesFolder]、[ProgramFiles64Folder]、[ProgramFilesFolder]、[System64Folder]、[SystemFolder]、および [WindowsFolder] です。 64 ビット フォルダーの場所の 1 つを選択すると、インストールは 64 ビット ターゲット システム上の 64 フォルダーの場所をチェックし、32 ビット ターゲット システム上の 32 フォルダーの場所をチェックします。 • パスまたはプロパティなしで、ファイル名のみを指定します。ターゲットマシン上の PATH 環境変数として定義されたすべてのディレクトリ内で、インストールは指定されたファイルを検索します。 • レジストリ キーをプロパティとして指定します。この場合、インストールはレジストリ値を使ってレジストリキーを解決します。レジストリ パスの最後の要素は、値名として処理されますので注意してください。パスの最後の文字が円記号の場合、そのレジストリ キーのデフォルト レジストリ エントリの値が利用されます。 例えば次のように値を入力した場合、インストールは HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥ORACLE_HOME レジストリ エントリの値として指定されたパスに oci.dll という名前のファイルを持つ bin ディレクトリが含まれるかどうかを確認します。 **[HKEY_LOCAL_MACHINE¥SOFTWARE¥ORACLE¥ORACLE_HOME]bin¥oci.dll** さらに、ファイルのバージョン番号を入力します。例: **6.7.8.9** 適切な一致オプションを選択して、InstallShield 前提条件を実行するタイミングを指定します。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
App パッケージ存在の有無	このオプションを選択した場合、次のどれかを行います。 このオプションを選択した場合、次のどれかを行います。 • App パッケージ識別名 (ターゲット マシンで検出したいパッケージ) を指定します。パッケージ識別情報は、パッケージ マニフェスト ファイルで使用できます。**Microsoft.DesktopAppInstaller** は、次のパッケージ マニフェスト ID のパッケージ ID 識別名です。 <Identity ProcessorArchitecture="x86" Version="1.11.11102.0" Publisher="CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond, S=Washington, C=US" Name="Microsoft.DesktopAppInstaller" /> • App パッケージ発行元 (ターゲット マシンで検出したいパッケージ) を指定します。パッケージ発行元情報は、パッケージ マニフェスト ファイルで使用できます。**CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond, S=Washington, C=US** は、上のパッケージ マニフェスト ID のパッケージ発行元です。 • App パッケージアーキテクチャ (ターゲット マシンで検出したいパッケージ) を指定します。パッケージアーキテクチャは、パッケージ マニフェスト ファイルで使用できます。"x86" は上のパッケージ マニフェスト ID のパッケージ アーキテクチャです。 • App パッケージ バージョン (ターゲット マシンで検出したいパッケージ) を指定します。パッケージバージョンは、パッケージ マニフェスト ファイルで使用できます。**1.11.11102.0** は上のパッケージ マニフェスト ID のパッケージ バージョンです。 ターゲット マシン上に App パッケージ条件が存在しない場合のみ、つまり、前提条件エンジンがターゲット マシン上に識別名、発行元、およびアーキテクチャを持つパッケージを見つけることができなかったときのみ、ターゲット マシンに前提条件インストーラーがインストールされます。ターゲット マシンにパッケージが存在する場合、前提条件エンジンがパッケージ バージョンをチェックします。インストール済みバージョンが検証中のパッケージ バージョンよりも古い場合は、ターゲット マシン上にそのパッケージもインストールされます。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
セットアップが指定のプラットフォーム上で実行されている	このオプションを選択した場合、**[前提条件を実行するプラットフォームを選択します]** ボックスで定義済みのプラットフォームの 1 つをクリックするか、または [カスタム] を選択して、任意のプラットフォーム要件を定義します。 この前提条件が True 評価されるサービス パックの番号を指定するには、[サービス パック] ボックスを使用します。たとえば、ターゲット システムにサービス パック 2、3、または 4 が必要な場合、最初のボックスに 2、2 番目のボックスに 4 を入力します。将来のサービス パックもすべて含めるには、2 番目のボックスを空白のままに残します。サービス パック 3 のみをターゲットとする場合は、両方のボックスに 3 と入力します。 [カスタム] を選択して、特定のオペレーティング システムを指定すると、[カスタム] 領域にある次の 1 つまたは複数の設定を構成することができます。 • **プラットフォーム ID** – 必要なプラットフォームを選択します。 選択した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の dwPlatformId メンバーに一致する場合、この部分のオペレーティング システム条件は、True に評価されます。 • **メジャー バージョン** – オペレーティング システムの必要なメジャー バージョンを指定します。 指定した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の dwMajorVersion メンバーに一致する場合、この部分のオペレーティング システム条件は、True に評価されます。 このボックスで、**0** と入力した場合、InstallShield が作成するオペレーティング システム条件に、メジャー バージョンおよびマイナー バージョンの要件は含まれません。この動作は、[メジャー バージョン] ボックスと [マイナー バージョン] ボックスを空白にした場合と同じになります。 • **マイナー バージョン** – オペレーティング システムの必要なマイナー バージョンを指定します。 指定した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の dwMinorVersion メンバーに一致する場合、この部分のオペレーティング システム条件は、True に評価されます。 このボックスで、**0** は有効な値です。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
セットアップが指定のプラットフォーム上で実行されている (続き.)	• **製品 (OS) の種類** — 必要なオペレーティング システムの種類を選択します。 選択した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の wProductType メンバーに一致する場合、この部分の条件は、True に評価されます。 InstallShield 前提条件がサーバー コントローラーとドメイン コントローラーを両方サポートする場合、**[サーバーまたはドメイン コントローラー]** オプションを選択することができます。 • **プロセッサ アーキテクチャ**—次のプロセッサから必要なプロセッサ アーキテクチャを選択します: ▪ Win32 ▪ Win64 (任意) ▪ Win64 (x64) ▪ Win64 (Itanium) ▪ ARM ▪ ARM64 選択したアーキテクチャが、ターゲット システムのオペレーティング システムのアーキテクチャに一致する場合、この部分のオペレーティング システム条件は、True に評価されます。 • **最小 CSD バージョン** — オペレーティング システムで必要な最小サービス パックまたは他のバージョン情報を指定します。 この設定は、InstallShield の以前のバージョンで作成された InstallShield 前提条件との互換性と、Windows 9x プラットフォームのサブバージョンを識別できるようにするために提供されています。この設定を使用して、特定のサービス パックが推奨されていないことを指定します。特定のサービス パックを指定する必要がある場合、[最小サービス パック] ボックスを使用します。 選択した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の szCSDVersion メンバーと同じバージョンか、それより以前のバージョンの場合、この部分のオペレーティング システム条件は、True に評価されます。 たとえば、“A” と指定したときに、ターゲット マシンの szCSDVersion 値が “B” の場合、この部分のオペレーティング システム条件は、True に評価されます。“B” と指定して、ターゲット マシンの値が “A” の場合、条件は、False に評価されます。 同様に、“Service Pack 1” と指定したときに、ターゲット マシンの値が “Service Pack 2” の場合、この部分のオペレーティング システム条件は、True に評価されます。“Service Pack 2” と指定して、ターゲット マシンの値が “Service Pack 1” の場合、条件は、False に評価されます。 比較の問題を防止するために、指定したバージョン内、およびターゲット システムの szCSDVersion メンバーの値内の先頭と末尾のスペースは、すべて無視されます。

テーブル 42 • [前提条件設定] ダイアログ ボックスのオプション (続き.)

オプション	説明
セットアップが指定のプラットフォーム上で実行されている (続き.)	• **最小ビルド番号** — オペレーティング システムで必要な最小ビルド番号を指定します。 選択した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の dwBuildNumber メンバーと同じバージョンか、それより以前のバージョンの場合、この部分のオペレーティング システム条件は、True に評価されます。 • **最大ビルド番号** — オペレーティング システムで必要な最大ビルド番号を指定します。 選択した値が、ターゲット システムのオペレーティング システムの OSVERSIONINFOEX 構造体の dwBuildNumber メンバーと同じバージョンか、それより大きいバージョンの場合、この部分のオペレーティング システム条件は、True に評価されます。 オペレーティング システム条件の指定に関する詳細については、「InstallShield 前提条件のオペレーティング システム条件を追加する」を参照してください。 OSVERSIONINFOEX 構造体に関する詳細については、MSDN Web サイトの OSVERSIONINFOEX 構造を参照してください。

[特権] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[特権] ダイアログ ボックスでは、構成中のサービスが必要とする 1 つ以上の特権を指定できます。次の状況下で、このダイアログ ボックスが開きます:

- [サービス] ビュー、または [セットアップ デザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューにあるコンポーネントの [詳細設定] 領域でサービスを構成しているとき。
- 右側ペインに表示される設定の [構成の設定] カテゴリにあるイベントの "構成の種類" 設定で、**[必要な特権のリストを変更する]** オプションが選択されていて、そのイベントの "引数" 設定にある省略記号ボタン (...) をクリックしたとき。

1 つ以上のチェック ボックスを選択したとき。

変更は、次回のシステム開始時に有効になります。

[製品条件ビルダー] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[製品条件ビルダー] ダイアログ ボックスは、有効な Windows Installer プロパティと条件式の演算子を提供し、製品条件の作成を簡単にします。

たとえば、製品を正しく実行するために RAM が 64 MB 必要な場合、[製品条件ビルだ] ダイアログ ボックスを使って、条件を作成できます。実行時、Windows Installer がターゲット システムをチェックして、インストールされている RAM のサイズを判断します。64 MB 以下の場合、Windows Installer がエラー メッセージを表示して、インストールが終了します。

テーブル 43 • [製品条件ビルダー] ダイアログ ボックスの設定

設定	説明
条件	このボックスには、次の列を含むグリッドが表示されます: • **条件** – この列に条件を直接入力するか、[プロパティ] リスト、[演算子] リスト、および [追加] ボタンを使って、条件式をビルドすることができます • **メッセージ** – この列に、条件が False 評価された場合に表示するエラー メッセージを指定します。 この列に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 このボックスに新しい条件を追加するには、[新しい条件] ボタンをクリックしてから、[条件] と [メッセージ] 列に適切な情報を入力します。 このボックスの条件を削除するには、条件を選択してから、[条件の削除] ボタンをクリックします。 ***重要:*** *このダイアログ ボックスの [OK] ボタンをクリックすると、InstallShield によって基本的な条件検証が行われますが、条件ステートメントが予定通りの結果に評価されるかどうか、確認してください。詳しい情報と条件のサンプルについては、「条件ステートメントのビルド」を参照してください。* *[メッセージ] 列で指定したテキストを表示する Windows Installer ダイアログは、¥r (改行)、¥n (新しい行) または ¥t (タブ) 等のエスケープ シーケンスを認識しません。*
Properties	このリストには、[プロパティ マネージャー] ビューで追加されたすべてのプロパティだけでなく、一般的な Windows Installer プロパティが一覧されます。評価するプロパティを選択してから、このリストの隣にある [**追加**] ボタンをクリックします。 リストに評価を行うプロパティが含まれていない場合、適切なプロパティ名を [条件] 列に入力できます。
演算子	このリストには、Windows Installer が認識する有効な演算子すべてが含まれています。使用する演算子を選択してから、このリストの隣にある [追加] ボタンをクリックします。[条件] ボックスの条件ステートメントに演算子が追加されます。

Conditional Statement Syntax (Windows Installer ヘルプ ライブラリ)

[プロジェクト設定] ダイアログ ボックス

[プロジェクト設定] ダイアログ ボックスには、現在開いているプロジェクトに関する情報を変更できます。

このダイアログ ボックスは、[プロジェクト] メニューの [設定] コマンドを選択すると、次のタブと一緒に表示されます。このダイアログ ボックスは、[オブジェクト] ビューの「InstallShield オブジェクト/マージ モジュール] ペインでアイテムを右クリックして [新しいオブジェクトの登録] または [プロパティ] を選択した場合、最後の 2 つのタブしか一緒に表示されません。その場合、すべてのプロパティは編集不可能です。

- [プロジェクト] タブ
- [アプリケーション] タブ (InstallScript プロジェクト) (Windows Installer ベースのプロジェクトの場合)
- [アプリケーション] タブ (InstallScript プロジェクト) (InstallScript および InstallScript オブジェクト プロジェクトの場合)
- [プラットフォーム] タブ (InstallScript および InstallScript オブジェクト プロジェクトの場合)
- [言語] タブ
- [メンテナンス] タブ (InstallScript プロジェクト)
- [言語非依存オブジェクトのプロパティ] タブ (InstallScript オブジェクト プロジェクト)
- [言語固有オブジェクトのプロパティ] タブ (InstallScript オブジェクト プロジェクト)

[プロジェクト] タブ

[設定] ダイアログ ボックスの [プロジェクト] タブには、現在開いているプロジェクトについての基本情報が表示されます。このダイアログ ボックスは、[プロジェクト] メニューの [設定] コマンドを選択すると開きます。

作成者

(オプション) プロジェクト作成者の名前を入力します。

Comments

(オプション)プロジェクトについてのコメントがあれば入力します。

[アプリケーション] タブ (Windows Installer ベースのプロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[設定] ダイアログ ボックスの [アプリケーション] タブでは、現在開いているプロジェクトがインストールする製品についての情報が表示され、その内容を変更できます。このダイアログ ボックスは、[プロジェクト] メニューから [設定] コマンドをクリックすると開きます。

テーブル 44 • [アプリケーション] タブの設定

設定	説明
製品コード	この製品を一意に識別する GUID を表示します。InstallShield を使って、自動的に異なる GUID を生成するには、この設定の横にある [変更] ボタンをクリックします。 このコードは製品を一意的に識別するため、リリースを既に配布している場合は製品コードの変更はお薦めできません。 詳細については、「Windows Installer ベースのプロジェクトで製品コードを設定する」を参照してください。
アップグレード コード	この設定は、製品のアップグレード コードに使用する GUID を表示します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 アップグレード コードは関連のある一連の製品を識別する GUID です。Windows Installer は、インストール済みの製品のメジャー アップグレードを行う際に、製品のアップグレード コードを使用します。UpgradeCode プロパティに格納されるアップグレード コードは、製品のすべてのバージョンにおいて同一でなくてはなりません。 詳細については、「アップグレード コードを設定する」を参照してください。
製品名	製品の名前を入力します。 製品名がどのように使用されるかについては、「製品名の指定」を参照してください。
製品バージョン	製品のバージョン番号を入力します。バージョン番号には、*aaa.bbb.ccccc* という形式で、数字のみを利用します。ここで、 *aaa* はメジャーバージョン番号、*bbb* はマイナーバージョン番号、そして *ccccc* はビルド番号を表します。*aaa* と *bbb* の最大値は 255 です。*ccccc* の最大値は、65,535 です。 詳細については、「製品バージョンを指定する」を参照してください。
アプリケーションの種類:	アプリケーションの種類をリストから選択するか、リストにない場合はアプリケーションの種類の名前を入力します。この情報は、プロジェクト ファイルに格納され、参照目的でのみ使用できます。エンド ユーザーに表示されることはありません。

ヒント: *[一般情報] ビューで前述の設定を構成することもできます。*

[アプリケーション] タブ (InstallScript プロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

[設定] ダイアログ ボックスの [アプリケーション] タブでは、現在開いているプロジェクトがインストールする製品についての情報が表示され、その内容を変更できます。このダイアログ ボックスは、[プロジェクト] メニューから [設定] コマンドをクリックすると開きます。

テーブル 45 • [アプリケーション] タブの設定

設定	説明
製品コード	この製品を固有に識別する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ({...}) をクリックします。 製品コードは、アンインストールまたはメンテナンスを元のインストールと関連付けるのに使用されます。新しい GUID は、(既存のプロジェクトのコピーを含む) 作成した各新規プロジェクトに対して自動的に生成されます。プロジェクトの製品コードを変更すると、以前の GUID を回復することはできません。このため、プロジェクトの製品コードの変更は、通常必要なく、変更する場合は注意が必要です。 詳細については、「InstallScript ベースのプロジェクトで製品コードを設定する」を参照してください。
製品名	製品の名前を入力します。 製品名がどのように使用されるかについては、「製品名の指定」を参照してください。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。* *製品名は、InstallScript システム変数 IFX_PRODUCT_NAME に格納されます。*

テーブル 45 • [アプリケーション] タブの設定

設定	説明
製品バージョン	製品のバージョン番号を入力します。バージョン番号には、*aaa.bbb.ccccc* という形式で、数字のみを利用します。ここで、 *aaa* はメジャーバージョン番号、*bbb* はマイナーバージョン番号、そして *ccccc* はビルド番号を表します。*aaa* と *bbb* の最大値は 255 です。*ccccc* の最大値は、65,535 です。 詳細については、「製品バージョンを指定する」を参照してください。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。*
会社名	会社の名前を入力します。この値は、(文字列 エントリ、COMPANY_NAME が存在しない場合) デフォルト スクリプトで TARGETDIR を設定するのに使用されます。これは、実行時に **MediaGetData** 関数を呼び出して取得することができます。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。*
実行可能ファイル	アプリケーションのメインの実行可能ファイルの名前を入力します。この値は、**MediaGetData** 関数を呼び出して実行時に取得できます。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。*
アプリケーションの種類:	アプリケーションの種類をリストから選択するか、リストにない場合はアプリケーションの種類の名前を入力します。この情報は、プロジェクト ファイルに格納され、参照目的でのみ使用できます。エンド ユーザーに表示されることはありません。
URL	製品 URL を入力します。この情報は、プロジェクト ファイルに格納され、参照目的でのみ使用できます。エンド ユーザーに表示されることはありません。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。*

ヒント: *[一般情報] ビューで前述の設定を構成することもできます。*

注意: *[製品バージョン] ボックスにはヌル以外の値を指定しなくてはなりません。ヌルを指定すると、ビルドエラー -7044 が発生します。*

[プラットフォーム] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

[設定] ダイアログ ボックスの [プラットフォーム] タブでは、プロジェクトでコンポーネントまたはリリースにその後オペレーティング システムを選択するときの選択肢に加えるプラットフォームを指定します。

メモ: *プロジェクト レベルでプラットフォームを指定しても、インストールを実行するターゲット は作成されません。InstallScript プロジェクトでターゲット を作成する場合、ターゲットのオペレーティング プラットフォームを識別する SYSINFO 構造を使用することができます。*

次のテーブルは、[プラットフォーム] タブで表示される設定です。

テーブル 46 : プロジェクトのための [プラットフォーム] ダイアログ ボックスの設定

設定	説明
プロジェクトは、InstallShield がサポートするすべてのプラットフォームをサポートする	InstallShield の使用中、次の設定でサポートされている実行時のプラットフォームをすべて一覧表示する場合、このオプションを選択します: • コンポーネント レベルの"オペレーティング システム" 設定 (InstallScript プロジェクトと InstallScript MSI プロジェクト) • リリース レベルの "プラットフォーム" サポート (InstallScript プロジェクト) このオプションを選択すると、プロジェクト内の特定のコンポーネントまたはリリースがいずれかのサポートされているプラットフォームでターゲットされるように指定できます。

テーブル 46 • プロジェクトのための [プラットフォーム] ダイアログ ボックスの設定 (続き.)

設定	説明
プロジェクトは、以下で選択されたプラットフォームのみをサポートする	InstallShield の使用中、次の設定で表示されているプラットフォームの一部のみを一覧表示する場合、このオプションを選択します: • コンポーネント レベルの"オペレーティング システム" 設定 (InstallScript プロジェクトと InstallScript MSI プロジェクト) • リリース レベルの "プラットフォーム" サポート (InstallScript プロジェクト) 表示するダイアログを選択します。 ***ヒント:*** *連続する複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択したあと SHIFT キーを押しながら最後のオペレーティング システムを選択します。連続しない複数のオペレーティング システムを選択するには、最初のオペレーティング システムを選択してから CTRL キーを押しながらその他の各オペレーティング システムを選択します。* 一般的に、プロジェクト レベルのこの設定で表示されないプラットフォームがあるとき、プロジェクト内にある特定のコンポーネントまたはリリースをこのプラットフォームにターゲットするという指定ができなくなります。

[言語] タブ

[設定] ダイアログ ボックスにある [言語] タブでは、開いているプロジェクトに含まれている言語のリストを表示および変更できます。このダイアログ ボックスは、[プロジェクト] メニューから [設定] コマンドをクリックすると開きます。

チェック ボックス

言語の横のチェック ボックスがチェックされていると、その言語がプロジェクトに含まれていることを示します。チェックされていない場合、その言語は含まれていません。プロジェクトに言語を追加したりプロジェクトから言語を削除するには、チェック ボックスをクリックして希望の状態に切り替えてください。

プロジェクトから言語を削除すると、その言語を使用していたすべてのコンポーネントまたはリリースも削除されます。

[メンテナンス] タブ

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

[設定] ダイアログ ボックスの [メンテナンス] タブでは、エンドユーザーがセットアップを再実行したときの動作オプションを選択できます。このダイアログ ボックスは、[プロジェクト] メニューから [設定] コマンドをクリックすると開きます。

複数インスタンスのオプションを使うと、エンドユーザーはセットアップを複数回、メンテナンスセットアップではなく最初のセットアップとして実行できます。これらのオプションの詳細については、「InstallScript インストールを複数回実行する」を参照してください。

[言語非依存オブジェクトのプロパティ] タブ

プロジェクト

[言語非依存オブジェクトのプロパティ] タブにはオブジェクト名、オブジェクトの作成会社、オブジェクトの場所などの、オブジェクトの基本的な情報が含まれています。このダイアログ ボックスは、InstallScript オブジェクト プロジェクトで [プロジェクト] メニューの [設定] コマンドを選択すると開きます。このダイアログ ボックスはまた、InstallScript プロジェクトで [オブジェクト] ビュー内からオブジェクトを登録したとき、または [オブジェクト] ビューでオブジェクトを右クリックしたときにも開きます。その場合、すべてのプロパティは編集不可能です。

オブジェクト ウィザード

このオプションでは、必要であればオブジェクトのデフォルトの言語で使用するウィザードの種類を指定できます。以下のウィザードを選択できます。

- **ウィザードなし** – オブジェクトにウィザードが不要な場合はこのオプションを選択します。
- **InstallShield オブジェクト ストック ウィザードを使用する** – オブジェクトに設定したプロパティに基づいて InstallShield がウィザードを作成するようにするには、このオプションを選択します。すべての読み取り専用プロパティが表示されますが、編集はできません。すべての読み込み／書き込みプロパティがユーザーにより変更可能です。書き込み専用プロパティは表示されません。

  ***メモ・** InstallShield ストックウィザードは配列プロパティをサポートしません。オブジェクトのプロパティが配列プロパティを必要とない場合、固有のウィザードを作成する必要があります。*

- **カスタム ウィザードを使用する**—自分で作成したウィザードを使用する場合、このオプションを選択します。InstallShield では、Visual C++ または Visual Basic のいずれかを使用してウィザードを作成できます。これら 2 つのプログラムのいずれかによって作成されたウィザードを使用するには、作成した DLL へのパスを入力するか、[参照] ボタンをクリックして目的のファイルに移動します。

次のコントロールは、[プロジェクト] メニューの [設定] コマンドを選択すると表示されます。

会社

このオブジェクトを作成した会社の名前を入力します。

バージョン

オブジェクトのバージョンを入力します。

メディアファイル

オブジェクトのメディア ファイル (Data1.hdr ファイル) へのパスを入力するか、[参照] ボタンをクリックしてメディアファイルに移動します。

[言語固有オブジェクトのプロパティ] タブ

プロジェクト

[言語固有オブジェクトのプロパティ] タブにはオブジェクト名、オブジェクトの作成会社、オブジェクトの場所などの、オブジェクトの基本的な情報が含まれています。このダイアログ ボックスは、InstallScript オブジェクト プロジェクトで [プロジェクト] メニューの [設定] コマンドを選択すると開きます。このダイアログ ボックスはまた、InstallScript プロジェクトで [オブジェクト] ビュー内からオブジェクトを登録したとき、または [オブジェクト] ビューでオブジェクトを右クリックしたときにも開きます。その場合、すべてのプロパティは編集不可能です。

IDE言語

オブジェクト情報を指定する言語を選択します。選択可能なオプションは [英語] と [日本語] です。

デフォルトを使用

デフォルトの言語の英語が選択されている場合は無効になります。このボックスをチェックすると、選択言語で表示したときにデフォルトの言語で指定したのと同じ情報がそのオブジェクトで指定されます。

*以下のコントロールは、***[デフォルトを使用する]***チェック ボックスが選択解除***されている場合のみ有効になります。

表示名

オブジェクトギャラリーに表示するオブジェクトの名前を入力します。

短い名前

必要に応じて、オブジェクト名の省略形を入力します。[表示名] フィールドに入力した名前と同じ名前を入力できます。

HTML ヘルプ

このオブジェクトのヘルプファイルへのパスを入力するか、[参照] ボタンをクリックしてこのファイルに移動します。このヘルプファイルは単一の .htm ファイルで構成されなければなりません。

アイコン ファイル

オブジェクトのアイコンファイルへのパスを入力するか、[参照] ボタンをクリックしてこのファイルに移動します。選択したアイコンがオブジェクトギャラリーのオブジェクトの隣に表示されます。オブジェクトに使用するアイコンは、最大16色で、16x16のサイズであることが必要です。

[キー名の変更] ダイアログ ボックス

[キー名の変更] ダイアログ ボックスは、[競合の解決] ダイアログ ボックスの [名前の変更] を選択して [編集] をクリックすると表示されます。[キー名の変更] ダイアログ ボックスの設定には、エクスポートしたダイアログ、コンポーネント、カスタム アクションの新しいキー値を指定できます。エクスポートする項目の名前を変更する場合、競合を避ける為、エクスポート先のプロジェクトファイルのオブジェクトの名前と重複しないようにしてください。

設定

テーブル 47 • [キー名の変更] ダイアログ ボックスの設定

設定	説明
キー値	項目の名前を変更するには、このフィールドに新規のキー値あるいは名前を入力して、[OK] をクリックします。

[必要な機能] ダイアログ ボックス

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

[必要な機能] ダイアログ ボックスでは、現在の機能がインストールされる時に必ずインストールする機能を指定できます。このダイアログ ボックスは、[セットアップのデザイン] ビューまたは [機能] ビューで、機能の ″必要な機能″ 設定にある省略記号ボタン (...) をクリックすると起動します。

[必要な機能] ダイアログ ボックスには、プロジェクトに含まれるすべての機能を表示する [*SelectedFeatureName* に必要な機能] ボックスがあります。現在の機能が必要とする各機能のチェック ボックスを選択します。

プロジェクト• *InstallScript MSI プロジェクトでは、現在の機能について、このチェック ボックスが自動的に選択されます。InstallScript および InstallScript オブジェクト プロジェクトでは自動的に選択解除されます。どちらの場合も、選択状態を変更することは不可能で、その機能がインストールされるかどうかには影響しません。*

[競合の解決] ダイアログ ボックス

[競合の解決] ダイアログ ボックスは、ダイアログ、コンポーネント、またはカスタム アクションを別のプロジェクトにエクスポートする時、または、DIM をインストール プロジェクトにインポートする時に発生する可能性がある競合を判別します。ターゲット プロジェクトに同一の名前が付いているアイテムがあり、エクスポートするアイテムとは異なる値が設定に指定されている場合、競合が発生します。こうした競合を解決するには、オプションの 1 つを選択して [OK] をクリックするか、キー列の項目の名前を変更します。

[キャンセル] をクリックすると、そのアイテムは別のプロジェクトにエクスポートされません。また、DIM の場合、インストール プロジェクトにインポートされません。

ダイアログ ボックスの設定

テーブル 48 • [競合の解決] ダイアログ ボックスの設定

設定	説明
テーブル	テーブルには、ダイアログ、コントロール、コンポーネント、カスタム アクション、その他のアイテムに関するデータがすべて表示されます。 あるアイテムを別のプロジェクトにエクスポートする場合、Source Data 列では、現在のプロジェクトから別のプロジェクトにエクスポートするアイテムの設定の値が表示されます。Target Data 列では、エクスポートするプロジェクト内に既に存在するアイテムの設定の値が表示されます。 DIM プロジェクトをインストール プロジェクトにインポートする場合、Source Data 列には、DIM プロジェクト内のアイテムの設定の値が表示されます。Target Data 列では、エクスポートするプロジェクト内に既に存在するアイテムの設定の値が表示されます。 レコード中のキー列には、識別用にキーアイコンが表示されます。異なる値間で競合が存在する場合、その列が赤色で表示されます。競合の解決で何を選択したかによって、既存のアイテムとその設定を上書きするかどうかが判別されます。
解決	競合の解決方法を選択します。選択可能なオプションは以下のとおりです: • **スキップ**—Source Data 列の値を使用します。競合がまだ残っている場合、次の競合が表示されます。 • **すべてスキップ**—現在の競合および残りすべての競合に対して、Source Data 列の値を使用します。 • **上書き**—Target Data 列の値を使用します。競合がまだ残っている場合、次の競合が表示されます。 • **すべて上書き**—現在の競合および残りすべての競合に対して、Target Data 列の値を使用します。 • **名前の変更**—このアイテムで競合が発生しないように、キー列に別の値を使用します。このオプションを使用するには、名前を変更するキーを含む行を選択し、[名前の変更] オプションを選択してから、[編集] ボタンをクリックします。[キー名の変更] ダイアログ ボックスが開き、キー列に新しい値を指定できます。

[名前を付けて保存] ダイアログ ボックス

[ファイル] メニューで、[名前を付けて保存] をクリックして [名前を付けて保存] ダイアログ ボックスを開きます。こうして、開いたプロジェクトのコピーをテンプレートとして、または新しいプロジェクトとして別の名前をつけて新しい場所に保存することができます。

ダイアログ ボックスの設定

[名前を付けて保存] ダイアログ ボックスは標準 Windows ダイアログ ボックスです。次の一覧に含まれていないコントロールについて詳細ヘルプを表示するには、コントロールを選択して F1 を押します。

テーブル 49 • InstallShield 固有の [名前を付けて保存] ダイアログ ボックスの設定

設定	説明
ファイルの種類	 ***プロジェクト: "ファイルの種類"*** *オプションは、次のプロジェクト タイプで使用できます:* • *基本の MSI* • *DIM* • *InstallScript* • *InstallScript MSI* • *マージ モジュール* **"ファイルの種類"** 一覧を使って、保存するファイルの種類を指定します。 現在のプロジェクトからテンプレートの作成を行うには、InstallShield テンプレート (.ist) を選択します。
[プロジェクト名] サブフォルダーを作成し、作成したフォルダーにプロジェクトを保存する	このチェック ボックスがシステム上でのプロジェクトファイルの配置場所を制御します。([オプション] ダイアログ ボックスの [ファイルの場所] タブにある) [プロジェクトの場所] ボックスで指定した場所に <プロジェクト名> サブフォルダーを作成して、プロジェクトをサブフォルダーに保存するには、このチェック ボックスを選択します。InstallScript が含まれるプロジェクトの場合、このサブフォルダーには [スクリプト ファイル] フォルダーも作成されます。 .ism プロジェクトをプロジェクトの場所ボックスで指定した場所に保存するには、このチェック ボックスをクリアします。
新しいプロジェクト GUID を作成し、保存されたプロジェクトに割り当てる	新しいプロジェクトの GUID をオリジナル プロジェクトとは別の GUID にするには、このチェック ボックスを選択します。新しいプロジェクトの GUID をオリジナルプロジェクトと同じ GUID にするには、このチェック ボックスをクリアします。
新しいプロジェクト名に基づいて、プロジェクト設定を適切に更新する	**[ファイル名]** ボックスで指定した名前を、新しいプロジェクトの "一般情報" ビューにある "プロジェクト名" 設定の値として使うには、このチェック ボックスを選択します。新しいプロジェクトの "名前" 設定の値を元のプロジェクトと同じにするには、このチェック ボックスをクリアします。

[スクリプト変換] ダイアログ ボックス

InstallShieldでは、InstallScript 関数名に異なる用語を使用しますたとえば InstallShield で使用されている*「機能」*という用語は、InstallShield Professional では*「コンポーネント」*と呼ばれていたものです。詳細については、「スクリプトの変更: 語彙変換」を参照してください。

InstallShield で .ipr ファイルを開くと、このダイアログが開き、InstallShield で自動的にスクリプト (.rul) ファイルをアップデートして InstallShield 用語を使用するようにするかたずねるメッセージが表示されます。

メモ: *InstallShield ヘルプ ライブラリとその他のマニュアルでは、InstallShield スクリプト用語を使用しています。*

[スクリプト エディターのプロパティ] ダイアログ ボックス

[スクリプト エディターのプロパティ] ダイアログ ボックスを使って、InstallShield の様々なビューでスクリプト コードを表示する方法を指定できます。このダイアログ ボックスは、スクリプト ファイルを編集できるビューで使用できます (たとえば、[InstallScript]、[SQL スクリプト]、[カスタム アクションとシーケンス] ビュー)。

[スクリプト エディターのプロパティ] ダイアログ ボックスを開くには、[スクリプト エディター] ペインを右クリックしてから [プロパティ] をクリックします。

ヒント: *スクリプト エディターのプロパティは、ユーザーごとのグローバル設定で、すべての InstallShield スクリプト エディターに反映されます。たとえば、すべてのスクリプト エディターで同じフォントが使用されます。[SQL スクリプト] ビューでフォントを変更すると、次回 InstallScript が [InstallScript] ビューに再ロードされたとき、そのビューのフォントも変更されています。*

[スクリプト エディターのプロパティ] ダイアログ ボックスに表示される設定は、次の主要なカテゴリに分かれています:

- 全般
- 色

[全般] の設定

[スクリプト エディターのプロパティ] ダイアログ ボックスの [全般] 領域には、以下の設定があります。

テーブル 50 · [スクリプト エディターのプロパティ] ダイアログ ボックスの [全般] 設定

設定	説明
フォント	この設定は、スクリプト エディターのコンテンツに使用されるフォント、スタイル、およびサイズを表示します。選択されたオプションを変更するには、この設定の省略記号ボタンをクリックします。[フォント] ダイアログ ボックスが開いて、スクリプト コンテンツのフォント、スタイル、およびサイズを確認および変更することができます。
構文ハイライト	すべてのスクリプト エディターで表示される様々なスクリプト要素を色で識別するかどうかを指定します。[はい] を選択すると、[スクリプト エディターのプロパティ] ダイアログ ボックスの [色] 領域にある設定で指定した選択にしたがって、スクリプト要素が色で識別されます。 詳細については、「スクリプト エディターで構文ハイライトの色を変更する」を参照してください。

テーブル 50 • [スクリプト エディターのプロパティ] ダイアログ ボックスの [全般] 設定 (続き.)

設定	説明
オートコンプリート	入力中の文字で始まる関数、キーワード、定数、その他のスクリプト用語をアルファベット順でポップアップ リストに表示するかどうかを指定します。オートコンプリート機能は、[InstallScript] ビューに文字列定数演算子 (@) を入力したときに、使用可能な文字列識別子のリストも表示します。 デフォルトの値は [はい] です。 オートコンプリートを使うと、コードを手入力する手間が省けるため、作業効率が上がります。また、コードのスペル ミスを回避することができます。 詳細については、「スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う」を参照してください。
ローカル変数を含める	選択されたスクリプトで定義されているすべてのローカル変数をポップアップ リストに表示するかどうかを指定します。デフォルトの値は [はい] です。 [いいえ] を選択した場合、[InstallScript] ビューの中央ペインにある関数、プロパティ、およびメソッド フォルダーには、スクリプトファイルからの関数、プロパティ、またはメソッドが一覧表示されません。 この設定で [はい] を選択して、[InstallScript] ビューでコードを入力しているときにパフォーマンスに影響が出た場合は、この設定を [いいえ] に設定します。 **"オートコンプリート"** 設定に [いいえ] を選択すると、この設定は無視されます。 詳細については、「スクリプト エディターでのコードの書き込みにオートコンプリート機能を使う」を参照してください。
関数呼び出しのヒントを表示	スクリプトで関数呼び出しを入力するときに、ツールヒントの一種である関数呼び出しのヒントを表示するかどうかを指定します。関数呼び出しのヒントは、関数のパラメーターに関する情報を表示します。その他、関数の説明や、入力しているパラメーターについての説明も表示します。デフォルトの値は [はい] です。 詳細については、「スクリプト エディター内で InstallScript 関数について関数呼び出しのヒントを表示する」を参照してください。
行番号指定	スクリプト エディターの左マージンに、行番号を表示するかどうかを指定できます。デフォルト値は [いいえ] です。 詳細については、「スクリプト エディター内で表示されているスクリプトの行番号へジャンプする」を参照してください。

テーブル 50 • [スクリプト エディターのプロパティ] ダイアログ ボックスの [全般] 設定 (続き.)

設定	説明
構文の折りたたみ	スクリプトで構文の折りたたみ行うかどうかを指定します。[はい] を選択すると、スクリプトで展開可能または折りたたみ可能なブロックで始まるコードの各行の横のマージンに、プラス (+) またはマイナス (–) 記号が追加されます。プラス記号をクリックすると、非表示のコードが展開されます。マイナス記号をクリックすると、コードが非表示になります。 デフォルト値は [いいえ] です。 構文の折りたたみ機能を使って、長いスクリプトを縮小することで、現在行っている作業に関連のあるコードに焦点を当てることができます。また、スクリプトの全体的な構造を確認するのに便利です。
自動字下げ	スクリプトでコードの新しい行を字下げするかどうかを指定します。デフォルトの値は [はい] です。 詳細については、「スクリプト エディターで自動インデント機能を有効または無効にする」を参照してください。
空白スペースを表示	スクリプト内で空白スペースがある場所に記号またはその他の印を表示するかどうかを指定します。たとえば、[はい] を選択すると、スクリプト内の各スペースは中黒 (·)、各タブは矢印記号で表示されます。また、空白スペースが有効な場合は、改行も表示されます。また、空白スペースが有効な場合は、改行も表示されます。 デフォルト値は [いいえ] です。
タブ幅	タブ サイズを文字サイズの倍数として指定します。たとえば、タブのスペースを 5 に設定した場合、Tab キーを押すとカーソルは 5 文字分のスペース移動します。
空白で置換	前後に移動させる文字スペースの数を指定します。たとえば、[はい] を選択した場合、1 文字スペース分後ろにカーソルが移動し、タブ幅の選択に従って、適宜カーソルが前に移動します。[いいえ] を選択すると、タブ幅の選択に従って、適宜カーソルが前後移動します。

色の設定

[スクリプト エディターのプロパティ] ダイアログ ボックスの [色] 領域では、様々なスクリプト要素の前景、または一部の状況では背景色も確認および変更できます。

[スクリプト エディターのプロパティ] ダイアログ ボックスの [色] 領域には、以下の設定があります。

テーブル 51 • [スクリプト エディターのプロパティ] ダイアログ ボックスにおける色の設定

設定	説明
Default	スクリプト用語で定義済みでない種類のテキストの色を指定します。
選択	スクリプト エディターで選択されたテキストの色を指定します。

テーブル 51 • [スクリプト エディターのプロパティ] ダイアログ ボックスにおける色の設定 (続き.)

設定	説明
カーソル	点滅カーソルの色を指定します。
キーワード	ビルトイン関数、データ型、および定数などのスクリプト キーワードの色を指定します。
識別子	識別子を表示するために使用する色を指定します。これには変数が含まれます。
演算子	演算子を表示するために使用する色を指定します。演算子の例として、丸括弧、角括弧、中括弧、算術記号 (+、ー、＊、/ など)、およびコンマが含まれます。
Comments	コード内のコメントに使用する色を指定します。コメントには、コメント区切り文字 (/*、*/、または // など) が含まれます。
文字列	文字列の色を指定します。文字列には、引用句が含まれます。
未終了の文字列	終了引用符で終わっていない文字列の色オーバーライドを指定します。これによって、コンパイル エラーの原因となる可能性のある文字列を見つけやすくなります。
プリプロセッサ命令	プリプロセッサ命令を表示するために使用する色を指定します。プリプロセッサ命令には、# 記号で始まる任意の行が含まれます。
定数	スクリプト定数を表示するために使用する色を指定します。
ラベル	ラベルを表示するために使用する色を指定します。
エラー	スクリプト エラーを表示するために使用する色を指定します。
行番号	行番号が有効なときに、左マージンに表示される行番号に使用する色を指定します。
関数呼び出しのヒント	InstallScript 関数の呼び出しのヒントに使用する色を指定します。
一致する括弧	始まりの括弧と、それに対応する閉じ括弧 { } を表示するのに使用する色を指定します。これは、挿入ポイントが丸括弧、角括弧、または中括弧内にあるときに常に使用されます。構文ハイライトには、ネストが考慮されます (例、**Foo(array[array.GetLength() – 1])**)。
一致しない括弧	始まりの括弧を表示するのに使用する色を指定します {対応する閉じ括弧が不足していることを示します}。これは、閉じ括弧が不足している始まりの丸括弧、各括弧、中括弧の後に挿入ポイントがあるときに常に使用されます。構文ハイライトには、ネストが考慮されます (例、**Foo(array[array.GetLength() – 1])**)。

スクリプト エディターの色を変更する方法については、「スクリプト エディターで構文ハイライトの色を変更する」を参照してください。

[SCM シャットダウンの遅延] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[SCM シャットダウンの遅延] ダイアログ ボックスを使って、他のシャットダウン タスクを進める前にサービス コントロール マネージャーが待機する時間を指定できます。次の状況下で、このダイアログ ボックスが開きます:

- [サービス] ビュー、または [セットアップ デザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューにあるコンポーネントの [詳細設定] 領域でサービスを構成しているとき。
- 右側ペインに表示される設定の [構成の設定] カテゴリにあるイベントの "構成の種類" 設定で、**[SCM シャットダウンの遅延を構成する]** オプションが選択されていて、そのイベントの "引数" 設定にある省略記号ボタン (...) をクリックしたとき。

適切なオプションを選択します。遅延時間をデフォルトの 3 分間にリセットするには、[デフォルト] オプションを選択します。

サービス コントロール マネージャーは、SERVICE_CONTROL_PRESHUTDOWN 通知をサービスに送信したあと、指定された時間待機します。変更は、次回のシステム開始時に有効になります。

[セキュリティ記述子] ダイアログ ボックス

"セキュリティ記述子" 設定にある省略記号ボタン (...) をクリックすると、[セキュリティ記述子] ダイアログ ボックスが開きます。この設定は、[サービス] ビューに追加したサービス、または [セットアップのデザイン] ビュー (インストール プロジェクトの場合)、または [コンポーネント] ビューで、コンポーネントの [詳細設定] 領域にある [サービス] ノードに追加したサービスにアクセス許可を構成するとき、"アクセス許可" 設定の下に表示されます。

[セキュリティ記述子] ダイアログ ボックスでは、有効なセキュリティ記述子定義言語 (SDDL) を使って、サービスのアクセス許可に有効な SDDL 文字列を入力します。SDDL についての詳細は、Microsoft Windows SDK (Software Development Kit) の "Access Control" セクションを参照してください。

ヒント: *山かっこ (例、<DomainName¥UserName>)または環境変数 (例、[%UserDomain][%UserName]) を使って、実行時にアカウント SID が決定されるユーザーのドメインとユーザー名を指定することができます。*

メモ: *アクセス許可の設定は Windows Installer 5 よりサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。*

1 つのインストールに MsiLockPermissionsEx *テーブルと* LockPermissions *テーブルの両方を含めることはできません。サービスに "アクセス許可" 設定およびその下にある設定を使うと、パッケージの* MsiLockPermissionsEx *テーブルを構成していることになります。[一般情報] ビューの "ロックダウンの設定方法" 設定で* ***[従来型の Windows Installer 処理]*** *を選択した場合に、プロジェクト内の 1 つ以上のファイル、フォルダー、またはレジストリ キーにアクセス許可を設定すると、パッケージの* LockPermissions *テーブルを構成していることになります。*

[ファイルの選択] ダイアログ ボックス

[ファイルの選択] ダイアログ ボックスを使って、[ダイレクト エディター] の DuplicateFile テーブルにファイルを入力します。

ダイアログ オプション

コンポーネント

このテーブルに入力するファイルが含まれるコンポーネントを選択します。

ファイル (*.*)

このテーブルに含めるファイルを選択します。

[メディアの場所選択] ダイアログ ボックス

MSI データベースの場合、新しいファイルをソース メディアに配置する場所を選択します。.cab ファイルを作成している場合、ファイルのパッケージへのストリームを選択できます。

プロジェクト: *MSM データベースの場合、[ファイルの場所] オプションは利用不可能です。これは、マージ モジュールが .msm データベースの内部となる .cab ファイルにファイルを格納する必要があるためです。*

メモ: *COM 情報を抽出するオプションは、ファイルをフォルダーにドラッグしている場合や、既存のファイルがないコンポーネントに単一のファイルを追加している場合などにのみ有効になります。その結果、COM ファイルが特定のコンポーネントに含まれる唯一のファイルとなります。*

[SQL Server の選択] ダイアログ ボックス

[SQL Server の選択] ダイアログ ボックスは、データベース インポート ウィザードの使用中、SQL Server パネルにあるサーバー名で [参照] ボタンをクリックしたときに表示されます。

[SQL Server の選択] ダイアログ ボックスでは、データベースをインポートする SQL Server を選択することができます。

[文字列の選択] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

[文字列の選択] ダイアログ ボックスに、プロジェクトのデフォルト言語用の言語非依存 ID 一覧と、対応する言語固有値が表示されます。このダイアログ ボックスでは、ハードコード化されたテキスト文字列の代わりに使用できる文字列 ID を選択または作成することができます。

[文字列の選択] ダイアログ ボックスへのアクセス

[文字列の選択] ダイアログ ボックスは、[InstallScript] ビューの [スクリプト エディター] ペイン内からアクセスできます。また、InstallShield 内のその他の場所でも利用できます。たとえば、ローカライズ可能なテキストを入力可能な設定で省略記号ボタン (...) をクリックすると、[文字列の選択] ダイアログ ボックスが開き、その設定で使用できる文字列エントリを選択、または作成することができます。ローカライズ可能なテキストを入力可能な設定の例として、"表示名" と "説明" 設定が挙げられます。

[文字列の選択] ダイアログ ボックスでの作業

[文字列の選択] ダイアログ ボックスは、2 つの要素で構成されています:

- ボタン行とその他のコントロール
- グループ ボックス領域 (ボタン行の下)
- スプレッドシート形式のテーブル

テーブルの各行は、プロジェクトに含まれる各文字列エントリを示します。

次の表では、[文字列の選択] ダイアログ ボックスに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 52 : [文字列の選択] ダイアログ ボックスのコントロール

コントロールの名前	説明
新規作成	[文字列エントリ] ダイアログ ボックスを表示します。このダイアログ ボックスでは、新しい文字列 ID、値、および内部コメントを指定できます。
Delete	選択された文字列エントリ (複数選択可) を削除します。
検索グリッド	このテキスト ボックスで指定した文字列に従って、[文字列の選択] ダイアログ ボックスに表示される文字列をフィルターします。

テーブル 52 • [文字列の選択] ダイアログ ボックスのコントロール (続き.)

コントロールの名前	説明
グループ化する列のヘッダーをここにドラッグ	このグループ ボックス領域を使って、[文字列の選択] ダイアログ内の行をグループにまとめます。このダイアログ ボックスでは、列のヘッダーをグループ ボックスにドラッグ アンド ドロップするだけで複数階層にグループ化することができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。

次の表は、[文字列の選択] ダイアログ ボックスの各列について説明します。

テーブル 53 • [文字列の選択] ダイアログ ボックスの列

列	説明
Identifier	この列には、その文字列の言語非依存 ID が含まれます。プロジェクトに含まれる各文字列 ID は、1 つまたは複数の値にリンクされています。
Value	この列には、ランタイム文字列が表示されます。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトでは、一部の文字列値は角括弧でかこまれた Windows Installer プロパティを含みます (例、***Install [ProductName]***)。実行時に、このプロパティと括弧はプロパティ値で置き換えられます。* *これらの同じプロジェクト タイプに含まれる文字列値の中には、中括弧でかこまれたフォント情報を含むものもあります (例、***{&MSSansBold8}OK***)。フォント情報は、実行時に文字列を表示するのに使用するスタイルの詳細を示します。*
Comments	この列には、文字列エントリについての内部メモが含まれます。コメントは実行時には表示されません。
Modified	この列には、文字列エントリが最後に更新された日時が表示されます。

シリアル番号違反ダイアログ ボックス

[シリアル番号違反] ダイアログ ボックスは、入力されたシリアル番号が既に使用中であることを InstallShield が検出した場合に表示されますInstallShield は、固有の有効なシリアル番号が入力されない限り、20 分間開始されません。

タスク ***シリアル番号を変更するには、次の手順を実行します。***

1. **[変更]** をクリックします。**[シリアル番号]** ダイアログ ボックスが開きます。
2. 一意の有効なシリアル番号を **[シリアル番号]** ボックスに入力します。ダッシュは必要ありません。

[サーバーの場所] ダイアログ ボックス

プロジェクト: *この情報は、トランスフォーム プロジェクトに適用します。*

[一般情報] ビューの "サーバーの場所" 設定にある省略記号ボタン (...) をクリックすると、[サーバーの場所] ダイアログ ボックスが開きます。[サーバーの場所] ダイアログ ボックスでは、Windows Installer が必要な場合に製品のインストール パッケージおよび関連ファイルを見つけられるように、それらのネットワークおよび URL パスを指定できます。たとえば、エンド ユーザーがネットワーク サーバーまたは Web サイトから製品をインストールする場合で、1 つ以上の機能が、製品を初めて使用するときにインストールされるように設定されているとき、最初のインストールの後に Windows Installer がサーバーまたは Web サイト上にある .msi パッケージおよび関連ファイルにアクセスする必要があります。

InstallShield は、指定されたパスを SOURCELIST プロパティに追加します。実行時に Windows Installer は、エンド ユーザーの既存の製品のソース リストの最後に、このリストを追加します。

指定したサーバーの場所が有効かどうかは、インストールがサーバーにリモート アクセスする必要があるときに判断されます。つまり、リソースが必要なとき、サーバーが使用できない、または無効なサーバーが追加された場合、エントリが無視され、場合によってはエラーが生成されます。指定された各場所には、インストールの完全なソースが配置されなくてはなりません。各ソースの場所のディレクトリ ツリーはすべて同じであり、.cab ファイルを含み必要なソース ファイルすべてを含まなくてはなりません。各場所には、同じファイル名と製品コードを持つ .msi ファイルが必要です。

ヒント: *サーバー パスには、パーセント記号 (%) で識別される環境変数を使用できます。たとえば、次のパスは Home 環境変数を利用します:*

```
¥¥Server1¥%Home%¥Office
```

テーブル 54 : [サーバーの場所] ダイアログ ボックスの設定

設定	説明
ソース パス	このボックスは、指定されたパスのリストを表示します。
新規作成	このボタンをクリックして、[ソース パス] ボックスへの新しいパスを追加します。
Delete	このボタンをクリックして、[ソース パス] ボックスから選択されたパスを削除します。
上へ移動	このボタンをクリックして、パスのリスト内で、選択されたパスを上に移動させます。
下へ移動	このボタンをクリックして、パスのリスト内で、選択されたパスを下に移動させます。

自動開始サービスの遅延に関する [サービスのオプション] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[サービスのオプション] ダイアログ ボックスを使って、構成中の Windows サービスをその他の自動開始サービスと遅延時間が過ぎた後に開始するかどうかを指定できます。次の状況下で、このダイアログ ボックスが開きます:

- [サービス] ビュー、または [セットアップ デザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューにあるコンポーネントの [詳細設定] 領域でサービスを構成しているとき。
- 右側ペインに表示される設定の [構成の設定] カテゴリにあるイベントの "構成の種類" 設定で、**[自動開始の遅延時間を構成する]** オプションが選択されていて、そのイベントの "引数" 設定にある省略記号ボタン (...) をクリックしたとき。

適切なオプションを選択します。

変更は、次回のシステム開始時に有効になります。

回復操作の実行のための [サービスのオプション] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[サービスのオプション] ダイアログ ボックスを使って、構成中のサービスで回復操作を行う条件を指定できます。次の状況下で、このダイアログ ボックスが開きます:

- [サービス] ビュー、または [セットアップ デザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューにあるコンポーネントの [詳細設定] 領域でサービスを構成しているとき。

- 右側ペインに表示される設定の [構成の設定] カテゴリにあるイベントの "構成の種類" 設定で、**[回復操作の実行条件を構成する]** オプションが選択されていて、そのイベントの "引数" 設定にある省略記号ボタン (...) をクリックしたとき。

適切なオプションを選択します。

変更は、次回のシステム開始時に有効になります。

[サービス SID] ダイアログ ボックス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[サービス SID] ダイアログ ボックスを使って、追加する適切なサービス セキュリティ識別子 (SID) を指定できます。次の状況下で、このダイアログ ボックスが開きます:

- [サービス] ビュー、または [セットアップ デザイン] ビュー (インストール プロジェクトのみ) または [コンポーネント] ビューにあるコンポーネントの [詳細設定] 領域でサービスを構成しているとき。
- 右側ペインに表示される設定の [構成の設定] カテゴリにあるイベントの "構成の種類" 設定で、**[プロセス トークンにサービス SID の種類を追加する]** オプションが選択されていて、そのイベントの "引数" 設定にある省略記号ボタン (...) をクリックしたとき。

適切なオプションを選択します。

同じプロセス内で複数のサービスがホストされている場合で、1 つのサービスが制限されたサービス SID を持つ場合、すべてのサービスが制限されたサービス SID を持つ必要があります。

変更は、次回のシステム開始時に有効になります。

[ファイルの URL の設定] ダイアログ ボックス

[ファイルの URL の設定] ダイアログ ボックスは、InstallShield 前提条件エディターの [含めるファイル] タブにある **[ファイルの URL の設定]** ボタンをクリックすると開きます。ダイアログ ボックスでは、InstallShield 前提条件とするファイルを指定することができます。

テーブル 55 - [ファイルの URL の設定] ダイアログ ボックスの設定

設定	説明
ファイルの親フォルダーへの URL	選択したファイル (複数可) の親ディレクトの URL を入力します。この URL は、インストール作成者が [再配布可能ファイル] ビューを利用してインターネットからローカル マシンへ InstallShield 前提条件をダウンロードする時に InstallShield が利用する URL と同じです。 たとえば、Notepad.exe および Paint.exe というファイルを選択して、これらのファイルの URL がそれぞれ http://www.mywebsite.com/Folder1/Notepad.exe および http://www.mywebsite.com/Folder1/Paint.exe となっている場合、次の通りこのボックスに入力します。 http://www.mywebsite.com/Folder1

[設定] ダイアログ ボックス

[設定] ダイアログ ボックスでは、InstallShield が現在開いているプロジェクトのインストールをビルドする方法について、その環境を設定することができます。

タスク

設定ダイアログ ボックスを開くには、以下の手順を実行します。

[ビルド] メニューで **[ビルドの設定]** をクリックします。

次のタブがこの [設定] ダイアログ ボックスに関連しています。

- [コンパイル/リンク] タブ
- [実行/デバッグ] タブ (InstallScript プロジェクトのみ)
- [MSI ログ ファイル] タブ (Windows Installer プロジェクトのみ)

[コンパイル/リンク] タブ

[設定] ダイアログ ボックスにある [コンパイル/リンク] タブでは、[スクリプトのコンパイル] を設定します。これらの設定は、Setup.rul およびこのファイルに含まれるすべてのファイルに適用されます。

テーブル 56 - [コンパイル/リンク] タブの設定

設定	説明
プリプロセッサ定義	プリプロセッサ定義を指定します。等号およびコンマの前後に空白を置かないようにして、以下の形式を使用します。 **MYVARIABLE1=123,MYVARIABLE2=456** スクリプト内のこれらの種類の定数は、スクリプトの流れをコントロールする #if および #ifdef ステートメントを使ってテストすることができます。たとえば、この編集ボックスに MYVARIABLE と入力すると、次の #ifdef ループ中のコードが実行されます。 **#ifdef MYVARIABLE** **//コマンド** **#endif** このボックスでプリプロセッサ定義を追加または変更した後、この追加または変更を有効にするにはスクリプトをコンパイルする必要があります。 ***メモ:*** *多くの Windows API 関数はヘッダーファイル* ISRTWindows.h *で宣言され、*Ifx.h *スクリプトに含んだときに自動的に含まれます。[設定] ダイアログ ボックスの [コンパイル/リンク] タブにある [プリプロセッサ定義] ボックス内にプリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を配置することで Windows API が自動的に定義されないようにすることが可能です。*

テーブル 56 • [コンパイル/リンク] タブの設定 (続き.)

設定	説明
インクルード パス	#include ステートメントによってメイン インストールの InstallScript コードに含められたソース ファイルを探すときに InstallShield が検索するディレクトリを指定します。 以下は、この設定についてのガイドラインです。 • 複数のパスを指定するには、各パスをカンマで区切ります。 • パスでは、任意の有効なパス変数を使用することができます。 • パスの指定に、引用符は使用できません。 • 現在のディレクトリがいつも定数とはかぎらないため、指定するパスでドット (.) 演算子を使用しないでください。代わりに、*<ISProjectFolder>* のような明示的パス変数を使用して、既知のフォルダーの場所を指定してください。 • InstallShield で、この設定で指定した値が Compile.exe に渡されると、指定された文字列が自動的に解析され、コンマで区切られたパスが判別されます。また、InstallShield は、パス変数を置き換えたり、パスを引用符で囲んだり、複数の /i スイッチを使用してそれぞれのパスを別々に Compile.exe へ渡したりします。ただし、カンマの代わりにセミコロンでパスを区切った場合、InstallShield は、指定された複数のパスを単一パスと見なして、単一パス文字列をそのままコンパイラへ渡します。この場合、パス変数の置換、および引用符の付加は行われません。 ***メモ:*** *InstallShield は、標準 InstallShield パスを検索する前に、指定された任意のパスを検索します。2 つの include ファイルが同じ名前を持つときに、1 つの include ファイルがカスタム include パスにあり、別の include ファイルが標準の InstallShield include パスに存在するとき、InstallShield は標準のInstallShield include パスではなく、カスタム include パスにある include ファイルにリンクします。* 詳細については、「#include」を参照してください。
最大警告数	表示される警告メッセージの最大数を指定します。
最大エラー数	表示されるエラーメッセージの最大数を指定します。
警告レベル	出力ウィンドウに表示される警告の種類を指定します。 • **なし** – 警告メッセージは表示されません。 • **レベル1** – で処理できないシステム警告メッセージを表示します。 • **レベル 2** – レベル 1 メッセージに加えて、文字列長が制限を越えた場合のメッセージを表示します。 • **レベル 3 (デフォルト)** – すべての警告メッセージを表示します。 詳細については、「コンパイラで定数を定義する」を参照してください。

テーブル 56 • [コンパイル/リンク] タブの設定 (続き.)

設定	説明
ビルド前にコンパイルする	リリースをビルドするたびに、InstallScript を自動的にコンパイルするには、このチェック ボックスを選択します。次の動作は、このチェック ボックスが選択されている場合のみ発生します。 • [ビルド] メニューで [ビルドのリフレッシュ] コマンドを選択すると、スクリプトは、最後にコンパイルしてから変更が発生した場合のみ再コンパイルされます。 • [ビルド] メニューで [ビルド *<リリース名>*] コマンドを選択すると、最後にコンパイルしてから変更が発生していない場合も、スクリプトは再コンパイルされます。 ***メモ•*** *ビルド プロセスはビルド中、コンパイル済みのスクリプトを自動的にフォルダーにコピーするため、ビルドが始める前に、コンパイル済みのスクリプトがディスク イメージ フォルダーにコピーされることはありません。コンパイラ エラーのためスクリプトがコンパイルに失敗した場合、ビルドは続行しません。*
インライン デバッグ情報を生成する	このチェック ボックスが選択された場合、デバッグ情報がコンパイル済みスクリプト ファイルに含まれるため、デバッグ情報ファイルは必要ありません。 作成したオブジェクトを含むインストールをデバッグして、オブジェクトからデバッグ情報を取得する場合、まず先にこのチェック ボックスを選択された状態で、オブジェクトのスクリプトをコンパイルします。 ***注意 •*** *このチェック ボックスを選択すると、コンパイル済みのスクリプトのサイズが大きくなり、インストールのスピードが遅くなります。また、他人によるコードの逆アセンブルが、より発生しやすくなります。したがって、エンドユーザーに配布するための最終インストールを作成しているときは、使用しないことをお勧めします。*
警告をエラーとする	警告をエラーとして扱う場合 (つまり、InstallScript のコンパイルで警告が発生したときに、インストールを実行しない場合)、このチェック ボックスを選択します。

テーブル 56 • [コンパイル/リンク] タブの設定 (続き.)

設定	説明
追加ライブラリパス	 ***プロジェクト•*** *InstallScript および InstallScript オブジェクト プロジェクトの場合、次のディレクトリにあるスクリプト ライブラリが自動的に検索されます:* • <ISProductFolder>¥Script¥Ifx¥Lib • <ISProductFolder>¥Script¥Isrt¥Lib *基本の MSI および InstallScript MSI プロジェクトの場合、次のディレクトリにあるスクリプト ライブラリが自動的に検索されます:* • <ISProductFolder>¥Script¥Iswi¥Lib • <ISProductFolder>¥Script¥Isrt¥Lib *上述のディレクトリには、ビルトイン InstallScript 関数を定義するライブラリと、デフォルトで ライブラリ (.obl) ボックスにリストされているライブラリが含まれています。* 独自のスクリプト ライブラリを含む追加のディレクトリを検索する場合、[追加ライブラリ パス] ボックスでディレクトリ パスを指定できます。 以下は、この設定についてのガイドラインです。 • 複数のパスを指定するには、各パスをカンマで区切ります。 • パスでは、任意の有効なパス変数を使用することができます。 • パスの指定に、引用符は使用できません。 • 現在のディレクトリがいつも定数とはかぎらないため、指定するパスでドット (.) 演算子を使用しないでください。代わりに、*<ISProjectFolder>* のような明示的パス変数を使用して、既知のフォルダーの場所を指定してください。 • InstallShield で、この設定で指定した値が Compile.exe に渡されると、指定された文字列が自動的に解析され、コンマで区切られたパスが判別されます。また、InstallShield は、パス変数を置き換えたり、パスを引用符で囲んだり、複数の /libpath スイッチを使用してそれぞれのパスを別々に Compile.exe へ渡したりします。ただし、カンマの代わりにセミコロンでパスを区切った場合、InstallShield は、指定された複数のパスを単一パスと見なして、単一パス文字列をそのままコンパイラへ渡します。この場合、パス変数の置換、および引用符の付加は行われません。 ***メモ•*** *InstallShield は、標準 InstallShield パスを検索する前に、指定された任意のパスを検索します。2 つのライブラリが同じ名前を持つときに、1 つのライブラリがカスタム ライブラリ パスにあり、別のライブラリが標準の InstallShield ライブラリ パスに存在するとき、InstallShield は標準のInstallShield ライブラリ パスではなく、カスタム ライブラリ パスにあるライブラリにリンクします。*

テーブル 56 • [コンパイル/リンク] タブの設定 (続き.)

設定	説明
ライブラリ (.obl)	コンパイラがリンクするスクリプト ライブラリ (コンパイル済みの .obl ファイル) の名前を指定します。各ファイルはコンマで区切ります。 ***メモ•*** *ライブラリ パスが [追加ライブラリ パス] ボックスで明示されているか、そのパスがコンパイル時に自動的に検索される InstallShield のビルトイン ライブラリ パスである限り、各ライブラリのパスを含む必要はありません。* ***プロジェクト•*** *InstallScript プロジェクトでは、デフォルトでビルトイン ライブラリ ifx.obl および isrt.obl がリストされています。* *基本の MSI および InstallScript MSI プロジェクトでは、デフォルトでビルトイン ライブラリ iswi.obl および isrt.obl がリストされています。* *InstallScript プロジェクトでは、デフォルトでビルトイン ライブラリ ifxobject.obl および isrt.obl がリストされています。* *プロジェクトの開発時に使用する InstallShield ビューによって、その他のビルトイン ライブラリもリストされます。*

[実行/デバッグ] タブ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

[実行/デバッグ] タブで、インストールの実行とデバッグに関連する設定を行います。

テーブル 57 • [実行/デバッグ] タブの設定

設定	説明
セットアップ コマンドラインの引数	このボックス内のテキストは、インストールを InstallShield から実行した際にシステム変数 CMDLINE に書き込まれます。この変数には、スクリプトからアクセスできます。
MIF ファイルを生成する	このチェック ボックスを選択した場合、InstallShield 内からインストールが事項された時、インストールは自動的に管理情報フォーマット (.mif) ファイルを生成します。配布するインストールに .mif ファイルを生成する方法については、「[Mif] セクション」を参照してください。

テーブル 57 • [実行/デバッグ] タブの設定 (続き.)

設定	説明
MIF ファイル名	InstallShield 内からインストールを実行した際に、インストールが生成する .mif ファイルの名前を指定します。このボックスを空白にすると、.mif ファイルは Status.mif と名づけられます。
製品のシリアル番号	シリアル番号を指定した場合、InstallShield 内からインストールを実行した時、インストールは生成した .mif ファイルにそれを追加します。

[MSI ログファイル] タブ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[設定] ダイアログ ボックスにある [MSI ログ ファイル] タブを使って、InstallShield 内からインストールを実行中に利用可能なログ ファイルを作成することができます。さらに、このタブではログファイルに書き込まれる情報の種類を選択することもできます。

テーブル 58 • [MSI ログ ファイル] タブの設定

設定	説明
MSI ログ ファイル オプション	ログファイルで表示するメッセージの種類のチェック ボックスを選択します。チェック ボックスを選択するとき、対応する引数が [MsiExec.EXE コマンドライン引数] ボックスに追加されます。
MsiExec.EXE コマンドライン引数	インストールの実行時に MsiExec.exe に渡すコマンドラインの引数を入力します。チェック ボックスが MSI ログファイルのオプション領域から選択されると、対応するコマンドラインパラメーターが、このボックスに追加されます。
ログファイル	インストール実行時に作成されるログファイルの名前を入力します。 ***プロジェクト•*** *InstallScript MSI プロジェクトの場合、このボックスにログファイルへの完全パスを入力する必要があります。*

[セットアップ言語] ダイアログ ボックス

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

"セットアップ言語" ダイアログ ボックスの動作は、プロジェクトが Windows Installer ベースのプロジェクト (基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクト) か、InstallScript ベースのプロジェクト (InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクト) かによって異なります。

[一般情報] ビューの "セットアップ言語" 設定にある省略記号ボタン (...) をクリックすると、[セットアップ言語] ダイアログ ボックスが開きます。

基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、"セットアップ言語" ダイアログ ボックスを使って、[リリース] ビューの "UI 言語" 設定にリストする言語を指定します。このため、プロジェクト レベルでこのダイアログ ボックスに言語がリストされていない場合、その特定の UI 言語をプロジェクトのリリースに含めることはできません。

InstallScript および InstallScript オブジェクト プロジェクトでは、"セットアップ言語" ダイアログ ボックスを使って、プロジェクトの [コンポーネント] ビューと [リリース] ビューの "言語" 設定にリストする言語を指定します。一般的に、プロジェクト レベルでこの設定に言語がリストされていない場合、プロジェクト内の特定のコンポーネントをその言語に基づいてターゲットにすることはできません。さらに、特定の言語に基づいてコンポーネントと UI 文字列をプロジェクトのリリースに含めることもできません。

この設定を使ってプロジェクトにサポート対象言語を追加すると、その言語の文字列エントリがプロジェクトに追加されます。文字列エントリには、翻訳済みのビルトイン ユーザー インターフェイス文字列リソースが含まれます。

テーブル 59 • [セットアップ言語] ダイアログ ボックスの設定

設定	説明
Languages	特定の言語の文字列をプロジェクトに含めるには、このチェック ボックスを選択します。特定の言語用の文字列をすべて除外するには、このチェック ボックスをクリアします。 ***エディション:*** *言語サポートは、ご使用中の InstallShield のエディション、プロジェクト タイプ、および InstallShield のインターフェイス言語 (英語、日本語) によって異なります。たとえば、InstallShield Professional Edition の英語版を使用している場合、このダイアログ ボックスでで選択可能な言語は 1 つだけです。その他の言語のチェック ボックスがリストされていても、それらは無効な状態です。* *言語サポートに関する全般情報は、「InstallShield 実行時の言語サポート」を参照してください。*
選択した言語のみ表示	[言語] ボックスに、その言語のチェック ボックスが選択されているものだけを一覧表示するには、このチェック ボックスを選択します。
利用可能な言語のみ表示する	[言語] ボックスに、ご利用中の InstallShield で有効な言語のみを一覧表示するには、このチェック ボックスを選択します。

[SQL Server 要件] ダイアログ ボックス

[SQL Server 要件] ダイアログ ボックスを使って、データベースのサーバー要件を定義できます。

定義済み最低データベース サーバー バージョン

アプリケーションがサポートする定義済み最低データベース サーバーバージョンを選択します。以下の利用可能オプション リストから選択します。

メモ: *インストールの実行時、エンドユーザーによって選択されたすべてのデータベースが指定された要件に合致しているかどうかを確認されます。*

カスタマイズ

個人でデータベース サーバー バージョンを定義する必要は例外的にしかありませんが、必要な場合は、[カスタマイズ] ボックスおよび次のオプションの内から適応するものをチェックします。

テーブル 60 : カスタマイズ オプション

オプション	説明
メジャーバージョン	これは SQL Server から戻されたメジャーバージョンです。たとえば、SQL Server バージョン 7 は、7:00 を戻します。
サービス パック レベル	これは SQL Server から戻されたサービスパックレベル番号です。たとえば、SQL Server バージョン 7 RTM は、623 を戻します。
メジャー バージョンよりも大きい	このオプションをチェックして将来のメジャー バージョンをサポートします。
サービスパックレベルよりも大きい	このオプションをチェックして将来のサービスパックをサポートします。

[優先] ダイアログ ボックス

エディション: *この情報は、基本の MSI プロジェクトに適用します。*

[一般情報] ビューの "優先" 設定にある省略記号ボタン (...) をクリックすると、[優先] ダイアログ ボックスが開きます。このダイアログ ボックスでは、作成中のインストールを優先するその他の .msi パッケージを指定できます (複数可)。InstallShield では、提供する .msi パッケージのリストがソフトウェア識別タグに含められます。このタグには、AdminStudio で、ユーザーがパッケージを AdminStudio アプリケーション カタログにインポートしたときに含められるデータが含まれています。AdminStudio では、この情報を Microsoft System Center 2012 Configuration Manager に公開する前に、確認およびテスト用として、ユーザーに提供しています。

作成中のパッケージを優先する .msi パッケージを選択するとき、現在のインストールの以前のバージョンを指定できます。またオプションで、田のツールの .msi パッケージを選択することもできます。たとえば、画像編集ツールのインストールで作業中の場合、作業中のインストールが、社内で既にサポートされていない画像編集ツールに対して優先されるようにすることもできます。

現在のインストールを優先する .msi パッケージを指定するには、[置き換える .msi ファイルの追加] ボタンをクリックして、リストに含めるパッケージを選択します。

[UI 言語] ダイアログ ボックス

エディション: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[UI 言語] ダイアログ ボックスを使って、[リリース] ビューで選択されたリリースに含めるユーザー インターフェイス言語を指定します。このダイアログ ボックスを開くには、[リリース] ビューを開いて、リリースを選択します。次に、[ビルド] タブで "UI 言語" 設定にある省略記号ボタン (...) をクリックします。

プロジェクトの [一般情報] ビューにある "セットアップ言語" 設定で言語が選択されていない場合、その言語は "UI 言語" 設定の利用可能な言語の一覧には表示されません。

[マージ モジュール検索パスのアップデート] ダイアログ ボックス

[マージ モジュールを参照する] オプションを使ってインストール プロジェクトにマージ モジュールを追加すると、[マージ モジュール検索パスの更新] ダイアログ ボックスが開きます。マージ モジュール検索パスに追加されるパスが表示されます。

[OK] をクリックしてアクションを承認します。

このダイアログ ボックスが表示される理由については、「マージ モジュールを参照した場合に起きること」を参照してください。

ウィザード リファレンス

InstallShield は、インストール プロジェクトの作成を支援する多くのウィザードを装備しています。

プロジェクト: *すべてのウィザードが、すべてのプロジェクトの種類で使用できるわけではありません。*

InstallShield で利用できるウィザードは次の通りです。

- コンポーネント ウィザード
- ソース パス変換ウィザード
- 新規 QuickPatch 作成ウィザード
- テーブル作成ウィザード
- カスタム アクション ウィザード
- データベース インポート ウィザード
- デバイス ドライバー ウィザード...
- ダイアログ ウィザード
- DirectX オブジェクト ウィザード
- ダイナミック スキャン ウィザード
- コンポーネントのエクスポート ウィザード
- 関数ウィザード
- DIM のインポート ウィザード
- REG ファイルのインポート ウィザード
- XML 設定のインポート ウィザード
- Microsoft .NET Framework オブジェクト ウィザード
- MSI からMSIX への変換ウィザード
- 新しい言語ウィザード
- MSI/MSM オープン ウィザード
- MSP オープン ウィザード
- トランスフォーム オープン ウィザード
- パブリッシュ ウィザード
- 再配布可能ファイル ダウンローダー ウィザード
- Reg-Free COM ウィザード
- リリース ウィザード
- セットアップ ベスト プラクティス ウィザード
- スタティック スキャン ウィザード

- システム検索ウィザード
- トランスフォーム ウィザード
- アップグレード検証ウィザード
- ユーザー インターフェイス ウィザード
- Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザード
- Visual Studio デプロイメント プロジェクト インポート ウィザード

コンポーネント ウィザード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

Windows Installer コンポーネントを作成して、それらに対して詳細設定を指定する方法は複数ありますが、コンポーネント ウィザードはそのプロセスを簡易化します。ウィザードは、ファイルに関する情報を自動的に取得することによってファイルをコンポーネントにまとめたり、詳細インストール必要条件を持つファイルを設定して、プロセスを支援します。

タスク

コンポーネント ウィザードを起動するには、以下のいずれかを実行します。

- インストール プロジェクトのみ: [**セットアップのデザイン**] ビューで機能を右クリックして、**コンポーネントウィザード**を選択します。
- 基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース プロジェクトの場合: **[コンポーネント]** ビューで **[コンポーネント]** エクスプローラーを右クリックしてから、**[コンポーネント ウィザード]** をクリックします。

コンポーネント ウィザードを起動すると、[ようこそ] パネルで次のオプションが表示されます:

テーブル 1 · [ウェルカム] パネルの設定

設定	説明
ベスト プラクティスを使用してコンポーネントを作成する	このオプションを選択して、アプリケーションのファイルをすべて指定し、セットアップ ベスト プラクティス に基づいて必要なコンポーネントをすべて作成します。

テーブル 1 - [ウェルカム] パネルの設定 (続き.)

設定	説明
タイプを選択してコンポーネントを定義する	このオプションは、インストールおよびアンインストール時に特別な処理を必要とするファイルが含まれるコンポーネントを作成します。ウィザードでは、次のファイルカテゴリの 1 つに属するコンポーネントを 1 度に 1 つだけ作成できます。別の種類のインストール イベントを選択すると、そのイベント ハンドラーが Setup.rul に貼り付けられます。 • COM サーバー • サービスのインストール • サービスのコントロール • フォント ***メモ-*** *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

[ようこそ] パネル

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

Windows Installer コンポーネントを作成して、それらに対して詳細設定を指定する方法は複数ありますが、コンポーネント ウィザードはそのプロセスを簡易化します。ウィザードは、ファイルに関する情報を自動的に取得することによってファイルをコンポーネントにまとめたり、詳細インストール必要条件を持つファイルを設定して、プロセスを支援します。

ヒント: *コンポーネント ウィザードを使用して既存のコンポーネントを変更しないでください。コンポーネントを作成したら、プロパティ、ファイル、詳細設定などを [セットアップのデザイン] ビュー、または [コンポーネント] ビューで変更できます。*

テーブル 2 · [ウェルカム] パネルの設定

設定	説明
ベスト プラクティスを使用してコンポーネントを作成する	このオプションを選択して、アプリケーションのファイルをすべて指定し、セットアップ ベスト プラクティス に基づいて必要なコンポーネントをすべて作成します。
タイプを選択してコンポーネントを定義する	このオプションは、インストールおよびアンインストール時に特別な処理を必要とするファイルが含まれるコンポーネントを作成します。ウィザードでは、次のファイルカテゴリの 1 つに属するコンポーネントを 1 度に 1 つだけ作成できます。別の種類のインストール イベントを選択すると、そのイベント ハンドラーが Setup.rul に貼り付けられます。 • COM サーバー • サービスのインストール • サービスのコントロール • フォント ***メモ:*** *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

セットアップ ベスト プラクティス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このコンポーネント ウィザードのパスは、アプリケーションのファイルすべてをコンポーネントに構成する最も簡単な手段です。ウィザードは、COM サーバーやフォントなどの特殊なインストール要件を持つインストール ファイルが指定のファイルに検出されても、これを処理することができます。コンポーネント ウィザードがファイルを構成するときに従うベスト プラクティスの詳細については、「コンポーネント作成におけるベスト プラクティス規則」を参照してください。

コンポーネント ウィザードで [ベスト プラクティス] オプションを選択すると、次にベスト プラクティスー[インストール先] パネルが表示されます。

[インストール先] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

最初に、コンポーネントのファイルのインストール先となるターゲット システム上のフォルダーを指定します。

メモ: *[ファイルとフォルダー] ビューでインストール先を右クリックすることによってコンポーネントウィザードを起動した場合、"インストール先" 設定は使用できません。コンポーネント ウィザードを起動する前に選択したインストール先が、作成したコンポーネントのインストール先になります。*

テーブル 3 · [インストール先] パネルの設定

設定	説明
Destination	作成されたすべてのコンポーネントに対して同じインストール先フォルダーが使用されます。このフィールドで選択するインストール先は、このウィザードで作成された各コンポーネントの "インストール先フォルダー" 設定になります。 標準インストール先フォルダーのドロップダウンリストから Windows Installer フォルダー プロパティを (角括弧で囲んで) 選択します。デフォルト値は、INSTALLDIR のルートをポイントします。この値は、製品の "インストール先フォルダー" 設定で初期化されています。 ドロップダウンリストから **[ディレクトリエントリを参照、新規作成または変更する]** を選択して、[ディレクトリの参照] ダイアログ ボックスを表示します。

[ファイル] パネル

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[ファイル] パネルで、コンポーネントにまとめるファイルを指定します。

テーブル 4 - [ファイル] パネルの設定

設定	説明
ファイル	[ファイル] リストには、このコンポーネントに関連付けられた各ファイルについての情報が表示されます。 次のいずれかの方法で、リストにファイルを追加することができます。 - Windows エクスプローラーからファイル リストにファイルをドラッグアンドドロップする。 - [ファイルの追加] をクリックし、アプリケーションのファイルを参照します。 ***メモ-*** *各ファイルへのパスは、ハードコード化されています。ファイルを移動またはファイル名を変更した場合、Link To 値が、*** *ファイルが見つかりません* *****と読み込みます。その後、設定パッケージを作成する際、リリース ウィザードでファイルへのリンクを解決できなくなります。*
COM 情報を今すぐ抽出	このオプションを選択しなかった場合、コンポーネントの "ビルド時に COM 抽出" 設定が [いいえ] に設定され、COM データがすぐに抽出されます。このチェック ボックスをクリアすると、COM データはビルド時に抽出されます。
ファイルの追加	[ファイルの追加] をクリックし、リストに追加するファイルを参照します。表示されたダイアログで、Shift キーまたは CTRL キーを押しながら追加するファイルをクリックすると、シングル フォルダーからファイルをいくつでも選択することができます。 Insert キーを使用してファイルを追加することもできます。
フォルダーの追加	[フォルダーの追加] をクリックして、リストに追加するフォルダーへ移動します。サブフォルダー内のファイルも含め、このフォルダーのファイルはすべてプロジェクトに追加されます。

テーブル 4 - [ファイル] パネルの設定 (続き.)

設定	説明
ファイルの削除	一つまたは複数のファイルを選択して、[ファイルの削除] をクリックすると、選択したファイルがリストから永久に削除されます。 Delete キーを使用してファイルを削除することもできます。

[概要] パネル

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[概要] パネルには、コンポーネントの設定が表示されます。概要を確認して [完了] をクリックしてコンポーネント作成プロセスを完了するとファイルがコンポーネントに編成されます。

機能を右クリックしてこのウィザードを起動した場合には、新規コンポーネントはその機能と関連付けられます。そうでない場合は、[セットアップのデザイン] ビューを開いて、機能を右クリックして [コンポーネントの挿入] を選択し、新しいコンポーネントを機能に関連付けます。

コンポーネントウィザードは、大部分のコンポーネントの設定と必要なすべての詳細設定に対する適切なデフォルト値を提供します。それらを編集するには、[コンポーネント] ビュー でコンポーネントを選択してから、コンポーネントの設定を変更します。

メモ- *機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。*

[コンポーネント タイプ] パネル

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[コンポーネントの種類] パネルでは、1 つの種類のコンポーネントを作成できます。これらのコンポーネントには、固有のインストールおよびアンインストール必要条件があるため、コンポーネントウィザードの後続のパネルは、インストールするコンポーネントの種類によって異なります。

テーブル 5 · [コンポーネントの種類] パネルの設定

設定	説明
コンポーネント名	コンポーネントの種類をクリックすると、ウィザードは "コンポーネント名" フィールドに新規コンポーネント用の固有の推奨名を表示します。名前を変更するには、推奨名を新しい名前で上書きします。 ***メモ·*** *1 つのファイルで複数のフォント ファイルや複数のサービスに単一のコンポーネントの種類を指定できますが、これらはすべて単一コンポーネントに配置され、1 つのコンポーネント名だけを必要とします。* *フォント タイプのコンポーネントを作成している場合、この設定は無効です。InstallShield は、フォント コンポーネントの名前として、フォント ファイルの名前を使用します。*
コンポーネントの種類	作成するコンポーネントの種類を選択します。 · COM サーバー · サービスのインストール · サービスのコントロール · フォント ***メモ·*** *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

COM サーバー コンポーネントの種類

プロジェクト· *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

コンポーネント ウィザードの COM サーバー コンポーネント タイプは、単一の COM サーバー (.exe、.dll または .ocx) ファイルを使用して、インストール プロジェクトにコンポーネントを作成します。

これらのファイルは、ターゲット システムで動作できるようにするために特殊な登録が必要なため、ウィザードは情報を抽出しようとします。ウィザードがファイルのレジストリ設定の抽出に失敗した場合には、CLSID、progID などを入力するプロンプトが表示されます。

COM 登録詳細設定にマップするすべての登録情報が記載されています。ウィザードは、InProcServer32 ThreadingModel 値など、すべての追加エントリをコンポーネントのレジストリ データに作成します。

ヒント: *InstallShield を 64 ビット システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*

インストーラーは、セットアップ ベスト プラクティスに違反する自動登録機能を呼び出す代わりに、コンポーネントのデータを使って、インストール過程でファイルを登録し、アンインストール過程でファイルの登録を解除します。

メモ: *ビルドされたマシンの PATH システム変数は、COM サーバーのリンク先となるすべての .dll ファイルのディレクトリを含むように設定する必要があります。この設定を怠った場合、ファイルの登録は失敗し、COM 情報の抽出が行なわれません。*

ウィザードが COM サーバーをサービスとして実行するかどうかを尋ねるので、ここでサービスのインストール パラメーターを指定します。

[COM サーバー ファイル] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[COM サーバー] ファイル パネルでは、COM サーバーの実行可能ファイルについての基本情報を指定します。

テーブル 6 ・ [COM サーバー ファイル] パネルの設定

設定	説明
COM サーバー ファイル	COM サーバー (.exe、.dll、または .ocx) ファイルのパスとファイル名を入力するか、[参照] ボタンをクリックしてファイルに移動します。 セットアップ ベスト プラクティスに従えば、コンポーネントごとに指定できるポータブル実行可能ファイルは 1 つだけです。ウィザードを再起動して追加の COM サーバーをインストールするか、コンポーネント ウィザードの [ベストプラクティス] オプションを使用してコンポーネントを作成してください。
登録情報を抽出する	ウィザードにファイルをスキャンさせてすべてのファイルの登録情報を抽出する場合には、このボックスを選択します。正常に抽出されると、次に [概要] 確認ダイアログが表示されます。 ***プロジェクト:*** *基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトで作業中の場合、ウィザードを使った COM 情報の抽出ならびに COM 登録詳細設定での情報管理以外に、データをビルド時に動的に抽出することもできます。ダイレクト編集モードで作業中の場合、COM 情報をビルド時に抽出することはできません。この場合、デザイン時に抽出されます。* このチェック ボックスを選択しないか、またはウィザードがファイルの登録情報の読み取りに失敗した場合には、[クラス] パネルが表示されます。 ***ヒント:*** *InstallShield を 64 ビット システムで使用している場合、InstallShield は 64 ビット COM サーバーから COM データを抽出できます。データを正しい場所にインストールするため、コンポーネントを 64 ビットとマークしなくてはなりません。64 ビット サポートに関する詳細は、「64 ビット オペレーティング システムをターゲットにする」を参照してください。*
このファイルを NT サービスとして実行する	このチェック ボックスは COM サーバーが .exe ファイルである場合にのみ使用可能になります。このオプションを選択すると、ウィザードはサービスのインストール パラメーターを指定できるパネルを表示します。 このチェック ボックスを選択してサービスの情報を指定すると、[コンポーネント] ビュー内のコンポーネントの [詳細設定] ノードの下にある [サービス] 領域にその情報が入力されます。ウィザードを再び起動して、この COM サーバーのサービス コンポーネントを作成する必要はありません。

テーブル 6 • [COM サーバー ファイル] パネルの設定 (続き.)

設定	説明
カスタム EXE コマンドライン	このチェック ボックスは COM サーバーが .exe ファイルである場合にのみ使用可能になります。使用可能な場合、コマンド行の引数を入力することができます。コマンドライン引数は、デフォルトの .exe ファイルへの /regserver コマンドラインをオーバーライドします。 サービス オプションを選択して [カスタム EXE コマンドライン] ボックスにパラメーターを指定すると、コマンドラインに /service が渡されます。デフォルトで、/service は /regserver コマンドラインが行う処理はすべて行いますが、さらにサービス サポートのレジストリ値も追加します。 このパネル内のすべてのオプションを選択すると、このボックスに入力したコマンドラインが、.exe ファイルに渡されるコマンドラインとなります。

COM サーバーが 64 ビット コンポーネントの一部であり、InstallShield を 64 ビット マシン上で実行している場合、InstallShield は 64 ビット COM 抽出を行います。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

[クラス] パネル

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このパネルから、COM サーバーの登録情報の入力を始めます。登録情報には progID や CLSID によって識別されるファイルのクラスが含まれます。

このパネルと次のいくつかの登録情報パネルが表示されるのは、ウィザードに情報を抽出させないオプションを選択した場合か、または何らかの理由 (.exe、.dll、または .ocx ファイルが COM サーバーを含んでいないなど) でウィザードが失敗した場合だけです。それ以外の場合、COM サーバーがサービスの場合には [サービス実行可能ファイル] パネルが、COM サーバーがサービスではない場合には [概要] パネルが表示されます。

テーブル 7 · [クラス] パネルの設定

設定	説明
ProgID	COM サーバーのプログラム識別子または progID をこのパネルに入力してください。各 progID はファイル中の COM クラスに対応します。Program.Component.N の形式で新規 progID を追加するには、[追加] ボタンをクリック、または Ins キーを押します。 ProgID にバージョン依存性がない (Program.Component の形式) 場合には、ProgID とバージョン非依存 ProgID フィールドに同じ値を入力します。 このパネルの他のすべての情報は、progID を使用してこのフィールドで指定したクラスに対応します。各 progID をクリックしてそのクラス ID およびバージョン独立の progID を設定し、それが DCOM/COM+ で使用可能かどうかを調べます。
追加	この COM サーバーに新規クラスを登録するには、[追加] ボタンをクリックします。サービスの名前を変更するには、[追加] ボタンをクリックしてすぐに新しい名前を入力するか、F2 キーを押して新しい名前を入力します。 Insert キーを使用して新しいサービスを追加することもできます。
削除	リストから progID を完全に削除する場合には、[削除] ボタンをクリックします。Delete キーを使用してサービスを削除することもできます。
クラス ID	対応するクラス ID を指定するには、progID をクリックします。このクラスに対して Windows Installer が登録する CLSID をの文字列 GUID を入力します。
バージョン非依存の ProgID	対応するバージョン非依存 progID がある場合には、progID をクリックして *Program.Component* の形式で入力します。 progID 自体がバージョン独立の場合には、このフィールドに同じ progID を入力します。
この COM サーバーを DCOM/COM+ で使用可能にする	COM クラスが DCOM または COM+ に対して使用可能であり、そのため追加の登録項目が必要な場合には、このボックスをチェックします。ウィザードの [AppID 一般情報] および [AppID 詳細情報] パネルでこれらの情報を入力します。

[コンテキストの種類] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

COM サーバーがあるファイルの種類を指定します。ファイルの種類は、次のとおりです:

- InprocServer – 16-bit DLL または OCX
- InprocServer32 – 32 ビット DLL または OCX
- LocalServer – 1ビット EXE
- LocalServer32 – 32ビット EXE

[タイプ ライブラリ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

この COM サーバーによって参照されるタイプライブラリがある場合には、それを指定します。

テーブル 8 · [タイプ ライブラリ] パネルの設定

設定	説明
タイプ ライブラリの説明	タイプライブラリの簡単な説明を入力します。 この値は、コンポーネントの COM 登録詳細設定で、COM 登録エクスプローラー中のタイプライブラリ名として使用されます。説明は、HKEY_CLASSES_ROOT¥TypeLib¥*libID*¥*version* のデフォルト値として登録されます。
タイプ ライブラリ GUID	このタイプライブラリを識別する GUID (LibID) を入力します。このフィールドに、自分の COM サーバーの LibID を貼り付ける (Ctrl-V) ことができます。

テーブル 8 • [タイプ ライブラリ] パネルの設定 (続き.)

設定	説明
言語	タイプ ライブラリのロケール ID の 10 進数値を入力します。 通常タイプ ライブラリは言語に中立なため、0 (ゼロ) がタイプ ライブラリの最も一般的な言語です。
バージョン	タイプ ライブラリのバージョン番号を入力します。このフィールドが空の場合は、実行時に Windows Installer によってタイプライブラリのバージョンが自動的に判別されます。

[AppID 一般情報] パネル

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このパネルには、DCOM または COM+ ファイルの AppID に関する基本的な情報を入力します。.AppID は HKEY_CLASSES_ROOT¥AppID に登録される GUID で、分散した COM オブジェクトのすべてのセキュリティと設定オプションをグループ化するために使用します。このパネルおよび [AppID 詳細情報] パネルに入力する情報は、このキー下に登録されます。

このパネルは、[クラス] パネル で **[この COM サーバーは、DCOM/COM+ に対応する]** チェック ボックスが選択されていない限り、表示されません。

テーブル 9 • [AppID 一般情報] パネルの設定

設定	説明
AppID GUID	この COM+ または DCOM コンポーネントの文字列 GUID (AppID) を入力します。このフィールドに AppID を貼り付ける (Ctrl+V) ことができます。
リモート サーバー名	このコンポーネントが実行されるコンピューターの名前を入力します。

[AppID 詳細情報] パネル

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このパネルには、DCOM または COM+ ファイルの AppID に関する、[AppID 一般情報] パネルに入力した情報よりも詳細な情報を入力します。

このパネルは、[クラス] パネル で **[この COM サーバーは、DCOM/COM+ に対応する]** チェック ボックスが選択されていない限り、表示されません。

テーブル 10 - [AppID 詳細情報] パネルの設定

設定	説明
ストレージで起動する	分散 COM サーバーがそのデータと同じコンピューター上にある場合、このボックスをチェック ボックスを選択します。 ストレージで DCOM ファイルがアクティブになると、そのオブジェクトのインスタンスがオブジェクトが使用するデータと同じコンピューター上に作成され、それによってネットワーク トラフィックが軽減されます。アプリケーションの呼び出しによって、DCOM API 関数を使用したこのデフォルトの動作が上書きされる場合があることに注意してください。
インタラクティブ ユーザーとして実行する	このチェック ボックスを選択すると、ユーザーが現在ログオンしているのと同じアカウントで COM サーバーが実行されます。このオプションによって、サーバーが対話型デスクトップに接続できるようになります。 サービスとしても実行できる COM サーバーは、インタラクティブ ユーザーとして実行されるようには設計されていないため、[Win32 サービスとして実行] チェック ボックスを選択した場合には、このチェック ボックスは無効になります。
Win32 サービスとして実行する	分散 COM .exe ファイルをサービスとして実行させたい場合には、このオプションを選択します。このオプションを選択した場合には、サービスの名前とパラメーターも指定する必要があります。 **メモ-** *[COM サーバー実行可能ファイル] パネルで、このファイルをサービスとして実行するように指定している場合には、次に [サービス実行可能ファイル] パネルが表示され、ここでファイルのサービスについて情報を入力する必要があります。*
サービス名	サービスの名前を入力します。
サービス パラメーター	サービスの起動時に渡すパラメーターを入力します。

[概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

コンポーネントの説明を確認し、[完了] をクリックしてウィザードでそのコンポーネントを作成します。入力した情報を変更する必要がある場合には [戻る] をクリックします。

COM 登録詳細設定にマップするすべての登録情報が記載されています。ウィザードは、InProcServer32 ThreadingModel 値など、すべての追加エントリをコンポーネントのレジストリ データに作成します。

ウィザードが**「IDE で、この情報を入力してください」**と表示した場合には、コンポーネントの COM 登録 (または サービスとしても実行するコンポーネントのサービスのインストール) 詳細設定を表示して値を指定します。

機能を右クリックしてこのウィザードを起動した場合には、新規コンポーネントはその機能と関連付けられます。そうでない場合は、[セットアップのデザイン] ビューを開いて、機能を右クリックして [コンポーネントの挿入] を選択し、新しいコンポーネントを機能に関連付けます。

メモ: *機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。*

サービスのコントロール コンポーネントの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

メモ: *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

サービスのコントロール タイプのコンポーネントは、インストールおよびアンインストール中に 1 つのサービスを開始または停止するために使用されます。たとえば、セットアップでサービスを更新をする前に、その実行中のサービスを削除しなければならない場合があります。

ウィザードによって順番に表示されるパネルに従って、コントロールするサービスの情報を入力します。ウィザードが完了すると、デフォルトのプロパティとサービスのコントロールで必要な詳細設定だけで構成される新しいコンポーネントが作成されます。後で、ファイル、ショートカット、レジストリ エントリなどのデータを、[セットアップのデザイン] ビューまたは [コンポーネント] ビューからコンポーネントに追加できます。

タスク ***別のサービスをコントロールするには以下の手順に従います:***

ウィザードをもう一度起動するか、コンポーネントのサービスのコントロールの詳細設定を変更します。

タスク ***サービスをインストールするには、以下の手順に従います:***

ウィザードをもう一度起動して、サービスのインストール コンポーネント タイプを選択するか、サービスのファイルをウィザードで作成したサービスのコントロール コンポーネントに追加して、[コンポーネント] ビューの [詳細設定] 領域でその [サービス] ノードを編集します。

[サービスの指定] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

最初に、インストールするサービスを指定します。ウィザードで実行する他の処理は、すべてこのサービスを参照します。

[コンポーネント] ウィザードを利用して、インストールおよびアンインストールで単一のサービスを制御することができます。新しいコンポーネントで別のサービスを制御するには、このウィザードを再び起動します。

メモ: *同じコンポーネントで複数のサービスを制御するには、[コンポーネント] ビューの [詳細設定] 領域の下にあるコンポーネントの [サービス] ノードを編集する必要があります。*

テーブル 11 • [サービスの指定] パネルの設定

設定	説明
インストール先システムに存在しているサービス	このオプションは、確実なサービスがセットアップの実行時または製品のアンインストール時にターゲット システムにインストールされるようコントロールする場合に選択します。

テーブル 11 - [サービスの指定] パネルの設定 (続き.)

設定	説明
サービス名	サービスの名前を入力します。これは InstallShield、Windows Installer、またはサービス コントロール マネージャー (サービスに使用するシステムのデータベースを保持し、またこれらのサービスを制御するインターフェイスを提供します)が 、サービスを識別するために使うもので、表示名ではありません。
このセットアップに含まれているサービス	セットアップ プロジェクトで作成したサービスから選択する場合、このオプションを使用します。 ***メモ-*** *コンポーネントのいずれも、[コンポーネント] ビューの [詳細設定] 領域の下にある [サービス] ノードにサービスを含まない場合、このオプションは使用できません。*

[インストール イベント] パネル

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このパネルには、製品のインストール中にサービスを制御するための標準オプションがあります。

テーブル 12 - [インストール イベント] パネルの設定

設定	説明
コンポーネントのインストール時に以下のイベントをトリガー	このオプションをクリックして、インストール コントロール イベントを有効にします。
サービスの開始	このイベントは、コンポーネントがインストールされる際にサービスを開始します。具体的には、セットアップの [インストール] シーケンスにある [サービス開始] アクションによって開始されます。 [引数] フィールドでサービスの開始に使用するコマンドラインパラメーターをすべて指定します。
引数	サービスが開始される際に渡される引数を指定します。各引数はコンマで区切ります。

テーブル 12 • [インストール イベント] パネルの設定 (続き.)

設定	説明
サービスの停止	このイベントは、コンポーネントがインストールされる際にサービスを停止します。セットアップの [インストール] シーケンスの [サービスの停止] アクションによって停止されます。
サービスの削除	このイベントは、コンポーネントがインストールされる際にサービスを削除します。セットアップの [インストール] シーケンスの [サービスの削除] アクションによって削除されます。
イベントなし (何もしない)	製品のセットアップ中にサービスを制御する必要がない場合には、このオプションを選択します。このオプションは上記のすべてのイベントをキャンセルします。

[アンインストール イベント] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このコンポーネントがアンインストールされるときにだけコントロールイベントがトリガーされる場合を除いて、このパネルでは、製品のインストールに使用することができた同じコントロールイベントを入力するよう指示が表示されます。

テーブル 13 • [アンインストール イベント] パネルの設定

設定	説明
コンポーネントのアンインストール時に以下のイベントをトリガー	このオプションをクリックして、アンインストール コントロール イベントを有効にします。
サービスの開始	このイベントは、コンポーネントがアンインストールされる際にサービスを開始します。具体的には、セットアップの [インストール] シーケンスにある [サービス開始] アクションによって開始されます。 [引数] フィールドでサービスの開始に使用するコマンドラインパラメーターをすべて指定します。
引数	サービスが開始される際に渡される引数を指定します。各引数はコンマで区切ります。

テーブル 13 • [アンインストール イベント] パネルの設定 (続き.)

設定	説明
サービスの停止	このイベントは、コンポーネントがアンインストールされる際にサービスを停止します。セットアップの [インストール] シーケンスの [サービスの停止] アクションによって停止されます。
サービスの削除	このイベントは、コンポーネントがアンインストールされる際にサービスを削除します。セットアップの [インストール] シーケンスの [サービスの削除] アクションによって削除されます。
イベントなし (何もしない)	製品のセットアップ中にサービスを制御する必要がない場合には、このオプションを選択します。このオプションは上記のすべてのイベントをキャンセルします。 [インストール イベント] パネルで **"イベントなし (何もしない)"** オプションを選択した場合、このパネルではこのオプションは無効になります。コントロールイベントは、インストール時またはアンインストール時のいずれかに選択する必要があります。

[待機のタイプ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

製品のインストールおよびアンインストールのすべてのイベントに対する待機オプションは、このパネルで設定することができます。コントロールイベントの後、Windows Installer はこのパネルで指定された通りに処理を続けます。

インストールまたはアンインストールに対してイベントが不可欠である場合、**[続行前に、サービスのイベント完了まで待機する]** を選択します。

[インストール イベント] または [アンインストール イベント] パネルで **[イベントなし (何もしない)]** オプションを選択した場合、対応する待機設定はこのパネルで使用できなくなります。

[概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ウィザードによって収集された情報を確認し、[完了] をクリックして、サービスのコントロール コンポーネントを作成します。

[コンポーネント] ビューまたは [セットアップのデザイン] ビューで新しいコンポーネントの設定を確認および編集することができます。

機能を右クリックしてこのウィザードを起動した場合には、新規コンポーネントはその機能と関連付けられます。そうでない場合は、[セットアップのデザイン] ビューを開いて、機能を右クリックして [コンポーネントの挿入] を選択し、新しいコンポーネントを機能に関連付けます。

メモ: *機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。*

サービスのインストール コンポーネントの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

メモ: *[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを作成できます。また、このビューを使って、コンポーネント ウィザードでは構成不可能な、拡張サービス カスタマイズ オプションを構成できます。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。*

サービスのインストール コンポーネントは、単一の .exe ファイルに存在するすべての Win32 サービスをインストールするために必要な情報を格納するコンポーネントの作成に使用します。

ヒント: *ファイルを COM サーバとしても実行する場合、[COM サーバー] コンポーネントの種類を選択し、サービスを含むように指定してコンポーネントを作成する方法をお勧めします。COM 登録データの入力が完了すると、[サービスのインストール] コンポーネントタイプに指定すべきサービス情報と同じ情報を入力するプロンプトが表示されます。*

ウィザードによって順番に表示されるパネルに従って、インストールするサービスの情報を入力します。ウィザードが完了すると、使用したファイル、デフォルトのコンポーネントプロパティ、およびサービスのコントロールのインストールおよびアンインストールに必要な詳細設定で構成される新しいコンポーネントが作成されます。

[コンポーネント] ビューの [詳細設定] 領域の下にある [サービス] ノードで、サービスのインストール情報を編集できます。Windows Installer はこれらの値を Windows API 関数の CreateService() に渡して、各サービスをターゲット システムにインストールし、それらを Service Control Manager (サービスのシステムのデータベースを維持し、これらのサービスを制御するインターフェイスを公開します)に登録します。

[サービス実行可能ファイル] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ファイルの完全修飾パスを入力し、サービスをすべて一覧表示します。.exe ファイルのみ可能です。ドライバー サービス (.sys ファイル) は、ウィザードでサポートされていません。

テーブル 14 • [サービス実行可能ファイル] パネルの設定

設定	説明
サービス実行可能ファイル	このコンポーネントのサービスに対する実行可能ファイルへの完全パスを入力するか、[参照] ボタンをクリックして、サービスの実行可能ファイルに移動します。セットアップ ベスト プラクティスに準拠する場合、コンポーネントには、実行可能ファイルを 1 つだけ含むことができます。後でウィザードを再度起動して、別のファイルを新しいコンポーネントに追加します。 サービスを含む COM サーバー コンポーネントがあるためにこのウィザードパネルが表示された場合、[サービス実行可能ファイル] ボックスは無効です。このウィザードは、[COM サーバー実行可能ファイル] パネルで入力したものと同じ実行可能ファイルが使用します。 ***注意 •*** *[参照] ボタンを使用してファイルに移動し、ファイルを選択すると、[サービス実行可能ファイル] ボックスに入力されていた値は新しい値で上書きされます。*
サービス	このリストには、ファイルにあるサービスがすべて含まれます。

テーブル 14 - [サービス実行可能ファイル] パネルの設定 (続き.)

設定	説明
追加	[追加] をクリックして、新しいサービスを追加します。サービスの名前を変更するには、[追加] ボタンをクリックしてすぐに新しい名前を入力するか、F2 キーを押して新しい名前を入力します。これは InstallShield、Windows Installer、CreateService()、およびサービス コントロール マネージャー (サービスに使用するシステムのデータベースを保持し、またこれらのサービスを制御するインターフェイスを提供します) が 、サービスを識別するために使うもので、表示名ではありません。 Insert キーを使用して新しいサービスを追加することもできます。
削除	サービスを選択して [削除] をクリックすると、[サービス] リストからそのサービスが恒久的に削除されます。Delete キーを使用してサービスを削除することもできます。

[サービスの種類の情報] パネル

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ドロップダウンリストの各サービスを選択し、必要な詳細を入力します。

テーブル 15 - [サービスの種類の情報] パネルの設定

設定	説明
サービス	[サービス実行可能ファイル] パネル の [サービス] リストで入力した各サービスを選択し、各サービスについて表示名とサービスタイプのインストールプロパティを指定します。
表示名	サービスの表示名を入力します。これは、SCM でユーザーに対して表示される名前です。[表示名] フィールドを空白にした場合、サービスに使用した名前が表示名として使用されます。 ***メモ-*** *ターゲット システムで表示されるサービスの説明などのその他のサービス プロパティは、コンポーネントの詳細設定で使用可能です。*

テーブル 15 • [サービスの種類の情報] パネルの設定 (続き.)

設定	説明
サービスの種類	サービスが、独自のプロセスで起動するようにするか、または他のサービスとプロセスを共有するようにするかを指定します。コンポーネントに 2 つ以上のサービスがある場合にのみ、**[このサービスは、他のサービスとプロセスを共有します]** オプションを選択します。 サービスのソース コードに関する知識があり、サービスの種類が SERVICE_WIN32_OWN_PROCESS の場合は、**[このサービスは、独自のプロセスで起動します]** を選択し、SERVICE_WIN32_SHARE_PROCESS の場合は、**[このサービスは、他のサービスとプロセスを共有します]** を選択します。 ***メモ•*** *サービスタイプで提供されているオプションには制限があります。コンポーネントウィザードはドライバーサービスをサポートしていません。*
引数	実行時にサービスに渡すコマンドライン引数を入力します。フォーム [PropertyName] の式は [PropertyName] と呼ばれる Windows Installer プロパティ の値に拡張されます。

[サービス開始タイプ情報] パネル

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ドロップダウンリストで各サービスを選択し、サービスをどの方法で開始させるか、またはサービスを使用不可にするか指定します。

開始の種類

サービスの開始タイプをリストから選択します。

- システム起動時に自動的に起動する
- サービス コントロール マネージャーを通して必要時に起動する
- 起動しない (使用不可)

[サービスのロード順] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このファイルの各サービスを選択し、ロード順グループおよび依存関係情報を入力します。ロード順グループに依存関係およびメンバーシップがある場合、サービスが開始される前に、他のサービスが実行されている必要があります。(サービスのロード順グループは、HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Control¥ServiceGroupOrder にリストされています。サービスの "起動タイプ" プロパティにより、グループ内でいつロードされるかが決定されます。)

テーブル 16 · [サービスのロード順] パネルの設定

設定	説明
サービス	[サービス実行可能ファイル] パネルの [サービス] リストで入力した各サービスを選択し、そのロード順情報と依存関係を指定します。
ロード順グループ	このサービスがメンバーであるロード順グループの名前を入力します。サービスがグループに属さない場合は、このフィールドを空白にします。
依存関係	サービスの依存関係を編集フィールドにリストします。サービスの依存関係が別のサービスである場合は、そのサービスの名前を入力し、ロード順グループの場合は、SCM でサービスから区別されるよう、ロード順グループの名前の前にプラス記号 (+) を付けます。 各依存関係はコンマで区切ります。

[エラー コントロール] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

[サービス実行可能ファイル] パネルの [サービス] リストに入力した各サービスを選択し、各サービスについて、サービスの開始を試みた際にエラーを受け取った場合にサービス コントロール マネージャーが実行するアクションを指定します。

- エラーをログに記録して、サービスの開始を続行する
- エラーをログしてメッセージボックスを表示してから、サービスを継続して起動する。
- エラーをログに記録して、可能な場合は、最後に有効であった設定でシステムを再起動する

[サービス ログオン] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

リストの各サービスを選択し、開始時にログオンされるサービスのアカウントの種類を指定します。

注意: *パスワードが競合する可能性があるため、Microsoft では、サービスがすべてローカルシステムアカウントでログオンされるよう推奨しています。*

テーブル 17: [サービス ログオン] パネルの設定

設定	説明
サービス	[サービス実行可能ファイル] パネルの [サービス] リストで入力した各サービスを選択し、サービスがログオンされる方法を指定します。
ローカルのシステム アカウント	このオプションを選択して、サービスがローカルシステムアカウントでログオンされるようにします。
サービスがデスクトップとの対話を許可する	サービスにユーザー インターフェイスがある場合、このオプションをクリックします。
アカウントを指定する	このオプションは、ログオン時にサービスに特定のユーザーを偽装させる場合に選択します。ユーザー名およびパスワードを入力する必要があります。これらを入力しない場合、ローカルシステムアカウントが使用されます。
ドメイン/ユーザー名	**DomainName¥UserName** という形式のアカウントを入力します。**.¥UserName** のように、ビルトインドメインにピリオド (.) を使用できます。

テーブル 17 • [サービス ログオン] パネルの設定 (続き.)

設定	説明
Password	このアカウントのパスワードを入力します。 パスワードは、ユーザー名を指定しないと使用されず、サービスはローカルシステムアカウントでログオンされます。

[概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ウィザードによって収集された情報を確認し、[完了] をクリックして、サービス コンポーネントを作成します。

ウィザードには、コンポーネントのすべてのプロパティについてデフォルトを提供します。プロパティシートは、[セットアップのデザイン] ビューで確認できます。Windows Installer では、ローカルシステムに常駐しないサービスをインストールすることはできないため、サービスに対して "リモートインストール" 設定は特に重要です。

機能を右クリックしてウィザードを起動した場合、新しいコンポーネントがその機能に関連付けられます。そうでない場合は、[セットアップのデザイン] ビューを開き、1 つまたは複数の機能を右クリックして [コンポーネントの挿入] を選択して、新しいコンポーネントを機能に関連付けます。

メモ: *機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。*

ウィザードが完了したあと、[コンポーネント] ビューの [詳細設定] 領域の下にあるコンポーネントの [サービス] ノードで、任意のインストール オプションを変更できます。

InstallShield には、このサービスや、ターゲット システム上の他のサービスを制御するための機能があります。詳細については、「Windows サービスのインストール、制御、および構成」を参照してください。

フォント コンポーネント タイプ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

- *MSI データベース*
- *MSM データベース*

さまざまな方法でコンポーネントを作成できますが、セットアップにフォント (.ttf、.fon、または.ttc ファイル) をインストールする最も簡単な方法は、コンポーネント ウィザードでフォント コンポーネントを作成することです。

ウィザードにある他のコンポーネントタイプとは異なり、フォントコンポーネントに対応する詳細設定はありません。フォント コンポーネントを区別し、違ったようにインストールするよう Windows Installer に指示する設定は、ファイルの "フォント タイトル" プロパティです。

フォント コンポーネントを作成する別の方法は、コンポーネント ウィザードで[セットアップ ベスト プラクティス] オプションを選択して、使用しているフォントファイルからフォント コンポーネントを InstallShield に作成させることです。

メモ: *アプリケーションがアンインストールされた後でもフォントをシステムで維持する場合は、コンポーネントの "パーマネント" プロパティを [はい] に設定します。*

[インストールされているフォントの追加] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ウィザードは開発システム上に登録されているすべてのフォントを検索します。フォントをリストで選択することによって、それらを含むコンポーネントを作成できます。

テーブル 18 : [インストールされているフォントの追加] パネルの設定

設定	説明
フォント	[フォント] コンポーネントに含めるフォントを選択します。
このシステムにインストールされていないフォントを追加する	このオプションをクリックして、[新規フォントの追加] パネルで追加フォントを参照できます。 このオプションを選択しない場合、ウィザードは次に [概要] パネルを表示します。

[新規フォントの追加] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

このパネルでは、フォントコンポーネントに .ttf ファイル、.fon ファイル、または .ttc ファイルを追加することができます。

メモ: *このパネルは、ご使用のマシンにインストールされていないフォントをインストールに追加できるよう設計されています。たとえば、ネットワークドライブに保存されているフォントファイルがあるとします。しかし、これらのファイルはローカルマシンに既にインストールされているため、Fonts フォルダーから選択できません。マシンにインストールされているフォントをインストールに追加する場合は、[戻る] をクリックして、リストから適切なフォントを選択します。*

テーブル 19 • [新しいフォント] パネルの設定

設定	説明
フォント	リストにファイルをドラッグアンドドロップするか、[フォントの追加] をクリックして、コンポーネントに新規フォントを参照および追加します。 InstallShield では、フォントファイルからフォントタイトルを読み取ります。フォント ファイル内でタイトルが指定されていない場合、フォントタイトルをクリックして、Marlett (TrueType) のように FontTitle (FontType) 形式でタイトルを指定します。
フォントの追加	[フォントの追加] をクリックして新しいフォントファイルを参照します。
フォントの削除	リストでフォントを選択して [フォントを削除] をクリックすると、選択されたフォントがコンポーネントからに恒久的に削除されます。

[概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

機能を右クリックしてウィザードを起動した場合、新しいコンポーネントがその機能に関連付けられます。そうでない場合は、[セットアップのデザイン] ビューを開き、1 つまたは複数の機能を右クリックして [コンポーネントの挿入] を選択して、新しいフォント コンポーネントを機能に関連付けます。

メモ: *機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。*

フォントの機能は、そのインストールが選択されると、Windows Installer によって自動的に登録されます。

フォントコンポーネントは、すべてデフォルトによって FontsFolder プロパティに保存されているディレクトリにインストールされます。コンポーネントのプロパティシートを編集することによって、コンポーネントのインストール先フォルダーを編集したり、追加プロパティを設定できます。

メモ: *プログラムがアンインストールされた後でもフォントをシステムで維持する場合は、[フォント] コンポーネントの [パーマネント] プロパティを [はい] に設定します。*

ウィザードにある他のコンポーネントタイプとは異なり、フォントコンポーネントに対応する詳細設定はありません。フォント コンポーネントを区別し、違ったようにインストールするよう Windows Installer に指示する設定は、ファイルの "フォント タイトル" プロパティです。

コンポーネント ウィザード [一般概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

ウィザードでコンポーネント作成プロセスの最後にこのパネルが表示されます。特定の種類のコンポーネントに関するヘルプは、以下の情報を参照してください。

- セットアップベストプラクティスオプション
- COM サーバー コンポーネントの種類
- サービスのインストール コンポーネントの種類
- サービスのコントロール コンポーネントの種類
- フォントコンポーネントの種類

メモ: *機能に関連付けられていないコンポーネントは、親のないコンポーネントのアイコンで表示されます。*

ソース パス変換ウィザード

ソースパス変換ウィザードを使用すると、ハードコードされた既存のパスをパス変数に変換できるため、セットアッププロジェクトの移植性を向上させることができます。

タスク ***ソースパス変換ウィザードを起動するには、以下の手順に従います:***

[プロジェクト] メニューで、**[ソース パス] ボタン**をクリックします。

[ようこそ] パネル

ソースパス変換ウィザードでは、直接入力されたセットアップのパスをパス変数に変換できます。プロジェクト内でハードコード化されたパスが検索され、検出されたパスは標準のパス変数に置換されます。

ファイルが既存のパス変数に設定されていると、新しい変数を選択するオプションは表示されずに、ファイルは自動的に変換されます。たとえば、Program Files フォルダーにファイルが 1 つあった場合、このパスは自動的に事前に定義されたパス変数である ProgramFilesFolder に変換されます。

[検索] パネル

[検索] パネルには、検索の状態が表示されます。ウィザードは、現在のプロジェクトの絶対パスを検索しています。ファイルが既存のパス変数と一致すると、ファイルは自動的に変換されます。

[検索結果と推奨] パネル

[検索結果と推奨] パネルには、まだパス変数が割り当てられていないディレクトリにあるプロジェクトの、ファイルとリンクのパス変数オプションが一覧表示されます。たとえば、ファイルが C:¥App Files¥Executable¥App.exe の場合、Executable というパス変数を作成してパスを示すことができます。

パネルのオプション

パス変数の推奨

対応するボックスを選択することによって、作成する変数を示します。提案された変数のいずれかを作成しない場合、変数の横のチェック ボックスをクリアします。

名前の変更

名前を変更する変数を選択し、[名前を変更] ボタンをクリックします。[パス変数の推奨] 列に新しい変数名を入力します。

すべて選択解除

[すべて選択解除] ボタンをクリックして、対応する推奨されたパス変数の横にあるチェック ボックスをすべてクリアします。

すべて選択

推奨されたパス変数をすべて選択する場合、[すべて選択] ボタンをクリックします。[適用] をクリックすると、これらの変数が作成されてプロジェクトに追加されます。

[プロジェクトのアップデート] パネル

このパネルには、プロジェクトで直接入力されたパスをパス変数に更新する際のウィザードの進行状況が表示されます。

[アップデート完了] パネル

このパネルは、ソースパスの変換ウィザードが終了したことを知らせます。[完了] ボタンをクリックすると IDE に戻ります。

新規 QuickPatch 作成ウィザード

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトは、規模の小さいシングル アップデートをエンドユーザーへ配布したいインストール作成者へお勧めします。QuickPatch はカスタマイズ可能な範囲が限られてはいますが、[パッチのデザイン] ビューを使わないシンプルなパッチ作成方法として利用できます。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

既存の .msi ファイルまたは既存の QuickPatch をパッチする QuickPatch プロジェクトを作成するには、新規 QuickPatch 作成ウィザードを利用します。

注意: *以前に作成した QuickPatch をパッチする QuickPatch を作成する場合、前もって以前の QuickPatch プロジェクトをビルドしておかなければ予期しない動作が発生する可能性があります。*

タスク ***新規 QuickPatch 作成ウィザードを開くには、以下の手順を実行します。***

InstallShield で新規プロジェクトの作成を行い、プロジェクトの種類として[QuickPatch プロジェクト] を選択します。

この QuickPatch 作成方法は、既存の .msi ファイルのパッチまたは既存の QuickPatch のパッチを行なう場合に利用することができます。

ヒント: *新規 QuickPatch 作成ウィザードを利用して、既存の QuickPatch をパッチする QuickPatch プロジェクトを作成することができます。また別の方法として、最新の QuickPatch プロジェクトを InstallShield で開き、現在の QuickPatch をパッチする QuickPatch プロジェクトの作成を指定することもできます。詳細については、「既存の QuickPatch をパッチする QuickPatch プロジェクトを作成する」を参照してください。*

[ようこそ] パネル

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトは、規模の小さいシングル アップデートをエンドユーザーへ配布したいインストール作成者へお勧めします。QuickPatch オーサリングはカスタマイズの範囲が限られてはいますが、[パッチのデザイン] ビュー以外でのパッチの作成を可能にします。基本的にどちらのパッチ作成方法も同じ配布タイプ (.msp と .exe ファイル) を作成します。

既存の .msi ファイルまたは既存の QuickPatch をパッチする QuickPatch プロジェクトを作成するには、新規 QuickPatch 作成ウィザードを利用します。

メモ: *ウィザードの [キャンセル] または [終了] をクリックすると、QuickPatch プロジェクトは未完成となり、QuickPatch プロジェクトを InstallShield で開くことはできません。*

[QuickPatch プロジェクト ベース] パネル

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[QuickPatch プロジェクトベース] パネルでは、新規 QuickPatch プロジェクトを既存の .msi ファイルまたは 既存の QuickPatch プロジェクトのどちらをベースに作成するかを指定します。

パネルのオプション

テーブル 20 • [QuickPatch プロジェクト ベース] パネルのオプション

オプション	説明
MSI パッケージをベースにする	既存 .msi ファイルに QuickPatch プロジェクトをビルドするにはこのプションを選択します。アプリケーションに作成する最初の QuickPatch の場合、このオプションを選択します。
既存の QuickPatch プロジェクトをベースにする	既存 QuickPatch をベースに QuickPatch プロジェクトをビルドするにはこのプションを選択します。このオプションは、元の .msi ファイルおよび既存の QuickPatch プロジェクトをパッチできるプロジェクトを作成します。 ***メモ•*** *前もって既存の QuickPatch プロジェクトをビルドせずに QuickPatch を作成すると、予期しない動作を行なう場合があります。*

"MSI パッケージをベースにする" オプションを選択して [次へ] をクリックすると、[元のセットアップ パッケージ] パネルが開きます。[既存の QuickPatch プロジェクトをベースにする] オプションを選択して [次へ]をクリックすると、[既存 QuickPatch の場所]パネルが開きます。

[元ののセットアップ パッケージ] パネル

プロジェクト• *この情報は、QuickPatch プロジェクトに適用します。*

既存の Windows Installer インストール用の QuickPatch プロジェクトをビルドする場合、[元のセットアップ パッケージ] パネルを使って .msi ファイルの場所を指定します。元のインストールは新規パッチのベースとなるインストールです。新規 QuickPatch 作成ウィザードは、このインストールをアップデートすることができるパッチを作成します。

元のインストールが圧縮されている場合、InstallShield はそのインストールの非圧縮イメージを作成します。[元のセットアップ パッケージ] パネルを利用して、非圧縮ベースイメージのパスを指定することができます。

メモ• *プロジェクトを作成して InstallShield で開いた後は、ベースとなるインストールへのパスを変更することはできません。したがって、異なるインストールをパッチする場合 [新規 QuickPatch 作成] ウィザードを再度実行して新たに QuickPatch プロジェクトを作成する必要があります。*

パネルのオプション

テーブル 21 • [元のセットアップパッケージ] パネルのオプション

オプション	説明
元のセットアップ	元/ベースとなるインストール先を指定する、または [参照] ボタンをクリックします。新規 QuickPatch 作成ウィザードは、このインストールを有効にすることができるパッチを作成します。 元のインストールが非圧縮の場合、[完了] をクリックして InstallShield で QuickPatch プロジェクトを開きます。
パスの圧縮解除	 ***メモ•*** *このオプションは元のインストールが非圧縮の場合は利用できません。非圧縮イメージを含むパスを指定します。パスを指定しない場合、デフォルトの場所 (<ISProjectDataFolder>¥BaseImage) が使用されます。*

[完了] をクリックして [新規 QuickPatch 作成] ウィザードを閉じ、InstallShield で新規 QuickPatch プロジェクトを開きます。

注意 • *ローカルまたはネットワークドライブから複数ディスクにわたるインストールを圧縮解除することはできません。別の方法でインストールの圧縮解除を行う必要があります。*

[既存の QuickPatch の場所] パネル

プロジェクト• *この情報は、QuickPatch プロジェクトに適用します。*

既存の QuickPatch 用の QuickPatch プロジェクトをビルドする場合、[既存 QuickPatch の場所] パネルを使ってその場所を指定します。作成された QuickPatch は元のインストールのアップデート、指定した既存 QuickPatch のアップデート、または指定した任意の QuickPatch 中間プロジェクトのアップデートに利用することができます。

メモ• *プロジェクトを作成して InstallShield で開いた後に、既存 QuickPatch プロジェクトへのパスを変更することはできません。したがって、異なる QuickPatch プロジェクトをパッチする場合 [新規 QuickPatch 作成] ウィザードを再度実行して新たに QuickPatch プロジェクトを作成する必要があります。*

パネルのオプション

既存 QuickPatch プロジェクト

パッチする QuickPatch プロジェクト (.ism ファイル) の場所を指定するか、[参照] ボタンをクリックします。

テーブル作成ウィザード

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

テーブル作成ウィザードは、ダイレクト エディターを使ってテーブルを追加するときに起動します。このウィザードはカスタム テーブルについての情報を収集して、テーブルリストに追加します。

タスク ***テーブル作成ウィザードを起動するには、以下の手順を実行します。***

1. **[追加ツール]** の下のビュー バー/ビュー リストにある **[ダイレクト エディター]** をクリックします。
2. **[テーブル]** エクスプローラーを右クリックし、**[テーブルの追加]** をクリックします。

テーブル作成ウィザードは次のパネルで構成されます:

- ようこそ
- 列のプロパティ
- 概要

[ようこそ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

[ようこそ] パネルは、テーブル作成ウィザードの概要を説明します。

タスク ***テーブルを作成するには、以下の手順を実行します。***

1. [テーブル名] ボックスに名前を入力します。

   Windows ロゴ ガイドライン: *Windows ロゴ プログラムでは、カスタム テーブルの名前に* **MSI** *(大文字、小文字、または両方の組み合わせ) を使用しないことを要求しています。「MSI」で始まる名前は、今後の新しい標準プロパティおよびテーブル用に予約されています。*

2. **[列の追加]** ボタンをクリックして、ウィザードの次のパネルに進みます。

[列のプロパティ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

[列のプロパティ] パネルでは、カスタムテーブルの列の情報を設定することができます。次のプロパティを設定できます。

テーブル 22 : [列のプロパティ] パネルのオプション

プロパティ	説明
列の名前	列の名前を入力します。
列の説明	列の説明を入力します。
列の種類	使用できる列の種類は、バイナリ、長い値、短い値および文字列です。
[NULL 許可] チェック ボックス	列の値にヌルを使用できる場合、このチェック ボックスを選択します。
[プライマリキー] チェック ボックス	この列をテーブルのプライマリキーにする場合、このチェック ボックスを選択します。
[ローカライズ可能文字列] チェック ボックス	文字列のローカライズを可能にする場合、このチェック ボックスを選択します。

テーブル 22 • [列のプロパティ] パネルのオプション (続き.)

プロパティ	説明
文字列長	文字列に使用できる最大長を入力します。任意の文字長を指定するには 0 を入力します。
最小値	数値の列の場合、最小値を指定します。
最大値	数値の列の場合、最大値を指定します。
外部キーテーブル	この列が外部テーブルを参照する場合、このリストから外部テーブルを選択します。
外部キー列	外部テーブルで参照した列を選択します。

[別の列の追加] をクリックして、追加の列を定義するか、[列の作成が終了しました] をクリックして [概要] パネルに進みます。

[概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

[概要] パネルで、プロジェクトに追加する前に新しいテーブルの設定を確認します。設定を変更する必要がある場合は、[戻る] をクリックして適切なパネルまで戻り、変更を追加します。

[完了] をクリックして、カスタムテーブルを追加します。

カスタム アクション ウィザード

プロジェクト: *カスタム アクション ウィザードは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

カスタム アクション ウィザードでは、カスタム アクションをインストールに追加することができます。これらのアクションで、次の操作を行うことができます: .dll ファイル関数の呼出し、実行可能ファイル (.exe) の起動、VBScript、JavaScript、InstallScript コードの実行、エラー メッセージの表示、ディレクトリまたはプロパティの設定、その他。

タスク ***カスタム アクションウィザードを起動するには、以下の操作を行います。***

1. **[動作とロジック]** の下のビュー バー/ビュー リストで、**[カスタム アクションとシーケンス]** (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または **[カスタム アクション]** (DIM、マージ モジュールおよび MSM データベース プロジェクトの場合) をクリックします。
2. **[カスタム アクション]** エクスプローラーを右クリックして、**[カスタム アクション ウィザード]** をクリックします。

カスタム アクションウィザードは次のパネルで構成されます。

- ようこそ
- 基本情報
- アクションの種類
- 関数定義
- アクションのパラメーター
- マネージ メソッドの定義
- シーケンス内スクリプト
- 追加オプション
- 応答オプション
- シーケンスへ挿入する
- 概要

Windows ロゴ *・Windows ロゴを適用する場合、インストールに含まれるすべてのカスタム アクションは、カスタム アクション作成に関するベスト プラクティス ガイドに従わなくてはなりません。InstallShield 検証スイートおよび完全 MSI 検証 スイートを使って、インストール パッケージがほとんどのカスタム アクション関連のロゴ要件を満たしているかどうかを検証することができます。ただし、一部の要件については検証スイートを使って検証することができません。詳細については、「Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン」を参照してください。*

[ようこそ] パネル

プロジェクト: *カスタム アクション ウィザードは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ようこそ] パネルでは、カスタム アクション ウィザードが簡単に説明されています。ウィザードによって、Windows Installer で本来サポートされていないタスクを実行できるカスタム アクションを作成できます。

Windows ロゴ: *Windows ロゴを適用する場合、インストールに含まれるすべてのカスタム アクションは、カスタム アクション作成に関するベスト プラクティス ガイドに従わなくてはなりません。InstallShield 検証スイートおよび完全 MSI 検証 スイートを使って、インストール パッケージがほとんどのカスタム アクション関連のロゴ要件を満たしているかどうかを検証することができます。ただし、一部の要件については検証スイートを使って検証することができません。詳細については、「Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン」を参照してください。*

[基本情報] パネル

プロジェクト: *カスタム アクション ウィザードは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[基本情報] パネルでは、カスタム アクションの名前と説明を入力できます。この情報は参照用に使用されるため、エンドユーザーには表示されません。以下は、このパネルで提供されている設定です。

テーブル 23 • [基本情報] パネルの設定

設定	説明
名前	カスタム アクションの名前を入力します。名前には、半角英数字、アンダースコア、およびピリオドを指定することができます。平仮名などの2バイト文字は、使用できません。また、名前の先頭には アルファベット、またはアンダースコアを使用する必要があります。
コメント	このボックスに、アクションについてのコメントを入力します (オプション)。この情報は参照用で、エンドユーザーには表示されません。

[アクションの種類] パネル

プロジェクト: *カスタム アクション ウィザードは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[アクション タイプ] パネルでは、作成するカスタム アクションのタイプ、およびそのアクションのコードの格納場所を指定できます。

パネルのオプション

種類

[タイプ] リストでは、異なるカスタム アクションの実行方法をオプションで選択することができます。たとえば、作成したカスタム アクションが実行可能ファイルの呼び出し、VBScript コードの実行、または .dll ファイルからの関数の呼び出しができるようにすることができます。以下のオプションは次のリストで利用可能です。

テーブル 24 • 使用可能なカスタム アクションの種類

種類	プロジェクトの種類	説明
標準ダイナミック リンク ライブラリの関数を呼び出す	基本の MSI、InstallScript MSI	このカスタム アクションは、.dll ファイル内にある関数を呼び出します。[関数定義] パネルで、関数名、パラメーター、および戻り値を指定してください。 ***メモ•*** *Binary テーブルに格納されている標準 .dll ファイルの関数を呼び出す場合、アクションの "スクリプト内実行" 設定で遅延実行またはロールバック実行は使用できません。* *製品と共にインストールされる標準 .dll ファイルの関数を呼び出し、アクションが ("スクリプト内実行" 設定で) "遅延" とスケジュールされている場合、呼び出す .dll ファイルはコンポーネントのキーファイルでなくてはなりません。*
Windows Installer のダイナミック リンク ライブラリの関数を呼び出す	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、Windows Installer 専用に書かれた .dll ファイルから関数を呼び出します。.dll ファイルのエントリ ポイントには、定義済みパラメーターおよび戻り値が必要です。
マネージ アセンブリのパブリック メソッドを呼び出す	基本の MSI、DIM、InstallScript MSI、マージ モジュール	このカスタム アクションは、Visual Basic .NET または C# などのマネージ コードで書かれた、マネージ アセンブリのパブリック メソッドを呼び出します。 詳細については、「マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す」を参照してください。

テーブル 24・使用可能なカスタム アクションの種類 (続き.)

種類	プロジェクトの種類	説明
エラー メッセージを表示して中止する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、指定されたエラー メッセージを表示し、失敗を戻して、インストールを終了します。
実行可能ファイルを起動する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは実行可能ファイルを起動します。
別の.msi パッケージを起動する	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	このカスタム アクションは「ネスト インストール」とも呼ばれ、インストール時に 2 番目の .msi パッケージを起動します。 ***重要:*** *Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*

テーブル 24 • 使用可能なカスタム アクションの種類 (続き.)

種類	プロジェクトの種類	説明
64 ビット JScript コードを実行する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、C や Visual Basic などのコンパイル済み言語の代わりに、64 ビットの JScript コードを実行します。
PowerShell コードの実行	基本の MSI、InstallScript MSI	このカスタム アクションは、PowerShell スクリプト (.ps1) を実行します。 ターゲットについては、このタイプのカスタム アクションに関する他の情報も含めて、「PowerShell カスタム アクションの呼び出し」を参照してください。
64 ビット VBScript コードを実行する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、C や Visual Basic などのコンパイル済み言語の代わりに、64 ビットの VBScript コードを実行します。
InstallScript コードを実行する	基本の MSI、InstallScript MSI、マージ モジュール	このカスタム アクションは、InstallScript コードから関数を呼び出します。
JScript コードを実行する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、C や Visual Basic などのコンパイル済み言語の代わりに、JScript コードを実行します。

テーブル 24・使用可能なカスタム アクションの種類 (続き.)

種類	プロジェクトの種類	説明
VBScript コードを実行する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、C や Visual Basic などのコンパイル済み言語の代わりに、VBScript コードを実行します。
プロパティを設定する	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、Property テーブル内のプロパティを設定します。これは、エンドユーザーから情報を取得し、インストールの後半で使用できるようにその情報を格納しておく必要がある場合に便利です。
ディレクトリを設定する	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このカスタム アクションは、実行時に、Directory テーブルのディレクトリを設定する。たとえば、エンド ユーザーが選択したインストール ディレクトリのサブディレクトリとして一時ディレクトリを作成する場合、このオプションを使って新しい一時ディレクトリを設定して、インストールの後半で使用できます。
ID で終了するプロセスを判別する	基本の MSI、InstallScript MSI	このカスタム アクションは、特定のプロセス ID (PID) を持つプロセスを強制終了します。 ***重要**・このタイプのカスタム アクションを作成するには、中止する PID を識別するプロパティも定義する必要があります。詳細については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。*

テーブル 24 - 使用可能なカスタム アクションの種類 (続き.)

種類	プロジェクトの種類	説明
名前で終了するプロセスを判別する	基本の MSI、InstallScript MSI	このカスタム アクションは、特定の実行可能ファイル名を持つプロセスを強制終了します。 ***重要*** *- このタイプのカスタム アクションを作成するには、中止するプロセスの名前を識別するプロパティも定義する必要があります。詳細については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。*

場所

カスタム アクションで呼び出されるコードは、いくつかの場所の 1 つに格納できます。インストーラーがコードを検索する場所を選択します。一部のカスタム アクションの種類では、サポートされていない場所があります。"場所" 設定は、エラー カスタム アクションには適用しません。。

テーブル 25 - カスタム アクションを格納できる場所

場所	カスタム アクションの種類	説明
アドバタイズされているか、またはインストール済みのアプリケーション	別の.msi パッケージを起動する	この場所を選択すると、ターゲットマシンに現在インストールされているすべての Windows Installer アプリケーション (以前、カスタム アクションを使ってインストールしたアプリケーションなど) がアンインストールされます。このカスタム アクションは、メインセットアップのアンインストール時にのみ起動してください。
インストール先マシンの検索パス	標準 DLL	標準 .dll ファイルから関数を呼び出すとき、ターゲット システムの検索パスにある .dll ファイルを指定することができます。
メインのセットアップに組み込む	別の.msi パッケージを起動する	起動する .msi ファイルがメインのインストール パッケージに格納されている場合、この場所を選択します。
製品と一緒にインストール	標準 DLL、MSI DLL、マネージ コード、.NET Core 実行可能ファイル、PowerShell コード、VBScript、64 ビット VBScript、JScript、64 ビット JScript	ターゲット システムにインストールされるファイルまたは実行可能ファイルからコードを呼び出す場合、この場所を選択します。

テーブル 25 • カスタム アクションを格納できる場所 (続き.)

場所	カスタム アクションの種類	説明
カスタム アクションに直接保存する	VBScript、 64 ビット VBScript、 JScript、 64 ビット JScript	ファイルの関連付けを行わずに、ウィザードに直接コードを入力する場合、この場所を選択します。
Binary テーブルに保存する	標準 DLL、 MSI DLL、 マネージ コード、 .NET Core 実行可能ファイル、 PowerShell コード、 VBScript、 64 ビット VBScript、 JScript、 64 ビット JScript	コード ベースを Binary テーブルに格納する場合、この場所を選択します。このオプションは、ターゲット システムにコード ファイルをインストールしない場合に便利です。 ***メモ:*** *マネージ コードおよび .NET Core タイプのカスタム アクションの場合、ビルド時に、.NET アセンブリに対して C++ Windows Installer ラッパー DLL が作成されます。ラッパー DLL には、アセンブリのほか、アセンブリの仲介、ロード、実行を行うために必要な情報が含まれています。ラッパー DLL が Binary テーブルに格納されます。*
Directory テーブルに保存する	実行ファイル	(ファイル名を使わずに) Directory テーブルに格納されているパスを参照する場合、この場所を選択します。
Property テーブルに保存する	マネージ コード、 実行可能ファイル、 VBScript、 64 ビット VBScript、 JScript、 64 ビット JScript	起動する実行可能ファイルへの完全パスを参照する場合、この場所を選択します。パスは Property テーブルに格納されます。
ソース メディア上に格納する	別の.msi パッケージを起動する	起動する .msi ファイルがメインのインストール パッケージと同じメディアに格納されている場合、この場所を選択します。

[関数定義] パネル

[関数定義] パネルで、標準 .dll ファイル中のエントリポイント関数のパラメーターを指定します。

このパネルは、カスタム アクション ウィザードの [アクションの種類] パネルで **[標準ダイナミック リンク ライブラリの関数を呼び出す]** を選択したときのみ表示されます。

メモ: *関数は必ず _stdcall 呼び出し規則を使用しなくてはなりません。この規則が使用されていない場合、複数のパラメーターを持つ関数は適切に動作しません。*

パネルのオプション

名前

呼び出す関数の名前を指定します。.dll ファイルは、各カスタム アクションに対して シングル エントリ ポイントを持つことができます。

引数

引数のリストを関数に渡す順序でビルドします。

まず、[引数] リストの最後の行をクリックして新規引数を追加します。次に、各引数の Type、Source および Value 列を完成させます。

種類

このリストで、パラメーターのデータ型を選択します。引数には 2 つの特殊な種類があります。これらを以下の方法で識別します。

タスク ***ポインターを NULL に初期化するには次の操作を実行します。***

1. **[タイプ]** リストから **POINTER** を選択します。
2. **[ソース]** リストから、**[定数]** を選択します。
3. **[Value]** リストから **[NULL]** を選択します。NULL の代わりに 0 (ゼロ) は入力しないでください。これは、「0 番をポイントするポインター」として解釈されます。

タスク ***ハンドルに .msi データベースまたは Windows Installer ウィンドウを設定するには、次の手順を実行します。***

1. **[タイプ]** リストから **[HANDLE]** を選択します。
2. **[ソース]** リストから、**[定数]** を選択します。
3. **[値]** リストで、ハンドルに MSI データベースを設定するか、MSI ウィンドウを設定するかによって、**MsiHandle** または **MsiWindowHandle** を選択します。

ソース

データを関数に渡す方法を示します。以下のオプションは次のリストで利用可能です。

テーブル 26 - データを関数へ渡す際のオプション

オプション	説明
定数	渡す値はリテラルとしてインストール プロジェクト中に保存されます。引数のソースに [定数] を選択した後、その定義した値を [値] 列に入力します。 この「定数」には、名前や ID は必要ありません。文字列リテラルは引用符で囲まないでください。
in プロパティ	ソースで [in プロパティ] を指定して、値が関数に渡される Windows Installer プロパティの名前を指定します。このタイプの引数は ByVal パラメーターに適しています。

テーブル 26 • データを関数へ渡す際のオプション（続き.）

オプション	説明
inout プロパティ	このソースの種類は ByRef パラメーターに似ています。ソースで [inout プロパティ] を指定して、値が関数に渡され、関数で変更することが可能な Windows Installer プロパティの名前を指定します。
out プロパティ	[out プロパティ] は、関数で設定することができる値のプレースホルダーとして機能します。関数に値は渡されませんが、関数が値を変更できるプロパティの名前は必要です。

引数のソースのプロパティを選択するたびに、[値] 列にプロパティの名前を指定する必要があります。

値

値は以下の 2 種類のうちの 1 つです。

- 数字または文字列リテラル 引数のソースが定数の場合
- Windows Installer プロパティ 引数のソースがプロパティの場合

メモ: *[タイプ] リストを HANDLE に設定し、[ソース] リストを [定数] に設定すると、Value 列に 2 つのオプション（MsiHandle MsiWindowHandle）が含まれます。これらの定数を使用して、.msi データベースまたは Windows Installer ウィンドウへのハンドルを取得できます。*

ソースがプロパティの場合、[値] リストでプロパティ マネージャー中のすべてのプロパティの名前が表示されます。既存のプロパティを選択するか、または新しい名前を入力できます。後者の場合、プロパティ マネージャーに新規プロパティが追加されます。

プロパティは単なる値のコンテナであることに留意してください。パラメーターがその値を[in プロパティ] または [inout プロパティ] に保存する場合は、プロパティマネージャーでそのプロパティの値を必ず指定してください。

(コンテキスト メニュー)

コンテキスト メニューで、引数を使用するためのオプションが表示されます。コンテキスト メニューにアクセスするには、[引数] グリッドで引数を右クリックします。コンテキスト メニュー オプションを以下に説明します。

テーブル 27 • コンテキストメニューのオプション

オプション	説明
追加	[追加] をクリックして、[引数] グリッドに引数を追加します。
削除	引数を削除するには、[削除] をクリックします。
上へ移動	引数は、関数がそれを受け取るのとまったく同じ順序でなければなりません。引数をリストの上に移動するには、[上へ移動] をクリックします。
下へ移動	引数をリストの下に移動するには、[下へ移動] をクリックします。

戻り値の型

関数の戻り値のデータ型を選択します。戻り値を確認する必要がない場合は、void を受け入れることができます。

戻り値プロパティ

値が関数の戻り値に設定されるプロパティ名を選択します。インストールの他の場所で戻り値をチェックする場合、戻りプロパティの値を初期化することも考えられます。

メモ: アクションが即時実行に構成されていない限り、関数の戻り値にプロパティを設定することはできません。したがって、アクションが遅延、ロールバック、またはコミット実行としてスケジュールされている場合、実行時に "戻り値プロパティ" 設定は無視されます。

サイレント モード

インストールでカスタム アクションが .dll ファイルのロードおよび関数呼び出しに失敗したときの警告メッセージを抑制する場合は、このチェック ボックスを選択します。

[アクション パラメーター] パネル

[アクションパラメーター] パネルで、カスタム アクションで呼び出す実際のファイルを指定できます。たとえば、実行可能ファイルの呼び出しを行う場合、該当するファイルを参照してコマンドラインオプションを追加することができます。

このパネルは、カスタム アクションで、アクション内に直接格納されている JScript や VBScirpt コードを実行している場合は表示されません。

パネルのオプション

ソース

このボックスは、[アクションタイプ] パネルで選択した [タイプ] と [場所] リストのオプションに従って異なる動作をします。"ソース" フィールドは、追加中のカスタム アクションの種類に適用しない場合 (たとえば、新しい Kill Process カスタム アクションなど) は無効です。以下は、使用可能なソース値を、カスタム アクションの種類と位置別に示したテーブルです。

テーブル 28 • カスタム アクションのソースの値

カスタム アクションの種類	ソース設定の手順
標準 DLLにある関数を呼び出す	このカスタム アクションタイプの位置によってソースを設定します。 • **インストール先マシンの検索パス** – .dll ファイルの場所を参照します。ウィザードでは、ファイル名は使用されますが、パスは使用されません。 • **製品と共にインストール** – [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、.dll ファイルが保存されているフォルダーを選択します。[ファイル] 一覧で、標準 .dll ファイルを選択します。 • **Binary テーブルに保存** – .dll ファイルの場所を参照します。

テーブル 28 • カスタム アクションのソースの値 (続き.)

カスタム アクションの種類	ソース設定の手順
Windows Installer DLL の関数の呼び出し	このカスタム アクションタイプの位置によってソースを設定します。 • **製品と共にインストール** — [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、.dll ファイルが保存されているフォルダーを選択します。[ファイル] 一覧で、Windows Installer .dll ファイルを選択します。 • **Binary テーブルに保存** — .dll ファイルの場所を参照します。
マネージ アセンブリのパブリック メソッドを呼び出す	このカスタム アクションタイプの位置によってソースを設定します。 • **製品と共にインストール** — [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、.NET アセンブリが保存されているフォルダーを選択します。ファイルの一覧で、アセンブリを選択します。 • **Binary テーブルに保存** — .NET アセンブリの場所を参照します。 • **プロパティを参照するパス** — 一覧から Windows Installer プロパティを選択し、新しいプロパティの名前を入力し、Directory テーブル内のディレクトリ名を入力します。プロパティ名とディレクトリ名は、角かっこ ([ ]) で囲む必要があります。必要に応じて、プロパティの後に文字列を追加することができます。作成されたパスは、.NET アセンブリの場所を示します。
エラー メッセージを表示して中止する	"ソース" 設定はこの種類のカスタム アクションに適用しないため、エラーカスタム アクションに対しては無効になっています。
実行可能ファイルを起動する	このカスタム アクションタイプの位置によってソースを設定します。 • **製品と共にインストール** — [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、実行可能ファイルが保存されているフォルダーを選択します。ファイルの一覧で、実行可能ファイルを選択します。 • **Binary テーブルに保存** — .exe ファイルの場所を参照します。 • **Property テーブルに保存** — Windows Installer プロパティを一覧から選択します。新しいプロパティの名前を指定した場合、後でプロパティ マネージャーを開いてプロパティを入力し、.exe ファイルをポイントする値を指定する必要があります。 • **Directory テーブルに保存** — 一覧から標準フォルダーを 1 つ選択するか、または Directory テーブルに新しいエントリの名前を入力します。入力した名前は、後でダイレクト エディターで編集することができます。指定したディレクトリは、[ターゲット] ボックスで指定したターゲットの作業ディレクトリになります。

テーブル 28 • カスタム アクションのソースの値 (続き.)

カスタム アクションの種類	ソース設定の手順
別の.msi パッケージを起動する	このカスタム アクションタイプの位置によってソースを設定します。(別の .msi パッケージを起動するカスタム アクションの詳細は、ネスト インストールを参照してください。) • **メインのセットアップに組み込み** – "子" の .msi ファイルの場所を参照。カスタム アクション ウィザードは、Binary テーブルに適当なエントリを自動作成します。 • **ソース メディアに保存** – "子" の .msi ファイルの場所を参照。カスタム アクション ウィザードは、このファイルを自動的に [サポート ファイル] ビューの Disk1 フォルダー に追加します。 • リリースをビルドするとき、InstallShield によって .msi ファイルがリリースの位置にコピーされます。.msi ファイルの未圧縮アプリケーションとサポート ファイルはコピーされません。 • **アドバタイズ、または既にインストールされているアプリケーション** – "子" の .msi ファイルの場所を参照。カスタム アクションウィザードは選択した .msi パッケージから製品コードを抽出します。
InstallScript コードを実行する	InstallScript ファイルの場所は変更できません。InstallScript カスタム アクションの場合、リストからエントリポイント関数の名前を選択する必要があります。
VBScript または 64-bit JScript コードの実行	このカスタム アクションタイプの位置によってソースを設定します。 • **製品と共にインストール** – [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、.js ファイルが保存されているフォルダーを選択します。ファイルの一覧で、.js ファイルを選択します。 • **Binary テーブルに保存** – JScript ソース ファイルの場所を参照します。 • **Property テーブルに保存** – リストからプロパティを 1 つ選択します。新しいプロパティを指定した場合、後でプロパティ マネージャーを開いて、プロパティの名前を入力し、JScript コードをプロパティの値として追加する必要があります。 • **カスタム アクションに直接保存** – [シーケンス内スクリプト] パネルを使用して、JScript コードをテキスト編集することができます。 ***メモ:*** *[シーケンス内スクリプト] パネルは下位互換性を目的としてのみ提供されています。[カスタム アクションとシーケンス]ビュー (および [カスタム アクション] ビュー)の [スクリプト] タブでさらに高度なスクリプト エディターを利用することができます。*

テーブル 28・カスタム アクションのソースの値 (続き.)

カスタム アクションの種類	ソース設定の手順
PowerShell コードの実行	このカスタム アクションタイプの位置によってソースを設定します。 ・ **製品と共にインストール** – [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、.ps1 ファイルが保存されているフォルダーを選択します。次に、ファイルの一覧で、.ps1 ファイルを選択します。 ・ **Binary テーブルに保存** – PowerShell ファイル (.ps1) の場所を参照します。
VBScript または 64 ビット VBScript コードを実行する	このカスタム アクションタイプの位置によってソースを設定します。 ・ **製品と共にインストール** – [参照] をクリックして、[ファイルの参照] ダイアログ ボックスを開きます。インストール先フォルダーの一覧で、.vbs ファイルが保存されているフォルダーを選択します。ファイルの一覧で、.vbs ファイルを選択します。 ・ **Binary テーブルに保存** – VBScript ソース ファイルの場所を参照します。 ・ **Property テーブルに保存** – リストからプロパティを 1 つ選択します。新しいプロパティを指定した場合、後でプロパティ マネージャーを開いて、プロパティの名前を入力し、VBScript コードをプロパティの値として追加する必要があります。 ・ **カスタム アクションに直接保存** – [シーケンス内スクリプト] パネルを使用して、VBScript コードを編集することができます。 ***メモ・** [シーケンス内スクリプト] パネルは下位互換性を目的としてのみ提供されています。[カスタム アクションとシーケンス]ビュー (および [カスタム アクション] ビュー)の [スクリプト] タブでさらに高度なスクリプト エディターを利用することができます。*
ディレクトリを設定する	ディレクトリを設定中は場所を設定できません。リストからフォルダー名を選択するか、または、新しい名前を入力してディレクトリ テーブルでその名前を指定する必要があります。ディレクトリ テーブルは、[ダイレクト エディター] からアクセスできます。
プロパティを設定する	プロパティを設定中は場所を設定できません。リストからプロパティ名を選択するか、または、新しい名前を入力してプロパティ マネージャーでその名前を指定する必要があります。
プロパティを設定する	プロパティを設定中は場所を設定できません。リストからプロパティ名を選択するか、または、新しい名前を入力してプロパティ マネージャーでその名前を指定する必要があります。
プロパティを設定する	プロパティを設定中は場所を設定できません。リストからプロパティ名を選択するか、または、新しい名前を入力してプロパティ マネージャーでその名前を指定する必要があります。

ターゲット

ターゲットの値はカスタム アクションの種類によって異なります。

テーブル 29・ターゲットのオプション

カスタム アクションの種類	ターゲット設定の手順
標準 DLLにある関数を呼び出す	標準 DLL タイプのカスタム アクションの場合、[ターゲット] ボックスは読み取り専用です。関数定義] パネルで指定された関数の定義を表示します。
Windows Installer DLL の関数の呼び出し	このボックスを使用して、呼び出す関数の名前を入力します。括弧などの特殊なフォーマットを使用する必要はありませんが、この関数名は大文字と小文字が区別されるため注意して入力してください。 Windows Installer .dll 関数には、定義済みのパラメーターと戻り値タイプがあります。詳細については、「カスタム アクションで DLL ファイル関数にパラメーターを渡す」を参照してください。
マネージ アセンブリのパブリック メソッドを呼び出す	マネージ アセンブリ タイプのカスタム アクションの場合、[ターゲット] ボックスは使用できません。[マネージ メソッドの定義] パネル (この種類のカスタム アクションに使われるウィザードの次のパネル) では、起動するメソッドを識別します。
エラー メッセージを表示して中止する	カスタム アクションの条件がターゲット システムで満たされたときに表示するエラー メッセージ テキストを入力します。 上記方法の代わりに、値にエラー テキストが含まれているプロパティ名を入力することもできます。プロパティ名は、角かっこ ([ ]) で囲む必要があります。
実行可能ファイルを起動する	実行可能ファイルを起動する場合、このボックスにコマンドライン引数を入力できます。たとえば、インストールと同時にインストールされる readme ファイルを起動する場合、Notepad.exe [INSTALLDIR]Readme.txt と入力する必要があります。
別の.msi パッケージを起動する	.msi パッケージに渡すパブリック プロパティを指定します。
InstallScript コードを実行する	ターゲットの値は、InstallScript カスタムアクションには適用されません。
PowerShell コードの実行	ターゲットの値は、PowerShell カスタム アクションには適用されません。
JScript/64 ビット JScript または VBScript/64 ビット VBScript コードの実行	呼び出す関数名を入力します。
プロパティやディレクトリの設定	選択または作成したプロパティの新しい値に設定されるフォーマット済みテキスト文字列を入力します。フォーマットされたテキスト文字列の詳細については、Windows Installer ライブラリを参照してください。

テーブル 29 • ターゲットのオプション (続き.)

カスタム アクションの種類	ターゲット設定の手順
ID で終了するプロセスを判別する	Kill Process カスタム アクションで終了するプロセスの PID を入力します。1 つ以上のプロセスを指定する場合、セミコロン (;) を使って PID を区切ります。
名前で終了するプロセスを判別する	強制終了カスタム アクションで終了するプロセスの実行可能ファイル名を入力します。1 つ以上のプロセスを指定する場合、セミコロン (;) を使ってファイル名を区切ります。

[マネージ メソッドの定義] パネル

[マネージ メソッドの定義] パネルで、マネージ アセンブリ内のパブリック クラス メソッドを指定できます。カスタム アクション ウィザードは、[アクションの種類] パネルで **[マネージ アセンブリのパブリック メソッドを呼び出す]** を選択したときのみ表示されます。

[マネージ メソッドの定義] パネルには、次の設定があります。

テーブル 30 • [マネージ メソッドの定義] パネルの設定

設定	説明
クラス	メソッドが関連付けられているパブリック クラスを入力します。また、[参照] ボタンをクリックして、[メソッドの参照] ダイアログ ボックスを指定することもできます。このダイアログ ボックスを使って、[アクションのパラメーター] パネルで選択したマネージ アセンブリにあるパブリック クラスの一覧からパブリック メソッドを選択することができます。 ***メモ:*** *参照機能は .NET Framework がインストールされているときのみ使用できます。また、この機能は、マネージ アセンブリの場所が* Binary *テーブルに設定されていて、製品と共にインストールされているときのみ使用可能です。アセンブリのパスがプロパティ、または* Directory *テーブルのディレクトリを参照する場合は使用できません。* *また、参照機能は 32 ビット アセンブリおよび Microsoft intermediate language (MSIL) ファイルでのみ使用できます。64 ビット アセンブリでは使用できません。*

テーブル 30 - [マネージ メソッドの定義] パネルの設定 (続き.)

設定	説明
メソッド	インストールで呼び出すパブリック メソッドを入力します。また、[参照] ボタンをクリックして、[メソッドの参照] ダイアログ ボックスを指定することもできます。このダイアログ ボックスを使って、[アクションのパラメーター] パネルで選択したマネージ アセンブリにあるパブリック クラスの一覧からパブリック メソッドを選択することができます。 ***メモ-*** *参照機能は .NET Framework がインストールされているときのみ使用できます。また、この機能は、マネージ アセンブリの場所が* Binary *テーブルに設定されていて、製品と共にインストールされているときのみ使用可能です。アセンブリのパスがプロパティ、または* Directory *テーブルのディレクトリを参照する場合は使用できません。* *また、参照機能は 32 ビット アセンブリおよび Microsoft intermediate language (MSIL) ファイルでのみ使用できます。64 ビット アセンブリでは使用できません。*
カスタム メソッド シグネチャを使用する	デフォルトのメソッド シグネチャを使用する場合、このチェック ボックスをクリアします。デフォルトのメソッド シグネチャを利用すると、シグネチャに 1 つまたは複数の無効なパラメーターまたは単一 MsiHandle パラメーターがあるとき、インストールで、メソッドが MsiHandle パラメーターが使われて呼び出されます。 カスタム メソッド シグネチャを使用する場合、このチェック ボックスを選択します。そのあと、必要に応じて、引数と戻り値プロパティを指定します。 [参照] ボタンをクリックしてクラスとメソッドを指定すると、関連付けられたパラメーターが自動的に追加されます。メソッドに渡す各パラメーターの値を指定します。 メソッドのシグネチャの指定については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。

テーブル 30 - [マネージ メソッドの定義] パネルの設定 (続き.)

設定	説明
引数	カスタム メソッド シグネチャを使用する場合、選択したメソッドに渡す引数の一覧を指定します。 [参照] ボタンをクリックしてクラスとメソッドを指定すると、関連付けられたパラメーターがこの領域にあるグリッドに自動的に追加されます。 プロパティを引数として使用する場合、パラメーターの値フィールドをクリックして、一覧からプロパティを選択します。 コンテキスト メニューで、引数を使用するためのオプションが表示されます。コンテキスト メニューにアクセスするには、[引数] グリッドで引数を右クリックします。以下は、提供されるコンテキスト メニューのコマンドです: • **[追加]** – グリッドに引数を追加します。 • **[削除]** – グリッドから引数を削除します。 このグリッドは、**[カスタム メソッド シグネチャを使用する] チェック ボックス**が選択されたときのみ使用できます。 引数の指定については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。
戻り値プロパティ	値がメソッドの戻り値に設定されるプロパティ名を選択します。 この一覧は、**[カスタム メソッド シグネチャを使用する] チェック ボックス**が選択されたときのみ使用できます。 戻り値プロパティの指定については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。

[シーケンス内スクリプト] パネル

メモ: *[シーケンス内スクリプト] パネルは下位互換性を目的としてのみ提供されています。[カスタム アクションとシーケンス]ビュー (および [カスタム アクション] ビュー)の [スクリプト] タブでさらに高度なスクリプト エディターを利用することができます。*

カスタム アクションが、カスタム アクションに直接格納されている JScript または VBScript コードを呼び出す場合、表示されるボックスにスクリプトを入力することができます。テキストボックスにコードを入力し、[次へ] をクリックして続けます。構文をチェックするためのコード検証は実行されません。

[追加のオプション] パネル

[追加のオプション] パネルでは、カスタム アクション スレッドの処理方法、および、カスタム アクションをパッチのアンインストール時のみ実行するかどうかを指定することができます。

[追加のオプション] パネルには、次の設定が含まれています:

テーブル 31 • [追加のオプション] パネルの設定

設定	説明
戻り値の処理	カスタム アクション スレッドの処理方法を指定します。以下は、[戻り値の処理] 一覧で提供されているオプションです: • 非同期 (完了を待機しない) • 非同期 (終了コードまで待機) • 同期 (終了コードを確認) • 同期 (終了コードを無視) これらのフラグは、メインとカスタム アクションのスレッドが同期実行 (インストーラーがメイン インストール スレッドを再開する前にカスタム アクション スレッドが完了するのを待機する) か非同期 (インストーラーがメイン インストールの続行と同時にカスタム アクションを実行する) かを指定するために使用されます。 オプションの中には、一部の種類のカスタム アクションには適用できないものがあります。選択された種類のカスタム アクションに関連するオプションのみが、一覧に表示されます。

テーブル 31 - [追加のオプション] パネルの設定 (続き.)

設定	説明
パッチのアンインストール時に実行	 ***プロジェクト-*** *ダイレクト編集モードでプロジェクト作業を行っているとき、データベース スキーマの最小が 405 (Windows Installer 4.5 以降) になっていない場合、この設定は適用しません。* このチェック ボックスを利用して、Windows Installer がパッチのアンインストール時にのみカスタム アクションを実行するかどうかを指定できます。このチェック ボックスは、.msi パッケージに含まれるカスタム アクションに対してのみ選択できます。このチェック ボックスは、パッチによって追加される新しいカスタム アクションに対しても選択することもできます。ただし、パッチによって既存のカスタム アクションに追加または削除されるカスタム アクションに対しては選択できません。このチェック ボックスはデフォルトでクリアになっています。 Windows Installer 4.5 がパッチのアンインストール カスタム アクションを実行するとき、アンインストールされるパッチに含まれるカスタム アクションが使用されます。 ***メモ-*** *Windows Installer 4.5 は、この設定のサポートを含みますが、Windows Installer 4.0 以前は、これを含みません。したがって、この設定で [はい] を選択するときに、一部のターゲット システムの Windows Installer が 4.0 以前の可能性がある場合、このカスタム アクションが Windows Installer 4.0 以前で実行されないように条件を追加します。この条件を追加しなかった場合、Windows Installer 4.0 以前はインストール中にそのカスタム アクションを呼び出して、パッケージを修復、または更新します。* 条件には、MSIPATCHREMOVE プロパティを使用します。詳細については、Windows Installer ヘルプ ライブラリの「MSIPATCHREMOVE」を参照してください。

[応答オプション] パネル

[応答オプション] パネルでは、カスタム アクションがインストールの残りの部分に対してどう応答するかを指定することができます。

テーブル 32 • [応答オプション] パネルのオプション

オプション	説明
スクリプト内実行	アクションの実行タイミングを指定します。 • **即時実行** – シーケンスでアクションの順番が来たときにアクションを実行する場合、このオプションを選択します。 • **即時実行 (ターミナル サーバーで使用可能)** – シーケンスでアクションの順番が来たときにアクションを実行する場合、このオプションを選択します。ターミナル サーバー マシン上でマシン単位のインストールが実行されいる間、カスタム アクションはユーザーに成り済まします。 • **遅延実行** – このオプションを選択すると、スクリプトの実行が開始されるまでアクションを待機させておくことができます。 • **ロールバック実行** – このオプションを選択すると、ロールバック中にのみアクションが実行されるよう設定できます。カスタム アクションを使用した [実行] シーケンスの途中でシステムに変更を加えた場合、ロールバック カスタム アクションを使って変更を元に戻す必要があります。 • **ロールバック実行 (ターミナル サーバーで使用可能)** – このオプションを選択すると、ロールバック中にのみアクションが実行されるよう設定できます。カスタム アクションを使用した [実行] シーケンスの途中でシステムに変更を加えた場合、ロールバック カスタム アクションを使って変更を元に戻す必要があります。ターミナル サーバー マシン上でマシン単位のインストールが実行されいる間、カスタム アクションはユーザーに成り済まします。 • **コミット実行** – インストールによってインストール スクリプトが正常に完了されるまでアクションの実行を遅延する場合、このオプションを選択します。 • **コミット実行 (ターミナル サーバーで使用可能)** – インストールによってインストール スクリプトが正常に完了されるまでアクションの実行を遅延する場合、このオプションを選択します。ターミナル サーバー マシン上でマシン単位のインストールが実行されいる間、カスタム アクションはユーザーに成り済まします。 • **システムの設定で遅延実行** – レジストリの編集などタスクの実行にシステム権限が必要なアクションがある場合、このオプションを選択します。アクションは、スクリプトの実行が開始されるまで遅滞されます。アクションは、実際に実行されるときはシステム権限を使用して実行されます。このプロパティの詳細については、「スクリプト内実行」を参照してください。 ***メモ:*** Binary *テーブルに格納されている標準のダイナミック リンク ライブラリの関数を呼び出す場合、アクションの [スクリプト内実行] プロパティで遅延実行またはロールバック実行は使用できません。*

テーブル 32 • [応答オプション] パネルのオプション (続き.)

オプション	説明
実行スケジュール	アクションを実行する回数を指定します。たとえば、[ユーザー インターフェイス] シーケンスおよび [実行] シーケンスの両方にアクションがリストされている場合、このアクションを 2 回実行するか、または 1 回だけ実行するかを設定できます。選択可能なオプションは以下のとおりです: • **常に実行** – アクションは順番が来るたびに実行します。つまり、該当するアクションを [ユーザー インターフェイス] シーケンスで 1 回実行した後、[実行] シーケンスでさらにもう 1 回実行することができます。 • **1度のみ実行** – [ユーザー インターフェイス] および [実行] シーケンスの両方にアクションが存在する場合でも、アクションは一度だけ実行します。[ユーザー インターフェイス] シーケンスが実行されている場合は、[実行] シーケンスのアクションはスキップされます。 ***メモ: [1 度のみ実行]** オプションは InstallScript MSI プロジェクトでは使用できません。[ユーザー インターフェイス] シーケンスと [実行] シーケンスの両方にカスタム アクションが存在する場合は 2 度実行されます。これは InstallScript MSI プロジェクトでは、[ユーザー インターフェイス] シーケンスは InstallScript エンジンで実行され、[実行] シーケンスは Windows Installer で実行されるためです。* • **1 プロセスにつき 1 度のみ実行** – [ユーザー インターフェイス] シーケンスと [実行] シーケンスは別々のプロセスで実行されます。[ユーザー インターフェイス] シーケンスと [実行] シーケンスで最低 1 回ずつカスタム アクションを実行する場合、このオプションを選択します。 • **クライアント上で常に最低 1 回実行** – このアクションはクライアントで少なくとも 1 回実行されます。 "スクリプト内実行" 設定で [即時実行] 以外を選択した場合、"実行スケジュール" 設定は利用不可能になり、[常に実行] が設定されます。

[シーケンスに挿入] パネル

[シーケンスに挿入] パネルで、ウィザードが作成しているカスタム アクションへのシーケンスの挿入場所を指定します。カスタム アクションは、1 つのシーケンスに挿入することも複数のシーケンスに挿入することもできます。

パネルのオプション

シーケンス

カスタム アクションを挿入する [インストール]–[ユーザー インターフェイス] または [インストール]–[実行] シーケンスの場所を指定します。

条件

参照ボタンをクリックして、[条件ビルダー] ダイアログ ボックスを起動し、カスタム アクションを起動するかどうかを判断するのに使用される条件を作成します。

[概要] パネル

[概要] パネルでは、インストールに追加する前にカスタム アクションの設定を確認できます。設定を変更する必要がある場合は、[戻る] をクリックして適切なパネルまで戻り、変更を追加します。

アクションが完成したら、[完了] ボタンをクリックします。

このカスタム アクションをシーケンスに挿入 したり、ダイアログのコントロール イベントの引数として指定することができます。

データベース インポート ウィザード

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。Oracle ユーザーの方は、Oracle データベース ユーティリティの Oracle Web ページで、InstallShield と共に利用可能なユーティリティについての情報をご覧いただけます。*

データベース インポート ウィザードは Microsoft SQL Server Database のインポートを支援し、インストール作成者が選択したオプションに基づいてデータベースから SQL Script ファイルを生成します。ウィザードはSQL スクリプトビューを右クリックして SQL スクリプトエクスプローラー内の次のノードから起動することができます。

- SQL Server
- 新しい接続
- 新しいスクリプト

メモ: *InstallShield は、SQL Server 7.0 以降をサポートします。*

データベースインポートウィザードには、次のウィザードパネルが関連付けられています。

- ようこそ
- SQL Server
- SQL データベース
- データベース テーブル
- スクリプト化するオブジェクト
- スクリプト作成オプション
- スクリプト作成の詳細オプション
- 概要

[ようこそ] パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

データベース インポート ウィザードは Microsoft SQL Server Database のインポートを支援し、インストール作成者が選択したオプションに基づいてデータベースから SQL Script ファイルを生成します。

このウィザードパネルの 2 番目の段落は、ウィザードが作成するものを示します。

SQL Server パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

サーバー名

次のサーバーリストから SQL Server を指定するか、[参照] をクリックしてネットワーク上のサーバーの場所を選択します。

メモ: *SQL sアーバーに関する次の情報を考慮してください:*

- *InstallShield は SQL Server 2005 (9.x) 以降をサポートします。*
- *データベースをインポートして SQL Server を参照するには、ODBC ドライバーバージョン 3.80 以降が必要です。*

ネットワーク ログイン ID を使った Windows NT 認証

これが推奨される認証方法です。SQL Servers をこのメソッドと共に動作するよう設定する必要があります。現在ログイン中のユーザーおよびユーザーの認証情報が利用されます。

次のログイン ID およびパスワードを使用した Server 認証

このオプションを選択した場合、ログイン ID を入力し、特定の SQL Server のパスワードを設定する必要があります。

[SQL データベース] パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

データベース名

SQL Server パネルで選択した SQL Server にあるデータベースを選択します。[データベース名] ドロップダウンリストで選択したデータベース用にスクリプト ファイルが生成されます。

スクリプト表示名

[データベース名] ドロップダウンリストで選択した SQL データベースから生成されたスクリプト ファイルの有効な識別子を入力します。スクリプト表示名は、[SQL スクリプト] ビューのエクスプローラーに表示される名前です。

メモ: *生成されたスクリプトがターゲットマシンで実行されるとき、このデータベースがターゲットサーバー上にない場合は自動的に作成されます。*

[データベース テーブル] パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

このパネルでは、データベースのどのテーブルを生成されたスクリプト ファイルに含むか、または除外するかを決定します。特定のテーブルを含む、または除外することを選択した場合、[利用可能なテーブル] リストから必要なテーブル (複数可) をクリックしてから ">" ボタンをクリックしてテーブルに印をつけます。

メモ: *[すべてのテーブルを含む] オプションを選択すると、データベースを再インポートするたびにデータベースに確実に新しいテーブルが追加されます。*

[スクリプト化するオブジェクト] パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

[スクリプト化するオブジェクト] パネルでは、データベースからインポートするオブジェクトを選択します。

オブジェクト

この手順ではマージ モジュールの TARGETDIR ディレクトリ、または TARGETDIR から直接引き出したディレクトリのみを上書きします。これらのオブジェクトの多くは単一の Microsoft SQL Server データベース オブジェクトの属性を表示します。

メモ: ***[レコード]*** *オプションに関する次の情報を考慮してください:*

- *[データベース テーブル ウィザード] パネルでインポートするテーブルが選択されていない場合は、****[記録]*** *オプションを利用することはできません。*
- *[データベース テーブル ウィザード] パネルでテーブルをインポート中に、****[記録]*** *オプション内にある* ***[読み取り専用をスキップ]*** *サブオプションを選択すると、ユーザーがテーブル内の指定した列の* ***UPDATE*** *権限を拒否されている場合に、指定した列のデータのスクリプト作成をスキップすることができます。*

[スクリプト作成オプション] パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

データベース インポート ウィザードの [スクリプト オプション] パネルでは、スクリプトが Microsoft SQL Server バージョン 7.0 と互換性を持つかどうかを指定します。このパネルの他のオプションでは、説明用ヘッダー、拡張プロパティ、および他の情報をスクリプトに含めるかどうかを指定できます。必要に応じて Transact-SQL ステートメントが生成されます。

一般オプション

テーブル 33 • 一般オプション

オプション	説明
存在しない場合のみ作成する	その存在を示すチェックが付いたコンポーネントを作成する Transcat-SQL ステートメントを生成するには、このチェック ボックスを選択します。スクリプトが実行される時、指定されたコンポーネントのコピーが存在しない場合のみコンポーネントが作成されます。 ***メモ:*** *現在、このオプションはテーブルにのみ適用されます。*
存在する場合、最初をドロップする	参照されたコンポーネントを削除する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。スクリプトは、コンポーネントの削除を試みる前に、その存在をテストします。
説明用ヘッダーを含める	スクリプト内の各 Transact-SQL ステートメントの前に説明的なヘッダー テキストを含めるには、このチェック ボックスを選択します。
拡張プロパティを含める	作成された SQL スクリプトに拡張ストアド プロシージャを含めるには、このチェック ボックスを選択します。 このチェックボックスを選択すると、Microsoft SQL Server でユーザーが作成したオブジェクトについてのメタ情報を含めることができます。詳細は、「sp_addextendedproperty (Transact-SQL)」を参照してください。

テーブル スクリプト作成オプション

テーブル 34 • テーブル スクリプト作成オプション

設定	説明
インデックス スクリプト	任意で選択されたテーブルに存在するインデックスを作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 ***メモ:*** *このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。*
フル テキスト インデックス スクリプト	フルテキスト インデックスを作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。

テーブル 34 • テーブル スクリプト作成オプション (続き.)

設定	説明
トリガー スクリプト	任意で選択されたテーブルに存在するトリガーを作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 *メモ• このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。*
CHECK 制約スクリプト	任意で選択されたテーブルに存在する CHECK 制約を作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 *メモ• このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。*
外部キー スクリプト	任意で選択されたテーブルに存在する外部キーを作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 *メモ• このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。*
主キー スクリプト	任意で選択されたテーブルに存在する主キーを作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 *メモ• このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。*
デフォルト スクリプト	選択されたテーブルに存在するデフォルトを作成する Transact-SQL ステートメントを生成するには、このチェック ボックスを選択します。 *メモ• このチェック ボックスの選択は、[データベース テーブル] パネルで 1 つまたは複数のテーブルが選択されている場合のみ有益です。*

[スクリプト作成詳細オプション] パネル

メモ• データベースのインポート機能は、Microsoft SQL Server Database で利用できます。

[スクリプト作成詳細設定] パネルでは、セキュリティ スクリプト作成オプションおよびファイル形式を指定することができます。

セキュリティ スクリプト作成オプション

テーブル 35 • セキュリティ スクリプト作成オプション

設定	説明
データベース スクリプト	既存のデータベース スキーマのスクリプトを作成する Transact-SQL ステートメントを生成する。
データベース ユーザーとデータベース ロール スクリプト	データベースへのアクセス権があるすべてのユーザーとロールを作成する Transact-SQL ステートメントを生成する。
SQL Server ログイン (Windows NT と SQL Server ログイン) スクリプト	現在サーバーへのアクセス権があるすべてのログインを作成する Transact-SQL ステートメントを生成する。
オブジェクト レベル権限スクリプト	[スクリプト化するオブジェクト] パネルで選択された各オブジェクトについて現在存在するすべての GRANT、REVOKE、DENY 権限を作成する Transact-SQL ステートメントを生成する。

ファイル形式

テーブル 36 • ファイル フォーマットのオプション

オプション	説明
Windows テキスト (ANSI)	デフォルトでは、これが設定されています。スクリプトに ANSI フォーマットを使用するには、このオプションを選択します。
インターナショナル テキスト (Unicode)	国際データベースは、必要であれば、Unicode ファイル形式をスクリプトに使うことができます。このオプションを選択すると、SQL スクリプトに Unicode BOM エンコーディングが使用されます。

メモ: *スクリプトの形式およびファイル形式に関するさらに詳しい情報は、SQL-SMO ヘルプ レファレンスを参照してください。*

[概要] パネル

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。*

[概要] パネルでは、ウィザードのすべての設定の概要が表示されます。

インストール プロジェクトをビルドする度にスクリプト ファイルを再生する場合、[ビルド時にスクリプトを再生する] オプションをチェックします。

メモ: *[ビルド時にスクリプトを再生成する] オプションを選択すると、ビルド処理に時間がかかります。これはサーバー接続を設立する必要があり、プロジェクトを再ビルドする度にデータベースを再びインポートしなくてはならないためです。*

[完了] をクリックして、スクリプト ファイル (.sql) を生成します。スクリプトが生成された後、[SQL スクリプト] ビューの [スクリプト] タブからこれを編集することができます。

デバイス ドライバー ウィザード...

プロジェクト: *デバイス ドライバー ウィザードは、次のプロジェクトタイプで使用することができます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

デバイス ドライバー ウィザードは Microsoft の Driver Installation Frameworks for Applications (DIFxApp) を使った Windows Installer ベースのインストールからのデバイス ドライバーのインストール処理を簡素化します。ウィザードは必要なテーブルとエントリ、カスタム アクション、機能、及びコンポーネントを作成します。

Driver Install Frameworks に関する最新の情報ついては、http://www.microsoft.com/whdc/ の「Windows Hardware and Driver Central」を参照してください。

タスク

デバイス ドライバー ウィザードを開くには、以下の手順のうちどれかを実行します。

- **[セットアップのデザイン]** ビューで機能を右クリックしてから**デバイスドライバー ウィザード**をクリックします。ウィザードが作成した機能は選択された機能のサブ機能となります。
- **[プロジェクト]** メニューで、**デバイス ドライバー ウィザード**をクリックします。ウィザードが作成した機能はルート階層の機能となります。

デバイス ドライバー ウィザードには、次のような関連パネルがあります。

- ようこそ
- デバイス ドライバー パッケージ
- デバイス ドライバー ファイル
- デバイス ドライバー情報
- プロジェクト全体のデバイス ドライバーの情報
- 概要

[ようこそ] パネル

デバイス ドライバー ウィザードは、Microsoft の Driver Install Frameworks for Applications (DIFxApp) を使った Windows Installer ベースのインストールからのデバイス ドライバーのインストール処理を簡素化します。ウィザードは必要なテーブルとエントリ、カスタム アクション、機能、及びコンポーネントを作成します。

ヒント: *[ツール] メニューの [オプション] コマンドを選択し、オプション ダイアログ ボックスの* ***[ユーザー インターフェイス] タブにある IDE ウィザードの [ようこそ] パネルを表示する*** *オプションを選択解除することで、[ようこそ] パネルを表示しないように設定できます。*

[デバイス ドライバー パッケージ] パネル

[デバイス ドライバー パッケージ] パネルでは、追加するデバイスドライバー用のパッケージファイル (.inf ファイル) を指定します。パッケージの完全修飾ファイル名を入力するか、[参照] ボタンをクリックしてファイルまで移動します。

メモ: *指定したパッケージファイルがセキュリティカタログを指定しない、指定したセキュリティカタログが見つからないない、あるいはそのパッケージファイルが署名されたあとで変更されている、または破損している場合、[次へ] をクリックすると警告メッセージが表示されます。詳細は http://www.microsoft.com/whdc/winlogo/drvsign/drvsign.mspx を参照してください。*

[デバイス ドライバー ファイル] パネル

[デバイス ドライバー ファイル] パネルは、ウィザードがデバイス ドライバーに属すると判断したファイルを表示します。

[デバイス ドライバー情報] パネル

[デバイス ドライバー情報] パネルで、.inf ファイルで指定したデバイス ドライバー タイプを表示して、オプションで次のランタイム オプションを選択することができます。

既存のデバイス ドライバーを常に上書きする

既存のドライバーをこのドライバーで置き換えます。このチェック ボックスがチェックされていないで、デバイスのドライバーが既に存在する場合、ドライバーのインストールは実行されません。

[デバイスをコンピューターへ接続] ダイアログを非表示にする

ドライバーに対応するデバイスがコンピューターへ接続されていない場合、デバイスドライバーのインストール中に [デバイスをコンピューターへ接続] ダイアログが表示されないようにするには、このオプションを選択します。

デバイス ドライバーの [プログラムの追加と削除] エントリを作成しない

このチェック ボックスを選択すると、インストールで、デバイス ドライバーの [プログラムの追加と削除] エントリは作成されません。このチェック ボックスをクリアすると、インストールで [プログラムの追加と削除] エントリが作成されます。

DIFxApp は Windows Vista 以降のシステム上では、この機能をサポートしません。

署名がないドライバー ファイルとファイルが足りないドライバーをインストールする

デフォルトでは、DIFxApp は、署名されていないドライバー パッケージや、ファイルが足りないドライバー パッケージをインストールすることができません。このデフォルトの動作をオーバーライドするには、このチェック ボックスを選択します。このチェック ボックスを選択すると、最終版の署名済みバージョンを出荷する前に完全なテストを手軽に行うことができます。

アンインストール時にドライバーに関連付けられているバイナリ ファイルを削除する

デフォルトで、DIFxApp は、ドライバー パッケージをアンインストールするとき、ドライバーをインストールしたときにシステムにコピーされたバイナリ ファイルを削除しません。このデフォルトの動作をオーバーライドするには、このチェック ボックスを選択します。

このチェック ボックスを選択すると、DIFxApp は、バイナリ ファイルがドライバー ストア内の対応するバイナリ ファイルと同一の場合のみ、そのバイナリ ファイルをシステムから削除します。

注意 *・ドライバーのバイナリ ファイルが他のドライバー パッケージまたはアプリケーションに必要ではないことを確認できるときのみ、[**アンインストール時にドライバーに関連付けられているバイナリ ファイルを削除する**] チェック ボックスを選択してください。*

デバイス ドライバー タイプ

[デバイス ドライバー情報] パネルは、次の要領で決定されるデバイス ドライバーの種類を表示します:

- パッケージファイル (.inf ファイル) の [Version] セクションに次のテーブルの値のどれかを含む DriverPackageType エントリがある場合、対応するドライバータイプが表示されます。
- それ以外の場合、表示されるデバイス ドライバー の種類は Plug and Play ドライバーです。

テーブル 37 ・ ドライバー パッケージ タイプの値と対応するドライバー タイプ

DriverPackageType 値	対応するドライバー タイプ
ClassFilter	クラスフィルター ドライバー
FileSystem	ファイルシステム ドライバー
FileSystemFilter	ファイルシステム フィルタードライバー
KernelModule	エクスポート ドライバー
KernelService	カーネルサービス ドライバー
Network	ネットワーク ドライバー
PlugAndPlay	プラグアンドプレイ ドライバー

InstallShield は Microsoft Windows Driver Install Framework (DIFx) を使ってドライバーをインストールします。デバイスドライバーがインストールされた時、DIFx はそれを [デバイスドライバー情報] パネルで表示されたタイプのデバイスドライバーとして取り扱います。

デバイス ドライバーのタイプを変更するには、パッケージ ファイル [バージョン] セクションにある DriverPackageType エントリの値を、前のテーブルから適切な値に変更してからパッケージファイルを再署名します。(パッケージファイルの署名についての詳細は、 http://www.microsoft.com/whdc/winlogo/drvsign/drvsign.mspx を参照してください。

プロジェクト全体の [デバイスドライバー情報] パネル

[デバイス ドライバーの情報] パネルで構成する設定は、このプロジェクトのすべてのデバイス ドライバーに適用されます。設定は、以下のとおりです。

ローカライズ済みランタイム ダイアログをすべて含める

これをチェック ボックスを設定すると、インストールは以下の言語のダイアログおよびテキストを含むカスタム アクション ランタイム .dll を使用します。

- アラビア語 (サウジアラビア)
- 中国語 (中国)
- 中国語(台湾)
- チェコ語 (チェコ)
- デンマーク語 (デンマーク)
- オランダ語 (オランダ)
- 英語 (U.S.)
- フィンランド語 (フィンランド)
- フランス語 (フランス)
- ドイツ語 (ドイツ)
- ギリシャ語 (ギリシャ)
- ヘブライ語 (イスラエル)
- ハンガリー語 (ハンガリー)
- イタリア語 (イタリア)
- 日本語 (日本)
- 韓国語 (韓国)
- ノルウェー語 (ブークモール) (ノルウェー)
- ポーランド語 (ポーランド)
- ポルトガル語 (ブラジル)
- ポルトガル語 (ポルトガル)
- ロシア語 (ロシア)

- スペイン語 – モダン ソート (スペイン)
- スウェーデン語 (スウェーデン)
- トルコ語 (トルコ)

このチェック ボックスをクリアすると、インストールは英語のランタイム ダイアログのみを使用します。英語のランタイム .dll ファイルはローカライズされた .dll ファイルよりも小さいため、容量的に問題がある場合や追加の言語ランタイム ダイアログが不要な場合は、このチェック ボックスをクリアします。

デバイス ドライバー マシン アーキテクチャ

このエリアには、いくつかのオプションがあります。適切なオプションを選択します。

- デバイス ドライバーは 32 ビット マシンをターゲットする
- デバイス ドライバーは Itanium 64 ビット マシンをターゲットする
- デバイス ドライバーは AMD 64 ビットマシンをターゲットにする

[概要] パネル

[概要] パネルにはウィザードで指定したオプションが表示されます。[完了] ボタンをクリックして、ウィザードを完成します。

ダイアログ ウィザード

InstallShield は、ウィザードパネルに従って新しいダイアログをインストールに追加し構成できるダイアログ ウィザードを提供しています。

タスク ***ダイアログウィザードを起動するには、以下の手順を実行します。***

1. **[ダイアログ]** ビューを開きます。
2. **[ダイアログ]** エクスプローラーで、**[すべてのダイアログ]** を右クリックしてから、**[新しいダイアログ]** をクリックします。ダイアログウィザードが起動します。
3. ウィザードパネルに従います。

ダイアログ ウィザードには、次の関連パネルがあります。

- ようこそ
- ダイアログ テンプレート
- ユーザーインターフェイスシーケンス:
- ダイアログの位置と条件

[ようこそ] パネル

[ようこそ] パネルはダイアログ ウィザードの初期パネルで、このウィザードの機能についての簡単な説明を表示します。ウィザードの使用を開始するには、[次へ] をクリックします。

[ダイアログ テンプレート] パネル

[ダイアログ テンプレート] パネルを使って、デフォルト テンプレートを選択するか、新しいダイアログの基礎となる既存のダイアログを選択します。

パネルのオプション

ダイアログ リスト

[ダイアログ テンプレート] パネル内のボックスには、[オプション] ダイアログ ボックスの [ファイルの場所] タブにある [ダイアログの場所] ボックスで指定したディレクトリにあるダイアログが一覧表示されます。リポジトリに格納されているダイアログも含まれます。さらに、いくつかのデフォルトの ダイアログ テンプレートも一覧となっています。

テーブル 38 ・ ダイアログ テンプレートのオプション

ダイアログ テンプレート	説明
ブランク ダイアログ	ダイアログコントロールを含まない空白のダイアログを作成します。 ***プロジェクト:*** *InstallScript および InstallShield MSI プロジェクトでは、このダイアログ テンプレートには、コントロール ID 値 2 を持つ非表示コントロールが含まれていません。NewScriptBasedDialog および NewSkinnableDialog テンプレートには、このコントロールが含まれています。エンド ユーザーが InstallScript ダイアログの右上にある閉じるボタンをクリックしてインストールをキャンセルできるようにするには、カスタム ダイアログにコントロール ID プロパティがこの値に設定されているボタン コントロールを含まなくてはなりません。詳細については、「InstallScript を使用してカスタム ダイアログを実装する」を参照してください。*
外部ウィザードパネル	***プロジェクト:*** *このダイアログ テンプレートは、基本の MSI プロジェクトで使用できます。* [ようこそ] ダイアログのフォーマットに似た、ダイアログの左側に大きなビットマップを持つダイアログを作成します。
内部ウィザードパネル	***プロジェクト:*** *このダイアログ テンプレートは、基本の MSI プロジェクトで使用できます。* ダイアログの上部に小さなビットマップを持つダイアログを作成します。

テーブル 38 • ダイアログ テンプレートのオプション (続き.)

ダイアログ テンプレート	説明
ログオン情報パネル、および関連する子ダイアログ	 ***プロジェクト•*** *このダイアログ テンプレートは、基本の MSI プロジェクトで使用できます。* *基本の MSI、InstallScript、または InstallScript MSI プロジェクトへの [ログイン情報] ダイアログの追加に関する情報は、「既存のユーザー アカウントを作成または設定する機能を追加する」を参照してください。* エンド ユーザーが別のユーザーには制限されているリソースへアクセスするために、既存するユーザー アカウントを作成および参照することを可能にする一連のダイアログを作成します。
NewScriptBasedDialog	 ***プロジェクト•*** *このダイアログは、次のプロジェクト タイプで使用できます:* • *InstallScript* • *InstallScript MSI* 内部ダイアログに含まれているのと同じコントロールを含む、スクリプトベースの新しいダイアログを作成します。このダイアログは [戻る]、[次へ]、[キャンセル]、[タイトル]、および [サブタイトル] といった基本のコントロールを含みます。
NewSkinnableDialog	 ***プロジェクト•*** *このダイアログは、次のプロジェクト タイプで使用できます:* • *InstallScript* • *InstallScript MSI* 内部ダイアログに含まれているのと同じコントロールのいくつかを含む、スクリプトベースの*スキン対応*ダイアログを作成します。このダイアログは [戻る]、[次へ]、[キャンセル] といった基本のコントロールを含みます。ダイアログ スキンを利用する予定であれば、このオプションを選択してください。 ***注意 •*** *このダイアログにコントロールを追加する場合、コントロール ID を 52 に設定しないでください。この値に設定されると、ダイアログ スキンが使えなくなります。*

レポジトリのダイアログを表示する

レポジトリに格納されているダイアログを表示するには、[レポジトリのダイアログを表示する] チェック ボックスを選択します。

参照

一覧に含まれていないダイアログ ファイル (.isd) を選択するには、[参照] ボタンをクリックします。

このダイアログをシーケンスに挿入する

このチェック ボックスを選択すると、新しいダイアログが [ユーザー インターフェイス] シーケンスへ自動的に挿入されます。このチェック ボックスは基本の MSI プロジェクトの内部および外部ダイアログでのみ利用可能です。InstallScript MSI プロジェクトでは、InstallScript を利用して、ユーザー インターフェイス内でのダイアログの外観をスケジュールし、ダイアログのコントロールを処理しなくてはなりません。

メモ: 基本の MSI プロジェクトでこのオプションが選択されていない場合、ウィザードの処理が終了し、新しいダイアログがプロジェクトへ追加されます。インストール中に表示するためには、シーケンスへ手動でダイアログを追加する必要があります。

基本の MSI プロジェクト

外部、内部、またはログイン ダイアログを選択した場合、[次へ] をクリックして次のパネルへ移動します。空白ダイアログ、レポジトリにあるダイアログ、または別の場所にあるダイアログを選択した場合、[完了] をクリックしてプロジェクトにダイアログを追加します。

メモ: これをインストール中に表示するためには、シーケンス内へ手動で[空白] ダイアログの挿入を行なう必要があります。

InstallScript および InstallScript MSI プロジェクト

[完了] をクリックして、新らしいダイアログを追加します。新しいダイアログをユーザー インターフェイスで表示する為には、これをスクリプトへ追加し、InstallScript を利用してダイアログコントロールを処理します。必要に応じてダイアログ コントロールの編集が可能です。

[ユーザー インターフェイス シーケンス] パネル

[ユーザー インターフェイス] パネルでは、新しいダイアログの挿入先となる [ユーザー インターフェイス] シーケンスを選択します。選択したシーケンス内に現在挿入されているダイアログを表示する、またはある既存のダイアログのあとに新しいダイアログを挿入することもできます。

パネルのオプション

テーブル 39 - [ユーザー インターフェイス シーケンス] パネルのオプション

オプション	説明
ユーザーインターフェイス シーケンス	メニューから新しいダイアログを挿入する先となるシーケンスを選択します。
このシーケンスに含まれるシーケンス	選択したシーケンスに含まれるダイアログを表示します。シーケンス内で、新しいダイアログの直前に表示するダイアログを選択します。

[ダイアログの位置と条件] パネル

[ダイアログの位置と条件] パネルを使って、シーケンス内のどこでダイアログを表示するかを示します。このパネルでは、ダイアログに配置する任意の条件を指定することもできます。

パネルのオプション

テーブル 40 - [ダイアログの位置と条件] パネルのオプション

オプション	説明
ダイアログの位置	このペインには、[ユーザー インターフェイス シーケンス] パネル で選択したシーケンス内の新しいダイアログの位置が表示されます。シーケンス内で新しいダイアログを移動させるには、そのダイアログを選択してから [上へ移動]、または [下へ移動] をクリックします。
条件	このフィールド内に新しいダイアログの条件を設定します。[ビルド] をクリックして、[条件ビルダー] ダイアログ ボックスを起動します。

[完了] をクリックしてウィザードを終了します。InstallShield は、新しいダイアログをプロジェクトに追加し、指定のシーケンスにそれを挿入します。

DirectX オブジェクト ウィザード

プロジェクト- *DirectX オブジェクト ウィザードは、次のプロジェクトの種類で使用することができます:*

- *基本の MSI*
- *InstallScript MSI*

DirectX オブジェクト ウィザードを使って、インストール プロジェクトに Microsoft DirectX 9c 再配布可能ファイルを含み、いくつかのオプションを設定することができます。

メモ: *DirectX オブジェクトに含めるファイルについては、「DirectX 9.0 オブジェクトを含める」を参照してください。*

DirectX 9c の ManagedDX コンポーネント (オプション) は、.NET Framework 1.1 以降がシステムに インストールされている必要があります。

DirectX オブジェクト ウィザードには、次の関連するウィザードパネルがあります。

- ようこそ
- オブジェクトの設定
- 概要

DirectX に関する詳しい情報は、MSDN をご覧ください。

[ようこそ] パネル

プロジェクト: *DirectX オブジェクト ウィザードは、次のプロジェクトの種類で使用することができます:*

- *基本の MSI*
- *InstallScript MSI*

DirectX オブジェクト ウィザードを使って、インストール プロジェクトに Microsoft DirectX 9c 再配布可能ファイルを含み、いくつかのオプションを設定することができます。

メモ: *DirectX オブジェクトに含めるファイルについては、「DirectX 9.0 オブジェクトを含める」を参照してください。*

DirectX 9c の ManagedDX コンポーネント (オプション) は、.NET Framework 1.1 以降がシステムに インストールされている必要があります。

DirectX に関する詳しい情報は、MSDN をご覧ください。

[オブジェクトの設定] パネル

プロジェクト: *DirectX オブジェクト ウィザードは、次のプロジェクトの種類で使用することができます:*

- *基本の MSI*
- *InstallScript MSI*

[オブジェクト設定] パネルでは、次のオプションを設定することができます:

テーブル 41 - [オブジェクトの設定] パネルのオプション

オプション	説明
DirectX ファイルを Disk1 フォルダー内のフォルダーに配置する	このチェック ボックスを選択すると、ビルド時に作成中のインストールに対して DirectX フォルダーが作成され、リリースの Disk1 フォルダーに配置されます。 このチェック ボックスをクリアすると、DirectX 再配布可能ファイルが .msi ファイルにストリームされます。単一実行可能インストールを作成している場合、ファイルが .msi ファイルに含められるようにこのチェック ボックスをクリアにします。 オブジェクトに含める DirectX ファイルについては、「DirectX 9.0 オブジェクトを含める」を参照してください。
DirectX 9 のインストールを開始する前に Microsoft DirectX EULA を表示する	デフォルトでは、Microsoft DirectX EULA が DirectX 9 のインストールが始まる前にエンド ユーザーに対して表示されます。EULA が表示されないようにするには、このチェック ボックスをクリアします。

DirectX に関する詳しい情報は、MSDN をご覧ください。

[概要] パネル

プロジェクト: *DirectX オブジェクト ウィザードは、次のプロジェクトの種類で使用することができます:*

- *基本の MSI*
- *InstallScript MSI*

[概要] パネルでは、[オブジェクトの設定] パネル で構成したオプションを確認することができます。設定を使用して、プロジェクトに DirectX 9 再配布可能ファイルを追加するには [完了] をクリックしてください。

ダイナミック スキャン ウィザード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードは、実行可能ファイルの実行中にシステムを監視する使い易いツールです。ウィザードでは実行可能ファイルが必要とする可能性のある .dll および .ocx ファイルのリストが表示され、それぞれをプロジェクトに含めるかどうかを指定できます。ウィザードを使ってプロジェクトに既に含まれている実行可能ファイルをスキャンするか、スキャン処理を開始する前にウィザードの [実行可能ファイルの指定] パネル を使ってスキャンする新しい実行可能ファイルを選択してプロジェクトに追加することができます。

タスク ***ダイナミック スキャン ウィザードを起動するには、次の手順を実行します。***

1. **[追加ツール]** の下のビュー バー/ビュー リストにある **[依存関係スキャナー]** をクリックします。
2. **[ダイナミック スキャンの実行]** ボタンをクリックします。

ダイナミック スキャン ウィザードには、次のような関連パネルがあります。

- ようこそ
- ファイルのフィルター
- 実行プログラムの指定
- アプリケーション ファイルの指定
- アプリケーションの起動
- アプリケーションの実行中
- ファイルの選択
- スキャン結果
- ダイナミック スキャン ウィザードの完了

[ようこそ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードを使って、アプリケーションの依存ファイルを簡単にプロジェクトに追加できます。スキャナーを使用する前に、対象の実行可能ファイルをプロジェクトに追加することをお勧めします。

ウィザードの使用を開始するには、[次へ] をクリックします。

[ファイルのフィルター] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードは、インストールに追加する必要のないファイルを依存関係としてリストする場合があります。例えば、ターゲット マシン上に既存する一般的なシステムファイルは通常、再インストールする必要がありません。スキャナーを実行した時にこれらのファイルがプロジェクトに追加されることを回避するには、[ファイルのフィルター] パネルで **[ファイルのフィルター]** チェック ボックスを選択します。

スキャンから除外するファイル リストをカスタマイズする方法については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

[実行可能ファイルの指定] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[実行可能ファイルの指定] パネルでは、プロジェクトに既に含まれている実行可能ファイル、またはまだ追加されていない実行可能ファイルをスキャンするかどうかを選択します。

テーブル 42 • [実行可能ファイルの指定] パネルの設定

設定	説明
マイプロジェクトから実行可能ファイルを選択する (推奨)	スキャンする実行可能ファイルがプロジェクトに既に追加されている場合は、このオプションを選択します。
新しい実行可能ファイルを選択する	スキャンする実行可能ファイルが現在プロジェクトに含まれていない場合は、このオプションを選択します。

[アプリケーション ファイルの指定] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[アプリケーション ファイルの指定] パネルでは、スキャンする特定の実行可能ファイル (.exe) を選択します。さらに、ファイルや作業フォルダーのコマンドラインパラメーターを指定することができます。

テーブル 43 • [アプリケーション ファイルの指定] パネルの設定

設定	説明
アプリケーション	スキャンする実行可能ファイルを指定します。プロジェクトに既に含まれているファイルをスキャンする場合は、プロジェクトに既存する実行可能ファイルのリストから選択します。まだプロジェクトの一部になっていないファイルをスキャンする場合は、ファイルへのパスを入力するか、[参照] ボタンをクリックしてファイルへ移動します。
コマンド ライン	実行可能ファイルに渡すコマンドライン パラメーターを入力します。これらのパラメーターはスキャン処理中のみ使用され、ウィザードの終了後は使用されません。
作業用フォルダー	このアプリケーションに対する作業フォルダーへのパスを入力するか、[参照] ボタンをクリックしてディレクトリに移動します。デフォルトでは、ディレクトリは、スキャンするアプリケーションが格納されているフォルダーと同じフォルダーに設定されています。

アプリケーションの起動

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードによるアプリケーションのスキャンを実行する前に、該当するアプリケーションの起動が必要です。ウィザードによるアプリケーションの起動後、アプリケーションのメニュー項目や機能が使用可能になっているか、できるだけ多くを試してみてください。これは、依存関係ファイルの場所を見つけてプロジェクトに追加するのに役立ちます。

[次へ] をクリックして、アプリケーションを起動し、依存関係のスキャンを開始します。

[アプリケーションが実行中です] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[アプリケーションが実行中です] パネルはアプリケーションを実行している間、表示されます。アプリケーションを終了してから、[完了] ボタンをクリックすると、スキャン結果が表示されます。スキャン結果を確認しなければ、ファイルは追加されません。

[ファイル選択] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[ファイルの選択] パネルには、プロジェクトに追加する必要がある可能性の高いファイルおよびマージ モジュールのリストが表示されます。このパネルを使って、インストールに含めるファイルおよびマージ モジュールを選択できます。詳細については、「依存関係スキャナー結果の確認」を参照してください。

テーブル 44 · [ファイル選択] パネルの設定

設定	説明
File	該当するチェック ボックスを選択して、プロジェクトに追加するファイルおよびマージ モジュールを指定します。
すべて選択解除	すべてのチェック ボックスをクリアする場合は、このボタンをクリックします。その後、プロジェクトに追加するファイルまたはモジュールに対応する各チェック ボックスを手作業で選択できます。
すべて選択	すべてのチェック ボックスを選択する場合は、このボタンをクリックします。その後、プロジェクトに追加するファイルまたはモジュールに対応する各チェック ボックスを手作業でクリアできます。

[スキャン結果] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[スキャン結果] パネルには、ウィザードが識別した依存関係の中からプロジェクトに追加することが選択されたファイルが表示されます。

これらの依存関係をプロジェクトに追加するには、[次へ] ボタンをクリックします。依存関係を追加しないでウィザードを終了するには、[キャンセル] ボタンをクリックします。再び依存関係の可能性があるファイルのリストを表示して、それらを追加または削除するには、[戻る] ボタンをクリックします。

[ダイナミック スキャン ウィザード完了] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ダイナミック スキャン ウィザードが [ダイナミック スキャン ウィザードの完了] パネルを表示する段階で、ウィザードは実行可能ファイルの依存関係をプロジェクトに追加済みです。まだプロジェクトに含まれていない実行可能ファイルのスキャンを選択した場合、その .exe ファイルも追加されます。

[完了] をクリックすると、ウィザードを閉じて InstallShield に戻ります。

コンポーネントのエクスポート ウィザード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントのエクスポート ウィザードでは、特定のファイルの種類 (.ism、.msi、.msm) から既存のプロジェクトまたは新しいプロジェクトにコンポーネントを簡単にエクスポートすることができます。このウィザードは、コンポーネントをマージ モジュールにエクスポートする際によく使用されます。

メモ: *コンポーネントのエクスポート ウィザードでは、.msi ファイルまたは .msm ファイルから .ism ファイルへコンポーネントをエクスポートすることはできません。また、.ism ファイルから .msi ファイルや .msm ファイルにエクスポートすることもできません。*

コンポーネントを他のプロジェクトにエクスポートする場合、このコンポーネントのコピーが、ファイル、ショートカット、レジストリ エントリ、詳細設定など、すべてのコンポーネントのデータと共に、指定したプロジェクトファイルに追加されます。

タスク ***コンポーネントのエクスポートウィザードを起動するには、以下の手順を実行します。***

1. **[編成]** の下にあるビュー バー/ビュー リストで、**[セットアップのデザイン]** (インストール プロジェクトのみ) または **[コンポーネント]** をクリックします。
2. 以下のいずれかを実行します。

- 特定のコンポーネントをエクスポートするには、エクスポートするコンポーネントを右クリックして、[**コンポーネントのエクスポートウィザード**] をクリックします。
- 複数のコンポーネントをエクスポートするには、このビューのメイン ノードを右クリックして、[**コンポーネントのエクスポートウィザード**] をクリックします。

メモ: *[プロジェクト] メニューからコンポーネントのエクスポートウィザードを起動することもできます。*

コンポーネントのエクスポート ウィザードには、次のような関連パネルがあります。

- ようこそ
- コンポーネントの選択
- ターゲット ファイルの情報を選択
- 概要

[ようこそ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントのエクスポート ウィザードはファイルの種類 (ism、msi、msm) から既存のプロジェクトまたは新しいプロジェクトにコンポーネントをエクスポートする処理を簡単にします。このウィザードは、コンポーネントをマージ モジュールにエクスポートする際によく使用されます。

メモ: *コンポーネントのエクスポート ウィザードでは、.msi ファイルまたは .msm ファイルから .ism ファイルへコンポーネントをエクスポートすることはできません。また、.ism ファイルから .msi ファイルや .msm ファイルにエクスポートすることもできません。*

[コンポーネントの選択] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネントの選択] パネルでは、開いているプロジェクト内にあるすべてのコンポーネントが表示されます。エクスポートするコンポーネントを選択します。このパネルのコンポーネントリストを機能またはコンポーネントのインストール先別にフィルタリングできます。

特定のコンポーネントを右クリックし、[コンポーネントのエクスポート ウィザード] をクリックしてコンポーネントのエクスポート ウィザードを起動した場合、[コンポーネントの選択] パネルはスキップされます。

メモ: *マージ モジュールからエクスポートする場合、マージ モジュールには機能がないため、機能のフィルターは利用できません。*

[ターゲット ファイルの選択] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ターゲット ファイルの選択] パネルでは、コンポーネントを既存または新規プロジェクトにエクスポートするかどうかを指定できます。いずれの場合もファイルの場所は参照してください。

メモ: *ウィザードで、コンポーネントが対象のプロジェクトにエクスポートされている時、競合の存在が確認されます。対象のプロジェクトが既に異なる設定で同じ名前のコンポーネントを持つ場合は、[競合の解決] ダイアログ ボックスが表示され、競合を解決するオプションを選択できます。*

また、マージ モジュールプロジェクトとしてエクスポートするオプションもあります。コンポーネントをこの特定のプロジェクトの種類に抽出するには、このオプションを選択します。

[ターゲット ファイル情報] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ターゲット ファイル] パネルでは、プロジェクト名、バージョン、言語など、新しいプロジェクトの情報を指定します。現在のプロジェクト内にあるエクスポート済みコンポーネントを生成されたマージモジュールで置換するかどうかを指定します。コンポーネントをエクスポートされたマージ モジュールで置き換えると、現在開いているプロジェクトからコンポーネントが削除されます。

[概要] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

概要パネルにはエクスポート処理の結果が表示されます。また概要には、エクスポートの結果生じたエラーも表示されます。処理を完成するには [完了] をクリックします。

メモ: *[概要] パネルに到達した後、[戻る] をクリックして前のパネルに戻ることはできません。*

関数ウィザード

関数ウィザードは、スクリプトに関数呼び出しを追加するプロセスを自動化します。2 つの簡単なウィザードパネルを完了するだけで、関数呼びだしがスクリプトのキャレットの場所に挿入されます。

関数ウィザードの起動

タスク ***関数ウィザードを起動するには、以下の手順に従います:***

- **[編集]** メニューで、**[InstallScript 関数の挿入]** をクリックします。

[関数名] パネル

アクティブなスクリプト エディター ウィンドウに貼り付ける InstallScript 関数を選択できます。

関数ウィザードでは、すべての関数のアルファベット順リストか、特定の関数カテゴリのリストから関数を選ぶことができます。ハイライトした各関数の説明を表示することができます。ハイライトされた関数についての完全な説明を見るには、[ヘルプ] ボタンを押してください。

パネルのオプション

関数カテゴリ (リストボックス)

関数カテゴリを選択します。選択したカテゴリに所属する InstallScript 関数が [関数名] リストボックスに表示されます。使用可能なすべての InstallScript 関数を表示する場合、[すべて] を選択します。

関数名 (リストボックス)

関数名を選択すると、関数パラメーターとその関数の簡単な説明が、このパネルの下部に表示されます。

[次へ] をクリックすると、関数ウィザードの [関数パラメーター] パネル が表示されます。

[関数のパラメーター] パネル

[関数名] パネル で選択された関数の名前とその関数のパラメーターを表示します。各パラメーターの横には、変数名または値を入力できる編集ボックスか、パラメーター値を選択できるドロップダウンメニューがあります。

タスク ***関数パラメーターを指定して関数をスクリプトに挿入するには、次の操作を実行します。***

1. 各関数の引数を入力します。編集フィールドにカーソルを合わせると、パネルの下にパラメーターの説明が表示されます。パラメーターに対して定義済みのオプションが使用できる場合は、編集フィールドにドロップダウンメニューが表示されます。オプションが排他的でない場合も、ドロップダウンメニューからは 1 つしかオプションを選択できないことに注意してください。

   メモ・ *編集フィールドには文字列 (または文字列識別子) を入力することができますが、パラメーターの目的には特別に注意を払ってください。変数名の InstallScript ハンガリー表記 は 1つの手がかりになります。*

   すべての関数呼び出しで使う変数名をすべて宣言する必要があります。関数ウィザードはこれを行いません。

2. [**完了**] をクリックすると、セットアップスクリプトに呼び出す関数が挿入されます。さらにサポートが必要な場合は、関数名にカーソルを置いて **F1** キーを押してください。

DIM のインポート ウィザード

プロジェクト・ *この情報は、基本の MSI プロジェクトに適用します。*

DIM のインポート ウィザードを利用して、静的に基本の MSI プロジェクトにインポートする DIM を判別できます。ウィザードを完了したら、DIM のプロジェクト データが開いている基本の MSI プロジェクトに統合されます。

DIM の静的なインポートは、デザイン時のタスクですので注意してください。DIM をプロジェクトに静的にインポートした場合、そのデータは、インポートされた時に基本の MSI プロジェクトの一部になります。DIM プロジェクトを後で更新した場合、基本の MSI プロジェクトに、その変更は反映されません。基本の MSI プロジェクトから静的にインポートされた DIM を削除するには、DIM からのファイル、コンポーネント、およびその他の要素を個々に判別して、それらを基本の MSI プロジェクトから手動で削除する必要があります。

タスク ***DIM のインポート ウィザードを起動するには、次の手順に従います。***

1. **[編成]** のビュー バー/ビュー リストにある **[セットアップのデザイン]** をクリックします。
2. **[セットアップのデザイン]** エクスプローラーで、DIM を含める機能を右クリックして、**[DIM のインポート ウィザード]** をクリックします。

DIM のインポート ウィザードでは、次のパネルが表示されます:

- ようこそ
- ソース ファイルの選択
- インポート

[ようこそ] パネル

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

DIM のインポート ウィザードを利用して、静的に基本の MSI プロジェクトにインポートする DIM を判別できます。ウィザードを完了したら、DIM のプロジェクト データが開いている基本の MSI プロジェクトに統合されます。

[ソース ファイルの選択] パネル

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

[ソース ファイルの選択] パネルで、静的に基本の MSI プロジェクトにインポートする DIM のパスとファイル名を指定できます。パスとファイル名を手動で入力せずにファイルへ移動するには、[参照] ボタンをクリックします。

DIM を静的にインポートする前に、基本の MSI プロジェクト ファイル (.ism) をバックアップする場合、バックアップ プロジェクト ファイルのパスとファイル名を指定します。パスとファイル名を指定したら、このパネルの [次へ] ボタンをクリックした時に、バックアップ ファイルが保存されます。

[インポート] パネル

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

[インポート] パネルでは、インポートする DIM プロジェクト ファイルの場所と名前が表示されます。DIM を静的にインポートする前に、基本の MSI プロジェクト ファイル (.ism) のバックアップが作成されるように指定した場合、その場所と名前も表示されます。

インポート プロセスを完了するには、[完了] ボタンをクリックします。

REG ファイルのインポート ウィザード

InstallShield では、他のインストール プロジェクトや InstallShield 以外で作成した既存のレジストリ (.reg) ファイルをインポートできます。

タスク ***[REG ファイルのインポート] ウィザードを起動するには、次の手順に従います。***

1. **[システム構成]** の下のビュー バー/ビュー リストにある **[レジストリ]** をクリックします。
2. *Windows Installer プロジェクトのみ*: [**ビュー フィルター**] リストで、インポートする .reg ファイルを含めるコンポーネントをクリックします。

   ***メモ・** [すべてのアプリケーションデータ] が選択されている場合、[レジストリ] ビューは読取り専用になります。InstallScript プロジェクトのみ: [インストール先コンピューターのレジストリ] ビュー ペインで、インポートする .reg ファイルを含めるレジストリセットを開きます。*
3. **[インストール先コンピューターのレジストリ]** ビュー ペインで、レジストリデータを追加するレジストリ キーを右クリックしてから、**[REG ファイルのインポート]** をクリックします。[Reg ファイルのインポート] ウィザードが開きます。

コンポーネントまたはレジストリ セットに .reg ファイルをインポートすると、そのレジストリ データがコンポーネントまたはレジストリ セットのレジストリ データに追加され、コンポーネントまたはレジストリ セットがエンド ユーザーのシステムにインストールされている場合に、そのシステムに書き込まれます。

[REG ファイルのインポート] ウィザードは、次のパネルで構成されます。

- ようこそ
- レジストリ ファイルのインポート
- 競合オプションのインポート
- 進行状況

[ようこそ] パネル

REG ファイルのインポート ウィザードを利用して、既存のレジストリデータ (.reg) を InstallShield プロジェクトにインポートすることができます。このレジストリデータは、セットアップ プロジェクト中にターゲット システムのレジストリに追加されます。

[次へ] をクリックして次のウィザード パネルへ進み、REG ファイルのインポートを開始します。

メモ: REG データをインポートする前に、このデータの追加先として正しい機能が選択されているか確認してください。機能を指定するには、ウィザードをキャンセルして、[レジストリ] ビューの [機能] リストから該当する機能を選択します。

[レジストリ ファイルのインポート] パネル

このパネルでは、インポートする REG ファイルを指定できます。

パネルのオプション

レジストリ ファイル

インポートする REG ファイルへのパスを入力するか、[参照] ボタンをクリックしてこのファイルに移動します。

[競合オプションのインポート] パネル

インポートしている REG ファイル内に格納されている情報と競合する可能性のあるレジストリデータが、すでにセットアップに含まれていることがあるので、このような競合をどのように処理するかを選択できます。

パネルのオプション

レジストリ データを上書きする

インポートしている REG ファイル内のデータで、すでにセットアッププロジェクトにある競合データを上書きする場合、このオプションを選択します。

レジストリ データを上書きしない

インポート中に競合が発生した場合、すでにセットアップ プロジェクトにあるデータを保守する場合、このオプションを選択します。競合しないレジストリデータがすべてインポートされます。

すべてのレジストリ競合をファイルにログ記録する

ウィザードで見つかったすべてのレジストリ競合の記録を保管する場合、このオプションを選択します。フィールドにパスとフォルダー名を入力するか、フォルダーボタンをクリックしてファイルを参照することができます。

[インポート進行状況] パネル

このパネルには、REG ファイルのインポートの進行状況が表示されます。[キャンセル] をクリックしてインポートを中止するか、ウィザードにより REG ファイルのインポートが終了するまで待ち、[完了] をクリックし IDE に戻ります。

XML 設定のインポート ウィザード

XML 設定のインポート ウィザードで、XML ファイルのリファレンスを [XML ファイル変更] ビューに追加できます。[XML ファイル変更] ビューでは、実行時に XML ファイルに加える変更を構成することができます。

重要 *[XML ファイルの変更] ビューは、XML ファイルにあるすべてのノードについてノードを表示するようにはデザインされていません。パフォーマンス速度を向上させるために、[XML ファイルの変更] ビューでは、ベースの XML ファイルとは異なる設定のみが表示されます。*

- *変更する XML ファイルがインストールの一部である場合、[XML ファイル変更] ビューには、XML ファイルが実行時にインストールされた後に追加、変更、または削除されるノード、またはノード セットのみが表示されます。*
- *変更する XML ファイルがターゲット システムに既に存在するファイルの場合、[XML ファイル変更] ビューには、実行時に追加、変更、または削除されるノードのみが表示されます。*

したがって、XML 設定のインポート ウィザードを使用するとき、実行時に変更する XML ファイル内のノードのみをインポートします。

[XML ファイルの変更] ビューでは、新しい要素が XML ファイルに表示されるときの順番を指定することはできませんので注意してください。従って、要素が特定の順番で表示されている必要がある製品の場合、実行時に変更するノードのみをインポートすると、潜在的な問題を回避することができます。

ウィザードには、以下のパネルがあります。

- ようこそ
- XML ファイル
- XML 要素
- XML インポート処理状況
- XML インポート結果

[ようこそ] パネル

XML 設定のインポート ウィザードで、XML ファイルのリファレンスを [XML ファイル変更] ビューに追加できます。[XML ファイル変更] ビューでは、実行時に XML ファイルに加える変更を構成することができます。

[XML 設定のインポート] ウィザードでは、最初に [ようこそ] パネルが表示されます。

重要 *[XML ファイルの変更] ビューは、XML ファイルにあるすべてのノードについてノードを表示するようにはデザインされていません。パフォーマンス速度を向上させるために、[XML ファイルの変更] ビューでは、ベースの XML ファイルとは異なる設定のみが表示されます。*

- *変更する XML ファイルがインストールの一部である場合、[XML ファイル変更] ビューには、XML ファイルが実行時にインストールされた後に追加、変更、または削除されるノード、またはノード セットのみが表示されます。*
- *変更する XML ファイルがターゲット システムに既に存在するファイルの場合、[XML ファイル変更] ビューには、実行時に追加、変更、または削除されるノードのみが表示されます。*

したがって、XML 設定のインポート ウィザードを使用するとき、実行時に変更する XML ファイル内のノードのみをインポートします。

[XML ファイルの変更] ビューでは、新しい要素が XML ファイルに表示されるときの順番を指定することはできませんので注意してください。従って、要素が特定の順番で表示されている必要がある製品の場合、実行時に変更するノードのみをインポートすると、潜在的な問題を回避することができます。

[XML ファイル] パネル

[XML ファイル] パネルでは、インポートする XML ファイルを指定するには、[参照] をクリックして XML ファイルを検索し、フィールドに情報を挿入します。そして [次へ] をクリックして先に進みます。

[XML 要素] パネル

[XML 要素] ビューでは、[XML ファイルの変更] ビューにインポートする XML ファイル内の要素を指定します。そして [インポート] をクリックして処理を開始するか、[戻る] をクリックして 1 つ前のパネルへ戻ります。

重要: *[XML ファイルの変更] ビューは、XML ファイルにあるすべてのノードについてノードを表示するようにはデザインされていません。パフォーマンス速度を向上させるために、[XML ファイルの変更] ビューでは、ベースの XML ファイルとは異なる設定のみが表示されます。*

- *変更する XML ファイルがインストールの一部である場合、[XML ファイル変更] ビューには、XML ファイルが実行時にインストールされた後に追加、変更、または削除されるノード、またはノード セットのみが表示されます。*
- *変更する XML ファイルがターゲット システムに既に存在するファイルの場合、[XML ファイル変更] ビューには、実行時に追加、変更、または削除されるノードのみが表示されます。*

したがって、XML 設定のインポート ウィザードの [XML 要素] パネルでは、実行時に変更する XML ファイルのノードのみを選択します。

[XML ファイルの変更] ビューでは、新しい要素が XML ファイルに表示されるときの順番を指定することはできませんので注意してください。従って、要素が特定の順番で表示されている必要がある製品の場合、実行時に変更するノードのみをインポートすると、潜在的な問題を回避することができます。

[XML 進行状況] パネル

[XML 要素のインポート] パネルにある [インポート] をクリックすると、[XML 進行状況] パネルが表示されます。これは、ファイルインポートの進行状況を表示します。

[XML 結果] パネル

[XML 結果] パネルには、インポートされた .xml ファイルの概要とその要素がすべて表示されます。[完了] ボタンをクリックして、インポート処理を完了し、ウィザードを終了します。

[完了] をクリックすると、インポートした XML ファイルの新しいノードが追加されます。XML ファイル ノードには、ウィザードで選択したノードがそれぞれ含まれています。各ノードは、実行時に発生する XPath クエリを示します。[XML ファイルの変更] ビューでインポートした XML ファイルの新しいコンポーネントも追加されます。インポートが完了したら、[XML ファイルの変更] ビューを使って、XML ファイルの設定を構成し、その要素、属性、および内容の変更を指定することができます。

基本的な XPath 式の作成方法については、「XPath 式を使って、XML ファイル内の XML データを検索する」を参照してください。

Microsoft .NET Framework オブジェクト ウィザード

プロジェクト: *Microsoft .NET Framework オブジェクト ウィザードは、InstallScript プロジェクトで提供されています。*

Microsoft .NET Framework オブジェクト ウィザードは、InstallScript プロジェクトの [オブジェクト] ビューで Microsoft .NET Framework オブジェクトを機能に追加したとき、開きます。このウィザードでは、プロジェクトに含める .NET Framework と言語パックのバージョンを指定することができます。言語パックには、英語以外の言語のための翻訳済みテキスト (エラー メッセージなど) が含まれています。

Microsoft .NET Framework オブジェクト ウィザードは、以下のパネルから構成されています。

- ステップ 1
- ステップ 2

ヒント: *このオブジェクトは、InstallShield と共にインストールされません。ご利用には、ダウンロードが必要です。詳細については、「InstallShield のアップデートを取得する」を参照してください。*

[ステップ 1] パネル

プロジェクト: *Microsoft .NET Framework オブジェクト ウィザードは、InstallScript プロジェクトで提供されています。*

Microsoft .NET Framework オブジェクト ウィザードの [ステップ 1] パネルでは、プロジェクトに含める .NET Framework と言語パックのバージョンを指定することができます。

テーブル 45 - [ステップ 1] パネルの設定

設定	説明
含めてインストールする .NET のバージョンを選択する	オブジェクトに含める .NET Framework のバージョンを選択します。
Web ダウンローダ バージョンの .NET 再配布可能ファイルを使用する	Web ダウンローダ再配布可能ファイルと完全パッケージ再配布可能ファイルが選択したバージョンの .NET について提供されている場合、このチェック ボックスを有効にします。Web ダウンローダ再配布可能ファイルを使用する場合、このチェック ボックスを選択します。 Web ダウンローダ再配布可能ファイルのサイズは完全パッケージ再配布可能ファイルよりも小さいですが、実行時にターゲット システムでインターネット接続が必要になります。
プラットフォーム	オブジェクトに含める .NET Framework のプラットフォーム固有のバージョンを選択します。

テーブル 45 • [ステップ 1] パネルの設定 (続き.)

設定	説明
Languages	オブジェクトに言語パックを含めるかどうかを指定します。 • **言語パックを含めない** — 言語パックを含めない場合、このオプションを選択します。 • **すべての適切な言語パックを含める** — プロジェクトでサポートされている、使用可能なすべての言語をオブジェクトに含めるには、このオプションを選択します。

このオブジェクトに関する詳しい情報は、[オブジェクト] ビューの [Microsoft .NET Framework オブジェクト] をクリックしたときに表示されるヘルプ ペインをご覧ください。

ヒント• *このオブジェクトは、InstallShield と共にインストールされません。ご利用には、ダウンロードが必要です。詳細については、「InstallShield のアップデートを取得する」を参照してください。*

[ステップ 2] パネル

プロジェクト• *Microsoft .NET Framework オブジェクト ウィザードは、InstallScript プロジェクトで提供されています。*

Microsoft .NET Framework オブジェクト ウィザードの [ステップ 1] パネルで **[複数のバージョンの .NET Framework を含める]** を選択すると、[ステップ 2] パネルが表示されます。

テーブル 46 • [ステップ 2] パネルの設定

設定	説明
オブジェクトに含める .NET 再配布可能ファイルを選択する	プロジェクトに含める .NET Framework のバージョンのチェック ボックスを選択します。実行時に、オブジェクトがインストールできる .NET Framework バージョンは、1 つだけですので注意してください。

テーブル 46 - [ステップ 2] パネルの設定 (続き.)

設定	説明
オブジェクトで、対応する .NET Framework をインストールせずに、.NET 言語パックをインストールできるようにする	対応するバージョンの .NET Framework がインストールされる前に、言語パックのインストールを許可するには、このチェック ボックスを選択します。 InstallShield は、この設定を使用して、オブジェクトの InstallLangPackOnly プロパティを構成します。この読み書き可能数値プロパティは、オブジェクトで、対応するバージョンの .NET Framework がインストールされる前に、言語パックのインストールを許可するかどうかを示します。フレームワークがオブジェクトに含まれてない場合、.NET Framework のインストールが先に実行される場合があります。 オブジェクトが特定のバージョンのフレームワークがインストールするように設定されていて、そのバージョンのフレームワークがオブジェクトに含まれていない場合、オブジェクトは、デフォルトで、DOTNET_STATUS_VERSION_NOT_IN_MEDIA のステータスを設定します。ただし、InstallLangPackOnly プロパティが設定されている場合、オブジェクトは失敗ステータスを設定しません。失敗ステータスが設定されないことにより、オブジェクトは、フレームワークを先にインストールすることなく、必要に応じて言語パックをインストールすることができます。このプロパティは通常、オブジェクト ウィザードで設定されます。

このオブジェクトに関する詳しい情報は、[オブジェクト] ビューの [Microsoft .NET Framework オブジェクト] をクリックしたときに表示されるヘルプ ペインをご覧ください。

ヒント- *このオブジェクトは、InstallShield と共にインストールされません。ご利用には、ダウンロードが必要です。詳細については、「InstallShield のアップデートを取得する」を参照してください。*

MSI からMSIX への変換ウィザード

プロジェクト- *この情報は、基本の MSI プロジェクトに適用します。*

MSI からMSIX への変換ウィザードを使って、既存の Windows Installer ファイル（.msi ファイル）を基本の MSI プロジェクトを通して MSIX パッケージに変換することができます。

タスク

MSI からMSIX への変換ウィザードを起動するには、次の手順に従います:

1. **[ホーム]** メニューで、**[新規]** ボタンをクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[Windows インストーラー]** タブで **[MSI からMSIX への変換ウィザード]** をクリックします。
3. **[OK]** をクリックします。[MSI からMSIX への変換ウィザード] が開きます。

[ようこそ] パネル

[MSI からMSIX への変換ウィザード] を使って、MSI パッケージを MSIX パッケージに変換することができます。

タスク

MSI から MSIX に変換するには、次の手順に従います:

1. **[ようこそ]** パネルで **[次へ]** をクリックします。
2. **[参照]** をクリックして、MSI ファイルの場所を指定します。
3. **[次へ]** をクリックします。
4. プロジェクト名を入力します。
5. **[参照]** をクリックして、指定の場所にデータ ファイルを格納します。
6. **[完了]** をクリックします。基本の MSI プロジェクト タイプが開きます。

メモ: *基本の MSI プロジェクト タイプを使って、自動的に MSI ファイルがビルドされます。*

新しい言語ウィザード

エディション: *新しい言語ウィザードは、InstallShield の Premier Edition で提供されています。*

プロジェクト: *新しい言語ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield Premier Edition では、新しい言語ウィザードを使って 35 ヶ国語以外のサポートされていない言語を追加することができます。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をプロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

新しい言語ウィザードを使ってサポートされていない言語を追加すると、InstallShield によって英語の文字列を含むいくつかのファイルがプレースホルダーとしてマシンに追加されます。

- *LanguageID*.ini

  LanguageID は、新しい言語の言語 ID です。InstallShield は、このファイルを次のディレクトリに追加します:

  InstallShield Program Files フォルダー¥Support

- _isres_*LanguageID*.dll

  LanguageID は、新しい言語の言語 ID です。InstallShield は、このファイルを次のディレクトリに追加します:

  InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386

  InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386¥Skins

サポートされていない言語を InstallScript または InstallScript MSI プロジェクトに追加する場合、これらのファイルの文字列を翻訳する必要があります。.dll ファイルをリソース エディターで開いて、英語の文字列を適切な翻訳済みの文字列と置き換えます。

タスク ***新規言語ウィザードを起動するには、次の手順に従います:***

[ツール] メニューで、**[新規言語の追加]** をクリックします。

[ようこそ] パネル

エディション: *新しい言語ウィザードは、InstallShield の Premier Edition で提供されています。*

プロジェクト: *新しい言語ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

InstallShield Premier Edition では、新しい言語ウィザードを使って 35 ヶ国語以外のサポートされていない言語を追加することができます。サポートされていない言語とは、デフォルトのランタイム文字列が全く翻訳されていない言語です。サポートされていない言語をプロジェクトに追加すると、その言語は InstallShield 内の様々な言語関連設定で使用可能となります。さらに、InstallShield は新しく追加されたサポートされていない言語の文字列用のプレースホルダーとして、プロジェクトのデフォルト言語の文字列を使用します。サポートされていない言語に翻訳済みの文字列を提供するには、[文字列エディター] ビューを使用します。

プロジェクト言語パネル

エディション: *新しい言語ウィザードは、InstallShield の Premier Edition で提供されています。*

プロジェクト: *新しい言語ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

[プロジェクト言語] パネルを使って、プロジェクトに追加する言語を指定します。サポートする言語の隣のボックスを選択します。

[プロジェクトファイル] パネル

エディション: *新しい言語ウィザードは、InstallShield の Premier Edition で提供されています。*

プロジェクト: *新しい言語ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

[プロジェクト ファイル] パネルでは、どの既存プロジェクトが新しい言語の追加機能を含むのかを指定します。(つまり、InstallShield が [一般情報] ビューの "セットアップ言語" 設定で選択可能な言語のリストに新しい言語を追加します。)

このパネルではまた、作成する新しいプロジェクトに言語を含めることも可能にするかどうかを指定できます。

プロジェクトへの言語の追加に関する詳細については、「インストール言語の選択」を参照してください。

パネルの設定

テーブル 1 ・[プロジェクト ファイル] パネルの設定

設定	説明
プロジェクト名	[一般情報] ビューの "セットアップ言語" 設定で新しい言語を選択可能にするプロジェクトを選択します。プロジェクトが デフォルトのプロジェクトの場所に保管されていない場合は、[参照] ボタンをクリックしてこのプロジェクトに移動できます。
InstallShield のアップデート	このチェック ボックスを選択すると、作成する新しいプロジェクトは、これらの言語をオプションでサポートします。このチェック ボックスを選択しても、新しい言語が今後のセットアップに自動的に含まれるわけではありません。(その代わり、[一般情報] ビューの "セットアップ言語" 設定で選択可能な言語のリストに新しい言語が追加されます。)

[概要] パネル

エディション: *新しい言語ウィザードは、InstallShield の Premier Edition で提供されています。*

プロジェクト: *新しい言語ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

[概要] パネルには、特定のプロジェクトに追加される新しい言語がリストされます。また、作成するすべての新しいプロジェクトに InstallShield が新しい言語を追加するかどうかを示します。[完了] ボタンをクリックして設定を確定し、新しい言語を使ってプロジェクトの更新を開始します。

新しい言語ウィザードを使ってサポートされていない言語を追加すると、InstallShield によって英語の文字列を含むいくつかのファイルがプレースホルダーとしてマシンに追加されます。

- *LanguageID*.ini

  LanguageID は、新しい言語の言語 ID です。InstallShield は、このファイルを次のディレクトリに追加します:

  InstallShield Program Files フォルダー¥Support

- _isres_*LanguageID*.dll

  LanguageID は、新しい言語の言語 ID です。InstallShield は、このファイルを次のディレクトリに追加します:

  InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386

InstallShield Program Files フォルダー¥Redist¥Language Independent¥i386¥Skins

サポートされていない言語を InstallScript または InstallScript MSI プロジェクトに追加する場合、これらのファイルの文字列を翻訳する必要があります。.dll ファイルをリソース エディターで開いて、英語の文字列を適切な翻訳済みの文字列と置き換えます。

MSI/MSM オープン ウィザード

MSI/MSM オープンウィザードには、既存のインストール プロジェクト (.msi ファイル) やマージ モジュール (.msm ファイル) を、InstallShield ユーザー インターフェイスで変更とビルドが可能な InstallShield インストール プロジェクト (.ism ファイル) に変換するためのツールが用意されています。

タスク ***MSI/MSM オープンウィザードを起動するには、次の操作を実行します。***

1. **[ホーム]** メニューで、**[開く]** ボタンをクリックします。**[開く]** ダイアログ ボックスが開きます。
2. ファイルの種類リストから **[Windows Installer パッケージ (*.msi)]** または **[Windows Installer モジュール (*.msm)]** を選択します。
3. 変換する Windows Installer パッケージまたはマージ モジュールを参照します。
4. **[開く]** ボタンで、矢印をクリックしてから **[ウィザードで開く]** を選択します。

MSI/MSM オープン ウィザードが開きます。

メモ: *パッケージを開くときにウィザードで発生したエラーは、変換したプロジェクトを開いたときに [出力] ウィンドウに表示されます。問題を解決するには、「MSI/MSM 変換エラー」をご覧ください。*

[ようこそ] パネル

MSI/MSM オープンウィザードでは、セットアップパッケージ (.msi) やマージ モジュール (.msm) を InstallShield にインポートできます。これは、ユーザーのためにカスタム セットアップを作成するネットワーク管理者にとって便利な機能です。

パネルのオプション

ダイレクト編集モードで MSI/MSM を開く

プロジェクトファイルを作成せずに、直接データベースを表示し編集する場合、このオプションを選択します。このオプションを選択すると、選択したデータベースが IDE で開きます。この内容は ダイレクトエディター を使って表示および編集できます。

[次へ] をクリックすると、データベースがダイレクトエディターで開きます。

MSI/MSM を InstallShield プロジェクトに変換する

選択したデータベースを InstallShield プロジェクトに変換するにはこのオプションを選択します。変換したデータベースは IDE 内で編集できます。

[次へ] をクリックすると、ウィザードでこのオプションを続けて使用できます。

[ファイル の場所] パネル

[ファイルの場所] パネルで、ウィザードが作成するプロジェクトの名前やそのデータファイルの場所を指定します。

パネルのオプション

プロジェクト名

ウィザードが既存の .msi プロジェクトまたは .msm プロジェクトを InstallShield (.ism) プロジェクトに変換したときに作成されるセットアッププロジェクトの名前を入力します。.ism プロジェクトは、デフォルトの場所に保存されます。

メモ: プロジェクト名は .ism ファイルの名前とプロジェクトのフォルダー構造のルートを決定するため、Windows ファイル名に有効な文字だけを使用する必要があります。

データ ファイルの場所

インポートされたアプリケーションのファイルが .cab ファイルに格納されている場合、それらのファイルは自動的にここで指定された場所に抽出され、格納されます。

プロジェクト名とデータファイルの場所を指定後は、[完了] をクリックして、プロジェクトを変換し、InstallShield IDEで開いてください。

メモ: ウィザードは、プロジェクトを作成中に発生したあらゆるエラーを表示します。問題を解決するには、「MSI/MSM 変換エラー」をご覧ください。

MSP オープン ウィザード

InstallShield で、パッチ プロジェクトを新規作成または編集するために パッチまたは MSP ファイルを開くとき、その前に MSP が適用されるベースの MSI が必要です。InstallShield で初めてパッチ (.msp) プロジェクトを開くと、MSP オープン ウィザードが開きます。

MSP オープン ウィザードは必要なファイル情報を収集してベースの MSI へパッチを適用し、パッチ プロジェクトをダイレクト MSP モード (ダイレクト編集モードと類似) で IDE 上に開きます。MSP オープン ウィザードで指定したデータは保存され、次回 MSP ファイルを開くときにはパッチ プロジェクトが IDE で開きます。

[ようこそ] パネル

このパネルは MSP オープン ウィザードの初期パネルで、このウィザードの機能についての簡単な説明を表示します。ウィザードの使用を開始するには、[次へ] をクリックします。

[基本の MSI パッケージ] パネル

[基本の MSI パッケージ] パネルでは、このパッチを適用する基本の MSI ファイルを指定します。

パネルのオプション

MSP ファイル名

MSP ファイルの名前とパスを提供する読み取り専用フィールド。

基本の MSI ファイル名

(MSP ファイル名編集ボックスで指定した) パッチを適用する基本の MSI ファイルの名前を指定します。[参照] ボタン (省略記号(…)) をクリックして .msi ファイルを検索します。

[完了] をクリックして MSP オープン ウィザードを閉じてパッチ プロジェクトを作成します。パッチは基本の MSI ファイル名編集ボックスで指定された MSI プロジェクトに適用され、パッチ プロジェクト (.msp ファイル) が ディレクト MSP モードで IDE 上で開きます。

トランスフォーム オープン ウィザード

編集または新しいトランスフォーム プロジェクトを作成するために InstallShield がトランスフォーム (.mst) ファイル を開くためには、.mst が適用されるベース .msi ファイルおよび、ベース .msi ファイルに適用されるその他の追加 .mst ファイルが必要です。

InstallShield で、.mst ファイルを初めて開いたとき、トランスフォーム オープン ウィザードが起動します。このウィザードでは、 .mst ファイル、ベース .msi ファイル、および編集の前に適用する追加トランスフォーム指定することができます。このウィザードの最後のパネル、[応答トランスフォームの作成] パネルでは、そのトランスフォームが応答トランスフォームであるかどうかを指定できます。応答トランスフォームには、インストールのユーザー インターフェイス部分のデフォルトの応答も含まれています。応答トランスフォーム作成を選択すると、インストールのユーザー インターフェイス要素が実行され、インストール ウィザードの各パネルのオプションをカスタマイズすることができます。

トランスフォーム オープン ウィザードが完了したとき、トランスフォームが適用され、.mst ファイルが InstallShield のダイレクト MST モード (ダイレクト編集モードに似ています) で開きます。ウィザードで指定されたデータは、次回 .mst ファイルを開いたとき、トランスフォーム プロジェクトが InstallShield で直接開くように保存されます。

タスク

トランスフォームを編集するために開くとき、トランスフォーム オープン ウィザードを起動するには以下の手順を実行します。

1. **[ホーム]** メニューで、**[開く]** ボタンをクリックします。**[開く]** ダイアログ ボックスが開きます。
2. 必要な .mst ファイルを選択します。
3. **[形式を指定して開く]** ボックスで、**[ウィザード]** を選択します。
4. **[開く]** をクリックします。トランスフォーム オープン ウィザードが開きます。

タスク ***新しいトランスフォームを作成するのにトランスフォーム オープン ウィザードを起動するには、以下の手順を実行します。***

1. **[ホーム]** メニューで、**[新規]** ボタンをクリックします。**[新規プロジェクト]** ダイアログ ボックスが開きます。
2. **[Windows Installer]** タブで **[トランスフォーム]** をクリックします。
3. **[プロジェクト名]** ボックスで、トランスフォーム プロジェクトの名前を入力します。
4. **[場所]** ボックスに、トランスフォーム プロジェクトが保存される場所へのパスを入力するか、または [参照] をクリックして場所を検索します。
5. **[OK]** をクリックします。トランスフォーム オープン ウィザードが開きます。

トランスフォームオープンウィザードには、次のような関連パネルがあります。

- ようこそ
- トランスフォーム情報
- 追加のトランスフォーム
- 応答トランスフォームの作成

[ようこそ] パネル

このパネルはトランスフォーム オープン ウィザードの初期パネルで、このウィザードの機能についての簡単な説明を表示します。ウィザードの使用を開始するには、[次へ] をクリックします。

[トランスフォーム情報] パネル

[トランスフォーム情報] パネルでは、このトランスフォームを適用する基本の .msi ファイルを指定します。

パネルのオプション

MST ファイル名

.mst ファイルの名前とパスを提供する読み取り専用フィールド。

基本の MSI ファイル名

(MST ファイル名 ボックスで指定した) トランスフォームを適用する基本の .msi ファイルの名前を指定します。省略記号 (…) をクリックして .msi ファイルを検索します。

[次へ] をクリックして、次のパネルに移ります。

[追加トランスフォーム] パネル

このパネルでは現在トランスフォームオープンウィザードが作成中のトランスフォームに適用する前に適用する追加トランスフォーム (複数可) の名前を入力することができます。たとえば、以前の言語固有の情報を含むカスタマイズまたはトランスフォームを含む場合があります。トランスフォームはリストされた順番に適用されます。

トランスフォームを追加する

タスク ***リストへトランスフォームを追加するには、以下の手順を実行します。***

1. [新規 (挿入)] または [挿入] をクリックします。
2. .mst ファイルの完全パスおよびファイル名を入力するか、省略記号 (...) ボタンをクリックして参照します。

リストからトランスフォームを削除する

タスク ***リストからトランスフォームを削除するには、以下の手順を実行します。***

1. 削除するリスト内のトランスフォームを選択します。
2. [削除] をクリック、または [削除] を押します。

トランスフォームの順番を変更する

タスク ***トランスフォームが適用される順番を変更するには、以下の手順を実行します。***

リストにあるトランスフォームを選択して [上へ移動] ボタンまたは [下へ移動] ボタンをクリックします。

[トランスフォーム オープン ウィザード] が作成中のトランスフォームは最後に適用されます。

注意 *・複数のトランスフォームを使用する場合、それらが適用される順番が非常に重要である点に注意してください。たとえば、新しいプロパティ値を作成する Windows Installer パッケージ用のトランスフォームを作成してから、そのトランスフォームが作成した値を変更する 2 番目のトランスフォームを作成すると、すべてが適切に動作します。しかし、2 番目のトランスフォームを先に適用すると、トランスフォームはプロパティの値を作成する代わりに変更を試みます。その結果エラーが発生します。*

簡単な例として、デフォルトの会社名についてのエラーがあげられます。デフォルトで値が設定されていない場合に最初のトランスフォームでその値を設定すると、新しいプロパティ値が作成されます。オリジナルパッケージおよび最初のトランスフォームをひとまとめに変更する 2 番目のトランスフォームを作成し、2 番目のトランスフォームがデフォルトの会社名を変更する場合、それはプロパティを変更するだけです。しかし、最初のトランスフォーム無しで 2 番目のトランスフォームを適用した場合、Windows Installer はヌル値を別の値に変更するものと勘違いするためエラーが発生します。

[次へ] をクリックして、次のパネルに移ります。

[応答トランスフォームの作成] パネル

このパネルでは、応答トランスフォームの作成を指定することができます。

パネルのオプション

テーブル 2 • [応答トランスフォームの作成] パネルのオプション

オプション	説明
応答トランスフォームの作成	応答トランスフォームを作成するには、このチェック ボックスを選択します。 このオプションを選択すると、[完了] をクリックした時インストールのユーザー インターフェイス部分が実行されます。ファイルは転送されず、コンピューターに変更を加えることもありません。必要に応じてインストールウィザードの各パネルで利用できるオプションをカスタマイズして、このシミュレートインストールを完成します。インストール シーケンスの最後で [インストール] をクリックすると、シミュレート インストールが終了します。InstallShield は応答トランスフォームのデフォルトのインストール値としてシミュレートインストールの間に加えられたすべての変更を保存します。
コマンドラインプロパティ (オプション)	応答トランスフォームを利用する場合、応答トランスフォームへ渡す追加コマンドラインプロパティ (セミコロンで分かれたプロパティ名と値のペア) を指定することができます。これらはパブリックプロパティでなくてはならず、応答トランスフォーム作成中のダイアログ ボックスの表示方法のみを制御します。作成中、[ユーザー インターフェイス] シーケンス以外では継続されません。たとえば、プロパティ/値のペア ARPHELPTELEPHONE=1-111-111-1111 を渡して、コントロールパネル の [プログラムの追加と削除] にあるヘルプ電話番号の値を設定することができます。 応答トランスフォームの作成中にプロパティ/値ペアを渡し、インストール中にシステム構成に基づいて表示されないすべてのダイアログ ボックスを表示することができます (たとえば、Windows NT プラットフォームで Windows 9x 特有のダイアログボックスを表示する場合)。こうして、適切な応答を作成してトランスフォームに含むことができます。

[完了] をクリックして [トランスフォーム オープン ウィザード] を閉じてトランスフォーム プロジェクトを作成します。トランスフォームは[トランスフォーム情報] パネルで指定した .msi ファイルに適用され、トランスフォームプロジェクト (.mst ファイル) は InstallShield の ダイレクト MST モードで開きます。

パブリッシュ ウィザード

パブリッシュウィザードを利用して、選択したインストール要素をリポジトリへ保存します。

パブリッシュウィザードには、次のような関連パネルがあります。

- ようこそ
- リポジトリ

- パブリッシュ情報
- 概要

[ようこそ] パネル

このパネルは、パブリッシュ ウィザードの最初のパネルです。ウィザードの使用を開始するには、[次へ] をクリックします。

[リポジトリ] パネル

パブリッシュ ウィザードの [リポジトリ] パネルでは、インストール要素がパブリッシュされるリポジトリの場所を指定します。

[パブリッシュ情報] パネル

パブリッシュ ウィザードの [パブリッシュ情報] パネルでは、リポジトリへパブリッシュする項目を識別する情報を指定します。ダイアログをリポジトリへパブリッシュする場合、たとえば パブリッシュ情報パネルへ入力した情報は、ダイアログのリポジトリを参照したときにそのダイアログ用に表示されます。

パネルのオプション

名前

パブリッシュするダイアログの名前、テンプレート、スクリプト ファイル、または他のインストール要素を指定します。

発行者

名前を入力します。

説明

このインストール要素について表示する説明を入力します。

[概要] パネル

パブリッシュ ウィザードの [概要] パネルを利用して、リポジトリへパブリッシュする項目の設定を確認することができます。変更する箇所がある場合は、[戻る] をクリックして適切なパネルまで戻り、変更を加えます。

作業が終了したとき、[完了] をクリックします。出力ウィンドウが開き、インストール要素が選択されたリポジトリへ無事にパブリッシュされたかどうかを通知します。

再配布可能ファイル ダウンローダー ウィザード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

再配布可能ファイル ダウンローダー ウィザードを利用して、サードパーティ再配布可能ファイル、マージ モジュールモジュール、前提条件、その他のファイルをローカルマシンに素早くダウンロードすることができます。再配布可能ファイル ダウンローダ ウィザードは、リリース ウィザードの [リリース] ビュー (".NET 1.1/2.0 コア言語" 設定および ".NET 1.1/2.0 言語パック" 設定) から起動することができます。

タスク ***再配布可能ファイル ダウンローダー ウィザードを利用するには、以下の手順に従います:***

1. **[ダウンロードするファイルの選択]** パネルで、ダウンロードする再配布可能ファイル/前提条件を選択し、**[次へ]** をクリックして **[進行状況]** パネルへ移動します。
2. 進行状況バーが完了したら、**[次へ]** をクリックします。
3. 最終パネルには、ウィザードがマシンへダウンロードした再配布可能ファイル/前提条件のリストが表示されます。**[完了]** をクリックしてウィザードを終了します。

Reg-Free COM ウィザード

Reg-Free COM を使用して、インストールに含める Reg-Free COM マニフェストを新規作成、または変更することができます。

メモ: *Reg-Free COM ウィザードを使用する前に、COM ライブラリ (.dll および .ocx ファイル) と、それを使用する実行可能ファイルを InstallShield プロジェクトに追加する必要があります。Reg-Free COM マニフェスト ファイル、実行可能ファイル、COM ライブラリはすべて、ターゲット マシン上の同一のフォルダーにインストールする必要があるので注意してください。*

タスク ***Reg-Free COM ウィザードを起動するには、以下の手順に従います:***

[プロジェクト] メニューで、**[Reg-Free COM ウィザード]** をクリックします。

Reg-Free COM ウィザードには、次のウィザードパネルが関連付けられています。

- ようこそ
- EXE ファイルの選択
- マニフェスト ファイルの選択
- ファイルの選択
- 概要

[ようこそ] パネル

[ようこそ] パネルでは、Reg-Free COM ウィザードが簡単に説明されています。ウィザードの使用を開始するには、[次へ] をクリックします。

メモ: *次からこのパネルを表示しないようにするには、[**今後ようこそパネルを表示しない**] チェック ボックスを選択します。*

[EXE ファイルの選択] パネル

[EXE ファイルの選択] パネルで、作成または変更するマニフェスト ファイルを使用する実行可能ファイルを指定します。

パネルのオプション

実行可能ファイルキー

実行可能ファイルキー リストから実行可能ファイルを選択し、[次へ] を選択します。このファイルは通常、インストール プロジェクトに既に含まれています。

[マニフェスト ファイルの選択] パネル

[マニフェスト ファイルの選択] パネルで、新しいマニフェスト (.manifest) ファイルを作成するか、または、既存のものを変更するかを指定します。マニフェスト ファイルを変更する場合、そのファイルの場所を指定します。

パネルのオプション

プロジェクトに追加される新しいマニフェスト ファイルを作成する

新しいマニフェスト ファイルを作成する場合、このオプションを選択します。

プロジェクトに追加される既存のファイル

既に存在しているが、プロジェクトにはまだ含められていないマニフェスト ファイルを指定する場合、このオプションを選択してから、参照ボタンをクリックしてファイルを選択します。

プロジェクトに既に存在するマニフェスト ファイル

プロジェクトに既に追加されているマニフェスト ファイルを指定する場合、リストからファイルを選択します。

[ファイルの選択] パネル

[ファイルの選択] パネルで、実行可能ファイルによって使用される必須 COM ライブラリ ファイル (.dll または .ocx) を指定します。また追加で、[**ビルド時に COM 情報を抽出**] チェック ボックスを選択して、各ビルド時中に COM 情報を抽出することもできます。このチェック ボックスをクリアすると、ウィザードを完了した時点で一度だけ、COM 情報がマニフェストに挿入されます。

[概要] パネル

Reg-Free COM ウィザードの [概要] パネルを使用して、選択したオプションを確認することができます。変更する箇所がある場合は、[戻る] をクリックして適切なパネルまで戻り、変更を加えます。

作業が終了したとき、[完了] をクリックします。ウィザードは、新規のコンポーネントを、マニフェスト ファイルがキーファイルとして設定された状態で作成します。マニフェスト コンポーネントは、関連付けられた実行可能ファイル コンポーネントと同じインストール先および条件を持ちます。次のファイルの名前付け規則が、マニフェスト ファイルに使用されています。

ExecutableFileName.exe.manifest

たとえば、**MyFile.exe**.manifest は、**MyFile.exe** という名前に実行可能ファイルに作成されます。

ウィザードを再度実行し、既存のマニフェスト ファイルを指定して、追加の COM サーバーを指定できます。

リリース ウィザード

リリース ウィザードを利用して、製品のリリースをビルドし、それぞれのリリースに対する設定を簡単に行うことができます。

タスク ***リリース ウィザードを起動して、セットアップ パッケージをビルドするには、以下のいずれかを実行します。***

- **[ビルド]** メニューで **[リリース ウィザード]** ボタンをクリックします。
- **[リリース]** ビューで **[リリース ウィザード]** 項目をダブルクリックします。
- **[リリース]** ビューのツリー コントロールで製品名またはリリース名を右クリックし、**[リリース ウィザード]** を選択します。

タスク ***以下のいずれかを実行して、リリース ウィザードの省略版を起動することもできます。***

- ツールバー上の [**ビルド**] ボタンをクリックします。
- **[ビルド]** メニューから **[ビルド]** を選択します。
- リリースエクスプローラーでリリース名を右クリックし、[**ビルド**] を選択する。

[ビルド] オプションによって [リリース] ビューで強調表示されているリリースが再ビルドされるか、またはデフォルトの設定を使用して製品の最初のリリースがビルドされます。ビルドのフィードバックおよびすべてのビルド エラーは、[出力] ウィンドウに表示されます。

リリース ウィザードの代わりに、コマンドラインでリリースをビルドすることもできます。

[ようこそ] パネル

このパネルでは、リリース ウィザードの概要を説明します。このウィザードでは、配布可能なリリースのビルドに必要なすべてのステップをガイドします。

ウィザードの使用を開始するには、[次へ] をクリックします。

[製品構成] パネル

プロジェクト: *このトピックは、InstallScript プロジェクトには適用されません。*

製品構成 は、[リリース] ビューの構成の中で最上位のものです。関連しているリリースを共通の製品構成の下にまとめることで、製品設定のプロパティを編集して各リリースに特有の設定を変更できます。

パネルのオプション

新しい製品構成

新しい製品構成でこのリリースをビルドする場合、このオプションを選択します。新しい構成名を指定してください。

既存の製品構成

このオプションを設定して既存の製品構成の下にこのリリースをビルドします。このオプションを選択すると、既存のリリースが上書きされます。

[リリースの指定] パネル

このパネルでは、新しいリリースを作成したり、既存のリリースを再ビルドするオプションがあります。

パネルのオプション

新しいリリース名

新規リリースをビルドする場合は、[新規リリース名] オプションを選択してフィールドに新しい名前を入力します。

既存のリリース名

既存のリリースをビルドする場合は、[既存のリリース名] オプションを選択してドロップダウンリストから名前を選択します。

[フィルターの設定] パネル

プロジェクト: *[フィルターの設定] パネルは、次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

[フィルターの設定] パネルでは、リリースに含める機能、コンポーネント、InstallShield 前提条件、および連鎖 .msi パッケージを選択するように要求されます。

テーブル 3 · [フィルターの設定] パネルのオプション

設定	説明
リリース フラグ	機能、InstallShield 前提条件、連鎖 .msi パッケージをフィルターするには、このリリースに含めるフラグを入力します。複数のフラグを含める場合は、カンマで区切ってください。 サブ機能を含めることはできますが、その親機能を含めることはできません。このような場合、サブ機能はリリースに最上位の機能として組み込まれ、この親はリリースからは除外されます。 ここに入力するリリースフラグは、製品構成フラグと組み合わせて使用します。二者間の関係についての詳しい情報は、「製品構成フラグとリリース フラグの違い」を参照してください。 ***プロジェクト·** "リリース フラグ" オプションは、マージ モジュール プロジェクトには使用できません。*
次の言語でアプリケーション データをフィルターする	各コンポーネントの言語に基づいてアプリケーション データをフィルターするには、このオプションを選択して、適切な言語のチェック ボックスを選択またはクリアします。 1 つ以上の言語のチェック ボックスを選択した場合、言語非依存とマークされたすべてのコンポーネントのほか、一致する言語でマークされたコンポーネントがすべてリリースに含められます。言語が一致しないコンポーネントはすべて除外されます。

[マージ モジュールのオプション]パネル

[マージ モジュールのオプション] パネルでは、新しく作成したモジュールをローカルマージ モジュールギャラリーにコピーすることにより、モジュールを他の InstallShieldがすぐに利用できるように設定できます。

パネルのオプション

ローカル マージ モジュール カタログへの追加

他のプロジェクトが新しいモジュールをすぐに利用できるようにするには、このオプションを選択します。[マージ モジュールの場所] フィールドは [オプション] ダイアログの [マージ モジュール] タブにあります。

[セットアップ言語] パネル

***プロジェクト·** [セットアップ言語] パネルの機能は、使用するプロジェクトの種類によって異なります。*

[セットアップ言語] パネルを使って、インストールの実行時の言語関連の設定を指定できます。

テーブル 4 ・ [セットアップ 言語] パネルの設定

設定	説明
含められているリリース言語	このボックスの機能は、使用するプロジェクトの種類によって異なります: • 基本の MSI、InstallScript MSI プロジェクトでは、この設定を使ってリリースに含めるユーザー インターフェイス言語を指定できます。 プロジェクトの [一般情報] ビューにある "セットアップ言語" 設定で言語が選択されていない場合、その言語はこのボックスで利用可能な言語の一覧には表示されません。 • InstallScript または InstallScript オブジェクト プロジェクトでは、この設定を使用して、各コンポーネントで選択された言語に基づいて特定のコンポーネントを含めたり、その他のコンポーネントを除外したりできます。この設定を使って、リリースに含めるユーザー インターフェイス言語を指定することもできます。コンポーネントに指定された言語が、このボックスで選択された言語のどれにも一致しない場合、そのコンポーネントはリリースに含まれません。また、プロジェクトに含まれる UI 言語がこのボックスで選択された言語の 1 つと一致しない場合、InstallShield はその UI 文字列をリリースに含みません。 デフォルトで、リリースは言語に依存しません。つまり、プロジェクトのコンポーネントまたは UI 文字列はすべてリリースに含まれます。 プロジェクトの [一般情報] ビューで言語が選択されていない場合、その言語はこのボックスで利用可能な言語の一覧には表示されません。
デフォルト値にする	[一般情報] ビューまたは [文字列エディター] ビューで構成されたリリースのデフォルト プロジェクト言語をオーバーライドするには、インストールで使用する適切なデフォルト ユーザー インターフェイス言語を選択してから、このボタンをクリックします。 詳細については、「デフォルトのプロジェクト言語の設定」を参照してください。
セットアップ言語ダイアログを表示する	インストールが完全なユーザー インターフェイスで実行されたときに、インストールで [セットアップ言語の選択] ダイアログを表示する場合、このチェック ボックスを選択します。 ***ヒント:*** *言語の選択ダイアログを表示するには、*Setup.exe *セットアップランチャーが必要です。詳細については、「セットアップランチャーの作成」を参照してください。*

プロジェクト: *基本の MSI プロジェクトと InstallScript MSI プロジェクトでは、[セットアップ言語] パネルはプロジェクトに含まれるユーザー インターフェイス要素にのみ反映されます。アプリケーション データの言語に基づいてコンポーネントを含めるには、「[リリース フィルター] パネル」を参照してください。*

プロジェクトへの言語の追加に関する詳細については、「インストール言語の選択」を参照してください。

[メディアの種類] パネル

このパネルでは、作成するリリースの配布メディアの種類を選択できます。

パネルのオプション

メディアの種類

セットアッププログラムを配布するメディアの種類を選択します。次のオプションから選択できます。

テーブル 5 · [メディア タイプ] パネルのオプション

オプション	説明
CD-ROM	セットアップを CD で配布する場合はこのオプションを選択します。この種類のメディアの最大サイズは、650 MB です。
カスタム	配布するメディアがメディアの種類のリストの中にない場合は、このオプションを選択します。たとえば、フロッピー ディスクや Zip(TM) ディスクでセットアップを配布できます。この場合、使用するメディアの種類によってフォーマットサイズとクラスタサイズが異なるため、これらのサイズを個別に設定する必要があります。
DVD-5	このオプションを選択すると、リリースを 4.38 GB DVD-ROM で出荷できます。
DVD-9	セットアップを 7.95 GB DVD-ROM で出荷する場合は、このオプションを選択します。
DVD-10	このオプションを選択すると、リリースを 8.75 GB DVD-ROM で出荷できます。
DVD-18	セットアップのサイズが特に大きい場合は、リリースを 15.83 GB DVD-ROM メディアで出荷できます。
ネットワークイメージ	セットアップをネットワーク上に配置する場合は、このオプションを選択します。このメディアの種類には、サイズの制限はありません。
Web	(Windows Installer ベースのプロジェクトのみ)リリースを Web 上で配布する場合は、このオプションを選択します。このメディアの種類には、サイズの制限はありません。

***プロジェクト・** インストールで複数のディスクを必要とする場合、InstallShield は自動的に必要な数のディスクに分割します。Windows Installer ベースのプロジェクトの場合、ウィザードは、シーケンスの後半で、ディスクの分割についての詳細を入力するようプロンプトを表示します。*

フォーマットサイズ

[カスタム] メディアタイプを選択すると、メディアの容量をこのフィールドに指定する必要があります。たとえば、インストール プログラムを Zip ディスクで出荷する場合は、メディア サイズを 100MB に設定します。他の種類のメディア選択すると、選択したメディアの最大許容サイズがこのフィールドに表示されます。

(Windows Installer ベースのプロジェクトのみ) バイト数で表したクラスタサイズ

このオプションでは、各ファイルに必要な最小の容量を定義します。ディスクが大きくなると、フォーマットの制約によりクラスタサイズは大きくなります。

[ディスク分割オプション] パネル

***プロジェクト・** このパネルは、InstallScript プロジェクトでは使用できません。*

セットアップのサイズが 1 つのディスクの容量を超える場合、セットアップを複数のディスクに分割して配布することができます。

パネルのオプション

自動

ウィザードで、セットアップを必要数のディスクに自動的に分割するには、このオプションを選択します。セットアップを複数のディスクに分割する場合は、このオプションを選択することをお勧めします。

カスタム

ユーザーが、セットアップに必要なディスクの数を手動で定義し、ブレークポイントを設定するには、このオプションを選択します。このオプションを選択した場合、各ディスクに格納する機能を [ディスクの分割詳細設定] パネルで指定する必要があります。

***注意・** 複数ディスクのセットアップは、ハード ドライブなどの非リムーバブル メディアからは実行できません。複数のディスクにまたがるセットアップの起動テストを行う場合は、セットアップをターゲットメディアに置く必要があります。セットアップをターゲットメディアに置いていない場合、セットアップは Windows Installer サービスの制限事項のため失敗します。*

ディスクサイズを強制

ウィザードで、ディスク イメージが、選択したメディアのサイズを超えていないことを確認するには、このオプションを選択します。ディスク イメージが、指定したサイズの制限を超えた場合、プロジェクトをビルドしようとすると、ビルド エラー −1531 が表示されます。

[ディスク分割詳細設定] パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

このパネルでは、セットアップに必要なディスクの数を指定するようプロンプトされます。また、各ディスクに格納する機能を選択できます。このパネルは、[ディスク分割オプション] パネルで [カスタム ディスク分割] オプションを選択した場合にのみ表示されます。

パネルのオプション

ディスクのマッピング

このダイアログには、最初のビルド時にリリースに組み込まれたすべての機能が一覧表示されます。リリースはビルド設定のスナップショットになるよう設計されるため、リリースの後に作成された機能は、ディスクのマッピングに利用できません。また、それらの機能が現在のリリースに組み込まれることもありません。新しい機能をマップするには、新しいリリースを開始する必要があります。

すべての機能は、デフォルトでは Disk 1 になります。このディスクの名前を変更するには、[ディスクの名前を変更する] ボタンをクリックして、新しい名前を入力します。新しいディスクを追加するには、[新規ディスク] ボタンをクリックします。

タスク ***機能を別のディスクに移動するには、以下の操作を実行します。***

1. 移動する機能を選択します。
2. **[1 つ上に移動]** または **[1 つ下に移動]** ボタンをクリックします。機能は現在のディスクの階層に従って移動し、限界に達すると別のディスクがある場合は次のディスクにスキップします。

ヒント: *セットアップを複数のディスクに分割する場合は、次の点に注意してください。*

- *単一の機能を、複数のディスクに分割することはできません。機能のサイズが大きいため、1 つのディスクに収まらない場合は、その機能をサブ機能 (小さい機能) に分割する必要があります。*
- *セットアップに含めたマージ モジュールは、最初のディスクに格納する必要があります。これは、Windows Installer が最初にデータベース (.msi ファイル) を処理する際に、マージ モジュールのファイルが利用可能でなければならないためです。*

セットアップの分割方法を定義した後で機能を追加すると、セットアップの最後のディスクに機能が追加されます。リリース ウィザードを使用して、いつでもインストールに新しいディスクを追加できます。

ディスク プロンプト

タスク ***次のディスクの挿入が必要であることをエンドユーザーに示すプロンプトを指定するには、次の操作を実行します。***

1. [**ディスクのマッピング**] セクションでディスクを選択します。
2. "**ディスクのプロンプト**" フィールドで次のいずれかの操作を行います。

- エンドユーザーに表示するプロンプトを入力します。
- 省略記号ボタン (...) をクリックして [文字列の選択] ダイアログを表示し、プロンプトに使用する文字列を選択します。

最後のディスク以外のすべてのディスクに対して、この手順を繰り返します。リリース ウィザードは、[ディスクプロンプト] の値として使用する、文字列テーブルのエントリを自動的に作成します。このエントリに対して、セットアッププログラムがサポートする各言語に翻訳した文字列を設定してください。

表示される文字列は、以下に述べるように、プロジェクト全体にわたる多くのソースから抽出されたものです。

1. 完全なプロンプトは、ダイレクト エディターに表示されるエラー テーブルで、エラー レコード 1302 として生成されます。
2. この値は文プロジェクトの文字列エントリからのもので、日本語のデフォルト値は、「次のディスクを挿入してください: [2]」です。
3. Windows Installer は、DiskPrompt プロパティの値で *[2]* を解決します。この値はプロパティ マネージャーで *[1]* に設定されています。
4. 最後に、Windows Installer は、"ディスクのプロンプト" フィールドに入力した文字列に対して、*[1]* を評価します。

通常は、下のディスクボリュームと同じ名前を入力します。たとえば、デフォルト値が変更しなかったと想定して、*Disk8* のディスク プロンプトを入力すると、ユーザーに「次のディスクを挿入してください : Disk8」というプロンプトが表示されます。

ディスク ボリューム

ディスクを選択して、作成するディスクイメージフォルダーの名前を入力します。各ディスクに対して、この手順を繰り返します。

CD-ROM などのリムーバブル メディアのリリースイメージをビルドする場合、作成するメディアのボリューム名は、必ずここで指定したボリューム名と同じにしてください。

ボリューム名が DISK? のままであれば、リリース ウィザードは、たとえば、DISK1、DISK2 というように 1 から順にディスクに番号を付けます。

ヒント: *ボリュームのデフォルトの名前は、すべて大文字で表記することに注意してください。ボリューム名の大文字表記を原則とする（CD-ROM 等）メディアタイプにも適用できるように、大文字で表記することをお勧めします。*

[Web タイプ] パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

[Web タイプ] パネルでは、Web インストール パッケージの構成を選択を求めるプロンプトが表意されます。

パネルのオプション

単一実行可能ファイル

リリースを単一の自己展開型 Setup.exe ファイルとしてビルドする場合、このオプションを選択します。インストール パッケージには必要はものがすべて含まれており、インストール パッケージを特定の場所に限定する構成情報が含まれていないため、この Web タイプは、複数の Web または FTP サイトからダウンロードされるパッケージに理想的です。

ただし、全インストール パッケージがダウンロードされるため、ダウンロード サイズと所要時間は、以下の Web からインストールする タイプより大きくなることに注意してください。

ダウンローダー

Setup.exe と .msi データベースを組み合わせてこのリリースをビルドする場合、このオプションを選択します。このオプションを利用して、製品のデータ ファイルの格納場所を指定することができます。データ ファイルは、.msi データベースにストリームすることもできますし、キャビネット (.cab) ファイルに外部的に格納することもできます。すべての .mst ファイルと .ini ファイルは Setup.exe ファイルに埋め込まれます。

[ダウンロード] オプションを使って、エンドユーザーは Setup.exe をダウンロードして実行します。そのあと、Setup.exe が MSI データベースをダウンロードして実行します。.cab ファイルが外部的に格納されている場合、エンド ユーザーが選択したセットアップの種類と機能に基づいて、必要な .cab ファイルのみがダウンロードおよびインストールされます。これにより、ダウンロードのサイズと時間が最小化されます。

リリース ウィザードの [ダウンローダ オプション] パネルで指定した URL は、製品メンテナンスと修復モードで使用されます。

Web からインストールする

Setup.exe、MSI データベースおよび外部キャビネット (.cab) ファイルを組み合わせてこのリリースをビルドする場合、このオプションを選択します。すべての .mst ファイルと .ini ファイルは Setup.exe ファイルに埋め込まれます。

[Web からインストールする] オプションで、エンド ユーザーが Setup.exe をダウンロードして実行すると、Setup.exe によって .msi データベースがダウンロードされ実行されます。このとき、エンド ユーザーが選択したセットアップの種類と機能に基づいて、必要な CAB ファイルのみダウンロードおよびインストールされます。これにより、ダウンロード時間を最小限に抑えられます。

リリース ウィザードの [[Web からインストールする] オプション] パネルで指定した URL は、製品メンテナンスと修復モードで使用されます。

[ダウンローダーのオプション]パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

[ダウンロード オプション] パネルでは、インストール パッケージの URL を指定します。このパネルでは、製品のデータ ファイルの場所も指定することもできます。

テーブル 6 - [ダウンロード オプション] パネルの設定

設定	説明
パッケージの URL	インストール パッケージの URL を **http://www.yourcompany.com/downloads** のように入力してください。(ファイル名は必要ありません。)
外部 .cab ファイルを作成する	製品のデータ ファイルを .msi パッケージにストリームするには、このチェック ボックスをクリアします。 製品のデータ ファイルを .msi パッケージの外 (外部 .cab ファイル) に格納する場合、このチェック ボックスを選択して、適切な **.cab ファイル** オプションを選択します。
.cab ファイル	**[外部 .cab ファイルを作成する]** チェック ボックスを選択した場合、.cab ファイルの作成方法を指定します。選択可能なオプションは以下のとおりです: - **コンポーネントごとに 1 つの .cab** — 各コンポーネントに個別のキャビネット ファイルをビルドします。実行時に、エンド ユーザーが選択したセットアップの種類と機能に基づいて、必要なキャビネット ファイルのみがダウンロードされます。 - **機能ごとに 1 つの .cab** — 各機能に個別のキャビネット ファイルをビルドします。実行時に、エンド ユーザーが選択したセットアップの種類と機能に基づいて、必要なキャビネット ファイルのみがダウンロードされます。 - **サイズに基づいて複数の .cab** — 各キャビネット ファイルについて特定のサイズを入力します (単位はキロバイト)。

[Web からインストールする オプション] パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

[Web からインストールする オプション] パネルでは、インストール パッケージの URL を指定します。このパネルでは、.cab ファイルの作成方法も指定します。

テーブル 7 - [Web からインストールする オプション] パネルの設定

設定	説明
Web からインストールするファイルの URL	インストール パッケージの URL を **http://www.yourcompany.com/downloads** のように入力してください。URL は**http://** で始める必要があります。

テーブル 7 • [Web からインストールする オプション] パネルの設定 (続き.)

設定	説明
.cab ファイル	.cab ファイルの作成方法を指定します。選択可能なオプションは以下のとおりです: • **コンポーネントごとに 1 つの .cab** – 各コンポーネントに個別のキャビネット ファイルをビルドします。実行時に、エンド ユーザーが選択したセットアップの種類と機能に基づいて、必要なキャビネット ファイルのみがダウンロードされます。 • **機能ごとに 1 つの .cab** – 各機能に個別のキャビネット ファイルをビルドします。実行時に、エンド ユーザーが選択したセットアップの種類と機能に基づいて、必要なキャビネット ファイルのみがダウンロードされます。 • **サイズに基づいて複数の .cab** – 各キャビネット ファイルについて特定のサイズを入力します (単位はキロバイト)。

One-Click Install パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

One-Click Install パネルでは、エンドユーザーがプロジェクトのインストールを開始できる HTML ページとキャビネット ファイルを生成するかどうかを指定します。

パネルのオプション

One-Click Install を作成する

One-Click Install(TM) を作成する場合はこのオプションをクリックします。One-Click Install は、最初のユーザー インターフェイスが HTML ページのインストール プログラムです。エンドユーザーが Web ページにアクセスしてページ上のボタンをクリックすると、インストールファイルがターゲット システムにダウンロードされ、プログラムが起動しますユーザーのシステムにダウンロードされるファイルは、現在のリリースの "Web タイプ" プロパティの設定に依存します。

HTML ベース ファイル名

作成する HTML ファイルのベース名を、*「install」*のように入力します。指定したベース名には、拡張子 ".htm" が追加されます。

CAB ベース ファイル名

生成キャビネット (.cab) ファイルのベース名を、**install**のように入力します。

[ローカル マシン] パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

[ローカルマシン] パネルでは、インストールファイルをターゲット システムにキャッシュするかどうか、またキャッシュする場所を指定します。

パネルのオプション

ローカル コンピューター上にキャッシュをインストールする

インストール ファイルをターゲット システム上にキャッシュする場合、このボックスをチェックします。ユーザーが製品のメンテナンスを実行すると、ここで指定されたキャッシュの場所がデフォルトのインストール ソースとして使用されます。

メモ: *このオプションを使用してターゲット システムにキャッシュしたファイルは、ユーザーが製品をアンインストールしても、自動的に削除されません。*

パス

ファイルをキャッシュするときのターゲット システム上のディレクトリを指定します。C:¥Cached Files のようにハードコード化されたパスを入力することもできますが、[LocalAppDataFolder]Downloaded Installations のようにパスのドロップダウンリストでインストール先変数を含める方法をお勧めします。

メモ: *[パス] リストで含められたインストール先変数だけがダウンロード場所に使用できます。*

[リリース構成] パネル

プロジェクト: *このパネルは、InstallScript プロジェクトでは使用できません。*

[リリース構成] パネルでは、リリースで使用する圧縮タイプがあるとき、その種類を指定できます。

既存のリリースを再ビルドする場合、ウィザードは以前のビルド設定を覚えています。

テーブル 8 : [リリース構成] パネルの設定

設定	説明
すべてのファイルを圧縮	すべての製品のファイルを、単一の実行可能ファイル (Setup.exe)、または単一の Windows Installer パッケージ (.msi ファイル) に圧縮する場合、このオプションを選択します。
ファイルは圧縮せず、インストール パッケージから分離する	製品のファイルを圧縮しない場合、またそれらのファイルが、インストール パッケージを含むフォルダーのサブフォルダーに含まれる場合、このオプションを選択します。

テーブル 8 • [リリース構成] パネルの設定 (続き.)

設定	説明
カスタム	リリースを圧縮するかどうかを指定します。 • **すべてのファイルを圧縮** – すべての製品のファイルを、単一の実行可能ファイル (Setup.exe)、または単一の Windows Installer パッケージ (.msi ファイル) に圧縮する場合、このオプションを選択します。 • **ファイルは圧縮せず、インストール パッケージから分離する** – 製品のファイルを圧縮しない、かつ、それらのファイルをインストール パッケージから分離する場合、このオプションを選択します。 • **カスタム** – 1 つまたは複数の機能に関連付けられているファイルのみを .cab ファイルに圧縮する場合、このオプションを選択します。このオプションを選択すると、リリース ウィザードで [次へ] をクリックしたとき、[カスタム圧縮設定] パネルが表示されます。このパネルで、圧縮方法を構成することができます。 ***ヒント•*** *ビルド プロセスの出力は、"メディアの種類"、"圧縮" および "分割" 設定によって異なります。*

メモ• *ビルド プロセスの出力は、"メディアの種類"、"圧縮" および "分割" 設定によって異なります。*

- *[ネットワーク イメージ] メディア タイプを選択して、**[すべてのファイルを圧縮]** オプションを選択すると、Setup.exe インストール ランチャを含めるかどうかによって、すべてのデータ ファイルが Setup.exe または .msi データベースに圧縮されます。*
- *CD-ROM、DVD-ROM または [カスタム] メディア タイプで **[すべてのファイルを圧縮]** を選択すると、各ディスク イメージにはデータ ファイルを含む DataN.cab ファイルが含まれます。*
- *CD-ROM、DVD-ROM または [カスタム] メディア タイプを [カスタム] オプションの圧縮と共に選択すると、圧縮するデータ ファイルは、ComponentName.cab という名前のキャビネット ファイルに配置されます。*
- *Web メディア タイプを選択すると、ビルド出力はリリース ウィザードの [Web タイプ] パネルに表示されます。*

[カスタム圧縮の設定] パネル

プロジェクト• *このパネルは、InstallScript プロジェクトでは使用できません。*

このパネルでは、圧縮する機能を選択します。

パネルのオプション

圧縮する機能

ここでは、このリリースにある機能がすべて表示されます。機能名の隣のチェック ボックスを選択して、圧縮する機能を選択します。

コンポーネント または マージ モジュール は、2 つ以上の機能に関連する可能性があります。この場合、リリース ウィザードは、コンポーネントやモジュールが関連付けられている*最初の*機能の圧縮設定を適用します。たとえば、コンポーネントA が機能1 (ファイルは圧縮) と機能2 (ファイルは非圧縮) の両方に属している場合、圧縮設定は次のように適用されます。リリースにビルドされる最初の機能が機能1 の場合、コンポーネントA のファイルは、Windows Installer パッケージ、または Setup.exe に圧縮されます。

すべて選択

このボタンを選択すると、すべての機能が圧縮されます。

すべてをクリア

このボタンをクリックすると、すべての機能が選択解除されます。つまりインストールに圧縮される機能はなくなります。

[セットアップランチャ] パネル

プロジェクト: *[セットアップランチャー] パネルは、次のプロジェクトの種類で使用可能です。*

- *基本の MSI*
- *InstallScript MSI*

[セットアップランチャー]では、Setup.exe セットアップランチャーを作成するかどうかを指定できます。また、Windows Installer 3.1 以前の再配布可能ファイルをインストールに含めるかどうかも指定することができます。

Windows Installer 再配布可能ファイルについては、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

テーブル 9 • [セットアップランチャー] パネルの設定

設定	説明
インストール ランチャ (Setup.exe) を作成する	現在のリリースに Setup.exe セットアップランチャーを作成するかどうか指定する場合、このオプションを使用します。 Setup.exe セットアップランチャーは、次のような場合に必要です: たとえば、ターゲット システムに Windows Installer エンジンをインストールする場合、セットアップランチャーがリリースに含まれている必要があります。[セットアップ言語] パネルで、[言語] ダイアログを表示するオプションを選択した場合も、セットアップランチャーが必ず必要になります。セットアップランチャーが必要になるシナリオについては、「セットアップランチャーの作成」をご覧ください。 ネットワーク イメージまたは Web ([単一の実行可能ファイル] メディア タイプ) 用の[リリース構成] パネルで **[インストール パッケージ内に圧縮保存]** オプションを選択した場合、セットアップ パッケージ全体およびアプリケーション ファイルは、すべて Setup.exe に圧縮されます。

[InstallShield 前提条件] パネル

[InstallShield 前提条件] パネルは、プロジェクトに 1 つまたは複数の InstallShield 前提条件が含まれる場合に表示されます。

このパネルを利用して InstallShield 前提条件をどこに配置するかを指定します。使用可能なオプションは次の通りです。

ヒント: *各 InstallShield 前提条件のプロパティに指定された場所を使用する場合、[リリース] ビューの Setup.exe タブにある "InstallShield 前提条件の場所" 設定で [個々の選択に従う] を選択する必要があります。詳細については、「特定の InstallShield 前提条件の実行時の場所を指定する」を参照してください。*

テーブル 10 • [InstallShield 前提条件] パネルで利用可能なオプション

オプション	説明
Web から前提条件をダウンロードする	プロジェクトに含まれるすべての InstallShield 前提条件ファイルを、必要に応じて各前提条件の InstallShield 前提条件ファイル (.prq) で指定された URL からダウンロードします。 ***ヒント:*** *このオプションは、インストールがインターネットからダウンロードされるパッケージサイズとダウンロード時間を最小化したい場合に推奨されます。InstallShield 前提条件は、適切なバージョンが既にターゲット マシンにある場合はダウンロードされません。*

テーブル 10 - [InstallShield 前提条件] パネルで利用可能なオプション (続き.)

オプション	説明
Setup.exe から前提条件を抽出する	InstallShield 前提条件ファイルを Setup.exe に圧縮し、実行時に必要に応じて抽出されるように設定する。 ***ヒント-*** *このオプションは、インストール全体が Setup.exe に完全に含まれていなければならない場合に選択します。[**Web から前提条件をダウンロードする**] オプションを選択すると、インストールが小さくなり、ダウンロード時間も短くなります。ただし、[**Setup.exe からエンジンを抽出する**] オプションは完全に独立したインストールを提供します。*
ソース メディアから前提条件をコピーする	InstallShield 前提条件ファイルを、ソース メディアに格納する。 ***ヒント-*** *このオプションは、CD、DVD またはローカル ネットワークなどの固定メディアからインストールが非圧縮で実行される場合に使用します。*
個々の選択に従う	[再配布可能ファイル] ビューまたは [前提条件] ビュー、で各セットアップ前提条件のプロパティに指定した場所を使います。詳細については、「特定の InstallShield 前提条件の実行時の場所を指定する」を参照してください。

メモ- *InstallShield 前提条件が別の前提条件の依存関係としてプロジェクトに追加される場合、前提条件の依存関係の場所は、それを必要とする前提条件の場所設定に従います。*

[デジタル署名] パネル

[デジタル署名] では、パッケージとパッケージのファイルのデジタル署名情報を指定できます。アプリケーションにデジタル署名をすることで、アプリケーション内のコードが発表以来変更または破損していないことをエンドユーザーに対して保証します。

テーブル 11 - [デジタル署名] パネルの設定

設定	説明
証明書 URL	完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。

テーブル 11 • [デジタル署名] パネルの設定 (続き.)

設定	説明
デジタル証明書情報	リリースに署名を行うために使用するデジタル証明書を指定するには、この設定の横にある [参照] ボタンをクリックします。[証明書の選択] ダイアログ ボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。 InstallShield では、パブリック キー証明書 (.cer) ファイルを使って EV トークン パスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。.cer ファイルは通常、USB eToken プロバイダー (たとえば、SafeNet 認証クライアント) と関連付けられた EV 認証クライアント ツールからパブリック キーをエクスポートして作成されます。.cer ファイルが指定された場合、InstallShield は Extended Validation 証明書 (EV) プロパティを構成するための追加オプションを表示します。詳細については、次を参照してください。 • InstallShield における拡張検証 (EV) 証明書情報の構成 • [証明書の選択] ダイアログ ボックス .pfx ファイルを指定するか、テスト ストアから証明書を選択すると、次が表示されます: • **証明書の拇印** – この読み取り専用設定は、証明書の拇印を表示します。 • **発行元** – この読み取り専用設定は、証明書の発行元情報を表示します。 • **有効期限日** – この読み取り専用設定は、証明書の有効期限日を表示します。 証明書ファイル (EV がエクスポートした .cer ファイル) を指定すると、次の設定が表示されます: • **暗号化サービス プロバイダー** – この設定を使って、暗号化サービス プロバイダー (CSP) を指定します。 • **コンテナ名** – この設定を使って、暗号化サービス プロバイダー (CSP) に関連付けられた秘密キーのコンテナ名を指定します。 • **トークン パスワード** – この設定を使って、暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。 ***メモ:*** *証明書ファイル (EV がエクスポートした .cer ファイル) を指定した後に表示される設定を使用する際は、以下の情報に注意してください:* • *暗号化サービス プロバイダー (CSP) 名とコンテナ名の両方は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*

テーブル 11 • [デジタル署名] パネルの設定 (続き.)

設定	説明
証明書パスワード	使用する .pfx にパスワードがある場合、それを入力します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。

[デジタル署名のオプション] パネル

[デジタル署名] パネルでデジタル署名情報を入力すると、リリース ウィザードに [デジタル署名のオプション] パネルが含まれます。

[デジタル署名のオプション] パネルを利用して、ビルド時にデジタル署名をするインストール内のファイルを指定することができます。

ヒント• *[リリース] ビューの [署名] タブで、リリースのデジタル署名情報を構成することもできます。*

テーブル 12 • [デジタル署名のオプション] ダイアログ ボックスの設定

設定	プロジェクトの種類	説明
Windows Installer パッケージに署名	基本の MSI、InstallScript MSI、マージ モジュール	Windows Installer パッケージ (.msi ファイル) にデジタル署名するには、このチェック ボックスを選択します。 ***メモ•*** *Setup.exe タブの "MSI エンジン/スキーマ" 設定で、バージョン 2.0 以降が選択されている場合のみ、Windows Installer パッケージに署名することができます。*
メディアの署名	InstallScript	メディア ヘッダー ファイル (Data1.hdr) にデジタル署名するには、このチェック ボックスを選択します。

テーブル 12 • [デジタル署名のオプション] ダイアログ ボックスの設定 (続き.)

設定	プロジェクトの種類	説明
Setup.exe に署名	基本の MSI、InstallScript、InstallScript MSI	Setup.exe ファイルに署名するには、このチェック ボックスを選択します。 この設定は、次の基準を満たすリリースにのみ適用されます。 • 単一ファイル Setup.exe • 圧縮ファイル • ネットワーク イメージ メディア タイプ
パッケージ内のファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	**[サポートファイル]** ビューで構成された InstallShield サポートファイル (英語、言語非依存、および詳細ファイル) を含む、インストール プロセス中にのみ使用されるすべてのファイルにデジタル署名を行う場合、まずこのチェック ボックスを選択してから、**[次のパターンとファイルを含める]** ボックスと **[次のパターンとファイルを除外する]** ボックスを使用して、署名するファイルを指定します。 ***Windows ロゴ*** *• インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

テーブル 12 • [デジタル署名のオプション] ダイアログ ボックスの設定 (続き.)

設定	プロジェクトの種類	説明
含めるパターンとファイル	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビルド時にデジタル署名するファイルおよびファイル パターンを指定します。 次の事項に注意してください。 • ボックスに直接入力することができます。代わりに、[ファイル] ボタンをクリックすると、[ファイルの参照] ダイアログ ボックスが起動されます。このダイアログ ボックスで、現在プロジェクトにあるすべてのスタティック ファイルが一覧表示されます。選択可能なファイル パターン (*.dll など) も一覧表示されます。 • ワイルドカード文字の指定には、アスタリスク (*) を使用します。 たとえば、すべての .exe ファイルに署名する場合、*.exe のように指定します。**.exe** ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイル含め、特定のパターンに一致するすべてのファイルに署名を行う場合、特に便利です。 • 各ファイルおよび各ファイル パターンはそれぞれの行に,改行コードで区切って入力します。 • 署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、**[次のパターンとファイルを含める]** ボックスと **[次のパターンとファイルを除外する]** ボックスで *.exe と指定した場合、InstallShield は .exe ファイルに署名を行いません。

テーブル 12 • [デジタル署名のオプション] ダイアログ ボックスの設定 (続き.)

設定	プロジェクトの種類	説明
除外するパターンとファイル	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビルド時にデジタル署名を行わないファイルおよびファイル パターンを指定します。 次の事項に注意してください。 • ボックスに直接入力することができます。代わりに、[ファイル] ボタンをクリックすると、[ファイルの参照] ダイアログ ボックスが起動されます。このダイアログ ボックスで、現在プロジェクトにあるすべてのスタティック ファイルが一覧表示されます。選択可能なファイル パターン (*.dll など) も一覧表示されます。 • ワイルドカード文字の指定には、アスタリスク (*) を使用します。 たとえば、.drv ファイルはどれも署名しない場合 ***.drv** のように指定します。 ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイル含め、特定のパターンに一致するファイルに署名を行なうのを避ける場合、特に便利です。 • 各ファイルおよび各ファイル パターンはそれぞれの行に,改行コードで区切って入力します。 • 署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、**[次のパターンとファイルを含める]** ボックスと **[次のパターンとファイルを除外する]** ボックスで *.exe と指定した場合、InstallShield は .exe ファイルに署名を行いません。
既に署名されているファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	プロジェクトにデジタル署名されているファイルが既に存在する場合、InstallShield で、既存のデジタル署名を [デジタル署名] タブで指定したデジタル署名で置き換えるかどうかを指定します。これは、**[次のパターンとファイルを含める]** ボックスと **[次のパターンとファイルを除外する]** ボックスで指定された要件を満たすファイルにのみ適用されますので注意してください。 • 既にファイルに含められている既存のデジタル署名情報の代わりに、[デジタル署名] タブで指定したデジタル署名情報を使ってファイルに署名をする場合は、このチェック ボックスを選択します。 • 既に署名されているファイルについて、既存のデジタル署名情報をそのまま残す場合は、このチェック ボックスをクリアします。 このチェック ボックスはデフォルトでクリアになっています。

テーブル 12 - [デジタル署名のオプション] ダイアログ ボックスの設定 (続き.)

設定	プロジェクトの種類	説明
元の場所にあるファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	InstallShield で署名をするとき、元のファイルにも署名をするか、またはリリースに組み込まれるファイルのみ署名するかを指定します。 • 各ファイルの一時コピーに署名して、その署名された一時コピーを使ってリリースをビルドする場合、このチェック ボックスをクリアします。このチェック ボックスをクリアすると、元のファイルは変更も署名もされませんので注意してください。 • InstallShield で元のファイルに署名する場合、このチェック ボックスを選択します。 このチェック ボックスはデフォルトでクリアになっています。 *ヒント- 基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで、このチェック ボックスを選択すると、もともと署名されていなかったファイルを含むリリースの圧縮バージョンと非圧縮バージョンの両方を更新する単一のパッチを作成する場合に便利です。*

[パスワードと著作権] パネル

インストールのパスワード保護を有効にするための [パスワードと著作権] パネルが表示されます。エンド ユーザーが Setup.exe を右クリックしてから [プロパティ] を選択したときに表示される [プロパティ] ダイアログ ボックスにある [バージョン] タブに、デフォルトの情報またはカスタム情報のどちらを表示するかを指定することもできます。

テーブル 13 - [パスワードと著作権] パネルの設定

設定	説明
Setup.exe をパスワードで保護	エンド ユーザーが Setup.exe を実行するときにパスワードの入力を要求する場合、このチェック ボックスを選択してから、[パスワード] ボックスにパスワードを入力します。
Password	インストールの実行にエンドユーザーが入力しなければならないパスワードを入力します。

テーブル 13 • [パスワードと著作権] パネルの設定

設定	説明
セットアップの初期化中にパスワード ダイアログ ボックスを表示する	 ***プロジェクト•*** *この設定は、InstallScript プロジェクトで提供されています。* **OnCheckMediaPassword** イベント ハンドラー関数のデフォルトのコードにあるパスワードチェック コードを起動するには、このチェック ボックスを選択します。 このチェック ボックスをクリアした場合、スクリプトで独自のコードを入力して、エンド ユーザーにパスワードの入力を求め、**FeatureValidate** を呼び出してパスワードを確認する必要があります。
カスタム バージョンのプロパティを使用する	 ***プロジェクト•*** *この設定は、次のプロジェクト タイプで使用できます:* • *基本の MSI* • *InstallScript MSI* デフォルトの情報をオーバーライドして、自社の著作権情報およびその他の情報を表示するには、このチェック ボックスを選択します。 この情報は、エンド ユーザーが Setup.exe を右クリックしてから [プロパティ] を選択したときに、[プロパティ] ダイアログ ボックスに表示されます。
著作権情報	 ***プロジェクト•*** *この設定は、次のプロジェクト タイプで使用できます:* • *基本の MSI* • *InstallScript MSI* エンド ユーザーが Setup.exe を右クリックしてから [プロパティ] を選択したときに表示される [プロパティ] ダイアログ ボックスに表示する著作権情報を入力します。

[詳細設定] パネル

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

[詳細設定] パネルで、リリース場所とファイル名の形式が設定できます。インストール プロジェクトでは、リリースと共に Autorun およびパッケージ定義ファイルを作成するかどうかを指定することもできます。パッチ作成を最適化するために、オプションで以前のリリースビルドを指定することもできます。

既存のリリースを再ビルドする場合、ウィザードを使用して、最後に指定したビルド設定を呼び出します。

パネルのオプション

リリースの設定

テーブル 14 • [詳細設定] パネルの [リリースの設定] 領域

設定	説明
場所	リリースを保存したいディレクトリを入力します。[参照] ボタンをクリックして、このディレクトリに移動します。または、リリースの場所のパス変数 を入力します。このフィールドでパス変数を使用するには、パス変数が示すディレクトリに移動するか、使用するパス変数の名前を入力します。この名前は、<MyVariable> のように、山かっこで括ります。
長いファイル名を使用する	リリースを保管するメディアが長いファイル名をサポートする場合、このチェック ボックスを選択します。長いファイル名をサポートしない場合、またはメディアがサポートするファイル名の形式が不確かな場合、このチェック ボックスをクリアします。
サイズを最適化する	InstallShield で、このリリースのキャビネット ファイルをビルドするときに、LZX 圧縮タイプを使用する場合、このチェック ボックスを選択します。MSZIP 圧縮を使用するには、このチェック ボックスをクリアします。 [リリース] ビューにあるリリースの "Cab 最適化の種類" 設定を使って、.cab ファイルの圧縮に LZX または MSZIP を選択することもできます。さらに、この設定を使って、.cab ファイルの圧縮を行わないことを選択することもできます。 ***重要:*** *圧縮を行うと、一般的に圧縮ファイルのサイズが減少します。ただし、ビルドプロセスの完了までの時間が長くなることがあります。圧縮されるファイルのサイズとその数によって、LZX 圧縮とビルドに数時間かかる場合もあります。したがって、このチェック ボックスを選択した場合、ビルドが完了するまでの所要時間をできるだけ短縮できるよう、ビルド マシンに最新のハードウェアを搭載することが推奨されます。* この設定は、リリースがファイルの一部または全部を .cab ファイルに圧縮するように構成されている場合にのみ使用されます。
パス変数のテスト値を使用する	パス変数にテスト値を使用した場合、ここで [はい] を選択すると、この時点で実際の値が設定されます。

テーブル 14 • [詳細設定] パネルの [リリースの設定] 領域 (続き.)

設定	説明
Autorun.inf を作成する	CD-ROM でインストールを配布し AutoPlay 機能の Windows ロゴ 要件をサポートする場合は、このオプションを選択します。インストールを自動実行するための指示が含まれた Autorun.inf というテキスト ファイルがディスク イメージ フォルダーのルートに作成されます。 このファイルを編集して AutoPlay オプションを追加したり、コマンドライン パラメーターを MsiExec.exe や Setup.exe に渡したりできます。 AutoPlay は、ルートドライブでのみ使用可能なため、開発システムのディスクイメージフォルダーでは機能しません。さらに、ユーザーは、システム上で AutoPlay をオフにできます。
パッケージ定義ファイルを作成する	エンド ユーザーが SMS ジョブとしてインストールを実行できるパッケージ定義ファイル (.pdf) を作成するには、このチェック ボックスを選択します。このチェック ボックスを選択すると、バージョン 2.0 の .pdf が作成されます。
UTF-8 データベースのビルド	Windows Installer データベース、およびすべてのインスタンスまたは言語トランスフォームを UTF-8 エンコードを使ってビルドする場合、このチェック ボックスを選択します。 UTF-8 エンコードは、すべての言語の文字を同時にサポートするため、エンド ユーザーに表示するテキストおよびファイル名とレジストリ キーの両方で、たとえば日本語とドイツ語、またはロシア語とポーランド語のように自由に言語を組み合わせて使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。ただし、一部の状況下では、ユーザー インターフェイスに問題が発生します。たとえば、エンド ユーザーが /qb コマンドライン オプションを指定するか、または [プログラムの追加と削除] から製品をアンインストールすると、Windows Installer がに非常に小さいフォントを使って UTF-8 データベースに含まれるユーザー インターフェイス テキストを表示します。 このチェック ボックスをクリアすると、リリースをビルドする際に ANSI データベースが作成されます。このオプションを使うと、異なるコードページを持つ言語からの文字を同時に使用することができません。 このチェック ボックスはデフォルトでクリアになっています。

[パッチの最適化] (オプション) 領域

できる限り小さいパッチを作成するには、File テーブルのファイルキーが以前と新しい .msi データベースで同じでなければなりません。パッチ作成プロセスは File テーブル キーを使って、2 つのファイルが同じファイルかどうかを判断します。(実際のファイル名は使用することができません。これは、パッケージが異なる条件下でインストールされた同じ名前のファイルを複数含んでいる可能性があるからです。)以前のパッケージがここで指定されている場合、同一ファイルには、同一の File テーブル キーが使用されます。

詳細については、「アップグレード関する考慮事項」を参照してください。

以前の Windows Installer パッケージ

msi ファイルの以前のリリースバージョンを参照する

[リリース設定の概要] パネル

設定を変更する場合は、[戻る] ボタンをクリックして変更するパネルに戻ります。すると、[リリース設定概要] パネルに戻り、ビルドを開始できます。

パネルのオプション

リリースをビルドする

[終了] をクリックしたときにリリースをビルドするにはこのオプションを選択します。ビルドの進行状況情報が [出力] ウィンドウに表示されます。

メモ: *リリース ウィザードを Microsoft Visual Studio から起動した場合、[リリースのビルド] オプションは [概要] パネルには表示されません。*

[一般オプション] パネル

プロジェクト: *[一般オプション] パネルは、次のプロジェクトの種類で使用できます。*

- *InstallScript*
- *InstallScript オブジェクト*

一般オプション パネルでは次の処理が可能です。

- インストールを配布するための自己展開型実行可能ファイルを作成する。
- コマンドライン オプションを Setup.exe へ渡す。
- プリプロセッサ変数定義をコンパイラへ渡す。
- コンパイルしたスクリプト ファイル (.inx ファイル) をキャビネット ファイルに配置するかどうかを選択する。

プロジェクト: *InstallScript オブジェクトプロジェクトでは、コンパイラ プリプロセッサ定義 リストおよび 詳細 ボタンが利用可能になっています。このパネルの他のオプションは使用できません。*

単一実行可能ファイルを作成する

このチェック ボックスを選択して、自己展開型実行ファイルを作成します。ファイル名 と アイコン リストは、このチェック ボックスが選択されている場合のみ有効になります。

ファイル名

自動展開実行可能ファイルのファイル名を入力、または パスワードを定義する パス変数 を選択します。

アイコン

オプションとして、ビルド時に取得する実行可能ファイルのアイコンが含まれるファイルの完全修飾パスを指定することができます。ファイルの指定がない場合、デフォルトのアイコンが使用されます。ファイルを指定するには、明示的パスをタイプし、パス変数を選択してそれに追加するか、参照ボタンをクリックして [アイコンの変更] ダイアログ ボックスを開いて [参照] ボタンをクリックするとファイルを選択できます。デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、[アイコンの変更] ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。

コンパイル済みのスクリプトファイル (.inx) を圧縮してメディアに入れる

コンパイル済みスクリプト ファイル (.inx ファイル) をキャビネット ファイルに配置する場合、このチェック ボックスを選択してください。ファイルを Disk1 ディスクイメージ フォルダーに非圧縮状態で配置する場合、このチェック ボックスをクリアにします。

セットアップ コマンドライン

インストールを起動したときに Setup.exe に渡すコマンドライン パラメーターをオプションで指定することができます。コマンドラインを入力するかパス変数を選択します。

コンパイラのプリプロセッサ定義

プリプロセッサ変数定義をオプションで指定します。定義を入力するか、ユーザー作成のビルド変数を選択します。ここに指定されたプリプロセッサ変数定義は現在のリリースにのみ適用されます。他のリリースのスクリプトをコンパイルしているときには使用されません。等号およびコンマの前後に空白を置かないようにして、次の形式を使用します。

```
MYVARIABLE1=123,MYVARIABLE2
```

このような変数は、スクリプトのフローを制御する #if と #ifdef ステートメントにより、スクリプト内でテストできます。InstallScript オブジェクトを作成する際、IFX_OBJECTS と入力します。

このボックスに、プリプロセッサ変数の名前を入力すると、その変数が定義されます。たとえば、このボックスに MYVARIABLE と入力すると、次の #ifdef ループ中のスクリプトコマンドが実行されます。

```
#ifdef MYVARIABLE
  //コマンド
#endif
```

この編集ボックスでプリプロセッサ変数定義を追加または変更した後、その追加または変更を有効にするためにインストール プロジェクトをコンパイルする必要があります。詳しくは、「スクリプトのコンパイル」をご覧ください。

その他のディスクファイル

このボタンは、[サポート ファイル/ビルボード] ビューの [アドバンス ファイル] の下にある [その他] フォルダーにファイルがあり、複数のディスクイメージを作成するメディア形式を選択した場合(ネットワークイメージを除くすべての形式に当てはまります) にのみ有効になります。

このボタンをクリックすると、[一般オプション - 他のディスクファイル] ダイアログ ボックス が開きます。このダイアログ ボックスでは、Other フォルダーにある各ファイルに対して、ディスクイメージフォルダー内の場所を指定します。

アドバンスト UI

このボタンをクリックすると、一般オプション – 詳細 ダイアログ ボックス が開きます。このダイアログ ボックスを使うと、ビルドされたファイルのカスタムの場所や、ディスクイメージフォルダーに予約されているスペース、およびインストールで MD5 の確認を行うかどうかを指定できます。

[機能] パネル

プロジェクト: *このパネルは次のプロジェクトの種類では表示されません。*

- *基本の MSI*
- *InstallScript MSI*

機能パネルを使うと、ビルドしたリリースに含める機能を指定できます。

パネルのオプション

"ビルドに含める" プロパティの使用

このオプションを選択すると、"ビルドに含める" 設定が [はい] に設定されている各機能がビルドされたメディアに含められ、"ビルドに含める" 設定が [いいえ] に設定されている各機能は含められません。

機能の指定

各機能について、その機能をビルドしたリリースに含めるかどうか指定できます。機能を含めるかどうかを切り替えるには、機能の横にあるチェック ボックスをクリックします。チェック ボックスをチェックすると、機能がビルドしたリリースにインクルードされます。チェックされていないと、機能はインクルードされません。

InstallScript オブジェクトのインクルード状態を切り替えることはできません。オブジェクトの選択状態は、親機能のインクルード状態と常に同じになります。

チェック ボックスが使用できない場合は、その機能が、含まれている機能にとって必須で、除外できないことを意味します。

機能を含めるように切り替えると、親機能、子機能、および必須機能すべてが含まれ、必須機能のすべての子機能も含まれます。機能を切り替えて除外すると、この機能の子機能も除外されます。この機能の親機能に含まれている機能がなくなると、親機能も除外されます。親機能を含めるには、少なくとも 1 つ以上の子機能が含まれている必要があります。必須機能のうち、可視の子機能の 1 つを除いてすべての機能を除外すると、必須機能のうち残りの可視の子機能は自動的に含まれて選択できないようになり、最低 1 つの子機能が含まれたままになります。

可視機能 ([可視] プロパティが [はい] に設定されている機能) は、色つきのアイコン ( ) で表示されます。不可視の機能は、色なしアイコン ( ) で表示されます。

[メディア レイアウト] パネル

プロジェクト: *このパネルは、Windows Installer ベースまたは InstallScript MSI プロジェクトでは表示されません。*

このパネルでは、各機能またはすべての機能に対して、機能のファイルをキャビネットフォルダーに保存するか、ディスクイメージ内に圧縮せずに配置するかを指定します。

パネルのオプション

キャビネット ファイル

機能のすべてのファイルをキャビネット ファイルに保存します。

メモ: *コンポーネントの [圧縮] プロパティで、コンポーネントのファイルをキャビネット ファイルに保存する際に、圧縮するか圧縮しないかを選択できます。*

CD-ROM フォルダー

機能のすべてのファイルを圧縮せずにディスクイメージに配置します。機能に関連したファイルは、機能の [CD-ROM フォルダー] プロパティで指定したフォルダーのディスクイメージに配置されます。フォルダーが指定されていない場合、機能のファイルはディスクイメージのルートに配置されます。

カスタム

このオプションが選択されていると、パネルの [次へ] ボタンをクリックすると [カスタムメディアレイアウト] パネルが表示され、ここで機能のファイルをキャビネット ファイルに保存するか、圧縮せずにディスクイメージに配置するか、個別の機能を指定できます。

[カスタム メディア レイアウト] パネル

プロジェクト: *このパネルは次のプロジェクトの種類では表示されません。*

- *基本の MSI*
- *InstallScript MSI*

このパネルでは、各機能またはすべての機能に対して、機能のファイルをキャビネットフォルダーに保存するか、ディスクイメージ内に圧縮せずに配置するかを指定します。

パネルのオプション

キャビネット内の機能

各コンポーネントに対して、コンポーネントのファイルをキャビネットフォルダーに保存するか、ディスクイメージ内に圧縮せずに配置するかを指定します。コンポーネントの保存モードを切り替えるには、コンポーネントの横にあるチェック ボックスをクリックします。チェックされたボックスは、コンポーネントのファイルがキャビネット ファイルに保存されていることを示します。チェックされていないボックスは、フォルダーコンポーネントのファイルが、コンポーネントの [CD-ROM フォルダー] プロパティで指定されたフォルダーのディスクイメージに配置されていることを示します。フォルダーが指定されていない場合、コンポーネントのファイルはディスクイメージのルートに配置されます。

すべて選択

「**キャビネットの機能**」リストボックスのすべての機能を選択します。

すべてをクリア

「**キャビネットの機能**」ボックスのすべての機能を選択解除します。

[ユーザー インターフェイス] パネル

プロジェクト: *このパネルは次のプロジェクトの種類では表示されません。*

- *基本の MSI*
- *InstallScript MSI*

このパネルではセットアップのエンドユーザーダイアログの外観を指定できます。

パネルのオプション

プロジェクトのデフォルトを使用する

このオプションを選択すると、エンドユーザー ダイアログ ボックスは[ダイアログ] ビューの [スキン] フォルダーで選択されたスキンと一緒に、[ダイアログプレビュー] グラフィックに表示されるとおりに表示されます。

スキンを使用しない

このオプションが選択されていると、エンドユーザー ダイアログ ボックスは、[ダイアログプレビュー] グラフィックに表示されるとおりに標準の Windows スタイルで表示されます。

以下に指定したスキンを使用

このオプションを選択すると、エンドユーザー ダイアログ ボックスはリストボックスで選択したスキンと一緒に、[ダイアログプレビュー] グラフィックに表示されるとおりに表示されます。BlueTC スキンは GIF ファイルを使用するため、16 ビット以下のカラー設定では、ビットマップを使用する(そのため大きい)ブルーのスキンはよく見えません。

ダイアログ プレビュー

オプションボタンとリストボックスで選択した結果、エンドユーザーのダイアログ ボックスがどのように見えるか、サンプルを表示します。

メモ: *次の関数によって表示されるダイアログ ボックスは、スキンを使って表示することはできません。スキンの指定に関わらず、同じように表示されます。*

- *AskYesNo*
- *EnterDisk*
- *MessageBox*
- *MessageBoxEx*
- *RebootDialog*
- *SdComponentDialog ― [利用できるディスク領域] ダイアログ ボックス*
- *SdComponentDialog2 ― [サブコンポーネントの選択] ダイアログ ボックス*
- *SdComponentDialogAdv ― [利用できるディスク領域] ダイアログ ボックス*

- *SdConfirmNewDir*
- *SdConfirmRegistration*
- *SdExceptions*
- *SdShowMsg*
- *SelectDir*
- *SelectDirEx*
- *SprintfBox*

小さい初期化ダイアログを表示

チェックされていると、セットアップ初期化ダイアログ ボックスは InstallShield Professional バージョン 6.31 以前で作成されたセットアップによって表示される小さいボックスになります (ただし [セキュリティ]、[セットアップを保存して実行]、[セットアップの実行]、[セットアップ言語の選択] または [正規の製品が検出されました] ダイアログ ボックスが表示される場合を除く。その場合、セットアップ初期化ダイアログ ボックスは大きくなり、チェック ボックスの状態に関わらず残りのエンドユーザーダイアログ ボックスと統一されます。)
チェックされていない場合、セットアップ初期化ダイアログ ボックスは大きくなり、残りのエンドユーザーダイアログ ボックスと統一されます。

このチェック ボックスが選択解除されていると、インストール初期化ダイアログ ボックス(および [セットアップ言語の選択] ダイアログ ボックスがあればこれも該当)は、スタートアップ グラフィックが閉じるまで表示されません。スタートアップ グラフィックの表示時間を指定するには、Setup.ini の [Startup] セクションの SplashTime 値を設定します。

[インターネット オプション] パネル

プロジェクト: *このパネルは、InstallScript プロジェクトで使用することができます。*

[インターネット オプション] パネルでは、さまざまなインターネット関連のオプションを指定できます。

メモ: *すべてのリリースは、メディアの種類を問わずインターネット上で実行できます。*

テーブル 15 : [インターネット オプション] パネルの設定

設定	説明
One-Click Install の生成	エンド ユーザーがインターネットからダウンロードしてインストールすることできる ClickOnce Install インストールを作成する場合、このチェック ボックスを選択します。このチェック ボックスを選択すると、適切なコマンドラインを使って Setup.exe ファイルのダウンロードと起動を行う外部セットアップ プレーヤー (Setup.exe ファイル) がインストールに含まれます。 このチェック ボックスを選択すると、このパネルの他の設定が有効にされます。 詳細については、「InstallScript プロジェクトの One-Click Install インストール」を参照してください。

テーブル 15 • [インターネット オプション] パネルの設定 (続き.)

設定	説明
セットアップのデフォルト Web ページを作成	InstallShield で、デフォルトの Web ページ (.htm ファイル) を作成し、Disk1 フォルダーに配置する場合、このチェック ボックスを選択します。
起動する URL の入力	次のいずれかを実行して、[ビルド] メニューで [Web から実行] コマンドをクリックしたときに起動される URL を示します。 • ビルド時に InstallShield によって作成され、Disk1 フィールドに配置される Setup.htm ファイルを起動する場合、このボックスを空白のままにします。 • 別の Web ページを起動する場合、このボックスに URL を入力するか、または参照ボタンをクリックして、Web ファイルを選択します。 ***ヒント:*** *URL を入力する場合、ページ名を含めないでください。また、末尾にスラッシュ (/) を使用することも避けてください。たとえば、Setup.htm ファイルの完全 URL が* **http://www.mypages.com/setup/Setup.htm** *の場合、ページ URL を* **http://www.mypages.com/setup** *のように入力します。.* *[ビルド] メニューで [Web から実行] コマンドをクリックしたとき、適切な URL (上記例の http://www.mypages.com/setup/Setup.htm) が起動されます。*

[アップデート] パネル

プロジェクト: *このパネルは次のプロジェクトの種類では表示されません。*

- *基本の MSI*
- *InstallScript MSI*

このパネルではリリースの形式と、現在のリリースをアップデートとして実行できる既存のリリースを指定できます。

ヒント: *現在、バージョンのない製品がインストールされているシステムにセットアップを作成する場合、[完全] オプション ボタンを選択のうえ、[非バージョン固有] オプション ボタンを選択してください。*

パネルのオプション

完全

現在のリリースが完全リリースであることを指定します。

バージョンの指定なし

[完全] オプション ボタンが選択されている場合のみ有効です。InstallShield Professional バージョン 6.0 以降で作成されたセットアップがインストールされていれば、現在のリリースでアプリケーションの既存のバージョンをアップデートできるよう、あるいはバージョンのない製品がインストールされているシステムにアプリケーションをインストールできるように指定します。

バージョンを指定

[完全] オプション ボタンが選択されている場合のみ有効です。現在のリリースで、コンボボックスで指定されたアプリケーションのバージョンだけをアップデートできるよう指定します。

コンボ ボックス

[バージョン固有] オプションが選択されている場合のみ有効です。アップデートを適用できる製品のバージョンを指定します。バージョン番号をセミコロンで区切って入力 (例 1.2.3;1.2.4) するか、リストボックスからバージョン番号のリストを選択します。このフィールドを空白にすると、製品のすべての以前のバージョンにアップデートが適用されます。

差分

現在のリリースが差分リリースであることを指定します。

リストボックス

[差分] オプションボタンが選択されている場合のみ有効です。新しい差分リリースを作成しているときに、現在のプロジェクトを比較する既存のリリースを表示します。同一ファイル (同一の日時、サイズ、属性を持つ) が指定の各リリースに存在する場合、現在のプロジェクトのファイルが差分リリースから除外されます。そうでない場合、ファイルは差分リリースに含められます。(差分プロパティが「いいえ」に設定されているコンポーネントのファイルは常に差分リリースに含まれます。)これらのリリースは、[インポート] ボタンと [追加] ボタンをクリックして指定します。

また、各リリースのバージョン情報を表示します。

メモ: *セットアップで指定のリリースのバージョン情報を決定できない場合、製品のバージョンをアップデートできません。リリースのバージョン情報が表示されない場合、リリースを選択して [変更] ボタンをクリックしてから [**以下にバージョン情報を指定する**] オプションを選択して、バージョン番号をタイプまたは選択します。*

追加

[差分] オプションが選択されている場合のみ有効です。現在のプロジェクトにないリリースを指定できる 既存のメディア ダイアログ ボックスを開きます。

サービス名

[差分] オプションが選択されている場合のみ有効です。現在のプロジェクトにないリリースを指定できる メディアファイル プロパティ ダイアログ ボックスを開きます。

変更

[差分] オプションが選択されていて単一リリースがリストボックスで選択されている場合に有効です。[メディアファイルプロパティ] ダイアログ ボックスが開くので、ここでリリース選択とリリースに関したバージョン情報を選択できます。

削除

[差分] オプションが選択されていて1 つ以上のリリースがリストボックスで選択されている場合に有効です。リストから選択されたリリースを削除します。

オブジェクト

[差分] オプションが選択されている場合のみ有効です。[オブジェクトの差分] ダイアログ ボックスが開くので、ここで差分リリースの InstallScript オブジェクトに含める条件を選択できます。

[オブジェクト差分] ダイアログ ボックス

このダイアログ ボックスは、リリース ウィザードの [アップデート] パネルで [オブジェクト] ボタンをクリックすると開きます。このダイアログ ボックスでは、差分メディアに InstallScript オブジェクトを含むための条件を指定することができます。

ダイアログ オプション

すべてを含める

差分ビルドが完全 (非差分) ビルドに含まれるすべてのオブジェクトを含むことを指定します。(この場合、たとえば [ビルドに含める] プロパティが [いいえ] に設定されている機能、または [リリース ウィザード] の [機能] パネルで選択解除された機能と関連付けられたオブジェクトを除きます。)

変更された場合に含める

完全ビルドに含まれていて、指定した比較メディアのいづれにも含まれていない、または以前のバージョン番号が含まれているオブジェクトのみを差分ビルドに含むことを指定します。

メモ: *現在のプロジェクトおよびすべての指定された比較メディアが InstallShield Object Installer オブジェクトを含む場合、メディアビルダーは現在のプロジェクトに含まれる InstallShield Object Installer がパッケージしたファイルと、指定されたメディアのパッケージ ファイルとを比較します。メディア ビルダーは、指定したメディアの何れにも含まれていない、または古い日時、異なるサイズ、あるいは属性を持ファイルのみを InstallShield Object Installer に含みます。*

すべてを除外する

差分ビルドにオブジェクトが含まれないことを指定します。

[ポストビルドのオプション] パネル

プロジェクト: *[ポストビルドのオプション] パネルは、次のプロジェクトの種類では表示されません。*

- *基本の MSI*
- *InstallScript MSI*

[ポストビルドのオプション] パネルで、リリースのビルドが完了した後に、ディスク イメージ フォルダーをフォルダーまたは FTP サイトにコピーしたり、またはバッチ ファイルを実行したりできます。

テーブル 16 • [ポストビルドのオプション] パネルの設定

設定	説明
リリース ファイルを以下に指定した FTP サイトへアップロード	リリースを自動的に FTP サイトに配布するには、このチェック ボックスを選択して、FTP ロケーションを入力します。また、必要に応じて、ユーザー名とパスワードも入力します。 • **FTP ロケーション** – 場所の FTP URL を指定します。 FTP デフォルト フォルダー以外のパスにリリースを配布する場合は、ダブル スラッシュ (//) を使います。たとえば、リリースを、FTP の URL が **ftp://ftp.mydomain.com** である **myproduct** という名前のルートレベル フォルダーへ配布する場合、FTP の場所に **ftp://ftp.mydomain.com//myproduct** と入力します。 • **ユーザー名** – FTP ロケーションへのアップロードにユーザー名が必要な場合、ユーザー名を入力します。 • **パスワード** – FTP ロケーションへのアップロードにパスワードが必要な場合、パスワードを入力します。
ビルドしたメディア ファイルを以下に指定したフォルダーへコピー	リリースをフォルダーに自動的に配布できるようにするには、このチェック ボックスを選択して、その場所の FTP URL を指定します。コピーされたフォルダーと同じ名前の既存のフォルダーは上書きされますが、フォルダーは削除されません。このボックスでパスを入力するか、場所を参照する参照ボタンをクリックして、フォルダーのパスを指定します。 選択されたリリースのメディア形式がネットワーク イメージの場合 (ディスク イメージ フォルダーは 1 つのみ作成されます)、フォルダー自体ではなくディスク イメージ フォルダーの内容がコピーされます。自己展開型実行可能ファイルを作成することを選択した場合、ディスク イメージ フォルダーではなく実行可能ファイルがコピーされます。
メディア ファイルをビルド後、以下に指定したバッチまたは実行ファイルを実行	リリースがビルドされたあと、バッチ ファイル (.bat) または実行可能ファイル (.exe) を起動するには、このチェック ボックスを選択して、ボックスでパスを指定します。パスを直接入力することもできますし、[参照] ボタンをクリックして、ファイルを参照することもできます。また、一覧からパス変数を選択して、残りのパスを入力することもできます。 このチェック ボックスを選択すると、プロジェクトのビルド変数と同じ名前 (山かっこを含む) と値が環境変数に設定されます。また InstallShield は、環境変数 <ISMEDIADIR> も設定します。この変数の値は、その中にリリースの Disk Images フォルダー、Log Files フォルダー、および Report Files フォルダーが作成されるフォルダーへのパスです。バッチ ファイルの環境変数の値は、変数名をパーセント記号 (%) で囲んで参照することができます。例、 set PATH = %<ISPROJECTDIR>%;%PATH% ファイルが終了すると、設定された環境変数は削除されます。

セットアップ ベスト プラクティス ウィザード

このウィザードは、コンポーネントにファイルを追加する際にセットアップ ベスト プラクティス反した場合に常に表示されます。このウィザードには、どのベストプロクティスに反したかが表示され、そのアクションを修正するオプションが提示されます。

ウィザードは、[オプション] ダイアログで、ベスト プラクティス モニターリングをアクティブにしたときのみ表示されます。

[ようこそ] パネル

コンポーネントにファイルを追加したときに、次のセットアップ ベスト プラクティスのうちの 1 つに違反したという警告がウィザードから発せられました。

テーブル 17 • セットアップベストプラクティスの違反

ベスト プラクティス	説明
コンポーネントに複数の EXE、DLL、OCX、CHM、または HLP ファイルを含めない	各コンポーネントには "ポータブル実行可能" ファイル (EXE、DLL、または OCX) あるいはヘルプ ファイル (HLP または CHM) を 1 つだけ含めてください。Windows Installer のコンポーネントは、コード GUID など、すべての詳細設定とコンポーネントのプロパティが理想的にはただ 1 つの高移植性実行ファイルまたはヘルプファイルを参照するよう作られています。1 つのコンポーネントに入らなかった EXE、DLL、OCX、CHM、または HLP は、それぞれ新しいコンポーネントに含めるようにしてください。
ファイルを複数のコンポーネントに関連付けることはできない	再利用可能なコンポーネントを出荷し、柔軟性を提供するためには、このベストプラクティスは非常に重要です。Microsoft は、製品間、製品バージョン間、および会社間でも、複数のコンポーネントにファイルを含めないように推奨しています。
マージ モジュールの使用	マージ モジュールには、個別の機能をインストールするために必要な、すべてのファイル、レジストリ エントリ、および論理が含まれています。マージ モジュールが可能なファイルを配布しないでください。また、マージ モジュールを使用すると、1 つのファイルを複数のコンポーネントに関連付けないというベストプラクティスと、いかなるコア コンポーネントも外に出さないという Windows ロゴ ガイドラインの 2 つの関連要件に準拠することができます。 ベストプラクティスをモニターするオプションを有効にすると、InstallShield が提供するマージ モジュールの一部となっているファイルをコンポーネントに追加したとき、常にセットアップベストプラクティスウィザードが警告を発します。

パネルのオプション

セットアップベストプラクティス違反を警告しない

セットアップベストプラクティスウィザードで、ベストプラクティス違反を通知しない場合、このオプションを選択します。このアクションは、後で [ツール] | [オプション] パネルで元に戻すことができます。

[準拠] パネル

このパネルには、[ようこそ] パネルでセットアップ ベスト プラクティスに違反するファイルが表示されます。違反ファイルの横に警告のアイコンが付き、そのベストプラクティスが表示されます。

パネルのオプション

ファイル

コンポーネントに追加したすべてのファイルが一覧表示され、ベストプラクティスに違反するファイルが示されます。[完了] をクリックしてウィザードの推奨事項を無視 (または [キャンセル] をクリックしてウィザードを終了) することも、[削除] ボタンをクリックしてベストプラクティスとの競合を修正することもできます。

削除

ファイルを選択して [削除] ボタンをクリックすると、ファイルがこのコンポーネントに追加されるのを回避できます。

スタティック スキャン ウィザード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

基本の MSI および InstallScript MSI プロジェクトで、コンポーネントの "ビルド時に .NET をスキャン" 設定で選択された値は、スタティック スキャン ウィザードがそのコンポーネントに含まれるファイルをスキャンする方法に影響します。詳細については、「スタティック スキャン」を参照してください。

スタティック スキャン ウィザードを利用して、プロジェクト含まれているファイルをスキャンして必要な依存関係があるかどうかをチェックすることができます。このウィザードはプロジェクトにあるすべての .exe、.dll、.ocx、.sys、.com、.drv、.scr、および .cpl ファイルをスキャンし、検出した依存関係をインストールに追加します。

プロジェクトに追加された新規ファイルは、ファイルが依存している同じ機能に追加されるので、インストールが必要なときに確実に実行されることを確認します。

タスク

スタティック スキャン ウィザードを起動するには、次の操作を実行します。

1. **[追加ツール]** の下のビュー リストで、**[依存関係スキャナー]** をクリックします。
2. **[スタティック スキャンの実行]**ボタンをクリックします。

スタティック スキャン ウィザードには、次のような関連パネルがあります:

- ようこそ
- ファイルのフィルター
- スキャンの進行状況
- ファイルの選択

- スキャン結果
- スタティック スキャン ウィザードの終了

[ようこそ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザードを利用して、プロジェクト含まれているファイルをスキャンして必要な依存関係があるかどうかをチェックすることができます。このウィザードはプロジェクトにあるすべての .exe、.dll、.ocx、.sys、.com、.drv、.scr、および .cpl ファイルをスキャンし、検出した依存関係をインストールに追加します。

[次へ] ボタンをクリックすると、プロジェクトに含まれるファイルの依存関係を調べるスキャンを開始します。

[ファイルのフィルター] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザードは、インストールに追加する必要のないファイルを依存関係としてリストする場合があります。例えば、ターゲット マシン上に既存する一般的なシステムファイルは通常、再インストールする必要がありません。スキャナーを実行した時にこれらのファイルがインストールに追加されることを回避するには、[ファイルのフィルター] パネルで **[ファイルのフィルター]** を選択します。

スキャンから除外するファイル リストをカスタマイズする方法については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

[スキャンの進行状況] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザードがプロジェクトに含まれるすべての .exe、.dll、.ocx、.sys、.com、.drv、.scr、および .cpl ファイルをスキャンして依存関係を調べている最中に [スキャンの進行状況] パネルが表示されます。スキャン結果を確認しなければ、ファイルは追加されません。

[ファイル選択] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[ファイルの選択] パネルには、プロジェクトに追加する必要がある可能性の高いファイルおよびマージ モジュールのリストが表示されます。このパネルを使って、インストールに含めるファイルおよびマージ モジュールを選択できます。詳細については、「依存関係スキャナー結果の確認」を参照してください。

テーブル 18 • [ファイル選択] パネルの設定

設定	説明
File	該当するチェック ボックスを選択して、インストールに追加するファイルを指定します。各種アイコンが示す意味は次のとおりです。 • : このアイコンは、ファイルがシステムまたはドライバー ファイルであることを示します。これらのファイルは通常、インストールの一部としては再配布されないため、ターゲット システムが動作不能になる可能性があります。これらのファイルが必要かどうか、含める前に確認してください。 • : このアイコンは、システムファイルではないことを示します。ファイルをプロジェクトに追加するには、対応するチェック ボックスを選択します。 • : このアイコンはマージ モジュールを示します。マージ モジュールをプロジェクトに追加するには、対応するチェック ボックスを選択します。 • : これは、プロジェクトの .NET アセンブリのアイコンです。 • : このアイコンは、スタティック スキャン ウィザードによって検出されたが、既にプロジェクトに含まれているファイルを示します。
すべて選択解除	すべてのチェック ボックスをクリアする場合は、このボタンをクリックします。その後、プロジェクトに追加するファイルまたはモジュールに対応する各チェック ボックスを手作業で選択できます。
すべて選択	すべてのチェック ボックスを選択する場合は、このボタンをクリックします。その後、プロジェクトに追加するファイルまたはモジュールに対応する各チェック ボックスを手作業でクリアできます。

[スキャン結果] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[スキャン結果] パネルには、ウィザードが識別した依存関係の中からプロジェクトに追加することが選択されたファイルが表示されます。

これらの依存関係をプロジェクトに追加するには、[次へ] ボタンをクリックします。依存関係を追加しないでウィザードを終了するには、[キャンセル] ボタンをクリックします。再び依存関係の可能性があるファイルのリストを表示して、それらを追加または削除するには、[戻る] ボタンをクリックします。

[スタティック スキャン ウィザードの完了] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

スタティック スキャン ウィザードが [スタティック スキャン ウィザードの完了] パネルを表示する段階で、ウィザードは選択された依存関係をプロジェクトに追加済みです。

[完了] をクリックすると、ウィザードを閉じて InstallShield に戻ります。

システム検索ウィザード

プロジェクト: *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム検索ウィザードは、インストール前にターゲットのシステム上にある特定のファイル、フォルダー、レジストリキー、.xml ファイルまたは .ini 値を探す Windows Installer の機能を提供します。

タスク ***システム検索ウィザードが起動するには、次の手順を実行します。***

1. **[動作とロジック]** の下のビュー バー/ビュー リストで、**[システム検索]** をクリックします。
2. 以下のいずれかを実行します。
   - 新しいシステム検索を作成するには、グリッド上で右クリックして **[追加]** を選択します。
   - 既存のシステム検索を編集するには、編集するアイテムを右クリックしてから、**[変更]** をクリックします。

システム検索ウィザードには、以下のパネルがあります。

- ようこそ
- 検索する対象を指定してください
- 検索方法
- この値の処理方法を指定してください

[ようこそ] パネル

プロジェクト・ *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム検索ウィザードの [ようこそ] パネルこのウィザードを使って、プロジェクトにシステム検索を追加または変更することができます。システム検索を追加または変更するには [次へ] をクリックします。

検索する対象を指定してくださいパネル

プロジェクト・ *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム検索ウィザードの **[検索内容]** パネルを使って、検索するアイテムの種類、およびターゲット システム内で検索する場所を指定します。選択可能なオプションは以下のとおりです:

- ファイル パス(フォルダーを検索)
- フォルダーパス、全てのドライブを検索する
- フォルダー パス、特定のフォルダーを検索する
- フォルダー パス(特定のファイルを検索)
- ファイル パス(レジストリ エントリで指定)
- フォルダー パス(レジストリ エントリで指定)
- レジストリ エントリ
- ファイル パス(.ini ファイル値で指定)
- フォルダー パス(.ini ファイル値で指定)
- .ini ファイル値
- ファイル パス(コンポーネントのキーファイルで指定)
- フォルダー パス(コンポーネントのキーパスで指定)
- XML ファイル値

ファイルパスまたはフォルダーを含むレジストリ エントリを検索する場合、無事に検索する為にはファイルまたはフォルダーがターゲット システム内に存在しなくてはなりません。そうでない場合、検索からの値をプロパティに設定することはできません。

ファイルまたはフォルダーを検索している場合、その項目への完全パスがプロパティに保存されます。レジストリ値を検索している場合、その値のデータはプロパティに保存されます。.ini ファイルでも同じです。コンポーネントの場合、そのコンポーネントのキーパスがプロパティに保存されます。

スクリプトを使ったプロパティの取得方法については、Windows Installer ヘルプ ライブラリの MsiGetProperty を参照してください。

検索方法パネル (システム検索メソッドの定義)

プロジェクト: *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[検索内容] パネルには、検索をカスタマイズするための設定があります。このパネルの設定は、前のパネルで選択した検索の種類によって異なります。

テーブル 19・検索をカスタマイズする

検索の種類	構成が必要な対応する設定
ファイル パス(フォルダーを検索)	ファイルの詳細とその検索場所を指定します。 詳細については、「[ファイルの詳細の指定] パネル (ファイル検索オプション)」を参照してください。
フォルダーパス、全てのドライブを検索する	検索するフォルダー名およびサブフォルダーの階層数を指定します。 詳細については、「検索方法パネル (フォルダー検索オプション)」を参照してください。
フォルダー パス、特定のフォルダーを検索する	フォルダー名と、それをターゲット システム上で探す場所を指定します。 詳細については、「検索方法パネル(特定のフォルダーオプション)」を参照してください。
フォルダー パス(特定のファイルを検索)	ファイルの詳細とその検索場所を指定します。 詳細については、「[ファイルの詳細の指定] パネル (ファイル検索オプション)」を参照してください。
ファイル パス(レジストリ エントリで指定)	レジストリ エントリの詳細を指定します。 詳細については、「検索方法パネル (レジストリ検索オプション)」を参照してください。
フォルダー パス(レジストリ エントリで指定)	レジストリ エントリの詳細を指定します。 詳細については、「検索方法パネル (レジストリ検索オプション)」を参照してください。
レジストリ エントリ	レジストリ エントリの詳細を指定します。 詳細については、「検索方法パネル (レジストリ検索オプション)」を参照してください。
ファイル パス(.ini ファイル値で指定)	.ini ファイル値についての詳細を指定します。 詳細については、「検索方法パネル (.ini ファイル検索 オプション)」を参照してください。
フォルダー パス(.ini ファイル値で指定)	.ini ファイル値についての詳細を指定します。 詳細については、「検索方法パネル (.ini ファイル検索 オプション)」を参照してください。
.ini ファイル値	.ini ファイル値についての詳細を指定します。 詳細については、「検索方法パネル (.ini ファイル検索 オプション)」を参照してください。

テーブル 19・検索をカスタマイズする (続き.)

検索の種類	構成が必要な対応する設定
ファイル パス(コンポーネントのキーファイルで指定)	コンポーネント ID を指定します。 詳細については、「検索方法パネル (コンポーネント検索オプション)」を参照してください。
フォルダー パス(コンポーネントのキーパスで指定)	コンポーネント ID を指定します。 詳細については、「検索方法パネル (コンポーネント検索オプション)」を参照してください。
XML ファイル値	XML ファイルの詳細とその検索場所を指定します。 詳細については、「[ファイルの詳細の指定] パネル (ファイル検索オプション)」を参照してください。

[ファイルの詳細の指定] パネル (ファイル検索オプション)

プロジェクト・ *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ファイル パス(フォルダーを検索)] タイプのシステム検索を構成する場合、**[フォルダー パス(特定のファイルを検索)]** タイプの検索を構成する場合、または **[XML ファイルの値]** タイプの検索を構成する場合、**[ファイルの詳細とその検索場所を指定]** パネルには、次の設定が表示されます:

テーブル 20・[ファイルの詳細とその検索場所を指定] パネルの設定

オプション	説明
ファイル名	探すファイルまたはアプリケーションの完全名と拡張子を入力します。 ***メモ・*** *ファイル名を入力すると、[詳細] ボタンが有効化されます。特定のバージョン、作成日、サイズ、または言語を含む検索に限定したい場合は、このボタンをクリックします。詳細については、「[ファイルの詳細] ダイアログ ボックス」を参照してください。*
検索先	ターゲット システム上でWindows Installer を検索する場所を指定します。

テーブル 20 • [ファイルの詳細とその検索場所を指定] パネルの設定 (続き.)

オプション	説明
検索するサブフォルダーの最大数	ターゲット システム上で検索するファイルのサブ フォルダーの最大数を指定します。

検索方法パネル (フォルダー検索オプション)

プロジェクト: *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[フォルダー パス(すべてのドライバーを検索)] タイプのシステム検索を構成する場合、**[検索方法]** パネルには、次の設定が表示されます:

テーブル 21 • 検索方法パネルの設定

オプション	説明
フォルダー名	Windows Installer が検索するフォルダーの完全名と拡張子を入力します。
検索先	ターゲット システム上でWindows Installer を検索する場所を指定します。
検索するサブフォルダー数	ターゲット システム上で Windows Installer が検索するサブフォルダーの階層数を指定します。

検索方法パネル(特定のフォルダーオプション)

プロジェクト: *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[フォルダー パス(特定のフォルダーを検索)] タイプのシステム検索を構成する場合、**[検索方法]** パネルには、次の設定が表示されます:

テーブル 22 ・ 検索方法パネルの設定

オプション	説明
フォルダー名	検索するフォルダーの完全名と拡張子を入力します。
検索先	ターゲット システム上でWindows Installer を検索する場所を指定します。完全パスを指定するか、前回の検索で使用したパスを選択できます。 完全パスを指定するには、既存ディレクトリを選択して [参照] ボタンをクリックするか、新しいパスを作成します。 前回の検索からのパスを指定する場合、一覧から選択します。リストが空白の場合、プロジェクトにその他の検索が含まれていないことを示します。
検索するサブフォルダー数	ターゲット システム上で Windows Installer が検索するサブフォルダーの階層数を指定します。

検索方法パネル (レジストリ検索オプション)

プロジェクト・ *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[フォルダー パス(レジストリ エントリで指定)] タイプのシステム検索、または**レジストリ エントリ** タイプのシステム検索を構成する場合、**[検索方法]** パネルに次の設定が表示されます:

テーブル 23 ・ 検索方法パネルの設定

設定	説明
レジストリ ルート	ターゲット システム上で Windows Installer が検索するレジストリ ルートを指定します。

テーブル 23・検索方法パネルの設定 (続き.)

設定	説明
レジストリ キー	検索する項目と関連したレジストリキーを正確に入力します。例えば、HKEY_LOCAL_MACHINE¥SOFTWARE¥Adobe¥Acrobat をターゲット システムで検索したい場合、この設定に**SOFTWARE¥Adobe¥Acrobat** のように入力します。 ***ヒント・** Windows レジストリ エディターから正しいキー名をコピーして構文が正しいことを確認してください。*
レジストリ値 (オプション)	特定のレジストリ値を検索するには、Windows レジストリ エディターに表示されるレジストリ値を正確に入力します。 ***メモ・** この設定が空白の場合、システム検索はレジストリ キーのデフォルト値を探します。*
レジストリの 64 ビットの部分を検索	64 ビット ターゲット システムは、通常 2 つの HKEY_LOCAL_MACHINE¥Software キーを持ちます: ・ HKLM¥Software – 64 ビット アプリケーション用 ・ HKLM¥Software¥Wow6432Node – 32 ビット アプリケーション用 Windows Instaler が 64 ビット ターゲット システム上でレジストリの 64 ビット 部分をチェックする場合は、このチェックボックスを選択します。

検索方法パネル (.ini ファイル検索 オプション)

***プロジェクト・** システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ファイル パス(.ini ファイル値で指定)] タイプのシステム検索、または**.ini ファイル値** タイプのシステム検索を構成する場合、**[検索方法]** パネルに次の設定が表示されます:

テーブル 24 • 検索方法パネルの設定

オプション	説明
INI ファイル名	.ini ファイル名をターゲット システムに表示されるとおりに指定します。このファイルは、Windows フォルダーに既存するファイルです。
INI セクション名	Windows Installer で検索する値を含む .ini ファイルのセクション名を指定します。
INI キー名	Windows Installer で検索する値を含むキーの名前を指定します。
行全体を読み込む	Windows Installer が .ini ファイルの行全体を読み込む場合は、このチェックボックスを選択します。

検索方法パネル (コンポーネント検索オプション)

プロジェクト: *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ファイル パス(コンポーネントのキー ファイルで指定)] タイプのシステム検索、または**[フォルダー パス(コンポーネントのキー パスで指定)]** タイプのシステム検索を構成する場合、システム検索ウィザードで **[検索方法]** パネルが表示されます。このパネルを使って、そのキー パスまたはキー ファイルを検索に使用するコンポーネントの ID を指定します。コンポーネント ID は波括弧 {} で囲む必要があります。

[XML ファイル] パネル (XML ファイル オプション) で検索するデータを指定します。

プロジェクト: *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

[XML ファイル値] タイプのシステム検索を構成する場合、システム検索ウィザードで **[XML ファイルで検索するデータを指定]** パネルが表示されます。このパネルでは、ターゲット システム上の .xml ファイル内で検索する .xml 要素を指定します。

テーブル 25 - [XML ファイル パネルで検索するデータを指定] パネルの設定

オプション	説明
XML 要素への XPath (例: 要素/サブ要素)	検索する XML ファイル内のデータを識別する XPath 式を指定します。 XPath 式の例は、「XPath 式を使って、XML ファイル内の XML データを検索する」を参照してください。
検索	検索する XML Xデータの種類を選択します。指定する要素の属性の値、要素のコンテンツ、または要素の存在を検索できます。

この値の処理方法を指定してくださいパネル

プロジェクト- *システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[値の処理方法] パネルで、検索結果を保存する場所をかを指定します。検索が失敗したとき、プロパティは未定義となります。選択可能なオプションは以下のとおりです:

テーブル 26 - この値の処理方法を指定してくださいパネルの設定

設定	説明
値を保存するプロパティ	システム検索がターゲット システムで検索を行う場所を格納するのに使用するパブリック プロパティを選択します。定義済みパブリック プロパティ以外で値を格納する場合、この設定でプロパティの名前を入力します。プロパティはパブリック プロパティであり、大文字で構成される識別子を含まなくてはなりません。 ***メモ-*** *[プロパティ マネージャー] ビューを使って、プロジェクトに新しいプロパティを追加することができます。作成する新しいプロパティはパブリック プロパティでなければならず、すべてが大文字の識別子で構成されている必要があります。*

テーブル 26・この値の処理方法を指定してくださいパネルの設定 (続き.)

設定	説明
追加オプション	適切なオプションを選択します: ・ **プロパティに値を保存する** — このウィザード パネルで識別されたパブリック プロパティの値を格納します。 ・ **インストール条件にプロパティを使用する** — このウィザード パネルで識別されたパブリック プロパティの値を格納して、インストール条件で使用します。 このオプションを選択してから [完了] ボタンをクリックすると、InstallShield で条件ビルダーが表示されます。ここでインストールが満たさなくてはならない条件を指定できます。

この値の処理方法を指定してくださいパネル

***プロジェクト・** システム検索ウィザードは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[値の処理方法] パネルで、検索結果を保存する場所をかを指定します。検索が失敗したとき、プロパティは未定義となります。選択可能なオプションは以下のとおりです:

テーブル 27・この値の処理方法を指定してくださいパネルの設定

設定	説明
値を保存するプロパティ	システム検索がターゲット システムで検索を行う場所を格納するのに使用するパブリック プロパティを選択します。定義済みパブリック プロパティ以外で値を格納する場合、この設定でプロパティの名前を入力します。プロパティはパブリック プロパティであり、大文字で構成される識別子を含まなくてはなりません。 ***メモ・** [プロパティ マネージャー] ビューを使って、プロジェクトに新しいプロパティを追加することができます。作成する新しいプロパティはパブリック プロパティでなければならず、すべてが大文字の識別子で構成されている必要があります。*

テーブル 27 • この値の処理方法を指定してくださいパネルの設定 (続き.)

設定	説明
追加オプション	適切なオプションを選択します: • **プロパティに値を保存する** — このウィザード パネルで識別されたパブリック プロパティの値を格納します。 • **インストール条件にプロパティを使用する** — このウィザード パネルで識別されたパブリック プロパティの値を格納して、インストール条件で使用します。 このオプションを選択してから [完了] ボタンをクリックすると、InstallShield で条件ビルダーが表示されます。ここでインストールが満たさなくてはならない条件を指定できます。 • **コンポーネントのインストール先としてこのプロパティを使用** — このウィザード パネルで識別されたパブリック プロパティの値を格納して、指定するコンポーネントのインストール先として使用します。 このオプションは、フォルダーパス、フォルダーを含むレジストリ エントリ、フォルダーを含む .ini ファイル、キーパスがフォルダーであるコンポーネント、または XML ファイルを検索する場合にのみ利用可能です。

トランスフォーム ウィザード

トランスフォーム ウィザードは、インストール プロジェクトのトランスフォームの作成および適用手順を案内します。トランスフォームは、類似する 2 つのインストール プロジェクトの差分を意味します。1 つのプロジェクトにトランスフォームを適用すると、2 つのプロジェクト間の変更を組み込むように更新または変更されます。

タスク

トランスフォーム ウィザードを起動するには、次を実行します。

[ツール] メニューで、**[トランスフォーム ウィザード] ボタン**をクリックします。InstallShield でプロジェクトを開く必要はありません。

トランスフォームウィザードには、次のような関連パネルがあります。

- ようこそ
- ファイルの指定
- 検証設定
- エラー条件を抑制する
- 出力ファイル名の指定
- 概要
- トランスフォーム ウィザードを完了しています

[ようこそ] パネル

トランスフォームウィザードでは、トランスフォームの作成とセットアッププロジェクトへの適用を、順を追って実行します。トランスフォームは、2 つのセットアッププロジェクトの相違点を表わします。たとえば、ネットワーク管理者がある製品を会社の部署によって異なる構成で配布する場合があります。この場合、製品の各構成に対しトランスフォームを作成し、必要に応じて適切なトランスフォームを適用します。

パネルのオプション

トランスフォームの作成

類似する 2 つのセットアップ プロジェクトを比較して、それらの差分をトランスフォームとして作成するには、このオプションを選択します。このトランスフォームは、最終セットアップをビルドする前や実行時に適用できます。

トランスフォームの適用

既存の Windows Installer セットアップにトランスフォームを適用するには、このオプションを選択します。このオプションは、ネットワーク管理者がさまざまなユーザー向けに製品をカスタマイズする場合に役立つことがあります。

[ファイルの指定] パネル

トランスフォームを作成中の場合は、比較する MSI ファイルを 2 つ選択します。トランスフォームを適用する場合は、トランスフォームの適用対象の MSI ファイルと、適用するトランスフォームを選択します。

パネルのオプション

ベースパッケージ

このフィールドには、現在開かれているプロジェクトへのパスが格納されています。この値を変更する場合は、別のプロジェクトへのパスを入力するか、または [参照] ボタンをクリックして別のプロジェクトを指定します。トランスフォームを実行中の場合は、トランスフォーム適用対象のファイルへのパスがこのフィールドに格納されます。

ターゲット パッケージ

このオプションは、トランスフォームを作成中の場合のみ使用できます。ベースファイルと比較する MSI ファイルへのパスを入力するか、または [参照] ボタンをクリックしてこのファイルを指定します。これら 2 つのファイルの間の違いが比較され、ベースパッケージをターゲットパッケージにアップデートするトランスフォームが作成されます。

トランスフォーム

このオプションは、トランスフォームを適用するよう選択した場合のみ使用できます。適用するトランスフォームファイル (*.mst) へのパスを入力するか、または [参照] ボタンをクリックしてこのファイルを指定します。

[検証設定] パネル

このウィザードの [ようこそ] パネルで [トランスフォームの作成] オプションを選択すると、[検証設定] パネルが表示されます。

[検証設定] パネルでは、トランスフォームをどのように検証したいかを聞いてきます。トランスフォームを実行するには、ここで指定した検証項目の条件が満たされている必要があります。

パネルのオプション

トランスフォームをベース パッケージへ適用する前に検証を実行しない

トランスフォームを適用する前に検証を行わない場合、このオプションを選択します。

トランスフォームをベース パッケージへ適用する前に以下を検証する

このオプションを選択して、トランスフォームの適用前に特定の条件をチェックします。これらの条件が満たされていない場合、トランスフォームは適用されません。

テーブル 28 • トランスフォームをベース パッケージに適用する際の条件

オプション	説明
デフォルト言語はベースパッケージと一致しなくてはいけません	このオプションを選択して、トランスフォーム適用対象のパッケージの言語と、トランスフォームの言語を一致させます。
製品はベースパッケージと一致しなくてはいけません	ベースパッケージと同じ製品コードを持つ .msi ファイルにのみトランスフォームを適用する場合、このオプションを選択します。
プラットフォームはベースパッケージと一致する	ベースパッケージの対象のプラットフォームと、トランスフォームのターゲットプラットフォームを一致させる場合、このオプションを選択します。
アップグレードコードはベースパッケージと一致しなくてはいけません	.msi ファイルとトランスフォームのアップグレードコードが一致する場合にのみトランスフォームを適用するときは、このオプションを選択します。
製品バージョン	評価の基準にするバージョンの種類を選択します。メジャーバージョン、マイナーバージョン、またはアップグレードバージョンを組み合わせて選択できます。製品バージョンの認証を行わないよう指定することもできます。

テーブル 28・トランスフォームをベース パッケージに適用する際の条件 (続き.)

オプション	説明
バージョンの関係	製品バージョンの評価方法を選択します。オプションは以下のとおりです。 ・ **なし** – 製品バージョンの検証を行いません。 ・ **適用バージョン < ベース バージョン**–トランスフォーム適用対象の .msi ファイルの製品バージョンが、ベース MSI ファイルのバージョン番号未満の場合にのみ、トランスフォームを適用します。 ・ **適用バージョン<= ベース バージョン**–トランスフォーム適用対象の .msi ファイルの製品バージョンが、ベース MSI ファイルのバージョン番号以下の場合にのみ、トランスフォームを適用します。 ・ **適用バージョン < ベース バージョン**–トランスフォーム適用対象の .msi ファイルの製品バージョンが、ベース MSI ファイルのバージョン番号 の場合にのみ、トランスフォームを適用します。 ・ **適用バージョン>= ベース バージョン**–トランスフォーム適用対象の .msi ファイルの製品バージョンが、ベース MSI ファイルのバージョンよりも大きい場合にのみ、トランスフォームを適用します。 ・ **適用バージョン > ベース バージョン**–トランスフォーム適用対象の .msi ファイルの製品バージョンが、ベース .msi ファイルのバージョン番号よりも大きい場合にのみ、トランスフォームを適用します。

[エラー条件の抑制] パネル

ターゲット MSI ファイルへのトランスフォームの適用中に、特定のエラーコードが生成されることがあります。このパネルでは、これらのエラーコードを抑制し、エンドユーザーに対して表示されないようにするためのオプションを指定できます。

パネルのオプション

ベースパッケージにトランスフォームを適用する際にエラー条件を抑制しない

トランスフォームの適用中にレポートされるエラーを非表示にしない場合、このオプションを選択します。

ベースパッケージにトランスフォームを適用する際に次のエラー条件を抑制する

特定のエラーコードをエンドユーザーに対して非表示にする場合、このオプションを選択します。以下のエラーを選択して抑制できます。

テーブル 29・抑制可能なエラー条件

エラー	説明
既存の行を追加しようとしたとき	トランスフォームで既存の MSI データベースに行を追加しようとすると、エラーが生成されます。このエラーを抑制するには、このオプションを選択します。
存在しない行を削除しようとしたとき	トランスフォームで、存在しない MSI データベースから行を削除しようとした結果発生するエラーを抑制するには、このオプションを選択します。
既存のテーブルを追加しようとしたとき	トランスフォームで既存の MSI データベースにテーブルを追加しようとした場合に発生するエラーを抑制するには、このオプションを選択します。
存在しないテーブルを削除しようとしたとき	トランスフォームで、存在しない MSI データベースからテーブルを削除しようとした結果発生するエラーを抑制するには、このオプションを選択します。
存在しない行を更新しようとしたとき	トランスフォームで、存在しない MSI データベースの行をアップデートしようとした場合、エラーが生成されます。このエラーを抑制するには、このオプションを選択します。
トランスフォームとデータベースコードのページが一致しない	トランスフォームのコードページとターゲットデータベースのコードページの不一致が原因で発生するエラーを抑制するには、このオプションを選択します。

[出力ファイル名の指定] パネル

このパネルでは、2 つの MSI ファイルを比較した結果作成されるトランスフォーム (*.mst) ファイルの場所と名前を指定するよう、メッセージが表示されます。

パネルのオプション

フォルダーの場所とファイル名

トランスフォームを作成する場合、作成するトランスフォームファイルのパスとファイル名を入力します。トランスフォームを適用する場合は、トランスフォームの適用後に作成される新しい MSI パッケージのパスと名前を入力します。

[概要] パネル

[概要] パネルには、トランスフォームウィザードで選択したすべての設定が表示されます。いずれかの設定を変更する必要がある場合は、該当するパネルが表示されるまで [戻る] ボタンをクリックします。

[トランスフォーム ウィザードの完了] パネル

トランスフォームウィザードは、最終パネルでウィザードが無事に完了したことを知らせます。[完了] ボタンをクリックすると、IDE に戻ります。

トランスフォームの詳細については、「トランスフォームの作成」を参照してください。このページでは、トランスフォームの概要について説明するとともに、トランスフォームの作成方法やトランスフォームを適用できるさまざまな方法を紹介するページへのリンクを示しています。

ヒント: *.msi データベースにトランスフォームが適用されているときに発生する可能性があるエラーの詳細については、Windows Installer ヘルプライブラリの 「MsiDatabaseApplyTransform」 を参照してください。*

アップグレード検証ウィザード

アップグレード検証ウィザードは一連のテストを実行し、インストールが古いバージョンを適切にアップグレードするかどうかを判断します。

プロジェクト: *このウィザードは、次のプロジェクトの種類には適用されません。*

- *InstallScript*
- *InstallScript オブジェクト*

タスク ***アップグレード検証ウィザードを起動するには、次の手順を実行します。***

[ビルド] メニューで **[検証]** ドロップダウン リストをクリックし、**[アップグレード検証ウィザード]** をクリックします。

このウィザードの関連パネルを以下に示します。

- ようこそ
- 設定
- 概要

[ようこそ] パネル

アップグレード検証ウィザードは一連のテストを実行し、セットアップが古いバージョンを適切にアップグレードするか否かを判断します。

続行するには [次へ] をクリックしてください。

[設定] パネル

アップグレード検証用の以下の基本構成を指定してください:

テーブル 30 • [設定] パネル

構成	説明
セットアップの最新バージョンを指定します。	参照ボタンをクリックして、このファイルを見つけます。 ***ヒント:*** *ターゲットマシンに配置する最新セットアップを新規イメージまたは新規バージョン、そしてアップグレード前のターゲットマシンの以前のセットアップを既存イメージまたは既存バージョンとして取り扱ってください。*
検証するセットアップの以前のバージョンを指定します。	[追加] をクリックして、既存のリストに付加します。*.msi または *.exe Windows Installer セットアップファイルを追加することができます。セットアップの以前のバージョンを複数指定することもできます。以前のバージョン リストからアイテムを削除するには、まずリスト コントロールで選択してから [削除] をクリックします。

[次へ] をクリックすると検証が始まります。

[概要] パネル

概要パネルには、検証結果が表示されます。<ISProjectDataFolder>¥Upgrade Validation Log¥Results.log で保存された対応するログファイルを開いて、結果を外部表示することもできます。

メモ: *[戻る] ボタンをクリックして設定を変更し、別の設定で再検証を行うことができます。*

次のテーブルは各ログファイルエントリの横にあるアイコンを説明します。

テーブル 31 • アイコン

ボタン	説明
	これは情報通知アイコンです。
	これはエラー通知アイコンです。
	これは警告通知アイコンです。

ユーザー インターフェイス ウィザード

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ユーザー インターフェイス ウィザードでは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに様々な定義済みウィザード ページを追加できます。以下に例を示します:

- エンド ユーザーが .msi パッケージのインストール ディレクトリを選択できるページ
- エンド ユーザーがカスタマー情報およびシリアル番号を入力できるページ
- 機能ツリーおよび機能の説明とサイズを表示してエンド ユーザーがインストールする機能を選択できるページ

タスク ***ユーザー インターフェイス ウィザードを起動するには、以下の手順に従います:***

1. **[ユーザー インターフェイス]** の下のビュー バー/ビュー リストにある **[ウィザード インターフェイス]** をクリックします。
2. **[ウィザード インターフェイス]** エクスプローラーで、**[ウィザード ページ]** を右クリックしてから、**[定義済みページの追加]** をクリックします。

ユーザー インターフェイス ウィザードは次のパネルで構成されます:

- [定義済みタスク ページ] パネル
- [タスク構成] パネル

[定義済みタスク ページ] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

[定義済み タスク ページ] パネルでは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する定義済みウィザード ページの種類を選択します。このパネルではまた、新しい定義済みページをウィザード ページのシーケンスにスケジュールする場所を指定することもできます。プロジェクトにウィザード ページを追加した後にシーケンスを変更することができます。

以下のテーブルは、使用可能な定義済みウィザード ページの各種類について説明します:

テーブル 32 • 定義済みタスク ページの種類

ウィザード ページの種類	説明
インストール フォルダーを参照	この BrowseFolder ウィザード ページを使って、エンド ユーザーは .msi パッケージのインストール ディレクトリを選択できます。アドバンスト UI またはスイート/アドバンスト UI インストールは、パッケージを起動したときに Windows Installer プロパティ INSTALLDIR を使って .msi パッケージにパスを渡します。 この種類のページは、通常 InstallationType または InstallationProgress ウィザード ページの前にシーケンスされます。
Web 配置 パッケージのユーザー情報を入力	 ***エディション:*** *InstallShield Premier Edition は、Web 配置パッケージをサポートします。* この CustomerInformation ウィザード ページでは、エンド ユーザーが Web 配布パッケージを公開するための情報を入力し、その値を COMPUTERNAME、USERNAME、PASSWORD、および SITE としてパッケージに渡します。 この種類のページは、通常 LicenseAgreement ウィザード ページの後にシーケンスされます。
MSI パッケージのユーザー情報を入力	この CustomerInformation ページで、エンド ユーザーが入力するユーザー情報および .msi パッケージのシリアル番号の値を Windows Installer プロパティ USERNAME、COMPANYNAME、および SERIALNUMBER を使ってパッケージに渡します。 この種類のページは、通常 LicenseAgreement ウィザード ページの後にシーケンスされます。

テーブル 32 • 定義済みタスク ページの種類 (続き.)

ウィザード ページの種類	説明
インストールのパスワードを入力	このパスワード ウィザード ページを使って、エンド ユーザーはインストールのパスワードを入力します。エンド ユーザーが正しいパスワードを入力しなかった場合、インストールが終了します。 このページは、通常インストールの最初にシーケンスされます。 ***メモ•*** *この種類のウィザード ページをプロジェクトに追加すると、InstallShield によってプレースホルダー イメージ コントロールが追加され、ページにロック イメージを表示することができます。コントロールの "リソース" 設定を構成してこのコントロールと共に表示するファイルを指定するか、イメージを除外する場合は、イメージ コントロールを削除します。*
データベース サーバー用のログイン情報を入力	 ***エディション•*** *スイートの QL サポートは、InstallShield の Premier Edition で提供されています。* この SQLLogin ウィザード ページを使って、エンド ユーザーはデータベース サーバー ログイン情報 (データベース サーバー名、認証資格情報、データベース カタログ名など) を入力し、スイートに含まれる 1 つ以上の .msi パッケージによって参照されるデータベース サーバーへの接続を設立することができます。 これらのダイアログに入力された値 (またはスイート コマンドラインに入力された値) は、選択された .msi パッケージ コマンドラインにパスワードが非表示とマークされた状態で渡されます。 ***メモ•*** *プロジェクトにこの種類のウィザード ページを追加する場合、データベース テクノロジ、ドライバー、および基本の Windows インストーラ プロパティを指定して、これらのプロパティを受け取る .msi パッケージを選択します 。これは、スイートには異なる SQL サーバーが必要、または同じ SQL サーバーの共有が必要な複数のパッケージを含んでいる可能性があるためです。*
メンテナンスへようこそページのアップデート オプション	MaintenanceUpdateWelcome ウィザード ページに含まれている [アップデート] オプションを使って、エンド ユーザーは最新のアドバンスト UI またはスイート/アドバンスト UI アップデートをダウンロードおよびインストールすることができます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

テーブル 32 • 定義済みタスク ページの種類 (続き.)

ウィザード ページの種類	説明
補足使用許諾契約の表示および同意	この LicenseAgreement ウィザード ページは、エンド ユーザーによる補足使用許諾契約 (EULA) の同意を求めます。 各 EULA ファイルの名前は一意でなくてはなりません。そうでない場合、テキストとプロパティーが関連付けられません。 この種類のページは、通常もう 1 つの LicenseAgreement ウィザード ページの後にシーケンスされます。
補足使用許諾契約のスクロール表示および同意	この LicenseAgreement ウィザード ページは、エンド ユーザーが画面をスクロールした上で補足使用許諾契約 (EULA) に同意することを求めます。 各 EULA ファイルの名前は一意でなくてはなりません。そうでない場合、テキストとプロパティーが関連付けられません。 この種類のページは、通常もう 1 つの LicenseAgreement ウィザード ページの後にシーケンスされます。
機能とそのサイズを表示	このうぃざーど ページは機能ツリー、およびプロジェクトで指定された機能の説明とサイズを表示します。ここでエンド ユーザーは、インストールする機能を選ぶことができます。 このウィザード ページには、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる各機能の "説明" および "コスト" 設定に値を入力する必要があります。 この種類のページは、通常、ビルトイン InstallationFeatures ウィザード ページの代わりとなります。

[タスク構成] パネル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ユーザー インターフェイス ウィザードの [タスク構成] パネルでは、プロジェクトに追加する定義済みウィザード ページの追加情報を指定できます。このパネルは、一部の種類の定義済みウィザード ページにのみ表示されます。

テーブル 33 • [タスク構成] パネルの必須情報

ウィザード ページの種類	必須情報の説明
インストール フォルダーを参照	エンド ユーザーがパスを指定できる、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージを選択します。
MSI パッケージのユーザー情報を入力	エンド ユーザーが顧客情報を指定できる、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージを選択します。
インストールのパスワードを入力	この種類のウィザード ページをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加すると、[タスク構成] パネルは表示されません。
補足使用許諾契約の表示および同意	補足 LicenseAgreement ウィザード ページに表示する EULA ファイル (.rtf) を指定します。
補足使用許諾契約のスクロール表示および同意	補足 LicenseAgreement ウィザード ページに表示する EULA ファイル (.rtf) を指定します。
機能とそのサイズを表示	この種類のウィザード ページをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加すると、[タスク構成] パネルは表示されません。

Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザード

Visual Basic .NET、C# .NET、および Visual C++ .NET 用の Visual Studio .NET ウィザードは、InstallShield の新規インストール プロジェクトを作成し、それを Microsoft Visual Studio ソリューションに追加します。

メモ: *Visual Studio .NET ウィザードは、システムに Microsoft Visual Studio がインストールされている場合のみ使用可能です。*

InstallShield インストール プロジェクトを作成中の場合、Visual Studio .NET ウィザードは以下の処理を行います。

- 新しい InstallShield プロジェクトを作成して ([新規プロジェクト] ダイアログ ボックスで指定されたファイル名) 、ソリューション (.sln ファイル) に追加します。
- ([オプション] ダイアログ ボックスの .NET タブで)ビルド時にスキャン オプションに 依存関係とプロパティを設定すると、ウィザードはビルド時にすべての依存関係をプロジェクトへ追加します。
- ソリューションにある各プロジェクトから InstallShield プロジェクトにプライマリ出 力を追加します。
- リリース設定をアップデートしてダウンロードで .NET Framework の適切なバージョンを配布します。

ウィザードを起動するには、[新規プロジェクト] ダイアログ ボックス の該当するプロジェクトタイプを選択します。

このウィザードの関連パネルを以下に示します。

- [ようこそ] パネル
- [ソリューション] パネル

[ようこそ] パネル

Visual Studio .NET ウィザードを利用して、InstallShield インストール プロジェクトを Microsoft Visual Studio .NET ソリューションに追加することができます。

メモ: *Visual Studio .NET ウィザードは、システムに Microsoft Visual Studio がインストールされている場合のみ使用可能です。*

[次へ] をクリックしてソリューションを選択してください。

[ソリューション] パネル

このパネルでは、InstallShield インストール プロジェクトを追加する Visual Studio ソリューションへのパスを入力または参照します。

[完了] をクリックして、InstallShield プロジェクトを作成し、選択済みのソリューションを追加します。

Visual Studio デプロイメント プロジェクト インポート ウィザード

Visual Studio デプロイメント プロジェクト インポート ウィザードを使って、Visual Studio セットアップまたはマージ モジュール プロジェクト (.vdproj) を InstallShield プロジェクト (.ism) にインポートすることができます。複数の Visual Studio プロジェクトを InstallShield プロジェクトにインポートする場合、このウィザードを繰り返し使用することができます。

重要: *InstallShield プロジェクトにインポートする Visual Studio セットアップ またはマージ モジュール プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合、その InstallShield は、Visual Studio セットアップまたはマージ モジュール プロジェクトおよびそのプロジェクトのすべての依存関係を含む、同じ Visual Studio ソリューションに含まれていなくてはなりません。*

タスク

Visual Studio デプロイメント プロジェクト インポート ウィザードを起動するには、以下の手順の 1 つを行います:

- InstallShield を Visual Studio に統合しないで使用している場合: [**プロジェクト**] メニューから [**VS デプロイメント プロジェクトのインポート**] ボタンをクリックします。
- InstallShield を Visual Studio 内部から使用している場合: InstallShield ツールバーから、[**Visual Studio デプロイメント プロジェクト インポート ウィザード**] をクリックします。

このウィザードには、以下のパネルがあります:

- ようこそ
- プロジェクト ファイル
- オプション
- 概要

[ようこそ] パネル

Visual Studio デプロイメント プロジェクト インポート ウィザードを使って、Visual Studio セットアップまたはマージ モジュール プロジェクト (.vdproj) を InstallShield プロジェクト (.ism) にインポートすることができます。複数の Visual Studio プロジェクトを InstallShield プロジェクトにインポートする場合、このウィザードを繰り返し使用することができます。

重要: *InstallShield プロジェクトにインポートする Visual Studio セットアップ またはマージ モジュール プロジェクトに 1 つ以上のプロジェクト出力が含まれている場合、その InstallShield は、Visual Studio セットアップまたはマージ モジュール プロジェクトおよびそのプロジェクトのすべての依存関係を含む、同じ Visual Studio ソリューションに含まれていなくてはなりません。*

ウィザードの使用を開始するには、[次へ] をクリックします。

[プロジェクトファイル] パネル

[プロジェクト ファイル] パネルを使って、インポートする Visual Studio プロジェクト (.vdproj) を指定します。プロジェクトには、セットアップ プロジェクトまたはマージ モジュール プロジェクトを選択できます。

[オプション] パネル

Visual Studio デプロイメント プロジェクト インポート ウィザードを使って、プロジェクト出力、ファイル、レジストリ キー、ファイル拡張子、カスタム アクション、ターゲット システム検索、および前提条件を Visual Studio プロジェクトから InstallShield プロジェクトにインポートできます。[オプション] パネルでは、適切な場合、どのプロパティを Visual Studio プロジェクトから InstallShield プロジェクトにインポートするのかも選択できます。

メモ- オプションのチェック ボックスを選択すると、InstallShield プロジェクト内の既存の値が Visual Studio プロジェクトで構成された値で上書きされます。たとえば、[製品名] チェック ボックスを選択すると、InstallShield プロジェクトの "製品名" 設定の値が Visual Studio プロジェクトで設定された値で上書きされます。

テーブル 34 - インポートで使用できるオプション

オプション	説明
製品名	インポートしている Visual Studio プロジェクトで構成された ProductName プロパティを InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 製品名は、InstallShield の [一般情報] ビューで構成されます。 ***プロジェクト-** このプロパティは、マージ モジュール プロジェクトには適用しません。*
製品バージョン	インポートしている Visual Studio プロジェクトで構成された ProductVersion プロパティの値を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 製品バージョンは、InstallShield の [一般情報] ビューで構成されます。
INSTALLDIR	インポートしている Visual Studio プロジェクトのアプリケーション ファオルダに構成された DefaultLocation プロパティの値を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 このチェック ボックスを選択すると、InstallShield は [一般情報] ビューの INSTALLDIR 設定の値を、Visual Studio プロジェクトで構成されたパスでアップデートします。 ***メモ-** Visual Studio では、アプリケーション フォルダーに複数のフォーマットされたプロパティが含まれたディレクトリ パス (例、[ProgramFilesFolder][Manufacturer]¥[ProductName]) を指定できます。Visual Studio プロジェクトは、実行時にディレクトリ カスタム アクションを使用して、パスを解決します。ただし、InstallShield では、この種類のディレクトリ パスはサポートされていません。したがって、InstallShield はパスを変換処理中に解決し、パスの INSTALLDIR プロパティを使用します。*

テーブル 34 • インポートで使用できるオプション (続き.)

オプション	説明
[プログラムの追加と削除] のプロパティ	Visual Studio プロジェクトで構成された"プログラムの追加と削除" プロパティ (AddRemoveProgramsIcon、Manufacturer、Description、ManufacturerUrl、Author、SupportUrl、および SupportPhone) を InstallShield プロジェクトで使用する場合は、このチェック ボックスを選択します。 このチェック ボックスを選択すると、InstallShield は [一般情報] ビューにある次の設定の値を、Visual Studio プロジェクトで構成された値でアップデートします。 • 表示アイコン (Visual Studio プロジェクトでは、AddRemoveProgramsIcon プロパティ) • 発行者 (Visual Studio プロジェクトでは、Manufacturer プロパティ) • [プログラムの追加と削除] のコメント (Visual Studio プロジェクトでは、Description プロパティ) • 発行元/製品 URL (Visual Studio プロジェクトでは、ManufacturerUrl プロパティ) • サポート連絡先 (Visual Studio プロジェクトでは、Author プロパティ) • サポート URL • サポート電話番号 ***プロジェクト:*** *このプロパティは、マージ モジュール プロジェクトには適用しません。*

テーブル 34 • インポートで使用できるオプション (続き.)

オプション	説明
[概要情報ストリーム] のプロパティ	インポートしている Visual Studio プロジェクトで構成された Summary Information Stream プロパティ (Title、Subject、Keywords、および TargetPlatform) を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 このチェック ボックスを選択すると、InstallShield は [一般情報] ビューにある次の設定の値を、Visual Studio プロジェクトで構成された値でアップデートします。 • タイトル • 件名 • キーワード • "テンプレートの概要" 設定の "プロセッサ" の種類 (Visual Studio プロジェクトでは、TargetPlatform プロパティ) ***プロジェクト:*** *このプロパティは、マージ モジュール プロジェクトには適用しません。*
製品コード	インポートしている Visual Studio プロジェクトで構成された ProductCode プロパティの値を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 製品コードは、InstallShield の [一般情報] ビューで構成されます。 ***プロジェクト:*** *このプロパティは、マージ モジュール プロジェクトには適用しません。*
アップグレード コード	インポートしている Visual Studio プロジェクトで構成された UpgradeCode プロパティの値を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 アップグレード コードは、InstallShield の [一般情報] ビューで構成されます。 ***プロジェクト:*** *このプロパティは、マージ モジュール プロジェクトには適用しません。*

テーブル 34 • インポートで使用できるオプション (続き.)

オプション	説明
すべてのユーザー	インポートしている Visual Studio プロジェクトで構成された InstallAllUsers プロパティの値を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 このチェック ボックスを選択すると、InstallShield は [プロパティ マネージャー] ビューの ALLUSERS プロパティの値を、Visual Studio プロジェクトの InstallAllUsers プロパティで構成された値に基づいてアップデートします。 ***プロジェクト:*** *このプロパティは、マージ モジュール プロジェクトには適用しません。*

テーブル 34・インポートで使用できるオプション (続き.)

オプション	説明
プロジェクト言語	インポートしている Visual Studio プロジェクトの Localization プロパティで選択された言語を InstallShield プロジェクトで使用するには、このチェック ボックスを選択します。 Visual Studio プロジェクトを InstallShield プロジェクトにインポートするときに、以下の条件がある場合、InstallShield はプロジェクトに含まれる既存の文字列エントリ値を Visual Studio プロジェクトの言語のデフォルト文字列エントリ値で置換します: ・ Visual Studio デプロイメント プロジェクト ウィザードで [プロジェクト言語] チェック ボックスを選択している。 ・ Visual Studio プロジェクトの言語が InstallShield プロジェクトのデフォルト言語と一致しない。(Visual Studio で、Localization プロパティがプロジェクトの言語を示します。InstallShield では、[文字列エディター] ビューの ″デフォルト言語″ 設定が、プロジェクトのデフォルト言語を示します。) ・ InstallShield は、Visual Studio プロジェクトで使用されている言語をサポートします。(InstallShield Profesioanl Edition を使用している場合、Visual Studio プロジェクトで使用した言語がサポートされていない場合があります。) たとえば、この [プロジェクト言語] チェック ボックスを選択していて、InstallShield プロジェクトの言語がスペイン語で、Visual Studio プロジェクトの言語がドイツ語であり、さらに InstallShield Premier Edition を使用している場合、InstallShield はプロジェクト内のスペイン語のランタイム文字列をデフォルトのドイツ語翻訳で置換します。このため、[一般情報] ビュー内の ″発行者″ 設定を更新している場合など、文字列エントリの値を編集した後にウィザードで Visual Studio プロジェクトの言語をインポートすると、InstallShield は ″発行者″ 設定の値およびその他の文字列の値を、デフォルトのドイツ語の文字列エントリの値に置換します。 したがって、Visual Studio プロジェクトのインポート中にプロジェクト言語が変更される場合は、[一般情報] ビューと [文字列エディター] ビューで設定を確認の上、適切な場合は文字列エントリを変更してください。 ***プロジェクト:*** *このプロパティは、マージ モジュール プロジェクトには適用しません。*

[概要] パネル

[概要] パネルでは、Visual Studio デプロイメント プロジェクト インポート ウィザードで指定した設定を確認できます。設定のいずれかを変更する場合は、該当するパネルが表示されるまで [戻る] ボタンをクリックします。プロジェクトをインポートするには、[完了] ボタンをクリックします。

ビュー リファレンス

InstallShield インストール開発環境 (IDE) は、あらゆる範囲にわたる機能をまとめた複数のビューで構成されます。[ビュー リファレンス] セクションは、InstallShield インターフェイスの各ビューについて説明します。

[インストール情報] ビュー

プロジェクト: *[インストール情報] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*
- *MSIX*

[インストール情報] ビューには、ビルド中のインストールについての一般設定を構成することができる他のビューへのリンクが含まれています。

一般情報

プロジェクト: *[一般情報] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[一般情報] ビュー には、プロジェクト、会社、および製品に関する基本情報が表示されます。このビューに入力した情報には、参照だけに使用する情報、(基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトで) Windows ロゴ要件を満たすために使用する情報 、およびプロジェクトの基本的な設定に使用する情報があります。

[一般情報] ビュー

プロジェクト: *[一般情報] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[一般情報] ビュー には、プロジェクト、会社、および製品に関する基本情報が表示されます。このビューに入力した情報には、参照だけに使用する情報、Windows ロゴ要件を満たすために使用する情報、さらにプロジェクトの基本的な設定を構成するために使用する情報があります。

新しいインストール プロジェクトを作成する場合、プロジェクトの [一般情報] ビューの設定を構成する必要があります。InstallShield は、デフォルトの設定を使用して新しいプロジェクトを作成しますが、独自のニーズに合ったデータをプロジェクトに含めるためには、独自の値を設定することをお勧めします。

[一般情報] ビューは、次の要素で構成されます:

- ボタンの列
- 設定を表示するグリッド

次のテーブルは、[一般情報] ビューに表示されるボタンについて説明します。

テーブル 1 · [一般情報] ビューのコントロール

コントロールの名前	アイコン	説明
カテゴリー別		カテゴリごとに設定を並べ替えます。
アルファベット順		設定をアルファベット順に並べ替えます。

[一般情報] ビューの各設定についての説明は、「[一般情報] ビューの設定」を参照してください。

[一般情報] ビューの設定

プロジェクト: *[一般情報] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[一般情報] ビューの設定は、次のメイン カテゴリに分かれています:

- [全般] の設定
- 概要情報ストリーム
- プログラムの追加と削除
- ソフトウェア識別タグ

[全般] の設定

[一般情報] ビューの ″全般″ 領域では、製品名や製品バージョンなどの詳細を指定します。この領域には、以下の設定があります。

テーブル 2 ・ 全般設定

設定	プロジェクトの種類	説明
プロジェクトのファイル名	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI、MSIX	これは、InstallShield プロジェクト ファイルのパスとファイル名を表示する読み取り専用設定です。プロジェクトの種類も表示します。 プロジェクト ファイルの拡張子は、そのプロジェクトの種類によって異なります: • **.ism プロジェクト ファイル** — この拡張子は、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSIX、およびマージ モジュールを含む、殆どのプロジェクト タイプに使用されます。 • **.dim プロジェクト ファイル** — この拡張子は、DIM プロジェクトに使用されます。 • **.issuite プロジェクト ファイル** — この拡張子は、アドバンスト UI およびスイート/アドバンスト UI プロジェクトに使用されます。 詳細については、「プロジェクトの種類」を参照してください。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
プロジェクト ファイルの形式	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSIX、マージ モジュール	プロジェクト ファイル (.ism または .dim) に使用するファイル形式を選択します。選択可能なオプションは以下のとおりです: • **バイナリ** – ism ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。 これは、基本の MSI、DIM、InstallScript MSI、MSIX、およびマージ モジュール プロジェクトでデフォルトの形式です。これらのプロジェクト タイプでこのプロジェクト ファイル形式を選択すると、Windows Installer データベース エディターで .ism ファイルを更新できます。また、Windows Installer API またはそのオートメーション インターフェイスを使って、.ism ファイルを変更することもできます。 • **XML** – .ism ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。XML ツールを使用して、プロジェクト ファイルを更新できます。 これは、InstallScript プロジェクトと InstallScript オブジェクト プロジェクトのデフォルト形式です。 ***メモ:*** *どちらのプロジェクト ファイル形式でも、コマンドラインからリリースをビルドできます。*
スイート GUID	アドバンスト UI、スイート/アドバンスト UI	アドバンスト UI またはスイート/アドバンスト UI インストールを一意に識別する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ({...}) をクリックします。 このコードはアドバンスト UI またはスイート/アドバンスト UI インストールを一意的に識別するため、リリースを既に配布している場合はスイート GUID の変更はお薦めできません。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
セットアップ言語	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	 ***プロジェクト:*** *この設定の動作は、プロジェクトの種類によって異なります。* 基本の MSI、InstallScript MSI、マージ モジュール、およびスイート/アドバンスト UI プロジェクトでは、"セットアップ言語" 設定を使って、[リリース] ビューの "UI 言語" 設定にリストする言語を指定します。プロジェクト レベルでこの "セットアップ言語" 設定に言語がリストされていない場合、その特定の UI 言語をプロジェクトのリリースに含めることはできません。 アドバンスト UI プロジェクト (InstallShield の Professional Edition で提供されています) では、1 言語のみサポートされています。従って、このプロジェクトの "セットアップ言語" 設定は読み取り専用になっています。 InstallScript および InstallScript オブジェクト プロジェクトでは、"セットアップ言語" 設定を使って、プロジェクトの [コンポーネント] ビューと [リリース] ビューの "言語" 設定にリストする言語を指定します。一般的に、プロジェクト レベルでこの設定に言語がリストされていない場合、プロジェクト内の特定のコンポーネントをその言語に基づいてターゲットにすることはできません。さらに、特定の言語に基づいてコンポーネントと UI 文字列をプロジェクトのリリースに含めることもできません。 この設定を使ってサポート言語をアドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、またはスイート/アドバンスト UI プロジェクトに追加すると、プロジェクトにその言語の文字列エントリが追加されます。文字列エントリには、翻訳済みのビルトイン ユーザー インターフェイス文字列リソースが含まれます。 DIM プロジェクトの "セットアップ言語" 設定を使って、プロジェクトでサポートする言語を指定します。この設定を使ってプロジェクトに言語を追加すると、InstallShield によってその言語の文字列エントリがプロジェクトに追加されます。文字列エントリには、翻訳が必要な文字列リソースが含まれています。 詳細については、「インストール言語の選択」を参照してください。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
デフォルト言語:	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI、MSIX	[一般情報] ビューまたは [文字列エディター] ビューで構成されたデフォルト プロジェクト言語をオーバーライドするには、選択されたリリースに適切なデフォルト ユーザー インターフェイス言語を指定します。 詳細については、「デフォルトのプロジェクト言語の設定」を参照してください。
フィルター処理	InstallScript、InstallScript オブジェクト	この設定では、プロジェクトでコンポーネントまたはリリースに行う処理を選択するときの選択肢に加えるプラットフォームを指定します。一般的に、プロジェクト レベルのこの設定で表示されないプラットフォームがあるとき、プロジェクト内にある特定のコンポーネントまたはリリースをこのプラットフォームにターゲットするという指定ができなくなります。 プロジェクトのプラットフォームを変更するには、この設定の省略記号ボタン (...) をクリックします。[プラットフォーム] ダイアログ ボックスが開き、ここでプロジェクトに適切なプラットフォームを選択できます。 ***メモ:*** *プロジェクト レベルでプラットフォームを指定しても、インストールを実行するターゲット は作成されません。InstallScript、または InstallScript オブジェクト プロジェクトでターゲットを作成する場合、SYSINFO 構造を使用して、ターゲット システムのオペレーティング プラットフォームを識別します。*

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
メンテナンス エクスペリエンス	InstallScript	製品が既にターゲットマシンにインストールされている状態でエンドユーザーがインストールを再実行をしたときの動作を選択します。有効なオプションは次のとおりです: • **標準** – 製品が既にターゲットマシンにインストールされている状態で、エンドユーザーがインストールを再実行をしたときにメンテナンス ユーザー インターフェイスを表示する場合、このオプションを選択します。[プログラムの追加と削除] では、製品に対してエントリがひとつだけリストされます。 • **複数インスタンス** – エンドユーザーがインストールを複数回、メンテナンスインストールではなく初回インストールとして再実行できるようにする場合、このオプションを選択します。エンドユーザーがインストールを実行するごとに、別のエントリが [プログラムの追加と削除] に追加されます。このオプションを選択すると、デフォルトで、エンドユーザーがインストール実行ごとに異なる場所に製品をインストールすることができるようになります。これは、エンドユーザーが製品を異なる場所にインストールして異なる製品構成で製品を試してみてから、後で特定のインスタンスのみアンインストールしたい場合、便利です。メンテナンス ユーザー インターフェイスは、エンドユーザーが [プログラムの追加と削除] からインストールを再実行するときのみ表示されます。 • **アンインストールまたはメンテナンスをしない** – エンドユーザーが製品をアンインストールしたり、メンテナンス モードで実行できないようにする場合、このオプションを選択します。[プログラムの追加と削除] で、製品に対するエントリは作成されません。 詳細については、「InstallScript インストールを複数回実行する」を参照してください。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
メンテナンスの有効化	InstallScript MSI	製品が既にインストールされているシステム上で、エンド ユーザーがインストールを再度実行したときに、完全なメンテナンス ユーザー インターフェイス (UI) またはアンインストール UI を表示するかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい** — /removeonly コマンドライン パラメーターが Setup.exe に渡されない限り、エンド ユーザーがインストールを再度実行したとき、システム変数 REMOVEONLY は FALSE に設定され、標準メンテナンス UI が表示されます。 • **いいえ** — エンド ユーザーがインストールを再度実行したとき、システム変数 REMOVEONLY は TRUE に設定され、アンインストール UI が表示されます。 詳細については、「"メンテナンスの有効化" 設定を構成する」を参照してください。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
InstallScript ユーザー インターフェイスの種類	InstallScript MSI	インストールに使用する InstallScript ユーザー インターフェイス (UI) の種類を指定します。有効なオプションは次のとおりです: • **従来のスタイル (Setup.exe が必要)** — InstallScript エンジンを InstallScript MSI インストールの 外部 UI ハンドラーとして使用する場合は、このオプションを選択します。このスタイルを使用する場合は、インストールに Setup.exe セットアップランチャーを必ず含めてください。セットアップランチャーはブートストラップ アプリケーションとして機能し、UI を表示する InstallScript エンジンを開始して InstallScript コードを実行し、また Windows Installer を開始して .msi パッケージの [実行] シーケンスを実行します。 この設定では、このオプションがデフォルトで設定されています。 • **新しいスタイル (Windows Installer 4.5 が必要)** — InstallScript エンジンを InstallScript MSI インストールの埋め込み UI ハンドラーとして使用する場合は、このオプションを選択します。このスタイルでは、InstallShield が InstallScript エンジンを .msi パッケージ内に埋め込みます。Windows Installer は InstallScript エンジンを呼び出して UI を表示します。Windows Instaler はまた、.msi パッケージの [実行] シーケンスを実行します。 このオプションには、ターゲット マシン上に Windows Installer 4.5 が必要です。また、いくつかの制限事項があるため、このスタイルを使用する場合は綿密な計画が必要です。 2 つのスタイルに関する詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
製品名	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、MSI データベース、MSM データベース、スイート/アドバンスト UI、トランスフォーム	製品、DIM、マージ モジュールまたはオブジェクトの名前を入力します。 製品名がどのように使用されるかについては、「製品名の指定」を参照してください。 ***プロジェクト:*** *InstallScript および InstallScript オブジェクト プロジェクトの場合、値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)* *製品名は、InstallScript システム変数 IFX_PRODUCT_NAME に格納されます。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトの異なるバージョンを内部的に識別するためのリファレンスとして使用できます。*

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
製品バージョン	アドバンスト UI、基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール、MSI データベース、MSM データベース、スイート/アドバンスト UI、トランスフォーム	製品のバージョン番号を入力します。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。詳細については、「製品バージョンを指定する」を参照してください。 ***プロジェクト:*** *基本の MSI、InstallScript、および InstallScript MSI プロジェクトの場合 – リリースに* Setup.exe *が含まれる場合、指定した製品バージョンが* Setup.exe *の [プロパティ] ダイアログ ボックスに表示されます。詳細については、「セットアップ プランチャーのファイルのプロパティをカスタマイズする」を参照してください。* *InstallScript および InstallScript オブジェクト プロジェクトの場合、値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトの異なるバージョンを内部的に識別するためのリファレンスとして使用できます。入力された値は、この DIM を含むプロジェクトの [DIM リファレンス] ビューに表示されます。*
製品コード	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、トランスフォーム	この製品を固有に識別する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 このコードは製品を一意的に識別するため、リリースを既に配布している場合は製品コードの変更はお薦めできません。 詳細については、「Windows Installer ベースのプロジェクトで製品コードを設定する」または「InstallScript ベースのプロジェクトで製品コードを設定する」を参照してください。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
アップグレード コード	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	製品のアップグレード コードに使用する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 アップグレード コードとは、製品が所属する製品ファミリを識別する一意な GUID です。アップグレード コードは、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があり、これを使って、Windows Installer がインストール済みの製品の関連バージョンを検索します。 詳細については、「アップグレード コードを設定する」を参照してください。
インストール条件	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	この設定を使って、インストールを実行するためには True 評価が必要な 1 つ以上の条件を指定できます。たとえば、特定のオペレーティング システムや最低要件のテストを行うことができます。実行時に条件が True 評価されなかった場合、エラー メッセージが表示されて、製品がインストールされません。 1 つ以上の条件を指定するには、この設定で省略記号ボタン (...) をクリックします。詳細については、「製品の条件を設定する」を参照してください。 条件を追加すると、InstallShield によって "インストール条件" 設定の下に 2 つの新しい設定が追加されます: • **条件** – この設定は、追加された条件ステートメントを表示します。条件ステートメントを編集するには、この設定で省略記号ボタン (...) をクリックします。条件とそのメッセージを削除するには、この設定で **[条件の削除]** ボタンをクリックします。 • **メッセージ** – この設定は、対応する条件がターゲット システムで満たされていない場合に、実行時に表示されるように構成されている、実行時エラー メッセージを表示します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
終了条件	アドバンスト UI、スイート/アドバンスト UI	この設定では、アドバンスト UI またはスイート/アドバンスト UI インストールがインストールの終了の前に、様々な条件に応じて表示する終了エラー メッセージを指定できます。たとえば、アドバンスト UI またはスイート/アドバンスト UI インストールに Windows Vista 以降が必要な場合、"終了条件" 設定とそのサブ設定を使って、エンドユーザーが、それ以前の Windows を使ってアドバンスト UI またはスイート/アドバンスト UI インストールを起動したときに表示するエラー メッセージを指定できます。エンドユーザーがエラー メッセージ ボックスを閉じると、インストールは終了します。 エラー メッセージおよび 1 つまたは複数の条件を指定するには、この設定で省略記号ボタン (...) をクリックします。
終了条件 (続き)		条件を追加すると、InstallShield によって "終了条件" 設定の下に 2 つの新しい設定が追加されます: - **終了メッセージ** – このメッセージで定義された条件が True となった時、実行時に表示するエラー メッセージを入力します。メッセージとその条件を削除するには、この設定で **[この条件を削除]** ボタンをクリックします。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 - **Any/All/None** – この設定は、定義を行う条件ステートメントを表示します。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する」を参照してください。
モジュール言語	DIM、マージ モジュール、MSM データベース	DIM またはマージ モジュールの言語を選択するか、[Language Independent (言語非依存)] を選択します。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
INSTALLDIR	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	Windows Installer プロパティ INSTALLDIR の値を指定します。この値は、実行時に製品、DIM、またはマージ モジュールのファイルのほとんどがインストールされるインストール先ディレクトリを示します。 パスをハードコード化する代わりに、パスの一部としてディレクトリ プロパティを入力することができます。ディレクトリ プロパティを選択するには、この設定で省略記号ボタン (...) をクリックします。ここで、適切なディレクトリをリストから選択するか、定義済みディレクトリ内に新しいディレクトリを作成できます。円記号を使用して、サブディレクトリの下位レベルを **[ProgramFilesFolder]MyApp¥Bin** のように区切ります。 ***Windows ロゴ •*** *Windows ロゴ プログラムに準拠するためには、製品のデフォルトのインストール先が Program Files フォルダーのサブフォルダー (**[ProgramFilesFolder]**) である必要があります。これは、システムのロケールやユーザー設定によって異なることがあります。* ***プロジェクト•*** *基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム プロジェクトでのデフォルト値は以下のとおりです:* [ProgramFilesFolder]*会社名¥製品名* *この設定に入力した値は* INSTALLDIR *プロパティに割り当てられ、これがすべての製品の機能とコンポーネントのデフォルトイ ンストール先フォルダーとなります。詳細については、「デフォルトの製品インストール先フォルダー (INSTALLDIR) の設定」を参照してください。* *DIM、マージ モジュールおよび MSM データベース プロジェクトでのデフォルト値は以下の通りです:* [TARGETDIR] *DIM、マージ モジュール、および MSM データベース プロジェクトで、この DIM またはマージ モジュールのユーザーがデフォルトのインストール先ディレクトリを上書きできるようにする場合、この設定をデフォルト値のままに残します。詳細については、「マージ モジュールのデフォルトのインストール先フォルダーを指定する」を参照してください。*

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
TARGETDIR	InstallScript、InstallScript オブジェクト	製品のメイン ターゲット ディレクトリのデフォルト値を指定します。通常、この値を次のような値に設定します。 <FOLDER_APPLICATIONS>¥<IFX_COMPANY_NAME>¥<IFX_PRODUCT_NAME> 詳細については、「TARGETDIR」を参照してください。
Module ID GUID	マージ モジュール、MSM データベース	この読み取り専用設定は、マージ モジュールを一意に識別する GUID を表示します。新しいマージ モジュールが作成されるたびに、InstallShield がその GUID を生成します。 モジュール ID GUID は、すべてのマージ モジュール GUID 外部キーに追加されます。たとえば、マージ モジュール プロジェクトでコンポーネントを作成すると、そのコンポーネントはダイレクト エディターと、ビルドされた .msm ファイルに **ComponentName.MergeModuleGUID** としてリストされます。 この GUID を変更するには、[ダイレクト エディター] ビューを使って、ModuleSignature テーブルの ModuleID フィールドを編集します。この値を変更する場合、プロジェクト内のすべてのテーブルの値も更新する必要があります。そのためには、[ダイレクト エディター] ビューを使用できます。
モジュールの依存関係	マージ モジュール、MSM データベース	この設定を使って、マージ モジュールに必要な 1 つ以上のマージ モジュールを指定できます。 再配布可能ファイル ギャラリーで提供されている 1 つ以上の依存関係を追加するには、この設定で省略記号ボタン (...) をクリックします。[モジュール依存関係] ダイアログ ボックスが開き、ここでマージ モジュールに必要なモジュールを選択できます。 マシンに存在しない依存関係を追加するには、**[新しいモジュール依存関係を追加する]** ボタンをクリックしてから、そのバージョン、言語、およびモジュール GUID を指定します。
名前	マージ モジュール、MSM データベース	これは、次の中から 1 つの値を表示する読み取り専用設定です。 - 再配布可能ファイル ギャラリーで提供されている依存関係を選択すると、この設定には、その依存関係の名前が表示されます。 - マシンに存在しない依存関係の追加を選択すると、この設定には **[セットアップ作成者によって追加されたマージモジュール]** と表示されます。 この設定は、プロジェクトにマージ モジュールの依存関係を追加した場合にのみ表示されます。 プロジェクトからこの依存関係を削除するには、この設定で **[このモジュール依存関係を削除する]** をクリックします。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
バージョン	マージ モジュール、MSM データベース	作成中のマージ モジュールに特定バージョンの依存関係が必要な場合、そのバージョン番号を指定します。依存関係のバージョンにこだわらない場合は、"バージョン" 設定を空白のままに残します。 この設定は、プロジェクトにマージ モジュールの依存関係を追加した場合にのみ表示されます。
言語	マージ モジュール、MSM データベース	作成中のマージ モジュールに特定言語の依存関係が必要な場合、そのバージョン言語を指定します。依存関係の言語にこだわらない場合は、[言語非依存] を選択します。 この設定は、プロジェクトにマージ モジュールの依存関係を追加した場合にのみ表示されます。
モジュール ID	マージ モジュール、MSM データベース	再配布可能ファイル ギャラリーで提供されている依存関係を選択すると、この設定には、その依存関係のモジュール ID が表示されます。 マシンに存在しない依存関係の追加を選択する場合、依存関係のモジュール ID を入力します。モジュール ID は、以下の形式でなくてはなりません: **ModuleName.ModuleGUID** たとえば、マージ モジュールの名前が **MyDependency** で、GUID が **{2560C1ED-E2F7-4FE6-A0E6-15A9DA4CE9B9}** の場合、この設定には次のように入力します。 **MyDependency.2560C1ED-E2F7-4FE6-A0E6-15A9DA4CE9B9** この設定は、プロジェクトにマージ モジュールの依存関係を追加した場合にのみ表示されます。
モジュールの除外	マージ モジュール、MSM データベース	この設定を使って、作成中のマージ モジュールと互換性を持たない 1 つ以上のマージ モジュールを指定できます。この設定は、たとえば以前のバージョンのマージ モジュールが、新しいモジュールと互換性を持たない場合に必要です。 再配布可能ファイル ギャラリーで提供されている 1 つ以上の除外モジュールを追加するには、この設定で省略記号ボタン (...) をクリックします。[モジュールの除外] ダイアログ ボックスが開き、ここでマージ モジュールと互換性を持たないモジュールを選択できます。 マシンに存在しないモジュールの除外を追加するには、**[新しいモジュールの除外を追加する]** ボタンをクリックしてから、そのバージョン、言語、およびモジュール GUID 要件を指定します。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
名前	マージ モジュール、MSM データベース	これは、次の中から 1 つの値を表示する読み取り専用設定です。 • 再配布可能ファイル ギャラリーで提供されている除外モジュールを選択すると、この設定には、その除外モジュールの名前が表示されます。 • マシンに存在しないモジュールの除外の追加を選択すると、この設定には、**[セットアップ作成者によって追加されたマージ モジュール]** と表示されます。 この設定は、マージ モジュールの除外をプロジェクトに追加した場合のみ表示されます。 この除外をプロジェクトから削除するには、この設定で **[このモジュールの除外を削除する]** ボタンをクリックします。
最大バージョン	マージ モジュール、MSM データベース	除外するマージ モジュールの最大バージョン番号を指定します。 この設定を空白のまま残すと、"最小バージョン" 設定で指定した値の後のすべてのバージョンが除外されます。"最大バージョン" と "最小バージョン" 設定の両方を空白に残すと、バージョンに基づく除外は行われません。 この設定は、マージ モジュールの除外をプロジェクトに追加した場合のみ表示されます。
最小バージョン	マージ モジュール、MSM データベース	除外するマージ モジュールの最小バージョン番号を指定します。 この設定を空白のまま残すと、"最大バージョン" 設定で指定した値の前のすべてのバージョンが除外されます。"最大バージョン" と "最小バージョン" 設定の両方を空白に残すと、バージョンに基づく除外は行われません。 この設定は、マージ モジュールの除外をプロジェクトに追加した場合のみ表示されます。
言語	マージ モジュール、MSM データベース	特定の言語を持つマージ モジュールを除外するか、この設定の現在の値を変更するには、この設定で省略記号ボタン (...) をクリックします。[除外言語] ダイアログ ボックスが開き、ここで特定の言語を除外するか、特定の言語以外の全ての言語を除外するかを指定します。このダイアログ ボックスでは、[言語非依存] を選択することもできます。これは、言語に基づいてマージ モジュールの除外を行わないことを示します。 この設定は、マージ モジュールの除外をプロジェクトに追加した場合のみ表示されます。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
モジュール ID	マージ モジュール、MSM データベース	再配布可能ファイル ギャラリーで提供されている除外モジュールを選択すると、この設定には、その除外のモジュール ID が表示されます。 マシンに存在しない除外モジュールの追加を選択する場合、除外モジュールの ID を入力します。モジュール ID は、以下の形式でなくてはなりません: **ModuleName.ModuleGUID** たとえば、マージ モジュールの名前が **MyExclusion** で、GUID が **{2560C1ED-E2F7-4FE6-A0E6-15A9DA4CE9B9}** の場合、この設定には次のように入力します。 **MyExclusion.2560C1ED-E2F7-4FE6-A0E6-15A9DA4CE9B9** この設定は、マージ モジュールの除外をプロジェクトに追加した場合のみ表示されます。
DIM GUID	DIM	この読み取り専用設定は、DIM を一意に識別する GUID を表示します。新しい DIM を作成すると常に、新しい GUID が自動生成されます。 DIM GUID は、すべての DIM GUID 外部キーに追加されます。たとえば、DIM プロジェクト内にコンポーネントを作成した場合、そのコンポーネントはダイレクト エディターで **ComponentName.DIM_GUID** と表示されます。 この GUID を変更するには、[ダイレクト エディター] ビューを使って、ModuleSignature テーブルの ModuleID フィールドを編集します。この値を変更する場合、プロジェクト内のすべてのテーブルの値も更新する必要があります。そのためには、[ダイレクト エディター] ビューを使用できます。
ビルドの手順	DIM	このプロジェクトに関して、この DIM をインストール プロジェクトに含める作業を行うリリース エンジニアの役に立つ特別な指示またはコメントを入力します。これらの手順は、この DIM を含むプロジェクトの [DIM リファレンス] ビューに表示されます。ビルドの手順は、エンド ユーザーには表示されません。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
ロックダウンの設定方法	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ロックダウン環境で製品を実行するエンド ユーザー向けに、ファイル、フォルダー、およびレジストリ キーを保護するためのアクセス許可の種類を選択します。選択可能なオプションは以下のとおりです: • **カスタム InstallShield 処理**— InstallShield は、プロジェクトに ISLockPermissions テーブルとカスタム アクションを追加して、ターゲット システム上のアクセス許可を設定します。このオプションがデフォルト値です。 • **従来型の Windows Installer 処理** — InstallShield は、.msi データベースの LockPermissions テーブルを使って、製品のアクセス許可情報を格納します。 多くの場合、従来型の Windows Installer 処理よりも、カスタム InstallShield 処理の方が有利です。例: • カスタム オプションを使うと、従来のオプションではサポートされていない、多くのよく知られているセキュリティ識別子 (SID) を使用できます。 • 従来型のオプションとは違い、カスタム オプションでは、多くのよく知られている SID に翻訳されたユーザー名を使用できます。従来型のオプションで、非英語システム上で翻訳された名前を使ってアクセス許可を設定すると、インストールが失敗する可能性があります。 • カスタム オプションを使うと、指定するアクセス許可を特定のユーザーまたはグループが所持することを拒否できます。従来型の処理で、これは不可能です。つまり、従来型の処理の場合、特定のアクセス許可を設定することのみが可能で、アクセス許可を拒否することはできません。 この機能はプロジェクト全体に反映される設定で、プロジェクトに含まれるファイル、フォルダー、およびレジストリ キーに設定する新しいアクセス許可すべてに適用します。プロジェクトで既にいくつかのアクセス許可を構成済みの場合にこの設定の値を変更する場合、InstallShield では、既存アクセス許可に別の処理方法を使用するかどうかを指定できます。 この設定についての詳細、およびアクセス許可の設定に関する InstallScript サポートについての情報は「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
[ユーザーごと] オプションの表示	基本の MSI	エンド ユーザーが製品をすべてのユーザー、または現在のユーザーのみにインストールするかを選択できるオプションを提供するかどうかを指定します。このランタイム オプションは Windows 7 以降のシステム、および Windows Server 2008 R2 以降のシステムのみで使用できます。選択可能なオプションは以下のとおりです: • **いいえ** – エンド ユーザーが製品をインストールする方法を指定できるボタンを表示しません。 • **はい** – ターゲット システムに Windows 7 または Windows Server 2008 R2 がインストールされている場合、InstallShield は ReadyToInstall ダイアログにボタンを追加します。これらのボタンを使って、エンド ユーザーは製品をインストールする方法を指定できます。昇格された権限が必要な場合、[すべてのユーザー] ボタンにシールド アイコンが含まれます。エンド ユーザーが [ユーザーごと] ボタンを選択した場合、ALLUSERS プロパティが 2 に、MSIINSTALLPERUSER プロパティが 1 に設定されます。エンド ユーザーが [すべてのユーザー] ボタンを選択した場合、ALLUSERS プロパティが 1 に設定され、MSIINSTALLPERUSER プロパティは設定されません。 ターゲット システムが Windows Vista 以前、または Windows Server 2008 以前である場合、ReadyToInstall ダイアログに、ユーザーが製品をインストールする方法を選択できるボタンは表示されません。 デフォルト値は [いいえ] です。 ***メモ:*** *[いいえ] を選択しても、エンド ユーザーは、インストールの実行時にコマンドラインから* MSIINSTALLPERUSER *を設定することができます。インストールがこれをサポートしない場合、起動条件を追加するか、その他の実行時のチェックを行って、これを回避することができます。* 詳細については、「ユーザーごとのインストールとマシンごとのインストールの違い」を参照してください。

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
MSI ログ記録の作成	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	Windows Installer 4.0 以降でインストールをログ記録するかどうかを指定するには、この設定で省略記号ボタン (...) をクリックして、[Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックスを起動します。このダイアログ ボックスで Windows Installer がインストールのログ記録を行うかどうかを指定します。このダイアログ ボックスを使用して、ログ記録されるメッセージの種類をカスタマイズすることもできます。 この設定には 3 種類の値から選択することができます。 • **いいえ** – インストールはログ記録されません。これがデフォルトの値です。 • **はい**–**voicewarmup** のデフォルト値が MsiLogging プロパティに挿入されます。 • **カスタム** – [Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックスで指定した値が MsiLogging プロパティに挿入されます。 この設定の値が [はい] または [カスタム] の場合に、インストールが Windows Vista 以降または Windows Server 2008 以降において、Windows Installer 4.0 以降を使って実行されたとき、以下の処理が行われます: • インストーラーが、適切なログ記録モード (″MSI ログ記録の作成″ 値が [はい] の場合は **voicewarmup**、または [Windows Installer 4.0 以降のログ記録オプション] ダイアログ ボックスで指定した任意のカスタム値) に従って、ログ ファイルを作成します。 • インストーラーが、MsiLogFileLocation プロパティに、ログ ファイルのパスを挿入します。 • SetupCompleteSuccess、SetupCompleteError および SetupInterrupted ダイアログに **[Windows Installer ログを表示]** チェック ボックスが追加されます。エンドユーザーがこのチェック ボックスを選択してから [終了] をクリックすると、テキスト ファイル ビューアーまたはエディターでログ ファイルが開きます。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
MSI ログ記録の作成 (続き)		"MSI ログの作成" 設定は、Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で Windows Installer 4.0 以降を使って実行するインストールに適用します。Windows Installer の古いバージョンを実行する以前のシステム上では、実行時ダイアログで **[Windows Installer ログを表示する]** チェック ボックスは表示されません。 ***メモ:*** *この設定の値が "カスタム" で、これを [はい] に変更した場合、プロパティ マネージャーの* MsiLogging *プロパティで定義したカスタム パラメーターは、デフォルト値で上書きされます。[いいえ] に変更した場合、InstallShield は* MsiLogging *プロパティからすべてのカスタム パラメーターを削除します。* ログ記録されるメッセージの種類をカスタマイズする方法についてなど、詳しい情報は「Windows Installer インストールをログ記録するかどうかを指定する」を参照してください。
高速インストール	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	サイズが大きい Windows Installer パッケージをインストールするのに要する時間を短縮するには、次のオプションから 1 つ以上選択することを考慮してください: • このインストールでシステム復元ポイントを保存しない • ファイル コスティングのみを実行して、その他のコスト チェックをスキップする • 進行状況メッセージの頻度を減らす この設定は、Windows Installer プロパティ MSIFASTINSTALL を構成します。このプロパティは、コマンドラインで設定可能です。 Windows Installer 5 で、この設定がサポートされています。以前のバージョンの Windows Installer はこれを無視します。
会社名	InstallScript、InstallScript オブジェクト	会社の名前を入力します。この値は、(文字列 エントリ、COMPANY_NAME が存在しない場合) デフォルト スクリプトで TARGETDIR を設定するのに使用されます。これは、実行時に **MediaGetData** 関数を呼び出して取得することができます。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)*

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
実行可能ファイル	InstallScript、InstallScript オブジェクト	アプリケーションのメインの実行可能ファイルの名前を入力します。この値は、**MediaGetData** 関数を呼び出して実行時に取得できます。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)*
URL	InstallScript、InstallScript オブジェクト	製品 URL を入力します。この情報は、プロジェクト ファイルに格納され、参照目的でのみ使用できます。エンド ユーザーに表示されることはありません。 ***ヒント:*** *値をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。(パス変数を使用するには、[プロジェクト] メニューから [設定] をクリックします。次に、[アプリケーション] タブで適切なパス変数を選択します。)*

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
サーバーの場所	トランスフォーム	製品のインストール パッケージとその関連ファイルをネットワーク サーバーまたは Web サイトに保存する場合、この設定の省略記号ボタン (...) をクリックして、サーバーまたは Web サイトの場所を指定します。その場所には .msi パッケージ、および製品の修復や機能のアドバタイズなどに必要な全てのファイルを配置します。 指定したサーバーの場所が有効かどうかは、インストールがサーバーにリモート アクセスする必要があるときに判断されます。つまり、リソースが必要なとき、サーバーが使用できない、または無効なサーバーが追加された場合、エントリが無視され、場合によってはエラーが生成されます。指定された各場所には、インストールの完全なソースが配置されなくてはなりません。各ソースの場所のディレクトリ ツリーはすべて同じであり、.cab ファイルを含み必要なソース ファイルすべてを含まなくてはなりません。各場所には、同じファイル名と製品コードを持つ .msi ファイルが必要です。 ***ヒント:*** *サーバー パスには、パーセント記号 (%) で識別される環境変数を使用できます。たとえば、次のパスは Home 環境変数を利用します:* ¥¥Server1¥%Home%¥Office 指定するパスは SOURCELIST プロパティに格納されています。実行時に Windows Installer は、エンド ユーザーの既存の製品のソース リストの最後に、このリストを追加します。
オブジェクトの登録	InstallScript オブジェクト	リリースをビルドするときに、InstallScript オブジェクトをマシンに登録するかどうかを指定します。開発マシンにオブジェクトを再登録すると、そのマシンで作成する InstallScript プロジェクトにそのオブジェクトを含めることができます。 ***ヒント:*** *ビルド時にオブジェクトが登録されていない場合、InstallScript プロジェクト内部からこれを登録することができます。詳細については、「InstallScript プロジェクトでオブジェクトを登録する」を参照してください。*

テーブル 2 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
オブジェクト ウィザード	InstallScript オブジェクト	オブジェクトのデフォルト言語で使用するウィザードがある場合、その種類を選択します。選択可能なオプションは以下のとおりです: • **ウィザードなし** – オブジェクトにウィザードが不要な場合は、このオプションを選択します。 • **InstallShield オブジェクト ストック ウィザードを使用する** – オブジェクトに作成されたプロパティに基づいてウィザードを作成するには、このオプションを選択します。ストック ウィザードを使うと、すべての読み取り専用プロパティが表示されますが、編集することはできません。すべての読み取り/書き込みプロパティは、ユーザーによる変更が可能です。書き込みのみプロパティは、表示されません。 その代わりに、この設定で省略記号ボタン (...) をクリックして、オブジェクトに作成したカスタム ウィザードを選択することもできます。 カスタム ウィザードの作成方法、および InstallShield ストック ウィザードの使用方法については、「オブジェクトのウィザードの設計」を参照してください。
カスタム ウィザードの登録	InstallScript オブジェクト	オブジェクトのリリースをビルドすると、InstallShield によって、このオブジェクトのカスタム オブジェクト ウィザードが登録されるようにするかどうかを指定します。 ***メモ:*** *この設定は、"オブジェクト ウィザード" 設定で [カスタム ウィザード] を指定する場合のみ適用します。*
フォント登録	InstallScript、InstallScript オブジェクト	実行時に、プロジェクト内のすべてのフォント ファイルをターゲット システムに登録するには、[有効] を選択します。スタティック ファイル リンクには、フォント登録の各設定も含まれます。この設定は、ダイナミックにリンクされたフォント ファイルの動作を設定するのに使用しなくてはなりません。
DIFx サポート (32 ビットのプラットフォーム用)	InstallScript、InstallScript オブジェクト	プロジェクトで、32 ビット システム上でデバイス ドライバーをインストールするための DIFx サポートを有効にするには、[有効] を選択します。[有効] を選択すると、リリースをビルドするときに InstallShield がプロジェクトに DIFxAPI ライブラリを追加します。 詳細については、「デバイス ドライバーのインストール」を参照してください。

テーブル 2 - 全般設定 (続き.)

設定	プロジェクトの種類	説明
DIFx サポート (64 ビット Itanium プラットフォーム用)	InstallScript、InstallScript オブジェクト	プロジェクトで、64 ビット Itanium システム上でデバイス ドライバーをインストールするための DIFx サポートを有効にするには、[有効] を選択します。[有効] を選択すると、リリースをビルドするときに InstallShield がプロジェクトに DIFxAPI ライブラリを追加します。 詳細については、「デバイス ドライバーのインストール」を参照してください。
DIFx サポート (64 ビット AMD プラットフォーム用)	InstallScript、InstallScript オブジェクト	プロジェクトで、64 ビット AMD システム上でデバイス ドライバーをインストールするための DIFx サポートを有効にするには、[有効] を選択します。[有効] を選択すると、リリースをビルドするときに InstallShield がプロジェクトに DIFxAPI ライブラリを追加します。 詳細については、「デバイス ドライバーのインストール」を参照してください。

概要情報ストリーム

Windows Installer データベースは COM 構造化ストレージとして実装され、通常、COM 構造化ストレージ ファイルには Summary Information Stream が含まれています。Summary Information Stream には、会社や、インストール中のソフトウェアに関する情報が含まれます。

[一般情報] ビューの [概要情報ストリーム] 領域には、次の設定があります。

テーブル 3 • 概要情報ストリームの設定

設定	プロジェクトの種類	説明
タイトル	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	作成するデータベースの種類を指定します。製品インストールの場合、デフォルト値は **インストール データベース** です。これが推奨される値です。 ***プロジェクト*** *基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクト — 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。* この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
サブジェクト	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	製品の名前を入力します。 ***プロジェクト*** *基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 — 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。入力された値は、この DIM を含むプロジェクトの [DIM リファレンス] ビューに表示されます。* この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 3 • 概要情報ストリームの設定 (続き.)

設定	プロジェクトの種類	説明
作成者	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	会社名を指定します。 ***プロジェクト:*** *基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 – 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。入力された値は、この DIM を含むプロジェクトの [DIM リファレンス] ビューに表示されます。* この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
キーワード	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	製品の Windows Installer データベースを説明するキーワードを指定します。 ***プロジェクト:*** *基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 – 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。*

テーブル 3 • 概要情報ストリームの設定 (続き.)

設定	プロジェクトの種類	説明
パッケージ コード	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、パッケージ コードを識別します。これは、インストール パッケージ、DIM、またはマージ モジュールの GUID です。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 ***プロジェクト:*** *基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、および MSM データベース プロジェクトの場合 — Windows Installer では、同じパッケージコードを持つ 2 つの .msi データベースは同一の内容を持っていなければならないため、パッケージを変更した場合、リリース前にパッケージコードを変更する必要があります。[リリース] ビューで、製品構成の "パッケージ コードの生成" 設定を使って、リリースをビルドする度に InstallShield が自動的に新しいパッケージ コードを生成するかどうかを指定できます。この設定のデフォルト値は [はい] です。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。入力された値は、この DIM を含むプロジェクトの [DIM リファレンス] ビューに表示されます。*

テーブル 3 • 概要情報ストリームの設定 (続き.)

設定	プロジェクトの種類	説明
テンプレートの概要	基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	インストールでサポートするプロセッサの種類とデフォルトの言語を指定することができます。最初にプロセッサの種類、次にインストールのデフォルト言語をセミコロンで区切って入力します。言語カテゴリーに複数のエントリがある場合は、コンマで区切ります。 たとえば、インストールが Intel プロセッサを搭載した英語版システム上でのみ動作する場合は、**Intel;1033** と入力します。製品が x64 プロセッサを搭載した英語およびドイツ語システム上で実行する場合は、**x64;1033,1031** と入力します。インストールが言語に左右されない場合、この設定の言語部分には、数値 0 を使用します。 有効なプロセッサ値は、次のとおりです: • Intel • Intel64 • x64 指定できるプロセッサ値は 1 つだけですので、ご注意ください。 詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。 ターゲット マシンが、この設定で指定された要件を満たさない場合、エラー メッセージが表示され、インストールが終了します。

テーブル 3 • 概要情報ストリームの設定 (続き.)

設定	プロジェクトの種類	説明
[概要情報ストリーム] のコメント	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	製品に関する任意のコメントを入力します。この設定の通常の値は、次の通りです: **このインストーラー データベースには、MyProduct をインストールするために必要なロジックとデータが含まれています。** ***プロジェクト:*** *基本の MSI、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 – 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。* *DIM プロジェクトの場合 この設定の値は実行時に使用または表示されません。DIM プロジェクトのこの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。* この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 3 - 概要情報ストリームの設定 (続き.)

設定	プロジェクトの種類	説明
スキーマ	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定を使って、インストール パッケージ または DIM に必要な Windows Installer の最小バージョンを識別する整数を指定します。Windows Installer 2.0 を最小バージョンとする場合、**200** と入力します。Windows Installer 3.0 を最小バージョンとする場合、**300** と入力します。Windows Installer 3.1 を最小バージョンとする場合、**301** と入力します。Windows Installer 4.5 を最小バージョンとする場合、**405** と入力します。 エンド ユーザーのシステム上にある Windows Installer のバージョンが、"スキーマ" 設定で指定された最小要件よりも古い場合 (たとえば、インストールが Windows Installer 4.5 の機能を使用するため、スキーマの値が 405 に設定されているが、エンド ユーザーの Windows Installer バージョンが 3.1 の場合)、インストール時にエラーメッセージが表示され、インストールは途中で終了します。 "スキーマ" 設定で入力した値は、Windows Installer データベースの Page Count Summary プロパティに使用されます。 ***メモ:*** *[リリース] ビューにある製品構成の "スキーマ" 設定を設定することで、プロジェクト内の特定の製品構成の関連付けられているすべてのリリースについて、この値をオーバーライドすることができます。*

テーブル 3 • 概要情報ストリームの設定 (続き.)

設定	プロジェクトの種類	説明
管理者特権が必要	基本の MSI、MSI データベース	.msi パッケージが、インストールの実行シーケンスに、管理者権限が必要がどうかを指定します。デフォルトの設定は [はい] です。 [いいえ] を設定した場合、InstallShield は Word Count Summary プロパティ の 3 ビット目を設定して、.msi パッケージのインストールに昇格された権限が必要であることを示します。[いいえ] が選択された状態で、適切な権限をもたずに .msi パッケージがタスクを実行しようとすると、Windows Installer によって実行時エラーが表示される場合があります。 この設定は、Windows Vista 以降のシステムまたは Windows Server 2008 以降のシステム上で Windows Installer 4.0 以降を使って実行するインストールに適用します。以前のバージョンの Windows Installer と Windows は、この設定を無視します。 エンド ユーザーのインストール エクスペリエンスは、インストールが必要とする権限によってのみ実行された場合、安全性がより一層向上します。アプリケーションは、システム管理者のみによる実行が必須の場合を除き、最も低い権限で実行されることが理想的です。 この設定と他の InstallShield 設定が、権限昇格のために Windows Vista 以降で表示されるユーザー アカウント制御プロンプトに与える影響について学ぶには、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

プログラムの追加と削除

コントロール パネルの [プログラムの追加と削除] (最新バージョンの Windows では [プログラム]) には、サポートへのリンク、連絡先の電話番号、製品のアップデート情報、および製品の製造元に関する情報をエンド ユーザーに提供できます。インストールの構成の仕方によって、エンド ユーザーはボタンをクリックしてインストールの削除、修復、変更を選択することができます。プロジェクトでこの情報を指定するには、[一般情報] ビューにある [プログラムの追加と削除] の設定を構成します。

プロジェクト: *基本の MSI プロジェクトの場合—製品が [プログラムの追加と削除] に表示されないようにするには、プロパティ マネージャーで Windows Installer プロパティ* ARPSYSTEMCOMPONENT *を 1 に設定します。このプロパティを設定すると、[プログラムの追加と削除] に製品を表示しないように抑制するだけです。エンド ユーザーは、コマンド ラインを使ってインストールをメンテナンス モードで実行することで、製品を削除することができます。*

InstallScript MSI プロジェクトの場合—製品が [プログラムの追加と削除] に表示されないようにするには、[リリース] ビューの "[追加/削除] パネルエントリを隠す" 設定で [はい] を選択します。

テーブル 4 · [プログラムの追加と削除] の設定

設定	プロジェクトの種類	説明
[プログラムの追加と削除] エントリの表示	アドバンスト UI、スイート/アドバンスト UI	コントロール パネルの [プログラムの追加と削除] でアドバンスト UI またはスイート/アドバンスト UI プロジェクトのエントリを表示するかどうかを示します。選択可能なオプションは以下のとおりです: • **はい**—アドバンスト UI またはスイート/アドバンスト UI 製品のエントリが、ターゲット システムの [プログラムの追加と削除] で表示されます。これがデフォルトの値です。 • **いいえ**—アドバンスト UI またはスイート/アドバンスト UI 製品のエントリは、ターゲット システムの [プログラムの追加と削除] で表示されません。エンドユーザーが [プログラムの追加と削除] を使って、製品の削除やメンテナンスを行ったり、サポート情報を表示できなくなります。このオプションを選択した場合、InstallShield はその他の "プログラムの追加と削除" 設定を無効にします。

テーブル 4 - [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
アイコンの表示	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	 ***プロジェクト:*** *この設定に入力する値は、使用するプロジェクトの種類によって異なります。* **基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合:** [プログラムの追加と削除] で製品のエントリに使用するアイコン リソースを含む .ico、.exe、または .dll ファイルへの、開発システム上でのパスを入力します。パスとファイル名を手作業で入力する代わりに、この設定の省略記号ボタン (...) をクリックして、ファイルを参照できます。 **InstallScript および InstallScript オブジェクト プロジェクトの場合:** [プログラムの追加と削除] で製品のエントリに使用するアイコン リソースを含む .ico または .exe への、ターゲット システム上でのパスを入力します。山かっこで囲まれたシステム変数への相対パスを指定することもできます。例: <TARGETDIR>¥icon.ico **アドバンスト UI またはスイート/アドバンスト UI プロジェクトの場合:** [プログラムの追加と削除] で製品のエントリに使用するアイコン リソースを含む .ico、.exe、または .dll ファイルへの、開発システム上でのパスを入力します。パスとファイル名を手作業で入力する代わりに、この設定の省略記号ボタン (...) をクリックして、ファイルを参照できます。ターゲット システム上に存在するファイルに含まれるアイコンを使用する場合 (たとえば、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの 1 つによってインストールされるファイルに含まれるアイコンを使用する場合)、この設定でリストからディレクトリ プロパティを選択してから、アイコン ファイルの残りのパスを入力します。この設定を空白のままにすると、[リリース] ビューの Setup.exe タブにある "Setup.exe アイコン ファイル" 設定で指定されたアイコンが使用されます。
アイコン インデックスの表示	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、スイート/アドバンスト UI	指定したアイコン ファイルに 1 つ以上のアイコン リソースがある場合、この設定にインデックスを指定します。 - 負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば、0 はファイル内の最初のアイコン、1 は 2 番目のアイコン、2 は 3 番目のアイコンを参照します。 - 負の数字は特定のリソース ID を参照するために使用します。たとえばアイコン インデックス -12 は、リソース ID が 12 のアイコンを示します。

テーブル 4 · [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
変更ボタンの無効	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	[プログラムの追加と削除] で、製品の [変更] ボタンを無効にするかどうかを指定します。[変更] ボタンは、エンドユーザーが製品のインストール後、インストール オプションを変更できるボタンです。エン ドユーザーは必要に応じて機能を追加または削除できます。 ***プロジェクト·*** *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。* *"変更ボタンを無効にする" および "削除ボタンを無効にする" 設定を [はい] に設定し、スクリプトで* **MaintenanceStart** *関数が呼び出される前にシステム変数 ADDREMOVE_COMBINEDBUTTON を TRUE に設定して、組み合わせた [変更/削除] ボタンを指定することができます。***MaintenanceStart** *は、***OnMoveData** *イベント ハンドラーのデフォルト コードで呼び出されます。)*
削除ボタンの無効	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	[プログラムの追加と削除] で、製品の [削除] ボタンを無効にするかどうかを指定します。[削除] ボタンを使用すると、エンドユーザーは 1 つのボタンをクリックするだけで製品を削除することができます。この場合、アンインストーラーはコンパクトなユーザー インターフェイスで実行します。 ***プロジェクト·*** *基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合 エンド ユーザーが [削除] ボタンをクリックして製品を削除すると、プロジェクトの [ユーザー インターフェイス] シーケンス内のアクションが実行されます。* *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*
修復ボタンの無効	基本の MSI、MSI データベース、トランスフォーム	[プログラムの追加と削除] で、製品の [修復] ボタンを無効にするかどうかを指定します。[修復] ボタンを使うと、ファイルが削除されていた、または壊れていた場合、エンドユーザーは Windows Installer の修復オプションを実行することができます。

テーブル 4 ・ [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
発行者	アドバンスト UI、基本の MSI、InstallScript MSI、MSI データベース、スイート/アドバンスト UI、トランスフォーム	製品を作成した会社の名前を指定します。この情報は、[プログラムの追加と削除] で製品のエントリに表示されます。入力した値は、Windows Installer Manufacturer プロパティに格納されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***Windows ロゴ ・*** *Windows ロゴ プログラムの要件に準拠するには、発行元の指定が必要です。*
発行元/製品 URL	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	会社または製品の一般的な URL を入力してください (例、**http://www.installshield.com**)。 Windows の一部のバージョンでは、[サポート情報] ダイアログ ボックスに表示される発行元の名前は、この URL へのハイパーリンクです。[サポート情報] ダイアログ ボックスは、エンド ユーザーが [プログラムの追加と削除] で製品のエントリ用のサポート情報ハイパーリンクをクリックすると表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト・*** *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*

テーブル 4 - [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
サポート連絡先	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、トランスフォーム	エンドユーザーのテクニカルサポート窓口となる担当者または部門の名前を入力します。 Windows の一部のバージョンでは、この情報は、[プログラムの追加と削除] で製品のエントリ用の [サポート情報] ダイアログボックスで表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト*** *InstallScript プロジェクトの場合、**OnMoveData** イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*
サポート URL	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、トランスフォーム	エンド ユーザーが製品のテクニカル サポート情報を参照できる URL を入力します。この URL は、[プログラムの追加と削除] で製品のエントリに表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***Windows ロゴ*** *- Windows ロゴ プログラムの要件に準拠するには、有効な URL の入力が必要です。* ***プロジェクト*** *InstallScript プロジェクトの場合、**OnMoveData** イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*

テーブル 4 - [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
サポート電話番号	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	製品のテクニカル サポートの電話番号を入力します。 Windows の一部のバージョンでは、この情報は、[プログラムの追加と削除] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*
README	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	製品の Readme ファイルの名前、またはパスを入力します。その代わりに、有効な URL を指定してインターネット上の Readme ファイルのリンクを表示することもできます。詳細については、「Readme ファイルを指定する」を参照してください。 Windows の一部のバージョンでは、この情報は、[プログラムの追加と削除] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*

テーブル 4 · [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
製品アップデート URL	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	エンド ユーザーが製品のアップデート情報を参照、または最新バージョンをダウンロードできる URL を指定します。 Windows の一部のバージョンでは、この情報は、[プログラムの追加と削除] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*
[プログラムの追加と削除] のコメント	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	製品についてのコメントを入力します。この情報は、[プログラムの追加と削除] で製品のエントリに表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *InstallScript プロジェクトの場合、***OnMoveData** *イベント ハンドラーは、この設定のデータをターゲット システムのレジストリに書き込みます。*
英語または日本語 IDE の設定	InstallScript オブジェクト	[英語 IDE 設定]、および [日本語 IDE 設定] 領域では、InstallShield ユーザーが InstallShield の英語版または日本語版で、[オブジェクト] ビューに表示されたオブジェクトを参照したときに表示される情報を指定できます。

テーブル 4 • [プログラムの追加と削除] の設定 (続き.)

設定	プロジェクトの種類	説明
デフォルトを使用	InstallScript オブジェクト	ここでデフォルト言語以外の言語に設定するには、この言語にデフォルト言語設定の値を使用するかどうかを指定します。[はい] を選択すると、″デフォルトを使用する″ 設定の下にある他の設定が無効になります。[いいえ] を選択すると、″デフォルトを使用する″ 設定の下にある他の設定が有効になり、言語固有の値を指定することができます。 この設定はデフォルトの言語には使えません。
表示名	InstallScript オブジェクト	[オブジェクト] ビューに表示するオブジェクトの名前を入力します。
短い名前	InstallScript オブジェクト	必要に応じて、オブジェクト名の省略形を入力します。″表示名″ 設定に入力した名前と同じ名前を入力できます。
HTML ヘルプ	InstallScript オブジェクト	単一の Web ファイルで構成されるオブジェクトのヘルプ ファイルを指定します。このファイルは、[オブジェクト] ビューでオブジェクトが選択された時に右側のペインに表示されます。明示的なパス、またはパス変数を入力します。
アイコン ファイル	InstallScript オブジェクト	アイコンは、[オブジェク] ビューのオブジェクトの横に表示されます。アイコンは 16 ピクセル X 16 ピクセルで、最大 16 色まで使用できます。アイコンは DLL、.exe または .ico ファイルからロードできます。ファイルが DLL または .exe の場合、アイコンインデックスまたはリソース ID を、パスとファイル名の後にコンマで区切って指定できます。インデックスではなくリソース ID を指定する場合は、前にダッシュ (-) をつける必要があります。インデックスが指定されない場合、数値 0 とみなされます。

ソフトウェア識別タグ

[一般情報] ビューの [ソフトウェア識別タグ] 領域を使って、インストールに ISO/IEC 19770-2 ソフトウェア識別タグを含めるかどうかを指定できます。タグが含まれている場合、この領域で、[一般情報] ビューの他の領域でまで指定されていない識別情報を指定することもできます。詳細については、「製品のソフトウェア識別タグを含める」を参照してください。

ソフトウェア識別タグを含める場合、InstallShield で、Microsoft System Center Configuration Manager のアプリケーション モデル データをそのタグに追加することもできます。詳細については、「製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める」を参照してください。

テーブル 5 ・ソフトウェア識別タグの設定

設定	プロジェクトの種類	説明
ソフトウェア識別タグを使用	基本の MSI	インストールに、ISO/IEC 19770-2 ソフトウェア識別タグを含めるかどうか指定します。[はい] を選択した場合、[一般情報] ビューのこの領域にある、他のタグ関係の設定を使って、[一般情報] ビューの他の領域でまで指定されていない識別情報を指定することができます。
ソフトウェア エンタイトルメントが必須である	基本の MSI	ソフトウェアが正しく運用されていることを判断するために、製品に対応するソフトウェア エンタイトルメントがあることを条件とするかどうかを指定します。一般的に、有料ソフトウェアの場合、この設定には [はい] を選択し、無償ソフトウェアである場合には、この設定に [いいえ] を選択します。
一意な ID	基本の MSI	この特定の製品の特定のバージョンを識別する一意の ID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン (...) をクリックします。 InstallShield では、タグ ファイルの名前の一部として入力した値 (*TagCreatorID_UniqueID*.swidtag) が使用されますので注意してください。したがって、入力する ID に、ファイル名には使えない文字は使用できません。
タグ作成者	基本の MSI	タグを作成した組織の名前を指定します。

テーブル 5・ソフトウェア識別タグの設定 (続き.)

設定	プロジェクトの種類	説明
タグ作成者 ID	基本の MSI	タグを作成した組織の登録 ID を入力します。この ID は、同じ作成者名を持ち、かつ、異なる国に属する法的組織が複数あるとき、それらを区別するために有益です。 登録 ID の表示規則は、次のとおりです: **regid.YYYY-MM.ReversedDomainName,division** 登録 ID ビューは、次の部分から構成されます: ・ regid.—文字列 *regid* は、XML 部分がソフトウェア識別タグの登録 ID であることを示します。この文字列の後には、必ずピリオド (.) が必要です。 ・ YYYY-MM.—登録 ID のこの部分は、ドメイン名がタグ作成者によって丸 1 ヶ月所有された最初の年月 (YYYY) および (MM) を示します。たとえば、タグを作成するとき、ドメイン名を 1999 年 2 月 15 日に購入している場合、登録 ID のこの部分に **1999-03** を使用します。つまり、ドメイン名を完全な 1 ヶ月間所有した最初の年月は 1999 年 (1999) の 3 月 (03) となります。年と月は、必ずダッシュで区切ります。 ・ ReversedDomainName—この部分は、ソフトウェア識別タグを作成している組織のドメイン名を逆さにしたものです。たとえば、revenera.com がドメイン名であるとすると、その逆さ表示は: **com.revenera** ・ ,division – この部分はオプションで、コンマ (,) で始まり、追加文字列が続きます。組織内の異なる部署や部門によって区別するのに役立つ文字列を入力できます。登録 ID に、このオプションの区別を使用しない場合は、コンマも追加文字列も不要です。 InstallShield では、タグ ファイルの名前の一部として入力した値 (*TagCreatorID_UniqueID*.swidtag) が使用されますので注意してください。したがって、入力する ID に、ファイル名には使えない文字は使用できません。
ソフトウェア作成者	基本の MSI	ソフトウェアを作成した組織の名前を指定します。 この設定はオプションです。この設定を空白のままにしておくと、ソフトウェア作成者の名前に "タグ作成者" 設定の値が使用されます。

テーブル 5 • ソフトウェア識別タグの設定 (続き.)

設定	プロジェクトの種類	説明
ソフトウェア作成者 ID	基本の MSI	ソフトウェアを作成した組織の登録 ID を入力します。この ID は、同じ作成者名を持ち、かつ、異なる国に属する法的組織が複数あるとき、それらを区別するために有益です。 この設定はオプションです。この設定を空白のままにしておくと、ソフトウェア作成者 ID に "タグ作成者 ID" 設定の値が使用されます。 登録 ID の表示規則は、次のとおりです: **regid.YYYY-MM.ReversedDomainName,division** 登録 ID ビューは、次の部分から構成されます: • regid.—文字列 *regid* は、XML 部分がソフトウェア識別タグの登録 ID であることを示します。この文字列の後には、必ずピリオド (.) が必要です。 • YYYY-MM.—登録 ID のこの部分は、ドメイン名がタグ作成者によって丸 1 ヶ月所有された最初の年月 (YYYY) および (MM) を示します。たとえば、タグを作成するとき、ドメイン名を 1999 年 2 月 15 日に購入している場合、登録 ID のこの部分に **1999-03** を使用します。つまり、ドメイン名を完全な 1 ヶ月間所有した最初の年月は 1999 年 (1999) の 3 月 (03) となります。年と月は、必ずダッシュで区切ります。 • ReversedDomainName—この部分は、ソフトウェア識別タグを作成している組織のドメイン名を逆さにしたものです。たとえば、revenera.com がドメイン名であるとすると、その逆さ表示は: **com.revenera** • ,division – この部分はオプションで、コンマ (,) で始まり、追加文字列が続きます。組織内の異なる部署や部門によって区別するのに役立つ文字列を入力できます。登録 ID に、このオプションの区別を使用しない場合は、コンマも追加文字列も不要です。
ソフトウェア ライセンサー	基本の MSI	ソフトウェアの著作権を所有する組織の名前を入力します。 この設定はオプションです。この設定を空白のままにしておくと、ソフトウェア ライセンサーの名前に "タグ作成者" 設定の値が使用されます。

テーブル 5 • ソフトウェア識別タグの設定 (続き.)

設定	プロジェクトの種類	説明
ソフトウェア ライセンサー ID	基本の MSI	ソフトウェアの著作権を所有する組織の登録 ID を入力します。この ID は、同じライセンサー名を持ち、かつ、異なる国に属する法的組織が複数あるとき、それらを区別するために有益です。 この設定はオプションです。この設定を空白のままにしておくと、ソフトウェア ライセンサー ID に "タグ作成者 ID" 設定の値が使用されます。 登録 ID の表示規則は、次のとおりです: **regid.YYYY-MM.ReversedDomainName,division** 登録 ID ビューは、次の部分から構成されます: • regid.—文字列 *regid* は、XML 部分がソフトウェア識別タグの登録 ID であることを示します。この文字列の後には、必ずピリオド (.) が必要です。 • YYYY-MM.—登録 ID のこの部分は、ドメイン名がタグ作成者によって丸 1 ヶ月所有された最初の年月 (YYYY) および (MM) を示します。たとえば、タグを作成するとき、ドメイン名を 1999 年 2 月 15 日に購入している場合、登録 ID のこの部分に **1999-03** を使用します。つまり、ドメイン名を完全な 1 ヶ月間所有した最初の年月は 1999 年 (1999) の 3 月 (03) となります。年と月は、必ずダッシュで区切ります。 • ReversedDomainName—この部分は、ソフトウェア識別タグを作成している組織のドメイン名を逆さにしたものです。たとえば、revenera.com がドメイン名であるとすると、その逆さ表示は: **com.revenera** • ,division – この部分はオプションで、コンマ (,) で始まり、追加文字列が続きます。組織内の異なる部署や部門によって区別するのに役立つ文字列を入力できます。登録 ID に、このオプションの区別を使用しない場合は、コンマも追加文字列も不要です。
SCCM アプリ モデル データを追加	基本の MSI	インストールのソフトウェア識別タグに、Microsoft System Center Configuration Manager のアプリケーション モデル データを含めるかどうかを指定します。[はい] を選択した場合、この設定のサブ設定を必要に応じて構成します。 詳細については、「製品の Microsoft System Center Configuration Manager アプリケーション モデル データを含める」を参照してください。
アプリケーションの種類:	基本の MSI	インストールするアプリケーションの種類を選択します。

テーブル 5 • ソフトウェア識別タグの設定 (続き.)

設定	プロジェクトの種類	説明
優先	基本の MSI	作成中のインストールを他のインストールに優先させる場合、この設定の省略記号ボタン (...) をクリックします。[優先] ダイアログ ボックスが開き、作成中のインストールが優先する Windows Installer パッケージ (.msi) を参照できます。
置き換えられたコンポーネントのアンインストール	基本の MSI	現在のインストールの実行中、優先されるアプリケーションのインストールの前に、とって代わられるアプリケーションをアンインストールするかどうかを指定します。

Configuring Add/Remove Programs with Windows Installer (Windows Installer ヘルプ ライブラリ)

[パッケージ情報] ビュー

プロジェクト: *[パッケージ情報] ビューは、MSIXのプロジェクト タイプで使用できます:*

[パッケージ情報] ビューには、ビルド中のインストールについての一般設定を構成することができる他のビューへのリンクが含まれています。

- [一般情報] ビュー
- [ビジュアル資産] ビュー
- [アプリ機能] ビュー
- [宣言] ビュー
- [コンテンツ URI] ビュー

[一般情報] ビュー

プロジェクト: *[一般情報] ビューは、MSIXのプロジェクト タイプで使用できます:*

[一般情報] ビュー には、プロジェクト、会社、および製品に関する基本情報が表示されます。このビューに入力した情報には、参照だけに使用する情報、Windows ロゴ要件を満たすために使用する情報、さらにプロジェクトの基本的な設定を構成するために使用する情報があります。

新しいインストール プロジェクトを作成する場合、プロジェクトの [一般情報] ビューの設定を構成する必要があります。InstallShield は、デフォルトの設定を使用して新しいプロジェクトを作成しますが、独自のニーズに合ったデータをプロジェクトに含めるためには、独自の値を設定することをお勧めします。

[一般情報] ビューは、次の要素で構成されます:

- ボタンの列
- 設定を表示するグリッド

次のテーブルは、[一般情報] ビューに表示されるボタンについて説明します。

テーブル 6 • [一般情報] ビューのコントロール

コントロールの名前	アイコン	説明
カテゴリー別		カテゴリごとに設定を並べ替えます。
アルファベット順		設定をアルファベット順に並べ替えます。

[一般情報] ビューの各設定についての説明は、「[一般情報] ビューの設定」を参照してください。

[一般情報] ビューの設定

プロジェクト: *[一般情報] ビューは、次のプロジェクト タイプで使用できます:*

- *MSIX*
- *MSIX 変更パッケージ*

[一般情報] ビューの設定は、次のメイン カテゴリに分かれています:

- 全般
- ベース パッケージの詳細
- パッケージ
- 表示プロパティ

全般

[一般情報] ビューの "全般" 領域では、製品名や製品バージョンなどの詳細を指定します。この領域には、以下の設定があります。

テーブル 7 • 全般

設定	プロジェクトの種類	説明
プロジェクトのファイル名	MSIX、MSIX 変更パッケージ	これは、InstallShield プロジェクト ファイルのパスとファイル名を表示する読み取り専用設定です。プロジェクトの種類も表示します。 プロジェクト ファイルの拡張子は、そのプロジェクトの種類によって異なります: • **.ism プロジェクト ファイル** – この拡張 MSIX モジュールを使用します。 詳細については、「プロジェクトの種類」を参照してください。

テーブル 7 • 全般

設定	プロジェクトの種類	説明
プロジェクト ファイルの形式	MSIX、MSIX 変更パッケージ	プロジェクト ファイル (.ism または .dim) に使用するファイル形式を選択します。選択可能なオプションは以下のとおりです: • **バイナリ** – ism ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。 これは、MSIX モジュール プロジェクトのデフォルト形式です。これらのプロジェクト タイプでこのプロジェクト ファイル形式を選択すると、Windows Installer データベース エディターで .ism ファイルを更新できます。また、Windows Installer API またはそのオートメーション インターフェイスを使って、.ism ファイルを変更することもできます。 ***メモ:*** *どちらのプロジェクト ファイル形式では、コマンドラインからリリースをビルドできます。*

ベース パッケージの詳細

[一般情報] ビューの **[ベース パッケージの詳細]** 領域を使って、MSIX パッケージと MSIX 変更パッケージとの関係を定義します。これらの詳細は、次のように変更パッケージ マニフェストに表示されます:

```
<Dependencies>
    <uap4:MainPackageDependency Name="Main.App" Publisher="CN=Revenera, C=US" />
</Dependencies>
```

この領域には、以下の設定があります:

テーブル 8 • ベース パッケージの詳細

設定	プロジェクトの種類	説明
ベース パッケージの名前	MSIX 変更パッケージ	必須。ベース パッケージを識別する名前を説明します (<Identity Name="Main.App">)。名前の属性は、大文字と小文字を区別します。
発行元の名前	MSIX 変更パッケージ	メイン パッケージの署名に使用した証明書の [サブジェクト¥ソフトウェア メーカー] 名を指定します。たとえば、 CN=Revenera、O=Revenera、L=Itasca、S=Illinois および C=US。

ヒント: 依存関係/メイン パッケージ manifest.xml ファイルの識別要素である [ソフトウェア メーカー] 属性に表示されるソフトウェア メーカー名と同じです。メイン パッケージと更新パッケージが同じ署名証明書を使用する場合、このフィールドはオプションです。

パッケージ

パッケージには、パッケージ名、バージョン、パッケージ コード、およびデフォルト言語についての情報が含まれます。

[一般情報] ビューの [パッケージ] 領域には、次の設定があります。

テーブル 9 • [パッケージ] の設定

設定	プロジェクトの種類	説明
パッケージ名	MSIX、MSIX 変更パッケージ	パッケージの コンテンツ を説明します。名前の属性は、大文字と小文字を区別します。文字列には、英数字、ピリオドおよびダッシュ文字で構成されます。
パッケージ バージョン	MSIX、MSIX 変更パッケージ	パッケージのバージョン番号です。4 つの部分から成る「メジャー.マイナー.ビルド.リビジョン」の形式で表記されたバージョン文字列。メジャー番号に 0 は使用できません。 Windows 10 (UWP) パッケージの場合、バージョン番号の最後 (4 番目) のセクションは、ストア用に予約されているので、パッケージのビルド時に 0 のままに残す必要があります (但し、ストアによってこのセクションの値が変更される場合があります)。その他のセクションは (0 に設定不可能な最初のセクション以外) 0 から 65535 の間の整数に設定する必要があります。
パッケージ コード	MSIX	このパッケージを一意に識別する GUID を入力します。 このコードはパッケージを一意に識別するため、パッケージ コードの変更は推奨されません。
デフォルト言語:	MSIX、MSIX 変更パッケージ	リソースの言語は、パッケージに含まれています。属性の構文は、IETF によって定義されます。

表示プロパティ

表示プロパティには、表示名と説明についての情報が含まれます。

[一般情報] ビューの [表示プロパティ] 領域には、次の設定があります。

テーブル 10 • 表示プロパティの設定

設定	プロジェクトの種類	説明
表示名	MSIX、MSIX 変更	ユーザーに表示可能なフレンドリ名です。

テーブル 10 • 表示プロパティの設定

設定	プロジェクトの種類	説明
説明	MSIX、MSIX 変更	ユーザーに表示可能な分かりやすい説明です。

[ビジュアル資産] ビュー

プロジェクト: *[ビジュアル資産] ビューは、次のプロジェクト タイプで使用できます:*

- *MSIX*
- *MSIX 変更パッケージ*

[MSIX ビジュアル資産] エクスプローラーでは、統合的に視覚的な方法で、MSIX アプリのビジュアル的な側面を説明することができます。

デフォルト タイトル、ロゴ イメージ、テキストおよび背景色、初期画面の方向、スプラッシュ画面ならびにロック画面のタイトル外観を変更することができます。

[ビジュアル資産] ビューの設定は、次のメイン カテゴリに分かれています:

- パッケージ ロゴの設定
- 既定のビジュアル資産の設定
- ユーザー定義の設定

パッケージ ロゴの設定

プロジェクト: *[パッケージ ロゴ] ビューは、次のプロジェクト タイプで使用できます:*

- *MSIX*
- *MSIX 変更パッケージ*

[ビジュアル資産] ビューの [パッケージロゴ] 領域を使って、表示設定、パッケージ ロゴおよびスケーリングされた画像の詳細を指定できます。

- ディスプレイの設定

ディスプレイの設定

Windows アプリ パッケージのビジュアル部分を説明します。

この領域には、以下の設定があります。

テーブル 11 • パッケージ ロゴの設定

パッケージ ロゴの設定	プロジェクトの種類	説明
パッケージ ロゴ	MSIX、MSIX 変更パッケージ	イメージを含むファイルへのパス。このイメージはアプリ インストーラー、パートナー センター、ストアの [アプリのレポート] オプションおよび [レビューの書き込み] オプションでで表示されます。
スケーリングされた資産	MSIX、MSIX 変更パッケージ	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

既定のビジュアル資産の設定

プロジェクト: *[デフォルト ビジュアル資産] ビューは、MSIXのプロジェクト タイプで使用できます:*

[ビジュアル資産] ビューの [デフォルト ビジュアル資産] 領域を使って、表示設定、タイル、アプリ アイコン、スプラッシュ画面、および通知などの詳細を指定します。

- ディスプレイの設定
- 小さいタイル
- 中タイル
- 横長タイル
- 大きなタイル
- アプリ アイコン
- スプラッシュ画面
- 通知

ディスプレイの設定

たとえば短い名前、背景色、スプラッシュ画面の背景、およびタイルの表示名など、Windows アプリ パッケージのビジュアル的な側面を説明します。

この領域には、以下の設定があります:

テーブル 12 • ディスプレイの設定

ビジュアル資産の設定	プロジェクトの種類	説明
短い名前	MSIX	タイトルに直接表示される、アプリの短い名前。1 文字から 40 文字までの文字列。
背景色	MSIX	アプリ タイトルの背景色を指定します。"#" または名前付きの色で始まる 3 バイトの 16 進数。
スプラッシュ画面の背景	MSIX	スプラッシュ画面の背景色を指定します。"#" または名前付きの色で始まる 3 バイトの 16 進数。
タイルに名前を表示する	MSIX	アプリ名を中、大、または/および横長タイルに表示するかどうかを指定します。タイトル サイズは、指定されたロゴのサイズでアプリの表示名を表示します。

小さいタイル

小さいサイズ (71x71) のタイル背景イメージへの相対パスを指定するか、省略記号ボタン (...) をクリックして .png または .jpg ファイルを参照します。

この領域には、以下の設定があります:

テーブル 13 • 小さいタイル

ビジュアル資産の設定	プロジェクトの種類	説明
小さいタイル イメージ	MSIX	小さいサイズ (71x71) のタイル背景イメージへの相対パスを指定するか、省略記号ボタン (...) をクリックして .png または .jpg ファイルを参照します。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

中タイル

中サイズ (150x150) のタイル背景イメージへの相対パスを指定するか、省略記号ボタン (...) をクリックして .png または .jpg ファイルを参照します。

この領域には、以下の設定があります:

テーブル 14 • 中タイル

ビジュアル資産の設定	プロジェクトの種類	説明
普通サイズのタイル イメージ	MSIX	中サイズ (150x150) のタイル背景イメージへの相対パスを指定するか、省略記号ボタン (...) をクリックして .png または .jpg ファイルを参照します。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

横長タイル

ピン留めおよびサイズの変更が行われた場合のみ、横長タイルに使用する 310x150 イメージ ファイルを指定します。この設定はオプションです。これを指定すると、タイルを横長に設定することができます。

この領域には、以下の設定があります:

テーブル 15 • 横長タイル

ビジュアル資産の設定	プロジェクトの種類	説明
横長タイル イメージ	MSIX	ピン留めおよびサイズの変更が行われた場合のみ、横長タイルに使用する 310x150 イメージ ファイルを指定します。この設定はオプションです。これを指定すると、タイルを横長に設定することができます。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

大きなタイル

ピン留めおよびサイズの変更が行われた場合のみ、大サイズ タイルに使用する 310x310 イメージ ファイルを指定します。この設定はオプションです。これを指定すると、タイルを大サイズに設定することができます。

この領域には、以下の設定があります:

テーブル 16 • 大きいタイル

ビジュアル資産の設定	プロジェクトの種類	説明
大きいタイル イメージ	MSIX	ピン留めおよびサイズの変更が行われた場合のみ、大サイズ タイルに使用する 310x310 イメージ ファイルを指定します。この設定はオプションです。これを指定すると、タイルを大サイズに設定することができます。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

アプリ アイコン

スタート メニューの [すべてのアプリ] リストの左側の列に表示される小さいロゴ アイコンとして使用する 44X44 イメージ ファイルを指定します。イメージ ファイルが無い場合、InstallShield によってターゲットのタイル構成またはショートカットのアイコンから小さいロゴが生成されます。

この領域には、以下の設定があります:

テーブル 17 • アプリ アイコン

ビジュアル資産の設定	プロジェクトの種類	説明
アプリ アイコン イメージ	MSIX	スタート メニューの [すべてのアプリ] リストの左側の列に表示される小さいロゴ アイコンとして使用する 44X44 イメージ ファイルを指定します。イメージ ファイルが無い場合、InstallShield によってターゲットのタイル構成またはアプリケーションのアイコンから小さいロゴが生成されます。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

スプラッシュ画面

アプリの起動中に表示されるスプラッシュ画面の外観を定義します。

この領域には、以下の設定があります:

テーブル 18 • スプラッシュ画面

ビジュアル資産の設定	プロジェクトの種類	説明
スプラッシュ画面イメージ	MSIX	アプリの起動中に表示されるスプラッシュ画面の外観を定義します。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

通知

システムがロック状態のときに表示されるロック画面上でアプリを表すバッジおよび通知を定義します。

この領域には、以下の設定があります:

テーブル 19 • 通知

ビジュアル資産の設定	プロジェクトの種類	説明
ロック画面の通知	MSIX	ロック画面上でアプリの表示に使用されるタイルの種類。これは Badge または BadgeAndTileText のどちらかを選べます。 この属性には次の値のひとつが可能です: • Badge – ステータスを表す数字またはグリフを表示します。 • BadgeAndTileText – バッジとテキストの両方を使って、詳細なステータスを表示できます。
バッジ ロゴ イメージ	MSIX	バッジの隣に表示される、アプリを識別するロゴ イメージ。このイメージはモノクロで、24 x 24 ピクセルの .png 形式でなくてはなりません。
スケーリングされた資産	MSIX	異なる表示倍率に同じアイコンの複数のバージョンを指定する場合は [はい] を選択します。倍率はテキストなどの UI 要素のサイズを決定します。 倍率は 100% から 400 % の範囲です。大きな値はより大きな UI 要素を作成し、高 DPI ディスプレイで読みやすくなります。

ユーザー定義の設定

プロジェクト: *[ユーザー定義] ビューは、MSIXのプロジェクト タイプで使用できます:*

デフォルトで、[ショートカット] ビューで作成されたすべてのショートカット エントリは [デフォルト ビジュアル資産] を参照します。コンテキスト メニュー オプションを使って、カスタム ビジュアル アセットも定義することができます。これらのカスタム定義のビジュアル資産は、[アプリケーションの種類] ビュー内の任意の存在ショートカットにマップすることができます。

新しいタイトル、ロゴ イメージ、テキストおよび背景色、初期画面の方向、スプラッシュ画面ならびにロック画面のタイトル外観を作成することができます。これらのビジュアル資産は、[アプリケーション] ビュー内の任意のショートカットにマップすることができます。

詳細については、「既定のビジュアル資産の設定」を参照してください。

[アプリ機能] ビュー

プロジェクト: *[アプリ機能] ビューは、MSIXのプロジェクト タイプで使用できます:*

特定の API または画像、音楽、またはカメラやマイクなどのデバイスといったリソースにアクセスするためには、アプリのパッケージ マニフェスト内で特定の機能が有効化されている必要があります。

[アプリ機能] ビューの設定は、次のメイン カテゴリに分かれています:

- デバイスのアプリ機能
- 全般アプリ機能
- 制限付きアプリ機能

デバイスのアプリ機能

プロジェクト: *[アプリ機能] ビューは、MSIXのプロジェクト タイプで使用できます:*

デバイスのアプリ機能は、周辺または内部デバイスへのアクセスを許可します。これは、アプリ パッケージ マニュフェストの [デバイスのアプリ機能] 要素で指定されます。

テーブル 20 ・ デバイスのアプリ機能

デバイス アプリ機能の設定	プロジェクトの種類	説明
Bluetooth	MSIX	アプリが GATT (Generic Attribute) または RFCOMM (Classic Basic Rate) プロトコルの両方を使ってペアリング済みの Bluetooth デバイスと通信できるようにします。

テーブル 20 • デバイスのアプリ機能

デバイス アプリ機能の設定	プロジェクトの種類	説明
視線入力 (視線追跡)	MSIX	互換性のある視線追跡デバイスが接続されているとき、アプリはユーザーがアプリケーション境界の何処を見ているのかを検出することができます。
場所	MSIX	PC の GPS センサーなどの専用ハードウェア、または使用可能なネットワーク情報から取得された場所機能へのアクセスを提供します。
低レベル	MSIX	GPIO、I2C、SPI、および PWM デバイスへのアクセスを提供します。
マイク	MSIX	マイクのオーディオ フィードへのアクセスを提供し、これによってアプリが接続されているマイクからの音声を録音できます。
店舗販売時点管理	MSIX	店舗販売時点管理機能は、Windows.Devices.PointOfService 名前空間の API へのアクセスを有効化します。この名前空間は、アプリが店舗販売時点管理 (POS) バーコード スキャンおよび磁気カードリーダーにアクセスできるようにします。 名前スペースは、ベンダーに依存しないインターフェイスで Windows ストア アプリの様々な製造元かからの POS デバイスにアクセスできるようにします。
近接	MSIX	近距離通信無線または Wi-Fi Direct を通じて、コンピューターの近くにあるデバイスに接続できるようにします。近距離通信を使用すると、ファイルを送信したり、近くのデバイス上のアプリケーションと通信したりできます。
Webcam	MSIX	ビルトイン カメラまたは Webcam のビデオ フィードへのアクセスを許可し、アプリが写真やビデオを取り込めるようにします。

全般アプリ機能

プロジェクト• *[アプリ機能] ビューは、MSIXのプロジェクト タイプで使用できます:*

全般アプリ機能は、一般的なアプリのシナリオへのアクセスを許可しまうs。

テーブル 21・全般アプリ機能

全般 アプリ機能の設定	プロジェクトの種類	説明
3D オブジェクト	MSIX	ユーザーの 3D オブジェクトへのプログラムによるアクセスを提供し、アプリがユーザー インターフェイスを使用せずにライブラリ内のすべてのファイルをエミュレートおよびアクセスできるようにします。この機能は通常、3D オブジェクト ライブラリ全体のアクセスを必要とする 3D アプリおよびゲームで使用されます。
AllJoyn	MSIX	ネットワーク上で AllJoyn が有効化されているアプリおよびデバイスが検出および相互インタラクトできるようにします。
予約 (カレンダー)	MSIX	ユーザーの予約ストアにアクセスできるようにします。この機能は、同期されたネットワーク アカウントから取得された予約への読み取りアクセス、および予約ストアに書き込む他のアプリへのアクセスを可能にします。この機能を使って、アプリで新しいカレンダーを作成し、そこに予約を書き込むことができます。
背景メディア再生	MSIX	デフォルトで背景再生を可能にするメディア API の動作を変更します。すべてのアクティブなオーディオ ストリームは消音されず、アプリが背景に切り替わるときに引き続きストリームされます。さらに、再生中はアプリケーションの有効な状態が自動的に延長されます。
ブロック対象チャット メッセージ	MSIX	スパム フィルター アプリによってブロックされた SMS および MMS メッセージをアプリで読むことができるようにします。
チャット メッセージのアクセス	MSIX	アプリがテキスト メッセージを読み取りおよび削除できるようにします。アプリがシステムデータ ストアにチャット メッセージを保管できるようにします。
コード生成	MSIX	アプリがコードをダイナミックに生成できるようにします。
連絡先	MSIX	電話で集計を列挙する機能を提供します。この機能でアプリケーションはすべてのリンクされた連絡先にアクセスでき、ユーザーの最高のビューを提供します。
画像キャプチャ	MSIX	キャプチャ選択制御によって戻されたターゲットとなる項目のフレームを開発者が取得することができます。

テーブル 21・全般アプリ機能

全般 アプリ機能の設定	プロジェクトの種類	説明
インターネット (クライアント & サーバー)	MSIX	空港や喫茶店などの公共の場所でインターネットおよびネットワークに対して受信および発信アクセスを提供します。この機能はインターネット (クライアント) の一部です。この機能が有効化されているとき、インターネット (クライアント) も有効化する必要はありません。 重要なポートへの入力方向のアクセスは常にブロックされています。
インターネット (クライアント)	MSIX	空港や喫茶店などの公共の場所でインターネットおよびネットワークに対する発信アクセスを提供します。たとえば、ユーザーによってネットワークが公共として設定されているイントラネット ネットワーク。 イントラネット アクセスを必要とするほとんどのアプリには、この機能を使用します。
ミュージック ライブラリ	MSIX	ローカル PC およびホームグループ PC のミュージック ライブラリにファイルを追加、変更または削除する機能を提供します。
オフライン マップ管理	MSIX	アプリによる MapControl API へのアクセスを可能にします。この機能は、ユーザーとアプリ間で共有されるマップ パッケージをダウンロードする為に必要です。
通話記録	MSIX	アプリがデバイス上のすべての電話回線にアクセスして、次の処理を行うこと許可します: ・ 電話回線上で通信を行い、ユーザーに対するプロンプトなしでシステム ダイヤラーを表示します。 ・ 回線関連のメタデータにアクセスします。 ・ 回線関連のトリガーにアクセスします。 ・ ユーザーが選択したスパム フィルター アプリが、禁止リストや発信元情報を設定およびチェックできるようにします。
通話記録 (パブリック)	MSIX	アプリがデバイス上の携帯データ通信および一部の VoIP 通話履歴を読み取り可能にします。この機能はまた、アプリによる VoIP 通話履歴の入力もできるようにします。
画像ライブラリ	MSIX	ローカルコンピューターおよびホームグループ コンピューターの画像ライブラリにファイルを追加、変更または削除できるようにします。

テーブル 21・全般アプリ機能

全般 アプリ機能の設定	プロジェクトの種類	説明
プライベート ネットワーク (クライアント & サーバー)	MSIX	認証されたドメイン コントローラーがあるか、ユーザーがホーム ネットワークまたは社内ネットワークと指定しているイントラネット ネットワークに、着信および発信アクセスできるようにします。重要なポートへの入力方向のアクセスは常にブロックされています。
リモート システム	MSIX	ユーザーの Microsoft アカウントに関連付けられているデバイスへのアクセスを提供します。この機能は、リモート デバイスを検出および接続するために必要です。
リムーバブル記録装置	MSIX	リムーバブル記録装置のファイルを追加、変更、または削除できるようにします。アプリケーションは、ファイルの種類の関連付け宣言を使ってマニフェストで定義された、リムーバブル記憶域内のファイルの種類にのみアクセスできます。
空間認識	MSIX	ユーザーの HoloLens やその他の Mixed Reality ヘッドセットのようなユーザーの環境に関連して追跡が行われるデバイスの場所にアクセスできるようにします。
ユーザーアカウント情報	MSIX	アプリがユーザーの名前と画像にアクセスできるようにします。
ユーザー データ タスク	MSIX	アプリがタスクに関する情報を選択および取得できるようにします。
ユーザー通知リスナー	MSIX	ユーザーの通知を読み取りおよび管理できるようにします。通常、ウェアラブル デバイスでユーザーの通知をミラーする比較デバイスメーカーに使用されます。
ベンダー ライブラリ	MSIX	ユーザーのビデオへのプログラムによるアクセスを提供し、アプリがユーザー インターフェイスを使用せずにライブラリ内のすべてのファイルをエミュレートおよびアクセスできるようにします。
VOIP 通話	MSIX	アプリが VoIP 通話 API にアクセスできるようにします。

制限付きアプリ機能

プロジェクト: *[アプリ機能] ビューは、MSIXのプロジェクト タイプで使用できます:*

制限付きアプリ機能を使って、必要で適切かつ特別なアプリ機能へのアクセスを許可します。

テーブル 22・制限付きアプリ機能

制限付きアプリ機能の設定	プロジェクトの種類	説明
昇格	MSIX	Microsoft パートナーやエンタープライズによって作成されたアプリが、アプリの起動時または有効期間中に自動昇格を必要とする既存のデスクトップ機能を保持することができます。
エンタープライズ認証	MSIX	この機能は、ストア ポリシーの対象です。ドメイン認証情報を必要とするエンタープライズのイントラネット リソースに接続する機能を提供します。この機能は通常、ほとんどのアプリで必要ありません。
フルトラスト アクセス許可レベル	MSIX	ユーザーのマシン上でフルトラスト アクセス許可レベルでアプリを実行できるようにします。この機能は、appx または mxix パッケージとして配布された任意のデスクトップ アプリケーションでも必要です。
ローカルのシステム サービス	MSIX	localSystemServices の制限付きアプリ機能は、Microsoft パートナーおよびエンタープライズによって作成されたアプリケーションをアプリと共に 1 つ以上のローカル システム サービスにインストールできるようにします (つまり、アプリケーションがサービスの StartAccount を LocalSystem として宣言することができます)。このシナリオには、packagesServices アプリ機能も必要です。
共有ユーザー証明書	MSIX	この機能は、ストア ポリシーの対象です。たとえば、ユーザーの id を検証するためのスマート カード証明書など、ソフトウェアおよびハードウェア証明書へのアクセス機能を提供します。アプリが証明書の宣言を使ったプライベート証明書を含む場合、この機能は必ず必要ではありません。
パッケージ サービス	MSIX	packagedServices の制限付きアプリ機能は、Microsoft パートナーまたはエンタープライズによって作成されたアプリケーションがそのパッケージ マニフェスト内で windows.service 拡張を宣言し、アプリと共に 1 つ以上のサービスをインストールすることができるようにします。これらのサービスは、[ローカル サービス]、[ネットワーク サービス]、または [ローカル システム] アカウントの下で実行するように構成できます。ローカル サービスおよびネットワーク サービスのみ、packagedServices アプリ機能を必要とします。ローカル システム サービスには、packagedServices および localSystemServices アプリ機能が必要です。

[宣言] ビュー

プロジェクト: *[宣言機能] ビューは、MSIXのプロジェクト タイプで使用できます:*

[宣言] ビューは、アプリケーション、パッケージ レベルの宣言を作成および管理できるビジュアル ツールを提供し、それらのプロパティを構成することができます。

宣言には次の 2 種類あります:

- [アプリケーションの宣言セット]
- パッケージの宣言

[アプリケーションの宣言セット]

プロジェクト: *[アプリケーション 宣言セット] ビューは、MSIXのプロジェクト タイプで使用できます:*

アプリケーション宣言セットを使って、宣言をアプリケーションに関連付けるために [アプリケーション] ビューでアプリケーション ショートカットに必要な複数の宣言セットを追加することができます。

ツリー内で宣言タイプが選択されている間、各宣言の設定がグリッドに表示されます。

宣言セットは次の宣言タイプで構成されます:

- アカウント画像プロバイダー
- アラーム
- アプリ拡張ホスト
- アプリ サービス
- 予定プロバイダー
- 自動再生コンテンツ
- 自動再生デバイス
- バックグランド タスク
- キャッシュ ファイル アップデータ
- カメラの設定
- ダイヤル プロトコル
- ファイル オープン ピッカー
- ファイル保存ピッカー
- ファイル タイプの関連付け
- ロック画面
- メディア再生
- パーソナル アシスタントの起動

- 3D 印刷ワークフロー
- プレインストールされた構成
- 印刷 タスクの設定
- プロトコル
- 制限付き起動
- 検索
- Service
- 共有ターゲット
- アップデート タスク
- Web アカウント プロバイダー
- Windows 共有フォント
- カスタム宣言

アカウント画像プロバイダー

この宣言を使って、Windows の [アカウント画像設定] コントロール パネルにアプリケーションを一覧表示します。ユーザーはこの宣言を選択して、既存のアカウント画像を変更するか、新しいアカウント画像をキャプチャすることができます。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[アカウント画像プロバイダー] 宣言タイプに表示される設定について説明します:

テーブル 23 : アカウント画像プロバイダー

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。

テーブル 23 • アカウント画像プロバイダー

宣言	プロジェクトの種類	説明
実行可能ファイル	MSIX	".exe" で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース (たとえば CPU およびメモリ) の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

アラーム

この宣言を使って、アプリケーションを [システム アラーム] アプリとして指定できます。ユーザーは、[システム アラーム] アプリとして指定された、選択済みのアプリケーションのみにシステム アラームを設定することができます。

メモ: *1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。*

- アプリの設定

アプリの設定

"アプリケーション" 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[アラーム] 宣言タイプに表示される設定について説明します:

テーブル 24 • アラーム

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

アプリ拡張ホスト

この宣言を使って、アプリでサポートされている HostNames を登録することができます。

***メモ:** 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。*

- アプリの設定
- 名前

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[アプリ拡張ホスト] 宣言タイプに表示される設定について説明します:

テーブル 25 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *

テーブル 25 • アプリの設定

宣言	プロジェクトの種類	説明
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

名前

アプリ拡張ホストでサポートされているホスト名を登録する設定。

次のテーブルは、[アプリ拡張ホスト] 宣言タイプに表示される設定について説明します:

テーブル 26 • 名前

宣言	プロジェクトの種類	説明
名前	MSIX	• **名前** – アプリでサポートされているアプリ拡張ホスト名を指定します。名前を追加するには、[**新しい名前の追加**] ボタンをクリックします。 ***重要 • 1 つの名前を指定することが必須条件です。***

アプリ サービス

この宣言を使って、コミュニケーションを取り合うアプリケーションのバックグラウンド タスクを提供または起動するための合意を使用できます。

- アプリの設定
- プロパティ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[アプリサービス] 宣言タイプに表示される設定について説明します:

テーブル 27 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

これらはアプリ サービスの設定です。

次のテーブルは、[アプリサービス] 宣言タイプに表示される設定について説明します:

テーブル 28 • プロパティ

宣言	プロジェクトの種類	説明
名前	MSIX	アプリ サービスの名前を指定します。
サーバー名	MSIX	アプリ サービスのサーバー名を指定します。

予定プロバイダー

この宣言は、アプリケーションがカレンダー API とインタラクトして、予定を作成するための機能を提供します。

メモ• 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定
- 起動アクション

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[予定プロバイダー] 宣言タイプに表示される設定について説明します:

テーブル 29 • 予定プロバイダー

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。

テーブル 29・予定プロバイダー

宣言	プロジェクトの種類	説明
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

起動アクション

アプリの ”カレンダー予定” 設定を登録する設定。

次のテーブルは、[予定プロバイダー] 宣言タイプに表示される設定について説明します:

テーブル 30・起動アクション

宣言	プロジェクトの種類	説明
必要なビュー	MSIX	”必要なビュー” フィールドを指定して、起動時にアプリのウィンドウが必要とするスペースを設定します。このフィールドには次の値のひとつが可能です: ・ **デフォルト** ・ **より少なく使用** ・ **半分使用** ・ **より多く使用** ・ **最小限を使用**

テーブル 30・起動アクション

宣言	プロジェクトの種類	説明
起動アクション	MSIX	アプリでサポートされている予定起動アクションを指定します。起動アクションを追加するには、[**新しい起動アクションの追加**] ボタンをクリックします。 ***重要・少なくとも 1 つの起動アクションを指定することが必須条件です。***
動詞	MSIX	起動時にアプリに渡される一意の ID。これには、次の値のひとつが可能です: ・ **予定の追加** ・ **予定の削除** ・ **予定の変更** ・ **時間枠の表示** ・ **予定の詳細を表示**
動詞 – エントリ ポイント	MSIX	アクティベート可能なクラス ID – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、|、?、または *
動詞 – 実行可能ファイル	MSIX	デフォルトの起動実行可能ファイル – ″.exe″ で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、|、?、または *
動詞 – リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。
動詞 – ランタイム タイプ	MSIX	ランタイム プロバイダーこのフィールドは通常、アプリで異なるフレームワークが存在するときに使用されます。
動詞 – スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、|、?、または *
動詞 – 必要なビュー	MSIX	”必要なビュー” フィールドを指定して、起動時にアプリのウィンドウが必要とするスペースを設定します。このフィールドには次の値のひとつが可能です: ・ **デフォルト** ・ **より少なく使用** ・ **半分使用** ・ **より多く使用** ・ **最小限を使用**

自動再生コンテンツ

この宣言を使って、アプリを自動再生コンテンツ イベントのオプションとして登録できます。自動再生コンテンツ イベントは、カメラ メモリ カード、サム ドライブ、または DVD などのボリューム デバイスが PC に挿入されたときに生成されます。

- アプリの設定
- 起動アクション

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[自動再生コンテンツ] 宣言タイプに表示される設定について説明します:

テーブル 31 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *

テーブル 31・アプリの設定

宣言	プロジェクトの種類	説明
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

起動アクション

アプリを自動再生コンテンツ イベントのオプションとして登録するには、次の手順に従います: 自動再生コンテンツ イベントは、カメラ メモリ カード、サム ドライブ、または DVD などのボリューム デバイスが PC に挿入されたときに生成されます。

次のテーブルは、[自動再生コンテンツ] 宣言タイプに表示される設定について説明します:

テーブル 32・起動アクション

宣言	プロジェクトの種類	説明
起動アクション	MSIX	アプリでサポートされている予定起動アクションを指定します。起動アクションを追加するには、[**新しい起動アクションの追加**] ボタンをクリックします。 ***重要**・少なくとも 1 つの起動アクションを指定することが必須条件です。*
起動アクション動詞	MSIX	イベントが届いたときにアプリに渡すアクティベーション文字列を指定します。「起動アクション動詞」は、自動再生コンテンツおよび自動再生デバイスの「起動アクション動詞」を通して一意でなくてはなりません。
アクションの表示名	MSIX	自動再生 UI で表示する文字列を指定します。
コンテンツ イベント	MSIX	届いたときに認識するイベントの種類を指定します。

自動再生デバイス

この宣言を使って、アプリを自動再生デバイス イベントのオプションとして登録できます。自動再生デバイス イベントは、デバイスが PC に接続されているときに生成されます。

- アプリの設定
- 起動アクション

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[自動再生デバイス] 宣言タイプに表示される設定について説明します:

テーブル 33 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

テーブル 33 • アプリの設定

宣言	プロジェクトの種類	説明
起動アクション	MSIX	アプリを自動再生デバイス イベントのオプションとして登録するには、次の手順に従います。自動再生デバイス イベントは、デバイスが PC に接続されているときに生成されます。

起動アクション

アプリを自動再生デバイス イベントのオプションとして登録するには、次の手順に従います。自動再生デバイス イベントは、デバイスが PC に接続されているときに生成されます。

次のテーブルは、[自動再生デバイス] 宣言タイプに表示される設定について説明します:

テーブル 34 • 起動アクション

宣言	プロジェクトの種類	説明
起動アクション	MSIX	アプリでサポートされている予定起動アクションを指定します。起動アクションを追加するには、[**新しい起動アクションの追加**] ボタンをクリックします。 ***重要** • 少なくとも 1 つの起動アクションを指定することが必須条件です。*
起動アクション動詞	MSIX	イベントが届いたときにアプリに渡すアクティベーション文字列を指定します。「起動アクション動詞」は、自動再生コンテンツおよび自動再生デバイスの「起動アクション動詞」を通して一意でなくてはなりません。
アクションの表示名	MSIX	自動再生 UI で表示する文字列を指定します。
デバイス イベント	MSIX	届いたときに認識するイベントの種類を指定します。

バックグランド タスク

アプリが一時停止中でも、バックグラウンド タスクを使ってアプリ コードを実行することができます。

- アプリの設定
- サポート対象タスク タイプ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[バックグランド タスク] 宣言タイプに表示される設定について説明します:

テーブル 35 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

サポート対象タスク タイプ

アプリがアプリが一時停止中でも、バックグラウンド タスクを使ってアプリ コードを実行できます。バックグラウンド タスは、ユーザーがインタラクション不要な小さい作業アイテム向けです。

次のテーブルは、[バックグランド タスク] 宣言タイプに表示される設定について説明します:

テーブル 36 ・ サポート対象タスク タイプ

宣言	プロジェクトの種類	説明
Bluetooth	MSIX	Bluetooth を処理するバックグラウンド タスクの実行を指定します。
Bluetooth デバイスの接続	MSIX	Bluetooth デバイス接続イベントを処理するバックグラウンド タスクの実行を指定します。
チャット メッセージの通知	MSIX	チャット メッセージの通知を処理するバックグラウンド タスクの実行を指定します。
コントロール チャンネル	MSIX	オープン ソケット接続を維持するリアルタイム アプリのバックグラウンド タスクの実行を指定します。
デバイスの使用トリガ	MSIX	バックグラウンド タスkうからセンサーとデバイスへのアクセスを指定します。
全般	MSIX	前景から呼び出すなど、一般的なバックグラウンド タスクを追加します。
場所	MSIX	バックグラウンド タスク内のジオフェンス イベントの取得を指定します。
メディア処理	MSIX	バックグラウンドでメディア ファイルを処理するバックグラウンド タスクの実行を指定します。
通話記録	MSIX	通話でバックグラウンド タスクの実行を指定します。
プッシュ通知	MSIX	Windows 通知サーバーからのプッシュ通知のバックグラウンド タスクの実行を指定します。
システム イベント	MSIX	システム イベント を処理するバックグラウンド タスクの実行を指定します。
タイマー	MSIX	タイマーでバックグラウンド タスクの実行を指定します。
VPN クライアント	MSIX	VPN プラグイン アプリケーションのバックグラウンド タスクの実行を指定します。

キャッシュ ファイル アップデータ

この宣言を使って、特定ファイルのアップデートを提供し、ユーザーがファイルの追跡および管理を行う中央リポジトリとしてアプリを使用できるようにします。

メモ: *1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。*

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[キャッシュ ファイル アップデータ] 宣言タイプに表示される設定について説明します:

テーブル 37 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

カメラの設定

この宣言を使って、UWP デバイス アプリにより多くのカメラ オプションを表示するポップアップをカスタマイズできます。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[カメラの設定] 宣言タイプに表示される設定について説明します:

テーブル 38 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *

テーブル 38 • アプリの設定

宣言	プロジェクトの種類	説明
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

ダイヤル プロトコル

ネットワーク デバイスを使用中の場合、DIAL プロトコルを使って Xbox One 上のアプリケーションを起動することができます。実行しようとしているアプリケーションが使用できない場合、DIAL ネットワークを使ってそれを実行前にインストールすることもできます。

- アプリの設定
- プロパティ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[ダイヤル プロトコル] 宣言タイプに表示される設定について説明します:

テーブル 39 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。

テーブル 39 • アプリの設定

宣言	プロジェクトの種類	説明
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

これらはダイヤル プロトコル名を設定します。

次のテーブルは、[ダイヤル プロトコル] 宣言タイプに表示される設定について説明します:

テーブル 40 • プロパティ

宣言	プロジェクトの種類	説明
名前	MSIX	ダイヤル プロトコルによって起動されるアプリケーションの名前を指定します。

ファイル オープン ピッカー

この宣言を使って、ユーザーは別のアプリを使用中にアプリから直接ファイルをピックすることができます。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定
- サポート対象ファイル タイプ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[オープン ファイル ピッカー] 宣言タイプに表示される設定について説明します:

テーブル 41 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

サポート対象ファイル タイプ

アプリをファイル タイプの処理に登録することを指定します。

次のテーブルは、[オープン ファイル ピッカー] 宣言タイプに表示される設定について説明します:

テーブル 42 • サポート対象ファイル タイプ

宣言	プロジェクトの種類	説明
サポート対象ファイル タイプ	MSIX	選択された場合: • **はい** – アプリはすべてのファイル タイプを認識します。 • **いいえ** – アプリは [サポート対象 ファイル タイプ] の下に宣言されているファイル タイプのみを認識します。
サポート対象ファイル タイプ	MSIX	アプリでサポートされているファイル タイプを指定します。ファイル タイプを追加するには、[**新しいファイル タイプを追加**] ボタンをクリックします。 ***重要 • 少なくとも 1 つ以上のファイル タイプを指定することが必須条件です。***
ファイル タイプ	MSIX	アプリがサポートするファイル タイプを指定します。'.' で始まる必要があります。

ファイル保存ピッカー

この宣言を使って、ユーザーは別のアプリを使用中、アプリに直接ファイルを保存することができます。

メモ • 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定
- サポート対象ファイル タイプ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[ファイル保存ピッカー] 宣言タイプに表示される設定について説明します:

テーブル 43 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

サポート対象ファイル タイプ

アプリをファイル タイプの処理に登録することを指定します。

次のテーブルは、[ファイル保存ピッカー] 宣言タイプに表示される設定について説明します:

テーブル 44 • サポート対象ファイル タイプ

宣言	プロジェクトの種類	説明
サポート対象ファイル タイプ	MSIX	選択された場合: • **はい** – アプリはすべてのファイル タイプを認識します。 • **いいえ** – アプリは [サポート対象 ファイル タイプ の下に宣言されているファイル タイプのみを認識します。
サポート対象ファイル タイプ	MSIX	アプリでサポートされているファイル タイプを指定します。ファイル タイプを追加するには、[**新しいファイル タイプを追加**] ボタンをクリックします。 ***重要** • 少なくとも 1 つ以上のファイル タイプを指定することが必須条件です。*
ファイル タイプ	MSIX	アプリがサポートするファイル タイプを指定します。'.' で始まる必要があります。

ファイル タイプの関連付け

この宣言 タイプを使って、特定のファイル タイプにデフォルトのハンドラーとなるアプリを登録することを選択できます。ユーザーは特定のファイル タイプに対してアプリをデフォルト ハンドラーとして選択し、選択されたアプリは、そのファイル タイプが起動されるたびにアクティベートされます。

- アプリの設定
- 編集フラグ
- プロパティ
- サポート対象ファイル タイプ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[ファイル タイプの関連付け] 宣言タイプに表示される設定について説明します:

テーブル 45 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

テーブル 45 • アプリの設定

宣言	プロジェクトの種類	説明
必要なビュー	MSIX	”必要なビュー” フィールドを指定して、ファイル タイプの起動時にアプリのウィンドウが必要とするスペースを設定します。このフィールドには次の値のひとつが可能です: • **デフォルト** • **より少なく使用** • **半分使用** • **より多く使用** • **最小限を使用**

編集フラグ

一連のファイル タイプの編集フラグを指定します。

次のテーブルは、[ファイル タイプの関連付け] 宣言タイプに表示される設定について説明します:

テーブル 46 • 編集フラグ

宣言	プロジェクトの種類	説明
安全なファイルを開く	MSIX	ファイル タイプの「開く」という単語が任意のダウンロード済みファイルを呼び出すことを指定します。
常に安全でない	MSIX	ファイル タイプが無効な時には、自動的にオーバーライドすることを指定します。

プロパティ

ファイルの種類の関連付けプロパティの設定

次のテーブルは、[ファイル タイプの関連付け] 宣言タイプに表示される設定について説明します:

テーブル 47 • プロパティ

宣言	プロジェクトの種類	説明
表示名	MSIX	一連のファイル タイプの表示名を指定します。
Logo	MSIX	ファイル タイプを識別するために使用されるロゴを指定します。
ヒント	MSIX	ユーザーがマウス ポインターをこの種類のファイルのアイコン上に配置したときに表示されるヒントを指定します。

テーブル 47 • プロパティ

宣言	プロジェクトの種類	説明
名前	MSIX	同じ表示名、ロゴ、ヒント、および編集フラグを共有する一連のファイル タイプの名前を選択します。 ***重要 • 名前は小文字でなくてはなりません。***

サポート対象ファイル タイプ

アプリをファイル タイプの処理に登録するための設定。

次のテーブルは、[ファイル タイプの関連付け] 宣言タイプに表示される設定について説明します:

テーブル 48 • サポート対象ファイル タイプ

宣言	プロジェクトの種類	説明
サポート対象ファイル タイプ	MSIX	アプリでサポートされているファイル タイプを指定します。ファイル タイプを追加するには、[**新しいファイル タイプを追加**] ボタンをクリックします。 ***重要 • 少なくとも 1 つ以上のファイル タイプを指定することが必須条件です。***
ファイル タイプ	MSIX	アプリがサポートするファイル タイプを指定します。'.' で始まる必要があります。
コンテンツ タイプ	MSIX	アプリの ファイル タイプに MIME コンテンツ タイプをシチエします。少なくとも 1 つの '/' を含む必要があります。

ロック画面

この宣言を使って、ユーザーは端末のロック画面を解除しなくても、VoIP コールに応答することができます。

メモ • 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

"アプリケーション" 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[ロック画面] 宣言タイプに表示される設定について説明します:

テーブル 49 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

メディア再生

Windows ストア アプリをサポートするためには、これらのアプリが DVD 再生をサポートすることを指定しなくてはなりません。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定
- サポートされているコーデック

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[メディア再生] 宣言タイプに表示される設定について説明します:

テーブル 50 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *

テーブル 50 • アプリの設定

宣言	プロジェクトの種類	説明
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

サポートされているコーデック

メディア再生にサポートされているメディア コーデックを登録する設定。

次のテーブルは、[メディア再生] 宣言タイプに表示される設定について説明します:

テーブル 51 • サポートされているコーデック

宣言	プロジェクトの種類	説明
サポートされているコーデック	MSIX	アプリでサポートされているメディア コーデックを指定します。ファイル タイプを追加するには、[**新しいコーデックを追加**] ボタンをクリックします。 ***重要 • 少なくとも 1 つのコーデックを指定することが必須条件です。***
コーデック	MSIX	コーデックの名前を指定します。

パーソナル アシスタントの起動

この宣言を使って、特定の状態またはコンテキストにおいて、バックグラウンドアプリから前景アプリにディープ リンクを提供することができます。

メモ • 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[パーソナル アシスタントの起動] 宣言タイプに表示される設定について説明します:

テーブル 52 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

3D 印刷ワークフロー

この宣言によって、Windows ストア デバイスを使って 3D 印刷ダイアログで固有のエクスペリエンスを提供することができます。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[3D 印刷ワークフロー] 宣言タイプに表示される設定について説明します:

テーブル 53 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *

テーブル 53 • アプリの設定

宣言	プロジェクトの種類	説明
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プレインストールされた構成

この宣言を使って、ユーザーとのインタラクションなしでアップデート タスクを即時起動させて、確実にアプリケーションをアップデートすることができます。

メモ• 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[プレインストールされた構成] 宣言タイプに表示される設定について説明します:

テーブル 54 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。

テーブル 54 • アプリの設定

宣言	プロジェクトの種類	説明
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

印刷 タスクの設定

この宣言によって、Windows ストア デバイスを使って高度な印刷設定を表示するポップアップをカスタマイズすることができます。

メモ: *1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。*

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[印刷タスクの設定] 宣言タイプに表示される設定について説明します:

テーブル 55 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロトコル

この宣言を使って、mailto などの既存の URI スキームの使用するか、アプリのカスタム URI スキームを作成することができます。URI アクティベーション拡張によって、カスタム URI スキームを定義するか、既存の URI スキームの処理を登録することができます。

- アプリの設定

- プロパティ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[プロトコル] 宣言タイプに表示される設定について説明します:

テーブル 56 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

アプリを URL プロトコルに登録する設定。

次のテーブルは、[プロトコル] 宣言タイプに表示される設定について説明します:

テーブル 57 • プロパティ

宣言	プロジェクトの種類	説明
Logo	MSIX	アプリと一致するプロトコルの Windows シェルで表示されるアイコン イメージを指定します。
表示名	MSIX	このプロトコルのハンドラーのオプション表示名を指定します。
名前	MSIX	アプリをサポートするプロトコルの名前を指定します。

制限付き起動

この宣言を使って、信頼されていないコンテンツからアプリを保護することができます。子アプリ コンテナは、Windows ストア アプリによって信頼されていないコンテンツが実行された場合、実行時に追加の分離階層を提供します。

メモ• 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[制限付き起動] 宣言タイプに表示される設定について説明します:

テーブル 58 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。

テーブル 58 • アプリの設定

宣言	プロジェクトの種類	説明
実行可能ファイル	MSIX	".exe" で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース (たとえば CPU およびメモリ) の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

共有ターゲット

この宣言を使って、ユーザーがアプリのコンテンツを別のアプリまたはサービスと共有する、またはその逆を行なえるようにします。共有ターゲットをサポートするアプリは、その宣言をサポートする任意のその他のアプリに、またはそれらから共有コンテンツを自動的に共有することができます。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定
- プロパティ
- データ形式
- サポート対象ファイル タイプ

アプリの設定

"アプリケーション" 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[共有ターゲット] 宣言タイプに表示される設定について説明します:

テーブル 59 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

アプリが共有可能なコンテンツを受信できるように登録するため、共有ターゲットのプロパティを指定します。

次のテーブルは、[共有ターゲット] 宣言タイプに表示される設定について説明します:

テーブル 60 • プロパティ

宣言	プロジェクトの種類	説明
共有の説明	MSIX	Windows で表示される共有ターゲットの説明を指定します。

データ形式

アプリでサポートされているデータ形式を指定する設定。

次のテーブルは、[共有ターゲット] 宣言タイプに表示される設定について説明します:

テーブル 61 • データ形式

宣言	プロジェクトの種類	説明
データ形式	MSIX	データ形式を追加するには、[**新しいデータ形式の追加**] ボタンをクリックします。

サポート対象ファイル タイプ

アプリをファイル タイプの処理に登録するための設定。

次のテーブルは、[共有ターゲット] 宣言タイプに表示される設定について説明します:

テーブル 62 • サポート対象ファイル タイプ

宣言	プロジェクトの種類	説明
サポート対象ファイル タイプ	MSIX	選択された場合: • **はい** – アプリはすべてのファイル タイプを認識します。 • **いいえ** – アプリは [サポート対象 ファイル タイプ の下に宣言されているファイル タイプのみを認識します。
サポート対象ファイル タイプ	MSIX	アプリでサポートされているファイル タイプを指定します。 ファイル タイプを追加するには、[**新しいファイル タイプを追加**] ボタンをクリックします。 ***重要** • 少なくとも 1 つ以上のファイル タイプを指定することが必須条件です。*
ファイル タイプ	MSIX	アプリがサポートするファイル タイプを指定します。'.' で始まる必要があります。

検索

この宣言を使って、アプリに検索ペインを追加し、ユーザーがアプリのコンテンツだけでなく、他のアプリからのコンテンツも検索できるようにします。ユーザーは検索クエリそのものを別のアプリに転送することもできます。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[検索] 宣言タイプに表示される設定について説明します:

テーブル 63 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。

テーブル 63 • アプリの設定

宣言	プロジェクトの種類	説明
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

Service

アプリと共にインストールおよび登録されるサービスを指定します。これらのサービスは、[ローカル サービス]、[ネットワーク サービス]、または [ローカル システム] アカウントの下で実行するように構成できます。Windows 10、バージョン 2004 から、サービスを含む MSIX パッケージの実行がサポートされています。

- アプリの設定
- プロパティ
- 依存関係

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[アップデート タスク] 宣言タイプに表示される設定について説明します:

テーブル 64 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。

テーブル 64 • アプリの設定

宣言	プロジェクトの種類	説明
実行可能ファイル	MSIX	".exe" で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース (たとえば CPU およびメモリ) の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

テーブル 65 • プロパティ

宣言	プロジェクトの種類	説明
名前	MSIX	サービスの名前を指定します。
スタートアップの種類	MSIX	サービスのスタートアップの種類。次の値がサポートされています: • 自動 • 手動 • 無効
開始アカウント	MSIX	サービスを実行するアカウントの種類。次の値がサポートされています: • localSystem • localServices • networkService

宣言	プロジェクトの種類	説明
引数	MSIX	(オプション) サービスに渡す引数。

依存関係

これらは windows.service 依存関係の設定です。パッケージ外部の依存関係サービスは、現在サポートされていません。

次のテーブルは、[MSIXサービス] 宣言依存関係のカテゴリに表示される設定について説明します:

テーブル 66 • 依存関係

宣言	プロジェクトの種類	説明
依存するサービス	MSIX	現在のサービスに依存する 1 つ以上のサービスを指定します。

アップデート タスク

この宣言は、ユーザーがアプリをより新しいバージョンにアップデートした後に直接実行されるバックグラウンド タスクです。

メモ• 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定

アプリの設定

"アプリケーション" 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[アップデート タスク] 宣言タイプに表示される設定について説明します:

テーブル 67 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	"アプリケーション" 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。

テーブル 67・アプリの設定

宣言	プロジェクトの種類	説明
実行可能ファイル	MSIX	".exe" で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、"、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース (たとえば CPU およびメモリ) の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

Web アカウント プロバイダー

この宣言を使って、ユーザーが一度サインオンすることを許可し、そのログインで複数アプリにアカウント情報を提供できるようにします。

メモ: 1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。

- アプリの設定
- プロパティ
- マネージド URL

アプリの設定

"アプリケーション" 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[Web アカウント プロバイダー] 宣言タイプに表示される設定について説明します:

テーブル 68 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

Web アカウント プロバイダーを登録するために必要なプロパティを指定します。

次のテーブルは、[Web アカウント プロバイダー] 宣言タイプに表示される設定について説明します:

テーブル 69 • プロパティ

宣言	プロジェクトの種類	説明
URL	MSIX	Web アカウント プロバイダーを識別するメイン URL を指定します。
バックグラウンド エントリ ポイント	MSIX	バックグラウンド要求でアカウント トークンを提供するアクティベート可能なクラスへのエントリ ポイントを指定します

マネージド URL

同じ Web アカウント プロバイダーに所属する追加 URL を指定します。

次のテーブルは、[Web アカウント プロバイダー] 宣言タイプに表示される設定について説明します:

テーブル 70 • マネージド URL

宣言	プロジェクトの種類	説明
マネージド URL	MSIX	同じ Web アカウント プロバイダーの追加 URL の作成を許可します。 URL を追加するには、[**新しいマネージドURLの追加**] ボタンをクリックします。
マネージド URL	MSIX	同じ Web アカウント プロバイダーに所属する追加 URL。有効な HTTPS URL でなくてはなりません。

Windows 共有フォント

この宣言を使って、カスタム フォントをインストールしてシステム上の他のアプリケーションと共有することができます。

- アプリの設定
- Windows 共有フォント

メモ: *1 つの宣言セットにつき 1 つのインスタンスのみ作成することができます。*

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[Windows共有フォント] 宣言タイプに表示される設定について説明します:

テーブル 71・アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめることができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

Windows 共有フォント

システムに登録する 1 つ以上のフォントを指定します。

次のテーブルは、[Windows共有フォント] 宣言タイプに表示される設定について説明します:

テーブル 72 • Windows 共有フォント

宣言	プロジェクトの種類	説明
Windows 共有フォント	MSIX	1 つ以上のフォントをシステムに登録できるようにします。

カスタム宣言

この宣言を使って、InstallShield では提供されていない MSIX 拡張機能を定義することができます。[カスタム宣言] メニューでは、名前空間およびコンテンツを指定して必要な拡張タイプを定義できるオプションが提供されています。InstallShield ビルド プロセスは、MSIX マニフェスト ファイルの準備中にこれらのカスタマイズを考慮に入れます。

- アプリの設定
- プロパティ

アプリの設定

”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。

次のテーブルは、[カスタム宣言] タイプに表示される設定について説明します:

テーブル 73 • アプリの設定

宣言	プロジェクトの種類	説明
アプリケーションの設定	MSIX	”アプリケーション” 設定は、宣言のデフォルト実行可能ファイルを構成する方法を指定します: • **デフォルト アプリケーション** – 定義された実行可能ファイルが [アプリケーション] ビューのアプリに使用されます。 • **カスタム設定** – デフォルト実行可能ファイルを宣言に指定することができます。 • **スタート ページ** – 処理を行う Web ページを拡張ポイントに指定することができます。
実行可能ファイル	MSIX	”.exe” で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言のデフォルト実行可能ファイルを指定します。指定されなかった場合、実行可能ファイルの値がアプリに使用された [アプリケーション] ビューから取得されます。

テーブル 73 • アプリの設定

宣言	プロジェクトの種類	説明
エントリ ポイント	MSIX	1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または * 宣言を処理するタスクを示します。これは、Windows ランタイム型の名前空間で修飾された名前です。実行可能ファイルの値が指定されている場合、アプリのエントリポイントが必須です。
スタート ページ	MSIX	拡張ポイントを処理する Web ページ – 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *
リソース グループ	MSIX	複数タスクの同じホスト プロセスを判別するアプリ定義の名前です。 リソース（たとえば CPU およびメモリ）の管理目的で、拡張アクティベーションを 1 つのグループにまとめことができるタグ。ResourceGroup を設定できる値は柔軟な自由形式です。

プロパティ

これらの設定はカスタム拡張機能の宣言に使用します。このオプションは、ビルトイン拡張タイプで提供されていない拡張機能を定義する時に役立ちます。

テーブル 74 • プロパティ

宣言	プロジェクトの種類	説明
表示名	MSIX	カスタム宣言の表示（区分）名を指定します。
名前空間	MSIX	このカスタム宣言アイテムのパッケージ マニフェストに追加する名前空間を指定します。 たとえば、マニフェスト ファイルに次のように指定して UAP10 スキーマを含みます:xmlns:uap10='http://schemas.microsoft.com/appx/manifest/uap/windows10/10' 複数の名前空間の宣言を定義する場合、区切り記号としてセミコロンを使います。 例: xmlns:uap10='http://schemas.microsoft.com/appx/manifest/uap/windows10/10';xmlns:desktop6='http://schemas.microsoft.com/appx/manifest/desktop/windows10/6'

テーブル 74 • プロパティ

宣言	プロジェクトの種類	説明
XML コンテンツ	MSIX	カスタム宣言のパッケージ マニフェストに追加する XML コンテンツを指定します。次の例は、installedLocationVirtualization 拡張を宣言します: <uap10:Extension Category="windows.installedLocationVirtualization"> <uap10:InstalledLocationVirtualization> <uap10:UpdateActions ModifiedItems="keep" DeletedItems="keep" AddedItems="keep" /> </uap10:InstalledLocationVirtualization> </uap10:Extension>

パッケージの宣言

プロジェクト• *[パッケージ宣言] ビューは、MSIXのプロジェクト タイプで使用できます:*

この宣言を使って、次のようなパッケージの宣言を追加できます:

- 証明書
- エンタープライズ データの保護
- InProcessServer
- カスタム宣言

証明書

この宣言を使って、windows.certificates タイプのパッケージ拡張ポイントを宣言できます。このアプリには、指定の証明書ストアからの1 つ以上の証明書が必要です。

メモ• *1 つのパッケージにつき 1 つのインスタンスのみ作成することができます。*

- プロパティ
- 証明書

プロパティ

このビューを使って、windows.certificates タイプのパッケージ宣言プロパティを宣言および構成することができます。証明書宣言を使って、パッケージが信頼されたルート証明書などのデジタル証明書をアプリと共にインストールできるようにします。これらの証明書は、SSL を介した Web サービスへのセキュリティで保護された接続に使用されます。

次のテーブルは、[証明書パッケージ] 宣言に表示される設定について説明します:

テーブル 75 • プロパティ

宣言	プロジェクトの種類	説明
排他的信頼	MSIX	パッケージの証明書が排他的でありるかどうか **[はい]** または **[いいえ]** のどちらかを選択し、証明書の有効性を判別します。
選択基準	MSIX	クライアント認証時にパッケージに定義される証明書の選択基準を定義します。 • **自動選択** – 証明書が自動的に選択される場合、このオプションを選択します。 • **ハードウェアのみ** – 指定された証明書がハードウェア固有の場合はこのオプションを選択します。

証明書

windows.certificates タイプのパッケージ拡張ポイントを宣言します。アプリケーション固有の証明書ストアにインストールされる証明書を指定します。

次のテーブルは、[証明書パッケージ] 宣言に表示される設定について説明します:

テーブル 76 • 証明書

宣言	プロジェクトの種類	説明
証明書	MSIX	• **証明書** – ここで 1 つ以上の証明書を宣言します。

エンタープライズ データの保護

この宣言を使って、自動暗号化を安全に使用することができ、Windows 情報保護 (WIP) ポリシーを介したデバイス登録なしでこれを管理できることを宣言します。

メモ: *1 つのパッケージにつき 1 つのインスタンスのみ作成することができます。*

- プロパティ

プロパティ

自動暗号化を安全に使用することができ、Windows 情報保護 (WIP) ポリシーを介したデバイス登録なしで管理できることを宣言します。

この拡張機能は、アプリがビジネス専用である場合および/またはアプリがビジネス以外の個人データに対して行った変更を復元できるメカニズムが提供されている場合のみ使用できます。アプリが作成または変更するすべてのファイルは、デバイスから保護ポリシーが削除されるとき、管理組織に代わって暗号化され、選択的に消去された後はアクセス不可能となります。

重要・*Windows 情報保護を使って管理する場合、ビジネスおよび個人データの両方を安全に処理するため、常に enterpriseDataPolicy アプリ機能を使用してください。*

次のテーブルは、[エンタープライズ データの保護パッケージ] 宣言に表示される設定について説明します:

テーブル 77・プロパティ

宣言	プロジェクトの種類	説明
保護モード	MSIX	データ保護のモード。 ・ **自動** ・ **デフォルト**

InProcessServer

この宣言を使って、windows.activatableClass.inProcessServer タイプのパッケージ拡張ポイントを宣言できます。1 つ以上のアクティベート可能なクラスを公開するアプリはダイナミック リンク ライブラリ (DLL) を使用します。

- プロパティ
- アクティベート可能なクラス

プロパティ

windows.activatableClass.inProcessServer タイプのパッケージ拡張ポイントを宣言します。1 つ以上のアクティベート可能なクラスを公開するアプリはダイナミック リンク ライブラリ (DLL) を使用します。

次のテーブルは、[InProcessServer パッケージ] 宣言に表示される設定について説明します:。

テーブル 78・プロパティ

宣言	プロジェクトの種類	説明
パス	MSIX	インプロセス DLLへのパス1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、:、%、“、|、?、または *

アクティベート可能なクラス

拡張ポイントに関連付けられているアクティベート可能なランタイム クラスを宣言します。

次のテーブルは、[InProcessServer パッケージ] 宣言に表示される設定について説明します:

テーブル 79・アクティベート可能なクラス

宣言	プロジェクトの種類	説明
クラス ID	MSIX	オペレーティング システムのランタイム クラスの識別子。

テーブル 79 • アクティベート可能なクラス

宣言	プロジェクトの種類	説明
スレッドモデル	MSIX	ランタイム クラスでサポートされているスレッド モデルの種類。この属性には次の値のどれかひとつが可能です: • 両方 • シングル スレッド アパートメント (STA) • マルチスレッド アパートメント (MTA)
アクティベート可能なクラス属性	MSIX	1 つ以上のアクティベート可能なクラス属性を定義します。
アクティベート可能なクラス属性	MSIX	Windows ランタイム プロパティ ストアに格納されているクラスの属性インスタンスを定義します。
名前	MSIX	属性名。1 文字から 255 文字までの英数字。アルファベット文字で始める必要があります。
Type	MSIX	属性タイプこの属性には次の値のひとつが可能です: • string • 整数
値	MSIX	属性値1 文字から 255 文字までの文字列。

カスタム宣言

この宣言を使って、InstallShield では提供されていない MSIX 拡張機能を定義することができます。[カスタム宣言] メニューでは、名前空間およびコンテンツを指定して必要な拡張タイプを定義できるオプションが提供されています。InstallShield ビルド プロセスは、MSIX マニフェスト ファイルの準備中にこれらのカスタマイズを考慮に入れます。

- プロパティ

プロパティ

これらの設定はカスタム拡張機能の宣言に使用します。このオプションは、ビルトイン拡張タイプで提供されていない拡張機能を定義する時に役立ちます。

テーブル 80 • プロパティ

宣言	プロジェクトの種類	説明
表示名	MSIX	カスタム宣言の表示 (区分) 名を指定します。

宣言	プロジェクトの種類	説明
名前空間	MSIX	このカスタム宣言アイテムのパッケージ マニフェストに追加する名前空間を指定します。 たとえば、マニフェスト ファイルに次のように指定して UAP10 スキーマを含みます:xmlns:uap10='http://schemas.microsoft.com/appx/manifest/uap/windows10/10' 複数の名前空間の宣言を定義する場合、区切り記号としてセミコロンを使います。 例: xmlns:uap10='http://schemas.microsoft.com/appx/manifest/uap/windows10/10';xmlns:desktop6='http://schemas.microsoft.com/appx/manifest/desktop/windows10/6'
XML コンテンツ	MSIX	カスタム宣言のパッケージ マニフェストに追加する XML コンテンツを指定します。次の例は、installedLocationVirtualization 拡張を宣言します: <uap10:Extension Category="windows.installedLocationVirtualization"> <uap10:InstalledLocationVirtualization> <uap10:UpdateActions ModifiedItems="keep" DeletedItems="keep" AddedItems="keep" /> </uap10:InstalledLocationVirtualization> </uap10:Extension>

[コンテンツ URI] ビュー

プロジェクト: *[コンテンツ URI] ビューは、MSIXのプロジェクト タイプで使用できます:*

このページでは、window.external.notify を使って Script-Notify イベントをアプリに送る URI を指定します。URI には、サブドメイン名にワイルドカード文字を含めることができます (たとえば、https://*.revenera.com or https://*.*.revenera.com)

[コンテンツ URI] ビューは、システムの位置情報にアクセス可能な Web コンテキストでページを指定し、(アプリがこの機能へのアクセス許可を持つ場合) クリップボードにアクセスします。

1 つ以上の規則が定義されている場合、規則の順番が重要になります。

Web リソースに URI との一致属性を定義するには、セキュリティで保護された “https:” サイトだけを指定でき、保護されていない “http:” サイトを使用することはできません。

[編成] ビュー

プロジェクト: *[編成] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

何かを構築する場合、最初にしっかりした基礎を築く必要があります。プロジェクトのベースは、[一般情報] ビューを使用して、アプリケーション情報を指定し、(インストール プロジェクトの場合) [機能] ビューで機能を作成することにより形成されます。

セットアップのデザイン

プロジェクト: *[セットアップのデザイン] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

インストールをデザインする際の最も重要なステップは、インストールのさまざまな構成要素 (インストールの「ビルド ブロック」) をレイアウトすることです。開発者の視点からだけでなく、エンド ユーザーの視点からも検討する必要があります。[セットアップのデザイン] ビューでは、セットアップとそれに関連付けられたすべての機能、コンポーネント、および再配布可能ファイルを見ることができます。

機能

プロジェクト: *[機能] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

機能は、インストールのビルド ブロックです。これらは、プログラムファイル、ヘルプファイル、クリップアートなど、アプリケーションの特徴的な部分をエンド ユーザーに表示します。機能やサブ機能は、[機能] ビューで作成できます。

コンポーネント

プロジェクト: *[コンポーネント] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントは、ファイル、レジストリ エントリ、論理グループへのショートカットなど、類似したアプリケーションデータを組織化するインストール オーサリング ツールです。機能とは異なり、コンポーネントは開発者から見た場合のプロジェクトの構成要素です。

セットアップの種類

プロジェクト: *[セットアップの種類] ビューは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

セットアップの種類は、複数の構成のアプリケーションを顧客に提供します。これらの構成には、一般的に [完全] および [カスタム] が含まれます。セットアップの種類では、ユーザーのニーズに一番ふさわしい構成を選択することができます。このビューは、InstallScript および InstallScript MSI セットアップ プロジェクトでのみ使用できます。

パッケージ

プロジェクト: *[パッケージ] ビューは、次のプロジェクトタイプで使用できます:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[パッケージ] ビューを使って、アドバンスト UI またはスイート/アドバンスト UI インストールに含めるパッケージおよび InstallShield 前提条件を追加できます。このビューではまた、各パッケージの条件を定義したり、各パッケージを 1 つ以上のアドバンスト UI またはスイート/アドバンスト UI 機能と関連付けたり、各パッケージについてその他の設定を構成したりできます。

DIM リファレンス

プロジェクト: *[DIM リファレンス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*

[DIM リファレンス] ビューでは、プロジェクト内のすべての DIM リファレンスを管理したり、参照された DIM 専用の具体的な手順を表示したり、それぞれの参照された DIM を 1 つまたは複数の機能に関連付けたりできます。このビューでは、基本の MSI インストールのシーケンスで DIM のカスタム アクションとダイアログをスケジュールすることもできます。

[セットアップのデザイン] ビュー

プロジェクト: *[セットアップのデザイン] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *トランスフォーム*

インストールをデザインする際の最も重要なステップは、インストールのさまざまな構成要素をレイアウトすることです。開発者の視点からだけでなく、エンド ユーザーの視点からも検討する必要があります。

[セットアップのデザイン] ビューで、アプリケーションのインストール階層全体を視覚的にデザインすることができます。これは、コンポーネントと機能およびサブ機能を関連付けることができる唯一のビューです。

機能

機能は、エンド ユーザーから見たアプリケーションの構成要素です。それらは、エンド ユーザーの選択に応じてインストールまたはアンインストールできます。アプリケーション全体は、特定の目的を実行する別々の機能に分けることができます。機能は、セットアップのデザインビューと機能ビューの両方で作成できます。

コンポーネント

コンポーネントによって、アプリケーションデータをグループ化できます。機能とは異なり、コンポーネントは、プロジェクトを開発者側の視点から見た構成であり、ファイル、レジストリ エントリ、およびショートカットなどのデータを含みます。コンポーネントは、セットアップのデザインビューで機能に関連付けられます。1 つのコンポーネントを複数の機能に割り当てることもできます。コンポーネントは、セットアップのデザインビューで機能に関連付けられます。1 つのコンポーネントを複数の機能に割り当てることもできます。

ヒント: *コンポーネントセクションにはそれ用の詳細設定のほかにファイル、レジストリ エントリ、ショートカットノードが含まれています。データを含まない下位ノードを表示しない場合、[セットアップのデザイン] をクリックし、**データを含むノードのみ表示**を選択します。このオプションを選択すると、データを含むノードのみが表示されます。*

DIMs

デベロッパー インストール マニフェスト (DIM) は、基本の MSI プロジェクトに組み込むと完全な製品インストールを構成する個別のインストール プロジェクトです。製品をロジックにしたがって個別のユニットに分割することで、コラボレーションや分散型のインストール開発を促進し、それらの再利用が可能になります。

再配布可能ファイル

再配布可能ファイルを利用して、既存の機能を個別にインストールすることができます。たとえば、アプリケーションに Visual Basic ランタイム .dll ファイルが必要な場合、ファイルをインストール プロジェクトに追加してインストールの必要条件を決める代わりに、Visual Basic Virtual Machine マージ モジュールを含めることもできます。

メモ: *[再配布可能ファイル] ビューを使って、インストールに InstallShield 前提条件、マージ モジュール、およびオブジェクトを含む再配布可能ファイルを追加できます (InstallScript プロジェクトの場合は [前提条件] ビュー、InstallScript および InstallScript オブジェクト プロジェクトの場合は [オブジェクト] ビューを使用)。*

ファイル、レジストリデータおよびショートカット

ファイル、レジストリデータ、およびショートカットといった要素をコンポーネントに追加して階層を完成させます。

詳細設定

コンポーネントの詳細設定を使うと、特別な要件を持つコンポーネントのインストールを処理することができます。詳細設定を指定することによって、コンポーネントのパブリッシュ、および COM サーバー、ファイル拡張サーバー、および MIME の種類の登録などができます。コンポーネントの詳細設定を使って、レジストリにアプリケーションパスのエントリを作成することもできます。

[機能] ビュー

プロジェクト: *[機能] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

機能は、エンド ユーザーから見て、個別にインストール可能な最小の製品構成単位です。これは、ヘルプ ファイルや製品スイートの一部などの製品の特定機能を表し、エンド ユーザーがこれらをインストールまたはアンインストールするかどうかを決定できます。インストール全体は特定の目的を実行する各機能ごとに分ける必要があります。

サブ機能は機能をさらに分割したものです。機能は、ユーザーが選択してインストールできる製品または製品スイートに含まれた自己完結型の要素である必要があります。そのため、インストールの各部分をある "親" 機能のサブ機能として構成するのが、合理的な方法と言えます。

ヒント: *多くのサブ機能を作成できますが、編成のためにデザインを可能な限り単純にしておくようお勧めします。*

機能の設定

プロジェクト: *機能の設定は、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *MSI データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

[機能] ビューの設定 (および [セットアップのデザイン] ビューの機能の設定) は、以下の主要なカテゴリに分かれています:

- 全般
- 機能イベント
- 実行時の設定

全般

[機能] ビューまたは [セットアップのデザイン] ビューで機能を選択すると、[一般] 領域で以下の設定を構成できます。

テーブル 81 • 機能の全般設定

設定	プロジェクトの種類	説明
表示名	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	CustomSetup ダイアログ (またはアドバンスト UI またはスイート/アドバンスト UI プロジェクトの場合、InstallationFeatures ウィザード ページ) でこの機能に対して表示する名前を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *アドバンスト UI およびスイート/アドバンスト UI プロジェクトの場合: この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。*

テーブル 81・機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
説明	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、MSI データベース、スイート/アドバンスト UI、トランスフォーム	機能の簡単な説明を入力します。機能の説明は、エンドユーザーが機能またはサブ機能をクリックしたときに CustomSetup ダイアログ (または、アドバンスト UI およびスイート/アドバンスト UI インストールの場合は InstallationFeatures ウィザード ページ) に表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト・** アドバンスト UI およびスイート/アドバンスト UI プロジェクトの場合: この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。*
リモート インストール	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	機能のデフォルトのインストール状態を示します。有効なオプションは次のとおりです: ・ **ソースを優先する**—この機能のファイルは、デフォルトで CD-ROM またはネットワークの場所といったソース メディアから直接実行されます。 ・ **ローカルを優先する**—この機能のファイルは、デフォルトでターゲット システム上にインストールされます。 ・ **親を優先する**—サブ機能のデフォルト状態は、親機能のそれと同じです。このオプションは、機能がサブ機能である場合にのみ使用できます。 詳細については、「機能の "リモート インストール" 設定を設定する」を参照してください。 ***注意・** 機能のコンポーネントに Windows サービスが含まれる場合は、[ローカルを優先する] を選択します。エンド ユーザーは CustomSetup ダイアログを使用してインストール状態を変更することができますが、Windows Installer はサービスをリモート インストールすることはできません。*

テーブル 81 • 機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
Destination	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	機能のファイルをインストールするターゲット システム上にあるフォルダーを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。 ***メモ•*** *インストール先を実行時に構成できるようにするには、選択するインストール先フォルダーに、必ずパブリック プロパティ (すべて大文字で表記) を選択してください。* 詳細については、「機能のインストール先を設定する」を参照してください。
インストール レベル	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	この機能のインストール レベルに整数を入力します。エンド ユーザーが CustomSetup ダイアログで機能を選択解除しない限り、パッケージの INSTALLLEVEL プロパティの値と同等またはそれ以下のインストールレベルの機能がすべてインストールされます。プロジェクト全体の INSTALLLEVEL プロパティは、プロパティ マネージャーで変更できます。 "インストールレベル" 設定は、実行時に INSTALLLEVEL プロパティと比較され、どの機能をインストールできるかを決定します。この設定を使って、特定の機能設定を作成することもできます。 ***プロジェクト•*** *InstallScript MSI インストールでは、プロジェクトのセットアップの種類が定義されていない場合にのみ、この設定が使用されます。*

テーブル 81・機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
ディスプレイ	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	CustomSetup ダイアログ (基本の MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) または機能選択ダイアログ (InstallScript MSI プロジェクトの場合) でエンド ユーザーに対して機能を表示する方法を指定します。 ・ **閉じて表示する** – 機能は実行時ダイアログに、デフォルトでサブ機能が閉じた状態で、表示されます。 ・ **展開して表示する** – 機能は実行時ダイアログに、デフォルトでサブ機能が展開した状態で、表示されます。 ・ **非表示** – 実行時ダイアログに、機能およびそのサブ機能は表示されません。 この設定によって、機能がインストールされるかどうかについて、直接影響を及ぼすことはありません。非表示にした機能は、自動的にすべてインストールされるというわけではなく、インストールが必要な機能の場合は選択解除できず、インストールするべきではない機能の場合は選択できないようになります。 詳細については、「機能をエンドユーザーへ表示する」を参照してください。
Visible	アドバンスト UI、InstallScript、スイート/アドバンスト UI	機能を InstallScript インストールでは機能の選択ダイアログ、アドバンスト UI およびスイート/アドバンスト UI インストールでは InstallationFeatures ウィザードで表示するかどうかを指定します。 この設定によって、機能がインストールされるかどうかについて、直接影響を及ぼすことはありません。非表示にした機能は、自動的にすべてインストールされるというわけではなく、インストールが必要な機能の場合は選択解除できず、インストールするべきではない機能の場合は選択できないようになります。 ***メモ:*** *不可視の機能の選択状態をエンド ユーザーが変更できる唯一の方法は、不可視の機能を必要とする可視のコンポーネントを選択または非選択にすることによってのみです。そのため、不可視の機能は、すべて 1 つまたは複数の可視の機能に対する必須コンポーネントにする必要があります。これを行わなかった場合、エンド ユーザーは、初めに選択解除された機能を選択またはインストールすることができなくなります。*

テーブル 81・機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
アドバタイズ	基本の MSI、MSI データベース、トランスフォーム	この機能の適切なアドバタイズ オプションを選択します。選択可能なオプションは以下のとおりです: • **アドバタイズを許可する** — エンド ユーザーは、CustomSetup ダイアログで、この機能のアドバタイズ オプションを選択できます。アドバタイズは許可されますが、インストール実行時のデフォルトのオプションではありません。 • **アドバタイズを優先する**—この機能はデフォルトでアドバタイズされます。エンド ユーザーは、CustomSetup ダイアログの機能のアドバタイズ オプションを変更できます。 • **アドバタイズを許可しない** — この機能ではアドバタイズが許可されません。エンドユーザーは、CustomSetup ダイアログで機能のアドバタイズを選択できません。 • **サポートされていない場合にアドバタイズを無効にする** — アドバタイズは、Internet Explorer 4.01 以降のシステムでのみ機能します。この条件を満たさないターゲット システムでは、アドバタイズは許可されません。ターゲット システムがアドバタイズをサポートする場合、アドバタイズは許可されます。 詳細については、「機能のアドバタイズ」を参照してください。
Required	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	ターゲット システムで機能が必須かどうかを示します。機能が必須である場合、エンド ユーザーが CustomSetup ダイアログ (基本の MSI、MSI データベース、およびトランスフォーム プロジェクト) または機能選択ダイアログ (InstallScript MSI プロジェクト) でこれを選択することはできません。 ***プロジェクト:*** *InstallScript MSI プロジェクトでは、この設定は初回インストール中にルート レベルの機能に適用します。アップグレードまたはパッチに含まれるサブ機能にも適用できます。この設定は、初回インストール中、サブ機能には適用しません。*

テーブル 81 • 機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
条件	アドバンスト UI、スイート/アドバンスト UI	この設定を使って、InstallationFeatures ウィザード ページで機能のインストールがデフォルトで選択されるかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。 たとえば、Windows の特定バージョンが搭載されたターゲット システム上では特定の機能をデフォルトで選択する場合、その Windows バージョンを指定する条件を作成します。条件の構成方法によって、エンド ユーザーがそのプラットフォーム上でアドバンスト UI またはスイート/アドバンスト UI インストールを実行するときに InstallationFeatures ウィザード ページでアドバンスト UI またはスイート/アドバンスト UI 機能をデフォルトで選択することができます。エンド ユーザーが機能の選択を解除しない限り、アドバンスト UI またはスイート/アドバンスト UI インストールはその機能をインストールします。 1 つ以上の新しい機能条件を追加するには、この設定で [新しい条件] ボタンをクリックします。InstallShield によって、″条件″ 設定の下に新しい行が追加されます。この行のリストから All、Any、または None のうち適切なオプションを選択します。次にこの行で、[新しい条件] ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。 1 つ以上の条件ステートメントが構成されると、″条件″ 設定には **(条件)** と表示されます。何も構成されていない場合、″条件″ 設定には **(空白)** と表示されます。 各条件設定についての詳細は、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。
リリース フラグ	アドバンスト UI、基本の MSI、InstallScript MSI、スイート/アドバンスト UI	リリース フラグを使って、インストールの異なるビルドにこの機能を含めたり、除外したり選択できるようにするには、この機能を識別する 1 つ以上のリリース フラグを入力します。複数のフラグは、コンマで区切ります。 詳細については、「リリース フラグを機能に割り当てる」を参照してください。

テーブル 81 • 機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
条件	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	この設定を使って、1 つ以上の条件インストール レベルを指定できます。ターゲット システム上で条件が True 評価されると、機能に対応するインストール レベルが与えられます。 実行時に、機能のインストール レベル値が、グローバル パブリック プロパティ INSTALLLEVEL の値と比較され、デフォルトで機能をインストールかどうかが判断されます。 1 つ以上の条件を指定するには、この設定で省略記号ボタン (...) をクリックします。詳細については、「機能の条件を設定する」を参照してください。 条件を追加すると、InstallShield によって、機能設定のグリッドに新しい行が追加されます。新しい行は、機能のインストール レベルが、追加した対応する条件ステートメントと共に表示されます。
コメント	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	機能に対する変更の履歴を残したり、将来の参照のためにコメントを入力します。機能のコメントはプロジェクトファイルにのみ保存され、インストールでは一切使用されません。
必要な機能	InstallScript、InstallScript MSI、InstallScript オブジェクト	選択された機能がインストールされるときに、1 つ以上の他の機能もインストールする必要がある場合、省略記号ボタン (...) をクリックして、他の必要な機能を指定します。詳細については、「"必要な機能" 設定を使用する」を参照してください。
UI 選択の変更を許可	アドバンスト UI、スイート/アドバンスト UI	エンド ユーザーが InstallationFeatures ページに表示されている機能を選択または選択解除できるようにするかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい** – エンド ユーザーがウィザード インターフェイスを使って機能を選択または選択解除できます。デフォルトでは、これが設定されています。 • **いいえ** – エンド ユーザーがウィザード インターフェイスを使って機能を選択または選択解除できません。InstallationFeatures ウィザード ページでこの機能が表示されるときは、読み取り専用の状態です (機能の条件などのその他の要因によって選択/非選択となります)。

テーブル 81・機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
親の状態に従う	アドバンスト UI、スイート/アドバンスト UI	この設定はサブ機能で使用できますが、ルート レベルの機能では使用できません。 親機能の状態によって、サブ機能の状態を決定するかどうかを示します。選択可能なオプションは以下のとおりです: ・ **はい** – 選択されたサブ機能は、親機能がインストールされているかどうかに従ってインストールされます。 ・ **いいえ** – 選択されたサブ機能の親機能がインストールされている場合、選択されたサブ機能はインストールまたはインストールされていない状態のどちらも可能です。デフォルトでは、これが設定されています。 選択されたサブ機能の親機能がインストールされていない場合、選択されたサブ機能をインストールすることはできません。 この設定は、InstallationFeatures ウィザード ページ上でサブ機能を非表示の状態にするときに使用できますが、親機能は表示された状態です。
ステータス テキスト	InstallScript、InstallScript オブジェクト	機能のステータステキストを指定します。エンドユーザーはこの機能の転送中、進行状況を示すインジケーターの最上行にこのテキストが表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 81 • 機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
ファイルの必要性	InstallScript、InstallScript オブジェクト	[カスタム] セットアップ タイプで、エンド ユーザーによる機能の選択解除についての許可レベルを示します。選択可能なオプションは以下のとおりです: • **標準** – エンド ユーザーは、機能を選択解除することが可能です。エラー メッセージは表示されません。 • **強く推奨** – エンド ユーザーに、機能の選択を強く推奨します。 • **重要** – エンド ユーザーに、機能の選択を解除しないように警告します。 スクリプトは、InstallScript 関数 **FeatureGetData** と FEATURE_FIELD_FILENEED 定数を使って、この設定を読み取ります。戻り値によって、重要または強く推奨される機能が選択解除された場合に、スクリプトがエンド ユーザーに対してアラートを表示します。
ビルドに含める	InstallScript、InstallScript オブジェクト	ビルド時に機能をリリースに含めるかどうかを指定します。 リリース ウィザードの [機能] パネルで、プロジェクト内の任意の機能に対するこの設定をオーバーライドできます。
パスワード	InstallScript	機能をパスワードで保護するには、パスワードを入力します。 各機能に異なるパスワードを指定できます。インストール全体にパスワードを指定することもできます ([リリース] ビューの [Setup.exe] タブにある "メディア パスワード" 設定を使用)。 デフォルト スクリプトは、機能のパスワードを自動的に検証しませんので、ご注意ください。スクリプトを変更して、パスワードを検証しなくてはなりません。そのためには、エンド ユーザーにパスワードの入力をプロンプトするダイアログ関数 (たとえば、**AskText**) を挿入してから、スクリプト内部から **FeatureValidate** 関数を呼び出します。 ***注意** • このパスワードは、巧妙なハッカーにより、リリースから抽出される可能性があります。セキュリティが重要な課題である場合、このパスワード保護機能だけに頼ることをお勧めしません。*

テーブル 81 • 機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
暗号化	InstallScript	機能のファイルを暗号化するかどうか指定します。暗号化されたファイルは、InstallScript エンジン (または経験豊富なハッカー) しか読み取ることができません。 配布メディアから製品を実行するオプションを直接エンドユーザーに提供する予定の場合、関連するファイルを暗号化しないでください。InstallShield は、エンド ユーザーのハード ドライブにインストール中のみファイルの暗号化を解除します。 暗号化は、パスワード保護とは異なります。ファイルの暗号化は、ファイルをスクランブルし、テキストエディターで開かれた場合に、文字列や他のデータを読み取って製品に関する機密情報が取得できないようにします。機能を保護するパスワードは、パスワードを知らない人によってアプリケーションがインストールされるのを防ぎます。機能のファイルを暗号化したり、機能をパスワード保護したり、または両方を行って最も高いセキュリティで保護することができます。 機能のファイルが暗号化されており、機能とリリースがいずれもパスワード保護されていない場合、配布メディア (またはコピー) を取得する人は、誰でもその機能をインストールできます。機能がパスワード保護されていても、ファイルが暗号化されていない場合、そのファイルをテキストエディターで開いて、機密情報 (機能のパスワードが含まれる可能性がある) を読み取ることができます。

テーブル 81・機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
CD-ROM フォルダー	InstallScript、InstallScript オブジェクト	[リリース] ウィザードの [メディア レイアウト] パネルで [CD_ROM フォルダー] オプションが選択されている場合、または、リリース ウィザードの [カスタム メディア レイアウト] パネルで機能のチェック ボックスが選択されている場合に、機能のファイルを配置するディスク イメージのフォルダーを指定します。これらの条件の 1 つが満たされているが、フォルダーが指定されていない場合、機能はディスクイメージのルートに配置されます。もしくは、サブ機能である場合、親機能と同じフォルダーになります。 CD-ROM のフォルダー名は、ISO 9660 (レベル 1) 標準に準拠する必要があります。つまり、フォルダー名は8文字以下 (拡張子を除く) で、A-Z、0-9、および _ (アンダースコア) のみを使用します。 CD-ROM にフォルダー構造を配置することにより、CD-ROM から製品を実行することが可能になります。CD-ROM 上のフォルダー構造は、ハードドライブにインストールするファイル構造と同じである必要はありません。必要なのは、製品が両方のフォルダー構造で機能するということだけです。 この機能は、一般的に CD-ROM 配布にのみ使用されます。他の配布メディアでは、通常全ての機能を .cab ファイルに圧縮します。 ***プロジェクト:*** *InstallScript オブジェクトプロジェクトの場合、オブジェクトがインストール プロジェクトに含まれる時、この設定の値はオブジェクトのファイルが配置される場所には影響しません。*
含められたコンポーネント	InstallScript、InstallScript オブジェクト	この読み取り専用設定には、機能に関連付けられたコンポーネントのリストが表示されます。 インストールでは、コンポーネントが機能に関連付けられていることが非常に重要です。そうでない場合、ファイルが一切インストールされません。[セットアップのデザイン] ビューを使って、コンポーネントを機能と関連付けます。詳細については、「コンポーネントと機能の関連付け」を参照してください。

テーブル 81 • 機能の全般設定 (続き.)

設定	プロジェクトの種類	説明
GUID	InstallScript、InstallScript オブジェクト	この機能を一意に識別する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ({...}) をクリックします。 この機能 GUID は、インストールのログ ファイルにある機能を識別します。アップグレードで機能の GUID を変更すると、InstallScript エンジンは既存の機能を更新するのではなく、これを新規機能として取り扱います。

機能イベント

InstallScript、InstallScript MSI、または InstallScript オブジェクト プロジェクトの [機能] ビューまたは [セットアップのデザイン] ビューで機能を選択するとき、[機能イベント] 領域では、以下の設定を構成できます:

テーブル 82 • 機能における機能イベントの設定

設定	プロジェクトの種類	説明
OnInstalling	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定では、この機能をインストールする前に呼び出す InstallScript 関数を指定できます。リストから適切なファイルを選択します。 エクスポートされた関数のみがドロップダウン リストに表示されます。エクスポートされた関数には、次のプロトタイプがあります。 export prototype FunctionName( );
OnInstalled	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定では、この機能をインストールした後で呼び出す InstallScript 関数を指定できます。リストから適切なファイルを選択します。 エクスポートされた関数のみがドロップダウン リストに表示されます。エクスポートされた関数には、次のプロトタイプがあります。 export prototype FunctionName( );
OnUninstalling	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定では、この機能をアンインストールする前に呼び出す InstallScript 関数を指定できます。リストから適切なファイルを選択します。 エクスポートされた関数のみがドロップダウン リストに表示されます。エクスポートされた関数には、次のプロトタイプがあります。 export prototype FunctionName( );

テーブル 82・機能における機能イベントの設定 (続き.)

設定	プロジェクトの種類	説明
OnUninstalled	InstallScript、InstallScript MSI、InstallScript オブジェクト	この設定では、この機能をアンインストールした後で呼び出す InstallScript 関数を指定できます。リストから適切なファイルを選択します。 エクスポートされた関数のみがドロップダウン リストに表示されます。エクスポートされた関数には、次のプロトタイプがあります。 export prototype FunctionName( );

実行時の設定

InstallScript、InstallScript MSI、または InstallScript オブジェクト プロジェクトの [機能] ビューまたは [セットアップのデザイン] ビューで機能を選択するとき、[ランタイム設定] 領域では、以下の設定を構成できます:

テーブル 83・機能におけるランタイムの設定

設定	プロジェクトの種類	説明
FTP ロケーション	InstallScript、InstallScript MSI、InstallScript オブジェクト	FTP の場所と機能を関連付けるには、次の形式で FTP アドレスを入力します: **ftp.domain.com** 入力したアドレスへは、**FeatureGetData** 関数と FEATURE_FIELD_FTPLOCATION 定数を使って、スクリプト内からアクセスできます。
HTTP ロケーション	InstallScript、InstallScript MSI、InstallScript オブジェクト	HTTP アドレスの場所と機能を関連付けるには、次の形式で HTTP アドレスを入力します: **http://www.domain.com** 入力したアドレスへは、**FeatureGetData** 関数と FEATURE_FIELD_HTTPLOCATION 定数を使って、スクリプト内からアクセスできます。
その他	InstallScript、InstallScript MSI、InstallScript オブジェクト	テキスト文字列を機能と関連付けるには、そのテキスト文字列を入力します。 入力したテキスト文字列へは、**FeatureGetData** 関数と FEATURE_FIELD_MISC 定数を使って、スクリプトからアクセスできます。

[コンポーネント] ビュー

プロジェクト: *[コンポーネント] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントは、ファイル、レジストリ エントリ、論理グループへのショートカットなど、類似したアプリケーションデータを組織化するのに役立つインストール作成要素です。機能とは異なり、コンポーネントは開発者から見た場合のプロジェクトの構成要素です。

コンポーネントは、[セットアップのデザイン] ビュー(インストール プロジェクトの場合)または [コンポーネント] ビューで修正できます。

ヒント: *[コンポーネント] エクスプローラーには、各コンポーネントのファイルやレジストリ データなどが含まれます。データを含まない下位ノードを表示しない場合、エクスプローラーのトップレベルを右クリックしてから、[データがあるノードのみ表示] をクリックします。このコマンドを選択すると、データを含むノードのみが表示されます。*

メモ: *サブフォルダーとダイナミックリンクされたファイルが、事前に設定した数を超過した場合、表示パフォーマンスの向上のため、ファイルビューのリストは省略表示になります。この状態のファイルリストでは、先頭項目に **リストが切り捨てられました** と表示されます。サブフォルダーに含まれるファイルはすべて、プロジェクトにビルドされます。*

コンポーネントと機能の関係

コンポーネントは、[セットアップのデザイン] ビューで機能に関連付けられます。詳細については、「新しいコンポーネントを機能に関連付ける」を参照してください。

コンポーネントの設定

コンポーネントをクリックしたときに表示される各設定についての詳細は、「コンポーネントの設定」を参照してください。

コンポーネントの詳細設定

コンポーネントについて構成可能な詳細設定については、「コンポーネントの詳細設定」を参照してください。

コンポーネントの設定

***プロジェクト・** コンポーネントの設定は、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューの設定 (および [セットアップのデザイン] ビューのコンポーネントの設定) は、以下の主要なカテゴリに分かれています:

- 全般
- ターゲット マシン
- .NET の設定
- 実行時の設定

全般

[コンポーネント] ビューまたは [セットアップのデザイン] ビューでコンポーネントを選択すると、[全般] 領域で以下の設定を構成できます。

テーブル 84 • [全般] 領域にあるコンポーネントの設定

設定	プロジェクトの種類	説明
インストール先	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	コンポーネントのファイルをインストールするターゲット システム上にあるフォルダーを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。 製品のデフォルトのインストール先フォルダーにファイルをインストールするには、使用中のプロジェクトの種類に従って、適切な場所を選択します: • **[INSTALLDIR]**—基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、またはトランスフォーム プロジェクト • **<TARGETDIR>**—InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクト ***プロジェクト:*** *基本の MSI、InstallScript MSI、MSI データベース、およびトランス フォーム プロジェクトで、インストール先を実行時に構成できるようにするには、選択するインストール先フォルダーに、必ずパブリック プロパティ (すべて大文字で表記) を選択してください。* *一部のプロジェクト タイプ (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム) では、コンポーネントの機能のインストール先フォルダーを設定することもできます。コンポーネントだけでなく、機能のインストール先を設定した場合の効果については、「コンポーネントのインストール先と機能のインストール先の違い」を参照してください。* *InstallScript と InstallScript オブジェクト プロジェクトでは、ファイルを WINSYSDIR64 にインストールする場合、場合によって、コンポーネントの "64 ビットのコンピューター" 設定で [はい] を選択する必要があります。[はい] を選択しなかった場合、SysWOW64 にインストールされるファイルが、32 ビット Windows システム フォルダーにリダイレクトされる可能性があります。詳細については、「64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする」を参照してください。*

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
保存先のアクセス許可	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	コンポーネントのインストール先フォルダーのアクセス許可を設定するには、この設定の省略記号ボタン (...) をクリックします。 ***メモ:*** *特定のアクセス許可を割り当てる必要がある場合を除き、通常のユーザーのアクセス許可がインストール先フォルダーに適用できるように、この設定を未定義のままにしておくことをお勧めします。*
コンポーネント コード	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このコンポーネントを一意に識別する GUID を入力します。InstallShield 使って、自動的に異なる GUID を作成するには、この設定の [新しい GUID の生成] ボタン ([...]) をクリックします。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
共有	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	 ***プロジェクト•*** *この設定の動作は、プロジェクトが Windows Installer ベースのプロジェクト (基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、またはトランスフォーム プロジェクト) か、InstallScript ベースのプロジェクト (InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクト) かによって多少異なります。* • **Windows Installer ベースのプロジェクト** – このコンポーネントの キー ファイル を共有としてマークするかどうか指定します。インストールがキー ファイルをインストールする場合で、そのコンポーネントの ″共有″ 設定で [はい] が選択されているとき、レジストリに参照カウントが既存しない場合はそれが作成され、既存する場合はそのカウントが増加されます。 • **InstallScript ベースのプロジェクト** – このコンポーネントを共有とマークするかどうか指定します。インストールがコンポーネントのファイルをインストールする場合で、コンポーネントの ″共有″ 設定で [はい] が選択されているとき、レジストリに各コンポーネントのファイルの参照カウントが既存しない場合はそれが作成され、既存する場合はそのカウントが増加されます。 コンポーネントを共有としてマークしているいないにかかわらず、インストーラーによってコンポーネントのすべてのファイルの参照カウントが増分されますので、ご注意ください。ただし、参照カウントが存在しない場合、この ″共有″ 設定が [はい] に設定されていない限り、インストールはこれを作成しません。 詳細については、「共有ファイルの参照カウントを管理する」を参照してください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
パーマネント	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このコンポーネントをパーマネントとしてマークするかどうかを指定します。[はい] を選択すると、コンポーネントの機能がアンインストールされるときに、コンポーネントのデータ (ファイル、レジストリ エントリ、ショートカットなど) はいずれもターゲット システムから削除されません。 検証規則では、[SystemFolder] にインストールされるコンポーネントすべてに [はい] を選択することが必要です。 ***メモ•*** *フォントをインストールする場合、[はい] を選択することをお勧めします。*
アンインストール	InstallScript、InstallScript オブジェクト	コンポーネントの機能がアンインストールされるときに、コンポーネントのファイル、レジストリ エントリ、およびその他のデータをアンインストールするかどうかを指定します。 この設定は、他の製品によって使用される可能性のあるファイルのアンインストールを防ぐのに便利です。コンポーネントのファイルが他の製品によって使用されない場合、"アンインストール" 設定は、通常 [はい] に設定します。
条件	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定を使って、ターゲット システム上でコンポーネントのデータをセットアップする前にインストールが評価を行うステートメントを入力します。このコンポーネントは、条件が False に評価された場合にはインストールされません。条件が True と評価された場合には、インストールする機能が選択されているとして、コンポーネントがインストールまたはアドバタイズされます。 コンポーネントの条件を作成するには、この設定で省略記号ボタン (...) をクリックします。詳細については、「コンポーネント条件を構成する」を参照してください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
リモート インストール	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	コンポーネントのデフォルトのインストール状態を示します。有効なオプションは次のとおりです: • **ソースを優先する** – このコンポーネントのファイルは、デフォルトで CD-ROM またはネットワークの場所といったソース メディアから直接実行されます。 • **ローカルを優先する** – このコンポーネントのファイルは、デフォルトでターゲット システム上にインストールされます。 • **オプション** – コンポーネントは、その機能の "リモート インストール" 設定を使用します。 詳細については、「コンポーネントの "リモート インストール" 設定を構成する」を参照してください。 ***注意 •*** *コンポーネントに Windows サービスが含まれる場合、[ローカルを優先する] オプションを選択します。エンド ユーザーは CustomSetup ダイアログを使用して機能のインストール状態を変更することができますが、Windows Installer はサービスをリモート インストールすることはできません。*

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
ビルド時の COM 抽出	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	ビルド時に InstallShield が自動的にコンポーネントの キー ファイルから COM データを抽出するかどうかを指定します。選択可能なオプションは以下のとおりです: • **いいえ** – ビルド時に COM 情報は抽出されません。 • **はい** – リリースをビルドするたびに、COM 情報が抽出されます。COM サーバーのインターフェイスを変更することがある場合、コンポーネントの [詳細設定] 領域の下にある [COM 登録] 領域で、静的に取得または提供されたデータを保持するために、このオプションを選択しておくことをお勧めします。 コンポーネントの [詳細設定] 領域の下にある [COM 登録] 領域に静的に含まれている情報に従って、InstallShield がファイルを登録するように設定するには、[いいえ] オプションを選択します。この詳細設定には、コンポーネント ウィザードで抽出されたか、または詳細設定に手動で入力された、ファイルの COM クラス、PprogID などの情報が保存されます。 ***ベスト プラクティス** • このコンポーネントのファイルを自己登録としてマークした場合、"ビルド時に COM 抽出" 設定を [いいえ] に設定する必要があります。自己登録関数の呼び出しは、セットアップ ベスト プラクティスの違反であることに注意してください。* COM 登録の詳細については、「ビルド時に COM 登録データを抽出する」を参照してください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
Languages	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コンポーネントの言語設定を使って、ビルド時のフィルタリングに使用する、コンポーネントが依存する言語を指定できます。デフォルトでは、コンポーネントは言語に依存しません。つまり、コンポーネントのデータ (ファイル、レジストリ エントリなど) はどれも特定の言語に固有ではありません。 ***プロジェクト:*** *InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトでは、プロジェクトの [一般情報] ビューで言語が選択されていない場合、その言語はプロジェクトのコンポーネント用に利用可能な言語の一覧には表示されません。* コンポーネントが特定の言語にのみ適用することを指定するには、この設定の省略記号ボタン (...) をクリックします。[言語] ダイアログ ボックスが開き、ここでコンポーネントに適切な言語を選択できます。 詳細については、「コンポーネントを言語依存としてマークする」を参照してください。 ***ヒント:*** *実行時にインストールされる言語依存コンポーネントを指定する方法については、「言語基づいてコンポーネントをインストールする」を参照してください。*
条件の再評価	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	製品が再インストールされるときに、Windows Installer がコンポーネントの条件を再評価するかどうかを指定します。 この設定は、製品が再インストールされるとき、たとえばオペレーティング システムのアップグレード後に相互交換を行う移行コンポーネントをオーサリングするのに役立ちます。詳細については、「再インストール中にコンポーネント条件を再評価する」を参照してください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
上書きしない	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ファイルがターゲット システム上に存在する場合、インストールによるファイルの上書きを避けるかどうかを指定します。 • [はい] を選択すると、(ターゲット システムに存在する場合) ファイルは、ファイルのバージョンに関係なく、決して上書きされません。この設定で [はい] を選択すると、ファイルのバージョン規則がオーバーライドされます。 • [いいえ] を選択しても、ターゲット システムのファイルバージョンがインストールされるバージョンより新しい場合、ターゲット システムのファイルは上書きされません。インストールされるバージョンの方が新しい場合、ターゲット システム上のファイルは 上書きされます 。 ***重要*** *• Windows Installer は、コンポーネントをインストールするかどうか決定する際に、コンポーネントのキーファイルの存在をチェックします。コンポーネントのキーファイルがターゲット システム上に存在しない場合、Windows Installer は、"上書きしない" 設定で [いいえ] が選択されているものとしてコンポーネントをインストールします。Windows Installer がコンポーネントをインストールするかしないかを決定する方法についての詳細は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。*
64 ビットコンポーネント	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	 ***プロジェクト*** *• この設定の使用は、プロジェクトが Windows Installer ベースのプロジェクト (基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、またはトランスフォーム プロジェクト) か、InstallScript ベースのプロジェクト (InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクト) かによって異なります。*

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
64 ビットコンポーネント (続き)		Windows Installer ベースのプロジェクトの場合 このコンポーネントを 64 ビットとしてマークするかどうかを指定します。次のいずれかが True の場合、コンポーネントを、64 ビットとマークする必要があります: • コンポーネントに 64 ビットの場所にインストールする必要があるファイルが含まれている。 • コンポーネントに、64 ビット領域のレジストリにインストールするレジストリ データが含まれている。 • コンポーネントに、自己登録型の必要がある 64 ビットの COM サーバーが含まれている。 次の事項に注意してください。 • 64 ビットのコンポーネントをインストールに含めると、インストールは 32 ビットのマシンで実行できなくなり、"テンプレート概要" プロパティで、64 ビットの値を指定する必要があります。 • コンポーネントが 64 ビットで、それによって 32 ビット コンポーネントが置き換えられる場合は、"コンポーネント コード" 設定で、コンポーネントの新しい GUID を必ず指定してください。 64 ビット サポートに関する詳細は、「64 ビットのオペレーティング システムを基本の MSI および InstallScript MSI インストールを使ってターゲットする」を参照してください。 InstallScript ベースのプロジェクトの場合: このコンポーネントを 64 ビットとしてマークするかどうかを指定します。 インストールで、このコンポーネント内のファイルとレジストリ データを 64 ビット システム上の 64 ビットの場所と 32 ビット システム上の 32 ビットの場所にインストールする場合、コンポーネントを 64 ビットとしてマークします。 64 ビット サポートに関する詳細は、「64 ビットのオペレーティング システムを 32 ビットInstallScript セットアップ ランチャーを使ってターゲットする」を参照してください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
ソースの場所	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コンポーネントのファイルを圧縮しない場合、ソースディスク イメージ上でコンポーネントのファイルが格納されるサブフォルダーの名前を指定します。リリースをビルドすると、InstallShield がコンポーネントのファイルをディスク イメージ上のこのサブフォルダーにコピーします。 この設定を使って、各コンポーネントのファイルに一意のフォルダーを作成できます。これは、複数のコンポーネントに同じ名前を持つ異なるファイルがある場合に必要となります。[ソースの場所] 設定の値を指定しない場合、非圧縮リリースのビルド時にファイルを同じ名前の別のファイルで上書きしてしまう危険性があります。 この設定には値は不要であり、たいていの場合は、空白のままで構いません。ただし値を指定する場合は、有効な Windows フォルダー名を入力する必要があります。 詳細については、「同じ名前のファイルをインストールする」を参照してください。
レジストリ リフレクションを無効にする	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	レジストリ リフレクションを使用して、ターゲット マシン上で 32 ビットの [レジストリ] ビューと 64 ビットの [レジストリ] ビューの同期を保つことができます。 指定されたコンポーネントによって影響を受けるすべての既存および新規のキーに対してレジストリ リフレクションを有効にする場合、[いいえ] を選択します。この値は既にデフォルトで設定されています。 このコンポーネントによって影響を受けるすべての既存および新規のキーについてレジストリ リフレクションを無効にする場合、[はい] を選択します。コンポーネントが各キーにアクセスする際、Windows Installer が **RegDisableReflectionKey**関数を呼び出します。この関数は、指定されたキーのレジストリ リフレクションを無効にします。キーのリフレクションを無効にしても、サブキーのリフレクションには影響ありません。 ***メモ:*** *この設定は、Windows Installer 4 以降がある 64 ビット システムでのみサポートされています。また、Windows Vista 以降と、Windows Server 2008 以降でのみサポートされています。他のシステムは、この設定を無視します。* レジストリ リフレクションの詳しい情報については、MSDN ライブラリの「Registry Reflection」を参照してください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
複数パッケージの共有コンポーネント	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	選択されたコンポーネントの共有コンポーネントのパッチを有効にするには [はい] を選択し、無効にするにはデフォルト値 [いいえ] を選択します。ターゲット システムにインストールされている少なくとも 1 つのパッケージについて、この複数パッケージの共有機能が有効な場合、Windows Installer 4.5 はこれらすべてのパッケージ間でコンポーネントが共有されているものとして処理します。このコンポーネントを共有するパッチがアンインストールされると、Windows Installer はシステム上で、コンポーネントのファイルの最も上位バージョンを共有し続けることができます。 複数パッケージ コンポーネントの共有を行う目的は、1 つまたは複数の他のインストール パッケージによって共有されているコンポーネントを含むパッチのアンインストール中に、ファイルがダウングレードされないように防ぐことです。これによって、そのパッチがアンインストールされた後でも、コンポーネントのファイルの最も上位バージョンがマシン上で保持されます。 次のシナリオは、この動作を分かりやすく説明します: • 共有ファイル 1.0.0.0 を持つ製品 A をインストールします。 • 共有ファイル 1.1.0.0 を持つ製品 B をインストールします。 • 共有ファイル 1.2.0.0 を持つ、製品 A のアンインストール可能なパッチを適用します。 • 製品 A のパッチをアンインストールします。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
複数パッケージの共有コンポーネント (続き)		結果として、以下のどちらかの可能性があります: • 製品 A または 製品 B のどちらかについて、"複数パッケージの共有コンポーネント" 設定が [はい] に設定されている場合で、Windows Installer 4.5 が存在するときは、パッチを製品 A からアンインストールすると、ターゲット システムでバージョン 1.1.0.0 が復元されます。 • 製品 A と製品 B でこの設定の値が [いいえ] の場合、製品 B が 1.1.0.0 を使用するのにも関わらず、ファイルのバージョンは 1.0.0.0 にダウングレードされます。このため、製品 B がバージョン 1.1.0.0 を必要とするときに、B が適切に動作しない可能性があります。 ***メモ:*** *ターゲット システムで DisableSharedComponent ポリシーが 1 に設定されている場合、Windows Installer はすべてのパッケージについて、この設定を無視します。* *Windows Installer 4.0 以前では、この設定は無視されます。* この設定に関する詳しい情報は、「コンポーネントの "共有コンポーネントのパッチ" を有効にするかどうかを指定する」をご覧ください。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
置き換えられたコンポーネントのアンインストール	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、特定の状況下において優先するパッチのインストール中に、Windows Installer が選択されたコンポーネントを処理する方法を決定します。これは、インストールまたはアンインストールされるパッチが機能のコンポーネント カウントを変更する場合に、コンポーネントの機能がアドバタイズ状態に変更されることを阻止するように設計されています。機能の状態がアドバタイズに変更されると、機能の残りのコンポーネントは一切保持されません。 現在のパッチに含まれるこのコンポーネントをアンインストール用にフラグすることで、優先するパッチが適用された後にターゲット システム上でこのコンポーネントが孤立しないようにするためには、[はい] を選択します。最初のパッチを優先するようにフラグされている後続のパッチがインストールされると、(つまり、[パッチのデザイン] ビューのパッチの構成で、[シーケンス] タブにある該当するパッチ ファミリーの [優先] 列に [はい] が選択されている場合)、Windows Installer は、適切な場合、このコンポーネントを登録解除およびアンインストールできます。[いいえ] を選択すると、優先するパッチがターゲット システム上に孤立したコンポーネントを残す可能性があります。この設定のデフォルト値は [いいえ] です。 [いいえ] を選択した場合で、優先されるパッチが製品のコンポーネントをインストールするが、優先するパッチがそのコンポーネントを削除する場合、コンポーネントの機能の状態はアドバタイズに変更され、再インストールはされません。さらに、その機能に関連付けられた残りのコンポーネントは一切保持されません。[はい] を選択すると、Windows Installer は優先するパッチが適用されるときに、コンポーネントを登録解除およびアンインストールします。 ***メモ•*** *Windows Installer 4.5 は、この "置き換えられたコンポーネントのアンインストール" 設定のサポートを含みます。以前のバージョンの Windows Installer はこれを無視します。* ***ヒント•*** MSIUNINSTALLSUPERSEDEDCOMPONENTS *プロパティを設定すると、パッチに含まれるすべてのコンポーネントの "置き換えられたコンポーネントのアンインストール" 設定で [はい] を選択するのと同じ効果があります。*

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
ビルド時にマージする REG ファイル	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	ビルド時に .reg ファイルをコンポーネントのレジストリ エントリとマージする場合、.reg ファイルの完全パスを入力します。その代わりに、この設定で省略記号ボタン (...) をクリックして、.reg ファイルを参照することもできます。
自己登録	InstallScript、InstallScript オブジェクト	コンポーネントのファイルが自己登録型であるかどうかを指定します。(自己登録ファイルは、他の OLE アプリケーションが認識できるようにレジストリにそれ自体の情報を入れることができる OLE サーバーです。このようなファイルは、アンインストール時にこの情報をレジストリから削除することもできます。) ***重要 •*** *[はい] を選択した場合に、コンポーネントに含まれる 1 つ以上のファイルが自己登録型ではない場合、インストール実行時にエラーが発生します。* ***プロジェクト•*** *InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトは、.dll ファイル、.exe ファイル、タイプ ライブラリ (.tlb および .olb ファイル) の自己登録をサポートします。* *基本の MSI と InstallScript MSI プロジェクトでは、自己登録がコンポーネント レベルではなくファイル レベルに設定されています。*
ファイルのロック	InstallScript、InstallScript オブジェクト	コンポーネントのファイルのインストール済みのバージョンをインストール中にロックされている (つまり、別の製品によって使用中である) 可能性があるかどうかを指定します。 インストール中、またはアンインストール中にファイルがロックされている場合で、そのファイルが "ファイルのロック" 設定が [はい] と設定されているコンポーネントにある場合は、再起動後、自動的にファイル処理が行われます。インストール中またはアンインストール中にファイルがロックされている場合で、"ファイルのロック" 設定が [いいえ] と設定されているコンポーネントにある場合は、**OnFileLocked** イベント ハンドラーが呼び出されます。(再起動後、この関数のデフォルトコードが自動的にファイル操作を行います。) InstallShield では、共有ファイルを常にロックされる可能性のあるファイルとして扱います。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
圧縮	InstallScript、InstallScript オブジェクト	実行時にコンポーネントのファイル .cab ファイルにビルドされる場合、圧縮するかどうかを指定します。 リリース ウィザードの [カスタム メディア レイアウト] パネルで、コンポーネントの機能のチェック ボックスが選択されている場合、この設定は効果を持ちません。
上書き	InstallScript、InstallScript オブジェクト	この設定は、インストール中のファイルと既存のファイルが同じ名前を持つ場合にインストールが行う動作を示します。コンポーネントに含まれる各ファイルについて、バージョン番号や更新日時に基づいて、選択的に置換することができます。動作を変更するには、この設定の省略記号ボタン (...) をクリックします。[上書き] ダイアログ ボックスが開き、ここで適切な実行時の動作を指定できます。 ***ヒント:*** *エンド ユーザー更新した可能性のあるデータ ファイルや構成ファイルなどでコンポーネントが構成されている場合、[上書きしない] オプションの選択を考慮してください。* "上書き" 設定で [上書きしない] オプションが選択されているコンポーネントに関連するファイルはアンインストールにログ記録されます。つまり、同じ名前のファイルが存在するため、ターゲット システムにコピーされなくても、アプリケーションがアンインストールされた時にそのファイルが削除されます。既存ファイルのアンインストールを防ぐため、ファイルのコンポーネントの "共有" プロパティを [はい] に設定します。

テーブル 84 • [全般] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
差分	InstallScript、InstallScript オブジェクト	差分リリースをビルドするとき、InstallShield が自動的にコンポーネントのファイルをリリースに含めるかどうかを指定します。(差分リリースを定義する場合、新しい差分リリースのビルド時に現在のプロジェクトと比較する複数の既存リリースを指定します。) 選択可能なオプションは以下のとおりです: • **適切な場合に含める** – 同一ファイル (同一の日時、サイズ、属性を持つ) が指定の各比較リリースに存在する場合、このコンポーネントのファイルが差分リリースから除外されます。デフォルトでは、これが設定されています。 • **常に含める** – 同一ファイル (同一の日時、サイズ、属性を持つ) が指定の各比較リリースに存在する場合でも、すべてのコンポーネントのファイルが差分リリースに含まれます。
リンクの種類	InstallScript、InstallScript オブジェクト	コンポーネントのファイル リンクがスタティックとダイナミックのどちらか、およびダイナミックの場合、リリースビルド時のリンクの決定方法を指定できます。
コメント	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このコンポーネントに関するコメントを入力します。入力したコメントは、参考用にプロジェクト ファイルに保存され、インストールでは使用されません。

ターゲット マシン

InstallScript、InstallScript MSI、または InstallScript オブジェクト プロジェクトの [コンポーネント] ビューまたは [セットアップのデザイン] ビューでコンポーネントを選択するとき、[ターゲット マシン] 領域では以下の設定を構成できます:

テーブル 85 • [ターゲット マシン] 領域にあるコンポーネントの設定

設定	プロジェクトの種類	説明
オペレーティング システム	InstallScript、InstallScript MSI、InstallScript オブジェクト	コンポーネントが 1 つまたは複数のオペレーティング システムに固有の場合、この設定を使用して、それらのオペレーティング システムを指定します。ターゲット マシンのオペレーティング システムがこの設定に指定されたオペレーティング システムの中にない場合、コンポーネントはインストールされません。 デフォルトでは、コンポーネントはオペレーティング システムに依存しません。つまり、コンポーネントに特定のオペレーティング システム固有のデータがないことを意味します。 この設定の値を変更するには、この設定の省略記号ボタン (...) をクリックします。

テーブル 85 • [ターゲット マシン] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
プラットフォーム スイート	InstallScript、InstallScript オブジェクト	コンポーネントが 1 つまたは複数のプラットフォーム スイートに固有の場合、この設定を使用して、それらのスイートを指定します。1 つ以上のスイートを指定する場合、コンポーネントがインストールされるために、指定されたスイートすべて、または 1 つ以上のスイートがターゲット マシンに存在している必要があるかどうかを示すことができます。 デフォルトでは、コンポーネントはプラットフォーム スイートに依存しません。つまり、コンポーネントのデータ (ファイル、レジストリ エントリなど) はどれも特定のプラットフォーム スイートに固有ではありません。 ***ヒント:*** *この設定により、"オペレーティング システム" 設定よりもさらに細かくフィルターすることができます。"プラットフォームスイート" 設定のプラットフォーム スイートは必要な場合のみ設定し、またアプリケーションが適切に機能するために必要なプラットフォーム スイートのみを確実に選択してください。たとえば、コンポーネントが WindowsXP の Home と Professional バージョンの両方にインストールされる必要がある場合、この設定の値を [スイート非依存] のままにしておくことができます。"オペレーティング システム" 設定に Windows XP を選択すると、両方のエディションが含まれます。* 実行時にインストールがサポートするプラットフォーム スイートは、**FeatureFilterOS** 関数を呼び出して制御できます。**OnFilterComponents** イベント ハンドラーで、一般的にフレームワークは、適切なコンポーネントだけがインストールされるように、この関数をターゲット システムに一致するプラットフォーム スイートで呼び出します。**FeatureFilterOS** を呼び出して、デフォルトの動作をオーバーライドし、指定したプラットフォーム スイート基準に基づいてコンポーネントをインストールしたり、またはコンポーネントのインストールを防いだりすることができます。

.NET の設定

[コンポーネント] ビューまたは [セットアップのデザイン] ビューでコンポーネントを選択すると、[.NET の設定] 領域で以下の設定を構成できます。

テーブル 86 • [.NET の設定] 領域にあるコンポーネントの設定

設定	プロジェクトの種類	説明
.NET COM Interop プロパティ	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コンポーネントに COM Interop (COM から .NET オブジェクトを呼び出す機能) を使用するかどうかを指定します。 たとえば、ComClass 属性を定義する Visual Basic .NET クラス ライブラリがあり、1 つ以上の COM 呼び出し可能なパブリック関数が含まれている場合、この設定に [はい] を選択できます。InstallShield はビルド時に COM Interop 情報を抽出して、それを .msi データベースの Registry テーブルに追加します。実行時に、COM オブジェクトで .NET アセンブリを呼び出せるレジストリ エントリがターゲット システムに作成されます。 ***メモ:*** *この設定は、コンポーネントのキー ファイルが .NET アセンブリの場合にのみ適用します。*
.NET プリコンパイル アセンブリ	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	インストールが実行時に .NET アセンブリからネイティブ イメージを作成して、ターゲット システム上にネイティブ イメージ キャッシュをインストールするかどうかを指定します。これによって、アセンブリのロードと実行がスピードアップします。これはコードおよびデータ構造がジャスト インタイム (JIT) コンパイルからではなく、ネイティブイメージキャッシュから復元されるためです。
.NET アセンブリ	InstallScript、InstallScript オブジェクト	コンポーネントのファイルをローカル .NET アセンブリとしてインストールするかどうかを指定します。選択可能なオプションは以下のとおりです: • **ローカル アセンブリ** – コンポーネントは .NET アセンブリを含みます。また、実行時にインストールはコンポーネントの COM 相互運用機能の登録を行い、.NET インストーラー クラス情報を構成します。 • **.NET アセンブリではない** – コンポーネントは .NET アセンブリを含みません。デフォルトでは、これが設定されています。 InstallShield がこのコンポーネントに含まれるアセンブリをスキャンして、実行時に .NET 依存関係を検出する場合、この設定で [ローカル アセンブリ] を選択しなくてはなりません。詳細については、「.NET アセンブリのプロパティおよび依存関係を識別する」を参照してください。

テーブル 86 • [.NET の設定] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
ビルド時に .NET をスキャン	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	 ***プロジェクト•*** *この設定の機能は、プロジェクトが Windows Installer ベースのプロジェクト (基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、またはトランスフォーム プロジェクト) か、InstallScript ベースのプロジェクト (InstallScript プロジェクトまたは InstallScript オブジェクト プロジェクト) かによって異なります。* • **Windows Installer ベースのプロジェクトの場合** – ビルド時に、このコンポーネントのキー ファイルをスキャンして .NET 依存関係、プロパティ、またはその両方をチェックするかどうかを指定します。 • **InstallScript ベースのプロジェクトの場合** – ビルド時に、このコンポーネントのファイルをスキャンして .NET 依存関係をチェックするかどうかを指定します。 詳細については、「.NET アセンブリのプロパティおよび依存関係を識別する」を参照してください。
.NET アプリケーション ファイル	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	この設定は、このコンポーネントがビルド時にスキャン (″ビルド時に .NETをスキャン″ 設定による) され時、またはスタティック スキャン ウィザードで使用されます。スキャナーはコンポーネントのインストール先と一緒にこの設定を使用して、アセンブリの ″ファイル アプリケーション″ 設定の値を決定します。 アドバタイズおよび修復を最適化するには、このコンポーネントでアセンブリを使用する .NET 実行可能ファイルを設定します。 ″ビルド時に .NET スキャン″ 設定が [依存関係とプロパティ] または [プロパティ] に設定されている場合、この設定にはコンポーネントのキー ファイルが表示されます。プロジェクト内のその他のキー、またはポータブル実行可能ファイルを選択することができます。 ***メモ•*** *この設定は、コンポーネントのキー ファイルが .NET アセンブリの場合にのみ適用します。* *選択されたファイルが .NET アセンブリ ファイルではない場合、この設定の値は無視されます。*

テーブル 86 • [.NET の設定] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
.NET インストーラ クラス	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	 ***メモ:*** *この設定は、このコンポーネントのキー ファイルが、システム設定 System.Configuration.Install.Installer から派生したクラスを含む .NET アセンブリである場合にのみ適用します。* 実行時に、アセンブリの Install、Commit、Rollback、および Uninstall メソッドを適切なタイミングで呼び出す場合は、[はい] を選択します。
.NET インストーラ クラス引数	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	.NET インストーラー クラスに渡す引数を指定するには、この設定で省略記号ボタン (...) をクリックします。[.NET インストーラー クラス引数] ダイアログ ボックスが開き、ここで Install、Commit、Rollback、および Uninstall メソッドの引数を指定できます。 サンプル コードは、「.NET Installer クラスへ渡されたプロパティを読み込む」を参照してください。 ***メモ:*** *この設定は、コンポーネントのキー ファイルが .NET アセンブリの場合にのみ適用します。*

実行時の設定

InstallScript プロジェクト、または InstallScript オブジェクト プロジェクトの [コンポーネント] ビューまたは [セットアップのデザイン] ビューでコンポーネントを選択するとき、[ランタイムの設定] 領域では、以下の設定を構成できます:

テーブル 87 • [実行時の設定] 領域にあるコンポーネントの設定

設定	プロジェクトの種類	説明
FTP ロケーション	InstallScript、InstallScript オブジェクト	FTP の場所をコンポーネントに関連付けるには、FPT アドレスを入力します。 入力する場所は、スクリプト内からはアクセスできません。機能でも、これと同じ設定が使用できますが、機能の設定に指定する場所は、スクリプト内からはアクセスできません。

テーブル 87 • [実行時の設定] 領域にあるコンポーネントの設定 (続き.)

設定	プロジェクトの種類	説明
HTTP ロケーション	InstallScript、InstallScript オブジェクト	HTTP アドレスをコンポーネントに関連付けるには、HTTP アドレスを入力します。 入力する場所は、スクリプト内からはアクセスできません。機能でも、これと同じ設定が使用できますが、機能の設定に指定する場所は、スクリプト内からはアクセスできません。
その他	InstallScript、InstallScript オブジェクト	テキスト文字列をコンポーネントと関連付けるには、そのテキスト文字列を入力します。 入力する場所は、スクリプト内からはアクセスできません。機能でも、これと同じ設定が使用できますが、機能の設定に指定する場所は、スクリプト内からはアクセスできません。

コンポーネントの詳細設定

プロジェクト• *コンポーネントの [詳細設定] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビュー (および [セットアップのデザイン] ビュー) 内にあるコンポーネントの [詳細設定] 領域では、特定のコンポーネント タイプのインストール条件を満たすことができます。たとえば .ocx ファイルをターゲット システムにコピーする場合、ファイルのメソッドが適切にアクセスされるようにクラス、ProgID、およびタイプ ライブラリを登録する必要があります。詳細設定は Windows Installer のビルトイン機能を利用して COM サーバーの登録、ODBC ドライバー、データソース、およびトランスレーターの設定、Windows サービスのインストールまたは制御、並びにファイル関連付けの登録を行います。

詳細設定を指定してコンポーネントのパブリッシュ、COM サーバー、ファイル拡張サーバー、および MIME の種類の登録を行なうことができます。コンポーネントが選択されている場合、コンポーネントがインストールまたはアドバタイズされるときにターゲット システム上で詳細設定が行なわれます。したがって、ファイルはインストールされた直後に実行可能となります。アドバタイズの一種であるコンポーネントのパブリッシュは、パブリッシュの詳細設定を通して行います。

詳細設定のカテゴリ

[コンポーネント] ビュー (および [セットアップのデザイン] ビュー) のコンポーネントの下にある [詳細設定] 領域は、以下の主要なカテゴリに分かれています:

テーブル 88 • コンポーネントの [詳細設定] のカテゴリ

分類	説明
アプリケーション パス	コンポーネントのファイルやフォルダーのパスを指定するには、この詳細設定を使用します。インストールを実行すると、次のレジストリ キーの下にコンポーネントのアプリケーション パスが作成されます: HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥App Paths¥ProgramName.exe
アセンブリ	コンポーネントがインストールされるときに Win32 または .NET アセンブリをインストールするには、この詳細設定を使用します。
COM 登録	[COM 登録] 領域で、コンポーネントのキー ファイルとして設定した COM サーバー登録情報を入力または変更することができます。[COM 登録] の詳細設定は、自己登録を置き換えるので、セットアップ ベストプラクティスの違反となります。 ファイル登録を行う方法としては、コンポーネント ウィザードを使用し、自動的に必要な情報を抽出する方法をお勧めします。COM 登録エクスプローラーは、詳細変更を行う必要がある場合にのみ使用してください。
ファイルの種類	この詳細設定は、コンポーネントがインストールまたはアドバタイズされている時に、ターゲット システム上でファイルの種類に関する情報を登録します。
サービス	[サービス] 領域を使って、インストールとアンインストール中の Windows サービスのインストール、構成、開始、停止、およびアンインストールを行います。 別の方法として、[サービス] ビューを使って Windows サービスをインストール、構成、開始、停止、およびアンインストールすることができます。これらの領域のいずれかを使ってサービス情報を構成した場合、対応する領域も自動的に更新されます。
パブリッシュ	コンポーネントをパブリッシュするには、この詳細設定を使用します。パブリッシュ (公開) は、アドバタイズ (ジャストインタイム インストール) の一種です。インストール中にコンポーネントのユーザー インターフェイス要素は作成されませんが、コンポーネントは [プログラムの追加と削除] アプレットからインストールすることができます。また、インストールされたコンポーネントがインストーラーからパブリッシュされたコンポーネントを要求した際にもインストールできます。
デバイスドライバー	[デバイス ドライバー] 詳細設定では、現在のコンポーネントがデバイス ドライバーを含むかどうか、またその種類、およびプロジェクトのデバイスドライバー (現在のコンポーネントのデバイス ドライバーだけでなく、すべてのドライバー) がインストールされる順番を指定することができます。
その他のデータ	[その他のデータ] 詳細設定には、コンポーネントと関連したさまざまな Windows Installer テーブルがリストされます。

アプリケーション パスの設定

プロジェクト: *コンポーネントの [アプリケーション パス] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューと [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [アプリケーション パス] 領域には、以下の列が表示されます:

テーブル 89 • アセンブリの設定

列	説明
チェック ボックス	コンポーネントにアプリケーション パスを追加するには、アプリケーション パスを定義するアプリケーションの実行可能ファイルのチェック ボックスを選択します。
File	選択されたコンポーネントに含まれる実行可能ファイルを表示する読み取り専用列。
アプリケーション パス	実行可能ファイルが、その DLL の検索パスとして使用するディレクトリのシーケンスを入力します (実行時に実行可能ファイルにリンクされる、または **LoadLibrary** API と共にロードされる)。複数のディレクトリはセミコロン (;) で区切ります。 パスには、**C:¥MyDir** のようにハード コード化されたパスを使用しないでください。代わりに、Windows Installer フォルダー プロパティを角かっこで囲み、**[INSTALLDIR]MyDir** および **[CommonFilesFolder]MyProgram** のように使用するか、または **[$ComponentName]** のように、ドル記号の後にコンポーネント名を使用します。Windows Installer はこれらの値を解決し、結果として与えられた場所をアプリケーション パス キーに書き込みます。

アセンブリの設定

プロジェクト: *コンポーネントの [アセンブリ] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューと [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [アセンブリ] 領域には、以下の設定があります:

テーブル 90 • アセンブリの設定

設定	説明
マニフェスト	.NET アセンブリの場合、この設定は自動的にコンポーネントの最初の .exe または DLL ファイルに設定されます。 Win 32 アセンブリの場合、コンポーネントの最初のファイルに拡張子 .manifest を付けて自動的に設定されます。マニフェストのファイル名は、拡張子 .manifest が付いたアプリケーションの実行可能ファイルの名前です。 この設定は、アセンブリの "ファイル アプリケーション" 設定にグローバル アセンブリ キャッシュを選択する場合、空白に残します。 "マニフェスト" 設定は削除できません。
ファイル アプリケーション	.NET アセンブリの場合、グローバル アセンブリ キャッシュにアセンブリをインストールするかどうか指定できます。適切なファイルを選択すると、アプリケーションのプライベート キャッシュにアセンブリがインストールされ、[グローバル アセンブリ キャッシュ] を選択すると、グローバル アセンブリ キャッシュにアセンブリがインストールされます。 Win32 アセンブリの場合、選択した .exe 用にアセンブリを個別にインストールするか、またはアセンブリをグローバル アセンブリ キャッシュにインストールするかを指定することができます。[グローバル アセンブリ キャッシュ] を選択すると、そのアセンブリは、ターゲット システム上で任意の .NET アプリケーションによる使用が可能となります。 "ファイル アプリケーション" 設定は削除できません。 この設定の値は、コンポーネントの "インストール先" 設定に影響します。"ファイル アプリケーション" 設定を [グローバル アセンブリ キャッシュ] に設定すると、コンポーネントの "インストール先" 設定も [グローバル アセンブリ キャッシュ] に設定されます。アドバタイズおよび修復を最適化するには、このコンポーネントでアセンブリを使用する実行可能ファイルを設定します。 ***メモ*** *• .NET アセンブリをグローバル アセンブリ キャッシュにインストールする場合、アセンブリが特定の基準を満たさなくてはなりません。詳細については、.NET ドキュメントを参照してください。* *Win 32 アセンブリをグローバル アセンブリ キャッシュにインストールするには、ファイルが署名済みであること、またカタログ (.cat) ファイルを含むことが必要です。詳細は、Platform SDK ヘルプの「Creating Signed Files and Catalogs」を参照してください。*

テーブル 90 • アセンブリの設定 (続き.)

設定	説明
プロパティ	プロパティ名と値を入力します。新しいエントリを作成するには、グリッド上で右クリックして [新規] を選択します。 Win32 アセンブリの場合、type、name、version、language、publicKeyToken、および processorArchitecture プロパティの値を必ず入力してください。 プライベート .NET アセンブリの場合、Name、Version、および Culture プロパティの値を入力しなくてはなりません。プライベート .NET アセンブリの場合、Name、Version、および Culture プロパティの値を入力しなくてはなりません。 InstallShield は、入力されたプロパティ名と値を Windows Installer データベースの MsiAssemblyName テーブルに追加します。このテーブルにあるレコードは、ダイレクト エディターを使って表示できます。 ***メモ•*** *アセンブリに入力するプロパティと値は、アセンブリのマニフェスト ファイル内の情報と一致しなくてはなりません。一致しなかった場合、製品がアンインストールされる時に、アセンブリがターゲット システムに残る可能性があります。*

COM 登録の設定

プロジェクト• *コンポーネントの [COM 登録] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューと [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [COM 登録] 領域は、次のカテゴリに分かれています:

- COM クラス
- ProgID
- タイプ ライブラリ

COM クラス

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [COM 登録] 領域で [COM クラス] を選択すると、以下の設定を構成できます:

テーブル 91 • COM クラスの設定

設定	説明
ProgID	この COM サーバーに対して作成したリストから ProgID を選択します。 新しい名前を入力して ProgID を作成することもできます。その場合、InstallShield によって ProgID ノードの下に新しい ProgID が追加されます。
ファイルの種類マスク	次の形式でファイルの種類マスクを入力します: **オフセット、cb、マスク、値** • **オフセット** – バイト単位でのファイルの始まりからの位置。バイト範囲がどこから始まっているかも示します。オフセット値が負の整数の場合、ファイルの終わりからのバイト数。 • **cb** – オフセット値で始まるファイルのバイト数。これはマスクと共に論理 AND 演算を実行するときに使用されます。 • **マスク** – 結果を指定された値 (次のエントリ) と比較できるように、ファイルのビットと共に論理 AND 演算を実行するときに使用される値。この値を空白にすると、全て 1 と仮定されます。 • **値** – ファイルのクラス ID を判別するために、マスクとファイルのバイト範囲の間の論理 AND 演算の結果と比較されるバイト数 ファイルの種類マスクは、HKEY_CLASSES_ROOT¥FileType¥*CLSID*¥*pattern* で登録されるパターンの一部です。**OleCreateFromFile** や **GetClassFile** などの Windows API は、このキーの情報とファイルのバイト数を比較することによってファイルのクラス ID を取得します。たとえば **0,4,FFFFFFFF,AABBCCDD** は、システムが指定されたファイルに対してこのコンポーネントのクラス ID を使用する場合、ファイルの最初の 4 バイトが AA BB CC DD でなくてはならないという意味です。 ファイルと一致するマスクを複数指定できます。複数のパターンはセミコロン (;) で区切ります。インストールによりこれらのキーが作成されると、各パターンのサブキーに対して数字 0 から順番に番号が自動的に付けられます。
アイコン ファイル	この COM クラスと関連付けられたアイコンを含むファイルへの完全修飾パスを入力するか、この設定の省略記号ボタン (...) をクリックしてパスを参照します。 .ico、.exe、または .dll ファイルを指定できます。実行可能ファイルまたは DLL を参照して特定のアイコンを選択すると、そのファイルからアイコンが抽出されます。

テーブル 91 • COM クラスの設定 (続き.)

設定	説明
アイコン インデックス	ファイルに複数のアイコン リソースがある場合、適切なアイコンにインデックスを入力します。 負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば 0 はファイルの最初のアイコン、1 は 2 番目、2 は 3 番目というようになります 負の数字は特定のリソース ID を参照するために使用します。たとえばアイコン インデックス -12 は、リソース ID が 12 のアイコンを示します。
AppID	このファイルの登録用に既に作成した AppID の名前のリストから、このDCOM または COM+ クラス用の AppID を選択します。 AppID は HKEY_CLASSES_ROOT¥AppID に登録される GUID で、分散したCOM オブジェクトのすべてのセキュリティと設定オプションをグループ化するために使用します。 新しい AppID を作成するには、ダイレクト エディターを使用します。ダイレクト エディターで AppID テーブルに移動して、新しい AppID に情報を入力します。一度 AppID を作成すると、この設定のリストからその AppID を選択できるようになります。 AppID を完全に削除するには、ダイレクト エディター内から AppID の行を削除します。
説明	COM クラスの説明を入力してください。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 91 • COM クラスの設定 (続き.)

設定	説明
相対パス	そのクラス ID の下にあるファイルへの相対パスを登録するかどうかを指定します。選択可能なオプションは以下のとおりです: • **いいえ**—そのクラス ID の下にあるファイルへの完全修飾パスが登録されます。その後、クライアント アプリケーションがこのクラスのオブジェクトのインスタンスを作成すると、システムは常に登録された COM サーバーをポイントします。デフォルトでは、これが設定されています。 • **はい** — インストールは、パスを含まずにファイルの名前のみを登録します。これにより、コンポーネントの サイドバイサイド (並行) インストールが可能になります。つまり、オペレーティング システムは現在のディレクトリ中の COM サーバーのコピーを使用し、他のアプリケーションはそれぞれのフォルダーにファイルの別のコピーを保持できます。 ***メモ:*** *パスが相対的でファイルがシステムのパス中にない場合は、オペレーティング システムは、クライアントがサーバーの現在のディレクトリの外にオブジェクトを作成しようとしたときに、そのクラスを見つけられません。*

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [COM 登録] 領域で LocalServer32、LocalServer、または InprocServer32 コンテキスト タイプを選択すると、以下の設定を構成できます:

テーブル 92 • COM クラス コンテキストの設定

設定	説明
デフォルト Inproc ハンドラー	ローカル サーバーまたはローカル サーバー 32 のコンテキストの種類の InprocHandler の値を選択します。
引数	COM サーバーを呼び出す時に OLE が使用する引数を入力します。

ProgID

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [COM 登録] 領域で [ProgID] を選択すると、以下の設定を構成できます:

テーブル 93 • ProgID の設定

設定	説明
COM クラス	この ProgID が参照する COM クラスがある場合は、その COM クラスを選択します。 後で、COM クラスのクラス ID (CLSID) を変更する場合、その関連付けを修復するにはもう一度リストから選択する必要があります。
説明	この ProgID の説明を入力します。 この説明は、コンポーネントがインストールされると、レジストリキー (HKEY_CLASSES_ROOT¥*class.object*) で ProgID のデフォルト値として登録されます。ファイル拡張子に関連付けられているファイルを右クリックしてから [プロパティ] を選択すると、ここに入力した説明が [全般] タブに表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
アイコン ファイル	この ProgID と関連付けられたアイコンを含むファイルへの完全修飾パスを入力するか、この設定の省略記号ボタン (...) をクリックしてパスを参照します。 .ico、.exe、または .dll ファイルを指定できます。実行可能ファイルまたは DLL を参照して特定のアイコンを選択すると、そのファイルからアイコンが抽出されます。
アイコン インデックス	ファイルに複数のアイコン リソースがある場合、適切なアイコンにインデックスを入力します。 負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば 0 はファイルの最初のアイコン、1 は 2 番目、2 は 3 番目というようになります 負の数字は特定のリソース ID を参照するために使用します。たとえばアイコン インデックス -12 は、リソース ID が 12 のアイコンを示します。

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [COM 登録] 領域でバージョン非依存型 ProgID を選択すると、以下の設定を構成できます:

テーブル 94 • バージョン非依存型 ProgID の設定

設定	説明
説明	このバージョン独立の ProgID の説明を入力します 説明は、このコンポーネントがインストールされると、ルートキー (HKEY_CLASSES_ROOT¥*バージョン非依存 ProgID*) の下にバージョンに依存しない progID のデフォルト値として登録されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

タイプ ライブラリ

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [COM 登録] 領域で [タイプ ライブラリ] を選択すると、以下の設定を構成できます:

テーブル 95 • タイプ ライブラリの設定

設定	説明
ロケールID	タイプ ライブラリのロケール ID の 10 進数値を入力します。 通常タイプ ライブラリは言語に依存しないため、数字 0 がタイプ ライブラリの最も一般的な言語です。
バージョン	タイプ ライブラリのバージョン番号を入力します。この設定を空白のままにすると、実行時に Windows Installer によってタイプ ライブラリのバージョンが自動的に判別されます。 Windows Installer が使用するタイプ ライブラリのバージョンは、*x.y* の形式で入力する必要があります。ここで、*x* および *y* は 10 進整数です。
コスト	このタイプのライブラリの登録に関連するコストを入力します。負の数値を利用することはできません。特定の値を使用しない場合は、数値 1 を入力します。
ヘルプ パス	このタイプ ライブラリのヘルプがインストールされるインストール先フォルダーを選択するか、省略記号ボタン (...) をクリックしてディレクトリを選択または作成します。 コンポーネントのデフォルトのインストール先ディレクトリを使用するには、[INSTALLDIR] を選択します。

テーブル 95 • タイプ ライブラリの設定 (続き.)

設定	説明
説明	このタイプ ライブラリの説明を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

ファイル タイプの設定

プロジェクト: *コンポーネントの [ファイルの種類] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューと [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [ファイルの種類] 領域は、次のカテゴリに分かれています:

- 拡張子
- 動詞
- MIME タイプ
- ProgID

拡張子

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [ファイルの種類] 領域で [拡張子] を選択すると、以下の設定を構成できます:

テーブル 96 • 拡張子の設定

設定	説明
ProgID	このファイル拡張子に登録する ProgID を選択します。 新しい名前を入力して ProgID を作成することもできます。その場合、InstallShield によって ProgID ノードの下に新しい ProgID が追加されます。 ファイルの種類の ProgID は任意の文字列ですが、ターゲット システム上で一意である必要があります。ProgID の命名規則には、拡張子にドットを付けず、*file* という語を追加する、というものがあります。従って、*.ext* 拡張子は *extfile* という ProgID となる場合もあります。別の命名規則には、**SampleApp.Document**のように、特定の種類のファイルを開くアプリケーション名からファイルの種類 ProgID の名前を付ける、というものがあります。
MIME タイプ	このファイル拡張子に関連付けられている MIME タイプが存在する場合はその種類を選択します。 拡張子用に新しい MIME タイプを作成するには、そのアイコンを右クリックして [新しい MIME タイプ] をクリックします。

動詞

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [ファイルの種類] 領域で、ファイル拡張子の下にある動詞を選択すると、以下の設定を構成できます:

テーブル 97 • 動詞の設定

設定	説明
コマンド シーケンス	コマンド動詞のシーケンス番号を入力します。このファイル拡張子に複数の動詞が関連付けられている場合、シーケンス番号に基づいてコンテキスト メニュー (右クリックメニューまたはポップアップ メニュー) に表示される動詞の順番が決定されます。 シーケンス番号を指定しなかった場合、または複数の動詞が同じシーケンス番号を持つ場合、動詞はアルファベット順に並べられます。 コマンド シーケンスの最初の動詞は、ファイルの種類のコンテキスト メニューのデフォルトの オプションとして、太字体で表示されます。

テーブル 97・動詞の設定 (続き.)

設定	説明
表示名	エンド ユーザーが現在の拡張子を持つファイルを右クリックしたとき、Windows Explorer で表示されるコンテキスト メニューのこの動詞用に表示するテキストを入力します。例えば、このファイル拡張子のコンテキスト メニューで「**SampleApp で開く(O)**」(*O* に下線) を表示するには、「**SampleApp で開く(&O)**」と入力します。 この設定はオプションです。表示名を指定しなかった場合、[拡張子] ツリーに表示される動詞の名前がターゲット システム上のファイル タイプのコンテキスト メニューに使用されます。*開く*、*印刷*、および*検索*などの標準的な動詞の 1 つを使用し、表示名を指定しなかった場合、Windows は各システムの動詞を自動的にローカライズします。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
引数	この動詞のコマンドライン引数を入力します。 選択されたファイル名の引数の位置に **%1** を使用してください。たとえば、エンド ユーザーが C:¥File.ext を右クリックして、この動詞の引数が **-p %1** と仮定した場合、コマンドライン引数は **-p C:¥File.ext** になります。 スペースを含むファイル名を正しく処理するために、**"%1"** のように %1 引数を引用符で囲む必要がある場合もあります。

MIMEタイプ

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [ファイルの種類] 領域で、ファイル拡張子の下にある動詞を選択すると、以下の設定を構成できます:

テーブル 98・MIME タイプの設定

設定	説明
クラス ID	この MIME タイプと関連付けられているクラス ID を入力します。

ProgID

[コンポーネント] ビューまたは [セットアップのデザイン] ビューにある [ファイルの種類] 領域で [ProgID] を選択すると、以下の設定を構成できます: 各設定の詳細については、「ProgIDs」を参照してください。

サービスの設定

***プロジェクト・** コンポーネントの [サービス] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

メモ: *Windows Installer はドライバー サービスをサポートしていないので、サービスは単一実行可能ファイル (.exe) である必要があります。*

サービス実行可能ファイルは、そのコンポーネントのキー ファイルでなくてはなりません。詳細については、「コンポーネントのキーファイル」を参照してください。

サービスの設定に関する情報は、「[サービス] ビュー」を参照してください。

パブリッシュの設定

プロジェクト: *コンポーネントの [パブリッシュ] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[コンポーネント] ビューと [セットアップのデザイン] ビューの [詳細設定] タブの下にある [パブリッシュ] 領域を使って、次の種類の項目を [パブリッシュ] エクスプローラーに追加することができます:

テーブル 99 • パブリッシュの設定

設定	説明
ComponentID	GUID である componentID は、修飾コンポーネントとしてグループ化された複数のコンポーネントのカテゴリを表すカテゴリ ID です。各 ComponentID には少なくとも 1 つの修飾子が必要です。 ***メモ:*** *ComponentID と コンポーネントの "コンポーネント コード" で入力する GUID とは異なりますので、ご注意ください。これらは両方とも一意の値でなくてはなりません。*

テーブル 99 • パブリッシュの設定 (続き.)

設定	説明
修飾語	修飾子とは、たとえば言語を指定する場合に、この言語またはコンポーネントのバージョンを他から区別するために使用する文字列です。修飾子は、コンポーネントに対して一意である必要があります。

[パブリッシュ] 領域で修飾子を選択して、以下の設定を構成できます:

テーブル 100 • 修飾子の設定

設定	説明
アプリケーション データ	修飾子に対応するアプリケーション データを入力します。アプリケーション データは、エンド ユーザーに表示できるテキスト 文字列です。この設定はオプションです。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

デバイス ドライバーの設定

プロジェクト: *コンポーネントの [デバイス ドライバー] 領域は、以 下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[コンポーネント] ビューと [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [デバイス ドライバー] 領域には、以下の設定があります:

- [共通] タブ
- [シーケンス] タブ

[共通] タブ

[共通] タブには、デバイス ドライバーに関する以下のランタイム設定があります:

テーブル 101 • [共通] タブにあるデバイス ドライバーの設定

オプション	説明
既存のデバイス ドライバーを常に上書きする	既存のドライバーをこのドライバーで置き換えます。このチェック ボックスがチェックされていないで、デバイスのドライバーが既に存在する場合、ドライバーのインストールは実行されません。
[デバイスをコンピューターへ接続] ダイアログを非表示にする	ドライバーに対応するデバイスがコンピューターへ接続されていない場合、デバイスドライバーのインストール中に [デバイスをコンピューターへ接続] ダイアログが表示されないようにするには、このオプションを選択します。
デバイス ドライバーの [プログラムの追加と削除] エントリを作成しない	このチェック ボックスを選択すると、インストールで、デバイス ドライバーの [プログラムの追加と削除] エントリは作成されません。このチェック ボックスをクリアすると、インストールで [プログラムの追加と削除] エントリが作成されます。 DIFxApp は Windows Vista 以降のシステム上では、この機能をサポートしません。
署名がないドライバー ファイルとファイルが足りないドライバーをインストールする	デフォルトでは、DIFxApp は、署名されていないドライバー パッケージや、ファイルが足りないドライバー パッケージをインストールすることができません。このデフォルトの動作をオーバーライドするには、このチェック ボックスを選択します。このチェック ボックスを選択すると、最終版の署名済みバージョンを出荷する前に完全なテストを手軽に行うことができます。
アンインストール時にドライバーに関連付けられているバイナリ ファイルを削除する	デフォルトで、DIFxApp は、ドライバー パッケージをアンインストールするとき、ドライバーをインストールしたときにシステムにコピーされたバイナリ ファイルを削除しません。このデフォルトの動作をオーバーライドするには、このチェック ボックスを選択します。 このチェック ボックスを選択すると、DIFxApp は、バイナリ ファイルがドライバー ストア内の対応するバイナリ ファイルと同一の場合のみ、そのバイナリ ファイルをシステムから削除します。 ***注意** • ドライバーのバイナリ ファイルが他のドライバー パッケージまたはアプリケーションに必要ではないことを確認できるときのみ、[**アンインストール時にドライバーに関連付けられているバイナリ ファイルを削除する**] チェック ボックスを選択してください。*

テーブル 101 • [共通] タブにあるデバイス ドライバーの設定 (続き.)

オプション	説明
ローカライズ済みランタイム ダイアログをすべて含める	これをチェック ボックスを設定すると、インストールは以下の言語のダイアログおよびテキストを含むカスタム アクション ランタイム .dll を使用します。 • アラビア語 (サウジアラビア) • 中国語 (中国) • 中国語(台湾) • チェコ語 (チェコ) • デンマーク語 (デンマーク) • オランダ語 (オランダ) • 英語 (U.S.) • フィンランド語 (フィンランド) • フランス語 (フランス) • ドイツ語 (ドイツ) • ギリシャ語 (ギリシャ) • ヘブライ語 (イスラエル) • ハンガリー語 (ハンガリー) • イタリア語 (イタリア) • 日本語 (日本) • 韓国語 (韓国) • ノルウェー語 (ブークモール) (ノルウェー) • ポーランド語 (ポーランド) • ポルトガル語 (ブラジル) • ポルトガル語 (ポルトガル) • ロシア語 (ロシア) • スペイン語 – モダン ソート (スペイン) • スウェーデン語 (スウェーデン) • トルコ語 (トルコ) このチェック ボックスをクリアすると、インストールは英語のランタイム ダイアログのみを使用します。英語のランタイム .dll ファイルはローカライズされた .dll ファイルよりも小さいため、容量的に問題がある場合や追加の言語ランタイム ダイアログが不要な場合は、このチェック ボックスをクリアします。 このチェック ボックスは、デバイス ドライバーを含むプロジェクトに含まれる全てのコンポーネントに適用します。

テーブル 101 • [共通] タブにあるデバイス ドライバーの設定 (続き.)

オプション	説明
デバイス ドライバー マシン アーキテクチャ	このエリアには、いくつかのオプションがあります。適切なオプションを選択します。 • デバイス ドライバーは 32 ビット マシンをターゲットする • デバイス ドライバーは Itanium 64 ビット マシンをターゲットする • デバイス ドライバーは AMD 64 ビットマシンをターゲットにする この設定は、デバイス ドライバーを含むプロジェクトに含まれる全てのコンポーネントに適用します。

[シーケンス] タブ

[シーケンス] タブでは、プロジェクトのデバイス ドライバー（現在のコンポーネントのデバイス ドライバーだけでなくすべてのデバイス ドライバー）をインストールする順番を指定することができます。

その他のデータの設定

プロジェクト: *コンポーネントの [その他のデータ] 領域は、以下のプロジェクト タイプで利用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

コンポーネントの [その他のデータ] 領域は、[コンポーネント] ビューと [セットアップのデザイン] ビューで利用できます。

タスク ***[その他のデータ] ノードを使用するには、以下の手順を実行します。***

- [その他のデータ] ノードをクリックして、コンポーネントに関連した Windows Installer テーブルのリストを表示します。リストにはコンポーネントと関連付けられた SQL Script ファイルの名前も含まれます。さらに、[その他のデータ] ノードには、Complus、DuplicateFile、Environment、IniFile、MoveFile、RemoveIniFile、RemoveRegistry、および ReserveCost テーブルで検出されたレコードの数がリストされます。
- ダイレクトエディターでテーブルに移動するには、テーブル名をクリックします。

[セットアップの種類] ビュー

プロジェクト: *[セットアップの種類] ビューは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript MSI*

基本の MSI プロジェクト用のセットアップ タイプを作成するには、機能の "インストール レベル" プロパティを使用します。

セットアップ タイプを使って、エンド ユーザーに様々なバージョンの製品を提供できます。たとえば、デフォルトのセットアップの種類は「完全」と「カスタム」です。「完全」セットアップは、インストールに含まれるすべてのファイルをインストールします。カスタム セットアップの種類は、どの機能をインストールするかエンド ユーザーが選択することができます。

セットアップの種類は機能に基づいています。各セットアップ タイプに関連付ける機能を選択します。次に、エンドユーザーが特定のセットアップの種類を選択すると、そのセットアップの種類に関連付けた機能だけがインストールされます。

デフォルトでは、作成する各プロジェクトは定義済みセットアップの種類 を含みます。セットアップの種類 ビューで、セットアップの種類を追加/削除したり、既存のセットアップの種類名を変更できるほか、各セットアップの種類に関連付けられている機能を変更することもできます。

デフォルト セットアップの種類

テーブル 102 ・ デフォルトのセットアップのタイプ

セットアップの種類	プロジェクトの種類	説明
完全	InstallScript、InstallScript MSI	[完全] セットアップ タイプには、インストール プログラムのすべての機能が含まれます。
カスタム	InstallScript、InstallScript MSI	[カスタム] セットアップ タイプでは、エンドユーザーがどの機能をインストールするか選択することができます。必要な機能は、常に確実にインストールされるようにマークします。しかし、オンラインヘルプのような機能はインストールする必要がないかもしれません。このような場合に、エンドユーザーはどのオプション機能をインストールするかを選択することができます。

その他のセットアップのタイプ

このビューではその他のセットアップのタイプを作成できます。プロジェクトに複数のセットアップの種類を追加する詳しい方法については、「セットアップの種類について」をご覧ください。

セットアップの種類の設定

[セットアップの種類] ビューでセットアップの種類を選択すると、以下の設定を構成できます:

テーブル 103 • セットアップの種類の設定

設定	説明
表示名	セットアップの種類の名前を入力します。この名前は、インストールの実行時にエンドユーザーに表示される名前になります。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
説明	セットアップの種類についての説明を入力します。 ***メモ•*** *この説明は、スクリプトで SdSetupTypeEx を呼び出した場合のみ 、インストール プロセス中にエンドユーザーに表示されます。* この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
コメント	このセットアップの種類に関するコメントを入力します。このコメントは書き込んだユーザー用ですので、インストールには表示されません。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

[パッケージ] ビュー

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

[パッケージ] ビューを使って、アドバンスト UI プロジェクトにパッケージを 1 つを、または、スイート/アドバンスト UI プロジェクトに 1 つ以上のパッケージを追加できます。追加するパッケージの種類の判別方法については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」をご覧ください。

このビューではまた、前提条件を実行するために指定されたファイルの種類に応じて、1 つ以上の InstallShield 前提条件を、.msi パッケージ、.msp パッケージ、または .exe パッケージとしてインポートすることもできます。

このビューでは、各パッケージの条件を定義したり、各パッケージを 1 つ以上のアドバンスト UI またはスイート/アドバンスト UI 機能と関連付けたり、各パッケージについてその他の設定を構成したりできます。

実行時に、[パッケージ] ビューでリストされたパッケージに定義した条件とパッケージの順番に従って、Setup.exe ファイルが各パッケージを起動します。

[パッケージ] ビューでパッケージを選択するときに、以下のタブを使用できます:

- 共通
- Features

[パッケージ] ビューでダイナミック ファイル フォルダーを選択したとき、2 つの設定が使用できるようになります。詳細については、「ダイナミック リンクの設定」を参照してください。

メモ: *機能に関連付けられていないパッケージは、親なしアイコンで表示されます。*

共通

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[パッケージ] ビューの [共通] タブは、次の主なカテゴリに分かれています:

- 全般
- 操作
- 構成 (これは Web 配置パッケージで使用できます。)
- イベント (このイベントは、スイート/アドバンスト UI プロジェクトで使用できます。)

[全般] の設定

パッケージの "全般" 設定を使って、アドバンスト UI またはスイート/アドバンスト UI パッケージの表示名や、パッケージの起動条件などの詳細を指定できます。

メモ: *一部の設定は、すべての種類のアドバンスト UI またはスイート/アドバンスト UI パッケージ ファイルに適用できません。*

テーブル 104 • [共通] タブにある全般設定

設定	パッケージ ファイルの種類	説明
パッケージ GUID	.msi .msp .exe .appx InstallScript Web 配置 基本の MSI プロジェクト InstallScript プロジェクト WPM パッケージ	このパッケージを一意に識別する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。
WPM パッケージ ID	WPM パッケージ	Windows パッケージ マネージャー (WPM) パッケージを識別する Windows パッケージ マネージャー (WPM) パッケージ ID を入力します。 Windows パッケージ マネージャー (WPM) パッケージ ID を取得するには、Microsoft リポジトリのパッケージ マニュフェストを検索するか、Windows パッケージ マネージャー クライアント winget コマンドライン ツールを **search** または **show** オプションで実行してください。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
バージョン	WPM パッケージ	Windows パッケージ マネージャー リポジトリからインストールするパッケージのバージョンを選択してください。選択可能なオプションは以下のとおりです: • **最新バージョン** – Windows パッケージ マネージャー リポジトリで使用可能な最新のパッケージをインストールするには、このオプションを選択します。 デフォルトは、**[最新バージョン]** オプションが選択されていて、Windows パッケージ マネージャー クライアント winget コマンドライン ツールが、リポジトリで使用可能な最新のパッケージをインストールします。 • **特定バージョン** – Windows パッケージ マネージャー リポジトリで使用可能な特定のパッケージをインストールするには、このオプションを選択します。
特定のバージョン	WPM パッケージ	Windows パッケージ マネージャー リポジトリからインストールするパッケージのバージョンを指定してください。 この設定は、**"バージョン"** 設定に **[特定のバージョン]** が選択されている場合に有効です。 ***メモ:*** *この設定は、"バージョン" 設定で **[最新バージョン]** を選択した場合は無効です。*
追加のパラメーター	WPM パッケージ	Windows パッケージ マネージャー リポジトリからパッケージをインストール中に渡したい任意の追加パラメーターを指定します。 たとえば **--accept-source-agreements** パラメーターを使って、ソース ライセンスの使用許諾契約に同意することで問題を回避し、パッケージが正しくインストールされるようにします。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
表示名	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	実行時にこのパッケージに表示する名前を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
製品構成	基本の MSI プロジェクト	選択されたパッケージに関連付ける InstallShield プロジェクトのリリースに含まれる製品構成を選択します。 1 つ以上の InstallShield プロジェクト パッケージを含むスイート/アドバンスト UI プロジェクトのリリースをビルドすると、関連する InstallShield プロジェクト内の指定されたリリースが最初にビルドされて、生成されるスイート/アドバンスト UI インストールにそれらがパッケージとして含まれます。
リリース	基本の MSI プロジェクト InstallScript プロジェクト	選択されたパッケージと関連付ける InstallShield プロジェクトのリリースを選択します。 1 つ以上の InstallShield プロジェクト パッケージを含むスイート/アドバンスト UI プロジェクトのリリースをビルドすると、関連する InstallShield プロジェクト内の指定されたリリースが最初にビルドされて、生成されるスイート/アドバンスト UI インストールにそれらがパッケージとして含まれます。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
場所	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	パッケージの場所を指定します。選択可能なオプションは以下のとおりです: • **[ソース メディアからコピーする]** – パッケージとそのファイルをソース メディアに格納します。 アドバンスト UI またはスイート/アドバンスト UI インストールを CD、DVD またはローカル ネットワークなどの固定メディアから非圧縮で実行する場合は、このオプションを使用します。 • **Setup.exe から抽出する** – パッケージとそのファイルを Setup.exe に圧縮して、必要に応じて実行時に抽出します。 アドバンスト UI またはスイート/アドバンスト UI インストール全体を Setup.exe に完全に含める場合、このオプションを選択します。[Web からダウンロードする] オプションを選択すると、インストールが小さくなりダウンロード時間も短縮されますが、[Setup.exe から抽出する] オプションは完全に独立したインストールを提供します。 • **Web からダウンロードする** – パッケージとそのファイルファイルを、必要に応じて、パッケージに指定された URL からダウンロードします。 このオプションは、インストールがインターネットからダウンロードされ、かつ、アドバンスト UI またはスイート/アドバンスト UI パッケージのサイズとダウンロード時間を最小化したい場合に推奨されます。アドバンスト UI またはスイート/アドバンスト UI パッケージは、適切なバージョンが既にターゲット システム上にある場合はダウンロードされません。 "場所" 設定で選択するオプションは、[リリース] ビューでオーバーライドされることがありますので、ご注意ください。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
URL	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	選択されたパッケージおよびそのフォルダーとファイルが含まれるルート フォルダーの URL を入力します。パッケージを起動する必要がある場合、実行時にパッケージおよび関連ファイルが、この場所からターゲット システムにダウンロードされます。 この設定は、"場所" 設定で [Web からダウンロードする] が選択されている場合に使用できます。
パッケージの種類	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	 ***プロジェクト:*** *スイート/アドバンスト UI プロジェクトでは、この設定は編集に使用できます。アドバンスト UI プロジェクトでは、プライマリ パッケージの数が 1 つのみサポートされているため、この設定は読み取り専用になっています。* スイート/アドバンスト UI プロジェクトの場合: このパッケージがターゲット システム上に存在するかどうかによって、スイート/アドバンスト UI インストールを初回インストール モードまたはメンテナンス モードで実行するかどうかを示す、パッケージの種類を選択します。 この設定で選択できるオプションは、次のとおりです: • **プライマリ** – 選択されたパッケージは、アドバンスト UI またはスイート/アドバンスト UI インストールのメインの部分です。 実行時、ターゲット システム上からインストールのプライマリ パッケージのすべてが不足している場合、インストールは初回インストールとして実行します。ターゲット システム上にスイート インストールのいずれかのプライマリ パッケージが存在する場合、インストールはメンテナンス モードで実行します。 • **依存ファイル** – 選択されたパッケージは、アドバンスト UI またはスイート/アドバンスト UI インストールをどのモードで実行するかを決定する要因として見なすことはできません。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
キャッシュ パス	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	キャッシュされたパッケージおよびその他のパッケージ ファイルを保存するエンド ユーザーのシステム上の場所を指定します。ハード コード化された値、たとえば **C:¥CachedFiles** を入力できますが、パスにはリストされているインストール先プロパティー値を使用することが推奨されます。次は、デフォルトの値です: **[LocalAppDataFolder]Downloaded Installations** [リリース] ビューの Setup.exe タブでは、ターゲット システム上で実行されるパッケージで、キャッシュ パスが定義されている場合に、ターゲット システム上にパッケージをキャッシュするかどうかを指定できます。非圧縮のリリースをビルドする場合は、ターゲット システム上でパッケージをキャッシュしないことをお勧めします。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
デジタル証明書ファイル	.appx	Windows アプリ パッケージをサポートする Windows のバージョンは、パッケージのソースを信頼しない限り、サイドロード パッケージをインストールしません。たとえば Visual Studio で生成されたカスタム証明書を使用する場合、スイート/アドバンスト UI インストールではターゲット システムの証明書ストアにカスタム証明書を追加することで、Windows がパッケージのソースを信頼するように設定できます。スイート/アドバンスト UI インストールは、.cer ファイルをサイドロード パッケージに含めて、ここでそれを指定した場合にこの処理を行います。
昇格された権限が必要	.msi、 .msp、 .exe、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	パッケージが Windows Vista 以降および Windows Server 2008 以降のシステム上で昇格された権限を必要とするかどうかを指定します。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
マイナー アップグレードの処理	.msi、基本の MSI プロジェクト	ターゲット システム上にパッケージの以前のバージョンが存在する場合に起こる、適切な動作を示します。選択可能なオプションは以下のとおりです: • **なし** – アドバンスト UI またはスイート/アドバンスト UI インストールはパッケージをマイナー アップグレード モードで実行せずに起動します。つまり、アドバンスト UI またはスイート/アドバンスト UI インストールは、このパッケージの REINSTALL または REINSTALLMODE プロパティを設定しません。 • **自動** – アドバンスト UI またはスイート/アドバンスト UI インストールは REINSTALL プロパティを *ALL* に、および REINSTALLMODE プロパティを *vomus* に設定して、パッケージをアップグレード モードで起動します。エンド ユーザーには、製品がアップグレードされる事を通知しません。 • **ユーザーに確認** – アドバンスト UI またはスイート/アドバンスト UI インストールは 2 番目のウィンドウを表示して、エンド ユーザーが処理の続行を希望するかどうかを問い合わせます。エンド ユーザーが続行を希望した場合、アドバンスト UI またはスイート/アドバンスト UI インストールはアップグレード モードでパッケージを起動します。エンド ユーザーが続行しないことを選択した場合、パッケージは実行しません。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
レポートされるステータス メッセージ	.msi、 .msp、 基本の MSI プロジェクト	アドバンスト UI またはスイート/アドバンスト UI インストールのユーザー インターフェイスで表示可能なステータス メッセージの種類を指定します。選択可能なオプションは以下のとおりです: • **アクション テキスト** – アドバンスト UI またはスイート/アドバンスト UI インストールのステータス メッセージは、パッケージに含まれる標準アクションおよびカスタム アクションからのアクション テキストを含みます。 • **アクション テキストおよびアクション データ** – アドバンスト UI またはスイート/アドバンスト UI インストールのステータス メッセージは、およびアクション テキストパッケージに含まれる標準アクションおよびカスタム アクションからのアクション データを含みます。 アドバンスト UI またはスイート/アドバンスト UI インストールの実行時、ISParcelStatus プロパティは、ISParcelStatus プロパティのアクション テキストおよびアクション データ (適切な場合) によって更新されます。アクション データは、デフォルトで、アドバンスト UI またはスイート/アドバンスト UI インストールの InstallationProgress ウィザード ページで表示されます。 たとえば、.msi パッケージの InstallFiles アクションを実行中に、ISParcelStatus プロパティはアクション テキスト "新しいファイルをコピーしています" で更新されて、エンド ユーザーに対してインストールの現在の進行状況を通知します。ステータス メッセージにアクション データを含める場合は、さらにターゲット システム上にインストール中の各ファイルの名前、ディレクトリ、およびサイズといったアクション データで ISParcelStatus プロパティが更新されます。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
共有	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	このパッケージを共有としてマークするかどうかを指定します。 共有パッケージ機能を使って、2 以上のアドバンスト UI またはスイート/アドバンスト UI インストールが 1 つのパッケージを共有している場合に、アドバンスト UI およびスイート/アドバンスト UI 製品のすべてが削除されるまで、ターゲット システムにそのパッケージが保持されるようにします。 この設定で [はい] を選択する場合、パッケージを共有するすべてのアドバンスト UI およびスイート/アドバンスト UI プロジェクトで、必ず同じパッケージ GUID を使用してください。 詳細については、「異なるアドバンスト UI およびスイート/アドバンスト UI インストール間で、共通パッケージを共有する」を参照してください。
ログの有効化	.msi、 .msp、 .exe、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI インストールが、/log コマンドライン パラメーターが使われて、コマンドラインから起動される場合、ログ ファイルを生成するかどうかを指定します。パッケージにログ機能がサポートされていない場合、そのパッケージに対して、ログ機能を有効にしないことをお勧めします。 [はい] を選択した場合、この設定の下にあるサブ設定も必要に応じて構成してください。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
ログファイル	.msi、 .msp、 基本の MSI プロジェクト	この設定は、"ログの有効化" 設定で [はい] が選択された時に有効になります。 ログ ファイルの名前を指定します。ファイルのパスを使用しないでください。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターによって、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。 この設定を空白のままにしておいた場合、インストールで作成されたログ ファイルに、*PackageGUID*.log という名前が使用されます。*PackageGUID* は、[パッケージ] ビューの "パッケージ GUID" 設定でパッケージに割り当てられた GUID です。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。
ログ オプション	.msi、 .msp、 基本の MSI プロジェクト	この設定は、"ログの有効化" 設定で [はい] が選択された時に有効になります。 ログ ファイルを生成するとき、パッケージで使用する Windows Installer ログ /L フラグを指定します。たとえば、すべてを詳細に記録するログ ファイルを作成する場合は、この設定で、次のように入力します: *v その他の使用可能なフラグについては、/L の説明を参照してください。 この設定を空白のままにしておくと、アスタリスク (*) と v フラグがログ ファイルの生成時に使用されます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
ログのコマンドライン	.exe、InstallScript、InstallScript プロジェクト	この設定は、"ログの有効化" 設定で [はい] が選択された時に有効になります。 アドバンスト UI またはスイート/アドバンスト UI インストールで、ログ機能を有効にするために .exe パッケージに渡すコマンドラインを指定します。サポートされている適切なフラグを含めます。構成中の .exe パッケージでログ機能がサポートされている場合、ファイルが作成されるディレクトリのパスの代わりに、アドバンスト UI またはスイート/アドバンスト UI のプロパティ ISLogDir を参照するパスを角かっこで囲んで含めます。 たとえば、InstallShield でビルドされた Setup.exe ファイルによって実行される .msi パッケージについて、すべてを詳細に記録するログ ファイルを生成する場合、次のコマンドラインを入力します: /v"/l*v ¥"[ISLogDir]FileName.log¥"" アドバンスト UI またはスイート/アドバンスト UI インストールによって [ISLogDir] が、ログ ファイルを含むフォルダーへのパスで置き換えられます。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターを使用すると、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
検出条件	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	この設定を使って、パッケージがターゲット システム上に既にインストール済みであるかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。たとえば、パッケージが特定のファイルまたはレジストリ エントリをインストールする場合、インストールが検索するファイルまたはレジストリ キーを指定する条件を作成できます。 1 つ以上の新しい検出条件を追加するには、この設定で [新しい条件] ボタンをクリックします。InstallShield によって、"検出条件" 設定の下に新しい行が追加されます。この行のリストから All、Any、または None のうち適切なオプションを選択します。次にこの行で、[新しい条件] ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。 1 つ以上の条件ステートメントが構成されると、"検出条件" 設定には **(条件)** と表示されます。何も構成されていない場合、"検出条件" 設定には **(空白)** と表示されます。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
対象条件	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	この設定を使って、ターゲット システムで実行するパッケージに必要な要件を満たしているかどうかを判断するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。たとえば、パッケージが 64 ビット システム上でのみ実行する場合、条件に x64 プラットフォーム要件を設定できます。これによって、アドバンスト UI またはスイート/アドバンスト UI インストールは 64 ビット システム上でのみパッケージを起動します。また、エンド ユーザーが現在のパッケージ バージョンを使って、将来リリースされる新しいバージョンを上書きインストールすることを防ぐために、対象条件を設定することもできます。 1 つ以上の新しい対象条件を追加するには、この設定で [新しい条件] ボタンをクリックします。InstallShield によって、"対象条件" 設定の下に新しい行が追加されます。この行のリストから All、Any、または None のうち適切なオプションを選択します。次にこの行で、[新しい条件] ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。 1 つ以上の条件ステートメントが構成されると、"対象条件" 設定には **(条件)** と表示されます。何も構成されていない場合、"対象条件" 設定には **(空白)** と表示されます。

テーブル 104 • [共通] タブにある全般設定 (続き.)

設定	パッケージ ファイルの種類	説明
Windows の機能	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト*** *設定は、スイート/アドバンスト UI プロジェクトで使用できます。* この設定を使って、選択されたパッケージが必要とする 1 つ以上の Windows の役割と機能を、Windows Vista 以降または Windows Server 2008 以降が搭載されているターゲット システム上で有効にすることを指定できます。実行時、このパッケージがインストールの対象であり、無効化されている 1 つ以上の Windows の役割または機能を必要とする場合、スイート/アドバンスト UI インストールは、パッケージを起動する前にこれらを有効化します。 必要な Windows の役割または機能を指定するには、この設定で [新しい Windows の機能] ボタンをクリックして、有効なオプションから 1 つを選択します。カスタム オプションを使って、この設定にオプションとしてリストされていない任意の Windows の役割または機能を指定できます。 オプションを選択すると、InstallShield によって "Windows の機能" 設定の下に新しい "Windows の機能" 行が追加されます。必要に応じて "Windows の機能" 設定を構成します。 詳細については、「スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する」を参照してください。
リリース フラグ	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	リリース フラグを使って、選択で、アドバンスト UI またはスイート/アドバンスト UI インストールの異なるビルドで、このパッケージを含めたり除外したりできるようにするには、この機パッケージを識別するリリース フラグを入力します (複数可)。複数のフラグは、コンマで区切ります。

操作の設定

パッケージの "操作" 設定を使って、アドバンスト UI またはスイート/アドバンスト UI インストールがパッケージをインストール、削除、修復、または変更する時に発生する動作などの情報を指定します。この情報には、パッケージの起動時に使用するコマンドラインや、ターゲット システムの再起動が必要なときに発生する動作などが含まれます。

メモ: 一部の設定は、すべての種類のアドバンスト UI およびスイート/アドバンスト UI パッケージ フォーマットに適用できません。

テーブル 105 • [共通] タブにある操作設定

設定	パッケージ ファイルの種類	説明
Install	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI インストールが初回インストール モードで実行するとき、このパッケージを初回インストールとして起動するかどうかを指定します。インストール操作はターゲット システムがパッケージのインストール条件を満たしている場合で、パッケージの製品がまだインストールされていない場合にのみ適用されます。 この設定で [はい] を選択する場合、この設定の下にあるサブ設定も必要に応じて構成してください。[いいえ] を選択した場合、この設定の下にあるサブ設定は無効です。
削除	.msi、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI インストールが削除モードで実行する場合、このパッケージがインストールした製品を削除するかどうかを指定します。削除操作は、ターゲット システムがパッケージのインストール条件を満たしている場合で、パッケージの製品がインストールされている場合にのみ適用されます。 この設定で [はい] を選択する場合、この設定の下にあるサブ設定も必要に応じて構成してください。[いいえ] を選択した場合、この設定の下にあるサブ設定は無効です。
修復	.msi、 .exe、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI インストールが修復モードで実行する場合、このパッケージによってインストールされた製品を修復する修復モードでパッケージを起動するかどうかを指定します。修復操作はターゲット システムがパッケージの対象条件を満たしている場合で、パッケージの製品がインストールされている場合にのみ適用されます。 この設定で [はい] を選択する場合、この設定の下にあるサブ設定も必要に応じて構成してください。[いいえ] を選択した場合、この設定の下にあるサブ設定は無効です。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
変更	.msi、 .msp、 .exe、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI インストールが変更モードで実行する場合、このパッケージによってインストールされた製品をエンド ユーザーがインストール、削除、または変更できるメンテナンス モードでパッケージを起動するかどうかを指定します。変更操作はターゲット システムがパッケージの対象条件を満たしている場合で、パッケージの製品がインストールされている場合にのみ適用されます。 この設定で [はい] を選択する場合、この設定の下にあるサブ設定も必要に応じて構成してください。[いいえ] を選択した場合、この設定の下にあるサブ設定は無効です。
ターゲット	.msi、 .msp、 .exe、 InstallScript	この設定は、.msi、.msp、.exe、および InstallScript パッケージのインストール設定の下にあるサブ設定です。また、.exe パッケージの ″削除″、″修復″、および ″変更″ 設定の下にもあります。 アドバンスト UI またはスイート/アドバンスト UI インストールが呼び出すパッケージ内のファイル (.msi ファイル、.msp ファイル、または実行可能ファイルのセットアップランチャー) を選択します。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
EXE コマンド ライン	.exe	この設定は、″インストール″、″削除″、″修復″、および ″変更″ 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスで実行するときにターゲット ファイル (″ターゲット″ 設定で選択されたファイル) を起動するのに使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 エンド ユーザーによる操作なしですべてのカスタマイズを完了する場合、.exe パッケージをサイレントで実行するコマンドライン パラメーターを含めることができます。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
MSI コマンド ライン	.msi、 基本の MSI プロジェクト	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスで実行するときにターゲット ファイル ("ターゲット" 設定で選択されたファイル) を起動するのに使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 .msi パッケージのインストールおよび削除操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer プロパティです。.msi パッケージの修復および変更操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer 機能プロパティです。 Windows Installer プロパティを入力する場合、以下の形式を使用します: MYPROPERTYNAME=MyPropertyValue アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msi パッケージは常にサイレントで起動されますので注意してください。そのため、.msi パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
MSI コマンド ライン	.msp	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスで実行するときにターゲット ファイル ("ターゲット" 設定で選択されたファイル) を起動するのに使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 .msp パッケージを使ってすべての機能をアップデートするには、以下のようなコマンドライン プロパティを入力します: REINSTALLMODE=vomus REINSTALL=ALL .msp パッケージに含まれている特定の機能のみをアップデートするには、アップデートするコンマ区切りの機能リストに REINSTALL を設定します。 REINSTALLMODE=vomus REINSTALL=Feature1,Feature3,Feature5 アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msp パッケージは常にサイレントで起動されますので注意してください。そのため、.msp パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
コマンド ライン	InstallScript、InstallScript プロジェクト	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスと共に実行するとき、選択したターゲット ファイル ("ターゲット" 設定で選択した data1.hdr ファイル) に関連付けられている InstallScript パッケージを起動するために使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 InstallScript パッケージの [インストール] 操作と [変更] 操作を構成する場合、コマンドラインで、アドバンスト UI およびスイート/アドバンスト UI のプロパティ ISFeatureInstall と ISFeatureRemove を使用できます。これらのプロパティでは、次のように、機能の名前をコンマ区切りで設定します: ISFeatureInstall=Feature1,Feature2 ISFeatureRemove=Feature3 上記の例では、**Feature1** と **Feature2** は、インストールされるように選択されています。**Feature3** は、存在していれば、削除されるように選択されています。 アドバンスト UI またはスイート/アドバンスト UI インストールでは、デフォルトで、InstallScript パッケージはサイレントで起動されますので注意してください。そのため、InstallScript パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
EXE サイレント コマンド ライン	.exe	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがサイレントで実行されるとき (ユーザー インターフェイスなしの実行)、ターゲット ファイル ("ターゲット" 設定で選択されたファイル) を起動するために使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 エンド ユーザーによる操作なしですべてのカスタマイズを完了する場合、.exe パッケージをサイレントで実行するコマンドライン パラメーターを含めることができます。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
MSI サイレント コマンド ライン	.msi、 基本の MSI プロジェクト	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがサイレントで実行されるとき (ユーザー インターフェイスなしの実行)、ターゲット ファイル ("ターゲット" 設定で選択されたファイル) を起動するために使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 .msi パッケージのインストールおよび削除操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer プロパティです。.msi パッケージの修復および変更操作を構成している場合、使用できるコマンドライン パラメータの種類は、Windows Installer 機能プロパティです。 Windows Installer プロパティを入力する場合、以下の形式を使用します: MYPROPERTYNAME=MyPropertyValue アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msi パッケージは常にサイレントで起動されますので注意してください。そのため、.msi パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
MSP サイレント コマンド ライン	.msp	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがサイレントで実行されるとき (ユーザー インターフェイスなしの実行)、ターゲット ファイル ("ターゲット" 設定で選択されたファイル) を起動するために使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 .msp パッケージを使ってすべての機能をアップデートするには、以下のようなコマンドライン プロパティを入力します: REINSTALLMODE=vomus REINSTALL=ALL .msp パッケージに含まれている特定の機能のみをアップデートするには、アップデートするコンマ区切りの機能リストに REINSTALL を設定します。 REINSTALLMODE=vomus REINSTALL=Feature1,Feature3,Feature5 アドバンスト UI またはスイート/アドバンスト UI インストールでは、.msp パッケージは常にサイレントで起動されますので注意してください。そのため、.msp パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
サイレント コマンド ライン	InstallScript、InstallScript プロジェクト	適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがサイレントで実行されるとき (ユーザー インターフェイスなしの実行)、選択したターゲット ファイル ("ターゲット" 設定で選択した data1.hdr ファイル) に関連付けられている InstallScript パッケージを起動するために使用するコマンドラインを指定します。この設定には、ファイル名を含めないでください。 InstallScript パッケージの [インストール] 操作と [変更] 操作を構成する場合、コマンドラインで、アドバンスト UI およびスイート/アドバンスト UI のプロパティ ISFeatureInstall と ISFeatureRemove を使用できます。これらのプロパティでは、次のように、機能の名前をコンマ区切りで設定します: ISFeatureInstall=Feature1,Feature2 ISFeatureRemove=Feature3 上記の例では、**Feature1** と **Feature2** は、インストールされるように選択されています。**Feature3** は、存在していれば、削除されるように選択されています。 アドバンスト UI またはスイート/アドバンスト UI インストールでは、デフォルトで、InstallScript パッケージはサイレントで起動されますので注意してください。そのため、InstallScript パッケージのユーザー インターフェイスを隠すコマンドライン パラメータを渡す必要はありません。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 この設定で使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
終了動作	.exe	この設定は、″インストール″、″削除″、″修復″、および ″変更″ 設定の下にあるサブ設定です。 パッケージが実行された後でもターゲット システム上で検出条件が満たされない場合、以下のいずれかが True である可能性があります。 • 実行可能ファイル パッケージの実行が失敗した。 • パッケージに指定された検出条件が不正確である。 たとえば、ターゲット システムに特定のファイルが存在しない場合に、検出条件が実行可能ファイルを起動する必要があることを示す可能性があります。アドバンスト UI またはスイート/アドバンスト UI インストールによって実行可能アフィル パッケージが実行された後でもファイルが不足している場合、条件に誤りがある可能性があります。 1 つ以上の検出条件が、パッケージを起動する必要があることを示し続けている場合に発生する動作を指定します。選択可能なオプションは以下のとおりです: • **セットアップを続行するかどうかを確認する** — エンド ユーザーに対してアドバンスト UI またはスイート/アドバンスト UI インストールを続行するかどうかを指定するようにプロンプトするメッセージ ボックスを表示する場合、このオプションを選択します。 • **セットアップを終了する** — アドバンスト UI またはスイート/アドバンスト UI インストールが、処理を続行せずに終了する場合は、このオプションを選択します。 • **セットアップを続行する** — アドバンスト UI またはスイート/アドバンスト UI インストールが、基本的に実行可能ファイルで満たされなかった条件を無視して (必要な場合に) スイート内の次のパッケージに進む場合は、このオプションを選択します。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
再起動要求	.msi、 .msp、 .exe、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 「アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動」でも説明されているように、アドバンスト UI またはスイート/アドバンスト UI パッケージをインストールするためにターゲット マシンを再起動する必要がある場合があります。 パッケージが、ターゲット システムの再起動を必要とする場合の動作を指定します。選択可能なオプションは以下のとおりです: • **マシンの再起動を許可する** – 再起動の必要性が検出された場合、アドバンスト UI またはスイート/アドバンスト UI インストールは終了して、パッケージによる再起動がトリガされます。パッケージによって、ターゲット システムが再起動された場合、再起動の後、次のパッケージが、アドバンスト UI またはスイート/アドバンスト UI で継続されます。パッケージによって、再起動がトリガされなかった場合、アドバンスト UI またはスイート/アドバンスト UI インストールは次の再起動の後に再開します。 • **再起動要求を無視する** – アドバンスト UI またはスイート/アドバンスト UI インストールは、ターゲット システムを再起動せずに続行します。つまり再起動が必要であると検出されても、アドバンスト UI またはスイート/アドバンスト UI インストールがこれをスキップするには、このオプションを選択します。パッケージによってターゲット システムが再起動された場合、アドバンスト UI またはスイート/アドバンスト UI インストールは、インストールの次のパッケージを続けて実行します。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
再起動要求 (続き)		• **プロンプトを遅延してから、終了またはマシンを再起動する** – パッケージでターゲット システムの再起動が必要な場合、最後のパッケージが完了した後、次のパッケージが再起動をトリガするまで、またはアドバンスト UI またはスイート/アドバンスト UI インストールが終了するまで再起動が遅延されます。次のパッケージによって再起動が発生する場合、再起動動作は、次のパッケージに構成された再起動動作に従います。アドバンスト UI またはスイート/アドバンスト UI インストールの最後で再起動が発生する場合、アドバンスト UI またはスイート/アドバンスト UI インストールは、エンド ユーザーに対して再起動を行うかどうかを問い合わせるメッセージ ボックスを表示します。エンド ユーザーが再起動を許可しないことを選択した場合、アドバンスト UI またはスイート/アドバンスト UI インストールはターゲット システムを再起動せずに終了します。 • **プロンプトしてから、終了またはマシンを再起動する** – パッケージがターゲット システムの再起動を必要としていることをアドバンスト UI またはスイート/アドバンスト UI インストールが検出した場合、アドバンスト UI またはスイート/アドバンスト UI インストールはエンド ユーザーに対して再起動を行うかどうかを問い合わせるメッセージ ボックスを表示します。エンド ユーザーが再起動を許可した場合、アドバンスト UI またはスイート/アドバンスト UI インストールは再起動のあと再開します。エンド ユーザーが再起動を許可しないことを選択した場合、アドバンスト UI またはスイート/アドバンスト UI インストールはターゲット システムを再起動せずに終了します。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
再起動要求 (続き)		• **常にプロンプトしてから、終了またはマシンを再起動する** – パッケージのインストールが完了したとき、アドバンスト UI またはスイート/アドバンスト UI インストールは再起動の必要性を検出しなかった場合でも、エンド ユーザーに対して再起動を行うかどうかを問い合わせるメッセージ ボックスを表示します。エンド ユーザーが再起動を許可した場合、アドバンスト UI またはスイート/アドバンスト UI インストールは再起動のあと再開します。エンド ユーザーが再起動を許可しないことを選択した場合、アドバンスト UI またはスイート/アドバンスト UI インストールはターゲット システムを再起動せずに終了します。 • **常にマシンを再起動する** – アドバンスト UI またはスイート/アドバンスト UI インストールは、再起動の必要性を検出したかどうかに関わらず、ターゲット システムを再起動します。アドバンスト UI またはスイート/アドバンスト UI インストールは、再起動後、継続して実行されます。 最適なオプションを選択する方法については、「再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの動作を指定する」を参照してください。
再起動コード	.exe	この設定は、"インストール"、"削除"、"修復"、および "変更" 設定の下にあるサブ設定です。 選択されたパッケージが製品のインストール後にターゲットマシンの再起動を要求する場合、この設定に戻りコードを入力します。 複数のリターンコードが存在する場合、各コードはカンマで区切ってリストしてください。 アドバンスト UI またはスイート/アドバンスト UI パッケージとして起動するファイルの戻りコードが分からない場合は、パッケージの作成者に問い合わせてください。 再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの詳細は、「アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動」を参照してください。

テーブル 105 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
アプリケーションの強制シャットダウン	.appx	インストール実行時に選択されたサイドロード アプリケーション パッケージ (またはこのパッケージに依存する任意のパッケージ) が使用中の場合、登録処理を行うためにパッケージに関連付けられたプロセスを強制的にシャットダウンすることを要求できます。この設定では、この状況下でアプリケーションをシャットダウンするかどうかを指定します。

構成の設定

"Web 配置パッケージの構成" 設定を使用して、"Web 配置" 設定の様々な値を指定します。詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

テーブル 106 • [共通] タブにある操作設定

設定	パッケージ ファイルの種類	説明
コンピューター名	Web 配置	Web 配置パッケージの配置先となるサーバーの名前を指定します。 プロジェクトにパッケージを追加するとき、InstallShield のデフォルトでこの設定に割り当てられているスイート/アドバンスト UI またはアドバンスト UI プロパティをそのままに残すか、独自のプロパティを指定することができます (角括弧で囲む)。代わりに、ハードコード化された値を指定することもできます。 詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。
ユーザー名	Web 配置	Web 配置パッケージを配置するエンド ユーザーのユーザー名を指定します。 プロジェクトにパッケージを追加するとき、InstallShield のデフォルトでこの設定に割り当てられているスイート/アドバンスト UI またはアドバンスト UI プロパティをそのままに残すか、独自のプロパティを指定することができます (角括弧で囲む)。代わりに、ハードコード化された値を指定することもできます。 詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

テーブル 106 • [共通] タブにある操作設定 (続き.)

設定	パッケージ ファイルの種類	説明
パスワード	Web 配置	Web 配置パッケージを配置するときに使用するユーザーアカウントのパスワードを指定します。 プロジェクトにパッケージを追加するとき、InstallShield のデフォルトでこの設定に割り当てられているスイート/アドバンスト UI またはアドバンスト UI プロパティをそのままに残すか、独自のプロパティを指定することができます (角括弧で囲む)。代わりに、ハードコード化された値を指定することもできます。 詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。
Site	Web 配置	Web 配置パッケージの配置先となるサイトを指定します。 プロジェクトにパッケージを追加するとき、InstallShield のデフォルトでこの設定に割り当てられているスイート/アドバンスト UI またはアドバンスト UI プロパティをそのままに残すか、独自のプロパティを指定することができます (角括弧で囲む)。代わりに、ハードコード化された値を指定することもできます。 詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。
パラメーター	Web 配置	Web 配置パッケージのパラメーター XML ファイルに構成可能な様々なパラメーターが含まれている場合、"パラメーター" サブ設定を使って、これらのパラメーターの値を指定できます。 プロジェクトにパッケージを追加するとき、InstallShield のデフォルトでこれらのサブ設定に割り当てられているスイート/アドバンスト UI またはアドバンスト UI プロパティをそのままに残すか、独自のプロパティを指定することができます (角括弧で囲む)。代わりに、ハードコード化された値を指定することもできます。 詳細については、「Web 配置パッケージをスイート/アドバンスト UI プロジェクトに追加する」を参照してください。

Events の設定

プロジェクト: *[イベント] 領域は、スイート/アドバンスト UI プロジェクトで使用できます。*

パッケージの "イベント" 設定を使って、パッケージのインストール、削除、修復、または変更時にスイート/アドバンスト UI インストールが起動する実行時のアクションをスケジュールします。詳細については、「スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する」を参照してください。

テーブル 107 • [共通] タブにあるイベントの設定

設定	パッケージ ファイルの種類	説明
パッケージの構成	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	この設定を使って、スイート/アドバンスト UI インストールが、パッケージのインストール、削除、修復、または変更を行う前に起動する 1 つ以上の実行時のアクションを指定できます。アクションは、[イベント] ビューの [アクション] エクスプローラーの下に追加済みでなくてはなりません。 アクションを指定するには、この設定で [イベントの追加] ボタンをクリックしてから、アクションを選択します。InstallShield によって、"パッケージの構成" 設定の下に新しい "アクション" 設定が追加され、その値が選択されたアクションの名前に設定されます。"アクション" 設定のボタンを使って、スイート/アドバンスト UI インストールでアクションを起動するかどうかを評価するのに使用する 1 つ以上の条件を指定します。
パッケージの構成	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	この設定を使って、スイート/アドバンスト UI インストールが、パッケージのインストール、削除、修復、または変更の後に起動する 1 つ以上の実行時のアクションを指定できます。アクションは、[イベント] ビューの [アクション] エクスプローラーの下に追加済みでなくてはなりません。 アクションを指定するには、この設定で [イベントの追加] ボタンをクリックしてから、アクションを選択します。InstallShield によって、"パッケージの構成" 設定の下に新しい "アクション" 設定が追加され、その値が選択されたアクションの名前に設定されます。"アクション" 設定のボタンを使って、スイート/アドバンスト UI インストールでアクションを起動するかどうかを評価するのに使用する 1 つ以上の条件を指定します。

テーブル 107 • [共通] タブにあるイベントの設定 (続き.)

設定	パッケージ ファイルの種類	説明
アクション	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	この読み取り専用設定は、選択されたパッケージと関連付けられ、[イベント] ビューで定義されたアクションの名前を表示します。このパッケージにアクションを実行するかどうかを判別する条件を定義するには、"アクション" 設定の下にあるサブ設定を必要に応じて構成します。 パッケージに複数のアクションを関連付ける場合、実行時に起動される順番に、上から下に時系列でアクションをリストします。アクションの順番を変更するには、移動させるアクションの設定で [条件を移動] ボタンをポイントしてから、適切な移動方向 ([上へ移動] または [下へ移動]) をクリックします。
条件	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	この設定を使って、インストールが選択されたパッケージのアクションを実行するかどうかを評価するためにスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。 1 つ以上の新しいアクション条件を追加するには、この設定で [新しい条件] ボタンをクリックします。InstallShield によって、"条件" 設定の下に新しい行が追加されます。この行のリストから All、Any、または None のうち適切なオプションを選択します。次にこの行で、[新しい条件] ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。 1 つ以上の条件ステートメントが構成されると、"条件" 設定には **(条件)** と表示されます。何も構成されていない場合、"条件" 設定には **(空白)** と表示されます。

テーブル 107 • [共通] タブにあるイベントの設定 (続き.)

設定	パッケージ ファイルの種類	説明
表示テキスト	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェクト InstallScript プロジェクト	選択されたアクションについて説明するテキストを入力します。たとえば、アクションがターゲット システムを構成するスクリプトを実行している場合、次の文字列を入力できます: システムの構成中 InstallationProgress ウィザード ページが表示されるときに、インストールがアクションを起動する場合、入力されたテキストがウィザード ページに表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定に値を入力するとき、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

Features

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[パッケージ] ビューの [機能] タブには、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの機能ツリーが表示されます。このタブで、選択した パッケージ を含める機能のチェック ボックスをすべて選択します。選択したパッケージを含めない機能のチェック ボックスをすべてクリアします。

ダイナミック リンクの設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[パッケージ] ビューの [ダイナミック リンク] ペインでは、プロジェクト内のパッケージに作成されたダイナミック リンクの設定が表示されます。"ソース" 設定と "フィルター" 設定は、すべてのダイナミック リンクに設定可能です。他の設定は、"フィルター" 設定の下に、サブ設定として表示されていることがあります。

テーブル 108 ・ ダイナミック リンクの設定

設定	説明
ソース	この設定では、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるパッケージを含むソース フォルダーのパスが表示されます。これはまた、パッケージに動的に含まれているファイルのソースでもあります。 場所をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。
フィルター	この設定で、ファイルとフォルダーをプロジェクトに含めたり、またはそれらをプロジェクトから除外するために構成するフィルターを指定できます (複数可)。 フィルターを追加するには、この設定で **[新しいフィルター]** ボタンをクリックします。フィルターの定義に使用できる新しい行が追加されます。 フィルターを削除するには、削除するフィルターのサブ設定をクリックして、そのサブ設定内にある **[このフィルターを削除]** ボタンをクリックします。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。

テーブル 108 • ダイナミック リンクの設定 (続き.)

設定	説明
含めるファイル	ダイナミック リンクに含めるダイナミック ファイルの名前を指定します。アスタリスク (*) をワイルドカード文字として使用できます。また、ソース フォルダーを参照する相対パスを含めることもできます。たとえば、**MyDirectory** という名のサブフォルダー内 (**MyDirectory** は、パッケージを含むフォルダーのサブフォルダーです) にある **ReadMe.txt** ファイルを参照する場合、次のようなフィルターを使うことができます: **.¥MyDirectory¥ReadMe.txt** InstallShield で、リリースに含める、または、リリースから除外するファイルおよびフォルダーがどう判別されるかなどを含む、さらに詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。
除外するファイル	ダイナミック リンクから除外するダイナミック ファイルの名前を指定します。アスタリスク (*) をワイルドカード文字として使用できます。また、ソース フォルダーを参照する相対パスを含めることもできます。たとえば、**MyDirectory** という名のサブフォルダー内 (**MyDirectory** は、パッケージを含むフォルダーのサブフォルダーです) にある **ReadMe.txt** ファイルを参照する場合、次のようなフィルターを使うことができます: **.¥MyDirectory¥ReadMe.txt** InstallShield で、リリースに含める、または、リリースから除外するファイルおよびフォルダーがどう判別されるかなどを含む、さらに詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。
含めるフォルダー	ファイルをダイナミック リンクに含めるダイナミック フォルダーの名前を指定します。アスタリスク (*) をワイルドカード文字として使用できます。また、ソース フォルダーを参照する相対パスを含めることもできます。たとえば、**MyLibrary** (**MyLibrary** は、パッケージを含むフォルダーのサブフォルダーです) を参照する場合、次のようなフィルターを使うことができます: **.¥MyLibrary** InstallShield で、リリースに含める、または、リリースから除外するファイルおよびフォルダーがどう判別されるかなどを含む、さらに詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。

テーブル 108 - ダイナミック リンクの設定 (続き.)

設定	説明
除外するフォルダー	ファイルをダイナミック リンクから除外するダイナミック フォルダーの名前を指定します。アスタリスク (*) をワイルドカード文字として使用できます。また、ソース フォルダーを参照する相対パスを含めることもできます。たとえば、**MyLibrary** という名のフォルダー (**MyLibrary** は、パッケージを含むフォルダーのサブフォルダーです) を参照する場合、次のようなフィルターを使うことができます: **.¥MyLibrary** InstallShield で、リリースに含める、または、リリースから除外するファイルおよびフォルダーがどう判別されるかなどを含む、さらに詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。

[DIM リファレンス] ビュー

プロジェクト: *[DIM リファレンス] ビューは、その後基本の MSI プロジェクトで使用できます。*

[DIM リファレンス] ビューを使って、基本の MSI プロジェクトにDIM プロジェクトのリファレンスを追加したり、管理したりすることができます。また、このビューでは、DIM の作成者が用意した関連する手順を表示したり、DIM リファレンスを基本の MSI プロジェクト内の機能に関連付けたり、DIM 内のファイルのデフォルトのインストール先を上書きしたり、カスタム アクションをスケジュールしたり、様々なことができます。

InstallShield の DIM プロジェクトは、コラボレーションを促進したいエンジニア チームのために提供されています。DIM プロジェクトを使うと、組織で、インストール開発を分担して、効率化を図ることができます。DIM は、機能サイズのプロジェクトで、インストール パッケージの論理的に別個に分かれている部分を構成する製品ファイル、ショートカット、レジストリ エントリ、テキスト ファイルの変更、IIS Web サイト、および要素など、関連するアイテムを集めたものです。DIM が作成されたら、そのリファレンスを 1 つ以上の基本の MSI プロジェクトに含めることができます。

[DIM リファレンス] ビューで DIM リファレンスを選択すると、いくつかのタブが表示されます:

- [全般] タブ
- [手順] タブ
- [ビルド オプション] タブ
- [機能] タブ
- [シーケンス] タブ

[全般] タブ

プロジェクト: *[DIM リファレンス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

[DIM リファレンス] ビューの [全般] タブでは、選択された DIM リファレンスに関する以下の読み取り専用の設定が表示されます。これら設定の値は実行時に使用または表示されませんので注意してください。これらの設定は、プロジェクトを内部的に識別するためのリファレンスとして使用できます。

テーブル 109 · [DIM リファレンス] ビューの [全般] タブの設定

設定	説明
File	この設定では、選択された DIM プロジェクト ファイル (.dim) の名前が表示されます。
名前	この設定は、この DIM が示す機能またはユニットの名前を示します。この設定は、DIM プロジェクトの [一般情報] ビューの "サブジェクト" 設定で構成されます。
DIM GUID	この設定では、DIM を一意に識別する GUID が表示されます。新しい DIM を作成すると常に、新しい GUID が自動生成されます。
パッケージ コード	この設定は、パッケージ コードを識別します。これは、作成した DIM の GUID です。
バージョン	この設定は、DIM のバージョン番号を示します。
説明	この設定では、DIM の説明が表示されます。この設定は、DIM プロジェクトの [一般情報] ビューの "タイトル" 設定で構成されます。
作成者	この設定は、この DIM の作成者の名前です。
リンク先	この設定では、選択された DIM プロジェクト ファイル (.dim) の保存場所が表示されます。

[手順] タブ

***プロジェクト·** [DIM リファレンス] ビューは、その後基本の MSI プロジェクトで使用できます。*

[DIM リファレンス] ビューの [手順] タブでは、DIM プロジェクトの作成者が [一般情報] ビューの "ビルドの手順" 設定で入力したガイダンス、ヒント、コメントが表示されます。これらの手順には、この DIM のリファレンスを含む基本の MSI プロジェクトを適切に構成するために必要な重要な情報が含まれていることがあります。

[ビルド オプション] タブ

***プロジェクト·** [DIM リファレンス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

[DIM リファレンス] ビューの [ビルド オプション] タブでは、選択された DIM リファレンスに関する以下の読み取り専用の設定が表示されます。

テーブル 110 • [DIM リファレンス] ビューの [ビルド オプション] タブの設定

設定	説明
Destination	DIM のファイルをインストールするターゲット システム上にあるフォルダーを選択するか、省略記号ボタン (...) をクリックして、ディレクトリを選択または作成します。この設定を利用して、DIM プロジェクトに設定された移動先をオーバーライドすることができます。この設定のデフォルト値は、DIM プロジェクトで設定された値です。 製品のデフォルトのインストール先フォルダーにファイルをインストールするには、**[INSTALLDIR]** を選択します。 インストール先を実行時に構成できるようにするには、選択するインストール先フォルダーに、必ずパブリック プロパティ (すべて大文字で表記) を指定してください。 コンポーネントの機能のインストール先フォルダーも設定することができます。コンポーネントだけでなく、機能のインストール先を設定した場合の効果については、「コンポーネントのインストール先と機能のインストール先の違い」を参照してください。
競合の解決	基本の MSI プロジェクト内の設定値と選択された DIM プロジェクトの設定値間で発生した競合の解決方法を指定します。選択可能なオプションは以下のとおりです: • **ベース プロジェクト値を使用する** – ビルド時に DIM データを基本の MSI インストールに結合した時に競合が発生した場合、DIM プロジェクト内の値の代わりに、基本の MSI プロジェクト内の値が使用されます。 • **DIM プロジェクト値を使用する** – ビルド時に DIM データを基本の MSI インストールに結合した時に競合が発生した場合、基本の MSI プロジェクト内の値の代わりに、DIM プロジェクト内の値が使用されます。 詳細については、「ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する」を参照してください。

[機能] タブ

プロジェクト: *[DIM リファレンス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

[DIM リファレンス] ビルドの [機能] タブでは、基本の MSI プロジェクトの機能ツリーが表示されます。このタブで、選択した DIM を含める機能のチェック ボックスをすべて選択します。選択した DIM を含めない機能のチェック ボックスをすべてクリアします。

[シーケンス] タブ

プロジェクト: *[DIM リファレンス] ビューは、その後基本の MSI プロジェクトで使用できます。*

シーケンスは、ファイル転送からユーザー インタフェースの表示までインストール中に実行されるすべてのアクションを制御します。これらのアクションにはシーケンス上の番号が付与され、小さい番号から順に実行されます。[DIM リファレンス] ビューの [シーケンス] タブで、DIM のアクションやダイアログをシーケンスに挿入したり、シーケンスのタイムラインを編集したりできます。詳細については、「DIM リファレンスからのカスタム アクションとダイアログをスケジュールする」を参照してください。

[アプリケーション データ] ビュー

プロジェクト: *[アプリケーション データ] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

アプリケーションデータには、プロジェクトに追加するすべてのファイルが含まれます。

ファイルとフォルダー

プロジェクト: *[ファイルとフォルダー] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ファイルとフォルダー] ビューで、プロジェクトにファイルを追加することができます。このビューは、Windows エクスプローラーと同様に機能するため、プロジェクトにファイルをドラッグアンドドロップすることができます。インストール プロジェクトでは、ファイルに関連付ける機能とコンポーネント、およびそのファイルへのインストール先のディレクトリを指定できます。

再配布可能ファイル

プロジェクト: *[再配布可能ファイル] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[再配布可能ファイル] ビューを使って、Windows Installer ベースのプロジェクトに InstallShield オブジェクトとマージ モジュールを追加できます。再配布可能ファイルをプロジェクトに含めて、インストールに個別機能を追加することができます。このタイプの機能の例としては、Visual Basic のランタイム .dll および Microsoft C のランタイムライブラリなどがあります。

[再配布可能ファイル] ビューを使って、InstallShield 前提条件を基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することもできます。InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

プロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖することができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。Setup.exe セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。

前提条件

プロジェクト: *[前提条件機能] ビューは、次のプロジェクト タイプで使用できます:*

- *InstallScript*

[前提条件] ビューを使って、InstallShield 前提条件を InstallScript プロジェクトに追加できます。

オブジェクト

プロジェクト: *[オブジェクト] ビューは、次のプロジェクト タイプで使用できます:*

- *InstallScript*
- *InstallScript オブジェクト*

[オブジェクト] ビューを使って、InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトにオブジェクトとマージ モジュールを追加できます。

[ファイルとフォルダー] ビュー

プロジェクト: *[ファイルとフォルダー] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *MSIX*
- *MSIX 変更パッケージ*

メモ: *Microsoft Visual Studio 内の [ファイルとフォルダー] ビューは、InstallShield の [ファイルとフォルダー] ビューと異なります。詳細については、「Visual Studio ソリューションにリファレンスを追加する」を参照してください。*

ほとんどのインストールの主な目的は、ファイルをソース メディアからターゲットマシンのインストール先に転送することです。InstallShield では、[ファイルとフォルダー] ビューでファイルをドラッグ アンド ドロップしてプロジェクトに追加できます。このビューには、ファイル エクスプローラーがあり、ソースマシンにあるフォルダーのファイルをインストール先マシンのフォルダーへドラッグ アンド ドロップすることができます。ファイル エクスプローラーの左側の 2 つのペインにはフォルダーが含まれ、右の 2 つのペインには、これらのフォルダー内にあるファイルが表示されます。

プロジェクト: *(InstallScript および 基本の MSI を含む) インストール プロジェクトでは、[ファイルとフォルダー] ビューはファイル エクスプローラー ペインの上にある [ビュー フィルター] リストを含みます。このリストには、プロジェクトのすべての機能が含まれています。このフィルターを使って、このビューのインストール先ペインのファイルとフォルダーを表示または非表示にします。*

- *このビューで特定の機能に属するファイルとフォルダーのみを表示するには、[ビュー フィルター] リストで機能を選択します。*
- *特定の機能にファイルまたはフォルダーを追加するには、[ビュー フィルター] リストで機能を選択します。次に、**[インストール先コンピューターのフォルダー]** ペインの適切な場所にファイルまたはフォルダーを追加します。*
- *プロジェクトに含まれるすべてのファイルとフォルダーを表示するには、[ビュー フィルター] リストで [すべてのアプリケーション データ] オプションを選択します。*

[ファイルとフォルダー] ビューには、次の 4 つのパネルがあります。

テーブル 111 ・ [ファイルとフォルダー] ビューのパネル

ペイン	説明
ソース コンピューターのフォルダー (左上)	**[ソース コンピューターのフォルダー]** ペインは、Windows エクスプローラーの左のペインに似ています。このペインには、ローカルまたはネットワーク上にあるフォルダーが含まれます。ここから、インストールに追加するファイルを含むフォルダーに移動できます。

テーブル 111 • [ファイルとフォルダー] ビューのパネル (続き.)

ペイン	説明
ソース コンピューターのファイル (右上)	**[ソース コンピューターのファイル]** ペインには、[ソース コンピューターのフォルダー] ペインで現在選択されているフォルダーに含まれるファイルが表示されます。このペインから、インストール先コンピューターのペインのインストール先フォルダーに、ファイルをドラッグできます。下部のペインにドロップするファイルは、インストール プロジェクトに追加されます。
インストール先コンピューターのフォルダー (左下)	**[インストール先コンピューターのフォルダー]** ペインで、ファイルをインストール先フォルダーまたはコンポーネントに追加したり、新しいインストール先フォルダーを作成できるほか、既存のコンポーネントを変更できます。
インストール先コンピューターのファイル (右下)	**[インストール先コンピューターのファイル]** ペインには、現在選択されているインストール先のフォルダーに追加されているファイルがすべて表示されます。このペインでファイルを右クリックすると、ファイルのプロパティのコピー、貼り付け、削除、またはプロパティの編集を行うことができます。(Windows Installer ベースのプロジェクトのみ)また、ファイルを右クリックして、コンポーネントのキーファイルとして設定、またはクリアすることもできます。

次の 3 つのいずれかの方法を実行して、インストール プロジェクトにファイルを追加できます:

テーブル 112 • プロジェクトへファイルを追加する方法

メソッド	説明
ファイルのドラッグアンドドロップ	[ファイルとフォルダー] ビューの [ファイル] エクスプローラーを使って、プロジェクトにファイルを追加できます。このビューの上部の 2 つのペインは、機能的に Windows エクスプローラーと同じです。下部の 2 つのペインは、ファイルの宛先を示します。 上部のペインから、下部のペインのインストール先フォルダーにソース ファイルをドラッグできます。InstallShield には、フォルダーのみをドラッグしてソース構造を保存する、追加のドラッグ アンド ドロップオプションを持つコンテキストメニューがあります。

テーブル 112 • プロジェクトへファイルを追加する方法 (続き.)

メソッド	説明
ダイナミック ファイル リンク	ファイルをプロジェクトに追加する 2 つ目の方法は、全体のフォルダーの内容、またはフォルダーの特定のファイルにリンクすることです。この方法を使用すると、インストールのファイルを含む、ローカルまたはネットワーク上の特定のフォルダーを指定できます。インストールをビルドするたびに、フォルダーの内容が機能に追加されます。 また、ワイルドカードを使用して、プロジェクトに追加するファイルをフィルタリングできます。詳細については、「ダイナミック リンクの作成」を参照してください。 ***メモ:*** *サブフォルダーとダイナミックリンクされたファイルが、事前に設定した数を超過した場合、表示パフォーマンスの向上のため、[ファイルとフォルダー] ビューのリストは切り詰めて表示されます。この状態のファイルリストでは、先頭項目に **リストが切り捨てられました** と表示されます。サブフォルダーに含まれるファイルはすべて、プロジェクトにビルドされます。*
依存関係 スキャン	ファイルをプロジェクトに追加する最後の方法は、[依存関係スキャナー] ビューを使用することです。このビューには、プロジェクト、実行アプリケーション、または Visual Basic プロジェクトで依存関係の可能性があるファイルをスキャンしてプロジェクトに追加する 3 つのウィザードがあります。これらの 3 つのウィザードは、すべて [プロジェクト] メニューから起動できます。

インストール先フォルダー

プロジェクト: *フォルダーによって、すべてのプロジェクトの種類で使用できない場合があります。プロジェクト固有の違いについては、必要に応じて記述されています。*

インストール プロジェクトにファイルを追加する場合、インストール先フォルダーにファイルを配置してこれを行います。デフォルトでは、以下のインストール先フォルダーが提供されます。それぞれはダイナミックです。つまり、ハードコード化されたパスは必要ありません。その代わり、各インストール先のフォルダーの値は、ターゲット マシンのオペレーティング システムから取得されます。

基本の MSI プロジェクトと InstallScript MSI プロジェクトのフォルダー

次のフォルダーは、基本の MSI と InstallScript MSI プロジェクトでのみ使用できます。

テーブル 113 • 基本の MSI と InstallScript MSI プロジェクトのデフォルトのインストール先フォルダー

フォルダー名	説明
AdminToolsFolder	管理ツールのあるフォルダーを指します。

テーブル 113 • 基本の MSI と InstallScript MSI プロジェクトのデフォルトのインストール先フォルダー (続き.)

フォルダー名	説明
ALLUSERSPROFILE	これは、すべてのユーザーのプログラム データを含むフォルダーへの完全パスです。共通パスは、次のとおりです。 C:¥ProgramData
AppDataFolder	このプロパティは、現在のユーザーのアプリケーション データ フォルダーへの完全パスを保持します。このプロパティの通常の値は、次の通りです: C:¥Users¥*UserName*¥AppData¥Roaming
CommonAppDataFolder	これは、すべてのユーザーのアプリケーション データを含むフォルダーへの完全パスです。共通パスは、次のとおりです。 C:¥ProgramData
CommonFiles64Folder	このプロパティの値は、64 ビット Common Files フォルダーへの完全修飾パスです。このプロパティには、Windows Installer バージョン 2.0 以降が必要です。
CommonFilesArmFolder	このプロパティの値 Arm¥Arm64 Common Files フォルダーへの完全修飾パスです。このプロパティには、Windows Installer バージョン 5.0 が必要で、テンプレート概要が Arm¥Arm64 をポイントしなくてはなりません。
CommonFilesFolder	このプロパティの値は、32 ビット Common Files フォルダーへの完全修飾パスです。
DesktopFolder	このプロパティは、現在のユーザーの [デスクトップ] フォルダーへの完全パスを保持するために使用します。セットアップが All Users に対して実行されているときに、ALLUSERS プロパティが設定されている場合、DesktopFolder プロパティには All Users デスクトップ フォルダーへの完全パスが保持されます。
FavoritesFolder	FavoritesFolder プロパティは現在のユーザーの[お気に入り] フォルダーまでの完全パスを含みます。
FontsFolder	このプロパティは、Fonts フォルダーへの完全パスを保持します。
IISROOTFOLDER	このプロパティは、ターゲット システムの Web サーバーのルート ディレクトリを格納します。インストール プロジェクトで IIS 機能を使用していて、Web サイトが既に追加されている場合 IISROOTFOLDER は自動的に追加されます。 詳細については、「IISROOTFOLDER サポートの追加」を参照してください。

テーブル 113 • 基本の MSI と InstallScript MSI プロジェクトのデフォルトのインストール先フォルダー (続き.)

フォルダー名	説明
INSTALLDIR	このプロパティには、インストールプログラムのファイルのデフォルトのインストール先フォルダーが格納されます。[一般情報] ビューで、INSTALLDIR の初期値を設定することができます。
LocalAppDataFolder	ローカルに保存されたアプリケーションデータの場所。このプロパティの通常の値は、次の通りです: C:¥Users¥*UserName*¥AppData¥Local
MyPicturesFolder	このプロパティは、現在のユーザーの [マイ ピクチャ] フォルダーへの完全パスを保持します。
PersonalFolder	このプロパティは、現在のユーザーの個人フォルダーへの完全パスを保持します。
ProgramFiles64Folder	このプロパティは、64 ビット Program Files フォルダーまでの完全パスを保持します。このプロパティには、Windows Installer バージョン 2.0 以降が必要です。
ProgramFilesArmFolder	このプロパティは Arm¥Arm64 Program Files フォルダーへの完全パスを保持します。このプロパティには、Windows Installer バージョン 5.0 が必要で、テンプレート概要が Arm¥Arm64 をポイントしなくてはなりません。
ProgramFilesFolder	このプロパティは、32 ビット Program Files フォルダーまでの完全パスを保持します。
ProgramMenuFolder	このプロパティは、現在のユーザーの [プログラム] メニューへの完全パスを保持するために使用されます。インストールが All Users に対して実行されるとき、ALLUSERS プロパティが設定されている場合、ProgramMenuFolder プロパティには All Users [プログラム] メニューへの完全パスが保持されます。
PUBLIC	このプロパティは、Users Public フォルダーへの完全パスを保持します。
SendToFolder	このプロパティは、現在のユーザーの SendTo フォルダーへの完全パスを保持します。
StartMenuFolder	このプロパティは、現在のユーザーの [スタート] メニューへの完全修飾パスを保持するために使用されます。インストールが All Users に対して実行されているときに、ALLUSERS プロパティが設定されている場合、StartMenuFolder プロパティには All Users [プログラム] メニューへの完全修飾パスが保持されます。

テーブル 113 • 基本の MSI と InstallScript MSI プロジェクトのデフォルトのインストール先フォルダー (続き.)

フォルダー名	説明
StartupFolder	このプロパティは、現在のユーザーの [スタートアップ] フォルダーへの完全修飾パスを保持するために使用されます。セットアップが All Users に対して実行されているときに、ALLUSERS プロパティが設定されている場合、StartupFolder プロパティには All Users [プログラム] メニューへの完全パスが保持されます。
System16Folder	このプロパティは、システムの 16 ビット .dll を含むフォルダーまでの完全パスを保持します。(Windows Installer バージョン 2.0 では、このプロパティは使用されていません。)
SystemArmFolder	このプロパティは Arm¥Arm64 Windows システム フォルダーへの完全パスを保持します。このプロパティには、Windows Installer バージョン 5.0 が必要で、テンプレート概要が Arm¥Arm64 をポイントしなくてはなりません。
SystemFolder	このプロパティは、32 ビット Windows System フォルダーまでの完全パスを保持します。
System64Folder	このプロパティは、64 ビット Windows System フォルダーまでの完全パスを保持します。このプロパティには、Windows Installer バージョン 2.0 以降が必要です。
TempFolder	このプロパティは、Temp フォルダーまでの完全パスを保持します。
TemplateFolder	このプロパティは、現在のユーザーの [テンプレート] フォルダーへの完全パスを保持します。
WindowsFolder	このプロパティは、Windows フォルダーへの完全パスを保持します。
WindowsVolume	このプロパティは、Windows フォルダーのボリュームを保持します。このプロパティは Windows がインストールされているドライブに設定されます。

InstallScript プロジェクトのフォルダー

次のフォルダーは、InstallScript プロジェクトでのみ使用できます。

テーブル 114 • InstallScript プロジェクトのデフォルトのインストール先フォルダー

フォルダー名	説明
スクリプト定義のフォルダー	このフォルダーでは、その場所をスクリプトで決定するフォルダーを作成できます。ファイルは直接このフォルダーに追加できませんが、サブフォルダーには追加できます。サブフォルダーを作成するには、フォルダーを右クリックして [新しいフォルダー] を選択します。フォルダー名を山かっこで囲って指定(例、<MYFOLDER>)します。スクリプトからフォルダーの場所を割り当てるには、FeatureSetTarget を呼び出します。
アプリケーションのターゲット フォルダー	このフォルダーの場所は、システム変数の TARGETDIR によって決定されます。
Program Files	このフォルダーの場所は、32 ビットの Program Files フォルダーです。
Program Files (64 ビット)	このフォルダーの場所は、64 ビットの Program Files フォルダーです。
Program Files¥Common Files	このフォルダーの場所は、32 ビットの Program Files フォルダーです。
Program Files (64 ビット)¥Common Files (64 ビット)	このフォルダーの場所は、64 ビットの Program Files フォルダーです。
Support フォルダー	このフォルダーの場所は、システム変数の SUPPORTDIR によって決定されます。
Windows	このフォルダーの場所は、Windows フォルダーです。
Windows¥Fonts フォルダー	このフォルダーの場所は、Windows Fonts フォルダーです。
Windows¥Windows System	このフォルダーの場所は、32-bit Windows System フォルダーです。
Windows¥Windows System (64 ビット)	このフォルダーの場所は、64-bit Windows System フォルダーです。

[再配布可能ファイル] ビュー

プロジェクト: *[再配布可能ファイル] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[前提条件] ビューと [オブジェクト] ビューを使って再配布可能ファイルを InstallScript プロジェクトに追加できます。

[再配布可能ファイル] ビューには、InstallShield に含まれるすべての InstallShield オブジェクトとサードパーティ マージ モジュールが含まれています。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトでは、このビューにはプロジェクトに追加可能な InstallShield 前提条件も含まれます。

InstallShield 前提条件 (基本の MSI プロジェクトおよび InstallScript MSI プロジェクト)

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

プロジェクトに InstallShield 前提条件を含めると、複数のインストールを連鎖することができるため、1 度に 1 つの実行シーケンスのみしか実行できない Windows Installer 制限をバイパスできます。Setup.exe セットアップランチャーは、連鎖を管理するブートストラップ アプリケーションとしての役割を果たします。

InstallShield は InstallShield 前提条件のベース セットを含みます。また、では InstallShieldInstallShield 前提条件エディターを利用してカスタム InstallShield 前提条件を定義したり、既存の InstallShield 前提条件の設定を編集したりできます。

InstallShield では、次の 2 つのタイプの InstallShield 前提条件がサポートされています:

- **セットアップ前提条件** – この種類の前提条件のインストールは、インストールの実行の前に実行されます。
- **機能前提条件** – この種類の前提条件は、1 つまたは複数の機能に関連付けられています。機能前提条件は、前提条件を含む機能がインストールされたときに、その前提条件がシステム上に既にインストールされていない場合にインストールされます。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

マージ モジュール

マージ モジュール (.msm ファイル) には、個別機能をインストールするために必要なロジックとファイルのすべてが含まれています。たとえば、多くのアプリケーションには Microsoft Visual Basic ランタイム .dll が必要です。コンポーネントにファイルを含めてインストール要件を調べる必要はなく、プロジェクトの機能に Visual Basic 仮想マシン マージ モジュールを添付するだけでこれを実行できます。

メモ: *[再配布可能ファイル] ビューに含まれているマージ モジュールの多くは Microsoft またはその他のサードパーティによるものです。InstallShieldでは、これらのモジュールを無料配布することによって、インストール プロジェクトの作成を支援します。ただし、サードパーティが作成したモジュールに存在する問題を InstallShield が修正したり直すことはできません。サード パーティが作成したモジュールに関する問題は、ベンダーへお問い合わせください。*

InstallShieldオブジェクト

オブジェクトには、マージ モジュール同様、個別の機能をインストールするために必要なロジックとファイルがすべて含まれています。InstallShield に含まれる DirectX オブジェクトなどのオブジェクトは、ウィザードを使ってカスタマイズする必要があります。オブジェクトをインストールに追加すると、すぐに該当するカスタマイズ ウィザードが開きます。オブジェクトを追加時点でカスタマイズすることも、ウィザードをキャンセルした後でオブジェクトを右クリックして [オブジェクトの設定変更] を選択することによってカスタマイズすることもできます。

ライブ再配布可能ファイルギャラリー

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

構成可能マージ モジュール

構成可能な再配布可能ファイルは、ModuleSubstitution テーブル内の少なくとも 1 行によって参照される ModuleConfiguration テーブル内の少なくとも 1 行を含むマージ モジュールまたはオブジェクトです。これによって再配布可能ファイルの値を変更することができます。[再配布可能ファイル] ビューで構成可能モジュールを選択した場合、表示される [マージ モジュール構成可能値] ダイアログ ボックスでモジュール追加時にそれを構成することができます。マージ モジュールを後でカスタマイズするには、それを右クリックして **[マージ モジュールの構成]**を選択します。

[再配布可能ファイル] ビューを使って作業する

[再配布可能ファイル] ビューは、次の要素で構成されます:

- ボタン行とその他のコントロール
- グループ ボックス領域 (ボタン行の下)
- 再配布可能ファイルの一覧
- 選択された再配布可能ファイルついての情報を表示する、詳細ペイン

次の表では、[再配布可能ファイル] ビューに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 115 ・ [再配布可能ファイル] ビューのコントロール

コントロールの名前	アイコン	説明
すべてのグループを展開する		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を表示します。
すべてのグループを折りたたむ		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を隠します。

テーブル 115 • [再配布可能ファイル] ビューのコントロール (続き.)

コントロールの名前	アイコン	説明
詳細の表示		提供されている再配布可能ファイルの一覧から選択された InstallShield 前提条件、マージ モジュール、またはオブジェクトの詳細を表示するビュー内の右側にあるペインの表示/非表示を切り替えます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。
グループ ボックスの表示		このビューのボタン行の下にある [グループ ボックス] 領域の表示/非表示を切り替えます。
リフレッシュ		再配布可能ファイルのリストを更新します。
検索グリッド		この検索ボックスで指定した文字列に従って、[再配布可能ファイル] ビューに表示される再配布可能ファイルをダイナミックにフィルターします。このボックスに文字列を入力すると、それを含まない行すべてが非表示となります。
[再配布可能ファイル] ビュー ヘルプ		[再配布可能ファイル] ビューのヘルプを表示します。
グループ化する列のヘッダーをここにドラッグ		このグループ ボックス領域を使って、ビュー内の行をグループ分けします。ビューでは、列のヘッダーをグループ ボックスにドラッグ アンド ドロップするだけで複数階層にグループ化することができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。 詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

次の表は、[再配布可能ファイル] ビューの各列について説明します。

テーブル 116 • [再配布可能ファイル] ビューの列

列	説明
チェック ボックス	この列には、各再配布可能ファイルのチェック ボックスが表示されます。再配布可能ファイルをプロジェクトに追加するには、そのチェック ボックスを選択します。
タイプ/状態	この列には、再配布可能ファイルの種類を示すアイコンと、再配布可能ファイルの状態が表示されます。各アイコンの詳細については、「再配布可能ファイル ギャラリーを管理する」を参照してください。
名前	この列には、各再配布可能ファイルの名前が表示されます。
バージョン	この列には、再配布可能ファイルのバージョン番号が表示されます。

テーブル 116 • [再配布可能ファイル] ビューの列 (続き.)

列	説明
Type	この列には、再配布可能ファイルの種類がリストされます。
場所	この列には、再配布可能ファイルが使用中のマシン上でローカルにインストール済みか、ダウンロードが必要であるかが示されます。詳細については、「再配布可能ファイルをコンピューターにダウンロードする」を参照してください。

[前提条件] ビュー

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[再配布可能ファイル] ビューを使って、InstallShield 前提条件を基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができます。

[前提条件] ビューには、InstallShield に含まれている InstallShield 前提条件のすべてが表示されます。InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

InstallShield は InstallShield 前提条件のベース セットを含みます。また、では InstallShieldInstallShield 前提条件エディターを利用してカスタム InstallShield 前提条件を定義したり、既存の InstallShield 前提条件の設定を編集したりできます。

InstallShield では、次の 2 つのタイプの InstallShield 前提条件がサポートされています:

- **セットアップ前提条件** – この種類の前提条件のインストールは、インストールの実行の前に実行されます。
- **機能前提条件** – この種類の前提条件は、1 つまたは複数の機能に関連付けられています。機能前提条件は、前提条件を含む機能がインストールされたときに、その前提条件がシステム上に既にインストールされていない場合にインストールされます。したがって、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

プロジェクト: *基本の MSI プロジェクトが、機能前提条件のサポートを含みます。*

ライブ再配布可能ファイルギャラリー

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

[前提条件] ビューを使って作業する

[前提条件] ビューは、次の要素で構成されます:

- ボタン行とその他のコントロール
- グループ ボックス領域 (ボタン行の下)
- 再配布可能ファイルの一覧
- 選択された再配布可能ファイルついての情報を表示する、詳細ペイン

次の表では、[前提条件] ビューに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 117 ・ [前提条件] ビューのコントロール

コントロールの名前	アイコン	説明
すべてのグループを展開する		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を表示します。
すべてのグループを折りたたむ		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を隠します。
詳細の表示		提供されている再配布可能ファイルの一覧から選択された InstallShield 前提条件の詳細を表示するビュー内の右側にあるペインの表示/非表示を切り替えます。この詳細ペインで、再配布可能ファイルがインストールするファイルなどの情報を確認することができます。
グループ ボックスの表示		このビューのボタン行の下にある [グループ ボックス] 領域の表示/非表示を切り替えます。
リフレッシュ		再配布可能ファイルのリストを更新します。
検索グリッド		この検索ボックスで指定した文字列に従って、[前提条件] ビューに表示される再配布可能ファイルをダイナミックにフィルターします。このボックスに文字列を入力すると、それを含まない行すべてが非表示となります。
[前提条件] ビュー ヘルプ		[前提条件] ビューのヘルプを表示します。
グループ化する列のヘッダーをここにドラッグ		このグループ ボックス領域を使って、ビュー内の行をグループ分けします。ビューでは、列のヘッダーをグループ ボックスにドラッグ アンド ドロップするだけで複数階層にグループ化することができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。 詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

次の表は、[前提条件] ビューの各列について説明します。

テーブル 118 • [前提条件] ビューの列

列	説明
チェック ボックス	この列には、各再配布可能ファイルのチェック ボックスが表示されます。再配布可能ファイルをプロジェクトに追加するには、そのチェック ボックスを選択します。
タイプ/状態	この列には、再配布可能ファイルの種類を示すアイコンと、再配布可能ファイルの状態が表示されます。各アイコンの詳細については、「再配布可能ファイル ギャラリーを管理する」を参照してください。
名前	この列には、各再配布可能ファイルの名前が表示されます。
バージョン	この列には、再配布可能ファイルのバージョン番号が表示されます。
場所	この列には、再配布可能ファイルが使用中のマシン上でローカルにインストール済みか、ダウンロードが必要であるかが示されます。詳細については、「再配布可能ファイルをコンピューターにダウンロードする」を参照してください。

[オブジェクト] ビュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

[再配布可能ファイル] ビューを使って、マージ モジュールを基本の MSI プロジェクトおよび InstallScript MSI プロジェクトに追加することができます。

[オブジェクト] ビューでは、InstallScript オブジェクトやサードパーティのマージ モジュールをプロジェクトに追加することができます。

マージ モジュール

マージ モジュール (.msm ファイル) には、個別機能をインストールするために必要なロジックとファイルのすべてが含まれています。たとえば、多くのアプリケーションには Microsoft Visual Basic ランタイム .dll が必要です。コンポーネントにファイルを含めてインストール要件を調べる必要はなく、プロジェクトの機能に Visual Basic 仮想マシン マージ モジュールを添付するだけでこれを実行できます。

メモ: *[オブジェクト] ビューに含まれているマージ モジュールの多くは Microsoft またはその他のサードパーティによるものです。InstallShieldでは、これらのモジュールを無料配布することによって、インストール プロ*

ジェクトの作成を支援します。ただし、サードパーティが作成したモジュールに存在する問題を InstallShield が修正したり直すことはできません。サード パーティが作成したモジュールに関する問題は、ベンダーへお問い合わせください。

InstallScript オブジェクト

InstallShield では、独自の InstallScript オブジェクトを作成して、他のインストール プロジェクトで使用したり、他のインストール開発者が使用できるように配布したりすることができます。

ライブ再配布可能ファイルギャラリー

多くの再配布可能ファイルはサイズが大きいため、プロジェクトで利用可能なものでも InstallShield のインストールと同時にコンピューターへ追加されない場合があります。その場合も、これらの再配布可能ファイルはインターネットからコンピューターへダウンロードすることができます。

[システム検索] ビュー

プロジェクト: *[システム構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

インストールでは、ターゲット システムの変更が必ず発生します。簡単なインストールでは、ファイルをコピーするだけのものもあります。より複雑なインストールではレジストリの変更、.ini ファイルの編集、ショートカットの作成などが行われます。InstallShield には、より詳細な構成を行うための次のビューがあります。

ショートカット

プロジェクト: *[ショートカット] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ショートカットを使用すると、インストールしたアプリケーションにすばやくアクセスできます。ショートカットおよびプログラム フォルダーは、デスクトップ上や [スタート] メニュー、その他さまざまな場所に作成することができます。

レジストリ

プロジェクト: *[レジストリ] ビューは、次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *QuickPatch*

[レジストリ] ビューでは、レジストリ エントリを作成したり、既存のレジストリデータをセットアップにインポートすることができます。

ODBC リソース

プロジェクト: *[ODBC リソース] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ODBC リソース] ビューでは、ドライバー、トランスレーターおよびデータ ソースをアプリケーションの 1 つまたは複数の機能に追加できます。インストールされた ODBC リソースを選択するか、または新しいリソースをリストに追加し、それらを機能に関連付けた後、その属性をカスタマイズします。

INI ファイルの変更

プロジェクト: *[INI ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、.ini ファイルを操作するためのいくつかの初期化ファイル関数が含まれます。

システム .ini ファイルは、[INI ファイルの変更] ビューで編集できますが、これらのファイルを編集することはお薦めできません。このビューでは、セクションおよびキーワードの作成やターゲット システムの .ini ファイルを編集することができます。

環境変数

プロジェクト: *[環境変数] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。

ターゲット システムの環境変数を作成または変更したり、ターゲット システムから環境変数を削除する場合、[環境変数] ビューでその設定を定義付けることができます。

XML ファイルの変更

プロジェクト: *[XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

場合によって、web.config および machine.config などの標準構成ファイルおよび、製品に関連する設定を保存する XML ファイルを変更する必要があります。InstallShield の [XML ファイルの変更] ビューを使って、ターゲット マシン上で変更する XML ファイルの変更をすべて指定することができます。

テキスト ファイルの変更

プロジェクト: *[テキスト ファイルの変更] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

InstallScript 言語には、文字列変数とリテラルを検出するための文字列関数が含まれています。

[テキスト ファイルの変更] ビューを使って、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。

ヒント: *[テキスト ファイルの変更] ビューを使って、MSXML を使用せずに XML ファイルを変更できます。[XML ファイルの変更] ビューを使って XML ファイルを変更する場合は、ランタイムに MSXML が必要です。*

スケジュール タスク

プロジェクト: *[スケジュール タスク] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[スケジュール タスク] ビューでは、ターゲット システムで、実行時に Windows のタスク スケジューラーで作成される自動タスクを作成および構成できます。スケジュール タスクで起動するファイルは、インストールに含まれているファイルでも、ターゲット システムに既に存在するファイルでも可能です。

サービス

プロジェクト: *[サービス] ビューは、次のプロジェクトの種類で提供されています:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[サービス] ビューでは、インストールとアンインストール中の Windows サービスのインストール、構成、開始、停止、およびアンインストールを指定できます。

別の方法として、[コンポーネント] ビューおよび [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [サービス] 領域を使って Windows サービスをインストール、構成、開始、停止、およびアンインストールすることもできます。これらの領域のいずれかを使ってサービス情報を構成した場合、対応する領域も自動的に更新されます。

[ショートカット] ビュー

プロジェクト: *[ショートカット] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ショートカット] ビューは、ターゲット システム上のアプリケーションへのショートカットを作成するためのシンプルかつビジュアルな方法を提供します。このビューには、ショートカットとサブフォルダーを作成できる一組の定義済みのインストール先フォルダーを表示する [ショートカット] エクスプローラーがあります。InstallShield には標準で次のショートカットのインストール先があります。

テーブル 119 ・ ショートカットの標準インストール先

ショートカットのインストール先	プロジェクトの種類	説明
[プログラム] メニューと [スタートアップ]	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	[プログラム] と [スタートアップ] フォルダーは、[スタート メニュー] の中にあります。[プログラム メニュー] フォルダーは、製品のショートカットをインストールする業界標準の場所となっています。[スタートアップ] フォルダーには、Windows が起動するときに常に起動する必要があるアイテムだけを配置します。

テーブル 119 • ショートカットの標準インストール先 (続き.)

ショートカットのインストール先	プロジェクトの種類	説明
SendTo	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	[送る] フォルダーは、ユーザーがファイルを右クリックしたときに利用できます。このコンテキスト メニューには [送る] と呼ばれるコマンドが含まれています。このフォルダーにプログラムのショートカットを作成した場合、エンドユーザーは任意のファイルをクリックして製品に送ることができます。 たとえば、エンド ユーザーがメモ帳で HTML ページを開けるようにしたいとします。[ショートカット] ビューの [送る] フォルダーに、メモ帳へのショートカットを作成すると、エンド ユーザーは HTML ファイルを右クリックしたときに [送る] メニューからメモ帳をクリックすることができます。このページのソースファイルがメモ帳で開かれます。
デスクトップ	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	[デスクトップ] フォルダーには、エンド ユーザーのデスクトップのショートカットが含まれています。デスクトップ フォルダー上にショートカットを作成すると、エンドユーザーのデスクトップに製品のアイコンが表示されます。 ***メモ:*** *デスクトップはショートカットの配置場所としては一番わかりやすい場所ですが、ショートカットを多く配置しすぎるとエンド ユーザーのデスクトップが混雑する可能性があります。*

新しいショートカットを追加すると、このビューの [ショートカット] エクスプローラーに追加されたショートカットが新しいノードとして追加されます。新しいショートカット ノードは、次のいずれかの条件が True ではないかぎり、ショートカットの "アイコン ファイル" 設定で選択されたアイコン (実行時に、ショートカットがターゲット システムに追加されたときに使用されます) と同じアイコンと共に表示されます。

- プロジェクトの種類が InstallScript である。
- ショートカットが、ターゲット システム上に既に存在するファイルに割り当てられている。

上記の両条件の場合、アイコン ファイルは実行時まで認識されません。したがって、[ショートカット] エクスプローラーでは、ターゲット システム上で実行時に使用されるアイコンの代わりに、各ショートカットに次のアイコンが表示されます。

アイコン ファイルがアイコンを含まない場合、このアイコンが [ショートカット] エクスプローラーのショートカットに使用されます。

メモ: *動的にリンクされたファイルへのショートカットを作成することはできません。詳細については、「ダイナミック ファイル リンクの制限事項」を参照してください。*

ショートカットとフォルダー、およびタイルについて構成可能な各設定についての詳細は、以下を参照してください:

- ショートカットの設定
- フォルダーの設定
- [タイル構成] の設定

ショートカットの設定

プロジェクト: *[ショートカット] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ショートカット] ビューで、ショートカットの設定は次のメイン カテゴリで構成されています:

- 外観
- 動作
- 全般
- Windows アプリ パッケージ タイルのオーバーライド
- パッケージ サポート フレームワーク

エディション: *「Windowsアプリ タイルのオーバーライド」 設定は、Premier Edition を使って基本の MSI プロジェクトを編集する際に使用できます。InstallScript MSI、ダイレクトモード プロジェクト (Windows Installer データベース、トランスフォーム)、および Professional または Express Editionでは、**[リリース]** ビューまたは **[ショートカット]** ビューに Windows アプリ関連のオプションが表示されません。*

外観

ショートカットの外観の設定を使って、ショートカットの表示名やアイコンなどの詳細を指定します。

テーブル 120 • 外観の設定

設定	プロジェクトの種類	説明
表示名	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ターゲット システムで表示されるショートカットの名前を指定します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、この設定の値を入力しなくてはなりません。そうでない場合は、ビルド エラーが発生します。* *表示名は、Windows Installer の Filename データ型です。入力した表示名が 8.3 形式ではない場合、InstallShield はこの設定に ShortName|LongName 形式を使用します。たとえば、この設定の値として My Product Name と入力した場合、InstallShield は MyProd~1|My Product Name のように設定して、実行時に必要な場合、短い名前を利用できるようにします。*

テーブル 120 • 外観の設定 (続き.)

設定	プロジェクトの種類	説明
表示リソース	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	複数言語アプリケーション用のインストールを準備中で、ショートカットの表示名用 .mui ファイルと言語ニュートラルなポータブル実行可能ファイルを切り離す場合、以下の設定を使用します: • **表示リソース** – 複数言語ユーザー インターフェイス (MUI) マニフェストを含む DLL ファイルを参照するには、この設定で省略記号ボタン (...) をクリックします。 DLL ファイルを参照してから選択するか、"表示リソース DLL" 設定でパスとファイル名を手入力した場合、InstallShield はこの設定にパスとファイル名をリストします。"表示リソース インデックス" 設定で指定されたリソース インデックスもリストします。 この設定に値が含まれている場合、"表示名" 設定の値は無視されます。この設定を空白のままにしておくと、Windows Installer は "表示名" 設定の値を使用します。 • **表示リソース DLL** – MUI マニフェストを含む DLL ファイルのパスとファイル名を手動で指定する場合は、それを入力します。ハードコード化されたディレクトリ パスの代りに、Windows Installer ディレクトリ プロパティ (例、**[INSTALLDIR]MyResource.dll**) を含めることができます。 "表示リソース" 設定で省略記号ボタンをクリックして DLL ファイルを参照した場合、InstallShield は "表示リソース DLL" 設定で [#filekey] 形式を使って DLL ファイルを識別します。 • **表示リソース インデックス** – ショートカットの表示名インデックスを指定します。この値に、負の数値は指定できません。 ***メモ:*** *DLL を指定する場合、"表示リソース インデックス" 設定の値も入力しなくてはなりません。* これらの設定により、言語依存ファイルをすべて含む .mui ファイルから、言語ニュートラルなポータブル実行可能ファイルを分離することができるため、後で追加言語のリソースを追加する際、アプリケーションの再コンパイルや再リンクの必要がなくなります。 Windows Vista 以降および Windows Server 2008 以降は、表示リソースをサポートします。それ以前のシステムは、これを無視します。

テーブル 120 • 外観の設定 (続き.)

設定	プロジェクトの種類	説明
説明	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ショートカットの説明を入力します。入力するテキストは、エンド ユーザーがショートカットの上にマウス ポインターを置いたときにツールヒントとして表示されます。また、ショートカットの [プロパティ] ダイアログ ボックスの "説明" フィールドにも表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 120 • 外観の設定 (続き.)

設定	プロジェクトの種類	説明
説明リソース	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	複数言語アプリケーション用のインストールを準備中で、ショートカットの説明用 .mui ファイルと言語ニュートラルなポータブル実行可能ファイルを切り離す場合、以下の設定を使用します: • **説明リソース** – 複数言語ユーザー インターフェイス (MUI) マニフェストを含む DLL ファイルを参照するには、この設定で省略記号ボタン (...) をクリックします。 DLL ファイルを参照してから選択するか、"説明リソース DLL" 設定でパスとファイル名を手入力した場合、InstallShield はこの設定にパスとファイル名をリストします。"説明リソース インデックス" 設定で指定されたリソース インデックスもリストします。 この設定に値が含まれている場合、"説明" 設定の値は無視されます。この設定を空白のままにしておくと、Windows Installer は "説明" 設定の値を使用します。 • **説明リソース** – DLL MUI マニフェストを含む DLL ファイルのパスとファイル名を手動で指定する場合は、それを入力します。ハードコード化されたディレクトリ パスの代りに、Windows Installer ディレクトリ プロパティ (例、**[INSTALLDIR]MyResource.dll**) を含めることができます。 "説明リソース" 設定で省略記号ボタンをクリックして DLL ファイルを参照した場合、InstallShield は "説明リソース DLL" 設定で [#filekey] 形式を使って DLL ファイルを識別します。 • **説明リソース インデックス** – ショートカットの説明インデックスを指定します。この値に、負の数値は指定できません。 ***メモ:*** *DLL を指定する場合、"説明リソース インデックス" 設定の値も入力しなくてはなりません。* これらの設定により、言語依存ファイルをすべて含む .mui ファイルから、言語ニュートラルなポータブル実行可能ファイルを分離することができるため、後で追加言語のリソースを追加する際、アプリケーションの再コンパイルや再リンクの必要がなくなります。 Windows Vista 以降および Windows Server 2008 以降は、説明リソースをサポートします。それ以前のシステムは、これを無視します。

テーブル 120 • 外観の設定 (続き.)

設定	プロジェクトの種類	説明
アイコン	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ショートカットに表示するアイコンを指定するには、以下の設定を使用します: • **アイコン** – 作成するショートカットのアイコンを含むファイルを指定します。アイコン リソースを含む .ico ファイル、または実行可能ファイル (.dll または .exe) を指定する必要があります。ファイルを参照してから選択するか、"アイコン ファイル" 設定でパスとファイル名を手入力した場合、InstallShield はこのアイコン設定にアイコン パスとインデックスをリストします。 ファイルを指定する方法は、使用しているプロジェクトの種類によって異なります。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 .ico ファイル、またはアイコン リソースを含む .exe または .dll ファイルを参照するには、この設定で省略記号ボタン (...) をクリックします。* *InstallScript プロジェクトの場合 – ターゲット システムにある .ico ファイル、またはアイコン リソースを含む .exe または .dll ファイルの場所を参照するには、この設定で省略記号ボタン (...) をクリックします。* • **アイコン ファイル** – アイコンを含むファイルのパスとファイル名を手動で指定する場合は、それを入力します。 • **アイコン インデックス** – 指定したアイコン ファイルに 1 つ以上のアイコン リソースがある場合、この設定にインデックスを入力します。 負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば、0 はファイル内の最初のアイコン、1 は 2 番目のアイコン、2 は 3 番目のアイコンを参照します。 負の数字は特定のリソース ID を参照するために使用します。たとえばアイコン インデックス -12 は、リソース ID が 12 のアイコンを示します。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 – Windows Installer は、コンポーネントをアドバタイズする場合に個別のアイコン ファイルを必要とするため、アイコン インデックスが指定されない限り、InstallShield が .exe または .dll ファイルから最初のアイコンを抽出します。アイコン ファイルを指定しなかった場合、このコンポーネントのキーファイルが自動的にショートカットのアイコンとして使用されます。*

動作

ショートカットの "動作" 設定を使って、ターゲットやキーボード ショートカットなどの詳細を指定します。

テーブル 121 • 動作の設定

設定	プロジェクトの種類	説明
アドバタイズ	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	アドバタイズ ショートカットを作成するかどうかを指定します。 [はい] を選択した場合、エンド ユーザーがこのショートカットを含む製品または機能をアドバタイズすると、ショートカットが作成されますが、コンポーネントのファイルは、エンド ユーザーがショートカットを起動するまでインストールされません。ショートカットを初めて起動すると、Windows Installer はコンポーネントのファイルと他のデータをインストールし、その後、ショートカットによってターゲット ファイルが起動されます。その後、ショートカットを使用すると常に、通常のショートカットと同じように動作します。 ***メモ•*** *アドバタイズ ショートカットを作成するためには、ショートカットのコンポーネントにキーファイルが必要です。キーファイルは、ショートカットのターゲットになります。*
ターゲット	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	エンド ユーザーがショートカットを起動したときに起動される、ターゲット システム上にあるファイルへのパスを入力します。値をテ入力する代りに、省略記号ボタン (...) をクリックしてショートカット ターゲットを参照することもできます。 ***プロジェクト•*** *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合 – ハードコード化されたディレクトリ パスの変わりに、Windows Installer ディレクトリ プロパティ (たとえば、**[INSTALLDIR]File.exe**) を使います。ショートカットがアドバタイズ ショートカットの場合、この設定は読み取り専用になります。ショートカット コンポーネントのキーファイルは、自動的にショートカットのターゲットとして設定されます。ショートカットのコンポーネントがキー ファイルを持たない場合、ターゲット システム上でそのショートカットは適切に機能しません。* *InstallScript プロジェクトの場合 – ハードコード化されたディレクトリ パスではなく、システム変数 (例、**<TARGETDIR>¥File.exe**) を使います。*

テーブル 121 • 動作の設定 (続き.)

設定	プロジェクトの種類	説明
引数	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ショートカットのコマンドライン引数を入力します。引数が、ターゲット システム上でショートカットの [プロパティ] ダイアログ ボックスの "ターゲット" 値に追加されます。これらの引数は、他のコマンドライン引数と同様に機能します。たとえば、1 つのファイルを実行可能ファイルにリンクさせたり、またはコマンドライン引数を渡して実行可能ファイルをサイレントに実行するように設定できます。 ***メモ:*** *InstallShield では構文の確認が行われないので、構文が正しいことを確認してください。* ***ヒント:*** *選択されたファイル名の引数には* **%1** *を使用してください。たとえば、エンドユーザーが* C:¥File.ext *を右クリックして、このショートカットの引数が* –p %1 *と想定します。この場合、コマンドラインの引数は* –p C:¥File.ext *になります。場合によっては、スペースを含むファイル名を正しく処理するために、* ***"%1"*** *のように %1 引数を引用符で囲む必要があります。*
作業ディレクトリ	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ショートカット ターゲットの作業ディレクトリを入力するか、省略記号ボタン (...) をクリックしてディレクトリを選択または作成します。 指定したディレクトリは、ターゲット システム上でショートカットの [プロパティ] ダイアログ ボックスの ***"作業フォルダー"*** フィールドに表示されます。作業ディレクトリは、製品が使用する現在のディレクトリであるだけでなく、標準の [ファイルを開く] および [ファイルの保存] ダイアログ ボックスで表示されるデフォルトのディレクトリでもあります。 ***プロジェクト:*** *たとえば、基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトで、作業ディレクトリを* **Files** *という* INSTALLDIR *のサブディレクトリに設定する場合は、リストから* ***[INSTALLDIR]*** *を選択して、その末尾にサブディレクトリ名の* **Files** *を追加します。設定が完了すると、* ***[INSTALLDIR]Files*** *と表示されます。* *InstallScript プロジェクトでは、たとえば、作業ディレクトリを* **Files** *という TARGETDIR のサブディレクトリに設定する場合は、リストから* ***<TARGETDIR>*** *を選択して、その末尾に* **¥Files** *を追加します。完了すると、* ***<TARGETDIR>¥Files*** *となります。*

テーブル 121 • 動作の設定 (続き.)

設定	プロジェクトの種類	説明
ホット キー	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、ショートカットに割り当てられたキーボード ショートカットの 10 進数値を含みます。キーボード ショートカットを製品のショートカットに割り当てると、エンド ユーザーは適切なホット キーを押してショートカットを起動できます。 InstallShield を使って、キーボード ショートカットの 10 進数値を自動的に計算するには、この設定で省略記号ボタン (...) をクリックします。 詳細については、「ショートカットにアクセスできるキーボード ショートカットを指定する」を参照してください。 ***注意 • ターゲット システム上の既存のキーボード ショートカットと競合する可能性があるため、ショートカットのキーボード ショートカットを構成することは避けることをお推めします。***
実行	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	エンド ユーザーがショートカットを起動したときに、ターゲット ファイルが使用するウィンドウのスタイルを選択します。選択可能なオプションは以下のとおりです: • **標準ウィンドウ** – ファイルを標準サイズのウィンドウで起動します。 • **最大ウィンドウ** – プログラムを全画面表示で起動します。 • **最小ウィンドウ** – ファイルが最小ウィンドウで起動し、タスクバーだけに表示されます。
管理者として実行	基本の MSI、InstallScript、InstallScript MSI	”管理者として実行” へのショートカットの設定を選択します。 選択可能なオプションは以下のとおりです: • **はい**– [管理者として実行] へのショートカットの設定を有効化します。 • **いいえ**– [管理者として実行] へのショートカットの設定を行ないません。 ***メモ• ショートカットを [管理者として実行] を使って設定するためには、インストーラーを昇格モードで実行する必要があります。*** *アドバタイズ ショートカットおよびインターネット ショートカットの場合、[管理者として実行] は無効化されます。*

テーブル 121 • 動作の設定 (続き.)

設定	プロジェクトの種類	説明
ピン留めを禁止する	InstallScript	エンド ユーザーが製品をインストールした後、タスクバーおよび [スタート] メニューにショートカットをピン留めできないように防ぐがどうかを指定します。選択可能なオプションは以下のとおりです: • **はい**—ショートカットがインストールされるときに、タスクバーおよび [スタート] メニューにショートカットをピン留めするためのコンテキスト メニュー コマンドが非表示となります。 • **いいえ**—ショートカットがインストールされるときに、タスクバーおよび [スタート] メニューにショートカットをピン留めするためのコンテキスト メニュー コマンドが有効となり、エンド ユーザーがショートカットをピン留めすることができます。デフォルトでは、これが設定されています。 Windows Installer 7 から、この設定がサポートされています。以前のバージョンの Windows はこの設定を無視します。 インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを無効化したい場合があります。このピン留めを行わないようにショートカットを構成した場合、[スタート] メニューの最もよく使われている製品のリストに、ショートカットのターゲットを含められなくなります。 詳細については、「エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する」を参照してください。

テーブル 121 • 動作の設定 (続き.)

設定	プロジェクトの種類	説明
新規として強調表示しない	InstallScript	エンド ユーザーが製品をインストールした後に、[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐかどうかを指定できます。選択可能なオプションは以下のとおりです: • **はい**—インストールが [スタート] メニューにショートカットを追加するとき、そのショートカットが新しくインストールされた項目として強調表示されません。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。 • **いいえ**—インストールが [スタート] メニューにショートカットを追加するとき、そのショートカットが新しくインストールされた項目として強調表示されます。デフォルトでは、これが設定されています。 Windows Installer 7 から、この設定がサポートされています。以前のバージョンの Windows はこの設定を無視します。 インストールの一部であるツールまたは従属的な製品のショートカットで、この設定に [はい] を選択する場合があります。 詳細については、「[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ」を参照してください。
スタート画面への初回のピン止めを許可	InstallScript	Windows 8 ターゲット マシン上で、ショートカットをデフォルトでスタート画面にピン留めするかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい** – ショートカットが Windows 8 システムにインストールされるとき、スタート画面にピン留めされます。エンド ユーザーは、オプションでショートカットのピン留めを外すことができます。デフォルトでは、これが設定されています。 • **いいえ** – ショートカットが Windows 8 システムにインストールされるとき、スタート画面にピン留めされません。システム上のすべてのアプリケーションへのショートカットを含むアプリのリストに表示されます。 詳細については、「Windows 8 スタート画面への初回のショートカットのピン留めを抑制する」を参照してください。

全般

ショートカットの ″全般″ 設定を使って、ショートカットを含むコンポーネントなどの詳細を指定します。

テーブル 122 • 全般設定

設定	プロジェクトの種類	説明
キー名	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ショートカットのキー名を入力します。このキー名は内部でのみ利用され、エンドユーザーには表示されません。
内部名	InstallScript	ショートカットの内部名を入力します。内部名はエンド ユーザーには表示されません。
コンポーネント	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、ショートカットを含むコンポーネントを指定します。省略記号ボタン (...) をクリックして [コンポーネントの参照] ダイアログ ボックスを表示し、ショートカット用の新しいコンポーネントを作成するか、既存のコンポーネントを選択できます。 選択されたコンポーネントがインストールされると、ターゲットシステム上にショートカットが作成されます。 ***プロジェクト:*** *InstallScript プロジェクトでは、ショートカットに複数のコンポーネントを選択できます。*
機能	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	この設定は、ショートカット コンポーネントが関連付けられている機能をすべて示します。機能 ーコンポーネント間の関連付けがない場合、この設定は空白です。
アンインストール	InstallScript	製品がアンインストールされる時にショートカットを削除するかどうかを指定します。
既存のショートカットを置換	InstallScript	ショートカットがターゲット システム上の同じ場所にある同じ表示名を持つ既存のショートカットを上書きするかどうかを選択します。
インターネット ショートカット	InstallScript	ショートカットがインターネット ショートカットかどうかを指定します。デフォルト設定は [いいえ] です。[はい] を選択する場合、ショートカットが有効な URL でなくてはなりません。
VS .NET プロジェクト出力	InstallScript	ショートカットが Visual Studio プロジェクト出力をポイントするかどうかを指定します。

テーブル 122 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
Type	InstallScript	ショートカットの作成方法を指定します。選択可能なオプションは以下のとおりです: • **自動** – インストールは、*ALLUSERS* システム変数の値を基に、ショートカットを表示する場所を自動的に判断します。 *ALLUSERS* の値が 0 以外の場合、ターゲット システム上で全てのエンド ユーザーが使用できるように、[すべてのユーザー] プロファイルにショートカットが作成されます。*ALLUSERS* が FALSE の場合、現在のユーザーのプロファイルにショートカットが作成されます。 • **個人** – ショートカットが現在のユーザーのプロファイルに作成されます。 • **共通** – ショートカットが [すべてのユーザー] プロファイルに作成されます。
コメント	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	このショートカットに関するコメントを入力します。コメントは参考のためにプロジェクトファイルに保存され、エンド ユーザーへは表示されません。

テーブル 122 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
シェル プロパティ	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定を使って、実行時に Windows シェルによる設定が必要な 1 つ以上のショートカット プロパティを指定できます。 1 つ以上のプロパティを指定するには、この設定で [新しいシェル ショートカット プロパティの追加] ボタンをクリックします。 • **ピン留めをしない**—エンド ユーザーが製品をインストールした後、このショートカットをタスクバーまたは [開始] メニューにピン留めするためのコンテキスト メニュー コマンドは表示されません。また、このオプションを使うと、[開始] メニューの最もよく使う製品リストにショートカットのターゲットを含むことができなくなります。 このオプションを選択すると、"キー名" 設定およびサブ設定の追加行が追加され、これらを必要に応じて構成できます。 • **新規として強調表示しない**—エンド ユーザーが製品をインストールした後、ショートカットの [開始] メニュー エントリは、新規でインストールされたアイテムとして強調表示されません。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。インストールの一部であるツールまたは従属的な製品のショートカットのプロパティを設定したい場合があります。このオプションを選択すると、"キー名" 設定およびサブ設定の追加行が追加され、これらを必要に応じて構成できます。 • **Windows 8 スタート画面への初回のピン止めを制御する** — このオプションを選択した場合、InstallShield は Windows 8 ターゲット システム上で初回インストール中に、ショートカットをスタート画面にピン留めする Windows 8 の既定動作を抑制します。インストールは Windows 8 で使用可能になった Windows シェル プロパティを設定します。インストールの一部であるツールまたは従属的な製品のショートカットのピン留めを抑制したい場合があります。 • **カスタム プロパティ**—インストールの実行時に Windows シェルによる設定が必要なショートカット プロパティのカスタム プロパティおよび値を指定するには、このオプションを選択します。新しい "キー名" 設定と、追加のサブ設定が追加され、ここで適切なプロパティ名と値を構成できます。 必要に応じて、追加プロパティを指定します。 ***メモ:*** *Windows Installer 5 より、シェル プロパティの設定がサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* 詳細については、「ショートカットのシェル プロパティを設定する」を参照してください。

テーブル 122 • 全般設定 (続き.)

設定	プロジェクトの種類	説明
キー名	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、"シェル プロパティ" 設定を使ってショートカットに 1 つ以上のシェル プロパティを追加している場合に表示されます。 シェル ショートカット プロパティのキー名を入力します。このキー名は内部でのみ利用され、エンドユーザーには表示されません。 ***メモ:*** *Windows Installer 5 より、シェル プロパティの設定がサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* 詳細については、「ショートカットのシェル プロパティを設定する」を参照してください。
プロパティ	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、"シェル プロパティ" 設定を使ってショートカットに 1 つ以上のシェル プロパティを追加している場合に表示されます。 実行時に Windows シェルが設定するショートカット プロパティを入力します。 ***メモ:*** *Windows Installer 5 より、シェル プロパティの設定がサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* 詳細については、「ショートカットのシェル プロパティを設定する」を参照してください。
値	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	この設定は、"シェル プロパティ" 設定を使ってショートカットに 1 つ以上のシェル プロパティを追加している場合に表示されます。 シェル ショートカット プロパティの値を入力します。 ***メモ:*** *Windows Installer 5 より、シェル プロパティの設定がサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* 詳細については、「ショートカットのシェル プロパティを設定する」を参照してください。

Windows アプリ パッケージ タイルのオーバーライド

エディション: *「Windowsアプリ パッケージ タイルのオーバーライド" 設定は、Premier Edition を使って基本の MSI プロジェクトを編集する際に使用できます。*

「Windows アプリ パッケージ タイルのオーバーライド」 設定を使って、Windows アプリ パッケージに作成されるタイルを構成します。これらの設定についての説明は、次のテーブルを参照してください。詳細は、「Windows アプリ ロゴの考慮事項」サブセクションを参照してください。

テーブル 123 • [Windows アプリ パッケージ タイルのオーバーライド] の設定

設定	プロジェクトの種類	説明
Windows アプリ パッケージを含む	基本の MSI	このプロジェクトから作成されるすべての Windows アプリ パッケージにこのショートカットを含めるかどうかを指定します。 • **はい**—このプロジェクトから作成されるすべての Windows アプリ パッケージにこのショートカットを含めます。Windows アプリ パッケージがインストールする .exe をポイントする場合、ショートカットがタイルになります。 • **いいえ**—このショートカットは、このプロジェクトから作成されたすべての Windows アプリ パッケージから除外されます。
短い名前	基本の MSI	Windows アプリ パッケージで作成された普通サイズまたは大きいタイル上に短い文字列を指定します。何も指定されていない場合、ショートカットの **"表示名"** 設定に指定された値がコピーされます。
背景色	基本の MSI	Windows アプリ パッケージで作成されるタイルの背景色に使う色を選択します。いくつかの定義済みの色を提供するドロップダウン リストから、または省略記号ボタン (...) を使ってその他の RGB 16 進数形式の色を選択できます。何も指定しなかった場合、Windows が設定する透明色が既定のタイル背景色として使用されます。
小さいロゴ	基本の MSI	スタート メニューの [すべてのアプリ] リストの左側の列に表示される小さいロゴ アイコンとして使用する 44X44 イメージ ファイルを指定します。イメージ ファイルが無い場合、InstallShield によってターゲットのタイル構成またはショートカットのアイコンから小ロゴが生成されます。

テーブル 123 • [Windows アプリ パッケージ タイルのオーバーライド] の設定 (続き.)

設定	プロジェクトの種類	説明

普通サイズのロゴ	基本の MSI	ピン留めされたときに中サイズのタイルとして、または縮小された小さいタイルとして使用する 150X150 イメージ ファイルを指定します。中サイズのロゴ イメージが必要です。イメージ ファイルが無い場合、InstallShield によってターゲットのタイル構成またはショートカットのアイコンから中ロゴが生成されます。
		
		次のサブ設定が使用できます:
		
		• **普通サイズのロゴに名前を表示**— Windows アプリ パッケージで作成される普通サイズのタイル上にアプリケーションの名前を表示するかどうかを指定します。選択可能なオプションは以下のとおりです:
		
		  • **はい**—普通サイズのタイルのロゴ イメージの上に **"短い名前"** フィールドで指定された名前を表示します。**"短い名前"** フィールドに何も指定されていない場合、ショートカットの **"表示名"** 設定に指定された値が使用されます。
		  • **いいえ**— Windows アプリ パッケージで作成された普通サイズのタイル上に名前を表示しません。

大きいロゴ	基本の MSI	ピン留めおよびサイズの変更が行われた場合のみ、大サイズ タイルに使用する 310x310 イメージ ファイルを指定します。この設定はオプションです。これを指定すると、タイルを大サイズに設定することができます。
		
		次のサブ設定が使用できます:
		
		• **大きいサイズのロゴに名前を表示**— Windows アプリ パッケージで作成される大きいタイル上にアプリケーションの名前を表示するかどうかを指定します。選択可能なオプションは以下のとおりです:
		
		  • **はい**—大きいタイルのロゴ イメージの上に **"短い名前"** フィールドで指定された名前を表示します。**"短い名前"** フィールドに何も指定されていない場合、ショートカットの **"表示名"** 設定に指定された値が使用されます。
		  • **いいえ**— Windows アプリ パッケージで作成された大きいタイル上に名前を表示しません。

テーブル 123 • [Windows アプリ パッケージ タイルのオーバーライド] の設定 (続き.)

設定	プロジェクトの種類	説明

横長ロゴ	基本の MSI	ピン留めおよびサイズの変更が行われた場合のみ、横長タイルに使用する 310x150 イメージ ファイルを指定します。この設定はオプションです。これを指定すると、タイルを横長に設定することができます。
		
		次のサブ設定が使用できます:
		
		• **大きいサイズのロゴに名前を表示**— Windows アプリ パッケージで作成される大きいタイル上にアプリケーションの名前を表示するかどうかを指定します。選択可能なオプションは以下のとおりです:
		
		  ▪ **はい**—横長タイルのロゴ イメージの上に **"短い名前"** フィールドで指定された名前を表示します。**"短い名前"** フィールドに何も指定されていない場合、ショートカットの **"表示名"** 設定に指定された値が使用されます。
		  ▪ **いいえ**— Windows アプリ パッケージで作成された横長タイル上に名前を表示しません。

Windows アプリ ロゴの考慮事項

重要 • *このセクションで説明されている Windows アプリ関連のイメージ形式を活用するには、InstallShield を Windows 10 マシンまたは Windows 10 SDK が搭載されたマシン上にインストールする必要があります。ビルド マシンに Windows 10 SDK、特に MakePRI.exe がない場合でも、前述の種類のイメージが収集されますが、resources.pri ファイルは生成されません。MakePRI が作成する resources.pri ファイルがない場合、Windows はニュートラル イメージのみを使用します。ニュートラル イメージは名前に修飾子を持たず、倍率 100 サイズのイメージが作成されます。*

次のテーブルに、Windows アプリ ロゴに関して考慮すべき点が記載されています。

トピック	考慮

ロゴのサイズ	すべてのロゴは、決められたサイズでなくてはなりません。複数のコントラスト形式と倍率 (DPI ベースの拡大縮小) を 2 つの方法の組み合わせて指定できます。見つかったそれぞれの形式に対して、InstallShield は必要に応じて変換またはサイズ調整を行ったファイルを含みます。スキャン プロセスは、InstallShield のスイート高 DPI サポートで使われる処理と似ています。両方の実装は、実行時に Windows が使用するプロセス「タイル通知とトースト通知のグローバリゼーションとアクセシビリティ (Windows ランタイム アプリ)」に従います。
	• ソース イメージの隣に表示される関連イメージ ファイルがスキャンされます。たとえば、<images>¥Logo.png という名前のファイルを含めた場合、InstallShield が完全修飾名ではないファイルとしてそのファイル、およびその他の完全修飾バリエーションを探します。
	  ▪ <images>¥contrast-black¥scale-nnn¥Logo.png
	  ▪ <images>¥contrast-white¥Logo.scale-nnn.png
	  ▪ <images>¥scale-nnn¥Logo.png
	  ▪ <images>¥Logo.scale-nnn.png
	• オリジナル ソースが .exe または .dll のアイコン リソースだったときに、前述の名前規則に基づいてより適切な倍率形式が見つからなかった場合、InstallShield が任意サイズのソースとしてオリジナルを使用します。

倍率	倍率によって、異なるサイズは必要になります。たとえば、大きいロゴは 310x310 ですが、倍率 -150 は、465x465 (310 * 150 / 100 = 465) です。InstallShield は現在、倍率 100、倍率 125、倍率 150、倍率 200、および倍率 400 イメージを含みま、スキャンを行います。倍率-*nnn* コンポーネントを持たない実際のイメージが必ず必要な訳ではありませんが、これが存在すると InstallShield でファイルを簡単に参照することができます。
ソース イメージの種類	指定されたソース イメージが .png または .jpg ファイルではない場合、またはサイズに誤りがある場合、ビルド時に正しいサイズの .png に変換されます。 ***メモ:*** *複数の倍率でアイコン リソースを変換すると、異なるサイズのアイコン リソースが各倍率で使用される結果となります。たとえば、倍率 200 中サイズ ロゴは 300x300 で、256x256 のアイコンを使用します。一方、倍率 100 中サイズ ロゴは、これを使用しません。*
ロゴ イメージの推奨レイアウト	Microsoft による、ロゴ イメージ レイアウトについての推奨およびその他の情報は、「タイルとアイコン アセットのガイドライン」を参照してください。Microsoft は、ロゴの中心部の周りに一定のパディングを使うことを推奨しています。たとえば、中サイズ タイル (150x150) の場合、45 ピクセル (33%) のパディングを使用して、ロゴの中心部分を 60x60 (または 50x50) とします。InstallShield が、この推奨に従うまたは強制することはありません。

フォルダーの設定

プロジェクト: *[ショートカット] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ショートカット] ビューのフォルダーには、以下の設定を構成できます。

テーブル 124 ・ フォルダーの設定

設定	プロジェクトの種類	説明
キー名	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	フォルダーのキー名を入力します。キー名は、Directory テーブルの Directory 列に表示されるフォルダーに使用される内部名であり、エンド ユーザーには表示されません。
内部名	InstallScript	フォルダーの内部名を入力します。内部名はエンド ユーザーには表示されません。

テーブル 124 • フォルダーの設定 (続き.)

設定	プロジェクトの種類	説明
表示名	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	ターゲット システムで表示されるフォルダーの名前を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***プロジェクト:*** *基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトの場合、この設定の値を入力しなくてはなりません。そうでない場合は、ビルド エラーが発生します。* *表示名は、Windows Installer の DefaultDirデータ型です。入力した表示名が 8 文字を超える場合、InstallShield はこの設定で ShortName|LongName 形式を使用します。たとえば、この設定の値として My Product Name と入力した場合、InstallShield は MyProd~1|My Product Name のように設定して、実行時に必要な場合、短い名前を利用できるようにします。* *InstallScript プロジェクトと InstallScript MSI プロジェクトの場合 – フォルダーの表示名として、グループ名システム変数 <SHELL_OBJECT_FOLDER> を入力することができます。その後、InstallScript でシステム変数 SHELL_OBJECT_FOLDERを設定することにより、このフォルダーの表示名を実行時に定義することができます。InstallScript MSI インストールでこの機能を使用する場合、フォルダーの "キー名" 設定で指定する文字は、すべて大文字でなくてはなりません (例、NEWFOLDER1)。詳細については、「SHELL_OBJECT_FOLDER」を参照してください。*
説明	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	フォルダーの説明を入力します。

テーブル 124 • フォルダーの設定 (続き.)

設定	プロジェクトの種類	説明
共有	InstallScript	このファイルを他のインストールと共有するかどうかを指定します: • **はい** – インストールによって作成されなかったサブフォルダーまたはファイルがフォルダーに含まれている場合、製品がアンインストールされる時に、そのフォルダーはターゲットシステムから削除されません。 • **いいえ** – 製品がターゲット システムからアンインストールされる時に、このフォルダーが削除されます。

[タイル構成] の設定

プロジェクト: *[タイル構成] の設定は、次のプロジェクトの種類の [ショートカット] ビュー内で使用できます。*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

タイルの構成の設定は、次のメイン カテゴリに分かれています:

- 全般
- 外観

全般

タイル構成の "全般" 設定を使って、タイルに関連付けられている .exe ファイルのパスを確認します。

テーブル 125 • 全般設定

設定	プロジェクトの種類	説明
ターゲット	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	タイルに関連付けられている .exe ファイルのパス (たとえば、**[INSTALLDIR]File.exe**) が表示されます。

外観

タイル構成の ″外観″ 設定を使って、タイルの背景色などの詳細を指定します。

テーブル 126・外観の設定

設定	プロジェクトの種類	説明
テキスト色	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	中サイズのタイルでアプリケーション名を表示する際に使う前景テキスト色を選択します。選択可能なドロップダウン オプション: ・ **明色**—中サイズのタイルに表示するテキストの色を明色 (白いテキスト) に設定します。 ・ **暗色**—中サイズのタイルに表示するテキストの色を暗色 (黒いテキスト) に設定します。 ***メモ・** 中サイズのタイル (150x150) のみ、タイル上に名前を表示できるスペースがあります。また、**″テキスト色″** 設定を反映させるためには、**″中タイルに名前を表示する″** 設定で **[はい]** を選択する必要があります。*
背景色	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	タイルの背景色に使う色を指定します。この色は、RGB 16進文字列フォーマット #rrggbb、または既知の色でなくてはなりません。ドロップダウン オプションを使って既知の色を選択するか、省略記号ボタン (…) を使ってカスタム色を選択できます。

テーブル 126 • 外観の設定 (続き.)

設定	プロジェクトの種類	説明
小さいタイル イメージ	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	小さいサイズ (70x70) のタイル背景イメージへの相対パスを指定するか、省略記号ボタン (...) をクリックして .png または .jpg ファイルを参照します。 ***重要*** *• カスタム イメージを使用する場合は、以下の点を考慮してください:* • *カスタム イメージはプロジェクトに手動で追加する必要があり、**小さいタイル イメージ** (70x70) と **中サイズのタイル イメージ** (150x150) の両方にイメージが必要です。1 つだけしか指定しなかった場合、Windows の規定のスタイルがアプリケーション アイコンおよび背景がタイルに適用されます。* • *サイズが固定されたイメージ、ハイコントラストなイメージ、またはローカライズされたイメージを使用する場合、手動で resources.pri ファイルを作成して、アプリケーションの .exe ファイルおよびその .visualelementsmanifest.xml ファイルと同じディレクトリに配置しなくてはなりません。イメージの使い方についての詳細、または resources.pri ファイルの作成方法については、次の MSDN 記事を参照してください: デスクトップ アプリのスタート画面のタイルをカスタマイズする方法 (Windows ランタイム アプリ)* • *タイル イメージを参照するとき、resources.pri を作成済み、または作成予定の場合は、選択された特定イメージに resources.pri ファイルのマップ先となる修飾子を含むことができます (たとえば、ISLogo.**scale-80**.png)。この場合、**小さいタイル イメージ** または **中サイズのタイル イメージ** の値から修飾子を削除する必要があります。ただし、resources.pri が使用されていない場合、またはこの特定のファイルのマッピングを含まない場合は、修飾子がファイル名の一部となるため、残す必要があります。修飾子が検出されると、使用するパスとファイル名を選択できるダイアログが表示されます。*

テーブル 126 • 外観の設定 (続き.)

設定	プロジェクトの種類	説明
普通サイズのタイル イメージ	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	中サイズ (150x150) タイルの背景イメージへの相対パスを指定します。 ***重要*** *• カスタム イメージを使用する場合は、以下の点を考慮してください:* • *カスタム イメージはプロジェクトに手動で追加する必要があり、**小さいタイル イメージ** (70x70) と **中サイズのタイル イメージ** (150x150) の両方にイメージが必要です。1 つだけしか指定しなかった場合、Windows の規定のスタイルがアプリケーション アイコンおよび背景がタイルに適用されます。* • *サイズが固定されたイメージ、ハイコントラストなイメージ、またはローカライズされたイメージを使用する場合、手動で resources.pri ファイルを作成して、アプリケーションの .exe ファイルおよびその .visualelementsmanifest.xml ファイルと同じディレクトリに配置しなくてはなりません。イメージの使い方についての詳細、または resources.pri ファイルの作成方法については、次の MSDN 記事を参照してください: デスクトップ アプリのスタート画面のタイルをカスタマイズする方法 (Windows ランタイム アプリ)* • *タイル イメージを参照するとき、resources.pri を作成済み、または作成予定の場合は、選択された特定イメージに resources.pri ファイルのマップ先となる修飾子を含むことができます (たとえば、ISLogo.**scale-80**.png)。この場合、**小さいタイル イメージ** または **中サイズのタイル イメージ** の値から修飾子を削除する必要があります。ただし、resources.pri が使用されていない場合、またはこの特定のファイルのマッピングを含まない場合は、修飾子がファイル名の一部となるため、残す必要があります。修飾子が検出されると、使用するパスとファイル名を選択できるダイアログが表示されます。*
中タイルに名前を表示する	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	アプリケーション名を中サイズ (150x150) タイルに表示するかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい**—中サイズ タイルが使用された場合、タイル上にアプリケーションの名前を表示します。 • **いいえ**—タイル上にアプリケーション名を表示しません。 *この設定は、小サイズ タイルの使用時には効果がありません。*

[MSIX アプリケーション] ビュー

プロジェクト: *[MSIX アプリケーション] ビューは、MSIXのプロジェクト タイプで使用できます:*

[MSIX アプリケーション] エクスプローラーを使って、パッケージで提供される機能の一部またはすべてを構成する Windows アプリのプロパティを視覚的および統合的にデザインすることができます。

- アプリケーションの設定

アプリケーションの設定

プロジェクト: *[MSIX アプリケーション] ビューは、MSIXのプロジェクト タイプで使用できます:*

[アプリケーション] ビューで、アプリケーションの設定は次のメイン カテゴリで構成されています:

- 概観
- 動作
- タイルの更新
- パッケージ サポート フレームワーク

概観

MSIX アプリケーションの ″外観″ 設定を使って、アプリケーションの表示名、説明、ビジュアル資産、サポートされる回転、コンテンツ URI、宣言セットなどの詳細を指定します。

テーブル 127 ・ 外観の設定

設定	プロジェクトの種類	説明
表示名	MSIX	ターゲット システムで表示されるアプリケーションの名前を入力します。これは、1 文字から 256 文字までの文字列です。
説明	MSIX	Windows アプリの説明を入力します。これは、1 文字から 2048 文字までの文字列で、タブ、改行コードや改行文字などを含めることはできません。
ビジュアル資産	MSIX	アプリケーションと関連付けるビジュアル資産の名前を指定します。このアプリケーションのビジュアル要素は、[ビジュアル資産] ビューで選択されているビジュアル資産名から取得されます。

テーブル 127・外観の設定 (続き.)

設定	プロジェクトの種類	説明
サポートされている回転	MSIX	最適なユーザー エクスペリエンスのために推奨されるアプリの表示方向を説明します。タブレットなど画面が回転可能なデバイス上では、ここで指定されていない方向にアプリが再描画されることはありません。たとえば、アプリが Landscape および LandscapeFlipped 方向のみを指定した場合、デバイスを縦に向けても、アプリは回転しません。 ***重要・*** *回転不可能なデバイスの場合、アプリはそのデバイスのデフォルト方向で表示され、アプリの優先方向は無視されることがあります。ただし、回転ロックが有効化されているデバイス上では、アプリの優先方向が維持されます。* この属性には次の方向のひとつが可能です: ・ Portrait ・ Landscape ・ PortraitFlipped ・ LandscapeFlipped
コンテンツ URI	MSIX	アプリケーションと関連付けるコンテンツ URI の名前を指定します。このアプリケーションのコンテンツ URIは、[コンテンツ URI] ビューで選択されたコンテンツ URI セットから取得されます。
宣言セット	MSIX	このアプリケーションと関連付ける宣言セットの名前を指定します。このアプリケーションの宣言は、[宣言] ビューで定義された宣言セットから取得されます。

動作

MSIX アプリケーションの "動作" 設定を使って、キー名、ターゲット、およびエントリ ポイントなどの詳細を指定します。

テーブル 128 • 動作の設定

設定	プロジェクトの種類	説明
キー名	MSIX	パッケージに含まれるアプリケーションの一意の識別子。この値は時々、PRAID (package-relative app identifier) と呼ばれます。ID はパッケージ内で一意ですが、グローバル値ではありません。システム上に同じ ID を使用する別のパッケージが存在する可能性があります。1 つのパッケージ内で同じ ID を複数使用することはできません。 この文字列には、ピリオドで区切られた英数字フィールドが含まれています。各フィールドは、ASCII アルファベット文字で始める必要があります。 これらのフィールド値を使用することはできません: "CON"、"PRN"、"AUX"、"NUL"、"COM1"、"COM2"、"COM3"、"COM4"、"COM5"、"COM6"、"COM7"、"COM8"、"COM9"、"LPT1"、"LPT2"、"LPT3"、"LPT4"、"LPT5"、"LPT6"、"LPT7"、"LPT8"、および "LPT9" アプリケーションのキー名を入力します。
ターゲット	MSIX	アプリのデフォルト起動実行可能ファイル。ファイルがパッケージに含まれていなくてはなりません。 これは、".exe" で終わる 1 文字から 256 文字までの文字列で、次の文字は使用できません: <、>、"、|、?、または *拡張子のデフォルト実行可能ファイルを指定します。指定されなかった場合、アプリに定義されている実行可能ファイルが使用されます。値をテ入力する代りに、省略記号ボタン (...) をクリックしてアプリケーション ターゲットを参照することもできます。
エントリ ポイント	MSIX	"Office.Winword.Class" など、アクティベート可能なクラス ID。Windows アプリケーションのデフォルトは、'Windows.FullTrustApplication' for Windows applications です。

タイルの更新

アプリ タイトルがアップデート通知を受け取る方法を説明します。

テーブル 129 • タイルの更新の設定

設定	プロジェクトの種類	説明
繰り返し	MSIX	アップデート通知を繰り返す間隔。 この属性には次の値のひとつが可能です: • HalfHour • Hour • SixHours • TwelveHours • Daily
URI テンプレート	MSIX	タイル アップデート通知の URI テンプレート。

パッケージ サポート フレームワーク

パッケージ サポート フレームワークは、ソース コードにアクセスできない場合でも既存する win32 アプリケーションに修正を適用できるキットなので、MSIX コンテナ内でこれを実行することができます。

これを使って、アプリケーションにモダンな実行環境のベスト プラクティスを適用できます。パッケージ サポート フレームワークには、実行可能ファイル、ランタイム マネージャー DLL、およびランタイム修正プログラムのセットが含まれます。

InstallShield 2019 R2 以降では、ファイル リダイレクト fixup およびカスタム fixup などのランタイム修正を含むパッケージ サポート フレームワークを MSIX で使用できます。

パッケージにランタイム修正プログラムを適用する方法を説明します。

テーブル 130 • パッケージ サポート フレームワークの設定

設定	プロジェクトの種類	説明
fixup の種類	MSIX、基本の MSI	アプリケーションによってサポートされているパッケージ サポート フレームの種類を指定します。 fixup タイプを追加するにfixup、[**fixup タイプを追加**] ボタンをクリックします。 サポートされている fixup タイプ: • ファイル リダイレクト fixup • カスタム fixup

テーブル 130・パッケージ サポート フレームワークの設定

設定	プロジェクトの種類	説明
アーキテクチャ	MSIX、基本の MSI	パッケージ サポート フレームワークが修正を行うための、様々なアーキテクチャ オプションを指定します。この選択に基づいて、アプリケーションは実行ファイルをピックアップします。 ビルド時に使用するアーキテクチャ検証の種類を指定します。選択可能なオプションは以下のとおりです: ・ x64 ・ x86
ファイル リダイレクト fixup	MSIX、基本の MSI	ファイル リダイレクト fixup を使って、MSIX コンテナ内で実行するアプリケーションからアクセスできないディレクトリ内にデータを書き込みまたは読み取りしようとする場合にリダイレクトを行います。 たとえば、アプリケーションがその実行ファイルと同じディレクトリにあるログ ファイルに書き込む場合、ファイル リダイレクト fixup を使ってそのログ ファイルを別の場所 (たとえば、ローカル アプリ データ ストア) に作成することができます。 ロケーション パスの例: C:¥Program Files (x86)¥InstallShield¥2021¥Samples¥Package Support Framework¥File Redirection Fixup
実行可能ファイル	MSIX、基本の MSI	リダイレクト fixup パラメーターの実行可能ファイル名を入力します。デフォルトで、値としてプライマリ プロセス名が入力されます。 リダイレクト fixup パラメーターの実行可能ファイルに複数プロセスが含まれる場合、1 つの特定のプロセスがファイルのリダイレクトを実行します。このフィールドには、その実行可能ファイルの名前を入力してください。 ***重要・実行可能ファイル名は、大文字と小文字を区別します。*** ファイルのリダイレクトを実行する実行可能ファイルであることが確実な場合のみ、この実行可能ファイル名を入力してください。そうでない場合、Process Monitor アプリケーション (procman.exe) を実行して、その実行可能ファイル名を取得できます。 追加されたすべてのアプリケーションに対して共通のファイル リダイレクト プロパティを適用する場合、.* を指定することができます。

テーブル 130 • パッケージ サポート フレームワークの設定

設定	プロジェクトの種類	説明
ベース	MSIX、基本の MSI	リダイレクトの適用先となるパスを指定します。指定の既知のフォルダー一覧から選択された既知のフォルダー、またはパッケージのルート ディレクトリに相対的なパスを指定できます。パッケージのルートレベルに相対的な任意のディレクトリにリダイレクトする場合は、空白に残すこともできます。 MSIX および基本の MSI の場合: • リダイレクトが INSTALLDIR で発生する場合、[INSTALLDIR] を選択します。 • 未知のフォルダーで発生する場合、相対パス部分のみを提供すると、パッケージのルートレベルに対して相対的であるとみなされます。 • 作成されたショートカットが任意の既知のフォルダーの下にあり、リダイレクトが現在のディレクトリで発生する場合、実行可能ファイルが存在するベースとして 既知のフォルダーを選択します。 基本の MSI の場合: • ファイルがパッケージ ルート レベル フォルダーの下に追加された場合、フォルダー構造に VFS/AppVPackageDrive を追加します。例: VFS/AppVPackageDrive/[SampleFolderName]
パターン	MSIX、基本の MSI	リダイレクトを適用するファイル タイプのパターン。その値がタイプ文字列の値となる配列。各配列はベース + 相対パスと一致するパスの正規表現として解釈されます。 例: • **ログファイル タイプの場合 – .*¥.log** • **テキストファイル タイプの場合 – .*¥.txt**
カスタム fixup	MSIX、基本の MSI	カスタム リダイレクト fixup は、カスタム JSON 構成ファイルで提供されているカスタム fixup の詳細に基づいて、アプリケーションの互換性問題を解決します。 ローケーション パスの例: C:¥Program Files (x86)¥InstallShield¥2021¥Samples¥Package Support Framework¥Custom Fixup

テーブル 130 • パッケージ サポート フレームワークの設定

設定	プロジェクトの種類	説明
JSON パス	MSIX、基本の MSI	DLL の詳細を含み、アプリケーションに適用されるカスタム fixup についての詳細を含むカスタム JASON 構成ファイルを指定します。 MSIX の場合、カスタム fixup DLL をパッケージのルート レベルに追加、またはそれらを x86 または x64 アーキテクチャに基づいて次のディレクトリの下に配置します: 基本の MSI の場合、カスタム fixup DLL を、x86 または x64 アーキテクチャに基づいて、これらのディレクトリの下に配置します: • x64 fixup DLL no場所:<InstallShield INSTALLDIR><InstallShield_VERSION>¥redist¥Language Independent¥x64¥PSF¥Fixups • x86 fixup DLL の場所:<InstallShield INSTALLDIR><InstallShield_VERSION>¥redist¥Language Independent¥i386¥PSF¥Fixups

パッケージ サポート フレームワークについての詳細は、 を参照してください。

fixup の作成についての詳細は、 を参照してください。

[レジストリ] ビュー

プロジェクト: *[レジストリ] ビューは、次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*
- *MSIX*
- *MSIX 変更パッケージ*

[レジストリ] ビューでは、インストールが作成するレジストリキーおよびレジストリ値を定義することができます。このビューは 4 つの異なるペインで構成されます。上部 2 つのペインには開発システム上に含まれるレジストリデータが表示され、下部 2 つのペインではターゲット システム上に作成されるレジストリデータを設定します。

Windows Installer プロジェクト (基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム プロジェクト) の場合、ビューの上にある [ビューフィルター] リストを利用してレジストリ データを表示するコンポーネントを選択することができます。

すべてのレジストリ データ (InstallScript プロジェクトの <Default> レジストリ セットを除く) をコンポーネントに関連付ける必要があります。これを行うことで、インストール プロジェクトでは、コンポーネントの機能がインストールに選択されると、そのコンポーネントのレジストリ データがターゲット システムに設定されます。

InstallScript プロジェクトでは、レジストリキーおよびレジストリ値をレジストリセットと呼びます。作成するレジストリセットの数、または 1 つのレジストリセッ内のキーおよび値の数に制限はありません。

インストーラーは、[一般情報] ビューに指定した値に基づいて、特定のレジストリ エントリを自動的に作成します。これらの情報キーは、Windows ロゴガイドラインで必要となります。また、コンポーネントの詳細設定はすべて、ターゲット システム上でファイルの登録に使用されます。

[ODBC リソース] ビュー

プロジェクト: *[ODBC リソース] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

システム構成のより複雑な領域の 1 つに、ODBC ドライバー、データソース名 (DSN)、トランスレーターの設定があります。ODBC リソースは、すべての必須属性とともにシステムに正しく登録されている必要があります。また、ドライバーおよびトランスレーターの場合、インストール .dll ファイルなどの必要なファイルをインストールする必要があります。このプロセスは、ODBC リソース ビューを使って単純化できます。このビューでは、開発システムにインストールされているドライバー、データ ソース、およびトランスレーターを選択できます。

ODBC リソースの構成可能な各設定についての詳細は、「ODBC リソースの設定」を参照してください。

プロジェクト: *インストール プロジェクトの場合: [ODBC リソース] ビューは、ODBC 関連リソースのインストール専用です。コア ODBC ファイルをインストールするには、[再配布可能ファイル] ビューで MDAC 2.5 マージ モジュールを選択します。*

DIM およびマージ モジュール プロジェクトの場合: [ODBC リソース] ペインに項目を追加すると、このビュー [関連付けられた機能] ペインの [マージされた機能] チェック ボックスが選択され、項目を削除すると選択が解除されます。これは、プロジェクトには影響しません。

ODBC リソースの設定

プロジェクト: *[ODBC リソース] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ODBC リソース属性、および利用できる値のユニバーサル リストはありません。インストール用に指定するものがわからない場合は、ファイルのベンダーまたは作成者にお問い合わせください。以下に示すように、いくつかの属性は ODBC リソースの各種類で共通です。

メモ: *独自の ODBC 属性を追加するには、[ODBC リソース] ビューの最終行をクリックしてプロパティと値を指定します。*

ドライバー

以下の必須属性のほかに、ドライバーにはツリー上で表示する名前が必要です。この名前は、ドライバーの説明として登録されます。ドライバーの名前をローカライズすることはできません。つまり、この名前はプロジェクトの文字列エントリではないようです 。したがって、システムの言語にかかわらず同じ値が使用されます。

テーブル 131 ・ ODBC ドライバー属性

属性	予期される値
Driver Component	選択した ODBC ドライバーとその属性をインストールするコンポーネントの名前を入力します。
ドライバー DLL	開発システム上にある DLL ファイルで、ODBC ドライバーとして機能するファイルへのパスを入力するか、省略記号ボタン (...) をクリックして、ファイルを参照します。このファイルは、Driver Component 設定で指定されたコンポーネントに含まれます。異なる DLL を選択した場合に [コンポーネント] ビューから以前の DLL を削除できるよう、各コンポーネントにつき 1 つのDLL を含めることを推奨します。
Setup Component	ODBC ドライバーのセットアップ DLL をインストールするコンポーネントの名前を入力します。通常、同じ DLL 内にセットアップ コードがある場合、このコンポーネントはドライバーのコンポーネントと同です。
セットアップ DLL	開発システム上のセットアップ DLL へのパスを入力するか、省略記号ボタン (...) をクリックして、ファイルを参照します。このファイルは、Setup Component 設定で指定されたコンポーネントに含まれます。ドライバー ファイルもセットアップ DLL である場合、この設定を空白のまま残します。異なる DLL を選択した場合に [コンポーネント] ビューから以前の DLL を削除できるよう、各コンポーネントにつき 1 つのDLL を含めることを推奨します。

データ ソース

以下の必須属性のほかに、DSN にはツリー上で表示する名前が必要です。この名前は、DSN の説明として登録されます。DSN の名前をローカライズすることはできません。つまり、この名前はプロジェクトの文字列エントリの一覧には表示されません。したがって、システムの言語にかかわらず同じ値が使用されます。

テーブル 132 ・ 必須データソース属性

属性	予期される値
登録	システム データ ソースまたはユーザー データ ソースのどちらを使用するか指定します。 ・ **システム データ ソース** – データ ソースは、システムのすべてのユーザーが使用できます。 ・ **ユーザー データ ソース** – データ ソースは、現在のユーザーにだけ登録されます。

トランスレーター

以下の必須属性のほかに、トランスレーターにはツリー上で表示する名前が必要です。この名前は、トランスレーターの説明として登録されます。トランスレーターの名前をローカライズすることはできません。つまり、この名前はプロジェクトの文字列エントリの一覧には表示されません。したがって、システムの言語にかかわらず同じ値が使用されます。

テーブル 133 ・ ODBC トランスレーター属性

属性	予期される値
トランスレーター コンポーネント	トランスレーターとその属性をインストールするコンポーネントの名前を入力します。
トランスレーター DLL	開発システム上にある DLL ファイルで、トランスレーターとして機能するファイルへのパスを入力するか、省略記号ボタン (...) をクリックして、ファイルを参照します。このファイルは、Translator Component 設定で指定されたコンポーネントに含まれます。異なる DLL を選択した場合に [コンポーネント] ビューから以前の DLL を削除できるよう、各コンポーネントにつき 1 つのDLL を含めることを推奨します。
Setup Component	トランスレーターのセットアップ DLL をインストールするコンポーネントの名前を入力します。通常、同じ DLL 内にセットアップ コードがある場合、このコンポーネントはドライバーのコンポーネントと同です。
セットアップ DLL	開発システム上のセットアップ DLL へのパスを入力するか、省略記号ボタン (...) をクリックして、ファイルを参照します。このファイルは、Setup Component 設定で指定されたコンポーネントに含まれます。ドライバー ファイルもセットアップ DLL である場合、この設定を空白のまま残します。異なる DLL を選択した場合に [コンポーネント] ビューから以前の DLL を削除できるよう、各コンポーネントにつき 1 つのDLL を含めることを推奨します。

メモ・ *トランスレーターに属性を追加することはできません。*

[INI ファイルの変更] ビュー

プロジェクト: *[INI ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

このビューは InstallScript プロジェクトでは使用できませんが、InstallScript 言語には、.ini ファイルデータを変更するための .ini ファイル関数が含まれています。

[INI ファイル変更] ビューでは、ターゲット システム上で 初期化 (.ini) ファイルに行なう変更を指定することができます。この種のファイルは、アプリケーションの次回の使用までの間において、情報の保存と取得を行うデータベースの役割を果たします。Boot.ini や Wininit.ini などの .ini ファイルは、オペレーティング システムで使用されます。ターゲット システムにあるすべての .ini ファイルを編集することができますが、システム .ini ファイルの変更は推奨しません。

タスク

.ini ファイルの編集には 3 つの手順があります。

1. .ini ファイル リファレンスを作成します。
2. .ini ファイルにセクションを追加します。
3. .ini ファイルにキーワードを追加します。

.ini ファイル リファレンスを作成するには、プロジェクトに少なくとも 1 つのコンポーネントが含まれている必要があります。.ini ファイル リファレンスを作成したときにコンポーネントがなかった場合は、[新しいコンポーネントの作成] ダイアログ が表示され、ここでコンポーネントを作成することができます。

ヒント: *InstallShield では、プロジェクトに既存する .ini ファイルをインポートできます。.ini ファイルをインポートした後、必要に応じて [INI ファイルの変更] ビューを使って、ターゲット システムに加える変更を構成することができます。詳細については、「既存の .ini ファイルをインポートする」を参照してください。*

.ini ファイルの変更で構成可能な各設定の詳細については、次を参照してください:

- .ini ファイルの設定
- .ini ファイルのセクションの設定
- .ini ファイルのキーワードの設定

.ini ファイルの設定

***プロジェクト・** [INI ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

.ini ファイルをプロジェクトに追加するとき、次の設定を構成します:

テーブル 134 ・ .ini ファイルの設定

設定	説明
表示名	INIFile.ini のように、編集する .ini ファイルの名前を拡張子と共に入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ・** 表示名は、Windows Installer の Filename データ型です。入力した表示名が 8.3 形式ではない場合、InstallShield はこの設定に ShortName|LongName 形式を使用します。たとえば、この設定の値として My File Name.ini と入力した場合、InstallShield は MyFile~1.ini|My File Name.ini のように設定して、実行時に必要な場合、短い名前を利用できるようにします。*
コンポーネント	この .ini ファイルと関連付けるコンポーネントを、以下いずれかの方法で選択します: ・ 既にプロジェクトに含まれているコンポーネントの一覧から選択するには、矢印をクリックします。 ・ .ini ファイルの変更に使用するコンポーネントを参照するか、新しいコンポーネントを作成するには、省略記号ボタン (...) をクリックします。 実行時に選択されたコンポーネントがインストールされると、.ini ファイルが変更されます。しかし、.ini ファイルはコンポーネントがインストールされない限り変更されません。
ターゲット	ターゲット システムにある .ini ファイルの保存場所を指定してください。パスをハードコード化するのではなく、一覧から Windows Installer フォルダーのプロパティを選択します。 ***ヒント・** サブフォルダーとフォルダーのプロパティを円記号で区切らず、下位レベルのサブフォルダーを円記号で区切ります (例、[ProgramFilesFolder]MyCompany¥Subdirectory)。*

.ini ファイルのセクションの設定

プロジェクト: *[INI ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[INI ファイルの変更] ビューでセクションを選択すると、次の設定が表示されます。

テーブル 135 · .ini ファイルのセクションの設定

設定	説明
表示名	編集する .ini ファイルの名前を入力します。.ini ファイルのセクション名は角かっこで囲まれていますが、この設定でセクション名を入力するときには角かっこは不要です。例えば、.ini ファイルの [INISection] セクションを変更する場合、この設定には **INISection** と入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

.ini ファイルのキーワードの設定

プロジェクト: *[INI ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[INI ファイルの変更] ビューでキーワードを選択すると、いくつかの設定が表示されます。

テーブル 136 • .ini ファイル キーワードの設定

設定	説明
表示名	編集するキーワードの名前を、ターゲットの .ini ファイルに表示される通りに正確に入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
アクション	実行するアクションを選択します。有効なオプションは次のとおりです: • **古い値を置き換える** – 既存の値は、[データ値] 設定に入力された値で置換されます。値が存在しない場合は、インストーラーによって作成されます。 • **上書きしない** – キーワードが既に存在しない場合は、.ini ファイルにその値が追加されます。.ini ファイルにキーワードが存在する場合、キーワードは変更されません。 • **タグを追加** – キーワード値に別のタグを追加する場合、このオプションを選択します。タグは、カンマで区切られます。タグの追加先となるキーワードが存在しない場合、変更されません。 • **値全体を削除する** – キーワードおよびその値は両方とも .ini ファイルから削除されます。指定したキーワードが存在しない場合、キーワードは変更されません。このオプションを選択する場合は、"データ値" 設定を空白に残します。 • **タグを削除する** – キーワードの複数値はタグとして認識されます。タグはコンマで区切られています。キーワードの値からタグを外すには、このオプションを選択します。"データ値" 設定で、削除するタグを入力してください。
データ値	キーワードの値を入力します。値を追加または付加する場合、新しい値を入力します。タグを削除する場合、削除するタグを入力します。キーワードと値を削除する場合、この設定を空白のまま残します。 キーワードの値として Windows Installer のプロパティを使用できます。Windows Installer のプロパティを使用するには、プロパティを角かっこで囲みます (例、[INSTALLDIR])。

[環境変数] ビュー

プロジェクト: *[環境変数] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*

- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。

環境変数は、ターゲット システム上の複数のアプリケーションでアクセスできる変数です。[環境変数] ビューでは、環境変数の新規作成、既存の変数値の変更、変数の削除などを行うことができます。環境変数と関連付けられたコンポーネントがインストールされると、ターゲット システム上で環境変数の作成、変更、または削除が行われます。

環境変数の設定

プロジェクト: *[環境変数] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallScript 言語には、環境変数の現在の値を読み出すための GetEnvVar 関数が含まれており、環境変数の作成が可能です。

環境変数の設定を指定すると、ターゲット システム上にある既存の変数の変更方法を指定したり、新しい変数を作成したりできます。各環境変数プロパティの説明を以下に示します。

テーブル 137 • 環境変数の設定

設定	説明
コンポーネント	この環境変数と関連付けるコンポーネントを、以下いずれかの方法で選択します: • 既にプロジェクトに含まれているコンポーネントの一覧から選択するには、矢印をクリックします。 • 環境変数に使用するコンポーネントを参照するか、新しいコンポーネントを作成するには、省略記号ボタン (...) をクリックします。 環境変数と関連付けられたコンポーネントがインストールされると、ターゲット システム上で環境変数の作成、変更、または削除が行われます。
値	この環境変数のパスまたは値を入力します。**[INSTALLDIR]bin** などのように、定義済みフォルダーをこの値の一部に使用できます。 複数のパスを入力する場合は、パスをセミコロン (;) で区切ります。 ***メモ:*** *"インストール時" 設定に [削除する] を選択すると、"値" 設定に入力された値はすべてクリアされて、"値" 設定が読み取り専用となります。*

テーブル 137 • 環境変数の設定 (続き.)

設定	説明
インストール時	関連付けられたコンポーネントがターゲット システムにインストールされたときに発生する動作を指定します。選択可能なオプションは以下のとおりです: • **作成する** — 指定された環境変数がターゲット システムに既存しない場合、インストールはその環境変数を作成して値を設定します。 • **設定する** — "配置" 設定と組み合わせて使用して、既存の環境変数の値を設定します。このオプションを選択した場合で、ターゲット システムに環境変数が存在しないときは、インストールは環境変数を作成して、その値を設定します。ターゲット システムに環境変数が存在する場合、インストールはその値を設定します。 • **削除する** — インストールは、指定された環境変数をターゲット システムから削除します。 ***メモ:*** *"インストール時" 設定に [削除する] を選択すると、"値" 設定に入力された値はすべてクリアされて、"値" 設定が読み取り専用となります。*
Placement	実行するアクションを選択します。選択可能なオプションは以下のとおりです: • **後に追加する** — インストールは、"値" 設定に入力された新しい値を既存する値の終わりに追加します。 • **前に追加する** — インストールは、"値" 設定に入力された新しい値を既存する値の始まりに追加します。 • **置換する** — インストールは、指定された環境変数の値を "値" 設定に入力された新しい値で置換します。 ***メモ:*** *"インストール時" 設定で [作成] を選択し、かつ指定された環境変数がターゲット システムに存在した場合、"配置" 設定は新しい値が既存の環境変数値に追加された状態か、または既存の環境変数値を置き換えた状態かを指定します。ただし、この場合において、指定された環境変数がターゲット システムにない場合、その値は作成され、"配置" 設定は無視されます。*

テーブル 137 • 環境変数の設定 (続き.)

設定	説明
アンインストール時	環境変数が関連付けられたコンポーネントがアンインストールされたとき、その環境変数を更新するかどうかを示します。選択可能なオプションは以下のとおりです: • 削除する—"配置" 設定に [前に追加] または [後に追加] が選択されていて、アンインストール時にターゲット システム上の環境変数値に指定された値が含まれる場合、その値のみが既存する変数の値から削除されます。 "配置" 設定で [置換する] を選択した場合で、アンインストール時に条件 (ターゲット システム上の値が指定された値と一致する、またはターゲット システム上の値が空白である) のどちらかが True 評価されたとき、環境変数全体が削除されます。 その他すべての状況下では、環境変数とその値がターゲット システムにそのまま残ります。 • 削除しない—環境変数と (存在する場合) その値がターゲット システムにそのまま残ります。
Type	環境変数名がシステム変数または環境変数のどちらの種類であるかを示します。選択可能なオプションは以下のとおりです: • システム — インストールは、指定されたシステム環境変数を作成、変更または削除します。 • ユーザー — インストールは、エンド ユーザー環境から環境変数を作成、変更または削除します。指定された環境変数は、インストール時にログオンしているエンド ユーザー以外は使用できません。

[XML ファイルの変更] ビュー

プロジェクト: *[XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイル変更] ビューでは、実行時に変更する XML ファイルのノードとノード セットの参照を追加します。XML ファイルは、インストールの一部であっても、またターゲット システム上に既に存在するファイルであっても構いません。

このビューの 2 つ重要な部分は、[ビュー フィルター] (プロジェクトの機能とコンポーネントの一覧) と [XML ファイル] エクスプローラー (ツリー ビュー) です。

- ビュー フィルターを利用して、データを機能またはコンポーネント別にフィルターすることができます。

- [XML ファイル] エクスプローラーでは、実行時に作成、変更、または削除する XML ファイルおよび XML データへの参照を追加します。各ノードは、XML ファイルにある既存のノードまたはノード セットを選択するために、インストールが実行時に実行する XPath 式を示します。

重要 *- [XML ファイルの変更] ビューは、XML ファイルにあるすべてのノードについてノードを表示するようにはデザインされていません。パフォーマンス速度を向上させるために、[XML ファイルの変更] ビューでは、ベースの XML ファイルとは異なる設定のみが表示されます。*

- *変更する XML ファイルがインストールの一部である場合、[XML ファイル変更] ビューには、XML ファイルが実行時にインストールされた後に追加、変更、または削除されるノード、またはノード セットのみが表示されます。*
- *変更する XML ファイルがターゲット システムに既に存在するファイルの場合、[XML ファイル変更] ビューには、実行時に追加、変更、または削除されるノードのみが表示されます。*

したがって、ファイルをインポートするとき、実行時に変更する XML ファイル内のノードのみをインポートします。

[XML ファイルの変更] ビューでは、新しい要素が XML ファイルに表示されるときの順番を指定することはできませんので注意してください。従って、要素が特定の順番で表示されている必要がある製品の場合、実行時に変更するノードのみをインポートすると、潜在的な問題を回避することができます。

[XML ファイルの変更] ビューではまた、[XML ファイル] エクスプローラーで選択したものにしたがって、右のペインに異なるタブが表示されます。これらのタブにある各設定のリファレンス情報については、次の参照してください。

- XML ファイルの [全般] タブ
- XML ファイルの [詳細] タブ
- XML ファイルの [名前空間] タブ
- XML 要素の [全般] タブ
- XML 要素の [詳細] タブ

詳しい [XML ファイルの変更] ビューの使い方と XML ファイルの変更を構成する方法については、「XML ファイルの変更」をご覧ください。

XML ファイルの [全般] タブ

プロジェクト *- [XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューの [全般] タブには、XML ファイルのための次の設定があります。

テーブル 138 • [XML ファイルの変更] ビューにおけるファイルのための [全般] タブの設定

設定	説明
ファイル名	ファイル名は、エクスプローラーで表示されるときの XML ファイル名になります。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
XML ファイルのインストール先	ターゲットマシン上の XML ファイルの保存場所を指定します。[参照] をクリックして既存のディレクトリを検索するか、新規のディレクトリを追加または作成します。
XML ファイルが属する機能を選択	XML ファイルと関連付ける 1 つ以上の機能のチェック ボックスを選択またはクリアします。InstallShield は、機能が [XML ファイルの変更] ビューに追加されると、自動的にそれを XML ファイルに関連付けます。 ***プロジェクト•*** *DIM およびマージ モジュール プロジェクトの場合 このビューで XML ファイルを選択すると、このペインの [マージされた機能] チェック ボックスは選択された状態で無効となります。DIM またはマージ モジュールをインストール プロジェクトに追加する場合、その DIM またはマージ モジュールを含める機能を指定します。*

XML ファイルの [詳細] タブ

プロジェクト• *[XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューの [詳細] タブには、XML ファイルのための次の設定があります。

XML ファイルオプション

次のファイル設定が実行時に使用できます:

テーブル 139 • 実行時に使用できる XML ファイルの設定

設定	説明
XML ファイルが存在しない場合、常に作成	実行時に XML ファイルがターゲット システム上に存在しない場合、[XML ファイルの変更] ビューで 指定された内容に基づいて XML ファイルがひとつ作成されます。
アンインストール時に XML ファイルを削除	アンインストール時にターゲット システム上の XML ファイルを削除する場合、このチェック ボックスを選択します。
変更後に XML をフォーマットする	ファイルにランタイムの変更が行われた後に XML ファイルをフォーマットする場合は、このチェック ボックスを選択します。ファイルをフォーマットするとき、インストールはファイルにインデントを追加して、空白要素タグを開始タグと終了タグに置換します。 フォーマットを行うと web.config ファイルに問題を起こす可能性があるため、XML ファイルのこのチェック ボックスをクリアしなくてはならない場合もあります。
絶対 Xpath を使用する	XML ドキュメント内のノードのパスには、絶対パスまたは相対パスの両方が使用できます。絶対パスはルートから始まります。既存の親要素内にある子要素と同じ名前を持つ XML ファイルに新しい子要素を追加するときは、絶対パスを使用する必要があります。そのためには、**[絶対 XPath を使用する]** オプションを選択してください。 • **選択** – このオプションが選択されている場合、子要素が追加されるときに絶対 XPath が使用されます。 • **非選択** – このオプションが選択されていない場合、子要素が追加されるときに汎用 XPath が使用されます。デフォルトでは、**"絶対 Xpath を使用する"** オプションは選択されていません。
新しいファイルに使用するエンコード	新しい XML ファイルに使用するエンコードを指定します。 XML ファイルがターゲット マシンに既存する場合、またはそれがインストールの一部としてインストールされる場合、インストールはこの設定で指定されたエンコーディングではなく、XML ファイルで指定されたエンコーディングを使用します。 ここで指定するエンコーディング値は、[XML ファイルの変更] ビューを使って XML ファイル全体を作成した場合にのみ使用されます。

XPath クエリ

この読み取り専用グリッドは、XML ファイルの変更 ビューで構成した XML の変更に基づいてランタイムにターゲット システム上で実行される XPath 式を表示します。

ヒント: Xpath 式についての詳細は、World Wide Web Consortium (W3C) Web サイト、および MSDN ライブラリの記事「 XPath を使って XML ドキュメントにクエリを実行する際に知っておくこと、避けること 」を参照してください。

XML ファイルの [名前空間] タブ

プロジェクト: *[XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューの [名前空間] タブには、XML ファイルのための次の設定があるテーブルがあります。

テーブル 140 · XML 名前空間の設定

設定	説明
前に追加する	この名前空間に関連付けられている要素に追加するプレフィックスを指定します。
URI	名前空間のインターネット リソースを識別する URI (Uniform Resource Identifier) を指定します。URI には、URL または文字列を指定できます。 ***メモ:*** *MSXML パーサーは、名前空間に関する情報を検索するときに、この URI を使用しません。URI の唯一の目的は、XML ファイル内で名前空間に一意の名前を与えることです。*

XML 要素の [全般] タブ

プロジェクト: *[XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*

- *トランスフォーム*

[XML ファイルの変更] ビューの [全般] タブには、XML 要素のための次の設定があります。

要素名

要素名は、XML ファイルの宣言済み要素の種類を表します。

この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

Attributes

テーブル 141 ・ XML 要素の属性の設定

設定	説明
属性	実行時に作成、アップデート、または削除される XML 属性の名前を入力します。属性には、Windows Installer プロパティを追加できます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
値	属性の種類の値を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
演算子	次の実行時の処理から、いずれかを選択します。 ・ **作成** – 実行時に、属性を要素に追加します。 ・ **削除** – 実行時に、属性から要素を削除します。 ・ **追加** – 指定された属性値を既存値の終わりに付加します。
スケジュール	スケジュール オプション (インストール、アンインストール、または両方) から選択して、属性変更を適用するタイミングを指定します。

ヒント: *Windows Installer プロパティからのダイナミック値、または、テキスト置換を利用して XML ファイルに異なる値を格納することができます。*

XML 要素の [詳細] タブ

プロジェクト: *[XML ファイルの変更] ビューは次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[XML ファイルの変更] ビューの [詳細] タブには、XML 要素に関する次の設定があります:

テーブル 142 ・ XML 要素の設定

設定	説明
最初に一致した要素のみを更新する	このチェック ボックスを選択すると、ターゲット XML ファイルで最初に一致した要素のみが変更されます。 このチェック ボックスをクリアすると、ターゲット XML ファイルで一致したすべての要素が変更されます。
この要素が存在しない場合、常に作成する	このチェック ボックスを選択すると、この要素が存在しないとき、常にターゲット マシンで作成されます。 ターゲット ファイルに存在しない要素に対してこのチェック ボックスがクリアされていると、実行時にその子要素は作成されません。従って、たとえば //A/B/C という XML ファイルで、システム B が存在しない、または作成されるように設定されていない場合、C がターゲット システムで作成されることはありません。
アンインストール時に要素を削除する	このチェック ボックスを選択すると、XML ファイルの変更を含むコンポーネントがアンインストールされたとき、XML ファイルからこの要素が削除されます。
要素のコンテンツの設定	テキスト コンテンツを要素に追加する場合、このチェック ボックスを選択して、[コンテンツ] ボックスにテキストを入力します。 次の XML の例では、**feature** は、要素 **product** のコンテンツです。 <product>feature</product>
コンテンツ	要素をアップデートするときの文字列を入力します。文字数は 255 に制限されています。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

[テキスト ファイルの変更] ビュー

プロジェクト: *[テキスト ファイルの変更] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

このビューは InstallScript プロジェクトでは使用できませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための文字列関数が含まれます。

[テキスト ファイルの変更] ビューを使って、ターゲット システム上で実行時に変更を行うテキスト ファイル (たとえば、.txt、.htm、.xml、.config、.ini、および .sql) 内の検索/置換処理を構成できます。テキスト ファイルは、インストールの一部またはシステム上に既存するファイルのどちらでも構いません。

タスク ***テキスト ファイルの変更を構成するには、以下の手順が必要です:***

1. テキスト 変更セットを作成する
2. テキスト ファイルの変更を指定する。

メモ: *各テキスト変更セットは、必ずプロジェクト内のコンポーネントに関連付ける必要があります。このため、テキスト変更セットを作成する前に、プロジェクトに 1 つ以上のコンポーネントが必要です。テキスト変更リファレンスを作成するときにコンポーネントがなかった場合は、[新しいコンポーネントの作成] ダイアログ が表示され、ここでコンポーネントを作成することができます。*

ヒント: *[テキスト ファイルの変更] ビューを使って、MSXML を使用せずに XML ファイルを変更できます。[XML ファイルの変更] ビューを使って XML ファイルを変更する場合は、ランタイムに MSXML が必要です。*

テキスト ファイルの変更で構成可能な各設定の詳細については、次を参照してください:

- 変更セットの設定
- 変更の設定

変更セットの設定

プロジェクト: *[テキスト ファイルの変更] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*

- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

このビューは InstallScript プロジェクトでは使用できませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための文字列関数が含まれます。

[テキスト ファイルの変更] ビューの [テキスト ファイルの変更] エクスプローラーで変更セット項目を選択すると、以下の設定を構成できます:

テーブル 143 • 変更セットの設定

設定	説明
ターゲット フォルダー	ターゲット システム上でのテキスト ファイル (つまり、.txt、.htm、.xml、.config、.ini、または .sql などの非バイナリファイル) の場所を指定します。パスをハードコード化するのではなく、一覧から Windows Installer フォルダーのプロパティを選択します。 ***ヒント:*** *サブフォルダーとフォルダーのプロパティを円記号で区切らず、下位レベルのサブフォルダーを円記号で区切ります (例、* **[ProgramFilesFolder]MyCompany¥Subdirectory***)。*
含めるファイル	検索する 1 つ以上のテキスト ファイルを指定します。複数のファイルはセミコロン (;) で区切ります。 ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、特定のディレクトリですべての .xml および .config ファイルを検索するには、次のように入力します: ***.xml;*.config** ***ヒント:*** *Windows Installer パブリック プロパティを使って、検索に含める/除外するテキスト ファイルの名前を指定することができます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。*

テーブル 143 • 変更セットの設定 (続き.)

設定	説明
除外するファイル	検索から除外するファイル (複数指定可) を指定します。複数のファイルはセミコロン (;) で区切ります。 ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、特定のディレクトリですべての .htm および .html ファイルを除外するには、次のように入力します: ***.htm;.html** ***ヒント:*** *Windows Installer パブリック プロパティを使って、検索に含める/除外するテキスト ファイルの名前を指定することができます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。*
サブフォルダーを含む	インストールが "ターゲット フォルダー" 設定で指定された場所のサブフォルダーを検索するかどうかを指定します。
インストール時に実行	変更セットのコンポーネントがターゲット システムにインストールされる時に、検索/置換処理を行うかどうかを指定します。
アンインストール時に実行	変更セットのコンポーネントがターゲット システム上から削除される時に、検索/置換処理を行うかどうかを指定します。
コンポーネント	このテキスト ファイルと関連付けるコンポーネントを、以下いずれかの方法で選択します: • 既にプロジェクトに含まれているコンポーネントの一覧から選択するには、矢印をクリックします。 • テキスト ファイルの変更に使用するコンポーネントを参照するか、新しいコンポーネントを作成するには、省略記号ボタン (...) をクリックします。

変更の設定

プロジェクト: *[テキスト ファイルの変更] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

このビューは InstallScript プロジェクトでは使用できませんが、InstallScript 言語には、文字列変数とリテラルを検索および変更するための文字列関数が含まれます。

[テキスト ファイルの変更] ビューの [テキスト ファイルの変更] エクスプローラーで変更項目を選択すると、以下の設定を構成できます:

テーブル 144 • 変更の設定

設定	説明
アクション	ターゲット システム上の該当するテキスト ファイル (複数可) に行う変更の種類を指定します。選択可能なオプションは以下のとおりです: • **置換** — 既存テキストを検索して、それを置換します。 • **ファイルの先頭に挿入** — テキストをファイルの最初に追加します。 • **ファイルの終りに挿入** — テキストをファイルの最後に追加します。 • **テキストの前に挿入** — 既存テキストを検索して、その前にテキストを追加します。 • **テキストの後に挿入** — 既存テキストを検索して、その後にテキストを追加します。
検索する文字列	この設定は、次の種類のテキスト変更アクションで使用できます: • Replace • テキストの前に挿入 • テキストの後に挿入 変更する文字列を入力するか、リストから Windows Installer プロパティを選択します。このリストには、プロジェクトの [プロパティ マネージャー] ビューで使用可能なすべてのプロパティが含まれています。 ***ヒント:*** *Windows Installer パブリック プロパティを使って、検索文字列と置換文字列を指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。*

テーブル 144 • 変更の設定 (続き.)

設定	説明
置換後の文字列	この設定は、置換タイプのテキスト変更アクションで使用できます。 検出された既存文字列と置き換える新しい文字列を入力するか、リストから Windows Installer プロパティを選択します。このリストには、プロジェクトの [プロパティ マネージャー] ビューで使用可能なすべてのプロパティが含まれています。 ***ヒント:*** *Windows Installer パブリック プロパティを使って、検索文字列と置換文字列を指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。* ***ヒント:*** *このフィールドにエスケープシーケンス文字を入力して、改行またはタブを指定することができます。詳細については、「エスケープシーケンスの解析」を参照してください。*
テキストの挿入	この設定は、次の種類のテキスト変更アクションで使用できます: • ファイルの最初に挿入 • ファイルの最初に挿入 • テキストの前に挿入 • テキストの後に挿入 テキスト ファイル (複数可) に挿入する新しい文字列を入力します。 ***ヒント:*** *Windows Installer パブリック プロパティを使って、検索文字列と挿入される文字列を指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時に製品のテキスト ファイルが変更される時に使用できるようになります。詳細については、「Windows Installer のプロパティを使用して、テキスト ファイルを動的に変更する」を参照してください。*

テーブル 144 • 変更の設定 (続き.)

設定	説明
単語単位で探す	この設定は、次の種類のテキスト変更アクションで使用できます: • Replace • テキストの前に挿入 • テキストの後に挿入 "検索する文字列" 設定に入力した値が単語単位で一致するインスタンスのみを検索するかどうかを指定します。たとえば、[はい] を指定した場合で、"検索文字列" 設定に **install** を入力すると、インストールは *install* のインスタンスのみを検索し、同じ単語の別の形式、たとえば *installs*、*installation*、または *uninstall* は検索しません。
大文字と小文字を区別する	この設定は、次の種類のテキスト変更アクションで使用できます: • Replace • テキストの前に挿入 • テキストの後に挿入 "検索文字列" 設定で使用している大文字と子文字の区別をそのまま利用して文字列を検索するように制限するかどうかを指定します。 たとえば、[はい] を指定した場合で、"検索文字列" 設定に **install** を入力すると、インストールはその文字列のすべて小文字を使ったインスタンスのみを検索します。[いいえ] を選択した場合、インストールは *install* だけでなく *INSTALL*、*Install*、その他の大文字と小文字を組み合わせたインスタンスも検索します。
一度だけ置換	この設定は、置換タイプのテキスト変更アクションで使用できます。 インストールが、最初に見つかった検索文字列のみを置換するかどうかを指定します。 [いいえ] を指定した場合、インストールは検索文字列のすべてのインスタンスを置換します。
1 回のみ挿入	この設定は、次の種類のテキスト変更アクションで使用できます: • テキストの前に挿入 • テキストの後に挿入 インストールが、最初に見つかった検索文字列のみにテキストを追加するかどうかを指定します。 [いいえ] を指定した場合、インストールは検索文字列のすべてのインスタンスに文字列を追加します。

テーブル 144 • 変更の設定 (続き.)

設定	説明
エスケープシーケンスの解析	インストーラーが**置換文字列**のエスケープシーケンス文字を解析する場合は **[はい]** を指定します。次のエスケープシーケンス文字がサポートされています: • 新しい行 (**¥r¥n**) • タブ (**¥t**)

[スケジュール タスク] ビュー

プロジェクト: *[スケジュール タスク] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[スケジュール タスク] ビューでは、ターゲット システムで、実行時に Windows のタスク スケジューラーで作成される自動タスクをプロジェクトに追加できます。作成したスケジュール タスクで、インストールの一部であるファイルや、ターゲット システムに既に存在するファイルを起動することができます。

スケジュール タスクに構成可能な各設定についての詳細は、「スケジュール タスクの設定」を参照してください:

メモ: *それぞれのスケジュール タスクは、必ずプロジェクト内のコンポーネントに関連付ける必要があります。したがって、スケジュール タスクを追加する前に、プロジェクトにコンポーネントが最低 1 つ存在している必要があります。スケジュール タスクを追加するときにコンポーネントが存在しない場合は、[新しいコンポーネントの作成] ダイアログ ボックスが表示され、ここでコンポーネントを作成することができます。*

スケジュール タスクの設定

プロジェクト: *[スケジュール タスク] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *トランスフォーム*

[スケジュール タスク] ビューで、スケジュール タスクの設定は、複数のメイン カテゴリに分かれています。

- 全般
- スケジュール

全般

スケジュール タスクの全般設定を使って、タスクの名前、タスクを含むコンポーネント、タスクで実行するファイルなどの詳細を指定できます。

テーブル 145 ・ スケジュール タスクの全般設定

設定	説明
タスク名	選択したスケジュール タスクに使用する名前を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
コンポーネント	次のいずれかの方法で、このスケジュール タスクを含めるコンポーネントを選択します: ・ 既にプロジェクトに存在するコンポーネントの一覧から選択するには、この設定の矢印をクリックします。 ・ スケジュール タスクに使用するコンポーネントを参照するか、新しいコンポーネントを作成するには、この設定の省略記号ボタン (...) をクリックします。 このコンポーネントがインストールの実行時にインストールされる場合、スケジュール タスクがターゲット システムに追加されます。
コメント	このタスクの Windows タスク スケジューラーで表示する説明を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定の隣にある省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
ターゲット	スケジュール タスクが実行されたときに起動されるターゲット システム上にあるファイルへのパスを入力します。ハードコード化されたディレクトリ パスではなく、Windows Installer ディレクトリ プロパティ (例、**[INSTALLDIR]File.exe**) を使います。値を手入力する代りに、省略記号ボタン (...) をクリックして、ターゲット場所を表示して、ファイル名を指定することもできます。
引数	ターゲット ファイル用のコマンドライン引数を入力します。

テーブル 145 • スケジュール タスクの全般設定 (続き.)

設定	説明
作業ディレクトリ	ターゲットの作業ディレクトリを入力するか、省略記号ボタン (...) をクリックしてディレクトリを選択または作成します。 たとえば、作業ディレクトリを **Files** という INSTALLDIR のサブディレクトリに設定する場合は、一覧から **[INSTALLDIR]** を選択して、その末尾に **Files** を追加します。設定が完了すると、**[INSTALLDIR]Files** と表示されます。
実行するアカウント名	スケジュール タスクの実行に使用するアカウントを入力します。例: **ドメイン名¥ユーザー名** このプロパティを [はい] に設定した場合、"パスワード" 設定にパスワードを指定する必要があります。そうでない場合、スケジュール タスクが実行時に作成されず、インストールが中止します。 ***ヒント:*** *この情報は、Windows Installer のパブリック プロパティを使って指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時にスケジュール タスクが構成された時に使用できるようになります。詳細については、「Windows Installer プロパティを使って、スケジュール タスクをダイナミックに構成する」を参照してください。*
パスワード	スケジュール タスクを実行するために、"実行するアカウント名" 設定で指定したユーザー アカウントと共に使用するパスワードを入力します。 "ログオン時のみ実行" 設定を [はい] に設定した場合、"パスワード" 設定にパスワードを指定する必要があります。そうでない場合、スケジュール タスクが実行時に作成されず、インストールが中止します。 ***ヒント:*** *この情報は、Windows Installer のパブリック プロパティを使って指定できます。これにより、エンド ユーザーがダイアログで入力したデータ、または実行時に判別された他の構成情報を、実行時にスケジュール タスクが構成された時に使用できるようになります。詳細については、「Windows Installer プロパティを使って、スケジュール タスクをダイナミックに構成する」を参照してください。*
ログオン時のみ実行	タスクがスケジュールされた時間、"実行するアカウント名" 設定で指定されたユーザー アカウントでシステムにログオンされている場合にのみタスクを実行するかどうかを指定します。 このプロパティを [はい] に設定した場合、"パスワード" 設定 ("別のユーザーとして実行" 設定の下) にパスワードを指定する必要があります。そうでない場合、スケジュール タスクが実行時に作成されず、インストールが中止します。

スケジュール

スケジュール タスクのスケジュール設定を使って、タスクを実行する頻度などの詳細を指定できます。一部の設定は、特定のスケジュール オプションを選択した場合にのみ表示されます。

テーブル 146 ・ スケジュール タスクのスケジュール設定

設定	説明
スケジュールの種類	スケジュール タスクを実行する頻度を示すオプションを選択します。
開始時間	タスクの実行の開始時間は、次のフォーマットで入力します: hh:mm hh は、深夜から経過した時間数を表し、mm は、1 時間ごとに経過した分数を表します。従って、指定できる時間は、00:00 から 23:59 までとなります。たとえば、開始時間が午後 2 時 45 分の場合、**14:45** と入力します。
開始日	タスクの実行開始日を入力します。日付フォーマットは、InstallShield を使用しているコンピューターの OS で設定されている形式を使用します。
間隔	指定された時刻と日付から開始して、スケジュール タスクを実行する頻度を指定します。
実行日	この設定は、"スケジュールの種類" 設定で選択したオプションに応じて異なります。 ・ **週単位スケジュール** – 週単位のスケジュール タスクを実行する曜日を指定するには、この設定の省略記号ボタン (...) をクリックして、適切な曜日のチェックボックスを選択します。 ・ **月単位スケジュール** – スケジュール タスクを特定の日に実行するか、または、特定の週に実行するかを指定します。 ・ **1 回のみ実行のスケジュール** – タスクを実行する日付を入力します。日付フォーマットは、InstallShield を使用しているコンピューターの OS で設定されている形式を使用します。
週	月単位のスケジュール タスクを実行する週を選択します。
日	月単位のスケジュール タスクを実行する日にちを指定するには、この設定の省略記号ボタン (...) をクリックして、適切な日にちのチェックボックスを選択します。
月	月単位のスケジュール タスクを実行する月を指定するには、この設定の省略記号ボタン (...) をクリックして、適切なチェックボックスを選択します。
アイドル時間	スケジュール タスクを実行する前にターゲット システムがアイドル状態である必要がある分数を指定します。

テーブル 146 ・ スケジュール タスクのスケジュール設定 (続き.)

設定	説明
繰り返し間隔 (分)	指定した時間の間に連続するタスクを実行する間隔を分数で指定します。この間隔は、前のスケジュール タスクの開始時点から計算されます。 たとえば、タスクが 8:00 AM から 5:00 PM まで 1 時間間隔で実行する場合、このフィールドには 60 と入力します。
時間間隔 (分)	アクティブなトリガでスケジュール タスクを繰り返す時間間隔を分数で指定します。ここに指定する間隔は、**"繰り返し間隔 (分)"** 設定で指定された間隔と等しいか、それよりも長く無くてなりません。 たとえば、8:00 AM から5:00 PM までスケジュール タスクを実行するには、540 と入力します。 ***メモ・** この設定は、**"繰り返し間隔 (分)"** 設定に値が指定されているときに有効化されます。*

[サービス] ビュー

***プロジェクト・** [サービス] ビュー (およびコンポーネントの [サービス] 領域) は、以下のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[サービス] ビューを使って、インストールまたはアンインストール中にサービスのインストール、開始、停止、または削除を行うコンポーネントを構成できます。また、このビューを使って、Windows Installer 5 で提供されている拡張サービスのカスタマイズ オプションを構成することもできます。[サービス] ビューの他にも、[コンポーネント] ビューまたは [セットアップのデザイン] ビューの [詳細設定] ノードの下にある [サービス] 領域を使うこともできます。これらの領域のいずれかを使ってサービス情報を構成した場合、対応する領域も自動的に更新されます。

サービスに関する作業について以下の点についてご注意ください:

- [サービス] ノードのサービスに入力する名前は、サービスの [プロパティ] ダイアログ ボックスに表示される名前と一致しなくてはなりません。(インストール済みサービスのプロパティにアクセスするには: [サービス] 管理ツールで、サービスを右クリックしてから [プロパティ] を選択します。)
- 開始、停止、削除、または構成を行うサービスは、インストールまたはアンインストール中にターゲット システムに既存するものか、インストールの一部としてインストールされるものかを問いません。

- サービス実行可能ファイルは、そのコンポーネントのキー ファイルでなくてはなりません。詳細については、「コンポーネントのキーファイル」を参照してください。
- サービスのコンポーネントの "リモート インストール" 設定は [ローカルを優先] でなくてはなりません。詳細については、「コンポーネントの "リモート インストール" 設定を構成する」を参照してください。

メモ: *Windows Installer はドライバー サービスをサポートしていないので、サービスは単一実行可能ファイル (.exe) である必要があります。*

サービスに構成可能な各設定についての詳細は、「[サービス] ビューの設定」を参照してください。

[サービス] ビューの設定

プロジェクト: *[サービス] ビュー (およびコンポーネントの [サービス] 領域) は、以下のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ヒント: *[サービス] ビューの上部にある [ビュー フィルター] を使って、ビューに表示するサービス データを含むコンポーネントまたは機能を選択します。ビューフィルターは、プロジェクトの機能、サブ機能、およびコンポーネントの階層をリスト表示します。機能を選択すると、その機能に含まれるすべてのコンポーネントのサービスが表示されます。*

サービスの設定は、次のメイン カテゴリに分かれています:

- [全般] の設定
- インストールの設定
- コントロールの設定
- 構成の設定

ヒント: *サービスの設定を構成するにあたっては、サービスの技術的な詳細についてよく知っておく必要があります。*

[全般] の設定

[全般] 領域を使って、サービスを含めるコンポーネントを指定します。

テーブル 147 • 全般設定

設定	説明
コンポーネント	サービス実行可能ファイルを含むコンポーネントを選択するか、この設定で省略記号ボタン (...) をクリックして、サービス実行可能ファイルのコンポーネントを作成または選択します。 サービス実行可能ファイルは、そのコンポーネントのキー ファイルでなくてはなりません。詳細については、「コンポーネントのキーファイル」を参照してください。 また、コンポーネントの "リモート インストール" 設定は [ローカルを優先] でなくてはなりません。詳細については、「コンポーネントの "リモート インストール" 設定を構成する」を参照してください。

インストールの設定

[インストールの設定] 領域を使って、インストール中にサービスをインストールするかどうかを指定します。サービスをインストールする場合、これらの設定を使って表示名や説明などの情報、およびサービスの開始条件を指定します。

テーブル 148 • インストールの設定

設定	説明
サービスのインストールを有効にする	サービスを実行時にインストールするかどうかを指定します。 この設定で [いいえ] を選択すると、"インストールの設定" セクション内の他の設定に入力した値はプロジェクトから削除されます。この設定で [いいえ] を選択すると、"インストールの設定" セクション内の他の設定は読み取り専用となります。
キー名	サービスの新しいキー名を入力します。このキー名はデータベース キーであり、エンドユーザーには表示されません。
表示名	サービス コントロール マネージャーにおける、このサービスの表示名を入力します。この設定を空白のままにすると、サービスの名前 ([詳細設定] ノードの下にあるサービスのサブノードの名前に使用されているテキスト) が使用されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 148 • インストールの設定 (続き.)

設定	説明
説明	サービスの説明を入力します。この説明は、サービスがインストールされる時にターゲット システムに登録され、サービス コントロール マネージャーの [説明] 列に表示されます。また、サービスの [プロパティ] ダイアログ ボックスの [一般] タブにある [説明] ボックスにも表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
サービスの種類	インストールするサービスの種類を選択します。選択可能なオプションは以下のとおりです: • 独自のプロセスで実行するWin32 • プロセスを共有するWin32 WIN32_OWN_PROCESS タイプのサービスには、1 つのサービスのコードが含まれます。WIN32_SHARE_PROCESS タイプのサービスには、1 つ以上のサービスのコードが含まれており、コードの共有が可能です。
デスクトップと対話する	サービスがデスクトップとインタラクトするかどうかを指定します。サービスにユーザー インターフェイスが含まれている場合は、[はい] を選択します。 [はい] を選択した場合、"ユーザー名" 設定は空白のままでなくてはなりません。これは、サービスが組み込み LocalSystem アカウントで実行するようにインストールされるためです。
開始の種類	サービスの開始時期を指定してください。選択可能なオプションは以下のとおりです: • **自動** – サービスはシステムが開始したときに自動的に開始します。 • **オン デマンド** – サービスは、サービス コントロール マネージャーを通して要求されたときに開始します。 • **無効** – サービスを開始することはできません。 一部のサービスは、別の開始タイプ (オペレーティング システムの初期化中、またはオペレーティング システム ローダによる) をサポートします。ただし、Windows Installer はこれらのサポートを含まないため、これらのオプションは "開始の種類" 設定では使用できません。

テーブル 148 • インストールの設定 (続き.)

設定	説明
エラーコントロール	サービスの開始に失敗した場合に、サービス コントロール マネージャーが行う適切な操作を選択します。選択可能なオプションは以下のとおりです: • エラーをログに記録してから作業を継続する • エラーをログに記録してから、メッセージを表示して作業を継続する • エラーをログに記録して再起動する
失敗時にインストールを中止する	サービスがターゲット システムにインストールできない場合は、インストール全体を失敗とするかどうかを指定します。デフォルト値は [いいえ] です。 ***メモ:*** *この設定で [はい] を選択して、エンド ユーザーがサイレントまたは基本の UI モードでインストールを実行する場合、ターゲット システムには Windows Installer 3 以降が必須となります。*
ロード順序グループ	このサービスがメンバーであるロード順グループがある場合、その名前を入力します。
依存関係	このサービスが必要とするサービス、またはロード順グループを入力してください。このサービスを開始する前に、システムは必要なサービス、またはロード順グループから最低 1 つのメンバーの開始を試行します。 複数の依存関係はコンマ (,) で区切ります。 サービス コントロール マネージャーがサービスとロード順グループを区別できるように、各ロード順グループ名の前に SC_GROUP_IDENTIFIER (通常は、プラス記号 (+)) を付ける必要があります。
ユーザー名	サービスにログオンするアカウントを入力してください。ローカル システム アカウントの下にサービスをインストールするには、この設定を空白のままに残します。(Microsoft は単一ユーザーの権限を偽装するサービスのインストールを推奨しません。) サービス タイプが**独自のプロセスで実行する Win32** の場合、入力する値には次の形式を使用します: **ドメイン名¥ユーザー名** サービスがビルトイン ドメインの下にログ記録される場合、次の形式を使用します: **.¥UserName**

テーブル 148 • インストールの設定 (続き.)

設定	説明
Password	このサービスのパスワードを入力してください。"ユーザー名" 設定が空白の場合 (つまり、サービスがローカル システム アカウントの下でログオンをされているとき) は、この設定を空白に残してください。ユーザー名を指定していない限りパスワードは使用されません。
開始パラメーター	サービスの実行に必要なコマンドライン パラメーターまたはプロパティを入力します。
アクセス許可	この設定を使って、サービスのアクセス許可を設定できます。 アクセス許可を構成するには、[新しいサービス アクセス許可の追加] ボタンをクリックします。InstallShield は、新しいアクセス許可設定と、その下に関連する設定の追加行を追加します。必要に応じて、個々の設定を構成します。 インストール中とアンインストール中に異なる引数を渡す場合などは、複数のアクセス許可項目を追加する必要があります。 ***メモ:*** *アクセス許可の設定は Windows Installer 5 よりサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* *1 つのインストールに* MsiLockPermissionsEx *テーブルと* LockPermissions *テーブルの両方を含めることはできません。この "アクセス許可" 設定と、その下にある設定を使う場合、パッケージの* MsiLockPermissionsEx *テーブルを構成していることになります。[一般情報] ビューの "ロックダウンの設定方法" 設定で* ***[従来型の Windows Installer 処理]*** *を選択した場合に、プロジェクト内の 1 つ以上のファイル、フォルダー、またはレジストリ キーにアクセス許可を設定すると、パッケージの* LockPermissions *テーブルを構成していることになります。*

テーブル 148 • インストールの設定 (続き.)

設定	説明
キー名	アクセス許可のキー名を入力します。このキー名は内部でのみ利用され、エンドユーザーには表示されません。 ***メモ:*** *アクセス許可の設定は Windows Installer 5 よりサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* *1 つのインストールに* MsiLockPermissionsEx *テーブルと* LockPermissions *テーブルの両方を含めることはできません。この "アクセス許可" 設定と、その下にある設定を使う場合、パッケージの* MsiLockPermissionsEx *テーブルを構成していることになります。[一般情報] ビューの "ロックダウンの設定方法" 設定で* ***[従来型の Windows Installer 処理]*** *を選択した場合に、プロジェクト内の 1 つ以上のファイル、フォルダー、またはレジストリ キーにアクセス許可を設定すると、パッケージの* LockPermissions *テーブルを構成していることになります。*
セキュリティ記述子	有効な SDDL (security descriptor definition language) を使用して、有効な SDDL 文字列を入力してください。SDDL についての詳細は、Microsoft Windows SDK (Software Development Kit) の "Access Control" セクションを参照してください。 ***ヒント:*** *山かっこ (例、* **<DomainName¥UserName>** *)または環境変数 (例、* **[%UserDomain][%UserName]** *) を使って、実行時にアカウント SID が決定されるユーザーのドメインとユーザー名を指定することができます。* ***メモ:*** *アクセス許可の設定は Windows Installer 5 よりサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* *1 つのインストールに* MsiLockPermissionsEx *テーブルと* LockPermissions *テーブルの両方を含めることはできません。この "アクセス許可" 設定と、その下にある設定を使う場合、パッケージの* MsiLockPermissionsEx *テーブルを構成していることになります。[一般情報] ビューの "ロックダウンの設定方法" 設定で* ***[従来型の Windows Installer 処理]*** *を選択した場合に、プロジェクト内の 1 つ以上のファイル、フォルダー、またはレジストリ キーにアクセス許可を設定すると、パッケージの* LockPermissions *テーブルを構成していることになります。*

テーブル 148 • インストールの設定 (続き.)

設定	説明
条件	この設定を使って、ターゲット システム上でサービスのアクセス許可を確保する前に、インストールが評価を行うステートメントを入力します。アクセス許可は、条件が Fase と評価された場合、構成されません。ただし、条件が True と評価された場合は、サービスの機能がインストールされているものとしてアクセス許可が構成されます。 アクセス許可の条件を作成するには、この設定で省略記号ボタン (...) をクリックします。詳細については、「[条件ビルダー] ダイアログ ボックス」を参照してください。 ***メモ:*** *アクセス許可の設定は Windows Installer 5 よりサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。* *1 つのインストールに* MsiLockPermissionsEx *テーブルと* LockPermissions *テーブルの両方を含めることはできません。この "アクセス許可" 設定と、その下にある設定を使う場合、パッケージの* MsiLockPermissionsEx *テーブルを構成していることになります。[一般情報] ビューの "ロックダウンの設定方法" 設定で **[従来型の Windows Installer 処理]** を選択した場合に、プロジェクト内の 1 つ以上のファイル、フォルダー、またはレジストリ キーにアクセス許可を設定すると、パッケージの* LockPermissions *テーブルを構成していることになります。*

コントロールの設定

[コントロールの設定] 領域を使って、このコンポーネントのインストールまたはアンインストール時にサービスを制御する方法を指定します。たとえば、実行中のサービスをアップデートする前に、インストールがそれを削除しなくてはならない場合があります。

テーブル 149 • コントロールの設定

設定	説明
イベント	この設定を使って、サービスに必要な 1 つ以上のコントロール イベントを指定できます。 イベントを追加するには、**[新しいサービス コントロール イベントの追加]** ボタンをクリックします。InstallShield が新しいイベント設定と、その下に関連設定の追加行を追加します。必要に応じて、個々の設定を構成します。 すべてのコントロール オプションを 1 つのイベントに配置することができます。ただし、インストールとアンインストールで異なった引数を渡す場合などは、別のイベントを作成しなければならない場合があります。
キー名	コントロール イベントのキー名を入力します。このキー名は内部でのみ利用され、エンドユーザーには表示されません。

テーブル 149 • コントロールの設定 (続き.)

設定	説明
操作時間	実行時にサービスに発生する操作をスケジュールする方法を指定するには、以下の各設定に対して適切なオプションを選択します: • **インストールの開始** — インストール中にサービスを開始するかどうかを指定します。[はい] を選択すると、インストールがインストール シーケンスの StartServices アクションに到達したときに、サービスが開始します。 • **インストールの停止** — インストール中にサービスを停止するかどうかを指定します。[はい] を選択すると、インストールがインストール シーケンスの StopServices アクションに到達したときに、サービスが停止します。 • **インストールの削除** — インストール中にサービスを削除するかどうかを指定します。[はい] を選択すると、インストールがインストール シーケンスの DeleteServices アクションに到達したときに、サービスが削除されます。 • **アンインストールの開始** — アンインストール中にサービスを開始するかどうかを指定します。[はい] を選択すると、アンインストールがインストール シーケンスの StartServices アクションに到達したときに、サービスが開始します。 • **アンインストールの停止** — アンインストール中にサービスを停止するかどうかを指定します。[はい] を選択すると、アンインストールがインストール シーケンスの StopServices アクションに到達したときに、サービスが停止します。 • **アンインストールの削除** — アンインストール中にサービスを削除するかどうかを指定します。[はい] を選択すると、アンインストールがインストール シーケンスの DeleteServices アクションに到達したときに、サービスが削除されます。
待機の種類	コントロール イベント開始後の、Windows Installer の実行方法を指定します。選択可能なオプションは以下のとおりです: • **イベントの完了を待機する** — Windows Installer は、続行する前に、最高 30 秒までイベント (開始、停止、または削除) の完了を待機します。イベントがインストールにとって重要である場合で、イベントがエラーを返すための追加時間を与えるときに、このオプションを選択します。 • **SCM の応答を待機する** — Windows Installer は、サービスコントロール マネージャーによりサービスが保留中であると報告があるまで待機します。 サービス コントロール マネージャーは、サービスに使用するシステムのデータベースを保持し、これらのサービスを制御するインターフェイスを提供します。

テーブル 149 • コントロールの設定 (続き.)

設定	説明
サービス引数	サービスに渡す引数を指定します。複数の引数はコンマ (,) で区切ります。

構成の設定

[構成の設定] 領域を使って、選択されたコンポーネントに含まれるサービスをカスタマイズする方法を設定します。

メモ• *アクセス許可の設定は Windows Installer 5 よりサポートされています。以前のバージョンの Windows Installer は、これらの設定を無視します。*

テーブル 150 • 構成の設定

設定	説明
イベント	この設定を使って、サービスに必要な 1 つ以上の構成の変更を指定できます。 イベントを追加するには、**[新しいサービス構成イベントの追加]** ボタンをクリックします。InstallShield が新しいイベント設定と、その下に関連設定の追加行を追加します。必要に応じて、個々の設定を構成します。 ***メモ•*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
キー名	構成イベントのキー名を入力します。このキー名は内部でのみ利用され、エンドユーザーには表示されません。 ***メモ•*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

テーブル 150 • 構成の設定 (続き.)

設定	説明
実行時	実行時にサービス構成イベントが発生する条件を指定するには、以下の各設定に対して適切なオプションを選択します: • **インストール中** – インストール中にイベントが発生するかどうかを指定します。 • **アンインストール中** – アンインストール中にイベントが発生するかどうかを指定します。 • **再インストール中** – 再インストール中にイベントが発生するかどうかを指定します。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

テーブル 150 • 構成の設定 (続き.)

設定	説明
構成の種類	サービスに行う構成の変更を指定します。変更は、次回のシステム開始時に有効になります。選択可能なオプションは以下のとおりです: • **自動開始の遅延時間を構成する** – このオプションは、サービスがインストール済み自動開始サービスの場合、または [インストールの設定] 領域で "開始の種類" 設定に [自動] が選択されている場合に適用されます。このオプションを選択するとき、"引数" 設定で省略記号ボタン (...) をクリックして遅延時間を示します。 • **回復操作の実行条件を構成する** – このオプションを選択するとき、"引数" 設定で省略記号ボタン (...) をクリックして、選択されたサービスの回復操作の実行条件を指定します。 • **プロセス トークンにサービス SID の種類を追加する** – このオプションを選択するとき、"引数" 設定で省略記号ボタン (...) をクリックして、サービスに追加する適切なセキュリティ識別子 (SID) の種類を指定します。 • **必要な特権のリストを変更する** – このオプションを選択するとき、"引数" 設定で省略記号ボタン (...) をクリックして適切な特権を選択します。 • **SCM シャットダウンの遅延を構成する** – このオプションを選択するとき、"引数" 設定に遅延時間をミリ秒単位で入力します。遅延時間をデフォルトの 3 分間にリセットするには、"引数" 設定を空白のまま残します。サービス コントロール マネージャーは、SERVICE_CONTROL_PRESHUTDOWN 通知をサービスに送信したあと、指定された時間待機します。 各オプションには、"引数" 設定に対応する一連の値があります。したがって、"引数" 設定の値を変更してから、"構成の種類" 設定の値を変更した場合、"引数" 設定の値を再び変更しなくてはならない場合があります。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
引数	"構成の種類" 設定で選択した構成変更に対応する値を指定するには、省略記号ボタン (...) をクリックします。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

テーブル 150 - 構成の設定 (続き.)

設定	説明
回復操作	この設定を使って、サービスが失敗した後に実行する 1 つ以上の回復操作を指定できます。 回復操作を追加するには、[**新しい回復操作の追加**] ボタンをクリックします。InstallShield が新しい回復操作設定と、その下に関連設定の追加行を追加します。必要に応じて、個々の設定を構成します。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
キー名	回復操作のキー名を入力します。このキー名は内部でのみ利用され、エンドユーザーには表示されません。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
実行時	サービスが失敗したあとに、回復操作の実行条件を指定するには、以下の各設定に対して適切なオプションを選択します: - **インストール中** – サービスのコンポーネントのインストール中に操作を実行します。 - **アンインストール中** – サービスのコンポーネントのアンインストール中に操作を実行します。 - **再インストール中** – サービスのコンポーネントの再インストール中に操作を実行します。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

テーブル 150 • 構成の設定 (続き.)

設定	説明
リセット期間	サービスのエラー カウントをリセットする間隔を秒単位で指定します。 サービス コントロール マネージャーは、システムが最後に再起動されてからサービスが失敗した回数をカウントします。指定された間隔が経過すると、リセット期間中にサービスが失敗しなかった場合、カウントが 0 にリセットされます。サービスが失敗した場合、システムは ″SCM 操作″ 設定で指定された操作を行います。適切な操作は、最後にシステムが再起動された以降のエラーの数から 1 を引いた数で決まります。たとえば、操作が 6 回目に失敗した場合、システムは、″SCM 操作″ 設定で 5 番めにリストされている種類の操作を行います。″SCM 操作″ 設定に操作が指定されていない場合、″リセット期間″ 設定は無視されます。 エラー カウントのリセットを行わない場合は、この設定を空白のままに残します。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
再起動メッセージ	サービス コントロール マネージャー操作の [コンピューターの再起動] に応答してコンピューターを再起動する前に表示するメッセージを指定します。[コンピューターの再起動] が、″SCM 操作″ 設定で操作の種類の 1 つとしてリストされていなくてはなりません。そうでない場合、″再起動メッセージ″ 設定は無視されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

テーブル 150 • 構成の設定 (続き.)

設定	説明
コマンドの実行	サービス コントロール マネージャーの [実行] コマンドに応答して実行するコマンドラインを指定します。絶対パスを使用します (UNC パスはサポートされていません)。例えば、**¥¥computername¥c$¥myscripts¥recovery.cmd** ではなく、**C:¥myscripts¥recovery.cmd** と指定します。エンド ユーザーからの入力を必要としないプログラムまたはスクリプトを指定します。 指定するコマンドラインは、サービスと同じアカウントの下で実行する新しいプロセスによって使用されます。 [コマンドの実行] が、"SCM 操作" 設定で操作の種類の 1 つとしてリストされていなくてはなりません。そうでない場合、"コマンドの実行" 設定は無視されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
SCM 操作	この設定を使って、サービスが失敗したときに発生するサービス コントロール マネージャーの 1 つ以上の操作を指定できます。操作は、この設定でリストされている順番で実行されます。 操作を追加するには、**[新しい SCM 操作イベントの追加]** ボタンをクリックします。InstallShield が新しい "種類" 設定と、その下に "遅延" 設定を追加します。必要に応じて、両方の設定を構成します。 ***メモ:*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

テーブル 150 • 構成の設定 (続き.)

設定	説明
Type	サービスの開始に失敗した場合に、サービス コントロール マネージャーが行う操作の種類を指定します。選択可能なオプションは以下のとおりです: • コマンドの実行 • コンピューターの再起動 • 何もしない • サービスの再開 ***メモ•*** *[サービスの再開] アクションは、.msi パッケージではサポートされていません。.msi パッケージと共にこのアクションを含めると、Windows Installer エラー1939 が発生します。* ***メモ•*** *"タイプ" 設定は、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*
遅延	サービス コントロール マネージャーが "種類" 設定で指定された操作を行う前に待機する時間を秒単位で指定します。 ***メモ•*** *このプロパティは、Windows Installer バージョン 5 からサポートされています。以前のバージョンの Windows Installer はこの設定を無視します。*

[サーバー構成] ビュー

プロジェクト• *[サーバー構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

InstallShield には、サーバー サイド インストールを構成するための以下のビューがあります。

インターネット インフォメーション サービス (IIS)

プロジェクト: *[IIS 構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

インストール時に IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張をターゲット マシン上に作成する場合、[IIS 構成] ビューで構成することができます。

コンポーネント サービス

プロジェクト: *[コンポーネント サービス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[コンポーネント サービス] ビューでは、インストール パッケージ用の COM+ サーバー アプリケーションとコンポーネントを管理できます。

SQL スクリプト

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

SQL スクリプトビューは、すべての SQL スクリプトを接続および設定に応じて管理並びに編成するための拠点です。

[IIS 構成] ビュー

プロジェクト: *[IIS 構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューでは、新しい IIS Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張を作成および管理することができます。

[IIS 構成] ビューには、次のような領域があります:

- Web サイト (この領域では、Web アプリケーションと仮想ディレクトリを追加できます。詳細については、「"アプリケーション" と "仮想ディレクトリ" の設定」を参照してください。)
- アプリケーション プール
- Web サービス拡張

次のテーブルは、[IIS 構成] ビューの設定の上に表示されるボタンについて説明します。これらのボタンは、中央ペインで Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張が選択されている場合に表示されます。

テーブル 151 • [IIS 構成] ビューのコントロール

コントロールの名前	アイコン	説明
カテゴリー別		カテゴリごとに設定を並べ替えます。
アルファベット順		設定をアルファベット順に並べ替えます。

ヒント• *Windows Installer パブリック プロパティ (基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトの場合)、またはテキスト置換文字列変数 (InstallScript プロジェクトの場合) を使って、実行時に IIS をダイナミックに構成することができます。これにより、エンドユーザーがターゲット マシンにインストールしている仮想ディレクトリの名前、TCP ポート、サイト番号、または Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張の他の IIS 設定を指定できるようにすることができます。詳しくは、「Windows Installer のプロパティを使用して、IIS 設定を動的に変更する」と「InstallScript テキスト置換を使用して、IIS 設定を動的に変更する」を参照してください。*

Web サイトの設定

プロジェクト• *[IIS 構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

[IIS 構成] ビューの [Web サイト] 項目を使って Web サイトの追加/削除、またシステム全体を通した Web サーバーの構成が可能です。

プロジェクト内のすべての Web サイトに適用される Web サーバーの設定

[IIS 構成] ビューで [Web サイト] エクスプローラーを選択すると、次の Web サーバーの設定が表示されます。

テーブル 152 • Web サーバー の設定

設定	説明
IIS を構成した後、Web サーバーを再起動する (IIS 6 以前のみ)	インストールで IIS の変更がシステムに加えられたとき、毎回インストールの完了時に IIS を再起動する場合、[はい] を選択します。アプリケーションによっては、IIS の再起動が必要です。 この設定は IIS 6 以前に適用します。IIS 7 は、この設定を無視します。
SSIEnableCmdDirective レジストリ値	HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥W3SVC¥Parameters レジストリ キーの SSIEnableCmdDirective レジストリ値をターゲット システムでどのように設定するかを指定します。SSIEnableCmdDirective レジストリ値は、サーバー側インクルード (SSI) の #exec CMD ディレクティブがシェル コマンドの実行に使用されることを Web サーバーが許可するかどうかを制御します。有効なオプションは次のとおりです: • **無視する** – ターゲット システム上の SSIEnableCmdDirective レジストリ値を変更しません。デフォルトでは、これが設定されています。 • **FALSE (0)**–ターゲット システム上の SSIEnableCmdDirective レジストリ値を 0 に設定します。これにより、サーバー側インクルードの #exec CMD ディレクティブがシェル コマンドの実行に使用されるを防ぐことができます。この値を選択すると、IIS Web サーバーに #exec CMD ディレクティブに依存するアプリケーションが存在した場合、インストール プロジェクトの Web サイトおよび仮想ディレクトリがインストールされたあと、これらのアプリケーションが誤作動を起こす可能性があります。 • **TRUE (1)**–ターゲット システム上の SSIEnableCmdDirective レジストリ値を 1 に設定します。これにより、サーバー側インクルードの #exec CMD ディレクティブがシェル コマンドの実行で使用できるようになります。 FALSE または TRUE オプションを選択すると、値 (FALSE の場合 0、TRUE の場合 1) がINSTALLSHIELD_SSI_PROP プロパティに格納されます。 セキュリティに関する懸念により、デフォルトの SSIEnableCmdDirective 値は FALSE (0) になっています。FALSE (0) 値により、エンドユーザーによって承認されていないサーバー側での実行可能ファイルの実行を防ぐことができます。 ***メモ:*** *製品のインストール中に SSIEnableCmdDirective レジストリ値が変更された場合でも、ターゲット システムから製品がアンインストールされるときに、SSIEnableCmdDirective レジストリ値が変更されることはありません。* 詳細については、「Web サーバーで CMD コマンドが SSI #exec ディレクティブに使用されるのを許可するかどうかを指定する」を参照してください。

メモ: 上記の Web サーバーの設定は、IIS アイテム (Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、または Web サービス拡張) が何もインストールされなかった場合、ターゲット システムで更新されません。

プロジェクトに含まれるの各 Web サイトについて構成可能な設定

エクスプローラーで Web サイトを選択すると、多くの設定が表示されます。Web サイトの設定は、いくつかのメイン カテゴリで構成されています:

- 識別
- 全般
- ホーム ディレクトリ
- アプリケーションの設定
- セキュリティ
- 詳細

識別の設定

[IIS 構成] ビューの Web サイトにある [識別] 領域には、次の設定があります:

テーブル 153 • Web サイトの識別の設定

設定	説明
名前	Web サイトの名前を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
IP アドレス	特定の IP アドレスをターゲットする場合に入力します。 代りに、デフォルト値のアステリスク (*) のままに残すこともできます。この設定の値をアステリスクまたは空白にすると、現在使用中でない任意の IP アドレスが使用されます。
TCP ポート	IIS Web サイト用の TCP ポート設定は、ターゲットマシン上でサービスが実行されているポート番号を示します。IIS Web サーバーには複数の Web サイトをホストできるバージョンもあります。それぞれの Web サイトは固有のポート番号に関連付けられています。 ターゲット システムのポート番号の指定が難しい場合、この設定で 0 を入力します。 Web サイトがインストールされたとき、Web サイトが使用するポート番号とサイト番号についての詳細は、「TCP ポート番号とサイト番号の構成」を参照してください。

テーブル 153 • Web サイトの識別の設定 (続き.)

設定	説明
ホスト ヘッダー名	インストール中にインストールされる IIS Web サイトを識別するホスト ヘッダー名を指定するには、このボックスにその名前を入力します。例: **www.mycompany.com** ホスト ヘッダー (ドメイン名とも呼ばれます) を利用して、複数の Web サイトを Web サーバー上の IP アドレスに割り当てることができます。
サイト番号	"サイト番号" 設定は Web サイトが作成される場所のパスに含まれる番号を示します (例、w3svc/3)。デフォルト値は 0 です。 ターゲット システムのサイト番号の指定が難しい場合、この設定で 0 を入力します。 Web サイトがインストールされたとき、Web サイトが使用するポート番号とサイト番号についての詳細は、「TCP ポート番号とサイト番号の構成」を参照してください。

[全般] の設定

[IIS 構成] ビューの Web サイトにある [全般] 領域には、次の設定があります:

テーブル 154 • Web サイトの全般の設定

設定	説明
コンポーネント	Web サイトが関連付けられているコンポーネントを選択します。省略記号ボタン (...) をクリックして、既存のコンポーネントを参照するか、新しいコンポーネントを作成することもできます。
ASP.NET バージョン	Web サイトの ASP.NET バージョンを設定するには、完全なバージョン番号を入力するか、一覧からそれを選択します。 たとえば、バージョン 2 の ASP.NET を指定する場合、**2.0.50727** と入力します。バージョン 1.1 の ASP.NET を指定するには、**1.1.4322** と入力します。 Web サイトの ASP.NET バージョンを指定すると、IIS はそのバージョン番号をすべての Web サイトのアプリケーションに使用します。 ***重要** • インストールが Windows Vista 以降のシステムで実行される可能性がある場合、ASP.NET バージョンを指定しないほうがよい場合があります。また、バージョン 3 の ASP.NET を指定すると、実行時にエラーが発生しますので注意してください。詳細については、「Web サイトまたはアプリケーションの ASP.NET バージョンを設定する」を参照してください。*

テーブル 154 • Web サイトの全般の設定 (続き.)

設定	説明
ASP.NET プラットフォーム	.NET Framework がインストールされている Windows の 64 ビット バージョンでインストールを実行できるようにするには、どの ASP.NET プラットフォームを使って Web サイト、またはアプリケーションを ASP.NET バージョンにマップするのかを指定します: • **32-bit** – ASP.NET IIS 登録ツールの 32 ビット バージョンを使用します。アプリケーション プールの "32 ビット アプリケーションを有効にする" 設定で [はい] を選択する場合は、このオプションを選択します。このオプションを選択しなかった場合は、インストールが失敗します。 • **64-bit** – ASP.NET IIS 登録ツールの 64 ビット バージョンを使用します。アプリケーション プールの "32 ビット アプリケーションを有効にする" 設定で [いいえ] を選択する場合は、このオプションを選択します。このオプションを選択しなかった場合は、インストールが失敗します。 詳細については、「64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮」を参照してください。
アンインストール時に削除する	選択した Web サイトをアンインストール中に削除するかどうかを指定します。 詳細については、「Web サイト、アプリケーション、および仮想ディレクトリのアンインストール」を参照してください。
デフォルト ドキュメント	Web サイトのデフォルト ページの名前を入力します。複数のページを指定するには、名前をコンマで区切ります。 Web サイトは、ブラウザーのリクエストでドキュメント名が指定されない場合に、デフォルト ページとして使用されます。

[ホーム ディレクトリ] の設定

[IIS 構成] ビューの Web サイトにある [ホーム ディレクトリ] 領域には、次の設定があります:

テーブル 155 • Web サイトのホーム ディレクトリの設定

設定	説明
コンテンツ ソース パス (ローカルまたは UNC)	この設定は、Web サイトのファイルを格納するローカル パスまたはネットワーク ディレクトリのパスを識別します。 • Web サイトのコンテンツがターゲット システムに存在する場合、この設定の省略記号ボタン (...) をクリックして、ローカル パスを指定します。[ディレクトリの参照] ダイアログ ボックスが開きます。基本の MSI または InstallScript MSI プロジェクトでは、このダイアログ ボックスで Windows Installer プロパティ (たとえば、[IISROOTFOLDER]) を選択するか、または新しいプロパティを作成できます。InstallScript プロジェクトでは、このダイアログ ボックスで InstallScript 変数 (たとえば、<IISROOTFOLDER>) を選択するか、または新しい変数を作成できます。 デフォルトでは、これらのファイルは IISROOTFOLDER に格納されています。 • Web サイトのコンテンツがターゲット システムに存在する場合、この設定の [UNC] ボタンをクリックして、ネットワークの場所を指定します。参考例: ¥¥server¥share ***ヒント:*** *各 Web サイトには一意の物理パスが必要です。一意のパスは、Web サイトが Windows Vista 以降または Windows Server 2008 以降のシステムにインストールされる場合、特に重要です。詳細については、「IIS サポートの実行時要件」を参照してください。*
スクリプトソースへのアクセス	読み取りまたは書き込み許可のどちらかが設定されている場合にエンドユーザーがソース コードへアクセスできるようにするかどうかを指定します。ソース コードには ASP アプリケーションのスクリプトが含まれます。
読み取りアクセス	エンドユーザーが Web サイトに読み取りアクセスできるようにするかどうかを指定します。
書き込みアクセス	エンドユーザーが Web サイトに書き込みアクセスできるようにするかどうかを指定します。これにより、エンド ユーザーがターゲット マシン上の Web サイトのプロパティを変更できるようになります。
ディレクトリの参照	エンド ユーザーが、この Web サイトの下にあるすべての仮想ディレクトリとサブディレクトリを参照できるかどうかを指定します。
アクセスのログ記録	ログ ファイルに、この Web サイトへのアクセスを記録するかどうかを指定します。この Web サイトのログ記録が有効な場合のみ、アクセスが記録されます。

テーブル 155・Web サイトのホーム ディレクトリの設定 (続き.)

設定	説明
このリソースにインデックスを付ける	Microsoft インデックス サービスが Web サイトの全文インデックス上に このディレクトリを含むようにするかどうかを指定します。 この設定は IIS 6 以前に適用します。IIS 7 は、この設定を無視します。

アプリケーションの設定

[IIS 構成] ビューの Web サイトにある [アプリケーション] 領域には、次の設定があります:

テーブル 156・Web サイトのアプリケーションの設定

設定	説明
アプリケーション プール	選択した Web サイトをアプリケーション プールに関連付けるには、その名前を一覧から選択します。その代わりに、省略記号ボタン (...) をクリックして、アプリケーション プールの文字列エントリを作成することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は IIS 6 以降に適用します。IIS の以前のバージョンは、この設定を無視します。 ***重要・指定するアプリケーション プールは、実行時にターゲット システムに存在するものか、インストールの一部になっているものを指定します。そうされなかった場合、サーバーでエラー (403.18 など) が発生する可能性があります。***
アプリケーション マッピング	ディレクトリのアプリケーションのマッピングをカスタマイズするには、この設定で省略記号ボタン (...) をクリックします。これによって [アプリケーションのマッピング] ダイアログ ボックスが開きます。ここではファイル名拡張子とそれらのファイルを処理するアプリケーションとの間のマッピングを編集または削除することができます。
MIMEタイプ	選択した Web サイトに MIME の種類を追加、変更、または削除するには、この設定で省略記号ボタン (...) をクリックします。[MIME の種類] ダイアログ ボックス が開きます。ここで、ファイル名 拡張子と、ターゲット システム上の Web サーバーからブラウザまたはメール クライアントにスタティック ファイルとして提供される対応コンテンツの種類とのマッピングを追加、編集、または削除します。
セッション タイムアウト(分)	サーバーが自動的に終了する前にセッションをアイドル状態に保つ時間を分単位で指定します。エンド ユーザーがこのタイムアウト期間中にページを更新したりページを要求しないとセッションが終了します。デフォルト値は 20 分です。
ASP スクリプト タイムアウト(秒)	.asp ページでスクリプトが実行でき、Windows イベントログ記録に書き込むことができる時間を秒数で指定します。このプロパティの最小値は 1 秒で、デフォルト値は 90 秒です。

セキュリティの設定

[IIS 構成] ビューで Web サイトを選択すると、[セキュリティ] 領域にいくつかのセキュリティ関連の設定が表示されます。[セキュリティ] 領域では、ユーザー ID を検証できるように Web サーバーを構成することができます。ユーザーを認証することで、認証されていないユーザーによる制限された内容への Web (HTTP) 接続を不可能にします。詳細については、IIS マニュアルを参照してください。

[セキュリティ] 領域の設定は、いくつかのカテゴリに分かれています:

- 匿名接続
- 認証済みアクセス
- セキュリティで保護された通信

[匿名接続] 領域の設定は次のとおりです:

テーブル 157 • [セキュリティの設定] 領域にある [匿名接続] の設定

設定	説明
匿名アクセスを有効にする	ユーザーが匿名アクセスできるようにするかどうかを指定します。匿名接続を許可する場合、適切な Windows ユーザー アカウント情報も入力します。 エンド ユーザーがコンテンツにアクセスする前に Web サーバーによる ID の確認を必要としない場合、この設定に [いいえ] を選択します。
IIS による匿名パスワードの制御	匿名パスワード設定がターゲット システム上の Windows での設定と自動的に統一されるようにするかどうかを指定します。匿名アカウント用に入力したパスワードが Windows のパスワードと異なる場合、匿名認証は正しく動作しません。 ***メモ:*** *パスワードの同期化はリモート コンピューター上の匿名アカウントではなく、ローカル コンピューター上で定義された匿名ユーザー アカウントと共に使用しなくてはなりません。*
匿名ユーザー名	匿名接続を有効にする場合、匿名アカウントの名前を入力します。
匿名パスワード	[IIS による匿名パスワードの制御] 設定で [いいえ] を選択した場合は、匿名ユーザー アカウントのパスワードを入力します。パスワードは Windows 内でのみ利用されます。匿名ユーザーはユーザー名とパスワードを使ってログオンしません。

[認証済みアクセス] 領域の設定は次のとおりです:

テーブル 158 • [セキュリティの設定] 領域にある [認証済みアクセス] の設定

設定	説明
基本認証	基本の認証メソッドを使って、Web サイトにアクセスするエンド ユーザーのユーザー名とパスワード情報を収集するかどうかを指定します。 ***重要 • 基本の認証方法では、ネットワークを介してユーザー名とパスワードが送信されるときに情報が暗号化されません。ネットワーク監視ツールを利用する無法なエンド ユーザーによって、ユーザー名およびパスワードが傍受される可能性があります。***
統合 Windows 認証	統合 Windows 認証を有効化するかどうかを指定します。統合 Windows 認証は、ユーザー ID を確認するためにユーザーのブラウザーと暗号化されたやり取りを行います。 統合 Windows 認証が有効になっている場合、Web サーバーは次の条件下でのみそれを利用します。 • 匿名アクセスは無効です。 • Windows ファイルシステム アクセス許可が設定されいているため、匿名アクセスは拒否されます。エンド ユーザーは制限されたコンテンツへ接続を確立する前に Windows ユーザー名とパスワードを提示するよう求められます。
フォーム認証	このオプションを **[はい]** に設定してフォーム認証を有効化します。ASP.NET フォーム ベースの認証は、多くの要求を受け取るパブリック Web サーバー上にあるサイトまたはアプリケーションに適しています。この認証モードでは、オペレーティング システムが提供している認証方法に頼ることなく、アプリケーション レベルでのクライアント登録および認証を管理することができます。 ***重要 • フォーム認証は、Web サーバーにユーザー名とパスワードをテキスト形式で送信します。ホームページ以外のアプリケーション内のすべてのページ、およびログオン ページには、Secure Sockets Layer (SSL) 暗号化を使用してください。***

[セキュリティで保護された通信] 領域の設定は、次の通りです。

テーブル 159 • [セキュリティの設定] 領域にある [セキュリティで保護された通信] の設定

設定	説明
SSL 証明書	ターゲット システムにインストールする必要があるサーバー証明書を指定するには、この設定で省略記号ボタン (...) をクリックしてから、適切なセキュリティ証明書ファイル (.cer または .pfx) を選択します。プロジェクトに 1 つまたは複数の証明書が既に含まれている場合は、リストから証明書を選択できます。 .cer ファイルが Binary テーブルに格納されます。 証明書のインストールが構成されていない場合、この設定は空白です。
SSL 証明書のパスワード	指定した証明書にパスワードがある場合、それをこの設定に入力します。

詳細設定

[IIS 構成] ビューの Web サイトにある [詳細] 領域には、次の設定があります。

テーブル 160 • Web サイトの詳細の設定

設定	説明
カスタム エラー	Web サーバー エラーが発生したときにクライアントに送信される HTTP エラーをカスタマイズするには、この設定で省略記号ボタン (...) を選択します。[カスタム エラー] ダイアログ ボックスが開き、ここで 1 つまたは複数の HTTP エラーで表示するページを指定できます。 管理者は汎用 HTTP 1.1 エラー、IIS が提供する詳細なカスタム エラー ページ、またはインストールに含めた独自のカスタム エラーを使用できます。
その他の IIS プロパティ	このビューの別の領域では表示されない IIS 設定の値を指定するには、この設定で省略記号ボタン (...) を選択します。[その他の IIS プロパティ] ダイアログ ボックスが開き、ここで 1 つまたは複数の IIS プロパティの値を設定できます。詳細については、「IIS の詳細設定を構成する」を参照してください。 詳細設定は IIS 6 以前に適用します。IIS 7 は、これらの設定を無視します。 特定の設定についてのヘルプ、MSDN Web サイトの「メタベース プロパティ リファレンス」を参照してください。

"アプリケーション" と "仮想ディレクトリ" の設定

プロジェクト: *[IIS 構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *マージ モジュール*

[IIS] ビューで、Web サイトにアプリケーションまたは仮想ディレクトリを追加することができます。このビューでアプリケーションまたは仮想ディレクトリを選択すると、多くの設定が表示されます。この設定は、いくつかのメイン カテゴリで構成されています:

- 全般
- 仮想ディレクトリ
- アプリケーションの設定
- セキュリティ
- 詳細

[全般] の設定

[IIS 構成] ビューの アプリケーションまたは仮想ディレクトリにある [全般] 領域には、次の設定があります。

テーブル 161・アプリケーションまたは仮想ディレクトリにおける全般の設定

設定	説明
名前	アプリケーションまたは仮想ディレクトリの名前を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
コンポーネント	アプリケーションまたは仮想ディレクトリが関連付けられているコンポーネントを選択します。省略記号ボタン (...) をクリックして、既存のコンポーネントを参照するか、新しいコンポーネントを作成することもできます。

テーブル 161 • アプリケーションまたは仮想ディレクトリにおける全般の設定 (続き.)

設定	説明
ASP.NET バージョン	 *メモ• この設定はアプリケーションで使用できますが、仮想ディレクトリには無効です。* アプリケーションの ASP.NET バージョンを設定するには、完全なバージョン番号を入力するか、一覧からそれを選択します。 たとえば、バージョン 2 の ASP.NET を指定する場合、**2.0.50727** と入力します。バージョン 1.1 の ASP.NET を指定するには、**1.1.4322** と入力します。 Web サイトの ASP.NET バージョンを指定すると、IIS はそのバージョン番号をすべての Web サイトのアプリケーションに使用します。 ***重要*** *• インストールが Windows Vista 以降のシステムで実行される可能性がある場合、ASP.NET バージョンを指定しないほうがよい場合があります。また、バージョン 3 の ASP.NET を指定すると、実行時にエラーが発生しますので注意してください。詳細については、「Web サイトまたはアプリケーションの ASP.NET バージョンを設定する」を参照してください。*
ASP.NET プラットフォーム	 *メモ• この設定はアプリケーションで使用できますが、仮想ディレクトリには無効です。* .NET Framework がインストールされている Windows の 64 ビット バージョンでインストールを実行できるようにするには、どの ASP.NET プラットフォームを使ってアプリケーションを ASP.NET バージョンにマップするのかを指定します: • **32-bit** – ASP.NET IIS 登録ツールの 32 ビット バージョンを使用します。アプリケーション プールの "32 ビット アプリケーションを有効にする" 設定で [はい] を選択する場合は、このオプションを選択します。このオプションを選択しなかった場合は、インストールが失敗します。 • **64-bit** – ASP.NET IIS 登録ツールの 64 ビット バージョンを使用します。アプリケーション プールの "32 ビット アプリケーションを有効にする" 設定で [いいえ] を選択する場合は、このオプションを選択します。このオプションを選択しなかった場合は、インストールが失敗します。 詳細については、「64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮」を参照してください。

テーブル 161 • アプリケーションまたは仮想ディレクトリにおける全般の設定 (続き.)

設定	説明
ASP.NET 登録	ASP.NET アプリケーション登録オプションを IIS (インターネット インフォメーション サービス) を使って設定するには、次のオプションの 1 つを選択します: • **再帰的** — 指定されたアプリケーションおよびすべてのサブアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。 • **非再帰的** — 特定のアプリケーションのスクリプト マップとアプリケーション プールの割り当てを更新します。サブアプリケーションは変更されません。
デフォルト ドキュメント	アプリケーションまたは仮想ディレクトリのデフォルト ページの名前を入力します。複数のページを指定するには、名前をコンマで区切ります。 アプリケーションまたは仮想ディレクトリは、ブラウザーのリクエストでドキュメント名が指定されない場合に、デフォルト ページとして使用されます。

[仮想ディレクトリ] の設定

[IIS 構成] ビューの アプリケーションまたは仮想ディレクトリにある [仮称ディレクトリ] 領域には、次の設定があります。

テーブル 162 ・ アプリケーションまたは仮想ディレクトリにおける仮想ディレクトリの設定

設定	説明
コンテンツ ソース パス (ローカルまたは UNC)	この設定は、アプリケーションまたは仮想ディレクトリのデフォルト ファイルを格納するローカル パスまたはネットワーク ディレクトリ パスを識別します。 • アプリケーション、または Web ディレクトリのコンテンツがターゲット システムに存在する場合、この設定の省略記号ボタン (...) をクリックして、ローカル パスを指定します。[ディレクトリの参照] ダイアログ ボックスが開きます。基本の MSI または InstallScript MSI プロジェクトでは、このダイアログ ボックスで Windows Installer プロパティ (たとえば、[IISROOTFOLDER]) を選択するか、または新しいプロパティを作成できます。InstallScript プロジェクトでは、このダイアログ ボックスで InstallScript 変数 (たとえば、<IISROOTFOLDER>) を選択するか、または新しい変数を作成できます。 デフォルトでは、これらのファイルは IISROOTFOLDER に格納されています。 • アプリケーションまたは仮想ディレクトリのコンテンツがターゲット システムに存在する場合、この設定の [UNC] ボタンをクリックして、ネットワークの場所を指定します。参考例: ¥¥server¥share ***ヒント・*** *各アプリケーションまたは仮想ディレクトリには一意の物理パスが必要です。一意のパスは、仮想ディレクトリが Windows Vista 以降または Windows Server 2008 以降のシステムにインストールされる場合、特に重要です。詳細については、「IIS サポートの実行時要件」を参照してください。*
スクリプトソースへのアクセス	読み取りまたは書き込み許可のどちらかが設定されている場合にエンドユーザーがソース コードへアクセスできるようにするかどうかを指定します。ソース コードには ASP アプリケーションのスクリプトが含まれます。
読み取りアクセス	エンドユーザーがアプリケーションまたは Web サイトに読み取りアクセスできるようにするかどうかを指定します。
書き込みアクセス	エンドユーザーがアプリケーションまたは Web サイトに書き込みアクセスできるようにするかどうかを指定します。これにより、エンド ユーザーがターゲットマシン上のアプリケーションまたは仮想ディレクトリのプロパティを変更できるようになります。
ディレクトリの参照	エンド ユーザーが、このアプリケーションまたは Web サイトの下にあるすべての仮想ディレクトリとサブディレクトリを参照できるかどうかを指定します。

テーブル 162 ・ アプリケーションまたは仮想ディレクトリにおける仮想ディレクトリの設定 (続き.)

設定	説明
アクセスのログ記録	ログ ファイルに、このアプリケーションまたは Web サイトへのアクセスを記録するかどうかを指定します。アクセス数は、ログ記録が有効化されている場合のみ記録されます。
このリソースにインデックスを付ける	Microsoft のインデックス サービスを許可して、このアプリケーションまたは仮想ディレクトリを全文インデックスに含めるかどうかを指定します。 この設定は IIS 6 以前に適用します。IIS 7 は、この設定を無視します。

アプリケーションの設定

[IIS 構成] ビューの アプリケーションまたは仮想ディレクトリにある [アプリケーションの設定] 領域には、次の設定があります。

テーブル 163 ・ アプリケーションまたは仮想ディレクトリにおけるアプリケーションの設定

設定	説明
アプリケーション名	選択した仮想ディレクトリをアプリケーションと関連付けるには、アプリケーションの名前を指定します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ・*** *この設定は仮想ディレクトリで使用できますが、アプリケーションには無効です。* *仮想ディレクトリが InstallShield 2009 以前で作成されていて、InstallShield の現在のバージョンにアップグレードされている場合、この設定が表示されます。それ以外の場合、この設定は含まれません。*

テーブル 163 • アプリケーションまたは仮想ディレクトリにおけるアプリケーションの設定 (続き.)

設定	説明
アプリケーション プール	 *メモ• この設定はアプリケーションで使用できますが、仮想ディレクトリには無効です。* 選択したアプリケーションをアプリケーション プールに関連付けるには、一覧からその名前を選択します。その代わりに、省略記号ボタン (...) をクリックして、アプリケーション プールの文字列エントリを作成することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 この設定は IIS 6 以降に適用します。IIS の以前のバージョンは、この設定を無視します。 ***重要*** *• 指定するアプリケーション プールは、実行時にターゲット システムに存在するものか、インストールの一部になっているものを指定します。そうされなかった場合、サーバーでエラー (403.18 など) が発生する可能性があります。*
アプリケーション マッピング	ディレクトリのアプリケーションのマッピングをカスタマイズするには、この設定で省略記号ボタン (...) をクリックします。これによって [アプリケーションのマッピング] ダイアログ ボックスが開きます。ここではファイル名拡張子とそれらのファイルを処理するアプリケーションとの間のマッピングを編集または削除することができます。
MIMEタイプ	選択したアプリケーションまたは仮想ディレクトリの MIME の種類を追加、変更、または削除するには、この設定で省略記号ボタン (...) をクリックします。[MIME の種類] ダイアログ ボックス が開きます。ここで、ファイル名 拡張子と、ターゲット システム上の Web サーバーからブラウザまたはメール クライアントにスタティック ファイルとして提供される対応コンテンツの種類とのマッピングを追加、編集、または削除します。
セッション タイムアウト(分)	サーバーが自動的に終了する前にセッションをアイドル状態に保つ時間を分単位で指定します。エンド ユーザーがこのタイムアウト期間中にページを更新したりページを要求しないとセッションが終了します。デフォルト値は 20 分です。
ASP スクリプト タイムアウト(秒)	.asp ページでスクリプトが実行でき、Windows イベントログ記録に書き込むことができる時間を秒数で指定します。このプロパティの最小値は 1 秒で、デフォルト値は 90 秒です。

テーブル 163 ・ アプリケーションまたは仮想ディレクトリにおけるアプリケーションの設定 (続き.)

設定	説明
実行アクセス許可	選択したアプリケーションまたは仮想ディレクトリに対して許可する、プログラムの実行レベルを指定します。選択可能なオプションは以下のとおりです: • **なし** — HTML やイメージ ファイルといったスタティック ファイルのみアクセス可能です。 • **スクリプトのみ** — ASP スクリプトなどのスクリプトのみ実行可能です。 • **スクリプトと実行可能ファイル** — すべてのファイル タイプにアクセスまたはそれらを実行できます。
アプリケーション保護	 ***メモ・** この設定はアプリケーションで使用できますが、仮想ディレクトリには無効です。* 保護レベルを指定します: • **高** — アプリケーションは、別の処理とは切り離された単独の処理で実行します。 • **中** — アプリケーションはその他のアプリケーションと共に切り離されたプール プロセスで実行します。 • **低** — アプリケーションは Web サービスと同じ処理で実行します。 この設定は IIS 5 以前に適用します。それ以降のバージョンはこの設定を無視します。

セキュリティの設定

[IIS 構成] ビューでアプリケーションまたは仮想ディレクトリを選択すると、[セキュリティ] 領域にいくつかのセキュリティ関連の設定が表示されます。[セキュリティ] 領域では、ユーザー ID を検証できるようにアプリケーションまたは仮想ディレクトリ を構成することができます。ユーザーを認証することで、認証されていないユーザーによる制限された内容への Web (HTTP) 接続を不可能にします。詳細については、IIS マニュアルを参照してください。

[セキュリティ] 領域の設定は、次のカテゴリに分かれています:

- 匿名接続
- 認証済みアクセス

[匿名接続] 領域の設定は次のとおりです:

テーブル 164 • [セキュリティの設定] 領域にある [匿名接続] の設定

設定	説明
匿名アクセスを有効にする	ユーザーが匿名アクセスできるようにするかどうかを指定します。匿名接続を許可する場合、適切な Windows ユーザー アカウント情報も入力します。 エンド ユーザーがコンテンツにアクセスする前に Web サーバーによる ID の確認を必要としない場合、この設定に [いいえ] を選択します。
IIS による匿名パスワードの制御	匿名パスワード設定がターゲット システム上の Windows での設定と自動的に統一されるようにするかどうかを指定します。匿名アカウント用に入力したパスワードが Windows のパスワードと異なる場合、匿名認証は正しく動作しません。 ***メモ•*** *パスワードの同期化はリモート コンピューター上の匿名アカウントではなく、ローカル コンピューター上で定義された匿名ユーザー アカウントと共に使用しなくてはなりません。*
匿名ユーザー名	匿名接続を有効にする場合、匿名アカウントの名前を入力します。
匿名パスワード	[IIS による匿名パスワードの制御] 設定で [いいえ] を選択した場合は、匿名ユーザー アカウントのパスワードを入力します。パスワードは Windows 内でのみ利用されます。匿名ユーザーはユーザー名とパスワードを使ってログオンしません。

[認証済みアクセス] 領域の設定は次のとおりです:

テーブル 165 • [セキュリティの設定] 領域にある [認証済みアクセス] の設定

設定	説明
基本認証	基本の認証メソッドを使って、アプリケーションまたは仮想ディレクトリにアクセスするエンド ユーザーのユーザー名とパスワード情報を収集するかどうかを指定します。 ***重要 •*** *基本の認証方法では、ネットワークを介してユーザー名とパスワードが送信されるときに情報が暗号化されません。ネットワーク監視ツールを利用する無法なエンド ユーザーによって、ユーザー名およびパスワードが傍受される可能性があります。*

テーブル 165 • [セキュリティの設定] 領域にある [認証済みアクセス] の設定 (続き.)

設定	説明
統合 Windows 認証	統合 Windows 認証を有効化するかどうかを指定します。統合 Windows 認証は、ユーザー ID を確認するためにユーザーのブラウザーと暗号化されたやり取りを行います。 統合 Windows 認証が有効になっている場合、Web サーバーは次の条件下でのみそれを利用します。 • 匿名アクセスは無効です。 • Windows ファイルシステム アクセス許可が設定されいているため、匿名アクセスは拒否されます。エンド ユーザーは制限されたコンテンツへ接続を確立する前に Windows ユーザー名とパスワードを提示するよう求められます。

詳細設定

[IIS 構成] ビューの アプリケーションまたは仮想ディレクトリにある [詳細] 領域には、次の設定があります。

テーブル 166 • アプリケーションまたは仮想ディレクトリにおける詳細の設定

設定	説明
カスタム エラー	Web サーバー エラーが発生したときにクライアントに送信される HTTP エラーをカスタマイズするには、この設定で省略記号ボタン (...) を選択します。[カスタム エラー] ダイアログ ボックスが開き、ここで 1 つまたは複数の HTTP エラーで表示するページを指定できます。 管理者は汎用 HTTP 1.1 エラー、IIS が提供する詳細なカスタム エラー ページ、またはインストールに含めた独自のカスタム エラーを使用できます。
その他の IIS プロパティ	このビューの別の領域では表示されない IIS 設定の値を指定するには、この設定で省略記号ボタン (...) を選択します。[その他の IIS プロパティ] ダイアログ ボックスが開き、ここで 1 つまたは複数の IIS プロパティの値を設定できます。詳細については、「IIS の詳細設定を構成する」を参照してください。 詳細設定は IIS 6 以前に適用します。IIS 7 は、これらの設定を無視します。 特定の設定についてのヘルプ、MSDN Web サイトの「メタベース プロパティ リファレンス」を参照してください。

アプリケーション プールの設定

プロジェクト• *[IIS 構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *マージ モジュール*

メモ: *アプリケーション プールは、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

[IIS 構成] ビューにあるアプリケーション プール アイテムを使って、アプリケーション プールを追加および削除します。

エクスプローラーでアプリケーション プールを選択すると、多くの設定が表示されます。アプリケーション プールの設定は、いくつかのメイン カテゴリで構成されています:

- 全般
- CPU の設定
- プロセス モデル
- 再利用

[全般] の設定

[IIS 構成] ビューのアプリケーション プールにある [全般] 領域には、次の設定があります。

テーブル 167 ・ アプリケーション プールにおける全般の設定

設定	説明
名前	構成するアプリケーション プールの表示名を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
コンポーネント	アプリケーション プールが関連付けられているコンポーネントを選択します。省略記号ボタン (...) をクリックして、既存のコンポーネントを参照するか、新しいコンポーネントを作成することもできます。
既存アプリケーション プールの上書き	実行時にアプリケーション プールが既にターゲットシステムに存在する場合に発生する動作を示す適切なオプションを選択します。選択可能なオプションは以下のとおりです: - **はい** – アプリケーション プールが既にターゲット システムに存在する場合、インストールはその設定をプロジェクトで構成された値で上書きします。アンインストール中に、アプリケーション プールがアンインストールされます。 - **いいえ** – アプリケーション プールが既にターゲット システムに存在する場合、インストールはその設定を一切上書きしません。アンインストール中、アプリケーション プールがターゲット システムに残ります。アプリケーション プールがターゲット システムに存在しない場合、インストールはそれを作成します。アンインストール中、アプリケーション プールがアンインストールされます。

テーブル 167 • アプリケーション プールにおける全般の設定 (続き.)

設定	説明
コンポーネントをパーマネントとしてマークする	コンポーネントの機能がアンインストールされるときに、アプリケーション プールを含むコンポーネントを削除するかどうかを指定します。コンポーネントをパーマネントとしてマークして、アンインストールされないようにするには、[はい] を選択します。
.NET Framework バージョン	適切な場合、アプリケーション プールがロードする .NET Framework バージョンを選択します。 この設定は IIS 7 以降に適用します。IIS の以前のバージョンは、この設定を無視します。
キューの長さ	HTTP.sys がアプリケーション プールのキューを行う要求の最大数を指定します。デフォルト値は 4000 です。
マネージ パイプライン モード	選択したアプリケーション プールに適切なリクエスト処理のパイプライン モードを指定します。 • **統合** – ASP.NET 要求処理は、IIS 7 のリクエスト処理パイプラインに統合されます。デフォルトでは、これが設定されています。 • **クラシック** – IIS ルートは、ISAPI を通してコードを要求します。これは、アプリケーションが IIS 6 で実行中であるかのように要求を処理します。 この設定は IIS 7 に適用します。IIS の以前のバージョンは、この設定を無視します。
32 ビット アプリケーションを有効にする	選択したアプリケーションの 32 ビット アプリケーションを 64 ビット システム上で実行可能にするかどうかを指定します。 この設定は、IIS 7 以降にインストールされたアプリケーション プールに適用します。IIS の以前のバージョンは、この設定を無視します。 詳細については、「64 ビット プラットフォーム上で IIS 6 をサポートする際の考慮」を参照してください。

CPU の設定

[IIS 構成] ビューのアプリケーション プールにある [CPU] 領域には、次の設定があります。

テーブル 168 • アプリケーション プールにおける CPU の設定

設定	説明
制限	アプリケーション プールのワーカー プロセスが "制限間隔" 設定で示されている時間よりも超えることができる、CPU 時間の最大パーセントを指定します (1/1000 パーセントで指定)。 ワーカー プロセスを CPU 時間のパーセントに制限しない場合は、番号 0 を指定します。この設定を空白に残すと、CPU 制限は構成されず、サーバーでデフォルトとして設定されている任意の値がアプリケーション プールに使用されます。
制限アクション	アプリケーション プールに設定された CPU 時間を超過した場合のアクションを指定します。選択可能なオプションは以下のとおりです: • **何もしない** – イベント ログ記録にイベント エントリが追加されます。その他のアクションは発生しません。 • **シャットダウン** – イベント ログ記録にイベント エントリが追加されます。また、制限間隔の残り時間もプリケーション プールがシャットダウンします。
制限間隔 (分)	アプリケーション プールの CPU モニターと帯域幅調整制限のリセット間隔を分数で指定します。 この間隔が経過したときに、IIS はログ記録の CPU タイマーをリセットして、間隔を制限します。 CPU のモニターを無効化するには、数字 0 を指定します。この設定を空白に残すと、CPU 制限間隔が構成されず、サーバーでデフォルトとして設定されている任意の値がアプリケーション プールに使用されます。

プロセス モデルの設定

[IIS 構成] ビューのアプリケーション プールにある [プロセス モデル] 領域には、次の設定があります。

テーブル 169 • アプリケーション プールにおけるプロセス モデルの設定

設定	説明
ID	アプリケーション プールのワーカー プロセスが実行するアカウントを指定します。選択可能なオプションは以下のとおりです: • **NetworkService** – このアカウントはユーザー グループのメンバーで、アプリケーションを実行するのに必要な最小限のユーザー権限を持ちます。このアカウントは、Web サーバーの乗っ取りなどの攻撃に対する最も高度なセキュリティを提供します。 • **LocalService** – このアカウントは、[ユーザー] グループのメンバーで、NetworkService アカウントと同じユーザー権限を持ちます。ただし、LocalService アカウントはローカル コンピューターに制限されています。 • **LocalSystem** – このアカウントはすべてのユーザー権限を持ち、[管理者] グループに属します。 • **SpecificUser** – 定義済みのアカウントのいずれも使用しない場合はこのオプションを選択して、使用するアカウントのユーザー名とパスワードを指定します。 • **ApplicationPoolIdentity** – このアカウントは、アプリケーション プールのワーカプロセスを実行するために選択されたアプリケーション プールに一意な仮想 ID を使用します。このオプションのサポートは、IIS 7 以降が搭載されたターゲット システムで使用できます。IIS 6 が搭載されているシステム上で実行されるインストールでこの新しいオプションが選択された場合、アプリケーション プールの ID には NetworkService アカウントが代わりに使用されます。 ***メモ:*** *SpecificUser ID を使用する場合、指定するユーザー アカウントが適切なリソースにアクセスできるように、そのアカウントが Web サーバー上で IIS_IUSRS グループのメンバーであることを確認してください。*
SpecificUser ユーザー名	"識別" 設定に SpecificUser を選択する場合、特定のユーザーのユーザー名を入力します。

テーブル 169 ・ アプリケーション プールにおけるプロセス モデルの設定 (続き.)

設定	説明
SpecificUser ユーザー パスワード	"識別" 設定に SpecificUser を選択する場合、特定のユーザーのパスワードを入力します。 ***プロジェクト:*** *基本の MSI と InstallScript MSI プロジェクトでは、パスワードの Windows Installer プロパティを指定できます。パスワードがログ記録されるのを防ぐには、*MsiHiddenProperties *プロパティの値として、パスワードに使用する Windows Installer プロパティの名前を設定します。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「MsiHiddenProperties」を参照してください。*
セッション タイムアウト(分)	ワーカー プロセスがシャットダウンする前にアイドル状態を保持する時間を分数で指定します。
最大ワーカー プロセス	アプリケーション プールの要求を処理できるワーカー プロセスの最大数を指定します。 1 より大きい数字を指定すると、アプリケーション プールが Web ガーデンであると見なされます。
ユーザー プロファイルの読み込み	アプリケーション プール エンティティのユーザー プロファイルをロードする時の適切なオプションを選択してください。選択可能なオプションは以下のとおりです: • **はい** – IIS がアプリケーション プールのユーザー プロファイルをロードします。 • **いいえ** – IIS がアプリケーション プールのユーザー プロファイルをロードしません。これは IIS 6.0 のときと同じ動作です。

リサイクルの設定

[IIS 構成] ビューのアプリケーション プールにある "リサイクル" 設定では、ワーカー プロセスのリサイクルを設定できます。ワーカー プロセスの分離モードで、IIS を構成して、アプリケーション プールで定期的にワーカー プロセスを再起動することができ、欠陥のあるワーカー プロセスを正確に管理することができます。これにより、これらのプールにある指定されたアプリケーションが正常で、システム リソースが回復可能であることを保証できます。

[IIS 構成] ビューのアプリケーション プールにある [リサイクル] 領域には、次の設定があります。

テーブル 170 ・ アプリケーション プールにおけるリサイクルの設定

設定	説明
定期的な間隔(分)	アプリケーション プールがリサイクルされるまでの時間を分数で指定します。 アプリケーション プールが定期的な間隔でリサイクルされないようにするには、数字 0 を入力します。

テーブル 170 • アプリケーション プールにおけるリサイクルの設定 (続き.)

設定	説明
要求数の制限	アプリケーション プールがリサイクルされる前に処理する要求の最大数を指定します。 アプリケーション プールが処理する要求の数を制限しない場合は、数字 0 を入力します。
特定時刻	アプリケーションがリサイクルされる時刻を指定するには、この設定で省略記号ボタン (...) をクリックします。[特定時刻にリサイクル] ダイアログ ボックスが開き、ここで特定のローカル時間または時刻を設定できます。 時刻を指定するとき、24 時間形式を使います。たとえば、10:00 P.M. を指定する場合、**22:00** と入力します。 この設定で、InstallShield は指定された時刻の数を示し、時刻の一覧が角かっこに囲まれて表示されます。時刻は垂直線で区切られます。たとえば、次の例では特定のリサイクル時刻が 5:00 A.M. と 11:00 P.M であることを示します。 **2 定義済み[05:00|23:00]**

Web サービス拡張の設定

プロジェクト: *[IIS 構成] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

メモ: *Web サービス拡張は、IIS 6.0 以降がインストールされているマシンでのみ利用可能です。*

IIS 7 があるシステムでは、Web サービス拡張に、[IIS メタベースと IIS 6 構成の互換性] 機能がインストールされている必要があります。詳細については、「InstallShield における IIS サポートのバージョン固有情報」を参照してください。

[IIS 構成] ビューの Web サービス拡張項目を使って、Web サービス拡張を追加および削除します。

エクスプローラーで Web サービス拡張を選択すると、次の設定を構成することができます。

テーブル 171 • Web サービス拡張における全般の設定

設定	説明
説明	新しい Web サービス拡張ファイルの名前を入力します。名前には、文字列 ID が関連付けられているため、翻訳が可能です。[参照] ボタンをクリックして既存の文字列を見つけるか、[文字列の選択] ダイアログ ボックスが起動されたときに新規の文字列を作成します。
コンポーネント	Web サービス拡張が関連付けられているコンポーネントを選択します。省略記号ボタン (...) をクリックして、既存のコンポーネントを参照するか、新しいコンポーネントを作成することもできます。
コンポーネントをパーマネントとしてマークする	コンポーネントの機能がアンインストールされるときに、Web サービス拡張を含むコンポーネントを削除するかどうかを指定します。コンポーネントをパーマネントとしてマークして、アンインストールされないようにするには、[はい] を選択します。
ファイルへの完全パス	Web サービス拡張ファイルの名前を入力します。 ***ヒント•*** *ファイルの指定に、ディレクトリ ID を使用することができます。たとえば、ディレクトリとファイルを* **[INSTALLDIR]file.exe** *と指定することができます。*
グループ ID	Web サービス拡張のグループ ID を入力します。グループ ID を使って、異なる DLL や共通のゲートウェイ インターフェイス (CGI) をクラス分けして、アプリケーションの依存関係を作成できます。 ***ヒント•*** *グループの指定に、ディレクトリ ID を使用することができます。*
許可	Web サービス拡張の状態を [許可] に設定するかどうかを指定します。
UI 削除可能	Web サービス拡張ファイルを IIS マネージャー内から削除可能にするかどうかを指定します。
既存の拡張子を上書きする	既存の拡張子を上書き可能にするかどうかを指定します。

[コンポーネント サービス] ビュー

プロジェクト• *[コンポーネント サービス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*

- *InstallScript MSI*
- *マージ モジュール*

[コンポーネントサービス] ビューでは、インストールパッケージ用の COM+ アプリケーションとコンポーネントを管理することができます。COM+ サーバー アプリケーションとアプリケーション プロキシの両方を管理することができます。COM+ アプリケーション プロキシはサーバー アプリケーション属性のサブセットで構成され、これはクライアン ト コンピューターからアプリケーションが存在するマシンへのリモート アクセスを可能にします。

InstallShield のコンポーネント サービスに関する次の情報をお読みください。

- COM+ システム アプリケーション以外のみプロジェクトに追加することができます。したがって、InstallShield は [コンポーネント サービス] ビュー の [COM+ アプリケーション] エクスプローラーの下に、COM+ システム アプリケーション以外のみを表示します。
- ローカル マシンにインストールされている COM+ アプリケーションのみが、[コンポーネント サービス] ビューに表示され、プロジェクトに追加することが可能です。
- アプリケーションプロキシは COM+ サーバーアプリケーションでのみ利用可能で、ライブラリアプリケーションでは利用できません。

[コンポーネント サービス] ビューで COM+ アプリケーションを選択するとき、以下のタブが表示されます:

- インストール
- 全般
- セキュリティ
- ID
- アクティベーション
- キュー
- 詳細
- ダンプ
- プール/再利用

[インストール] タブの設定は、InstallShield 固有の設定です。その他のタブにある設定は、コントロール パネルにある [コンポーネント サービス] 管理ツールにある設定に類似しています。[インストール] タブにある各設定についての詳細は、「[インストール] タブ」を参照してください。その他のタブにある設定については、「コンポーネント サービス」ヘルプを参照してください。

[インストール] タブ

プロジェクト: *[コンポーネント サービス] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

[インストール] タブは、[コンポーネント サービス] ビューで COM+ アプリケーションを選択したときに表示されるタブの 1 つです。

テーブル 172 • [インストール] タブの設定

設定	説明
サーバー	選択した COM+ アプリケーションを他のマシンにインストールする場合、このチェック ボックスを選択します。
インストール先 (サーバー)	COM+ アプリケーションのデフォルトのインストール先は、 [ProgramFilesFolder]COMPlus Applications¥{UID} です。 COM+ ファイルを別の場所にインストールする場合、ターゲット先を選択します。指定するインストール先が一覧にない場合、**[ディレクトリ エントリを参照、作成、または変更しま]** オプションを選択します。
条件 (サーバー)	必要に応じて、COM+ サーバー アプリケーションの条件を指定します。
ロールと共にユーザー ID をインストールする	選択した COM+ アプリケーションをローカル マシンの COM+ アプリケーションについて構成したユーザー ID とロールと共にインストールする場合、このチェック ボックスを選択します。
ビルド時にクライアントマシンから COM+ 設定をリフレッシュする	[コンポーネント サービス] ビューでは、ローカル マシンのコンポーネント サービスで使用できる COM+ の設定が表示されます。 プロジェクトで表示されている COM+ の設定をローカル マシンのコンポーネント サービスで提供されている設定で更新する場合、このチェック ボックスを選択します。リリースをビルドしたとき、そのつど設定がリフレッシュされます。
InstallFinalize アクションの後でインストールする	選択した COM+ アプリケーションに、グローバル アセンブリ キャッシュ (GAC) にインストールする必要がある .NET アセンブリが含まれている場合、このチェック ボックスを選択します。このチェック ボックスを選択すると、ISComponentServiceFinalize アクションは選択した COM+ アプリケーションを InstallFinalize アクションの後でインストールします。Windows Installer は、InstallFinalize が実行されるまでスクリプト内のセッションで加えられた変更をコミットしません。
プロキシ	選択した COM+ アプリケーションをアプリケーション プロキシとしてインストールする場合、このチェック ボックスを選択します。COM+ アプリケーション プロキシはサーバー アプリケーション属性のサブセットで構成され、これはクライアン ト コンピューターからアプリケーションが存在するマシンへのリモート アクセスを可能にします。
インストール先 (プロキシ)	COM+ アプリケーションのデフォルトのインストール先は、 [ProgramFilesFolder]COMPlus Applications¥{UID} です。 COM+ ファイルを別の場所にインストールする場合、ターゲット先を選択します。指定するインストール先が一覧にない場合、**[ディレクトリ エントリを参照、作成、または変更しま]** オプションを選択します。

テーブル 172 • [インストール] タブの設定 (続き.)

設定	説明
条件 (プロキシ)	必要に応じて、プロキシ サーバー サポートの条件を指定します。
リモート サーバー名	アプリケーションが常駐するリモート サーバー コンピューターの名前を指定します。正しい名前を入力するか、またはデフォルトの [REMOTESERVERNAME] プロパティを利用することが可能です。このデフォルト値はインストールで COM+ アプリケーションの [プロキシ] チェック ボックスを選択したときに自動的に作成されます。 ***メモ:*** REMOTESERVERNAME *プロパティのデフォルト値は、InstallShieldで COM+ アプリケーションを インストール プロジェクトへ追加するために使用されるマシンの名前です。*[REMOTESERVERNAME] *プロパティの値を変更するには、[プロパティ マネージャー] ビューを使用します。* エンドユーザーがリモート サーバーを指定できるようにしたい場合は、[ダイアログ] ビューのエンドユーザー ダイアログにリモート サーバー編集フィールド コントロールを追加します。このコントロールの [プロパティ] 値に **REMOTESERVERNAME** を設定します。
クライアントマシンに配布された COM を有効にする	必要に応じて、このチェック ボックスを選択します。DCOM (分散コンポーネント オブジェクト モデル) がすべてのクライアント マシン上で既に有効であることが分かっていて、クライアント マシン上での管理者権限を持たないことが分かっている場合、このチェック ボックスをクリアします。 このチェック ボックスを選択した場合、インストール時に Y が HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Ole レジストリキーの EnableDCOM エントリに書き込まれて、DCOM が有効となります。 ***メモ:*** *エンドユーザーは [コントロール パネル] にある [コンポーネント サービス] 管理ツールを利用して、マシン上で DCOM を有効または無効にすることができます。ただし、そのマシンの DCOM が無効になっている場合、アプリケーション プロキシはクライアント マシン上では動作しません。そのため、**[クライアント マシン上で 分散 COM を有効にする]** チェック ボックスの選択をお勧めします。* *エンドユーザーがアプリケーション プロキシ サポートをアンインストールする場合、EnableDCOM レジストリ エントリは、インストール プロセスでこのレジストリ エントリがターゲット マシンで Y に変わったとしても、変更はされません。*
機能	選択した COM+ アプリケーションを含める機能を選択します。COM+ アプリケーションを新しい機能に追加するには、まず [機能] ビューで機能を作成してから、この一覧でそのチェック ボックスを選択します。

[SQL スクリプト] ビュー

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

SQL スクリプトビューは、すべての SQL スクリプトをサーバー接続および設定に応じて管理並びに編成するための拠点です。このビューでは、Microsoft SQL Server、Microsoft Windows Azure、MySQL、および Oracle がサポートされています。[SQL スクリプト] ビューから、SQL サーバーを構成するための、以下の主要な関数のほとんどを実行することができます。一部のサーバーの種類には制限があります。

- SQL サーバーへ接続する。
- カタログスキーマおよび/またはデータをインポートする。
- SQL スクリプトとコンポーネントを関連付ける。
- 必要な SQL サーバー/スクリプトプロパティ (サーバー名、データベース名、認証メソッドなど) を設定する。
- インストールまたはアンインストール中に実行するための SQL スクリプトの設定する。
- SQL スクリプトを編集する。
- Windows Azure SQL、SQL Server、MySQL、または Oracle の特定バージョンを要求またはターゲットする、もしくはその両方。
- SQL スクリプトのテキスト置換を定義する。
- Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer でスクリプトを開く。

メモ: *データベースのインポート機能は、Microsoft SQL Server Database で利用できます。Oracle ユーザーの方は、Oracle データベース ユーティリティの Oracle Web ページで、InstallShield と共に利用可能なユーティリティについての情報をご覧いただけます。*

使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:

1. *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
2. *Microsoft SQL Server 2005 Management Studio (*SqlWb.exe*)*
3. *Microsoft SQL Server 2005 Management Studio Express (*ssmsee.exe*)*
4. *Microsoft SQL Server 2000 Query Analyzer (*isqlw.exe*)*

SQL 接続

SQL スクリプト ビューでは、スクリプトは接続によって編成されます。これは接続が確立されるまでスクリプトがサーバー上で実行することができないためです。さらに対応するスクリプトと接続をグループに分けることで、別のスクリプトと接続設定を共有することが可能になります。

[SQL スクリプト] ビューで SQL 接続を作成または選択すると、次のタブが表示されます:

- [全般] タブ
- [要件] タブ
- [詳細] タブ

[全般] タブ

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[SQL スクリプト] ビュー内の SQL 接続の [全般] タブで、以下の設定を構成することができます。

テーブル 173 • SQL 接続の [全般タブ] の設定

設定	説明
カタログ名	 ***メモ:*** *Oracle データベースの設定に基づいて新しいアプリケーション カタログを作成することは、InstallShield で SQL Server の設定に基づいて新しいデータベースを作成することに相当します。ただし、Oracle では用語が若干異なります。Oracle のカタログは、すべての新しいアプリケーション カタログに使用されているスキーマに相当します。したがって、ドキュメントの SQL Server と Oracle のサポートの違いを理解するとき、「カタログ」と「データベース」の意味的な違いに注意してください。* インストール中にそのプログラムへの接続を作成する SQL カタログの名前を入力します。 この設定を空白のままにすると、実行時の接続テスト段階でインストールはデフォルトのカタログへ接続しようとします。指定されたカタログが実行時に見つからなかった場合も、インストールは接続の検証時に指定されたカタログを作成します。これは、InstallShield 上で [存在しない場合カタログを作成する] オプションがデフォルトとして選択されているためです。

テーブル 173 • SQL 接続の [全般タブ] の設定 (続き.)

設定	説明
存在しない時、カタログを作成する	デフォルトではこのオプションが選択されています。このオプションが選択されている場合、インストールは実行時に SQLLogin ダイアログで接続が検証されるときに、指定されたカタログを作成します。 **"存在しない場合カタログを作成する"** オプションが選択されている場合、異なる SQL コマンドがさまざまなサポートされているサーバー タイプに発行されます。 Microsoft SQL Server、Microsoft Windows Azure および MySQL でこのオプションを選択した場合、次のコマンドが発行されます: CREATE DATABASE *CatalogName* *CatalogName* は、[SQL] スクリプト ビュー内の接続の [全般] タブにある [カタログ名] ボックスに指定された値です。 Oracle で **"存在しない場合カタログを作成する"** オプションを選択すると、次のコマンドが発行されます: CREATE USER CatalogName IDENTIFIED BY CatalogName DEFAULT TABLESPACE USERS QUOTA UNLIMITED on USERSGRANT CONNECT TO CatalogNameGRANT DBA TO CatalogNameALTER USER CatalogName DEFAULT ROLE ALL つまり、Oracle ではデフォルトでスキーマ名が選択されたカタログのユーザー名とパスワードとして使用されます。

テーブル 173 • SQL 接続の [全般タブ] の設定 (続き.)

設定	説明
存在しない時、カタログを作成する (続き)	カタログの作成に **"存在しない場合カタログを作成する"** オプションを利用しない場合、カスタマイズしたスクリプトを実行してカタログを作成することができます。カスタム スクリプトを実行するとき、[SQL スクリプト] ビュー内のスクリプトの [実行] タブを使って、ログイン中にスクリプトの実行をスケジュールすることができます。 カスタム スクリプトの実行についての詳細は、以下の手順を参照してください: • カスタマイズした SQL スクリプトを実行して SQL Server データベースを作成するサンプル プロジェクトを作成する • カスタマイズした SQL スクリプトを実行して Oracle スキーマを作成するサンプル インストールを作成する ***プロジェクト:*** *インストールのデザイン中に* ***"存在しない場合カタログを作成する"*** *オプションがクリアされている場合で、実行時にカタログが見つからなかったとき、次のプロジェクトの種類の実行時のインストール動作は異なります。* • *基本の MSI、DIM、および InstallScript MSI プロジェクトの場合 - インストールは実行時に SQLLogin ダイアログのデフォルト カタログに接続して接続テストを行います。これによってインストールは InstallExecuteSequence 中にカスタムコードを実行することができます。ただし、プロジェクトの SQL スクリプトが実行されるとき、カタログが必ず存在している必要があります。ランタイムは指定されたカタログとの接続を再確立してスクリプトを実行します。この時点でカタログが存在していないと、接続は失敗します。* • *InstallScript プロジェクトの場合 – ランタイムは、一度接続が確立するとインストールが終わるまでそれを保持するため、カタログは SQLLogin ダイアログで必ず存在している必要があります。この場合には、SQLLogin ダイアログ が表示される前に、カスタム InstallScript コードを実行してカタログを作成することができます。*

テーブル 173 • SQL 接続の [全般タブ] の設定 (続き.)

設定	説明

デフォルトの ターゲット サーバー名	Microsoft SQL Server および MySQL では、ここでターゲット SQL Server のマシン名を入力することができます。この設定はオプションです。
	
	Microsoft Windows SQL Azure の場合、完全修飾サーバー名を次のフォーマットで入力します:
	
	tcp:*ServerName*.database.windows.net
	
	参考例:
	
	tcp:wbzdh64drd.database.windows.net
	
	Oracle 11g クライアントソフトウェアがインストールされている Oracle クライアントマシンでは、次の形式で SQL 接続 URL 文字列を入力しなくてはなりません:
	
	//host: [ポート] [/サービス名]
	
	特定の Oracle リモート マシンへ接続するための入力例は次の通りです。
	
	//sch01jsmithrxp.installshield.com:1521/ORCL
	
	クライアントマシンで tnsnames.ora を構成する場合、ローカルネットサービス名を指定することもできます。詳細については、「Oracle サポート Web サイト」を参照して下さい。

接続方法	指定されたカタログの接続に使用する認証の種類を選択します。SQL Server 認証の場合、ターゲット サーバーへのログインおよびパスワード情報を入力します。
コメント	この接続に関するコメントを入力します。これらのコメントは、エンドユーザーには表示されません。

[要件] タブ

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[要件] タブで、データベース サーバーを選択して、実行時にインストールがターゲットにするサーバーの種類のバージョン要件を指定することができます。特定のデータベース サーバーに関連する新規バージョン要件を追加したり、既存のバージョン要件を編集または削除するオプションがあります。

タスク ***データベース サーバーを選択して、実行時に、インストールがターゲットするサーバータイプのバージョン要件を指定するには、以下の手順に従います:***

1. **[データベース サーバー]** リストで、適切なデータベース サーバーのチェックボックスを選択し、そのサーバー タイプをハイライトします。

   データベース サーバーを 1 つ以上選択すると、インストールの実行時に、接続が確立されてから、テーブルに表示されたサーバー タイプの順番に基づいて、選択されたサーバータイプについて指定要件のチェックが実行されます。つまり、インストールで、リストの最初のデータベース サーバーの要件を確認したあとで一致が検出された場合、接続およびチェックはテーブルの中のその後の選択されたどのデータベース タイプについては行われません。接続に関連付けられた SQL スクリプトは、検証された最初のデータベース サーバーにインストールされます。

2. **[追加]** ボタンをクリックします。ハイライトしたデータベース タイプに対して [新しいバージョン要件] ダイアログ ボックスが開きます。

3. インストールの実行時に、ターゲットするサーバータイプのバージョン要件を指定します。

4. **[OK]** をクリックします。

新しいデータベース サーバー エントリと適切なバージョン情報が [データベース サーバー] リストの横にあるテーブルに追加されます。

タスク ***データベース サーバー テーブルのエントリを編集または削除するには、以下の手順に従います:***

1. インストールの実行時にターゲットにするデータベース サーバーを選択します。

2. 削除または編集するバージョン要件を選択します。

3. **[削除]** または **[編集]** をクリックします。**[編集]** をクリックすると、適切なデータベース サーバー タイプに対する [バージョン要件の編集] ダイアログ ボックスが開きます。

4. 設定を更新して、**[OK]** をクリックします。

メモ: *テーブルのバージョン要件セクションでのアイテムの複数選択は、[削除] 操作にのみ適用します。*

最小要件が満たされなかったときも、インストールの続行を許可する

最小データベース サーバーの要件が満たされなかったときも、インストールを続行できるようにする場合、このチェック ボックスを選択します。

このチェック ボックスが選択されている状態で、最小要件が満たされなかった場合、インストールは SQL 接続とそのすべての SQL スクリプトをスキップして、残りのインストールを完了します。

このチェック ボックスをクリアしていて最小要件が満たされない場合、エンド ユーザーはインストールを続行することができません。

このチェック ボックスはデフォルトでクリアになっています。

Microsoft SQL Server Desktop Engine/SQL Server Express へのインストールを許可する

許可する場合、このチェック ボックスを選択します。SQL Server Desktop Engine および SQL Server Express へのインストールを禁止する場合、このチェック ボックスをクリアします。

データベース サーバーの SQL ドライバーを選択する

InstallShield 2019 R3 以降では、使用するデータベース サーバー用のドライバー一覧から、異なるプロバイダーを選択することができます。接続文字列のプロパティ (ISSQLDBMetaData テーブル内) は、選択されたプロバイダーに基づいて自動的に更新されます。

デフォルトで、新しい SQL 接続に Microsoft OLE DB Driver for SQL Server (MSOLEDBSQL) が表示されます。MSOLEDBSQL が選択されると、次の前提条件がプロジェクト構成に自動的に追加されます:

- Microsoft OLE DB Driver for SQL Server 18.2.2.0 (x86).prq
- Microsoft OLE DB Driver for SQL Server 18.2.2.0 (x64).prq

[再配布可能ファイル] ビューを選択して、必要な前提条件をダウンロードまたは構成します。

[詳細] タブ

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[詳細] タブに含まれる設定は、InstallShield で提供されているデフォルトの機能の拡張およびカスタマイズが必要な上級開発者による利用をお勧めします。

テーブル 174 • SQL 接続の [詳細] タブの設定

設定	プロジェクトの種類	説明
コマンド タイムアウト(秒):	基本の MSI、DIM、InstallScript、InstallScript MSI	コマンド タイムアウトの秒数を指定します。デフォルトは 30 秒です。この値に有効な範囲は 0 から 2147483647 です。値 0 は制限がないことを意味します。コマンドを完了せずにこの期間が経過すると、エラーが発生して ADO がコマンドをキャンセルします。

テーブル 174 • SQL 接続の [詳細] タブの設定 (続き.)

設定	プロジェクトの種類	説明
バッチ区切り記号	基本の MSI、DIM、InstallScript、InstallScript MSI	選択された接続に適切なバッチ区切り記号を指定します。バッチ区切り記号は、この接続に関連付けられたすべての SQL スクリプトで使用されます。 デフォルト値は Microsoft 製品でも利用されている **GO**です。Oracle ユーティリティでは、スラッシュ (/) がデフォルト値として使われています。 バッチ区切り記号は、InstallShield の他に osql、isql、および Microsoft SQL Server Management Studio などの SQL ユーティリティで認識されます。ランタイムはバッチ区切り記号をシグナルとして解釈し、SQL ステートメントの現在のバッチをデータベース サーバーへ送ります。 ***メモ:*** *セミコロン (;) は、それのみで別の新しい行に配置しなくても良い唯一のバッチ セパレーターです。そのたのバッチ セパレーターは、個別の行に配置しなくてはなりません。たとえば、スラッシュをバッチ セパレーターとして指定して、次のような SQL スクリプトがある場合、***some statement*** *がます最初のバッチとして SQL サーバーに送られ、それから* ***another statement*** *が別のバッチとして送られます。* some statement / another statement / *次のようなスクリプトがある場合、インストールでは、行の終わりにあるスラッシュも含め、すべての行がバッチとして送信されます:* some statement/ another statement/

テーブル 174 • SQL 接続の [詳細] タブの設定 (続き.)

設定	プロジェクトの種類	説明
ターゲット サーバー プロパティ名	基本の MSI、DIM、InstallScript MSI	インストールでターゲットするデータベース サーバーのタイプに応じて、次のいずれかを識別するプロパティを選択します: • ターゲット サーバー インスタンスの名前 (Microsoft SQL Server と MySQL の場合) • 完全修飾サーバー名 (Microsoft Windows Azure SQL の場合) • 現在の URL 文字列またはローカル ネット サービス名 (Oracle の場合) デフォルトのプロパティは IS_SQLSERVER_SERVER です。 プロジェクトに複数の SQL 接続が含まれていて、各接続に異なるプロパティを使用したい場合には、それが可能です。詳細については、「新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する」を参照してください。
ターゲットカタログ プロパティ名	基本の MSI、DIM、InstallScript MSI	この設定を使って、インストール中にそのプログラムへの接続を作成する SQL カタログの名前を識別するプロパティを選択できます。デフォルトのプロパティは IS_SQLSERVER_DATABASE です。 プロジェクトに複数の SQL 接続が含まれていて、各接続に異なるプロパティを使用したい場合には、それが可能です。詳細については、「新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する」を参照してください。
認証の種類 プロパティ名	基本の MSI、DIM、InstallScript MSI	この設定を使って、指定されたカタログの接続に使用する認証の種類を識別するプロパティを選択できます。デフォルトのプロパティは IS_SQLSERVER_AUTHENTICATION です。プロパティ値には、次の数値が有効です: • 0 – 現在のユーザーの Windows 認証情報 • 1 – サーバー認証 プロジェクトに複数の SQL 接続が含まれていて、各接続に異なるプロパティを使用したい場合には、それが可能です。詳細については、「新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する」を参照してください。

テーブル 174 • SQL 接続の [詳細] タブの設定 (続き.)

設定	プロジェクトの種類	説明
サーバー認証ログイン ID プロパティ名	基本の MSI、DIM、InstallScript MSI	この設定を使って、サーバー認証に使用するログイン ID を識別するプロパティを選択できます。デフォルトのプロパティは IS_SQLSERVER_USERNAME です。 プロジェクトに複数の SQL 接続が含まれていて、各接続に異なるプロパティを使用したい場合には、それが可能です。詳細については、「新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する」を参照してください。
サーバー認証パスワードプロパティ名	基本の MSI、DIM、InstallScript MSI	この設定を使って、サーバー認証に使用するパスワード を識別するプロパティを選択できます。デフォルトのプロパティは IS_SQLSERVER_PASSWORD です。 プロジェクトに複数の SQL 接続が含まれていて、各接続に異なるプロパティを使用したい場合には、それが可能です。詳細については、「新しい SQL 接続が同じ Windows Installer のプロパティを共有するかどうかを指定する」を参照してください。

プロジェクト: *基本の MSI プロジェクトで SQL 接続の [詳細] タブにある SQL プロパティのいずれかを変更した場合に、[ダイアログ] ビューの SQLLogin ダイアログ内にある対応するプロパティが自動的に更新されることはありません。従って、ダイアログのプロパティを手動で変更して、SQL 接続の [詳細] タブで選択したプロパティと一致させる必要があります。*

SQL スクリプト レベル

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[SQL スクリプト] ビューで SQL スクリプトをクリックすると、次のタブが表示されます:

- [全般] タブ
- [スクリプト] タブ
- [ランタイム] タブ
- [データベース インポート] タブ
- [テキスト置換] タブ

メモ: *使用中のシステムに Microsoft SQL Server Management Studio または Microsoft SQL Server Query Analyzer がインストールされている場合、プロジェクトに追加した新しい SQL スクリプトを開いて、スクリプトをテスト、編集、または構文チェックを行うことができます。これらのツールの 1 つを起動して InstallShield 内でスクリプトを開くには、[SQL スクリプト] ビューでスクリプトを右クリックしてから [Microsoft SQL Server Management Studio でスクリプトを開く] をクリックします。InstallShield が次のツールから 1 つを検索して、最初に検出されたツールを起動します:*

1. *Microsoft SQL Server 2008 Management Studio (Express;* ssms.exe *を含む任意のエディション)*
2. *Microsoft SQL Server 2005 Management Studio (*SqlWb.exe*)*
3. *Microsoft SQL Server 2005 Management Studio Express (*ssmsee.exe*)*
4. *Microsoft SQL Server 2000 Query Analyzer (*isqlw.exe*)*

[全般] タブ

プロジェクト: *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[SQL スクリプト] ビュー内の SQL スクリプトの [全般] タブで、以下の設定を構成することができます。

テーブル 175 • SQL スクリプトの [全般] タブの設定

設定	説明
SQL スクリプト ファイル名	この設定の横にある省略記号ボタン (...) をクリックして、インストールが実行時に実行する SQL スクリプトを参照します。
SQL Script が属する機能を選択してください	SQL スクリプトを含める 1 つ以上の機能のチェック ボックスを選択またはクリアします。 ***プロジェクト:*** *DIM プロジェクトの場合 - このビューで SQL スクリプトを選択すると、このペインの [マージされた機能] チェック ボックスは選択された状態で無効となります。DIM をインストール プロジェクトに追加する場合、その DIM を含める機能を指定します。*
コンポーネント名	個の設定は、SQL スクリプトを含むコンポーネントの名前を表示します。[コンポーネント] ビューに移動し、この SQL スクリプトを含むコンポーネントを探して、そのハイパーリンクをクリックします。

テーブル 175 • SQL スクリプトの [全般] タブの設定 (続き.)

設定	説明
スキーマ バージョン	スクリプトバージョンを有効にするため、バージョン番号を指定します。この設定についての詳細は、「SQL スクリプト ファイルのバージョン番号を指定する」を参照してください。 ***ヒント•*** *"スキーマ バージョン" 設定で番号を指定して、InstallShield によって、ターゲット データベースにそのスキーマ バージョン番号を格納するための* InstallShield *テーブルが追加されると、データは、アンインストールが実行されたときも自動的に削除されません。従って、インストールが変更内容をロールバックできるようにするためには、アンインストール時にテーブルを削除するカスタム スクリプトを作成して、*InstallShield *テーブルを削除する必要があります。*
コメント	この SQL スクリプトについてのコメントを入力します。これらのコメントは、エンドユーザーには表示されません。

[スクリプト] タブ

プロジェクト• *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[スクリプト] タブは上級開発者向けです。手順に従って既存データベースからスクリプトを作成するには、データベースのインポート ウィザードをご利用ください。

メモ• *現在、この機能は Microsoft SQL Server Database で使用できます。*

[ランタイム] タブ

プロジェクト• *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[SQL スクリプト] ビュー内の SQL スクリプトの [ランタイム] タブで、以下の設定を構成することができます。

テーブル 176 • SQL スクリプトの [ランタイム] タブの設定

設定	プロジェクトの種類	説明
スクリプト実行	基本の MSI、DIM、InstallScript、InstallScript MSI	インストールが SQL スクリプトを実行するタイミングを指定するには、適切なチェック ボックスを選択します。選択可能なオプションは以下のとおりです: • **インストール中にスクリプトを実行** – このオプションを利用すると、インストール中にカスタム スクリプトを実行できます。また、各スクリプトは機能と関連付けられているため、このオプションはコンポーネントの状態にも関連付けられています。したがって、基本の MSI、DIM、および InstallScript MSI プロジェクトでは、このタブにある [スクリプト条件] 領域で、この設定と関わりのある条件ステートメントを指定することができます。 基本の MSI、DIM、および InstallScript MSI プロジェクトの場合、ISSQLServerInstall アクションがこのオプションに関連付けられています。 InstallScript プロジェクトの場合、InstallScript 関数 **SQLRTComponentInstall** がこのオプションに関連付けられています。 • **アンインストール中にスクリプトを実行** – このオプションを利用すると、アンインストール中にカスタム スクリプトを実行できます。また、各スクリプトは設計上機能と結ばれているため、このオプションはコンポーネントの状態にも関連付けられています。したがって、このタブにある [スクリプト条件] (基本の MSI および InstallScript MSI プロジェクトでのみ利用可能です) セクションで、設定と関わりのある条件ステートメントを指定することができます。 基本の MSI、DIM、および InstallScript MSI プロジェクトの場合、**ISSQLServerUninstall** アクションがこのオプションに関連付けられています。 InstallScript プロジェクトの場合、InstallScript 関数 **SQLRTComponentUninstall** がこのオプションに関連付けられています。

テーブル 176 • SQL スクリプトの [ランタイム] タブの設定 (続き.)

設定	プロジェクトの種類	説明
スクリプト実行 (続き)		• ロールバック中にスクリプトを実行—このオプションを利用すると、ロールバック中にカスタム スクリプトを実行できます。InstallShield は、自動的に変更をロールバックしません。[ロールバック中にスクリプトを実行] オプションを選択した場合、ロールバックを許可するカスタムスクリプトを実行する必要があります。 基本の MSI、DIM、および InstallScript MSI プロジェクトの場合、ISSQLServerRollback アクションがこのオプションに関連付けられています。 • ログイン中にスクリプトを実行 — このオプションを利用すると、ログイン中にスクリプトを実行できます。このオプションには、いくつかの制限があります。たとえば、実行時にエンド ユーザーが SQLLogin ダイアログで [次へ] ボタンをクリックした後、スクリプトの変更を元に戻すことはできません。エンド ユーザーが ReadyToInstall ダイアログで [インストール] ボタンをクリックする前にインストールをキャンセルしようとすると、スクリプトの変更はロールバックしません。 また、スクリプトは認証情報および要件が検証された後、およびカタログへの接続が作成される前に実行されます。したがって、インストールのデザイン中にスクリプトに設定したスキーマバージョン要件は実行時に実行されません。 基本の MSI および DIM プロジェクトの場合、ISSQLServerValidate アクションがこのオプションに関連付けられています。 InstallScript MSI プロジェクトの場合、InstallScript 関数 **SQLRTServerValidate** がこのオプションに関連付けられています。 InstallScript プロジェクトの場合、InstallScript 関数 **SQLRTConnect** がこのオプションに関連付けられています。
スクリプトエラー処理	基本の MSI、DIM、InstallScript、InstallScript MSI	実行時に SQL スクリプト エラーを処理する方法を指定します。選択可能なオプションは以下のとおりです: • エラー時に、次のスクリプトへ移動する • エラー時に、次のステートメントへ移動する • エラー時に、インストールを中止する

テーブル 176 • SQL スクリプトの [ランタイム] タブの設定 (続き.)

設定	プロジェクトの種類	説明
スクリプト条件	基本の MSI、DIM、InstallScript MSI	インストールまたはアンインストール中、SQL スクリプトが実行される前に Windows Installer が評価する条件を指定する場合、このチェックボックスを選択して、テキスト ボックスに条件ステートメントを入力します。条件が false 評価された場合、SQL スクリプトは実行されません。 ステートメントを手動で入力する代わりに、省略記号ボタン (...) をクリックして [条件ビルダー] ダイアログ ボックスを起動すると、条件の作成処理を簡素化することができます。

[データベースのインポート] タブ

プロジェクト• *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ヒント• *データベースのインポート機能は、現時点では Microsoft SQL Server でのみ利用することができます。Oracle ユーザーの方は、Oracle データベース ユーティリティの Oracle Web ページで、InstallShield と共に利用可能なユーティリティについての情報をご覧いただけます。*

このタブからデータベースインポート ウィザードを実行して、インポート設定を確認および変更することができます。ウィザードは、既存する Microsoft SQL Database のすべてまたはその一部を作り直すための SQL スクリプトの生成プロセスを順を追って説明します。ウィザードが終了したら、このタブにある [生成] ボタンをクリックしてスクリプトを更新します。

[ビルド時に SQL スクリプトを再生成する] オプションを使って、リリースをビルドする度にスクリプトを再生成するかどうかを指定します。リリースをビルドする度にスクリプトを再生成すると、ビルド処理の速度が低下します。

[テキスト置換] タブ

プロジェクト• *[SQL スクリプト] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

[テキスト置換] タブを使って、実行時に SQL スクリプトで置換する文字列を指定します。スクリプト構文エラーを避けるため、置換テキストが一意の場合のみ、テキスト置換が可能です。実行時に、インストールは指定されたパラメーターに従ってテキストを置換します。

新しい検索/置換エントリを追加するには、[追加] ボタンをクリックします。既存の検索/置換エントリを変更するには、そのエントリを選択してから [編集] ボタンをクリックします。どちらの場合も、検索/置換のパラメーターを指定できる [検索/置換] ダイアログ ボックス が開きます。

既存の検索/置換エントリを削除するには、そのエントリを選択してから [削除] ボタンをクリックします。

[動作とロジック] ビュー

プロジェクト: *[動作とロジック] ビューは、次のプロジェクトの種類で使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

カスタム アクションの追加、アクションとダイアログのシーケンス、サポート ファイルの追加、InstallScript の使用、ターゲット システム上での必須ファイル、フォルダーおよび他の要素の検索、Windows Installer プロパティの構成を行うことにより、インストールで必要なカスタム機能をデザインすることができます。

InstallScript

プロジェクト: *[InstallScript] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallScript は、C プログラミング言語に似たインストール オーサリング言語です。InstallScript ビューは、スクリプトの作成および編集を行うことができるスクリプト エディター ペインを提供します。

カスタム アクションとシーケンス

プロジェクト: *[カスタム アクションとシーケンス] ビューは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

シーケンスは、ファイル転送からユーザー インタフェースの表示までインストール中に実行されるすべてのアクションを監督します。[カスタム アクションとシーケンス] ビューを使用して、プロジェクトのアクションとダイアログのシーケンスを定義することができます。

Windows Installer が、インストール プログラムで必要な機能を直接サポートしていない場合、[カスタム アクションとシーケンス] ビューでカスタム アクションを使ってインストール プログラムを拡張できます。

カスタム アクション

プロジェクト: *[カスタム アクション] ビューは、次のプロジェクトの種類で使用できます。*

- *DIM*
- *マージ モジュール*
- *MSM データベース*

マージ モジュールでは、カスタム アクションの使用がサポートされていますが、マージ モジュール プロジェクト専用のビューでは、シーケンスを定義することができません。

ヒント: *ダイレクト エディターで* ModuleInstallExecuteSequence *テーブルを変更して、マージ モジュール内のカスタム アクションの起動を制御することができます。インストール プロジェクトにマージ モジュールを追加する際、マージ モジュールの中に含められたすべてのカスタム アクションおよびダイアログは [カスタム アクションとシーケンス] ビューを通してインストールのシーケンスに挿入することができます。*

サポート/ビルボード

プロジェクト: *[サポート ファイル] ビューは、次のプロジェクトの種類で使用できます。*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

[サポート ファイル/ビルボード] ビューは、次のプロジェクトの種類で使用できます。

- *InstallScript*
- *InstallScript MSI*

サポート ファイル

[サポート ファイル] ビューを使って、*インストール プロセス中にのみ*インストールが必要とするサポート ファイルを追加、ソートおよび削除することができます。

ビルボード

[サポート ファイル/ビルボード] ビューの [ビルボード] 領域で、プロジェクトにビルボードを追加することができます。ビルボードは、インストールの実行中に特定の時間だけ表示されるイメージです。ビルボードは、インストール中の製品に関する情報を表示したり、または単にインストールが終了するまでエンドユーザーを楽しませるために使用できます。

システム検索

プロジェクト: *[システム検索] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

また、[システム検索] ビューを利用して、インストールの前に、ターゲット システム上にある特定のファイル、フォルダー、レジストリキー、または .ini 値を検索することもできます。

プロパティ マネージャー

プロジェクト: *[プロパティ マネージャー] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、[プロパティ マネージャー] ビューを使って InstallShield インターフェイス内部から Property テーブルを編集できます。Windows Installer のプロパティからは、ユーザーの名前や会社名などのマシン固有の多くの変数にアクセスできます。

アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、[プロジェクト マネージャー] ビューで、実行時に、アドバンスト UI またはスイート/アドバンスト UI インストールで使用できるアドバンスト UI またはスイート/アドバンスト UI のプロパティを設定できます。

[InstallScript] ビュー

プロジェクト: *[InstallScript] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

InstallScript ビューでは、InstallScript 言語を使用してセットアップ スクリプトをカスタマイズすることができます。

ファイル、関数、プロパティ、およびメソッド フォルダー

InstallScript ビューの中央のペインにある InstallScript エクスプローラーには、次のフォルダーが含まれています:

テーブル 177 · InstallScript ビューのフォルダー

フォルダー	プロジェクトの種類	説明
ファイル	基本の MSI、マージ モジュール、InstallScript MSI、InstallScript、InstallScript オブジェクト、スイート/アドバンスト UI	[ファイル] フォルダーには、すべてのスクリプト (.rul) ファイルが表示されます。スクリプト ファイルをクリックすると、スクリプト エディター上にファイルのコンテンツが表示されます。 セットアップ スクリプトの名前は、必ず Setup.rul とします。異なる名前のファイルを含めることは可能ですが、そのファイルを #include プリプロセッサ ステートメントを使って Setup.rul に含める必要があります。
関数	基本の MSI、マージ モジュール、InstallScript MSI、InstallScript、InstallScript オブジェクト、スイート/アドバンスト UI	関数フォルダーには、プロジェクトに含まれるすべてのスクリプト ファイルの InstallScript 関数のリストが含まれます。関数をクリックすると、スクリプト エディターにその関数が表示されます。 ***メモ:*** *スクリプト エディタでローカル変数のオートコンプリート機能が無効になっている場合、関数フォルダーはスクリプト ファイルにある関数を一切リストしません。ローカル変数のオートコンプリート機能を有効にする方法については、「スクリプト エディターでオートコンプリート機能を有効または無効にする」を参照してください。*

テーブル 177 • InstallScript ビューのフォルダー (続き.)

フォルダー	プロジェクトの種類	説明
プロパティ	InstallScript、InstallScript オブジェクト	プロパティ フォルダーには、プロジェクトに含まれるすべてのスクリプト ファイルの InstallScript プロパティのリストが含まれます。プロパティをクリックすると、スクリプト エディターでそのプロパティの宣言が表示されます。(プロパティのプロシージャは [関数] フォルダーに一覧表示されます。) ***メモ•*** *スクリプト エディタでローカル変数のオートコンプリート機能が無効になっている場合、プロパティ フォルダーはスクリプト ファイルにあるプロパティを一切リストしません。ローカル変数のオートコンプリート機能を有効にする方法については、「スクリプト エディターでオートコンプリート機能を有効または無効にする」を参照してください。*
メソッド	InstallScript オブジェクト	メソッド フォルダーには、プロジェクトに含まれるすべてのスクリプト ファイルの InstallScript メソッドのリストが含まれます。メソッドをクリックすると、スクリプト エディターでそのメソッドが表示されます。 ***メモ•*** *スクリプト エディタでローカル変数のオートコンプリート機能が無効になっている場合、メソッド フォルダーはスクリプト ファイルにあるメソッドを一切リストしません。ローカル変数のオートコンプリート機能を有効にする方法については、「スクリプト エディターでオートコンプリート機能を有効または無効にする」を参照してください。*

スクリプト エディター

InstallScript ビューで任意のフォルダーの下にあるアイテムをクリックすると、InstallShield インターフェイスの右側のペインにスクリプト エディターが表示されます。詳細については、「[InstallScript] ビューのスクリプト エディター」を参照してください。

[InstallScript] ビューのスクリプト エディター

プロジェクト• *スクリプト エディターは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

- *スイート/アドバンスト UI*

スクリプト エディターには、選択した InstallScript ファイルのテキストが表示されます。編集およびツールメニューのコマンドを使用してスクリプトを変更または編集できます。

スクリプト エディターのスクリプト ツールバー

プロジェクト: *スクリプト ツールバーは、次のプロジェクトの種類で使用できます:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

基本の MSI およびマージ モジュール プロジェクトは、イベント ハンドラーの代わりにカスタム アクションを使って InstallScript コードを実行します。スイート/アドバンスト UI プロジェクトは、イベント ハンドラーの代わりにアクションを使って InstallScript コードを実行します。

スクリプト ツールバーでは、InstallScript ビューのスクリプト エディターの上部に表示されます。ツールバーには、イベント リスト ボックスおよびイベント カテゴリ リスト ボックスが含まれています。スクリプト ツールバーを使うと、スクリプト ファイルのインストール イベント ハンドラー関数ブロックを貼り付けることができます。スクリプト ファイルのイベント ハンドラーは、インストールイベントに関連付けられているデフォルトのアクションを上書きします。イベント ハンドラーコードを修正して、インストール中に実行されるアクションを変更できます。

イベントカテゴリ (左のリストボックス)

スクリプト ファイルにイベント ハンドラー関数ブロックを貼り付けるインストールイベントが入ったカテゴリを選択します。選択されたカテゴリ中のイベントが、[イベント] リストボックスに表示されます。

イベント (右のリストボックス)

スクリプト ファイルにイベント ハンドラー関数ブロックを貼り付けるインストールイベントを選択します。機能イベントを選択すると、そのイベント ハンドラーが FeatureEvents.rul に貼り付けられます。このファイルがプロジェクトに存在しない場合、機能イベントを選択したとき、ファイルが自動的に作成されます。別の種類のインストールイベントを選択すると、そのイベント ハンドラーが Setup.rul に貼り付けられます。

FeatureEvents.rul でデフォルトの機能のイベント ハンドラーコードを変更した場合、Setup.rul に次のステートメントを入れてインストールに変更を含める必要があります。

```
#include "FeatureEvents.rul"
```

[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー)

プロジェクト: *[カスタム アクションとシーケンス] ビューは、次のプロジェクトの種類で使用できます。*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*

- *トランスフォーム*

このビューは、次のプロジェクト タイプでは、[カスタム アクション] ビューという名前で呼ばれます:

- *DIM*
- *マージ モジュール*
- *MSM データベース*

[カスタム アクションとシーケンス] ビューには、3 つの異なる領域 ([カスタム アクション] 領域、[アクション テキスト] 領域と [シーケンス] 領域) があります。[カスタム アクション] ビューには、[カスタム アクション] 領域があります。

カスタム アクション

Microsoft Windows Installer は、直接サポートされていない機能をインストールに柔軟に追加できるよう設計されています。この追加の機能は、カスタム アクションを使用することによって実現できます。

InstallShieldでは、カスタム アクションを使った InstallScript、VBScript、または JavaScript コードの実行、DLL 関数の呼び出し、実行可能ファイルの実行、マネージ アセンブリ内にあるマネージ メソッドの呼び出し、プロパティやディレクトリの設定、エラーのトリガとインストールの中止、PowerShell スクリプトの実行、プロセスの終了、他のインストール パッケージの実行がサポートされています。

[カスタム アクションとシーケンス] ビューの [カスタム アクション] エクスプローラーは、[カスタム アクション] ビューの [カスタム アクション] エクスプローラーと同じ動作をします。カスタム アクションの各タイプについての詳細は、「カスタム アクションの種類」を参照してください。各カスタム アクションの設定に関する詳細は、「カスタム アクションの設定」を参照してください。

アクション テキスト

エンド ユーザーに進行状況を伝えるために、インストールは一般的に、実行中の処理を説明する進行状況ダイアログで説明テキストを表示します。通常、インストールのステータスを表示する進行状況バーも同時に表示されます。標準アクションおよびカスタム アクションが発生すると、そのアクションについてのメッセージが進行状況ダイアログに表示されます。この機能は、実行するのに時間がかかるアクションで、特に役立ちます。実行時にログ ファイルを作成する場合、同じアクション テキストが記録されます。

[カスタム アクションとシーケンス] ビュー (基本の MSI、InstallScript MSI、MSI データベース、およびトランスフォーム プロジェクトの場合) の [アクション テキスト] エクスプローラーを使って、プロジェクト内の任意のアクションにアクション テキストを指定できます。各アクション テキスト設定についての説明は、「アクション テキストの設定」を参照してください。

重要 *• [アクション テキスト] エクスプローラーの下にあるアクション テキスト項目の名前は、プロジェクトに含まれる標準およびカスタム アクションの名前と一致します。カスタム アクションの名前を変更する場合は、そのアクション テキスト項目の名前も変更しなくてはなりません。そうしないと、実行時にアクション テキストが表示されないか、インストールのログ ファイルに記録されません。*

シーケンス

シーケンス は、ファイル転送からユーザー インターフェイスの表示までインストール処理中に実行されるすべてのアクションを管理します (基本の MSI、MSI データベース、およびトランスフォーム プロジェクト)。これらのアクションにはシーケンス上の番号が付与され、小さい番号から順に実行されます。すべてのアクションに数値をマニュアルで割り当てることなく、[カスタム アクションとシーケンス] ビューを使用してシーケンスにアクションを挿入するか、シーケンス タイムラインを編集できます。

プロジェクト: *InstallScript MSI プロジェクトで、インストールのユーザー インターフェイスはスクリプトを通して生成されるため、ダイアログは [カスタム アクションとシーケンス] ビューのシーケンスには挿入されません。*

カスタム アクションまたはカスタム ダイアログを DIM またはマージ モジュールで作成する場合、まず DIM またはモジュールをインストール プロジェクトにインポートしてから、そのインストール プロジェクトの [カスタム アクションとシーケンス] ビューを使って、それをシーケンスに追加する必要があります。

ダイアログ (基本の MSI、MSI データベース、およびトランスフォームプロジェクトの場合) やカスタム アクションを挿入できるシーケンスには、次の 3 つの主なシーケンスがあります。

- [インストール] シーケンス
- [アドバタイズ] シーケンス
- [管理] シーケンス

各シーケンスには異なる役割があります。[インストール] シーケンスは通常のインストール中に実行されます。[アドバタイズ] シーケンスはアプリケーションのインストール時ではなくアドバタイズ時に実行されます。[管理] シーケンスは管理者用インストールで実行されます。

シーケンス、アクション、およびダイアログの変更

[カスタム アクションとシーケンス] ビューでは、ドラッグアンドドロップ編集およびコピー機能がサポートされています。このビューのコンテキスト メニューでは、追加の編集方法が提供されています。

- 新しいカスタム アクションをシーケンスするには、[カスタム アクション] エクスプローラーからそれを、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグします。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。
- ダイアログ、標準アクション、またはカスタム アクションを、シーケンス内の異なる位置へ(または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。
- カスタム アクションをあるシーケンスから別のシーケンスへコピーするには、CTRL を押しながら、そのカスタム アクションをもう 1 つのシーケンスまでドラッグし、直前にくるアクションまたはダイアログの上にドロップします。
- [シーケンス] エクスプローラーでダイアログまたは標準アクションを選択すると、右にある [シーケンス] タブで、シーケンス番号の変更および、関連する条件の追加または変更を行うことができます。
- [シーケンス] エクスプローラーでカスタム アクションを選択すると、右側に次のようなタブが表示されます。
  - [シーケンス] タブ — このタブでは、シーケンス番号の変更および、アクションの条件の追加や変更を行います。
  - [アクション] タブ — このタブでは、カスタム アクションの設定を変更します。
  - [スクリプト] タブ — このタブは、VBScript および JScript カスタム アクションに対して表示されます。このタブにあるスクリプト エディターを使って、VBScript または JScript コードを編集することができます。
- ダイアログのレイアウトまたは動作を編集するには、[シーケンス] エクスプローラーでダイアログを右クリックし、[動作の編集] または [レイアウトの編集] をクリックします。

メモ: *カスタム アクションは、同じシーケンス内で 2 度呼び出すことはできません。これは、カスタム アクションの名前が CustomAction テーブルでキーとして使用されているためです。したがって、カスタム アクションを、そのカスタム アクションを既に含んでいるシーケンスに移動またはコピーすることはできません。*

ダイアログおよび標準アクションは、ドラッグアンドドロップ操作で、異なる種類のシーケンスへ移動することはできません。

基本の MSI プロジェクトの [カスタム アクション] ビューのスクリプトエディター

基本の MSI プロジェクトの **[カスタム アクションとシーケンス > カスタム アクション]**ビューの **[スクリプト]** タブでPowerShell スクリプト エディターを使用できます。

カスタム アクションの種類

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

一部のカスタム アクション タイプは、一部のプロジェクト タイプで使用できませんので注意してください。プロジェクト固有の違いについては、必要に応じて記述されています。

InstallShield では、いくつかの種類のカスタム アクションがサポートされています。

テーブル 178 ・ InstallShield でサポートされているカスタム アクション

アイコン	アクションの種類	プロジェクトの種類	カスタム アクションの動作
	InstallScript	基本の MSI、InstallScript MSI、マージ モジュール	InstallScript コードを実行する。
	EXE	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	実行可能ファイルを起動する。

テーブル 178 • InstallShield でサポートされているカスタム アクション (続き.)

アイコン	アクションの種類	プロジェクトの種類	カスタム アクションの動作
	標準 DLL	基本の MSI、InstallScript MSI	標準 DLLにある関数を呼び出す。
	MSI DLL	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	Windows Installer 専用に書かれた DLL にある関数を呼び出す。 .dll ファイルのエントリ ポイントには、必ず定義済みパラメーターおよび戻り値が必要です。
	.NET Core	基本の MSI、InstallScript MSI	Visual Basic .NET または C# などのマネージ コードで書かれた Microsoft .NET Core ベースのマネージ アセンブリでパブリック メソッドを呼び出します。 詳細については、「.NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す」を参照してください。
	マネージ コード	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	Visual Basic .NET または C# などのマネージ コードで書かれた Microsoft .NET Framework ベースのマネージ アセンブリでパブリック メソッドを呼び出します。 詳細については、「マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す」を参照してください。
	プロパティの設定	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	プロパティを Property テーブルから削除します。 これは、エンドユーザーから情報を取得し、インストールの後半で使用できるようにその情報を格納しておく必要がある場合に便利です。
	ディレクトリの設定	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	実行時に、Directory テーブルのディレクトリを設定する。 たとえば、エンド ユーザーが選択したインストール ディレクトリのサブディレクトリとして一時ディレクトリを作成する場合、このオプションを使って新しい一時ディレクトリを設定して、インストールの後半で使用できます。

テーブル 178 • InstallShield でサポートされているカスタム アクション (続き.)

アイコン	アクションの種類	プロジェクトの種類	カスタム アクションの動作
	ネストされた MSI	基本の MSI、InstallScript MSI、MSI データベース、トランスフォーム	別の .msi パッケージを起動する。これは、ネスト インストールとも呼ばれます。 ***重要*** • *Windows Installer ではネスト インストールの使用を避けるようにしてください。ネスト インストールが含まれている場合エンドユーザーが適切に処理することが困難であるため、時折アプリケーションの誤作動を引き起こします。マイクロソフト社では一般向けにリリースされる製品のインストールについて、ネスト インストールおよびネスト インストール カスタム アクションの使用を避けることを推奨しています。詳細については、Windows Installer ヘルプ ライブラリの「Concurrent Installations」を参照してください。*
	VBScript	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	32 ビットまたは 64 ビットの VBScript コードを実行します。
	JScript	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	32 ビットまたは 64 ビットの JScript コードを実行します。
	エラー	基本の MSI、DIM、InstallScript MSI、マージ モジュール、MSI データベース、MSM データベース、トランスフォーム	指定されたエラー メッセージを表示し、失敗を戻して、インストールを終了します。 例えば、エンド ユーザーが製品の現在のバージョンを将来のメジャー バージョンの上にインストールしようと試みた場合などに、これを処理するエラー カスタム アクションをプロジェクトに追加することができます。詳細については、「現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ」を参照してください。

テーブル 178 • InstallShield でサポートされているカスタム アクション (続き.)

アイコン	アクションの種類	プロジェクトの種類	カスタム アクションの動作
	Kill-Process	基本の MSI、InstallScript MSI、トランスフォーム	実行時にプロセスを強制終了します。詳細については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。
	PowerShell	基本の MSI、InstallScript MSI	PowerShell スクリプトを実行して、インストールの実行時に、システムの構成タスクを実行します。ターゲットについては、このタイプのカスタム アクションに関する他の情報も含めて、「PowerShell カスタム アクションの呼び出し」を参照してください。
	WiseScript	基本の MSI	 ***エディション:*** *この種類のカスタム アクションは、AdminStudio の特定のエディションに含まれる InstallShield のバージョンで使用できます。* WiseScript 実行可能ファイルを実行します。 この種類のアクションには、Windows Installer プロパティを評価して Windows Installer 条件を取得および設定するラッパー MSI DLLが含まれます。

プロジェクト内の異なるカスタム アクション タイプ (即時、遅延、コミット、およびロールバック) を識別できるように、一部のアクションのアイコンには、色分けされた点が追加されます。点は、[カスタム アクションとシーケンス] ビューまたは [カスタム アクション] ビューの [カスタム アクション] エクスプローラーにリストされているアクションのアイコンに表示されます。次のテーブルは、実行可能ファイル カスタム アクションのアイコンを一覧表示します。

テーブル 179 • アイコンからカスタム アクションのスクリプト内実行を判別する

アイコン	説明
	カスタム アクションに点が付いていない場合、その種類のカスタム アクションには "スクリプト内実行" 設定を適用しない、または以下の値の 1 つが選択されていることを意味します: • 即時実行 • 即時実行(ターミナル サービスで使用可能)
	カスタム アクションに青色の点が付いている場合、"スクリプト内実行" 設定に以下の値の 1 つが選択されています: • 遅延実行 • 遅延実行 (ターミナル サービスで使用可能) • システム コンテキストの遅延実行

テーブル 179 • アイコンからカスタム アクションのスクリプト内実行を判別する (続き.)

アイコン	説明
	カスタム アクションに緑色の点が付いている場合、"スクリプト内実行" 設定に以下の値の 1 つが選択されています: • コミット実行 • コミット実行 (ターミナル サービスで使用可能) • システム コンテキストでコミット実行
	カスタム アクションに赤色の点が付いている場合、"スクリプト内実行" 設定に以下の値の 1 つが選択されています: • ロールバック実行 • ロールバック実行 (ターミナル サービスで使用可能) • システム コンテキストでのロールバック実行

様々な機能をサポートするために InstallShield プロジェクトに自動的に追加される各 InstallShield カスタム アクションについての詳細は、「InstallShield カスタム アクション リファレンス」を参照してください。

Windows ロゴ • *Windows ロゴを適用する場合、インストールに含まれるすべてのカスタム アクションは、カスタム アクション作成に関するベスト プラクティス ガイドに従わなくてはなりません。InstallShield 検証スイートおよび完全 MSI 検証 スイートを使って、インストール パッケージがほとんどのカスタム アクション関連のロゴ要件を満たしているかどうかを検証することができます。ただし、一部の要件については検証スイートを使って検証することができません。詳細については、「Windows ロゴ プログラムの要件を満たすカスタム アクションを作成するときのガイドライン」を参照してください。*

カスタム アクションの設定

プロジェクト• *[カスタム アクションとシーケンス] ビューを使って、次のプロジェクトの種類でカスタム アクションの設定を構成できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

以下のプロジェクト タイプでは、[カスタム アクション] ビューを使ってカスタム アクションの設定を構成できます:

- *DIM*
- *マージ モジュール*
- *MSM データベース*

一部のカスタム アクション タイプは、一部のプロジェクト タイプで使用できませんので注意してください。

[カスタム アクションとシーケンス] ビュー (または [カスタム アクション] ビュー) の設定は、次のカテゴリーに分かれています:

- [アクション] 領域の設定
- [シーケンス] 領域の設定

進行状況ダイアログとインストールのログ ファイルにアクション情報を表示する方法については、「アクション テキストを使う」を参照してください。

[アクション] 領域の設定

次のカスタム アクションの設定は、[アクション] 領域で提供されています。

テーブル 180 ・ [アクション] 領域のカスタム アクション設定

設定	説明
関数名	 ***メモ・** この設定は InstallScript カスタム アクションに適用します。* プロジェクトの InstallScript ファイルに含まれている関数リストで、呼び出す InstallScript 関数を選択します。
DLL ファイル名	 ***メモ・** この設定は、Binary テーブルに格納されている標準 DLL カスタム アクション、またはターゲット システムの検索パスにある標準 DLL カスタム アクションに適用します。この設定はまた、Binary テーブルに格納されている MSI DLL カスタム アクションにも適用します。* Binary テーブルの場所の場合、カスタム アクションに使用する DLL ファイルへのパスを入力します。または、Binary テーブルで選択可能な DLL ファイルの一覧から選択するか、省略記号ボタン (...) をクリックして適切な DLL を見つけることもできます。 ターゲット システムの検索パスの場所の場合、カスタム アクションに使用する DLL ファイルの名前を入力します。または、省略記号ボタン (...) をクリックして、適切な DLL ファイルを参照します。カスタム アクションはターゲット マシンの検索パスを検索してこの DLL を見つけるため、ファイル名以外は必要ありません。 カスタム アクション タイプが MSI DLL の場合、必要な Windows Installern の関数署名に準拠している必要があります。詳細については、「Windows Installer DLL ファイル内の関数を呼び出す」を参照してください。
DLL ファイルキー	 ***メモ・** この設定は、製品と共にインストールされる標準および MSI DLL カスタム アクションに適用します。* プロジェクトに含まれている DLL ファイル リストで、カスタム アクションに使用する DLL ファイルを選択します。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
関数の署名	 ***メモ:*** *この設定は標準 DLL カスタム アクションに適用します。* DLL ファイル内のエントリ ポイント関数のパラメーターを指定するには、この設定で省略記号ボタン (...) をクリックします。[関数シグネチャ] ダイアログ ボックスが開き、標準 DLL のエントリ ポイント関数に、関数名、引数、戻り値の型、および戻りプロパティを指定できる様々な設定が表示されます。
関数名	 ***メモ:*** *この設定は MSI DLL カスタム アクションに適用します。* 呼び出す MSI DLL 内の関数名を入力します。
実行可能ファイル名	 ***メモ:*** *この設定は、Binary テーブルに格納されている実行可能ファイル カスタム アクションに適用します。* カスタム アクションに使用する実行可能ファイルへのパスを入力します。または、Binary テーブルで選択可能な実行可能ファイルの一覧から選択するか、省略記号ボタン (...) をクリックして適切なファイルを見つけることもできます。
実行可能ファイルキー	 ***メモ:*** *この設定は、製品と共にインストールされる実行可能ファイル カスタム アクションに適用します。* プロジェクトに含まれている実行可能ファイル リストで、カスタム アクションに使用する実行可能ファイルを選択します。
作業ディレクトリ	 ***メモ:*** *この設定は、ディレクトリを参照するパスに存在する実行可能ファイル カスタム アクションに適用します。* カスタム アクションに使用する実行可能ファイルの作業ディレクトリ名を入力するか、使用できるディレクトリの一覧から選択します。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
ファイル名とコマンドライン	 *メモ• この設定は、ディレクトリを参照するパスに存在する実行可能ファイル カスタム アクションに適用します。* カスタム アクションに使用する実行可能ファイルのパスと名前を入力します。必要に応じて、実行可能ファイルに渡す任意のコマンドライン パラメーターを含めることもできます。長いファイル名は引用符で括ります。例: • "[INSTALLDIR]subdirectory1¥filename.exe" • "[WindowsFolder]Notepad.exe" • "[SUPPORTDIR]¥fileFromSupportFilesView.exe" パスの入力は、実行可能ファイルがオペレーティング システム検索パス (つまり、WindowsFolder または SystemFolder) に既存する場合のみ、オプションとなります。
実行可能ファイルのプロパティ	 *メモ• この設定は、プロパティによって識別されたパスに存在する実行可能ファイル カスタム アクションに適用します。* 実行可能ファイルへの完全パスを識別するプロパティの名前を入力するか、プロジェクト内のプロパティ リストから該当するプロパティを選択します。
コマンド ライン	 *メモ• この設定は、Binary テーブルに格納されている実行可能ファイル カスタム アクション、製品と共にインストールされる実行可能ファイル カスタム アクション、プロパティに識別されたパスに存在する実行可能ファイル カスタム アクションに適用します。* 実行可能ファイルに渡すコマンドライン パラメーターを入力します。長いファイル名は引用符で囲んでください。
アセンブリ ファイル	 *メモ• この設定は、場所が Binary テーブルに設定されているマネージ アセンブリを使用するカスタム アクションに適用します。* マネージ コードまたは .NET コア カスタム アクションに使用する NET アセンブリ ファイルを指定します。**[ファイルを参照]** オプションを選択して、ファイルを指定するか、Binary ファイルに格納される DLL または .exe ファイルの一覧からファイルを選択してください。 ビルド時に、マネージ アセンブリ が C++ Windows Installer ラッパー DLL に追加され、ラッパー DLL が .msi パッケージの Binary テーブルに追加されます。ラッパー DLL には、アセンブリの仲介、ロード、実行を行うために必要な情報が含まれています。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
アセンブリ ファイル キー	 *メモ• この設定は、製品と共にインストールされるマネージ アセンブリを使用するカスタム アクションに適用します。* カスタム アクションに使用する Microsoft .NET Framework/Microsoft .NET Core ベースのマネージ アセンブリを指定するには、プロジェクトに含まれている DLL および .exe ファイル一覧からファイルを選択します。
アセンブリ プロパティ	 *メモ• この設定は、パスが Directory テーブルのプロパティまたはディレクトリを参照するマネージ アセンブリを使用するカスタム アクションに適用します。* カスタム アクションに使用するマネージ アセンブリを指定する場合、次のいずれかを実行します: • 一覧内にある Windows Installer プロパティを選択する。 • 新しいプロパティの名前を入力する。 • Directory テーブルでディレクトリの名前を入力する。 プロパティ名とディレクトリ名は、角かっこ ([ ]) で囲む必要があります。必要に応じて、プロパティの後に文字列を追加することができます。作成されたパスは、アセンブリ ファイルの場所を示します。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
メソッド シグネチャ	 *メモ• この設定は、マネージ アセンブリを使用するカスタム アクションに適用します。* マネージ コード タイプのカスタム アクションでは、省略記号ボタン (...) をクリックすると、[メソッド シグネチャ] ダイアログ ボックスが表示されます。このダイアログ ボックスで、マネージ メソッドの引数と戻り値を指定することができます。このダイアログ ボックスでシグネチャ情報を指定すると、"メソッド シグネチャ" 設定の値として使用されます。 .NET Core タイプのカスタム アクションでは、次の使用可能なサブ設定を使って、クラス、メソッド名、引数、およびマネージ メソッドの戻り値を指定することができます: • **クラス**— メソッドが関連付けられているパブリック スラスの名前を完全な名前空間を含めて指定します。次のフォーマットを使用してください: **MyNamespace.MyClass** 例えば、**SampleCoreClassLibrary.SimpleWrapper** を次のディレクトリで使用可能な .NET Core サンプルのクラス名として指定することができます: <ISProductFolder>¥Samples¥Windowsinstaller¥Core CustomActions • **メソッド**— インストールで呼び出すパブリック メソッドを指定します。次のフォーマットを使用してください: **<MyMethod>** 例えば、**Note** を次のディレクトリで使用可能な .NET Core サンプルのメソッド名として指定することができます: <ISProductFolder>¥Samples¥WindowsInstaller¥Core Custom Actions • **引数**— この設定を使って、マネージ メソッドの 1 つ以上のパラメーターを指定できます。 **"引数"** 設定で使用可能な [追加] パラメーター ボタンをクリックして、パラメーター フィールドを追加します。パラメーターとしてプロパティを追加する場合、インストールが使用するパラメーター フィールドにプロパティ名を入力します。 • **戻りプロパティ**— その値がメソッドの戻り値に設定されるプロパティの名前を指定します。 詳細については、「アセンブリ カスタム アクション内のマネージ メソッドに署名を指定する」を参照してください。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
エラーメッセージ	 *メモ• この設定は、エラー メッセージを表示するカスタム アクションに適用します。* カスタム アクションの条件がターゲット システムで満たされたときに表示するエラー メッセージ テキストを入力します。 上記方法の代わりに、値にエラー テキストが含まれているプロパティ名を入力することもできます。プロパティ名は、角かっこ ([ ]) で囲む必要があります。
関数名	 *メモ• この設定は Kill-Process カスタム アクションに適用します。* 適切な関数を選択します: • KillProcess – "スクリプト内実行" 設定で即時オプションのいずれかが選択されていて、特定の名前を持つプロセスを強制終了する場合、このオプションを選択します。 • KillProcessByID – "スクリプト内実行" 設定で即時オプションのいずれかが選択されていて、特定のプロセス識別子 (PID) を持つプロセスを強制終了する場合、このオプションを選択します。 • KillProcessDeferred – "スクリプト内実行" 設定で、遅延、コミット、またはロールバック オプションのいずれかが選択されていて、特定の名前を持つプロセスを強制終了する場合、このオプションを選択します。 • KillProcessByIDDeferred – "スクリプト内実行" 設定で、遅延、コミット、またはロールバック オプションのいずれかが選択されていて、特定の PID を持つプロセスを強制終了する場合、このオプションを選択します。 このタイプのカスタム アクションの構成については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。
PowerShell ファイル名	 *メモ• この設定は、Binary テーブルに格納されている PowerShell カスタム アクションに適用します。* Binary テーブルに格納されているファイル リスト、またはプロジェクトに含まれているファイル リスト内で、PowerShell スクリプト ファイル (.ps1) を選択します。指定した場所が、Binary テーブルに格納されている場合、この設定で、省略記号ボタン (...) をクリックして、ファイルを参照します。 ターゲットについては、このタイプのカスタム アクションに関する他の情報も含めて、「PowerShell カスタム アクションの呼び出し」を参照してください。

テーブル 180 - [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
PowerShell スクリプト ファイルキー	 ***メモ**- この設定は、製品と共にインストールされる PowerShell スクリプトを使用する PowerShell カスタム アクションに適用します。* カスタム アクションに使用する PowerShell スクリプト ファイル (.ps1) を指定するには、プロジェクトに含まれている PowerShell ファイルの一覧からファイルを選択します。 ターゲットについては、このタイプのカスタム アクションに関する他の情報も含めて、「PowerShell カスタム アクションの呼び出し」を参照してください。
WiseScript 実行可能ファイル	 ***メモ**- この設定は WiseScript カスタム アクションに適用します。* ***エディション**- この種類のカスタム アクションは、AdminStudio の特定のエディションに含まれる InstallShield のバージョンで使用できます。* カスタム アクションに使用する WiseScript 実行可能ファイルへのパスを入力します。または、省略記号ボタン (...) をクリックして、適切な DLL ファイルを参照します。
プロセス	 ***メモ**- この設定は Kill-Process カスタム アクションに適用します。* インストールを続行する前に、強制終了するプロセスの実行可能ファイルの名前または PID を入力します。複数のプロセスは、セミコロンで区切ります。 - **関数名** が **KillProcess** または **KillProcessDeferrred** の場合、**"プロセス"** 設定で実行可能ファイル名を指定します。 - **関数名** が **KillProcessByID** または **KillProcessByIDDeferred** の場合、**"プロセス"** 設定で PID を指定します。 このタイプのカスタム アクションの構成については、「プロセスの強制終了カスタム アクションの呼び出し」を参照してください。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
戻り値の処理	カスタム アクション スレッドの処理方法を指定します。有効なオプションは次のとおりです: • 非同期 (完了を待機しない) • 非同期 (終了コードまで待機) • 同期 (終了コードを確認) • 同期 (終了コードを無視) これらのフラグは、メインとカスタム アクションのスレッドが同期実行 (インストーラーがメイン インストール スレッドを再開する前にカスタム アクション スレッドが完了するのを待機する) か非同期 (インストーラーがメイン インストールの続行と同時にカスタム アクションを実行する) かを指定するために使用されます。 オプションの中には、一部の種類のカスタム アクションには適用できないものがあります。選択された種類のカスタム アクションに関連するオプションのみが、一覧に表示されます。
スクリプト内実行	どのシーケンスによってアクションがトリガーされるのかを選択します。各オプションの詳細については、「スクリプト内実行」を参照してください。 これらのオプションはアクションコードを実行、ロールバックまたはコミットスクリプトにコピーします。[ターミナル サーバーに対応] オプションを選択すると、ターミナル サーバー マシン上でマシン単位のインストールが実行されいる間、カスタム アクションはユーザーを偽装します。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
実行スケジュール	アクションを実行する回数を指定します。たとえば、[ユーザー インターフェイス] シーケンスおよび [実行] シーケンスの両方にアクションがリストされている場合、このアクションを 2 回実行するか、または 1 回だけ実行するかを設定できます。選択可能なオプションは以下のとおりです: • **常に実行** – アクションは順番が来るたびに実行します。つまり、該当するアクションを [ユーザー インターフェイス] シーケンスで 1 回実行した後、[実行] シーケンスでさらにもう 1 回実行することができます。 • **クライアントで UI シーケンス後に実行される場合のみ実行** – アクションは、実行シーケンスが、ユーザー インターフェイス シーケンスの後に実行される時のみ実行されます。 • **1度のみ実行** – [ユーザー インターフェイス] および [実行] シーケンスの両方にアクションが存在する場合でも、アクションは一度だけ実行されします。[ユーザー インターフェイス] シーケンスが実行されている場合は、[実行] シーケンスのアクションはスキップされます。 • **1 プロセスにつき 1 回実行** – アクションは、[ユーザー インターフェイス] および [実行] シーケンスの両方にアクションが存在する場合、1 プロセスにつき 1 回実行されます。[実行] シーケンスが [ユーザー インターフェイス] シーケンスと同じプロセスで実行される場合、インストール時に、このアクションは [実行] シーケンスでスキップされます。このスケジュールによって、プロパティなどセッションの状態を変更するアクションの再実行を防ぐことができます。 "スクリプト内実行" 設定で [即時実行] 以外を選択した場合、"実行スケジュール" 設定は利用不可能になり、[常に実行] が設定されます。 ***プロジェクト・[1 度のみ実行]** オプションは InstallScript MSI プロジェクトでは使用できません。[ユーザー インターフェイス] シーケンスと [実行] シーケンスの両方にカスタム アクションが存在する場合は 2 度実行されます。これは InstallScript MSI プロジェクトでは、[ユーザー インターフェイス] シーケンスは InstallScript エンジンで実行され、[実行] シーケンスは Windows Installer で実行されるためです。*

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
ターゲット列の値を非表示にする	CustomAction テーブルの対応する [Target] 列に格納されている CustomActionData プロパティの値を詳細な MSI ログファイルに記録しないようにインストーラーを制限するかどうかを指定します。次のオプションを選択できます: • **はい**—CustomAction テーブルの対応する [Target] 列に格納されている CustomActionData プロパティの値を詳細な MSI ログファイルに記録しないようにインストーラーを制限します。 • **いいえ**—CustomAction テーブルの対応する [Target] 列に格納されている CustomActionData プロパティの値を詳細な MSI ログファイルにインストーラーが記録できるようにします。 デフォルトで、**"ターゲット列の値を非表示"** 設定は**[いいえ]** になっています。 ***メモ:*** *CustomActionData プロパティは、その名前がカスタム アクションの名前と一致するパブリック プロパティを使用して設定されます。このパブリック プロパティの値が詳細な MSI ログに表示されないようにするには、このプロパティも MsiHiddenProperties プロパティに追加する必要があります。これにより、Windows インストーラーは値をログ ファイル内で非表示として扱うように指示されます。* *詳細は、 Microsoft ドキュメント を参照してください。*
64 ビットの InstallScript を実行する	***プロジェクト:*** *この設定は、次のプロジェクト タイプの InstallScript カスタム アクションに適用します:* • *基本の MSI* • *InstallScript MSI* • *マージ モジュール* 64 ビット ターゲット システム上の 64 ビットファイルシステムおよび 64 ビット レジストリ ロケーションへのアクセスを有効にするために、InstallScript カスタムアクションを 64 ビット InstallScript エンジンによって実行するかどうかを指定します。次のオプションを選択できます: • **はい**–このオプションを選択すると、InstallScript カスタムアクションが 64 ビット InstallScript エンジンを使用してのみ実行されます。このオプションは、カスタムアクションが 64 ビットターゲットシステム上の 64 ビット ファイル システムまたは 64 ビット レジストリの場所にアクセスする必要がある場合に適しています。 • **いいえ**–このオプションを選択すると、InstallScript カスタムアクションが 32 ビット InstallScript エンジンを使用してのみ実行されます。このオプションは、カスタムアクションが 64 ビットターゲットシステム上の 32 ビット ファイル システムまたは 32 ビット レジストリの場所にアクセスする必要がある場合に適しています。 デフォルトで、この設定は **[いいえ]** になっています。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
サイレント モード	 *メモ• この設定は標準 DLL カスタム アクションに適用します。* インストールでカスタム アクションが DLL ファイルのロードおよび関数呼び出しに失敗した場合に表示される警告メッセージを抑制するかどうかを指定します。
MSI タイプ番号	この読み取り専用設定は、選択したカスタム アクションの Windows Installer 数値を識別します。基本的なカスタム アクション タイプについては、「カスタム アクションの種類についての概要」を参照してください。
コメント	このカスタム アクションを記録する内部コメントを入力します。入力したコメントは参照する目的でのみ使用され、エンドユーザーに表示されることはありません。
ヘルプ ファイル パス	省略ボタン (...) をクリックして、カスタム アクションの動作を説明するファイルを参照します。.txt、.htm、.rtf ファイルなど、テキストベースのファイルを指定してください。 基本の MSI、DIM、InstallScript MSI、およびマージ モジュール プロジェクトのビルド時に、InstallShield はこのファイルのコンテンツを ISCustomActionReference テーブルの Description 列にストリームします。 ダイレクト編集モードで作業を行っている場合、ヘルプ ファイルが選択されるとすぐに、InstallShield は選択されたファイルのコンテンツを ISCustomActionReference テーブルの Description 列にストリームします。また、InstallShield はこの設定を読み取り専用とし、この設定の値を [テキスト データ] として表示します。 ***Windows ロゴ •** 各カスタム アクションの意図された動作は、Windows ロゴ プログラムに基づいて文書化する必要があります。これは、システム管理者が製品を企業環境に配布する場合など、カスタム アクションの動作を把握する必要がある場面で特に有益です。インストール パッケージを検証した時、ヘルプ ファイル パス 設定の値が指定されていない場合、InstallShield は検証エラー ISICE10 を出力します。詳細については、「ISICE10」を参照してください。* すべてのビルトイン InstallShield カスタム アクションについて、InstallShield はこの設定を読み取り専用とし、InstallShield カスタム アクションをその値として表示します。

テーブル 180 • [アクション] 領域のカスタム アクション設定 (続き.)

設定	説明
パッチのアンインストール時に実行	 ***プロジェクト:*** *ダイレクト編集モードでプロジェクト作業を行っているとき、データベース スキーマの最小が 405 (Windows Installer 4.5 以降) になっていない場合、この設定は適用しません。* Windows Installer が、パッチのアンインストール時にのみカスタム アクションを実行するかどうかを指定します。.msi パッケージに含まれるカスタム アクションに対して [はい] を選択できます。パッチによって追加される新しいカスタム アクションに対して [はい] を選択することもできます。ただし、パッチによって既存のカスタム アクションに追加または削除されるカスタム アクションに対して [はい] を選択することはできません。この設定のデフォルト値は [いいえ] です。 Windows Installer 4.5 がパッチのアンインストール カスタム アクションを実行するとき、アンインストールされるパッチに含まれるカスタム アクションが使用されます。 ***メモ:*** *Windows Installer 4.5 は、この設定のサポートを含みますが、Windows Installer 4.0 以前は、これを含みません。したがって、この設定で [はい] を選択するときに、一部のターゲット システムの Windows Installer が 4.0 以前の可能性がある場合、このカスタム アクションが Windows Installer 4.0 以前で実行されないように条件を追加します。この条件を追加しなかった場合、Windows Installer 4.0 以前はインストール中にそのカスタム アクションを呼び出して、パッケージを修復、または更新します。* 条件には、MSIPATCHREMOVE プロパティを使用します。詳細については、Windows Installer ヘルプ ライブラリの「MSIPATCHREMOVE」を参照してください。

[シーケンス] 領域の設定

次のカスタム アクションの設定は、[シーケンス] 領域で提供されています。

テーブル 181 • [シーケンス] 領域のカスタム アクション設定

設定	説明
インストール UI シーケンス	[インストール] モードの [ユーザー インターフェイス] シーケンス中に、このカスタム アクションをスケジュールするには、適切なオプションを選択します。このカスタム アクションをシーケンスから除外するには、**<シーケンスにない>** を選択します。 代わりに、アクションを [カスタム アクション] エクスプローラーから、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグして、アクションをスケジュールすることもできます。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。アクションを、シーケンス内の別の位置へ (または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。

テーブル 181 • [シーケンス] 領域のカスタム アクション設定 (続き.)

設定	説明
インストール UI 条件	ターゲット システムで、このシーケンスでカスタム アクションを実行する前に、Windows Installer が評価する必要がある条件を指定する必要がある場合、この設定で条件ステートメントを入力する必要があります。条件が False に評価された場合、カスタム アクションは実行されません。 ステートメントを手動で入力する代わりに、この設定の省略記号ボタン (...) をクリックして [条件ビルダー] ダイアログ ボックスを起動すると、条件の作成処理を簡素化することができます。 詳細については、「条件ステートメントのビルド」を参照してください。
インストール実行シーケンス	[インストール] モードの [実行] シーケンス中に、このカスタム アクションをスケジュールするには、適切なオプションを選択します。このカスタム アクションをシーケンスから除外するには、**<シーケンスになし>** を選択します。 代わりに、アクションを [カスタム アクション] エクスプローラーから、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグして、アクションをスケジュールすることもできます。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。アクションを、シーケンス内の別の位置へ (または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。
インストール実行条件	ターゲット システムで、このシーケンスでカスタム アクションを実行する前に、Windows Installer が評価する必要がある条件を指定する必要がある場合、この設定で条件ステートメントを入力する必要があります。条件が False に評価された場合、カスタム アクションは実行されません。 ステートメントを手動で入力する代わりに、この設定の省略記号ボタン (...) をクリックして [条件ビルダー] ダイアログ ボックスを起動すると、条件の作成処理を簡素化することができます。 詳細については、「条件ステートメントのビルド」を参照してください。
アドバタイズ実行シーケンス	[アドバタイズ] モードの [実行] シーケンス中に、このカスタム アクションをスケジュールするには、適切なオプションを選択します。このカスタム アクションをシーケンスから除外するには、**<シーケンスになし>** を選択します。 代わりに、アクションを [カスタム アクション] エクスプローラーから、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグして、アクションをスケジュールすることもできます。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。アクションを、シーケンス内の別の位置へ (または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。

テーブル 181 • [シーケンス] 領域のカスタム アクション設定 (続き.)

設定	説明
アドバタイズ実行条件	ターゲット システムで、このシーケンスでカスタム アクションを実行する前に、Windows Installer が評価する必要がある条件を指定する必要がある場合、この設定で条件ステートメントを入力する必要があります。条件が False に評価された場合、カスタム アクションは実行されません。 ステートメントを手動で入力する代わりに、この設定の省略記号ボタン (...) をクリックして [条件ビルダー] ダイアログ ボックスを起動すると、条件の作成処理を簡素化することができます。 詳細については、「条件ステートメントのビルド」を参照してください。
管理 UI シーケンス	[管理] モードの [ユーザー インターフェイス] シーケンス中に、このカスタム アクションをスケジュールするには、適切なオプションを選択します。このカスタム アクションをシーケンスから除外するには、**<シーケンスになし>** を選択します。 代わりに、アクションを [カスタム アクション] エクスプローラーから、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグして、アクションをスケジュールすることもできます。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。アクションを、シーケンス内の別の位置へ (または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。
管理 UI 条件	ターゲット システムで、このシーケンスでカスタム アクションを実行する前に、Windows Installer が評価する必要がある条件を指定する必要がある場合、この設定で条件ステートメントを入力する必要があります。条件が False に評価された場合、カスタム アクションは実行されません。 ステートメントを手動で入力する代わりに、この設定の省略記号ボタン (...) をクリックして [条件ビルダー] ダイアログ ボックスを起動すると、条件の作成処理を簡素化することができます。 詳細については、「条件ステートメントのビルド」を参照してください。
管理実行シーケンス	[管理] モードの [実行] シーケンス中に、このカスタム アクションをスケジュールするには、適切なオプションを選択します。このカスタム アクションをシーケンスから除外するには、**<シーケンスになし>** を選択します。 代わりに、アクションを [カスタム アクション] エクスプローラーから、[シーケンス] エクスプローラーの下にあるシーケンス内の適切な位置までドラッグして、アクションをスケジュールすることもできます。次いで、それをシーケンス内で直前にくるアイテムの上にドロップします。アクションを、シーケンス内の別の位置へ (または、あるシーケンスから別のシーケンスへ) 移動するには、それを元の位置からドラッグして、シーケンス内で直前にくるアイテムの上にドロップします。

テーブル 181 • [シーケンス] 領域のカスタム アクション設定 (続き.)

設定	説明
管理実行条件	ターゲット システムで、このシーケンスでカスタム アクションを実行する前に、Windows Installer が評価する必要がある条件を指定する必要がある場合、この設定で条件ステートメントを入力する必要があります。条件が False に評価された場合、カスタム アクションは実行されません。 ステートメントを手動で入力する代わりに、この設定の省略記号ボタン (...) をクリックして [条件ビルダー] ダイアログ ボックスを起動すると、条件の作成処理を簡素化することができます。 詳細については、「条件ステートメントのビルド」を参照してください。

ショートカットのプロパティの構成に関する詳細は Windows Installer ヘルプ ライブラリの「CustomAction Table」を参照してください。

アクション テキストの設定

プロジェクト: *[カスタム アクションとシーケンス] ビューを使って、次のプロジェクトの種類でアクション テキストの設定を構成できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

エンド ユーザーに進行状況を伝えるために、インストールは一般的に、実行中の処理を説明する進行状況ダイアログで説明テキストを表示します。通常、インストールのステータスを表示する進行状況バーも同時に表示されます。標準アクションおよびカスタム アクションが発生すると、そのアクションについてのメッセージが進行状況ダイアログに表示されます。この機能は、実行するのに時間がかかるアクションで、特に役立ちます。インストールのログ記録が作成される場合、同じアクション テキストが記録されます。

[カスタム アクションとシーケンス] ビューの [アクション テキスト] エクスプローラーを使って、プロジェクト内の任意のアクションにアクション テキストを指定できます。

[カスタム アクションとシーケンス] ビューの [アクション テキスト] 領域でカスタム アクション アイテムまたは標準アクション アイテムをクリックして、以下の設定を構成できます。

テーブル 182 • アクション テキストの設定

設定	説明
説明	選択されたアクションについて説明するテキストを入力します。たとえば、InstallFiles アクションのデフォルトの説明は、次のとおりです: 新しいファイルをコピーします Windows Installer がこのアクションを起動すると、入力したテキストが進行状況ダイアログに表示されます。詳細については、「進行状況ダイアログにアクションの説明を表示する」を参照してください。 実行時にログ ファイルが作成される場合、Windows Installer がこのアクションを起動したときに、説明がログ ファイルに記述されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 182 • アクション テキストの設定 (続き.)

設定	説明
テンプレート	アクション データの記録をフォーマットするのに使用するテンプレートを入力します。たとえば、InstallFiles アクションのデフォルトのテンプレートは、次のとおりです: ファイル: [1]、ディレクトリ: [9]、サイズ: [6] Windows Installer がこのアクションを起動すると、アクションの説明が進行状況ダイアログに表示されます。前述の InstallFiles の例で、ターゲット システムにファイルが転送されると、[1] プロパティがそのファイル名に置換されます。[9] プロパティは、そのファイルを含むディレクトリで、また [6] プロパティは、そのファイルのサイズ (バイト単位) で置換されます。 デフォルトで、アクション データは進行状況ダイアログに表示されませんので、ご注意ください。詳細については、「進行状況ダイアログにアクション データを表示する」を参照してください。 選択されたアクションにテンプレートが指定されていない場合、アクションが起動されたときにデータは表示されません。また、インストールは完全なユーザー インターフェイスを使って実行しなくてはなりません。インストールがサイレント、または基本のユーザー インターフェイスで実行されると、アクション データは表示されません。 特定の標準アクションの "テンプレート" 設定の値として使用できるデータ フィールドに関する詳細は、Windows Installer ヘルプ ライブラリの「Standard Actions Reference」で、特定のアクションについてのヘルプを参照してください。 ***プロジェクト:*** *InstallScript MSI インストールでは、状態ダイアログにアクション データを表示できません。ただし、そのテンプレートはインストールのログ ファイルに記録されます。* ***メモ:*** *この設定の値は、一部の標準アクションで必ず* [1] *でなくてはなりません。* - *GenerateScript* - *Rollback* - *RollbackCleanup* *このため、これらのアクションの "テンプレート" 設定は、無効となっています。* この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

[サポート ファイル] ビュー (アドバンスト UI、基本の MSI、InstallScript オブジェクト、およびスイート/アドバンスト UI プロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript オブジェクト*
- *スイート/アドバンスト UI*

[サポート ファイル] ビューでは、プロジェクトにサポート ファイルを追加したり、基本の MSI プロジェクトではディスク イメージ フォルダーにファイルを追加したりできます。

[サポート ファイル] ビューには、以下のような主要な領域があります。

サポート ファイル

[サポート ファイル] 領域では、インストール プロセス中にのみインストールが必要とするサポート ファイルを追加、ソートおよび削除することができます。インストールが完了すると、サポート ファイルは削除されます。任意の言語依存ファイルを、適切な言語固有領域に追加します。言語非依存 ファイルを、言語非依存領域に追加します。

スプラッシュ画面

プロジェクト: *[スプラッシュ画面] 領域は、基本の MSI プロジェクトで使用できます。*

[スプラッシュ画面] 領域では、エンド ユーザーがインストールを開始したときに表示されるスタートアップ画像をプロジェクトに追加できます。画像ファイルはビットマップ (.bmp) 形式でなくてはなりません。ファイルを適切な言語固有、または言語非依存フォルダーに追加します。

メモ: *単一言語のインストールでは、単一言語用に指定されたスプラッシュ画面 (または、この単一言語にスプラッシュ画面が指定されていない場合は、言語非依存のスプラッシュ画面) が表示されます。*

複数言語のインストールでは、インストールを実行中の言語でスプラッシュ画面が表示されます。インストールを実行中の言語用にスプラッシュ画面が指定されていない場合、言語非依存のスプラッシュ画面が使用されます。

インストール処理中に、1 つのスプラッシュ画面のみがエンド ユーザーに表示されます。言語非依存領域、または特定の言語に複数のスプラッシュ画面がある場合、実行時には一覧にある最初のファイルが表示されます。

アドバンス ファイル

プロジェクト: *[アドバンス ファイル] 領域は、基本の MSI プロジェクトで使用できます。*

[アドバンス ファイル] 領域の Disk1 アイテムでは、インストールメディアの Disk1 に入れるファイルとフォルダーを指定することができます。これらのファイルとフォルダーは、インストールの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストール メディアにリンクさせることができます。たとえば、アプリケーションのインストールに含めたくない大きな再配布可能ファイルをアプリケーションに含める場合があるとします。このようなファイルは Disk1 フォルダーに入れます。

[サポート ファイル/ビルボード] ビュー (InstallScript および InstallScript MSI プロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

[サポート ファイル/ビルボード] ビューでは、プロジェクトにサポート ファイルとビルボード ファイルを追加したり、ディスク イメージ フォルダーにファイルを追加したりできます。

[サポート ファイル/ビルボード] ビューには、以下の主要な領域があります。

サポート ファイル

[サポート ファイル] 領域では、インストール プロセス中にのみインストールが必要とするサポート ファイルを追加、ソートおよび削除することができます。任意の言語依存ファイルとフォルダーを、適切な言語固有領域に追加します。言語非依存 ファイルとフォルダーを、言語非依存領域に追加します。

InstallScript コードでは、サポート ファイルは一時ディレクトリに展開され、インストール完了時に削除されます。

ビルボード

[ビルボード] アイテムを使用すると、インストールプロセス中にエンドユーザーに表示するファイルを指定できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。詳細については、「InstallScript および InstallScript MSI インストールでビルボードを表示する」を参照してください。

スプラッシュ画面

プロジェクト: *InstallScript プロジェクトでは、スプラッシュ画面のファイルとして .bmp と .gif ファイルを使用できます。InstallScript MSI プロジェクトでは、スプラッシュ画面のファイルとして .bmp ファイルのみを使用できます。*

[スプラッシュ画面] 領域では、エンド ユーザーがインストールを開始したときに表示されるスタートアップ画像をプロジェクトに追加できます。ファイルを適切な言語固有、または言語非依存フォルダーに追加します。

メモ: *単一言語のインストールでは、単一言語用に指定されたスプラッシュ画面 (または、この単一言語にスプラッシュ画面が指定されていない場合は、言語非依存のスプラッシュ画面) が表示されます。*

複数言語のインストールでは、インストールを実行中の言語でスプラッシュ画面が表示されます。インストールを実行中の言語用にスプラッシュ画面が指定されていない場合、言語非依存のスプラッシュ画面が使用されます。

インストール処理中に、1 つのスプラッシュ画面のみがエンド ユーザーに表示されます。言語非依存領域、または特定の言語に複数のスプラッシュ画面がある場合、実行時には一覧にある最初のファイルが表示されます。

プロジェクト: *InstallScript プロジェクトの場合、スプラッシュ画面ファイルの名前は必ず* Setup.bmp *または* Setup.gif *にします。InstallScript MSI プロジェクトの場合、このファイル名を使う必要はありません。*

アドバンス ファイル

[アドバンスファイル] アイテムでは、インストールメディアのディスクに配置するファイルとフォルダーを指定できます。これらのファイルとフォルダーは、インストール プログラムの実行時、ターゲット システムに自動的にインストールはされません。代わりに、アプリケーションまたはインストールからインストール メディアにリンクさせることができます。たとえば、アプリケーションのインストールに含めたくない大きな再配布可能ファイルをアプリケーションに含める場合があるとします。

テーブル 183 ・ アドバンス ファイル項目

項目	説明
Disk1	インストールメディアの最初のディスクに配置するファイルとフォルダーを追加します。
最後の Disk	インストールメディアの最初のディスクに配置するファイルとフォルダーを追加します。 ***プロジェクト:*** *この情報は、InstallScript プロジェクトに適用します。*
その他	インストールメディアの特定のディスクにファイルとフォルダーを追加します。ディスクはリリース ウィザードの [一般オプション] パネルでビルド時に指定します。 ***プロジェクト:*** *この情報は、InstallScript プロジェクトに適用します。*

[システム検索] ビュー

プロジェクト: *[システム検索] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[システム検索] ビューは、インストール前に、ターゲット システム上にある特定のファイル、フォルダー、レジストリキー、.ini ファイル値、または、.xml ファイル値を検索するための Windows Installer 機能を備えています。この機能を使って、アプリケーション、バージョンおよび構成データを検索することができます。

[システム検索] ビューを使うと、ターゲットのシステムで実行する各検索を一覧にしたグリッドが表示されます。InstallShield に含まれる検索、またはレポジトリに格納されている検索のどちらの場合でも、このビューを使って、定義済みシステム検索をプロジェクトに追加できます。また、システム検索ビューを利用して定義済み検索をカスタマイズしたり、プロジェクトのシステム検索を独自に定義したりすることもできます。

独自の検索を定義すると常に、システム検索ウィザードが起動します。検索オプション一覧から選択して、検索するサブフォルダーレベル数などの検索詳細を指定できます。既存の検索を変更する場合、システム検索ウィザードで初期の選択を変更できます。

[プロパティ マネージャー] ビュー

プロジェクト: *[プロパティ マネージャー] ビューは、次の Windows Installer ベースのプロジェクトで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

このビューは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます。

Windows Installer ベースのプロジェクトにおける [プロパティ マネージャー] ビューとアドバンスト UI およびスイート/アドバンスト UI プロジェクトのプロパティーには、いくつかの異なる点があります。これらの違いについては、適宜、説明が記載されています。

Windows Installer エンジンとアドバンスト UI およびスイート/アドバンスト UI エンジンは、プロパティを使って、グローバル情報を保守します。[プロパティ マネージャー] ビューには、Windows Installer エンジンまたはアドバンスト UI およびスイート/アドバンスト UI エンジンが実行時に使用するインストール プロパティの一覧が表示されます。[プロパティ マネージャー] ビューでインストーラー プロパティを作成、変更、または削除できます。ビルド時に、InstallShield は、[プロパティ マネージャー] ビューのプロパティを、作成するインストールに書き込みます。

次の一覧は、このビューで処理できるタスクの一部です:

- プロジェクトで定義されているプロパティを参照する。
- プロパティを追加、変更、および削除する。
- ビューに表示されているプロパティをフィルターして、特定の文字列を含むプロパティを非表示にする。
- ビュー内の列のサイズを変更、およびその順序を変更する。
- 列のヘッダーをクリックして、ビュー内の行を列ごとに並べ替える。
- 列ヘッダーをグループ ボックス領域 (ビューのボタンの下にある領域) にドラッグ アンド ドロップして、ビュー内の行を階層形式に編成する。

- Windows Installer ベースのプロジェクトの場合: プロパティをローカライズ可能にして、インストールが使用する言語に基づいて異なる値を設定する。

プロジェクト: *Windows Installer ベースのプロジェクトでは、すべて大文字のプロパティはパブリック プロパティなので、実行時にエンド ユーザーがコマンドラインで変更することができます。その他のプロパティは、カスタム アクションまたはダイアログの動作を介して、リリースのビルド以前または実行時に設定しなければなりません。*

[プロパティ マネージャー] ビューを使って作業する

[プロパティ マネージャー] ビューは、次の要素で構成されます:

- ボタン行とその他のコントロール
- グループ ボックス領域 (ボタン行の下)
- スプレッドシート形式のテーブル

テーブルの各行は、プロジェクトに含まれるプロパティを示します。

次の表では、[プロパティ マネージャー] ビューに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 184 • [プロパティ マネージャー] ビューのコントロール

コントロールの名前	アイコン	説明
新しいプロパティ		プロジェクトに新しいプロパティを追加します。 ***プロジェクト:*** *Windows Installer ベースのプロジェクトの場合: ローカライズ可能なプロパティを作成するには、このボタンの隣にある矢印をクリックしてから、[ローカライズ可能なプロパティ] をクリックします。詳細については、「ローカライズ可能なプロパティを作成する」を参照してください。*
選択したプロパティを削除する		選択した行 (複数可) を削除します。
選択したプロパティをクリアする		選択したプロパティの値を削除します。
選択したプロパティをローカライズ可能にする		 ***プロジェクト:*** *このボタンは、Windows Installer ベースのプロジェクトで使用できます。* 選択したプロパティの値に文字列 ID を追加して、プロジェクトがサポートする各言語に対して異なるプロパティ値の設定を可能にします。詳細については、「既存プロパティをローカライズ可能にする」を参照してください。

テーブル 184 • [プロパティ マネージャー] ビューのコントロール (続き.)

コントロールの名前	アイコン	説明
すべてのグループを展開する		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を表示します。
すべてのグループを折りたたむ		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を隠します。
検索グリッド		この検索ボックスで指定した文字列に従って、[プロパティ マネージャー] ビューに表示されるプロパティをダイナミックにフィルターします。このボックスに文字列を入力すると、それを含まない行すべてが非表示となります。
プロパティ マネージャー ヘルプ		[プロパティ マネージャー] ビューのヘルプを表示します。
グループ化する列のヘッダーをここにドラッグ		このグループ ボックス領域を使って、ビュー内の行をグループ分けします。ビューでは、列のヘッダーをグループ ボックスにドラッグ アンド ドロップするだけで複数階層にグループ化することができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。 詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

次の表は、[プロパティ マネージャー] ビューの各列について説明します。

テーブル 185 • [プロパティ マネージャー] ビューの列

列	説明
名前	この列には、Windows Installer またはアドバンスト UI またはスイート/アドバンスト UI プロパティの名前が表示されます。
値	この列には、プロパティの値が表示されます。 ***プロジェクト:*** *Windows Installer ベースのプロジェクトの場合: プロパティがローカライズ可能に設定されている場合、この列には、文字列値の前に中括弧で囲まれた文字列 ID が表示されます。*
コメント	この列には、プロパティについての内部メモが含まれます。コメントは実行時には表示されません。

テーブル 185 • [プロパティ マネージャー] ビューの列 (続き.)

列	説明
フォーマット済み	 ***プロジェクト•*** *この列は、次のプロジェクト タイプで使用できます:* • *アドバンスト UI* • *スイート/アドバンスト UI* この列には、"値" 列の値が、実行時に解決させるプロパティ名、環境変数リファレンス、その他の特殊文字列を含む形式化された式であるかどうかを指定できるチェック ボックスがあります。実行時の解決処理は、インストール中に OnBegin イベントのアクションが実行される前、早い段階で発生します。 実行時にプロパティ値をそのまま残す場合、このチェックボックスをクリアします。各括弧その他の文字は、何の意味も持ちません。 形式化された式を実行時に置換するには、このチェックボックスを選択します。 形式化された式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

[イベント] ビュー

プロジェクト• *[イベント] ビューは、その後スイート/アドバンスト UI プロジェクトで使用できます。*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[イベント] ビューを使って、独自のアクションを作成してスイート/アドバンスト UI プロジェクトに追加できます。また、このビューを使って、1 つ以上のランタイム イベント中に発生するアクションをスケジュールするか、[パッケージ] ビューを使って、インストールの特定のパッケージが実行する直前または直後にアクションをスケジュールできます。

ヒント• *オプションで、アクションをスイート/アドバンスト UI インストールのウィザード インターフェイス要素と関連付けることができます。詳細については、「ウィザード インターフェイス内の要素のアクションを構成する」を参照してください。*

[イベント] ビューには [アクション] タブと [イベント] 領域の 2 つの個別の領域があります。

アクション

[イベント] ビューの [アクション] エクスプローラーを使って、プロジェクト内のアクションを定義および構成します。実行可能ファイルを実行、DLL 関数を呼び出し、PowerShell スクリプトを実行、スイート/アドバンスト UI プロパティを設定、InstallScript コードを実行、またはマネージ アセンブリでパブリック メソッドを呼び出すアクションを作成できます。

各アクションの設定に関する詳細は、「[イベント] ビューの設定」を参照してください。

イベント

イベントは、インストール モード、メンテナンス モード、およびステージングのみモード中に実行されるアクションのすべてを指示します。スイート/アドバンスト UI エンジンは、イベントにスケジュールされているアクションを、このビューでリストされる順番で実行します。イベントの各種類についての詳細は、「スイート/アドバンスト UI インストールにおけるイベントの種類」を参照してください。

[イベント] ビューの設定

プロジェクト: *[イベント] ビューは、その後スイート/アドバンスト UI プロジェクトで使用できます。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[イベント] ビューの設定は、次の主なカテゴリに分かれています:

- [アクション] タブの [アクション]領域の設定
- .exe アクションの [アクション] タブにある [戻り値の処理] 領域の設定
- [イベント] 領域の設定

[アクション] エクスプローラー、または [イベント] エクスプローラーで PowerShell アクションを選択すると、InstallShield の右側に [PowerShell スクリプト] タブが表示されます。このタブを使ってスクリプトを編集できます。

[アクション] タブの [アクション]領域の設定

[アクション] タブの [アクション] 領域では、次のアクションの設定を行うことができます。

テーブル 186 • [アクション] タブの [アクション] 領域にあるアクションの設定

設定	説明
File	 ***メモ•*** *この設定は .exe、DLL、マネージ コード、および . NET Core アクションに適用します。* アクションに使用するファイルへのランタイム パスを入力します。別の方法として、実行時にそのファイルが存在しない場合で、スイート/アドバンスト UI インストール内のいずれのパッケージもそれをインストールしないとき、この設定で省略記号ボタン (...) をクリックして、ファイルを参照できます。 ファイルを参照すると、スイート/アドバンスト UI プロジェクトの [サポート ファイル] ビューにそのファイルがサポート ファイルとして追加されます。実行時、このファイルは製品のインストール処理中の時だけターゲット システムで使用できます。
ソース パス	 ***メモ•*** *この設定は PowerShell アクションに適用します。* アクションに使用する PowerShell スクリプト (.ps1) へのソース パスを入力します。その代わりに、この設定で省略記号ボタン (...) をクリックして、スクリプトを参照することもできます。 ファイルを参照すると、スイート/アドバンスト UI プロジェクトの [サポート ファイル] ビューにそのファイルがサポート ファイルとして追加されます。実行時、このファイルは製品のインストール処理中の時だけターゲット システムで使用できます。 場所をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。 ***ヒント•*** *このビューの [PowerShell スクリプト] タブを使って、このアクションのコードを編集します。*
引数	 ***メモ•*** *この設定は .exe アクションに適用します。* 適切な場合、アクションと共に使用するコマンドライン パラメーターを指定します。複数のパラメーターはスペースで区切ります。

テーブル 186 • [アクション] タブの [アクション] 領域にあるアクションの設定 (続き.)

設定	説明
関数名	 ***メモ•*** *この設定は DLL および InstallScript アクションに適用します。* DLL アクションの場合: 呼び出す DLL 内の関数名を入力します。 InstallScript アクションの場合: プロジェクトの InstallScript ファイルに含まれている関数リストで、呼び出す InstallScript 関数を選択します。
クラス	 ***メモ•*** *この設定はマネージ コードおよび . NET Core アクションに適用します。* 選択されたアクションが呼び出すマネージ メソッドのクラス名を入力します。 詳しくは、次を参照してください: • スイート/アドバンスト UI インストールでマネージ コード アクションを使用する. • スイート/アドバンスト UI インストールで .NET Core アクションを使用する
メソッド	 ***メモ•*** *この設定はマネージ コードおよび . NET Core アクションに適用します。* インストールで呼び出すパブリック メソッドを入力します。 詳しくは、次を参照してください: • スイート/アドバンスト UI インストールでマネージ コード アクションを使用する. • スイート/アドバンスト UI インストールで .NET Core アクションを使用する
管理者権限が必要	 ***メモ•*** *この設定は .exe、DLL、PowerShell、InstallScript、マネージ コード、および . NET Core アクションに適用します。* このアクションが Windows Vista 以降または Windows Server 2008 以降のシステム上で管理者権限を必要とするかどうかを指定します。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える」を参照してください。

テーブル 186 • [アクション] タブの [アクション] 領域にあるアクションの設定 (続き.)

設定	説明
動詞	 ***メモ•*** *この設定は .exe アクションに適用します。* 選択されたアクションのファイルで使用する動詞を指定するか、リストから適切な動詞を選択します。 この設定を空白にすると、ターゲット システム上でこのファイルのデフォルトの動詞が使用されます。デフォルトの動詞が指定されていない場合は、*open* 動詞が使用されます。いずれの動詞も使用できない場合、レジストリで最初にリストされている動詞が使用されます。 runas 動詞の使用は推奨されません。アクションに管理者権限が必要な場合、アクションの "管理者権限が必要" 設定に [はい] を選択します。詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える」を参照してください。
ウィンドウ	 ***メモ•*** *この設定は .exe アクションに適用します。* .exe アクションが起動されたときの最初のウィンドウの状態を選択します。選択可能なオプションは以下のとおりです: • **表示**—ウィンドウを現在のサイズと位置で表示します。 • **隠す**—ウィンドウを非表示にします。 • **最小化**—ウィンドウを最小化して表示します。 • **最大化**—ウィンドウを最大化して表示します。 • **標準**—ウィンドウを元のサイズと位置で表示します。
終了を待機する	 ***メモ•*** *この設定は .exe アクションに適用します。* アクションが完了するまでインストール、インストールを待機してから、残りのインストール処理を続行するどうかを指定します。選択可能なオプションは以下のとおりです: • **はい**—インストールは、アクションが完了するまで待機してから再開します。デフォルトでは、これが設定されています。 • **いいえ**—インストールは、次のインストール処理と同時にアクションを実行します。 このせ設定で [はい] を選択するとき、[戻り値の処理] 領域にある設定を使って、アクションが戻す可能性のある様々なコードを指定できます。

テーブル 186 • [アクション] タブの [アクション] 領域にあるアクションの設定 (続き.)

設定	説明
プロパティ名	 ***メモ•*** *この設定は、プロパティを設定するアクションに適用します。* プロパティ名を入力するか、使用できるプロパティー覧からプロパティを選択します。 リストに表示されるプロパティ名は、[プロパティ マネージャ] ビューに含まれるものと同じです。別の任意のプロパティ名を入力して、実行時に利用することができます。新しいプロパティを明示的に作成する場合、[プロパティ マネージャー] ビューを利用することができます。
プロパティ値	 ***メモ•*** *この設定は、プロパティを設定するアクションに適用します。* プロパティの値を入力します。 "プロパティ値をフォーマット" 設定に [はい] を選択した場合、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
プロパティ値をフォーマット	 ***メモ•*** *この設定は、プロパティを設定するアクションに適用します。* "値" 設定に入力する値が、実行時に解決させるプロパティ名、環境変数リファレンス、その他の特殊文字列を含む形式化された式であるかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい**—実行時に形式化された式を置換します。 • **いいえ**—実行時にプロパティ値をそのまま残します。入力する値は、形式化された式を含みません。 これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

.exe アクションの [アクション] タブにある [戻り値の処理] 領域の設定

.exe アクションの [アクション] タブの [戻り値の処理] 領域では、次のアクションの設定を行うことができます。

テーブル 187 • .exe アクションの [アクション] タブにある [アクション] 領域の [戻り値の処理] の設定

設定	説明
無視する値	*メモ• この設定は .exe アクションに適用します。* インストールが次のインストール処理を続行するときにアクションが戻す可能性のあるコードのリストをコンマ区切りで指定します。 この設定は、アクションの "終了を待機" 設定に [はい] を選択した時に使用できます。
中止する値	*メモ• この設定は .exe アクションに適用します。* インストールが中止するときにアクションが戻す可能性のあるコードのリストをコンマ区切りで指定します。アクションが起動またはロードできない場合、インストールは中止します。 この設定は、アクションの "終了を待機" 設定に [はい] を選択した時に使用できます。
再起動する値	*メモ• この設定は .exe アクションに適用します。* ターゲット システムが再開始するときに、アクションが戻す可能性のあるコードのリストをコンマ区切りで指定します。 この設定は、アクションの "終了を待機" 設定に [はい] を選択した時に使用できます。
キャンセルする値	*メモ• この設定は .exe アクションに適用します。* インストールがキャンセルするときにアクションが戻す可能性のあるコードのリストをコンマ区切りで指定します。 この設定は、アクションの "終了を待機" 設定に [はい] を選択した時に使用できます。

テーブル 187 • .exe アクションの [アクション] タブにある [アクション] 領域の [戻り値の処理] の設定 (続き.)

設定	説明
デフォルトの応答	 ***メモ•*** *この設定は .exe アクションに適用します。* アクションが戻すコードが [戻り値の処理] 領域のその他の設定のどれにも指定されていない時に、インストールの応答方法を説明するオプションを選択します。選択可能なオプションは以下のとおりです: • **無視**—インストールは次のインストール処理を続行します。デフォルトでは、これが設定されています。 • **中止**—インストールが中止します。 • **再起動**—ターゲット システムを再起動します。 • **キャンセル**—インストールがキャンセルします。 この設定は、アクションの ″終了を待機″ 設定に [はい] を選択した時に使用できます。

[イベント] 領域の設定

次のアクションの設定は、[イベント] 領域で使用できます。

テーブル 188 • [イベント] 領域にある [アクション] の設定

設定	説明
条件	この設定を使って、アクションを実行するかどうかを評価するためにスイート/アドバンスト UI インストールが使用する 1 つ以上の条件を指定できます。たとえば、アクションが特定のプラットフォーム上でのみ実行する場合、このアクションの条件にプラットフォーム要件を作成することができます。スイート/アドバンスト UI は、適切なプラットフォーム上でのみアクションを起動します。 1 つ以上の新しいアクション条件を追加するには、この設定で [新しい条件] ボタンをクリックします。InstallShield によって、"条件" 設定の下に新しい行が追加されます。この行のリストから All、Any、または None のうち適切なオプションを選択します。次にこの行で、[新しい条件] ボタンをクリックし、適切なオプションを選択してから条件ステートメントのビルドを続行します。 ***ヒント•*** *アクションに "機能の操作" 条件または "パッケージの操作" 条件を定義する場合、アクションをスケジュールできる最初のイベントは OnStaging です。* 詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。 1 つ以上の条件ステートメントが構成されると、"条件" 設定には **(条件)** と表示されます。何も構成されていない場合、"条件" 設定には **(空白)** と表示されます。
表示テキスト	***メモ•*** *この設定は .exe、DLL、PowerShell、InstallScript、マネージ コード、および . NET Core アクションに適用します。* 選択されたアクションについて説明するテキストを入力します。たとえば、アクションがターゲット システムを構成するスクリプトを実行している場合、次の文字列を入力できます: システムの構成中 InstallationProgress ウィザード ページが表示されるときに、インストールがアクションを起動する場合、入力されたテキストがウィザード ページに表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

[ユーザー インターフェイス] ビュー

プロジェクト: *[ユーザー インターフェイス] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *スイート/アドバンスト UI*
- *トランスフォーム*

インストールの外観は、他社製品から差別化する上で非常に重要な要素の 1 つです。インストールの外観は、以下にリストされているビューを使って簡単にカスタマイズできます。

ダイアログ

プロジェクト: *[ダイアログ] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ダイアログ] ビューで、実行時に表示するダイアログを選択することができます。ダイアログの動作を制御する方法は、どのエンジンがユーザー インターフェイスを制御するかによって異なります。基本の MSI プロジェクトでは、Windows Installer エンジンがダイアログを表示し、InstallScript プロジェクトと InstallScript MSI プロジェクトでは、InstallScript エンジンがダイアログを表示します。詳細については、次を参照してください。

- [ダイアログ] ビュー (InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクト)
- ダイアログ ビュー (基本の MSI、その他の Windows Installer ベースのプロジェクト)

ウィザード インターフェイス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューを使って、アドバンスト UI またはスイート/アドバンスト UI インストールのユーザー インターフェイスをカスタマイズします。このビューでは、デフォルト ページの一覧からウィザード ページを追加、および削除して、各ページのレイアウトや動作をカスタマイズすることができます。

ビルボード

プロジェクト: *[ビルボード] ビューは、基本の MSI プロジェクトで使用できます。*

InstallScript または InstallScript MSI プロジェクトにおけるビルボード サポートについての詳細は、「[サポート ファイル/ビルボード] ビュー (InstallScript および InstallScript MSI プロジェクト)」を参照してください。

ビルボードとは、インストールのファイル転送段階で、指定された時間表示されるイメージまたは Adobe Flash アプリケーション ファイルです。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

文字列エディター

プロジェクト: *[文字列エディター] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクトのローカライズ処理を簡素化するために、インストール処理中、実行時に表示されるすべてのテキスト文字列は、統合された 1 つのビュー、[文字列エディター] ビューに表示されます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。またこのビューでは、各言語の文字列エントリをファイルにエクスポートして、ファイルにリストされている値を翻訳してから、翻訳済みファイルをプロジェクトにインポートすることができます。

[ダイアログ] ビュー (InstallScript、InstallScript MSI、および InstallScript オブジェクト プロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

[ダイアログ] ビューには、標準エンドユーザー ダイアログの一覧が含まれています。ダイアログは関数名で識別されます。このビューでダイアログをクリックすると、右のペインにサンプル ダイアログが表示されます。

一覧内のダイアログ名は、レイアウトを編集するまで*ゴースト (非実体) 化*になっています。ゴースト化された名前の場合、(isres.dll の) デフォルトのダイアログがユーザー インターフェイスで使用されます。ダイアログのレイアウトを編集すると、ダイアログ名が太字で一覧に表示され、実行時にこのダイアログが isuser.dll から転送されます。

インストール プロジェクトがその他の言語をサポートする場合、これらの言語は編集されたダイアログの下にノードとして表示されます。各言語のレイアウトを別々に編集することができます。

ヒント: *ダイアログ エディターでダイアログを選択し、編集しても、エンドユーザー インターフェイスにそのダイアログが自動的に表示されるわけではありません。実行時にダイアログを表示するには、それを InstallScript コードに含まなくてはなりません。たとえば、初回インストール中に表示されるダイアログのコードは、* **OnFirstUIBefore** *および* **OnFirstUIAfter** *イベント ハンドラーの一部です。詳細については、「InstallScript および InstallScript MSI インストールでダイアログを表示する」を参照してください。*

ダイアログ リソース ファイル

デフォルトで、InstallScript と InstallScript MSI プロジェクトのすべてのダイアログはデフォルトで _IsRes.dll というリソース .dll ファイルに保存されます。.dll ファイルはインストールのビルド時に配布パッケージにビルドされます。サポート言語にはそれぞれ個別の _IsRes.dll が用意されていて、InstallShield フォルダーの Redist フォルダー (例、*InstallShield Program Files フォルダー*¥Redist) に保存されています。_IsRes.dll はいずれも変更しないでください。

ダイアログ エディターでダイアログを編集すると、InstallShield は元の _IsRes.dll ファイルからダイアログのコピーを作成します。このコピーはプロジェクトファイルに保存されます。プロジェクトがビルドされると、編集されたすべてのダイアログ (および作成された新しいダイアログ) は _IsUser{現在の言語}.dll というリソース .dll ファイルに組み込まれます。英語の場合、.dll ファイルは _IsUser1033.dll と呼ばれます。その後、_IsUser{LanguageID}.dll が配布パッケージに組み込まれます。

ダイアログを表示するエンジンは、実行時にまず _IsUser{LangId}.dll のダイアログを探します。ダイアログが見つからない場合、_IsRes.dll のデフォルトのバージョンを探します。エンジンはリソース識別子に基づいてダイアログを探します。ダイアログのリソース ID を見るには、ダイレクト エディター の Dialog テーブルを調べます。

ダイアログ ビュー (基本の MSI、その他の Windows Installer ベースのプロジェクト)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

[ダイアログ] ビューには [ダイアログ] エクスプローラーがあります。[ダイアログ] エクスプローラーには、[テーマ]、[すべてのダイアログ] という 2 つのメイン フォルダーがあります。

テーマ

[テーマ] フォルダーには、プロジェクトでダイアログに使用できるダイアログ テーマの全一覧が含まれています。ダイアログ テーマとは、エンドユーザー ダイアログに統一感のとれた個性的な印象を与えることができる、あらかじめ定義されている 1 セットのイメージです。

すべてのダイアログ

[すべてのダイアログ] フォルダーには、プロジェクト内のダイアログ一覧が含まれています。[ダイアログ] エクスプローラーの各ダイアログ アイテムには、ダイアログに関連付けられている動作を構成することができる [動作] アイテムと、選択された言語のダイアログ エディターでダイアログを表示する言語アイテムが最低 1 つ含まれています。ダイアログ エディターを利用して、異なる言語ごとにレイアウトを別々に編集することができます。

新しい基本の MSI プロジェクトを作成すると、通常 Windows Installer パッケージがダイアログを表示する 2 つの [ユーザー インターフェイス] シーケンスに一連のデフォルト ダイアログが用意されます。1 つめは [インストール] シーケンス (.msi ファイルをダブルクリックした時など、インストールをデフォルトのモードで起動したときに実行される) + パッケージを初めてインストールする、再インストールする、またはアンインストールするかによって異なる別々のダイアログ、2 つめは [管理] シーケンス (/a コマンドライン オプションを使ってインストールを起動したときに実行されるアクションの一覧) です。(エンドユーザー ダイアログは通常 [アドバタイズ] シーケンスでは表示されません。[アドバタイズ] シーケンスには、インストールを /j コマンドライン オプションを使って起動したときに実行されるアクションの一覧が入っています。)

プロジェクト: *DIM およびマージ モジュール プロジェクトでダイアログを作成することができますが、その DIM またはマージ モジュールをインストール プロジェクトに追加するまでダイアログをシーケンスに追加することはできません。その後で、モジュールにインクルードされているすべてのダイアログを含めることができます。*

[ウィザード インターフェイス] ビュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューには、ウィザード インターフェイス エクスプローラーがあります。[ウィザード インターフェイス] エクスプローラーは、[スタイル]、[ブラシ]、[コントロール テーマ]、[ウィザード ページ]、[2 番目のウィンドウ]、および [最小ウィザード ページ] の領域で構成されています。

スタイル

[ウィザード インターフェイス] ビューの [スタイル] 領域では、ウィザードの 1 ヶ所以上に適用するフォーマット情報のまとまりを構成して、素早く外観を変更することができます。[スタイル] 領域では、次の 2 種類のスタイルを定義できます:

- **フォント セット**—フォント セット とは、フォント名、サイズ、太さなどの属性を含むフォントのコレクションです。フォント セットに含まれる各フォントについて、それが適用される言語を指定できます。これによって、プロジェクトでサポートする各言語に異なるフォントを選択できます。
- **テキスト スタイル** — テキスト スタイルは、テキストの色、およびフォント名、スタイル、サイズといったテキスト属性を定義します。テキスト スタイルは、一般的に、アドバンスト UI またはスイート/アドバンスト UI ウィザード インターフェイスで表示されるテキストに適用されます。

フォント セットは、テキスト スタイルと組み合わせて使用します。テキスト スタイルはフォント セットを参照しますが、オプションでフォント セットのレベルで定義されている様々なフォント属性をオーバーライドすることが可能です。

プロジェクトにスタイルを追加するには、[スタイル] ノードを右クリックしてから、追加するスタイルの種類に応じて、適切なコマンドをクリックします。

詳しくは、次を参照してください:

- カスタム スタイルをウィザード インターフェイスに定義する
- テキスト スタイルをウィザード インターフェイスのテキストに適用する

ブラシ

ブラシは、ウィザード インターフェイスの様々な要素、たとえばウィザード ページやコントロール背景に純色、グラデーション、またはイメージを指定します。

プロジェクトにブラシを追加するには、[ブラシ] ノードを右クリックしてから [ブラシの追加] をクリックします。

詳しくは、次を参照してください:

- カスタム ブラシをウィザード インターフェイスに定義する
- ウィザード インターフェイスのヘッダー、本文、およびナビゲーション領域に背景ブラシを指定する
- ウィザード ページまたは 2 番目のウィンドウのデフォルト 背景ブラシをオーバーライドする

コントロールのテーマ

コントロールのテーマは、コントロールの様々な部分の色（テキストの色、背景の色、および境界線の色）や、クリック済みまたはホバー状態といった、コントロールの様々な状態に使用する色を集めたものです。デフォルトで、使用中のユーザー インターフェイスで使われているすべてのコントロールについて、使用するコントロールのテーマを指定できます。ユーザー インターフェイス内のウィザード ページやウィンドウに含まれる特定のコントロールのテーマをオーバーライドすることもできます。コントロールのテーマを使用すると、インストールのユーザー インターフェイス内のその他のアイテムが 3D 効果なしで平面的に表示されます。これは Windows Store アプリケーションのスタイルに似ています。

プロジェクトにコントロールのテーマを追加するには、[コントロールのテーマ] ノードを右クリックしてから [コントロールのテーマを追加] を選択します。

詳しくは、次を参照してください:

- カスタム コントロール テーマをウィザード インターフェイスに定義する
- テーマをウィザード インターフェイスのコントロールに適用する

ウィザード ページ

アドバンスト UI またはスイート/アドバンスト UI インストールでエンド ユーザーに表示される主要なウィンドウはウィザード ページです。[ウィザード インターフェイス] ビュー の [ウィザード ページ] 領域を使って、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるウィザード ページを参照、編集、追加および削除することができます。

アドバンスト UI またはスイート/アドバンスト UI プロジェクトで使用されているすべてのウィザード ページに適用する設定を構成および編集するには、[ウィザード ページ] ノードをクリックします。このノードに表示されている設定で、寸法、ウィザード ページのタイトル バーで使用されるキャプションなどを指定できます。

[ウィザード ページ] ノードの下の特定のウィザード ページ サブノードでは、必要に応じてページを追加、シーケンス、または削除することができます。また、これらのサブノードでは、様々な異なる種類のコントロールを追加、移動または削除することでウィザード ページのレイアウトをそれぞれ編集することもできます。ウィザード ページのランタイムでの順序を変更するには、シーケンスするページをクリックしてから CTRL+上矢印 または CTRL+下矢印 を押して、そのページを [ウィザード ページ] ノード内で適切な位置に移動させます。

プロジェクトにウィザード ページを追加するには、[ウィザード ページ] ノードを右クリックしてから、追加するページの種類に応じて、適切なコマンドをクリックします。

2 番目のウィンドウ

アドバンスト UI またはスイート/アドバンスト UI インストールの 2 番目のウィンドウはポップアップ ウィンドウで、このウィンドウを使ってエンド ユーザーがコマンドを実行できるようにしたり、またエンド ユーザーに対する質問や情報を表示したりできます。2 番目のウィンドウの例として、FilesInUse ウィンドウが挙げられます。このウィンドウは、既に開かれている 1 つ以上のアプリケーションによって、インストールがアップデートしなくてはならないファイルがロックされていることをエンド ユーザーに通知します。[2 番目のウィンドウ] ノードの下の特定のウィザード ページ サブノードでは、必要に応じてページをプレビューまたは削除することができます。また、これらのサブノードでは、ウィザード ページのレイアウトを編集のように、ウィンドウのレイアウトをそれぞれ編集することもできます。

プロジェクトに 2 番目のウィンドウを追加するには、[2 番目のウィンドウ] ノードを右クリックしてから [空白ウィンドウの追加] を選択します。

最小ウィザード ページ

最小ウィザード ページを使って、フレームを使わない画面でエンド ユーザーにモダンなインストール/アンインストール エクスペリエンスを提供することができます。

最小ウィザード ページがサポートされている最小バージョンは、Windows 8.1 および Windows Server 2012 R2 です。最小ウィザード セットアップは、.Net 4.0 以降が搭載されている他の古いオペレーティング システムでも動作します。

最小ウィザード ページを選択

セットアップに [最小ウィザード ページ] インストールを選択するには、[ウィザード ページ] ノードをクリックします。このノードに表示される設定を使って、インストールの種類に [最小] を選択することができます。

詳しくは、次を参照してください:

- 最小ウィザード ページのカスタマイズ

ウィザード インターフェイスの設定

[ウィザード インターフェイス] ビューの各領域にある様々な設定についての情報は、このビューにある設定をクリックすると表示されるヘルプ ペインを参照してください。

[ウィザード インターフェイス] ビューのツールバー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

[ウィザード インターフェイス] ビューでウィザード ページまたは 2 番目のウィンドウを選択すると、ウィザード インターフェイスのプレビュー ペインのすぐ上に表示されるツールバーに、選択したページまたはウィンドウを変更できるボタンやコントロールが追加表示されます。また、ウィザード ページまたは 2 番目のウィンドウでコントロールを選択したときも、[ウィザード インターフェイス] ビューにこのツールバーが表示されます。

テーブル 189 · [ウィザード インターフェイス] ビューのツールバー

コントロールの名前	アイコン	説明
新しいコントロール		コントロールを選択したウィザード ページまたは二次ウィンドウに追加できます。コントロール ボタンの 1 つを選択するか、コントロール ボタンの横にあるドロップダウン リストで表示されているコントロールの種類を選択します。

テーブル 189 • [ウィザード インターフェイス] ビューのツールバー (続き.)

コントロールの名前	アイコン	説明
新しいコントロール (続き)		以下は、[ラベル] ボタンの横にあるドロップダウン リストで選択できるコントロールの種類です: • **ラベル** – [ラベル] コントロールではテキストが表示されます。このタイプのコントロールは、他のコントロールの横に配置するラベル (テキスト ボックス コントロール) や、ウィザード ページや二次ウィンドウに含める手順や情報に使用します。 • **ハイパーリンク** – ハイパーリンク コントロールは HTML リンクを表示します。このリンクを実行時にクリックすると、ターゲット システム上のデフォルト ブラウザでページが開きます。 • **グループ ボックス** – グループ ボックスは、あるエリア内の複数のコントロールを囲むのに使用します。また、グループ ボックスには、そこに含まれるコントロール間の関係を表現するのに使用できるラベルがあります。 • **フレーム** – ボックスの輪郭を描くことができます。 • **イメージ** – イメージが表示されます。 • **ビルボード** – ISInstallProgress などのプロパティに応答して更新可能なイメージ、または一定時間表示されるように構成できるイメージが表示されますビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。 • **進行状況バー** – 進行状況バーは、ISInstallProgress プロパティなどのプロパティに応答して色が変わるダイナミック バーです。 • **プログレス サークル** – プログレス サークルは、2 つのプロパティ (ISParcelProgress など、現在のパッケージの完了率を示すプロパティと、ISInstallProgress など、インストール全体の完了率を示すプロパティ) に応答して色が変わるダイナミックなサークルです。プログレス サークルの外側では、通常、全体的な進行状況が表示され、内側の進行状況バーでは、通常、パッケージ全体の進行状況が表示されます。

テーブル 189 • [ウィザード インターフェイス] ビューのツールバー (続き.)

コントロールの名前	アイコン	説明
新しいコントロール (続き)		以下は、[テキスト ボックス] ボタンの横にあるドロップダウン リストで選択できるコントロールの種類です: • **テキスト ボックス** – テキスト ボックス コントロールは、エンド ユーザーがテキストや数値を入力できるフィールドです。 • **パスワードボックス** – パスワード ボックス コントロールは、エンドユーザーがパスワードを入力できるフィールドです。 • **リッチ テキスト ボックス** – リッチ テキスト ボックスは、.rtf ファイルからのテキストを表示する読み取り専用のコントロールです。このタイプのコントロールは、通常、使用許諾契約書やリリースノートのウィザード ページで使用されます。
新しいコントロール (続き)		以下は、[ボタン] ボタンの横にあるドロップダウン リストで選択できるコントロールの種類です: • **ボタン** – ボタン コントロールは、エンド ユーザーがクリックしたときにコマンドが実行されるコントロールです。 • **コマンド リンク** – コマンド リンク コントロールでは、ラジオ ボタンのような、相互に排他的なオプションが提供されています。各オプションには、アイコン、アイコンの横にあるコマンド リンク ラベル、および、オプションに関する補足説明 (オプション) があります。 • **チェック ボックス** – チェック ボックス コントロールは、エンド ユーザーが選択またはクリアしてオプションの表示を切り替えできるチェック ボックスを表示します。 • **ラジオ ボタン** – ラジオ ボタン コントロールは、エンド ユーザーがその中から 1 つだけを選択できる、2 つまたはそれ以上のオプションの 1 つです。ラジオ ボタンは、グループ ボックス グループに挿入しなくてはなりません。これは、そのコントロールの一部として機能します。 • **イメージ ボタン** – イメージ ボタン コントロールでは、イメージが表示され、エンド ユーザーがそれをクリックしたときにコマンドが実行されます。

テーブル 189 • [ウィザード インターフェイス] ビューのツールバー (続き.)

コントロールの名前	アイコン	説明
新しいコントロール (続き)		以下は、[コンボ ボックス] ボタンの横にあるドロップダウン リストで選択できるコントロールの種類です: • **コンボ ボックス** — コンボ ボックス コントロールは、定義された値を持つドロップダウン リストを含むボックスです。ボックスには、エンドユーザーがカスタム値が入力できるテキスト ボックスです。 このコントロールをテキストボックスのないドロップダウン リストに変更するには (つまり、エンドユーザーが定義済みの値を選択できるが、カスタム値を入力できるようにする場合)、このコントロールの CBS_DROPDOWNLIST スタイルを False に設定します。 • **リスト ボックス** — リスト ボックス コントロールは、エンドユーザーが定義済みオプション リストから単一のオプションを選択できる、標準のリスト ボックスです。 • **チェック済みリスト ボックス** — チェック済みリスト ボックスは、定義済みの値がチェック ボックスに関連付けられているリスト ボックスです。チェック済みリスト ボックスで、エンドユーザーは、オプションを選択できます (複数可)。また、すべてのオプションを非選択済みのままにしておくこともできます。 • **機能の選択ツリー** — 機能の選択ツリー コントロールは、機能を階層的に編成したアドバンスト UI またはスイート/アドバンスト UI インストールの機能の集まりを含む特別なツリー ビューです。
位置コントロール		選択した 2 つ以上のコントロールの位置を調節できます。

テーブル 189 • [ウィザード インターフェイス] ビューのツールバー (続き.)

コントロールの名前	アイコン	説明
位置コントロール (続き)		以下は、[左揃え] ボタンの横にあるドロップダウン リストで選択できる位置の変更オプションです: • **左揃え** – 2 つ以上の選択したコントロールの左端を、最後に選択したコントロールの左端に合わせて配置します。 • **中央揃え** – 2 つ以上の選択したコントロールの中央を、最後に選択したコントロールの中央に合わせて配置します。 • **右揃え** – 2 つ以上の選択したコントロールの右端を、最後に選択したコントロールの右端に合わせて配置します。 • **上揃え** – 2 つ以上の選択したコントロールの上端を、最後に選択したコントロールの上端に合わせて配置します。 • **中揃え** – 2 つ以上の選択したコントロールの中央点を、最後に選択したコントロールの中央点に合わせて配置します。 • **下揃え** – 2 つ以上の選択したコントロールの下端を、最後に選択したコントロールの下端に合わせて配置します。 • **左右均等配置** – 3 つ以上選択したコントロールを、水平方向に均等配置します。 • **上下均等配置** – 3 つ以上選択したコントロールを、垂直方向に均等配置します。
位置コントロール (続き)		以下は、[幅を揃える] ボタンの横にあるドロップダウン リストで選択できる位置の変更オプションです: • **幅を揃える** – 選択したコントロールの幅を、最後に選択したコントロールの幅に合うように調節します。 • **高さを揃える** – 選択したコントロールの高さを、最後に選択したコントロールの高さに合うように調節します。 • **幅を揃える** – 選択したコントロールの幅を、最後に選択したコントロールの幅に合うように調節します。
デフォルト言語:		このビューのウィザード ページおよび二次ウィンドウで表示する文字列を、プロジェクトで使用する別の言語の文字列に切り替えることができます。選択可能な言語オプションは、[一般情報] ビューの ″セットアップ言語″ 設定で選択されている言語です。 この設定では、プロジェクトのデフォルト言語を変更することもできます。

[ビルボード] ビュー

プロジェクト: *[ビルボード] ビューは、基本の MSI プロジェクトで使用できます。*

InstallScript または InstallScript MSI プロジェクトにおけるビルボード サポートについての詳細は、「[サポート ファイル/ビルボード] ビュー (InstallScript および InstallScript MSI プロジェクト)」を参照してください。

ビルボードをプロジェクトに追加して、インストール処理中にエンド ユーザーに対して情報を提供できます。ビルボードは、エンド ユーザーと連絡を取ったり、広告、教育、およびエンターテインメントを提示するために使用することができます。たとえば、ビルボードを使ってインストール中の製品に含まれる新しい機能の概要や貴社の他の製品についての情報を提供できます。各ビルボードは、貴社のグラフィック担当者がファイル転送の外観を完全にカスタマイズできるファイルです。

プロジェクトに 1 つ以上のビルボードを追加すると、実行時に Windows Installer がシステムに追加している変更について SetupProgress ダイアログ がレポートを提示するのと同時に、ビルボードが表示されます。ビルボードが表示される時間および場所は、[ビルボード] ビューで設定を構成することによって制御できます。

[ビルボード] ビューの設定についての詳細は、次の項目を参照してください:

- ビルボード設定 – これらは、プロジェクト共通のビルボードの設定です。
- Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定 – これらの設定は、中央ペインで Flash ビルボードまたはイメージ ビルボードをクリックすると、[ビルボード] ビューの右側のペインに表示されます。

ビルボード設定

プロジェクト: *[ビルボード] ビューは、基本の MSI プロジェクトで使用できます。*

[ビルボード] ビューの中央ペインで [ビルボード] エクスプローラーをクリックすると、右側のペインに以下の設定が表示されます。これらは、プロジェクト共通のビルボードの設定です。

テーブル 190 ・ ビルボード設定

設定	説明
ビルボードの種類	インストールに使用するビルボードの種類を選択します。選択可能なオプションは以下のとおりです: • **全画面表示、右下に小さい進行状況ボックスを表示** – インストールが標準エンドユーザー ダイアログを表示するときに、全画面の背景も表示します。ファイルの転送中、インストールが全画面背景を使用し、ビルボードを前画面に、また小さい進行状況ボックスを画面の右下に表示します。 • **ウィンドウ表示、標準の進行状況を表示する** – ファイルの転送中、インストールはビルボードを表示する標準サイズのダイアログを表示します。このダイアログの下の部分に、進行状況バーが表示されます。このスタイルの場合、インストールは背景を表示しません。 • **ウィンドウ表示、右下に小さい進行状況ボックスを表示する(ビルボードなし)** – インストールは、ファイル転送中に小さい進行状況ボックスを画面の右下に表示しますが、ビルボードまたは背景は表示されません。 さらに詳しい情報、ならびに各ビルボード タイプのサンプル スクリーンショットは、「基本の MSI プロジェクトにおけるビルボードの種類」を参照してください。

テーブル 190 • ビルボード設定 (続き.)

設定	説明
ビルボードのループ	インストールがファイルの転送が完了するまでイメージ ビルボードをループして、適切な SetupComplete ダイアログを表示するかどうかを指定します。 この設定で [いいえ] を選択して、ビルボードに割り当てた時間よりもファイルの転送に時間がかかった場合、インストールはファイルの転送が終了するまで、最後のイメージ ビルボードを表示し続けます。 この設定で [はい] を選択して、ビルボードに割り当てた時間よりもファイルの転送に時間がかかった場合、インストールは最初のビルボードから再び表示します。必要な場合、ループはファイルの転送が終了するまで継続します。 この設定のデフォルト値は [いいえ] です。 この設定は、Adobe Flash アプリケーション ファイル ビルボードには効果がありません。

Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定

プロジェクト: *[ビルボード] ビューは、基本の MSI プロジェクトで使用できます。*

Flash またはイメージ ビルボードの設定では、表示するファイル、その表示時間、および画面上の表示位置を決定します。これらの設定にアクセスするには、[ビルボード] ビューを開いて、[ビルボード] エクスプローラーから構成するビルボードを選択します。

テーブル 191 • Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定

設定	説明
ファイル名	以下のいずれかを実行します。 • **Adobe Flash アプリケーション ファイル ビルボードの場合** – 選択されたビルボードに使用する Flash アプリケーション ファイル (.swf) へのパスを入力するか、省略記号 (...) ボタンをクリックしてファイルを参照します。 Flash アプリケーション ファイルは、ビデオ、動画、音声、インタラクティブ インターフェイス、ゲーム、テキスト、その他の .swf ファイルがサポートするあらゆる要素で構成されます。Flash ビデオ ファイル (.flv) や MP3 オーディオ ファイルは .swf ファイルに埋め込んで、ファイル転送中にターゲット システム上のローカルで使用できるようにすることが推奨されます。.swf ファイルは Web サイト上に配置された外部ファイルを参照することが可能ですが、この外部実装ではエンド ユーザーがインターネットに接続されていることが必須となります。 • **イメージ ビルボードの場合** – 選択されたビルボードに使用するイメージ ファイル (.bmp、.gif、.jpg、または .jpeg) へのパスを入力するか、省略記号 (...) ボタンをクリックしてファイルを参照します。 動画 .gif ファイルはサポートされていませんので、ご注意ください。ビルボードで動画を使用したい場合は、Adobe Flashアプリケーション ファイル ビルボードの使用をご検討ください。 ***メモ:*** *.swf ファイルの作成に使用した Flash またはその他のツールのバージョンがターゲット システムにインストールされている Flash Player よりも新しい場合、ターゲット システム上で一部の Flash 機能が予定どおりに動作しない可能性があります。*
期間	ビルボードが表示される時間を、秒単位で入力してください。入力する数字は、1 から 32767 まで (9 時間と少し) でなくてはなりません。 期間が実行時の動作に及ぼす影響は、インストールが Flash ビルボードかイメージ ビルボードのどちらを表示するかによって異なります。詳細については、「ビルボードを含む基本の MSI インストールにおける実行時の動作」を参照してください。

テーブル 191 • Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定 (続き.)

設定	説明
元の場所	画面上でビルボードを表示する場所を選択します。選択可能なオプションは以下のとおりです: • 右上 • 左上 • 右下 • 左下 • 中央揃え X および Y 座標は、この点から測定されます。 ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*
X 軸	ビルボードの横位置を "原点" 設定で選択した場所に相対して変更するには、距離をピクセルで指定します。たとえば、ビルボードの原点が左下で、X 座標値が 100 の場合、画面の左側から 100 ピクセル離れたビルボードの左側に表示されます。 ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*
Y 軸	ビルボードの縦位置を "原点" 設定で選択した場所に相対して変更するには、距離をピクセルで指定します。たとえば、ビルボードの原点が左下で、Y 座標値が 100 の場合、画面の下側から 100 ピクセル離れたビルボードの下側に表示されます。 ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*
効果	このビルボードの移動効果を選択します。画面に表示されて、指定した時間の経過後に非表示にするだけではなく、移動効果では、ビルボードとビルボードの変更がよりスムーズに行われます。 ***メモ:*** *この設定は、イメージ ビルボードに適用されますが、Adobe Flash アプリケーション ファイル ビルボードには適用されません。* *また、この設定は [ビルボードの種類] 設定で、* ***[ウィンドウ表示、標準の進行状況を表示する]*** *オプションまたは、* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*

テーブル 191 • Adobe Flash アプリケーション ファイル ビルボードとイメージ ビルボードの設定 (続き.)

設定	説明
背景色	この設定には、現在選択されているビルボードの背景色が表示されます。この色を変更するには、省略記号ボタン (...) をクリックします。背景色に希望する色を選択、またはカスタム色を定義できる [色] ダイアログ ボックスが表示されます。 ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*
タイトル	背景の左上に表示される、このビルボードのタイトルを入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***ヒント:*** *インストールが実行時にタイトルとして表示できる最大文字数は、そのフォント、フォント サイズ、フォントの属性、およびこの設定に指定するタイトル文字列の長さによって異なります。また、ターゲット システムのスクリーン解像度にも左右されます。このため、長いタイトルを指定する場合は、異なるスクリーン解像度を使ってビルボードをプレビューして、タイトル全体が実行時に表示されるかどうかをテストしてください。* ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*
背景スタイル	使用する背景のスタイルを選択します。選択可能なオプションは以下のとおりです: • **グラデーション** – 背景は濃い色から薄い色へ徐々に変化します。 • **純色** – 背景は単一色で表示されます。 ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*
フォント	省略記号 (...) ボタンをクリックして、選択されたビルボードの背景にあるタイトルに使用するフォントを選択します。ターゲットマシンに指定したフォントがない場合、デフォルトのシステム フォントが使用されます。 ***メモ:*** *この設定は、"ビルボードの種類" 設定で* ***[全画面表示、右下に小さい進行状況ボックスを表示する]*** *オプションが選択されている場合のみ有効です。*

[文字列エディター] ビュー

プロジェクト: *[文字列エディター] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[文字列エディター] ビューでは、インストール処理で実行時に表示されるローカライズ可能なすべてのテキスト文字列を、1 ヶ所でまとめて制御することができます。このビューを使って、ボタン テキストから機能の説明まで全ての文字列を編集できます。

[文字列エディター] ビューは、プロジェクト内の言語非依存文字列 ID の一覧と対応する言語固有の値が表示されます。次の一覧は、このビューで処理できるタスクの一部です:

- プロジェクト内のすべての文字列を表示する。
- 文字列を追加、変更、および削除する。
- 表示されている文字列をフィルターして、特定の文字を含む文字列を非表示にする。
- ビュー内の列のサイズを変更、およびその順序を変更する。
- 列のヘッダーをクリックして、ビュー内の行を列ごとに並べ替える。
- 列ヘッダーをグループ ボックス領域 (ビューのボタンの下にある領域) にドラッグ アンド ドロップして、ビュー内の行を階層形式に編成する。
- 文字列 ID と、それに対応する値をテキスト ファイル (.txt) にエクスポートする。
- 翻訳済み文字列を .txt ファイルからプロジェクトにインポートする。
- プロジェクトを検索して、特定の文字列 ID が使用されているすべてのインスタンスを識別する。文字列がプロジェクト内の別の場所で使用されていないか確認する。

[文字列エディター] ビューを使って作業する

[文字列エディター] ビューは、次の要素で構成されます:

- ボタン行とその他のコントロール
- グループ ボックス領域 (ボタン行の下)
- スプレッドシート形式のテーブル

テーブルの各行は、プロジェクトに含まれる各文字列エントリを示します。

次の表では、[文字列エディター] ビューに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 192 • [文字列エディター] ビューのコントロール

コントロールの名前	アイコン	説明
新しい文字列エントリ		新しい文字列エントリを追加できる、[文字列エントリ] ダイアログ ボックスを表示します。
選択した文字列の編集		選択した文字列エントリを編集できる、[文字列エントリ] ダイアログ ボックスを表示します。
選択した文字列の削除		選択した行 (複数可) を削除します。
すべてのグループを展開する		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を表示します。
すべてのグループを折りたたむ		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を隠します。
文字列の検索		文字列のインスタンスを検索できる、[検索] ダイアログ ボックスを表示します。このダイアログ ボックスでは、大文字と小文字を区別するかどうかなどの条件を指定できます。
次を検索		指定した文字列の次の位置を検索します。
検索/置換		文字列のインスタンスを検索して、それを新しい文字列と置換できる、[置換] ボックスを表示します。このダイアログ ボックスを使って、特定の大文字を含む文字列を検索、または削除するかなどの条件を指定できます。
プロジェクト内で選択した文字列を検索		行が選択されている文字列の全てのインスタンスをプロジェクト全体で検索し、検索結果を [出力] ウィンドウに表示します。
文字列のエクスポート		特定の言語の全ての文字列をテキスト ファイル (.txt) にエクスポートします。この .txt ファイルを、翻訳されたテキストで更新することができる翻訳者に渡すことができます。
文字列のインポート		プロジェクトにインポートする文字列を含むテキスト ファイル (.txt) を選択します。また、それらの文字列の言語も指定できます。
検索グリッド		この検索ボックスで指定した文字列に従って、[文字列エディター] ビューに表示される文字列をダイナミックにフィルターします。このボックスに文字列を入力すると、それを含まない行すべてが非表示となります。

テーブル 192 - [文字列エディター] ビューのコントロール (続き.)

コントロールの名前	アイコン	説明
デフォルト言語:		プロジェクトのデフォルト言語を表示、および変更できます。
文字列エディターのヘルプ		[文字列エディター] ビューのヘルプを表示します。
グループ化する列のヘッダーをここにドラッグ		このグループ ボックス領域を使って、ビュー内の行をグループ分けします。ビューでは、列のヘッダーをグループ ボックスにドラッグ アンド ドロップするだけで複数階層にグループ化することができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。 詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

次の表は、[文字列エディター] ビューの各列について説明します。

テーブル 193 - [文字列エディター] ビューの列

列	説明
言語	この列には、文字列エントリの言語が表示されます。
Identifier	この列には、その文字列の言語非依存 ID が含まれます。プロジェクトに含まれる各文字列 ID は、1 つまたは複数の値にリンクされています。
値	この列には、ランタイム文字列が表示されます。 ***プロジェクト:*** *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、一部の文字列値は角括弧でかこまれた Windows Installer プロパティを含みます (例、***Install [ProductName]***)。実行時に、このプロパティと括弧はプロパティ値で置き換えられます。* *これらの同じプロジェクト タイプに含まれる文字列値の中には、中括弧でかこまれたフォント情報を含むものもあります (例、***{&MSSansBold8}OK***)。フォント情報は、実行時に文字列を表示するのに使用するスタイルの詳細を示します。*
コメント	この列には、文字列エントリについての内部メモが含まれます。コメントは実行時には表示されません。
Modified	この列には、文字列エントリが最後に更新された日時が表示されます。

[メディア] ビュー

プロジェクト: *[メディア] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

インストール プロジェクト作成の最後の手順では、インストールのビルドとテストを行います。InstallShield は、InstallShield 内からインストールをテストできる機能、および様々な選択可能なメディタイプを提供します。

パス変数

プロジェクト: *[パス変数] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*
- *MSIX*

プロジェクトを移動したり、ディレクトリ構造を変更するたびに各ソースファイルのパスを変更する必要がないように、パス変数を使って共通して使用するパスを中央ディレクトリに定義することができます。

アップグレード

プロジェクト: *[アップグレード] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[アップグレード] ビューでは、.msi データベースの Upgrade テーブル内の設定を分かりやすく統合された方法で追加または作成することができます。

リリース

プロジェクト: *[リリース] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[リリース] ビューでは、インストールの異なる構成の構築、ユーザー インターフェイスのテスト、またはトライアル実行用のインストールの起動ができます。

パッチのデザイン

プロジェクト: *[パッチのデザイン] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューを利用して、Windows Installer パッチ パッケージ (.psp) を作成し、エンド ユーザーのシステム上にあるアプリケーションをアップデートすることができます。

[パス変数] ビュー

プロジェクト: *[パス変数] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *QuickPatch*
- *スイート/アドバンスト UI*
- *MSIX*
- *MSIX 変更パッケージ*

ソースファイルをインストール プロジェクトにリンクする従来の方法は、ハードコード化されたパスを使用して、そのファイルへのリファレンスを作成することでした。たとえば、ソースファイルをインストールに含める場合は、**C:¥Work¥Files** にある **Program.exe** のように指定していました。

ハードコード化したパスの使用

ハードコード化したパスを使用すると、ディレクトリからソースファイルを関連付けるときに毎回完全パスを入力する必要がありました。ファイルを別のディレクトリに移動すると、インストール プロジェクトで使用できるようにハードコード化したパスを変更する必要があります。インストールに含まれているソース ファイルの数が少ない場合は、このことは問題にならない場合もあります。残念ながら、インストールによっては、数千のファイルを含んでいることがあり、このような場合、フォルダー構造を変えたり、別のマシンにプロジェクトを移行すると、これらすべてのファイルを再度マッピングし直す必要があります。

パス変数の使用

プロジェクトを移動したり、ディレクトリ構造を変更するたびに各ソースファイルのパスを変更する必要がないように、パス変数を使って共通して使用するパスを中央ディレクトリに定義することができます。上記の例において、アプリケーションのすべてのソース ファイルを **C:¥Work¥Files** の下のさまざまなサブフォルダーに入れる場合は、Files フォルダーをポイントする 1 つの変数 (*<MyFiles>*) を作成することができます。**C:¥Work¥Files¥Images** にあるファイルをインストールに含める場合は、**<MyFiles>¥Images** と入力します。ファイルを **D:¥Work¥Files** に移動する場合は、変数 *<MyFiles>* に移動し、ポイントするフォルダーを変更します。

すべてのパス変数は、[パス変数] ビューで表示、修正することができます。ダイアログ エディター、ダイナミックファイルリンク、リリースフォルダーなど、ソースファイルにリンクされている InstallShield のほとんどすべてのフォルダーのパス変数を使用できます。パス変数を自分で入力するかわりに、パスを参照するたびに InstallShield の推奨値を使用することができます。

メモ: *パス変数は、インストール プロジェクトの開発中に使用されます。これらのパスは、アプリケーションがインストールされるターゲットマシンには適用されません。代わりに、インストール プロジェクトのソースファイルにリンクするために使用されます。プロジェクトをビルドする際、これらのリンクが評価され、ポイント先のファイルがインストールに組み込まれます。*

パス変数の種類

ユーザーは、4 種類のパス変数を使用することができます。それぞれの型は、互いに部分的に機能が異なります。使用するパス変数の種類にかかわらず、変数名は InstallShield 内で同じように提供されます。

テーブル 194 • パス変数の種類

変数型	説明
	デフォルトパス変数は、最も一般的に使用されるいくつかのフォルダーにポイントされるパス変数です。他の型のパス変数と違い、これらの変数は InstallShield では編集できません。詳細については、「定義済みパス変数」を参照してください。
	レジストリベースのパス変数の値は、作成済みのレジストリキーから呼び出されます。レジストリキーの作成後に、パス変数をこのキーに設定する必要があります。詳細については、「レジストリ パス変数」を参照してください。
	環境パス変数は、システムの環境変数の値に基づきます。環境変数パス変数に既存の環境変数を設定できます。詳細については、「環境パス変数」を参照してください。

テーブル 194 • パス変数の種類 (続き.)

変数型	説明
	標準、またはユーザー定義パス変数は、InstallShield によって定義されます。たとえば <MyFiles> などのパス変数の値を C:¥Work¥Files と指定することができます。これらの変数は、レジストリ、システムパスなど、外部のソースには依存しません。詳細については、「標準パス変数」を参照してください。

オプションで、既存のスタティックリンクを、ソース パス変換ウィザードを持つパス変数に変換することもできます。このウィザードは、インストール プロジェクトのスタティックリンクをスキャンし、これらのリンクをパス変数に変換します。これにより、プロジェクトの移植性を高めることができます。

[パス変数] ビューを使って作業する

[パス変数] ビューは、次の要素で構成されます:

- ボタン行とその他のコントロール
- グループ ボックス領域 (ボタン行の下)
- プロジェクトで定義されているパス変数の一覧

次の表では、[パス変数] ビューに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 195 • [パス変数] ビューのコントロール

コントロールの名前	アイコン	説明
新しいパス変数		新しい標準パス変数をプロジェクトに追加します。 レジストリ パス変数または環境変数を作成するには、このボタンの隣にある矢印をクリックしてから、適切なコマンドを選択します。詳細については、「カスタム パス変数の作成と定義」を参照してください。
選択されたパス変数の削除		選択された変数 (複数可) を削除します。
リフレッシュ		ビュー内の変数リストを更新します。
すべてのグループを展開する		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を表示します。
すべてのグループを折りたたむ		グループを使って行を階層形式に編成する場合に、グループに含まれるすべての行を隠します。
グループ ボックスの表示		このビューのボタン行の下にある [グループ ボックス] 領域の表示/非表示を切り替えます。

テーブル 195 - [パス変数] ビューのコントロール (続き.)

コントロールの名前	アイコン	説明
検索グリッド		この検索ボックスで指定した文字列に従って、[パス変数] ビューに表示されるパス変数をダイナミックにフィルターします。このボックスに文字列を入力すると、それを含まない行すべてが非表示となります。
[パス変数] ビュー ヘルプ		[パス変数] ビューのヘルプを表示します。
グループ化する列のヘッダーをここにドラッグ		このグループ ボックス領域を使って、ビュー内の行をグループ分けします。ビューでは、列のヘッダーをグループ ボックスにドラッグ アンド ドロップするだけで複数階層にグループ化することができます。ビュー内の行は、グループ ボックスの配列に従って階層構造で表示されます。 詳細については、「様々なビューで、[グループ ボックス] 領域を使って作業する」を参照してください。

次の表は、[パス変数] ビューの各列について説明します。

テーブル 196 - [パス変数] ビューの列

列	説明
名前	この列には、変数の名前を入力します。この場合、山かっこを入力する必要はありませんが、パス変数は、パス内で使用される際は常に山かっこで囲まれて表示されます。 たとえば、変数の名前が *MyRegVar* で、コンポーネントのファイルがその変数の値に含まれているフォルダーにリンクしている場合、このコンポーネントの リンク先フォルダーは **<MyRegVar>** と表示されます。

テーブル 196 • [パス変数] ビューの列 (続き.)

列	説明
定義された値	この列で、パス変数の値を定義します。 **標準パス変数の場合** 標準変数の場合、変数がポイントするディレクトリを入力します。 ***メモ•*** *また、参照先となる他のパス変数の名前を山括弧で囲むことにより、定義済みの値の中でそのパス変数を参照することができます。たとえば、*C:¥ *という値を持つ MyRoot というパス変数がある場合、そのパス変数を Games という別の変数のパス変数定義で参照することができます。Games 変数の実際のパスは* C:¥Programs¥GameFiles *のようになりますが、Games を* <MyRoot>¥Programs¥GameFiles *と定義することができます。ただし、パス変数を自己参照しようとした場合は、使用した文字列そのものが代わりに使用されます。たとえば、Games を* <MyRoot>¥Programs¥<Games> *と定義すると、実際には 、Games は* C:¥Programs¥<Games> *のように定義されます。* **レジストリ パス変数の場合** 完全なレジストリ キーを入力します。 最後の「サブキー」は、フォルダーが含まれる値名にします。たとえば、*MyRegVar* を次のように定義します。 HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey¥TestValue TestKey には次のサブキーと値があると想定します。 [HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey] @="C:¥¥MyPath1" "TestValue"="C:¥¥MyPath2" [HKEY_LOCAL_MACHINE¥Software¥TestKey¥TestValue] @="C:¥¥MyPath3" HKEY_LOCAL_MACHINE¥TestKey には TestValue というサブキーがありますが、*MyRegVar* は値 TestValue を指し、現在値は C:¥MyPath2 になります。(ただし、TestValue という値が存在しない場合、InstallShield はサブキー HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey¥TestValue のデフォルト値 (C:¥MyPath3) を読み取ります。) **環境パス変数の場合** 環境パス変数について、[環境] ダイアログ ボックスに表示されているとおりの変数名を入力します。
現在の値	この列は読み取り専用で、パス変数がポイントする実際のパスを含んでいます。環境およびレジストリパス変数は InstallShield の外部で定義されることから、この値を参照すると便利です。

テーブル 196 • [パス変数] ビューの列 (続き.)

列	説明
テスト値	 ***プロジェクト•*** *この列は、次のプロジェクトの種類に適用します:* • *基本の MSI* • *DIM* • *InstallScript MSI* • *マージ モジュール* この列を使って、レジストリ パス変数が解決される時にハードコード化されたパスを入力することができます。リリースでテスト値を使用するには、リリース ウィザードの [詳細設定] パネルで **[パス変数のテスト値を使用する]** オプションを選択する必要があります。デフォルトではこのオプションは選択されていません。このオプションを選択すると、すべてのレジストリパス変数にテスト値が使用されます。選択しないとテスト値は一切使用されません。
Type	この列は、変数の種類 (定義済み、標準、環境、またはレジストリ) を表示します。 この列のフィールドをクリックして、パス変数の種類を別の種類に変更できます。定義済みパス変数の種類は変更できません。

ヒント• *ビルド時に特定のリリースにおけるカスタム標準パス変数の値、環境変数、またはレジストリパス値をオーバーライドするには、そのリリースの [ビルド] タブにある ″パス変数のオーバーライド″ 設定を使います。詳細については、「リリースのカスタム パス変数の値をオーバーライドする」を参照してください。*

[アップグレード] ビュー

プロジェクト• *[アップグレード] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

[アップグレード] ビューでは、.msi データベースの Upgrade テーブル内の設定を分かりやすく統合された方法で追加または作成することができます。

アップグレード アイテムを追加するには、[Windows Installer セットアップのアップグレード] ノードを右クリックします。選択可能なオプションは、以下のとおりです。

テーブル 197 • [アップグレード] ビューのオプション

オプション	説明
メジャー アップグレード項目	メジャー アップグレードは、製品の既存インストールを効果的にアンインストールしてから最新バージョンをインストールします。メジャーアップグレードは、製品のメジャーバージョン番号を変更する場合 (たとえばバージョン 1.1.0 から 2.0.0 へのアップグレード) と変更しない場合にかかわらず、根本的なインストール アーキテクチャの変更に適しています。次のどれかが当てはまる場合、メジャー アップグレードが必要です。 • 既存のコンポーネントのコンポーネント コードが変更されたか、コンポーネントが製品ツリーから削除された場合。 • 既存コンポーネントのキー ファイルが変更された場合。 • 既存の機能が製品ツリー内で移動、または削除された場合。 • .msi ファイルの名前が変更されました。 詳細については、「メジャー アップグレードを作成する」を参照してください。
マイナー アップグレード項目	スモールアップデートとマイナーアップグレードは基本的には同じですが、スモールアップデートでは製品バージョンが変更されませんが、マイナーアップグレードでは変更されるという点で異なります。マイナー アップグレードは既存のアプリケーションを上書きインストールしますが、メジャーアップグレードは製品の既存インストールを効果的にアンインストールしてから新しい製品バージョンをインストールします。 マイナー アップグレードをインストールするのに必要な機能は Setup.exe 起動プログラムに含まれています。詳細については、「マイナー アップグレードの実行時の動作」を参照してください。 詳細については、「マイナー アップグレードを作成する」を参照してください。
自動アップグレード アイテム	 ***メモ:*** *[自動アップグレード アイテム] オプションは将来的な製品コード変更の可能性を考慮し、メジャー アップグレードとマイナー アップグレードの両方を処理します。アップグレードの種類の違いが良く分からない場合および、または特にこだわらない場合はこのアップグレードオプションを選択してください。* 自動アップグレードアイテムを追加したとき、以前のインストールを無事にアップグレードするためにはどの設定が必要かをビルドエンジンが判断します。このオプションではその他の詳細設定を行う必要はありません。詳細については、「InstallShield を構成してアップグレードの種類を自動判別する」を参照してください。

テーブル 197 • [アップグレード] ビューのオプション (続き.)

オプション	説明
すべてのアイテムを検証	このオプションを利用して最新版のリリースを検証したり、別のリリースと検証比較するために特定のパッケージを参照することができます。詳細については、「アップグレード、パッチ、および QuickPatch パッケージを検証する」を参照してください。 ***メモ•*** *特定のビルド済みアップグレード項目を必要に応じて検証するには、その項目を右クリックしてから [項目の検証] を選択します。* ***ヒント•*** *[アップグレード] ビューで変更を行った後は必ず、検証を実行する前にパッケージの再ビルドを行ってください。*

[共通] タブ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

[アップグレード] ビューにある [Windows Installer セットアップのアップグレード] アイテムの [共通] タブには、メジャーおよびマイナー アップグレードの両方のグローバル設定が含まれています。

スモール/マイナーアップグレードの設定

スモールアップデートとマイナーアップグレードは基本的には同じですが、スモールアップデートでは製品バージョンが変更されませんが、マイナーアップグレードでは変更されるという点で異なります。マイナー アップグレードは既存のアプリケーションを上書きインストールしますが、メジャー アップグレードは製品の既存インストールを効果的にアンインストールしてから新しい製品バージョンをインストールします。

マイナー アップグレードをインストールするのに必要な機能は Setup.exe 起動プログラムに含まれています。詳細については、「マイナー アップグレードの実行時の動作」を参照してください。実行時、ターゲットシステム上に製品の以前のバージョンが存在することを Setup.exe が検出した場合の動作を指定するには、適切なオプションを選択してください:

テーブル 198 ・ スモールとマイナー アップグレードのオプション

オプション	説明
Disable	このオプションを選択すると、アップグレード シナリオを検出してセットアップの最新バージョンがアップグレード モードで実行していることを確認する必要があります。
ユーザーに確認のプロンプトを表示せずに、アップグレードをインストールする	このオプションは自動的にマイナーアップグレードモードでセットアップを開始します。
プロンプト	このオプションを選択すると、「続行しますか?」ダイアログが表示されます。[はい] を選択した場合、セットアップはアップグレード モードで開始します。[いいえ] を選択した場合、アップグレード モードでセットアップがキャンセルされます。

メジャーアップグレードの設定

メジャー アップグレードを実行するとき、アップグレードの処理方法を選択することができます。次のオプションから選択できます。

テーブル 199 ・ メジャーアップグレード実行のためのオプション

オプション	説明
新規セットアップをインストールする前に、以前のセットアップを完全にアンインストールする。	これは最も信頼性の高い設定ですが、効率は良くありません。最初にすべてのファイル、レジストリ エントリ、ショートカット、そして古いセットアップの設定を削除します。その後、セットアップの最新バージョンから新しいデータを適用します。
セットアップをインストールし、不要なファイルを削除する	これは最も効率的な設定ですが、決まった作成規則に従う必要があります。このオプションは、最初にセットアップの最新バージョンをインストールしてから不要なファイル、レジストリ エントリ、ショートカット、および設定を削除します。 ***注意 ・*** *不要なリソースの削除には、コンポーネントの参照カウントが正確であることが重要です。参照カウントはコンポーネント レベルで発生することに注意してください。したがって、コンポーネントを削除したり、関連リソースを別のコンポーネントへ移動する時は注意を払う必要があります。既存リソースを別のコンポーネントへ移動した場合にはこのオプションを選択しないでください。*

テーブル 199 • メジャーアップグレード実行のためのオプション (続き.)

オプション	説明
以前のファイルの削除に失敗した場合、すべての変更を元に戻す。	アップグレードに失敗した場合、マシンは以前の正常な状態に戻ります。このオプションは以前のバージョンのアンインスールや、最新バージョンのインストールによって行われた変更を元に戻します。

メモ• *この設定は [インストール]-[実行] シーケンスの RemoveExistingProducts アクションのシーケンスを制御します。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「RemoveExistingProducts」を参照してください。*

[詳細] タブ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

プロジェクト固有の違いについては、必要に応じて記述されています。

[詳細] タブには、共通設定の他にもメジャーとマイナー アップグレードの特定の設定が表示されます。

テーブル 200 • [アップグレード] ビューにある [Windows Installer セットアップ アイテムのアップグレード] アイテムの詳細設定

プロパティ	説明
アップグレード コード	この設定は、製品が所属する製品ファミリを識別する一意な GUID です。アップグレード コードは、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があり、これを使って、Windows Installer がインストール済みの製品の関連バージョンを検索します。 この設定の構成方法については、「アップグレード コードを設定する」を参照してください。 アップグレード コードがアップグレードでどのように使用されるかについては、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。

テーブル 200 • [アップグレード] ビューにある [Windows Installer セットアップ アイテムのアップグレード] アイテムの詳細設定 (続き.)

プロパティ	説明
アップグレード時	Setup.exe がターゲット システム上で製品の以前のバージョンを検出したときに発生する動作を指定します。選択可能なオプションは以下のとおりです: • **無効**—以前のバージョンが存在する場合、Setup.exe はエンド ユーザーに対してアップグレードが行われることを通知するメッセージ ボックスを表示しません。さらに、Setup.exe は REINSTALLMODE および REINSTALL プロパティを設定しないため、これらのプロパティを別の方法で設定する必要があります。そうでない場合、Windows Installer は .msi データベースのキャッシュされたコピーを使って、新しいバージョンに含まれている変更を無視します。 • **ユーザーに確認のプロンプトを表示せずに、アップグレードをインストールする**—以前のバージョンが存在する場合、Setup.exe は REINSTALLMODE を **voums** に、REINSTALL プロパティを **ALL** に設定します。その結果、インストールがアップグレード モードで実行します。以前のバージョンが存在しなかった場合、インストールは初回インストールとして動作します。 • **プロンプト**—以前のバージョンが存在する場合、Setup.exe はエンド ユーザーに対してアップグレードが行われることを通知するメッセージ ボックスを表示します。このメッセージ ボックスを使って、エンド ユーザーはアップグレードを続行するか、アップグレードしないで終了するかを選択できます。エンド ユーザーが続行を選択した場合、Setup.exe は REINSTALLMODE を **voums** に、REINSTALL プロパティを **ALL** に設定し、インストールがアップグレード モードで実行します。これがデフォルトの値です。

テーブル 200 • [アップグレード] ビューにある [Windows Installer セットアップ アイテムのアップグレード] アイテムの詳細設定 (続き.)

プロパティ	説明
スタイル	インストールがターゲット システム上で製品の以前のバージョンが存在することを検出したときに、メジャー アップグレードをどのように進めるのかを指定します。選択可能なオプションは以下のとおりです: • **アンインストールを完了してから、再インストール** – これは最も信頼性の高い設定ですが、効率は良くありません。このオプションは、最初にすべてのファイル、レジストリ エントリ、ショートカット、そして古いインストールの設定を削除します。その後、インストールの最新バージョンから新しいデータを適用します。 • **インストールしてから、未使用のファイルを削除** – これは最も効率的な設定ですが、一定のオーサリング規則に従う必要があります。このオプションは、最初に新しいバージョンをインストールしてから、新しいバージョンには不要なすべてのファイル、レジストリ エントリ、ショートカットおよび設定を削除します。 • **インストールしてから、ロールバックで未使用のファイルを削除** – これは最も効率的な設定ですが、決められたオーサリング規則に従う必要があります。このオプションは、最初に新しいバージョンをインストールしてから、新しいバージョンには不要なすべてのファイル、レジストリ エントリ、ショートカットおよび設定を削除します。 アップグレードが失敗した場合、[ロールバック] オプションがターゲット システムを以前の良好な状態に戻します。このオプションは以前のバージョンのアンインスールや、最新バージョンのインストールによって行われた変更を元に戻します。 ***注意 •*** *最初にインストールしてから、不要なリソースを削除するオプションの両方について、不要なリソースの削除は、コンポーネントの参照カウントが正確であることが前提となります。参照カウントはコンポーネント レベルで発生します。したがって、コンポーネントを削除したり、関連リソースを別のコンポーネントへ移動する時は注意を払う必要があります。既存リソースを異なるコンポーネントに移動させている場合、これらの「インストールしてから削除する」オプションは使用できません。*

メジャー アップグレード アイテム: [共通] タブ

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

インストールの製品コードとアップグレードを行うインストールの製品コードとが異なる場合、メジャーアップグレードが必要です。製品コードの変更が必要かどうかを判断する方法については、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。

実行時、メジャーアップグレードはアプリケーションの最新バージョンをインストールする前に以前のバージョンを事実上アンインストールします。

[全般] タブには、メジャー アップグレード項目で頻繁に利用される設定が表示されます。さらに詳しい設定が必要な場合は、[詳細] タブを利用してください。

メジャー アップグレード

メジャーアップグレードはアプリケーションの以前のバージョンを検出するためにアップグレードコードを利用します。インストールのアップグレードコードはインストールを特定の製品ファミリーにグループ分けします。製品ファミリー内の特定のインストールをターゲットにする場合、このアップグレード項目の製品バージョン属性を構成することができます。

テーブル 201 ・ メジャー アップグレード アイテムのオプション

オプション	説明
[アップグレード コード] を共有する製品	メジャー アップグレード アイテムにこのオプションを選択すると、InstallShield が [アドバンスト] タブにあるこの設定の値をプレースホルダー値 **{00000000-0000-0000-0000-000000000000}** に設定します。InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、適切なアップグレード コード値を使用します。
別のアップグレードコードを持つ製品	メジャー アップグレード アイテムにこのオプションを選択すると、このメジャー アップグレード アイテムがターゲットとするベース パッケージのアップグレード コードを指定するか、省略記号ボタン (...) をクリックしてパッケージを参照することができます。パッケージを参照した場合、InstallShield は選択されたパッケージのアップグレード コードを識別し、そのコードをこの設定で使用します。

製品バージョン

ターゲットとする、または製品範囲を定義するため、特定の製品バージョンを選択します。以下の選択肢があります:

テーブル 202 ・ 製品バージョンに関連付けられたオプション

オプション	説明
以前のバージョン	ターゲット システム上の製品を、現在開いているプロジェクトの製品バージョンよりも古い場合にアップデートするには、このオプションを選択します。 メジャー アップグレード アイテムにこのオプションを選択すると、InstallShield が [アドバンスト] タブにある "最大バージョン" 設定の値をプレースホルダー値 *****ALL_VERSIONS***** に設定します。InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、現在開いているプロジェクトの製品バージョンを使用します。

テーブル 202 • 製品バージョンに関連付けられたオプション (続き.)

オプション	説明
特定の範囲のバージョン内: (最小バージョン – 最大バージョン)	アップグレードがターゲットとする製品バージョンの範囲を指定するには、このオプションを選択します。次に、左側のボックスに最小バージョン、右側のボックスに最大バージョンを指定します。[バージョン範囲を含める] チェックボックスを使って、指定されたバージョン番号をターゲット バージョンの範囲に含める、または除外することもも可能です。 たとえば、最小バージョンが 1.00 で最大バージョンが 4.00 の時、バージョン番号が 1.00 から 4.00 の間である、指定した製品ファミリー内の既存インストールすべてをアップグレードするインストールを作成します。[バージョン範囲を含める] チェックボックスを選択すると、バージョン検索でバージョン 1.00 と 4.00 が確認されます。それらのバージョンが存在する場合、アップグレードが行われます。
バージョン範囲を含める	(指定された最小と最大バージョン範囲内のバージョン番号だけでなく) アップグレードがターゲットとするバージョン範囲の最小および最大バージョンの両方も含める場合、このチェックボックスを選択します。 たとえば、最小バージョンが 1.00 で最大バージョンが 4.00 の時、バージョン番号が 1.00 から 4.00 の間である、指定した製品ファミリー内の既存インストールすべてをアップグレードするインストールを作成します。[バージョン範囲を含める] チェックボックスを選択すると、バージョン検索でバージョン 1.00 と 4.00 が確認されます。それらのバージョンが存在する場合、アップグレードが行われます。 最小バージョン番号または最大バージョン番号のみを含めるとき、両方ではなく片方だけの場合は、[アドバンスト] タブの ″最小バージョンを含める″ または ″最大バージョンを含める″ 設定を使用します。 このチェックボックスは、**[特定の範囲のバージョン内]** オプションが選択されている場合に使用できます。
特定のバージョンを指定	製品の以前のバージョンを 1 つだけターゲットとする場合、このオプションを選択してから、適切なバージョン番号を入力します。

メジャー アップグレード アイテム: [詳細] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *MSI データベース*
- *トランスフォーム*

[詳細] タブには、[共通] タブにある設定に加え、さらに詳細なレベルの構成を行うことができます。プロパティグリッドの各設定をクリックして次の設定を構成します:

テーブル 203 • メジャー アップグレード アイテムの詳細プロパティ

プロパティ	説明
アップグレード コード	この設定は、製品が所属する製品ファミリを識別する一意な GUID です。アップグレード コードは、関連製品ファミリの異なるバージョンおよび言語にわたって統一されている必要があり、これを使って、Windows Installer がインストール済みの製品の関連バージョンを検索します。 メジャー アップグレード項目の [共通] タブで **[アップグレード コードを共有している製品]** を選択すると、InstallShield が [詳細] タブにあるこの設定の値をプレースホルダー値 **{00000000-0000-0000-0000-000000000000}** に設定します。InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、適切なアップグレード コード値を使用します。 この設定の構成方法については、「アップグレード コードを設定する」を参照してください。 アップグレード コードがアップグレードでどのように使用されるかについては、「メジャー アップグレード、マイナー アップグレード、およびスモール アップデートの違い」を参照してください。
最小バージョン	このアップグレードがアップデートする最小製品バージョンを入力します。バージョン番号は、*aaa.bbb.ccccc* という形式を使用します。ここで、 *aaa* はメジャーバージョン番号、*bbb* はマイナーバージョン番号、そして *ccccc* はビルド番号を表します。*aaa* と *bbb* の最大値は 255 です。*ccccc* の最大値は、65,535 です。 ***メモ:*** *製品バージョン比較を行なう際、最初の 3 つのバージョン フィールドのみが利用されます。Windows Installer は 4 番目のフィールドのみが異なる製品バージョンを区別することができません。詳細については、「製品バージョンを指定する」を参照してください。* デフォルトで、各プロジェクトには ISPreventDowngrade メジャー アップグレード アイテムが含まれます。これを使って、エンド ユーザーが製品の現在のバージョンで同じ製品の将来のメジャー バージョンを上書きできないように防ぐことができます。このデフォルト メジャー アップグレード アイテムについて、"最小バージョン" 設定のデフォルト エントリは次のプレースホルダー値です: ***ALL_VERSIONS*** InstallShield は、ビルド時に生成する前述の .msi パッケージのプレースホルダー値の代わりに、現在開いているプロジェクトの製品バージョンを使用します。ISPreventDowngrade のサポートに関する詳細は、「現在のインストールによる同製品の将来のメジャー バージョンの上書きを防ぐ」を参照してください。

テーブル 203・メジャー アップグレード アイテムの詳細プロパティ (続き.)

プロパティ	説明
最小バージョンを含める	この設定は、作成するアップグレードがサポートする製品バージョンの範囲を特定する場合に便利です。最小バージョン番号を範囲に含めるには、[はい] を選択します。
最大バージョン	このアップグレードがアップデートする最大製品バージョンを入力します。バージョン番号は、*aaa.bbb.ccccc* という形式を使用します。ここで、 *aaa* はメジャーバージョン番号、*bbb* はマイナーバージョン番号、そして *ccccc* はビルド番号を表します。*aaa* と *bbb* の最大値は 255 です。*ccccc* の最大値は、65,535 です。 ***メモ:*** *製品バージョン比較を行なう際、最初の 3 つのバージョン フィールドのみが利用されます。Windows Installer は 4 番目のフィールドのみが異なる製品バージョンを区別することができません。詳細については、「製品バージョンを指定する」を参照してください。* メジャー アップグレード項目の [共通] タブで **[以前のバージョン]** を選択すると、InstallShield が [アドバンスト] タブにあるこの設定の値を次のプレースホルダー値に設定します: *****ALL_VERSIONS***** InstallShield は、ビルド時に生成する .msi パッケージのプレースホルダー値の代わりに、現在開いているプロジェクトの製品バージョンを使用します。
最大バージョンを含める	この設定は、作成するアップグレードがサポートする製品バージョンの範囲を特定する場合に便利です。最小バージョン番号を範囲に含めるには、[はい] を選択します。
言語	アップグレードがターゲットとする、またはターゲットとしない言語のリストをコンマ区切りで指定してから、"指定した言語を除外する" 設定を使ってこの設定で構成された言語を除外するかどうかを指定します。 言語識別子 は 10 進法で指定します。たとえば、英語 (UK) は **1033** です。
指定した言語を除外する	"言語" 設定で識別された言語をインストールのターゲットにするか、ターゲットにしないかを指定します。選択可能なオプションは以下のとおりです: • **いいえ**—ターゲット システム上の製品の言語が "言語" 設定で指定されている言語の 1 つと一致した場合で、バージョン番号が一致するなどの他の要件も満たしている場合、インストールはアップグレードを実行します。 • **はい**—ターゲット システム上の製品の言語が "言語" 設定で指定されている言語のいずれにも一致しない場合で、バージョン番号が一致するなどの他の要件を満たしている場合、インストールはアップグレードを実行します。

テーブル 203・メジャー アップグレード アイテムの詳細プロパティ (続き.)

プロパティ	説明
検出のみ	インストールがアップグレードを実際に実行する前に、それが必要かどうかを検出するかどうかを指定します。デフォルト値は [いいえ] です。 Windows Installer がターゲット システムに関連製品がインストール済みであることを検出した場合、"検出プロパティ" 設定で指定されたプロパティにその関連製品の製品コードを追加します。
検出プロパティ	Windows Installer がターゲット システムに関連製品がインストール済みであることを検出した場合、この設定で指定されたプロパティにその関連製品の製品コードを追加します。 条件ステートメントで指定したプロパティを利用してインストールの流れをコントロールしたり、インストールを完全に停止することができます。 Windows Installer は、FindRelatedProducts アクションがインストール シーケンスで実行するときにこのプロパティを設定します。
指定した機能だけを削除する	インストールがターゲット システムから製品の以前のバージョンすべてを削除するように指定する場合、この設定を空白のままにします。 インストールがターゲットシステムから製品全体を削除するのではなく、製品の以前のバージョンの一部の機能のみを削除するように設定する場合、削除する機能の名前をコンマ区切りのリストで指定します。インストールは、空白文字列と評価を行うフォーマット済みテキストを指定した場合、インストールはいずれの機能も削除しません。 ***注意**・この設定に値を指定した場合、インストールが製品の以前のバージョンを完全にアンインストールすることはありません。したがって、[プログラムの追加と削除] には 2 つのエントリが残ります。*
失敗時に続行する	以前のバージョンのアンインストールが失敗した場合でもインストールを続行するかどうかを指定します。選択可能なオプションは以下のとおりです: ・ **はい**—以前のバージョンのアンインストールが成功したかどうかにかかわらず、新しいバージョンのインストールが発生します。 ・ **いいえ**—以前のバージョンのアンインストールが成功した場合のみ、新しいバージョンのインストールが発生します。これがデフォルトの値です。
機能の状態を移行する	インストールが現在インストール済み製品の機能の状態を検出して、アップグレードを適用する際に同じ機能の状態を使用するかどうかを指定します。 この設定は、製品の新しいバージョンの機能ツリーが元のバージョンの機能ツリーとほとんど変更が無い場合に便利です。

マイナー アップグレード/スモール アップデート アイテムの設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

マイナーおよびスモールアップグレードは基本的には同じですが、マイナーアップグレードでは製品バージョンが変更されません。

最新バージョンのセットアップと同じ製品コードを持つセットアップをアップグレードするとき、マイナー/スモールアップグレードを適用する必要があります。機能からコンポーネントを削除、またはセットアップから機能を削除しない限り、セットアップの製品コードを変更する必要はありません。

メモ: *コンポーネントをある機能から別の機能へ移す場合、はじめの機能からそのコンポーネントを削除することになるので、製品コードを変更しなければなりません。*

セットアップの製品コードが変わったアップグレード シナリオでは、メジャーアップグレードを適用します。

このタブには以下のような設定があります。

テーブル 204 ・ マイナーとスモール アップグレード アイテムのオプション

オプション	説明
アップグレードするセットアップ	マイナー アップグレードをインストールするために必要な機能は setup.exe にあります。マイナー アップグレードを適切に動作させるためにはこれを含まなくてはなりません。ビルド時に指定したセットアップはアップグレードの検証にも利用されます。ビルドはリファレンスされたセットアップがマイナーアップグレードを使ってアップデートすることが可能かどうかを検証します。setup.exe を含んでいない場合、リリースのビルド時にビルドが警告を発します。
リリース フラグ	リリース フラグを指定して、リリースの種類によって機能を含めることも除外することもできます。

自動アップグレード アイテムの設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

メモ: *アップグレード設定の構成には、この方法が有効です。このオプションは将来的な製品コード変更の可能性を考慮し、メジャー アップグレードとマイナー アップグレードの両方を処理します。アップグレードの種類の違*

いが良く分からない場合および、または特にこだわらない場合はこのアップグレードオプションを選択してください。

自動アップグレードアイテムを追加したとき、以前のインストールを無事にアップグレードするためにはどの設定が必要かをビルドエンジンが判断します。このオプションではその他の詳細設定を行う必要はありません。

アップグレード ビューで、自動アップグレードアイテムのための以下のオプションを構成することができます。

テーブル 205 ・ 自動アップグレードアイテムのオプション

オプション	説明
アップグレードするセットアップ	アップグレードが必要なセットアップ プロジェクトへのパスを入力するか、[参照] ボタンをクリックしてそれを参照します。
リリース フラグ	リリースフラグを自動アップグレード項目に関連付けると、リリースごとに、オプションで、ビルドから除外することができます。

自動アップグレード動作における Settings.xml 構成サポート

InstallShield には、Settings.xml ファイルにLanguagesExclusiveAttributeInAutoUpgradeという名前のプロパティが含まれています。これにより、複数の言語をサポートするプロジェクトの自動アップグレード アイテムの動作が強化され、より予測可能で効率的なアップグレードの実行が保証されます。このプロパティは、特定の自動アップグレード アイテムの **Language**列 (**[ダイレクト ディター]** ビューの**Upgrade**テーブル) にリストされている言語をアップグレード条件で除外するかどうかを制御します。

LanguagesExclusiveAttributeInAutoUpgrade プロパティを **[はい]** に設定すると、指定した自動アップグレード アイテムの **[言語]** 列にリストされているすべての言語が除外または削除されます。

[リリース] ビュー

***プロジェクト・** [リリース] ビューは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*
- *MSIX*

InstallShield でプロジェクトの設計が完了したら、テスト用にリリースをビルドすることができます。このビルドは、最終的に、エンドユーザーに配布できるものです。[リリース] ビューには、リリースのビルド方法を指定できる設定が含まれています。

リリースをビルドすると、InstallShield でインストール プロジェクトのすべての情報が集められ、Windows Installer インストール パッケージ (.msi ファイル)、マージ モジュール (.msm ファイル)、実行可能ファイルと関連ファイルが、プロジェクトの種類に応じてコンパイルされます。このため、サポートされている Windows プラットフォームであれば、製品をインストールできます。

ヒント: *一部のプロジェクト タイプでは、リリース ウィザードを使用して、リリースの設定も構成することもできます。*

[製品構成] の設定

プロジェクト: *製品構成は、次のプロジェクトの種類で使用することができます:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、ビルドしたすべてのリリースは 1 つの製品構成に属します。製品構成により、製品名、製品コード、パッケージコードなど、類似したプロパティを共有するリリースを一緒にグループ化することができます。

各製品構成には 2 つのタブがあります:

- [全般] タブ (基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで利用可能)
- [複数インスタンス] タブ (基本の MSI プロジェクトで利用可能)

製品構成の [全般] タブ

プロジェクト: *製品構成の [全般] タブは、次のプロジェクトの種類で使用できます:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

各製品構成の [全般] タブには、以下の設定があります。これらの設定を利用して、作成したリリースで他の設定をオーバーライドすることができます。

テーブル 206 • [製品構成] の設定

設定	プロジェクトの種類	説明
製品名	基本の MSI、InstallScript MSI、マージ モジュール	この製品構成の下でビルドする各リリースの製品名をオーバーライドするには、新しい名前を入力します。 製品名がどのように使用されるかについては、「製品名の指定」を参照してください。 この設定で製品名を変更しても、ビルドされるリリースに影響はありません。製品名およびプロジェクトのフォルダーとファイルに変更が反映されているかどうかを確認するには、各リリースを再ビルドする必要があります。
製品バージョン	基本の MSI、InstallScript MSI、マージ モジュール	この製品構成の下にある各リリースで製品バージョンを上書きするには、バージョン番号を入力します。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。詳細については、「製品バージョンを指定する」を参照してください。 リリースに Setup.exe が含まれる場合、指定した製品バージョンが Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
パッケージ コード	基本の MSI、InstallScript MSI、マージ モジュール	[一般情報] ビューで入力済みのパッケージ コードをオーバーライドするには、新しい GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ({...}) をクリックします。 ***メモ:*** *"パッケージ コードの生成" 設定で [はい] が選択されている場合、このパッケージ コードは無視されます。*

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
パッケージ コードの生成	基本の MSI、InstallScript MSI、マージ モジュール	この製品構成の下にあるリリースがビルドされるたびに InstallShield が新しいパッケージ コードを生成するかどうかを指定します。 • **はい** – ビルド時に InstallShield が新しいパッケージ コードを生成して、それを .msi パッケージに含みます。[一般情報] ビューの "パッケージ コード" 設定に表示されるパッケージ コードは変更されません。 • **いいえ** – ビルド時に InstallShield は新しいパッケージ コードを生成しません。製品構成にパッケージ コードを入力した場合、そのパッケージ コードが使用されます。製品構成にパッケージ コードを指定しなかった場合、InstallShield は [一般情報] ビューで設定されたパッケージ コードを使用します。
製品構成フラグ	基本の MSI、InstallScript MSI、マージ モジュール	この製品構成に関連付けられている、リリースに含める機能、InstallShield 前提条件、および連鎖 .msi パッケージのリリース フラグを入力します。複数のフラグは、コンマで区切ります。 製品構成フラグを使って、この製品構成の下でビルドを行う各リリースごとに、特定の機能および InstallShield 前提条件、連鎖 .msi パッケージを含めるか除外するかを指定することで、インストールをカスタマイズすることができます。リリースレベルでリリース フラグを指定することもできます。 フィルター機能、InstallShield 前提条件、連鎖 .msi パッケージについての詳しい情報は、「リリース フラグ」をご覧ください。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
サブジェクト	基本の MSI、InstallScript MSI、マージ モジュール	この製品構成の下でビルドされる各リリースについて、[一般情報] ビューの "サブジェクト" 設定の値をオーバーライドするには、プロジェクトの名前を入力します。 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
タイトル	基本の MSI、InstallScript MSI、マージ モジュール	この製品構成の下でビルドされる各リリースについて、[一般情報] ビューの "タイトル" 設定の値をオーバーライドするには、新しいタイトルを入力します。 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
テンプレートの概要	基本の MSI、InstallScript MSI、マージ モジュール	特定の製品構成について、[一般情報] ビューの "テンプレートの概要" をオーバーライドするには、インストールがサポートするプロセッサの種類とデフォルト言語を指定します。最初にプロセッサの種類、次にインストールのデフォルト言語をセミコロンで区切って入力します。言語カテゴリーに複数のエントリがある場合は、コンマで区切ります。 たとえば、インストールが Intel プロセッサを搭載した英語版システム上でのみ動作する場合は、**Intel;1033** と入力します。製品が x64 プロセッサを搭載した英語およびドイツ語システム上で実行する場合は、**x64;1033,1031** と入力します。インストールが言語に左右されない場合、この設定の言語部分には、数値 0 を使用します。 有効なプロセッサ値は、次のとおりです: • Intel • Intel64 • x64 指定できるプロセッサ値は 1 つだけですので、ご注意ください。 詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。 ターゲット マシンが、この設定で指定された要件を満たさない場合、エラー メッセージが表示され、インストールが終了します。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
アーキテクチャの検証	基本の MSI、マージ モジュール	ビルド時に使用するアーキテクチャ検証の種類を指定します。選択可能なオプションは以下のとおりです: • **厳密な検証をしない**—この種類の検証を使って、x86 および x64 .msi パッケージ (Template Summary プロパティで指定) をビルドでき、両方の種類のパッケージで x86 と x64 製品ファイルとカスタム アクション ファイル を混合することができます。 アーキテクチャの厳密な検証をしない場合、"テンプレートの概要" プロパティで指定されたアーキテクチャとリリースに含まれる 1 つ以上の製品ファイルまたはカスタム アクション ファイルのアーキテクチャが一致しないとき、ビルド エラーまたは警告がトリガされません。 ビルド時に、ISRedistPlatformDependentFolder と ISRedistPlatformDependentExpressFolder パス変数は、x86 InstallShield カスタム アクション DLL を含む x86 の場所をポイントします。InstallShield は、プロジェクトにこれらのカスタム アクションを必要とするサポートが含まれている場合、これらの x86 カスタム アクションDLL をビルドに含みます。 [厳密な検証をしない] オプションがデフォルトの選択です。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
アーキテクチャの検証		• **厳密な検証をする**—この種類の検証を行うと、Template Summary プロパティで Intel (x86 用) または x64 のどちらが指定されているかによって、InstallShieldは x86 専用または x64 専用 .msi パッケージをビルドしようとします。 厳密な検証を行うオプションは、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上のカスタム アクション ファイルのアーキテクチャと一致しない場合に、ビルド エラーをトリガする場合があります。この種類の検証ではまた、"テンプレートの概要" プロパティで指定されているアーキテクチャが、リリースに含まれている 1 つ以上の製品ファイルのアーキテクチャと一致しない場合にビルド警告をトリガする場合もあります。 "テンプレートの概要" プロパティで Intel が指定されている場合、ISRedistPlatformDependentFolder および ISRedistPlatformDependentExpressFolder パス変数は x86 InstallShield カスタム アクション DLL を含む x86 の場所をポイントします。ただし、"テンプレートの概要" プロパティで x64 が指定されていて、厳密な検証を行う場合、これらのパス変数は x64 InstallShield カスタム アクション DLL を含む x64 の場所をポイントします。プロジェクトにこれらのカスタム アクションのいずれかを必要とするサポートが含まれている場合、InstallShield は適切な x86 または x64 バージョンの DLL を追加します。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
コメント	基本の MSI、InstallScript MSI、マージ モジュール	特定の製品構成について、[一般情報] ビューの "概要情報ストリーム コメント" 設定をオーバーライドするには、製品についての任意のコメントを入力します。この設定の通常の値は、次の通りです: **このインストーラー データベースには、MyProduct をインストールするために必要なロジックとデータが含まれています。** 入力した値は、[プロパティ] ダイアログ ボックスの [概要] タブに使用され、Windows Installer データベースを右クリックしてから [プロパティ] をクリックすると表示されます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。
スキーマ	基本の MSI、InstallScript MSI、マージ モジュール	スキーマ バージョンは、インストール パッケージに必要な Windows Installer の最小バージョンを識別する整数です。 特定の製品構成について、[一般情報] ビューの "スキーマ" 設定をオーバーライドするには、適切な整数値を入力します。 Windows Installer 2.0 を最小バージョンとする場合、**200** と入力します。Windows Installer 3.0 を最小バージョンとする場合、**300** と入力します。Windows Installer 3.1 を最小バージョンとする場合、**301** と入力します。Windows Installer 4.5 を最小バージョンとする場合、**405** と入力します。 エンド ユーザーのシステム上にある Windows Installer のバージョンが、"スキーマ" 設定で指定された最小要件よりも古い場合 (たとえば、インストールが Windows Installer 4.5 の機能を使用するため、スキーマの値が 405 に設定されているが、エンド ユーザーの Windows Installer バージョンが 3.1 の場合)、インストール時にエラーメッセージが表示され、インストールは途中で終了します。 "スキーマ" 設定で入力した値は、Windows Installer データベースの Page Count Summary プロパティに使用されます。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
製品コード	基本の MSI、InstallScript MSI	特定の製品構成について、[一般情報] ビューの "製品コード" 設定をオーバーライドするには、この製品を一意に識別する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 このコードは製品を一意的に識別するため、リリースを既に配布している場合は製品コードの変更はお薦めできません。 詳細については、「Windows Installer ベースのプロジェクトで製品コードを設定する」を参照してください。
アップグレード コード	基本の MSI、InstallScript MSI	特定の製品構成について、[一般情報] ビューの "アップグレード コード" 設定をオーバーライドするには、この製品のアップグレード コードに使用する GUID を入力します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 アップグレード コードは関連のある一連の製品を識別する GUID です。Windows Installer は、インストール済みの製品のメジャー アップグレードを行う際に、製品のアップグレード コードを使用します。UpgradeCode プロパティに格納されるアップグレード コードは、製品のすべてのバージョンにおいて同一でなくてはなりません。 詳細については、「アップグレード コードを設定する」を参照してください。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
プリプロセッサ定義	基本の MSI、InstallScript MSI	特定の製品構成の [設定] ダイアログ ボックスで [コンパイル/リンク] タブ にある "プリプロセッサ定義" 設定をオーバーライドするには、任意のプリプロセッサ定義を指定します。 等号およびコンマの前後に空白を置かないようにして、以下の形式を使用します。 **MYVARIABLE1=123,MYVARIABLE2=456** スクリプト内のこれらの種類の定数は、スクリプトの流れをコントロールする #if および #ifdef ステートメントを使ってテストすることができます。たとえば、この編集ボックスに MYVARIABLE と入力すると、次の #ifdef ループ中のコードが実行されます。 **#ifdef MYVARIABLE** **//コマンド** **#endif** このボックスでプリプロセッサ定義を追加または変更した後、この追加または変更を有効にするにはスクリプトをコンパイルする必要があります。 ***メモ:*** *多くの Windows API 関数はヘッダーファイル* ISRTWindows.h *で宣言され、*Ifx.h *スクリプトに含んだときに自動的に含まれます。[設定] ダイアログ ボックスの [コンパイル/リンク] タブにある [プリプロセッサ定義] ボックス内にプリプロセッサ定数 ISINCLUDE_NO_WINAPI_H を配置することで Windows API が自動的に定義されないようにすることが可能です。*

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
MSI パッケージ ファイル名	基本の MSI、InstallScript MSI	InstallShield がビルド時に生成する .msi および該当する場合はパッケージ定義ファイルに使用するファイル名を指定します (ピリオドまたは拡張子は除きます)。この設定が空白の場合、製品名が使用されます。 このフィールドに Property テーブルからのプロパティの値を含めることもできます。詳細は「セットアップ ファイル名」を参照してください。 ***重要 •*** *製品のアップデートを行うためのマイナー アップグレードまたはスモール アップデートをリリースできるようにしたい場合は、インストールの以前のバージョンと最新バージョンの .msi パッケージ名が同じでなくてはなりません。.msi ファイル名が異なる場合にマイナー アップグレードまたはスモール アップデートを実行しようとすると、Windows Installer ランタイム エラー 1316 の原因となる可能性があります。*
セットアップ ファイル名	基本の MSI、InstallScript MSI	InstallShield がビルド時に生成するセットアップランチャー ファイルに使用するファイル名を指定します (.exe ファイル拡張子は除きます)。この設定が空白の場合、InstallShield はデフォルト値 *Setup* を使用し、セットアップランチャーの名前は Setup.exe となります。 setup.exe の元のファイル名プロパティ (Setup.exe を右クリック > [プロパティ] > [詳細]) をカスタマイズすることができます。 このフィールドに、Property テーブルからのプロパティの値を含めることもできます。たとえば、次の任意のプロパティを入力します: setup[ProductVersion] setup[CustomVersion] setup[ProductCode] setup[ProductCode][ProductVersion] たとえば、**setup[ProductVersion]** と入力した場合、セットアップ名が setup14.10.1234.exe となります。

テーブル 206 • [製品構成] の設定 (続き.)

設定	プロジェクトの種類	説明
64 ビット セットアップ ランチャー	基本の MSI	64 ビット セットアップ ランチャーと 32 ビット セットアップ ランチャーのどちらをビルドするのかを指定します。 **[はい]** を選択して、64 ビット セットアップ ランチャーをビルドします。 **[いいえ]** を選択して、32 ビット セットアップ ランチャーをビルドします。詳細については、「64 ビット セットアップ ランチャーのビルド」を参照してください。
カスタム アクションのヘルプを含める	基本の MSI、InstallScript MSI、マージ モジュール	選択された製品構成に含まれるリリースをビルドする度に、自動的にカスタム アクション ヘルプ ファイルの各コンテンツを .msi ファイルにストリームするかどうかを指定します。カスタム アクションのヘルプ ファイルへのパスは、[カスタム アクションとシーケンス] ビュー (基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合) または [カスタム アクション] ビュー (マージ モジュール プロジェクトの場合) の ″ヘルプ ファイル パス″ 設定に表示されます。 製品がシステム管理者によって企業環境に配布される場合、システム管理者が各カスタム アクションの動作を把握する必要があるため、ここで [はい] を選択することが望まれます。[はい] を選択と、ビルド時に次のタスクが発生します。 • InstallShield は、ビルド中の .msi ファイルに ISCustomActionReference テーブルを追加します。[カスタム アクションとシーケンス] ビューで指定されたカスタム アクションのヘルプ ファイルのコンテンツすべてが、このテーブルの Description 列にストリームされます。 • プロジェクトにカスタム アクションを持つマージ モジュールが含まれる場合、マージ モジュール プロジェクトの [カスタム アクション] ビューで指定されたヘルプ ファイルのコンテンツも ISCustomActionReference テーブルに含まれます。 詳細については、「カスタム アクションの動作をドキュメント化する」を参照してください。

製品構成の [複数インスタンス] タブ

プロジェクト: *製品構成の [複数インスタンス] タブは、基本の MSI プロジェクトで使用できます。*

InstallScript プロジェクトにおける複数インスタンスのサポートに関する詳細は、「InstallScript インストールを複数回実行する」を参照してください。

Windows Installer は、マシン コンテキストでは 1 インスタンスの製品コードのみ、そして各ユーザー コンテキストでは 1 インスタンスのみインストールを許可します。Windows Installer 3.x 以降には、製品コードを変換するトランスフォームがサポートされています。この種類のトランスフォーム (インスタンス トランスフォーム) は、各インスタンスの製品コードを変更するため、同じ .msi パッケージを使って、同じ製品の複数インスタンスを同じコンテキストでインストールすることができます。

エンド ユーザーが製品の複数インスタンスをインストールすることができるインストールを作成するには、[複数インスタンス] タブを使用します。このタブを使って、製品の異なるインスタンスを定義し、各インスタンスに関連付けられたプロパティを構成します。ビルド時に、各インスタンスに対してインスタンス トランスフォームが作成され、インスタンス トランスフォームは .msi パッケージにストリームされます。実行時に、通常インストールはインスタンスの選択ダイアログを表示し、そこでユーザーは新しいインスタンスをインストールするか、既存のインスタンスを保持するかを指定することができます。

注意 *・エンド ユーザーが製品の複数インスタンスを同じマシン上に同じコンテキストでインストールできるインストールを作成するには、インストール開発者の洗練されたオーサリング能力と献身的な取り組みが必要です。この機能の使用は、上級インストール開発者にのみお勧めします。*

リリースの設定

プロジェクト: *リリースの設定は、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSIX*
- *MSIX 変更パッケージ*
- *スイート/アドバンスト UI*

[リリース] ビューのリリースについての設定は、異なる複数のタブに分類されています:

- リリースの [ビルド] タブ(アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI プロジェクトで使用できます)
- リリースの [Setup.exe] タブ(アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI プロジェクトで使用できます)

- リリースの [署名] タブ(アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI プロジェクトで使用できます)
- リリースの [インターネット] タブ (基本の MSI、InstallScript および InstallScript MSI プロジェクトで使用できます)
- リリースの [イベント] タブ(アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI プロジェクトで使用できます)
- リリースの [Appx/MSIX] タブ(アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI プロジェクトで使用できます)
- リリースの [アップデート] タブ(アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます)
- MSIX リリースの [全般] タブ (MSIX プロジェクトで使用できます)

リリースの [ビルド] タブ

プロジェクト: *[ビルド] タブは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[ビルド] タブでは、リリースのパッケージ方法を構成します。

テーブル 207 · [ビルド] タブの設定

設定	プロジェクトの種類	説明
リリースの場所	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リリースのビルドを開始するトップレベルのディレクトリのパスを入力するか、省略記号ボタン (...) をクリックして場所を参照します。
ビルドの場所	アドバンスト UI、スイート/アドバンスト UI	リリースのビルドを開始するトップレベルのディレクトリのパスを入力するか、省略記号ボタン (...) をクリックして場所を参照します。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
製品名	アドバンスト UI、スイート/アドバンスト UI	選択されたリリースについて、[一般情報] ビューで指定された製品名を新しい値でオーバーライドするには、適切な名前を入力してください。 製品名がどのように使用されるかについては、「製品名の指定」を参照してください。
製品バージョン	アドバンスト UI、スイート/アドバンスト UI	選択されたリリースについて、[一般情報] ビューで指定された製品バージョンを新しい値でオーバーライドするには、適切なバージョン番号を入力してください。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。詳細については、「製品バージョンを指定する」を参照してください。 指定した製品バージョンは、Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
スイート GUID	アドバンスト UI、スイート/アドバンスト UI	選択されたリリースについて、[一般情報] ビューで指定された GUID を新しい値でオーバーライドするには、適切な GUID を入力してください。スイート GUID は、アドバンスト UI またはスイート/アドバンスト UI インストールを一意に識別します。InstallShield を使って、自動的に異なる GUID を作成するには、この設定の **[新しい GUID の生成]** ボタン ([...]) をクリックします。 このコードはアドバンスト UI またはスイート/アドバンスト UI インストールを一意的に識別するため、リリースを既に配布している場合はスイート GUID の変更はお薦めできません。
メディアのフォーマット	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	この読み取り専用の設定では、このリリース用に選択したメディアの種類を表示できます。メディアの種類は、リリース ウィザードの [メディアの種類] パネルでのみ設定できます。InstallScript オブジェクト プロジェクトでは、メディアの種類は設定できません。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
圧縮	基本の MSI、InstallScript MSI、マージ モジュール	リリースを圧縮するかどうかを指定します。 • **圧縮** — InstallShield が製品のすべてのデータ ファイルを .cab ファイルに圧縮します。 • **非圧縮** — InstallShield が製品のファイルを .cab ファイルに圧縮しません。 また、1 つまたは複数の機能に関連付けられているファイルのみを .cab ファイルに圧縮することができるカスタム圧縮方法を利用することもできます。カスタム圧縮オプションを設定するには、リリース ウィザードの [リリースの構成] パネルと[カスタム圧縮の設定] パネルを使用します。 リリース ウィザードを使ってカスタム圧縮を構成すると、[リリース] ビューの "圧縮" 設定は **[カスタム (機能ごとに 1 つの Cab])** または **[カスタム (コンポーネントごとに 1 つの Cab)]** という読み取り専用の値に変更されます。圧縮設定をカスタムから標準圧縮または非圧縮に変更するには、再度リリース ウィザードを使用する必要があります。[リリース] ビューの "圧縮" 設定でこの変更を行うことはできません。 ***メモ:*** *ビルド プロセスの出力は、ビルドするメディアの種類 (CD-ROM、DVD-5、またはネットワーク イメージなど) や、圧縮方法 ("圧縮" 設定で指定)、セットアップランチャーを含めるかどうか (Setup.exe タブにある "セットアップランチャー" 設定で指定)、およびディスク分割を行うかどうか (CD-ROM、DVD、およびカスタム メディアの種類の "ディスクの分割" 設定で指定) によって異なります。* *たとえば、"圧縮" 設定で [圧縮] を選択し、"セットアップランチャー" 設定で [はい] を選択して、ネットワーク イメージ リリースをビルドした場合、データ ファイルが .cab ファイルに圧縮され、.cab ファイルが* Setup.exe *ファイルにストリームされます。"圧縮" 設定に [非圧縮] を選択し、"セットアップランチャー" 設定に [いいえ] を選択して、1 枚の CD に収まる CD-ROM リリースを自動ディスク分割設定を使ってビルドした場合、データ ファイルは圧縮されずに、.msi ファイルを含むフォルダーのサブフォルダーに配置されます。*

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
Cab 最適化の種類	基本の MSI、InstallScript MSI、マージ モジュール	"圧縮" 設定で "圧縮" またはカスタム オプションの 1 つを選択した場合、このリリースの .cab ファイルをビルドするときに使用する圧縮の種類も指定します。選択可能なオプションは以下のとおりです: • **LZX** – InstallShield は LZX 圧縮を使って、製品のデータ ファイルを .cab ファイルに圧縮します。このオプションを選択すると、最もサイズの小さい .cab ファイルが作成されますが、実行時に .cab ファイルからデータが抽出されるとき、最も時間がかかります。 • **MSZIP** – InstallShield は MSZIP 圧縮を使って、製品のデータ ファイルを .cab ファイルに圧縮します。デフォルトでは、これが設定されています。 • **なし** – .cab ファイルが作成されるとき、圧縮は行われません。 ***重要*** *• 圧縮を行うと、一般的に圧縮ファイルのサイズが減少します。ただし、ビルドプロセスの完了までの時間が長くなることがあります。圧縮されるファイルのサイズとその数によって、LZX 圧縮とビルドに数時間かかる場合もあります。したがって、LZX オプションを選択した場合、ビルドが完了するまでの所要時間をできるだけ短縮できるよう、ビルド マシンに最新のハードウェアを搭載することが推奨されます。*
スクリプトの圧縮	InstallScript、InstallScript オブジェクト	コンパイル済みのスクリプト ファイル (.inx ファイル) をキャビネット ファイルに配置する (はい) か、圧縮せずに Disk1 イメージ フォルダーに配置するか (いいえ) を指定します。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
コンパイラのプリプロセッサ定義	InstallScript、InstallScript オブジェクト、スイート/アドバンスト UI	オプションで、任意のプリプロセッサ変数定義を指定します。ここに指定されたプリプロセッサ変数定義は現在のリリースにのみ適用されます。他のリリースのスクリプトをコンパイルしているときには使用されません。等号およびコンマの前後に空白を置かないようにして、次の形式を使用します。 MYVARIABLE1=123,MYVARIABLE2 このような変数は、スクリプトのフローを制御する #if と #ifdef ステートメントにより、スクリプト内でテストできます。InstallScript オブジェクトを作成するには、**IFX_OBJECTS** と入力します。 この設定にプリプロセッサ変数の名前を入力すると、その変数が定義されます。たとえば、この設定に **MYVARIABLE** と入力すると、次の #ifdef ループ中のスクリプトコマンドが実行されます。 #ifdef MYVARIABLE //コマンド #endif この設定でプリプロセッサ変数定義を追加または変更した後、その追加または変更を有効にするためにインストールをコンパイルする必要があります。
ディスクの分割	基本の MSI、InstallScript MSI	この読み取り専用の設定は、インストール パッケージがどのようにディスク分割されるかを示します。リリース ウィザードの [ディスク分割オプション] パネルで、異なるオプションを選択できます。選択可能なオプションは以下のとおりです: • **自動** – 必要なディスク数に応じて、リリース ファイルを自動的に分割します。 • **カスタム – ディスク サイズを強制** – いずれかの機能のファイルがディスク サイズよりも大きい場合は、警告を発してビルドを中止します。 • **カスタム – ディスク サイズを強制しない** – 必要なディスク数に応じてリリースを分割します。 この設定は、[CD-ROM]、[DVD]、および [カスタム] メディア タイプに適用されます。 ***重要** • 複数ディスクのインストールは、ハード ドライブなどの非リムーバブル メディアからは実行できません。複数のディスクにまたがるインストールのテストを行う場合、インストールをターゲット メディアに置く必要があります。そうしなかった場合、Windows Installer の制限事項により、インストールは失敗します。*

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
ファイル名のフォーマット	基本の MSI、InstallScript MSI、マージ モジュール	ファイルのパスを .msi パッケージにどう格納するかを判別するために使用されるファイル名形式を選択します。InstallShield では、.msi パッケージの [Summary Information Stream] で指定したオプションが格納されます。 インストールを、長いファイル名をサポートしないメディア (Unix サーバーなど) で配布する場合、[短いファイル名] オプションを選択します。
プラットフォーム	InstallScript、InstallScript オブジェクト	このリリースが 1 つまたは複数のプラットフォーム固有の場合、この設定を使ってプラットフォームを指定します。これには、この設定の値をクリックしてから省略記号 (...) ボタンをクリックし、適切なプラットフォームを指定します。 コンポーネントに指定されたプラットフォームが、この設定で選択されたプラットフォームのどれにも一致しない場合、そのコンポーネントはリリースに含まれません。 この設定のデフォルト値は [プロジェクト設定を使用] です。この値は、リリースがプロジェクト レベルで指定されたプラットフォームをサポートすることを示します。
レイアウト	InstallScript	この読み取り専用設定は、プロジェクトのファイルをキャビネット ファイルに保存するか、またはディスク イメージに圧縮せずに配置するかを示します。この設定は、リリース ウィザードの [メディアのレイアウト] パネルで変更できます。選択可能なオプションは以下のとおりです: • **キャビネット ファイル** — すべてのデータ ファイルは、ビルド時に、1 つまたは複数のデータ キャビネット ファイルに配置されます。 • **CD-ROM フォルダー** — すべてのデータ ファイルは、ビルド時に 1 つまたは複数の CDROM フォルダーに配置されます。 • **カスタム** — ビルド時に、一部のデータ ファイルは、1 つまたは複数の CD-ROM フォルダーに配置され、他のデータは、データ キャビネット ファイルに配置されます。
データ言語	基本の MSI、InstallScript MSI、マージ モジュール	各コンポーネントで選択された言語に基づいて特定のコンポーネントを選択または除外する場合、この設定の省略記号ボタン (...) をクリックして、適切な言語を指定します。コンポーネントに指定された言語が、この設定で選択された言語のどれにも一致しない場合、そのコンポーネントはリリースに含まれません。 デフォルトでは、リリースは言語に依存しません。つまり、プロジェクトのコンポーネントはすべてリリースに含まれます。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
UI 言語	アドバンスト UI、基本の MSI、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	この設定を使って、リリースに含めるユーザー インターフェイス言語を指定することができます。適切な言語を選択するには、この設定で省略記号ボタン (...) をクリックします。 プロジェクトの [一般情報] ビューにある "セットアップ言語" 設定で言語が選択されていない場合、その言語は "UI 言語" 設定の利用可能な言語の一覧には表示されません。
言語	InstallScript、InstallScript オブジェクト	この設定を使用して、各コンポーネントで選択された言語に基づいて特定のコンポーネントを含めたり、その他のコンポーネントを除外したりできます。この設定を使って、リリースに含めるユーザー インターフェイス言語を指定することもできます。コンポーネントに指定された言語が、この設定で選択された言語のどれにも一致しない場合、そのコンポーネントはリリースに含まれません。また、プロジェクトに含まれる UI 言語がこの設定で選択された言語の 1 つと一致しない場合、InstallShield はその UI 文字列をリリースに含みません。 デフォルトで、リリースは言語に依存しません。つまり、プロジェクトのコンポーネントまたは UI 文字列はすべてリリースに含まれます。 プロジェクトの [一般情報] ビューで言語が選択されていない場合、その言語は " 言語" 設定の利用可能な言語の一覧には表示されません。
デフォルト言語:	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、スイート/アドバンスト UI	[一般情報] ビューまたは [文字列エディター] ビューで構成されたリリースのデフォルト プロジェクト言語をオーバーライドするには、インストールで使用する適切なデフォルト ユーザー インターフェイス言語を選択します。 詳細については、「デフォルトのプロジェクト言語の設定」を参照してください。
言語ダイアログ	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、スイート/アドバンスト UI	インストールが完全なユーザー インターフェイスで実行されたときに、インストールで言語の選択ダイアログを表示するかどうかを指定します。 ***ヒント:*** *基本の MSI プロジェクトの場合、言語の選択ダイアログを表示するには、*Setup.exe *セットアップランチャーが必要です。詳細については、「セットアップランチャーの作成」を参照してください。*

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
UTF-8 データベースのビルド	基本の MSI、InstallScript MSI、マージ モジュール	Windows Installer データベース、およびすべてのインスタンスまたは言語トランスフォームを UTF-8 エンコードを使ってビルドするかどうかを指定します。 UTF-8 エンコードは、すべての言語の文字を同時にサポートするため、エンド ユーザーに表示するテキストおよびファイル名とレジストリ キーの両方で、たとえば日本語とドイツ語、またはロシア語とポーランド語のように自由に言語を組み合わせて使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。 ***重要 •*** *Microsoft KB 979849 に記述されている通り、Windows Installer は UTF-8 データベースを完全にサポートしません。したがって、一部の状況においてユーザー インターフェイスに問題が生じる場合があります。たとえば、エンド ユーザーが /qb コマンドライン オプションを使って基本のユーザー インターフェイスでインストールを実行するか、または [プログラムの追加と削除] から製品をアンインストールすると、Windows Installer は UTF-8 データベースに含まれるユーザー インターフェイス テキストを表示するのに非常に小さいフォントを使用します。* [いいえ] を指定すると、リリースをビルドする際に ANSI データベースが作成されます。このオプションを使うと、異なるコード ページを持つ言語からの文字を同時に使用することができません。 この設定のデフォルト値は [いいえ] です。
以前のパッケージ	基本の MSI、InstallScript MSI、マージ モジュール	将来のパッチ パッケージのサイズを最小化するには、以前のパッケージ (インストール プロジェクトの場合、.msi ファイル、マージ モジュール プロジェクトの場合、.msm ファイル) への完全修飾パスを入力します。 この設定は、リリース ウィザードの "パッチの最適化" 設定に対応します。 ***メモ•*** *パッケージがダイナミック ファイル リンクを利用する場合、ファイル キーがリリース間で統一されるように、この設定で以前のパッケージを指定することが推奨されます。* 詳細については、「アップグレード関する考慮事項」を参照してください。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
ファイルのハッシュ値の生成	基本の MSI、InstallScript MSI、マージ モジュール	ビルド内のバージョンがない各ファイルに対して MsiFileHash テーブルを作成するかどうかを示します。選択可能なオプションは以下のとおりです: • **はい** – ビルド内のバージョンがない各ファイルについて、**MsiFileHash** テーブルを作成します。 • **いいえ** – **MsiFileHash** テーブルにデータを挿入しません。[いいえ] を選択すると、プロジェクトの (ダイレクト エディターを使って作成された) **MsiFileHash** テーブルにあるすべてのエントリがビルドされます。 特定のファイルについて **MsiFileHash**テーブルが既に作成されている場合、ビルドは、ビルド時に情報を生成する代わりに、その情報を使用します。 さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「MsiFileHash Table」を参照してください。
シャロー フォルダー構造	基本の MSI、InstallScript MSI	InstallShield で、"リリースの場所" 設定で指定された場所に直接 .msi ファイルとそれに関連するファイルを、サブフォルダー無しに作成されるようにする場合、[はい] を選択します。 リリースの場所が <ISProjectDataFolder> または <ISProjectFolder> の場合、リリースをビルドするには、[ビルド] メニューの [テーブルをビルドしてファイルをリフレッシュする] をクリックします。
Autorun.inf を作成する	基本の MSI、InstallScript MSI	インストールを CD-ROM または DVD-ROM で配布するときに、Autoplay 機能をサポートする場合、[はい] を選択します。インストールを自動再生するための指示が含まれた Autorun.inf というテキスト ファイルがディスク イメージ フォルダーのルートに作成されます。 このファイルを編集して AutoPlay オプションを追加したり、コマンド ライン パラメーターを MsiExec.exe、Setup.exe、または Update.exe に渡したりできます。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
リリース フラグ	アドバンスト UI、基本の MSI、InstallScript MSI、スイート/アドバンスト UI	リリース フラグを使用すると、各リリースで特定のアイテムを追加または削除することによって、インストールをスタマイズすることができます。このリリースに含めるフラグを入力します。複数のフラグは、コンマで区切ります。 ***プロジェクト:*** *基本の MSI および InstallScript MSI プロジェクトの場合、リリース フラグを機能、InstallShield 前提条件、および連鎖 .msi パッケージに割り当てた後、これらのフラグに基づいて機能、InstallShield 前提条件、および連鎖 .msi パッケージを含むリリースを作成できます。デフォルトでは、すべての機能、InstallShield 前提条件、および連鎖 .msi パッケージがリリースに含まれます。[リリース] ビューまたはリリース ウィザードのどちらかでフラグを指定すると、フラグが付いていないアイテムおよび指定されたリリース フラグを持つアイテムだけがインストールに含まれます。* *アドバンスト UI およびスイート/アドバンスト UI プロジェクトの場合、リリース フラグをアドバンスト UI またはスイート/アドバンスト UI プロジェクトの機能およびパッケージに割り当てると、割り当てたフラグに従って機能およびパッケージを含めたリリースを作成することができます。デフォルトでは、すべての機能およびパッケージがリリースに含まれます。[リリース] ビューでフラグを指定すると、フラグが付いていないアイテムおよび指定されたリリース フラグを持つアイテムだけがインストールに含まれます。* ***メモ:*** *リリースがリリース フラグを持たない場合、リリース フラグを持つ該当するすべてのアイテムが含められます。フラグが付いていないアイテムのみを含める場合、存在しないフラグを指定します。たとえば、**NoFlags** の様に指定します。こうすると、フラグが付いていないアイテムだけがリリースに組み込まれます。* *基本の MSI および InstallScript MSI プロジェクトの場合、製品構成レベルでもリリース フラグを指定することができます。詳細については、「製品構成フラグとリリース フラグの違い」を参照してください。*
パス変数のテスト値を使用する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	パス変数にテスト値を使用した場合、ここで [はい] を選択すると、この時点で実際の値が設定されます。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
パス変数のオーバーライド	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	選択したリリースをビルドする際にプロジェクトのパス変数をオーバーライドするには、この設定で省略記号ボタン (...) をクリックして、[パス変数のオーバーライド] ダイアログ ボックスを開きます。このダイアログ ボックスを使って、オーバーライドする 1 つ以上のパス変数を選択します。InstallShield は、選択された各パス変数に新しいパス変数設定を追加します。必要に応じて、各パス変数の設定を構成します。 [パス変数] ビューで構成されたユーザー定義のパス変数、環境変数、およびレジストリ変数をオーバーライドすることができますが、*<WindowsFolder>* などの定義済みパス変数をオーバーライドすることはできませんので、ご注意ください。 詳しくは、次を参照してください: • リリースのカスタム パス変数の値をオーバーライドする • パス変数を使用する
パス変数のオーバーライド アイテム	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	ビルド時に InstallShield が選択されたリリースに使用するパス変数の新しい値を指定します。デフォルト値は、[パス変数] ビューで構成された設定です。InstallShield はこの設定で指定された新しい値を使って、[パス変数] ビューで設定された値をオーバーライドします。 パス変数のオーバーライドを削除するには、この設定で [削除] ボタンをクリックします。 ***ヒント:*** *参照先となる他のパス変数の名前を山括弧で囲むことにより、定義済みの値の中でそのパス変数を参照することができます。たとえば、C:¥ という値を持つ MyRoot というパス変数がある場合、そのパス変数を Games という別の変数のパス変数定義で参照することができます。Games 変数の実際のパスは* C:¥Programs¥GameFiles *のようになりますが、Games を* <MyRoot>¥Programs¥GameFiles *と定義することができます。ただし、パス変数を自己参照しようとした場合は、使用した文字列そのものが代わりに使用されます。たとえば、Games を* <MyRoot>¥Programs¥<Games> *と定義すると、実際には 、Games は* C:¥Programs¥<Games> *のように定義されます。* 詳しくは、次を参照してください: • リリースのカスタム パス変数の値をオーバーライドする • パス変数を使用する

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
機能	InstallScript、InstallScript オブジェクト	この読み取り専用設定は、ビルドしたリリースに含める機能を示します。この設定の値は、リリース ウィザードの [機能] パネルで変更できます。 • **"ビルドに含める" 機能プロパティの使用** — このオプションが選択されている場合、"ビルドに含める" 設定が [はい] に設定されている各機能がビルド済みリリースに含まれます。また "ビルドに含める" 設定が [いいえ] に設定されている各機能はビルド済みリリースに含まれません。 • **直接含める機能を指定する** — 各機能をビルド済みリリースに含めるかどうかを指定します。
スキンの指定	InstallScript	このリリースに適用する ダイアログ スキン を選択します。デフォルト値 <プロジェクト設定の使用> は、[ダイアログ] ビューの [スキン] フォルダーで選択されたスキンを使用します。この設定に別の値を選択した場合、[スキン] フォルダーで行なった選択は上書きされます。
差分メディア	InstallScript	現在のリリース ([リリース] ビューで選択されたリリース) が差分リリース (つまり、1 つまたは複数の指定された既存リリースに存在しないファイルだけを含むリリース) か、または完全リリース (いずれのバージョンもインストールされていないシステム上に製品をインストールできるように製品のすべてのファイルを含むリリース) かを指定します。 詳細については、「差分リリースと完全リリース」を参照してください。
サポート バージョン	InstallScript	この設定は、リリースの "差分メディア" 設定で [いいえ] が選択された場合にのみ適用します。 リリースをアップデートして適用することができる、以前のバージョンの製品のバージョン番号の一覧をセミコロンで区切って (例、1.2.3;1.2.4) 指定します (オプション)。この設定を空白のままにしておくと、現在システムにインストールされている製品のバージョン、またはバージョンがあるかないかにかかわらず、インストールを実行することができます。

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
オブジェクトの差分	InstallScript	この設定は、"差分メディア" 設定で [はい] が選択されている場合のみ適用されます。差分リリースに InstallShield オブジェクトを含めるときの条件を選択します。 • **すべてを含める** – 差分リリースは、相当する完全アップデート リリースに含まれるすべてのオブジェクトを含みます。 • **すべてを除外する** – 差分リリースは、オブジェクトを含みません。 • **変更された場合に含める** – 分リリースは、相当する完全アップデート リリースに含まれるオブジェクトで、最低 1 つの比較リリースに見つからない、またはその中の対応オブジェクトと異なるオブジェクトを含みます。
未使用のディレクトリを保持する	基本の MSI、InstallScript MSI、マージ モジュール	このリリースをビルドしたとき、.msi ファイルの Directory テーブルから未使用のディレクトリを削除するかどうかを指定します。選択可能なオプションは以下のとおりです: • **いいえ** – Directory テーブルの Directory 列に一覧されているディレクトリが .msi ファイル内の既知の場所で参照されていない場合、InstallShield はビルド時に作成する .msi ファイルの Directory テーブルからそのディレクトリを削除します。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、これはマージ モジュールがマージされてから削除されますが、.msi ファイルに存在するディレクトリのみが削除の対象となります。したがって、マージ モジュールの Directory テーブルに新しい未使用のディレクトリが含まれている場合、そのディレクトリはインストールに追加されます。 • **はい** – InstallShield がビルド時に作成する .msi ファイルの Directory テーブルにあるディレクトリは、そのまま保持されます。 デフォルト値は [いいえ] です。 ***メモ:*** *特定の条件下では、定義済みディレクトリが解決されないためにインストールが失敗する場合があります。Directory テーブルから未使用ディレクトリを削除することで、無駄なエラーを回避することができます。したがって、この設定に [いいえ] を選択することが推奨されます。*

テーブル 207 • [ビルド] タブの設定 (続き.)

設定	プロジェクトの種類	説明
[追加/削除] パネルエントリを隠す	InstallScript MSI	コントロール パネルの [プログラムの追加と削除] でアプリケーションのエントリを非表示にするかどうかを指定します。 • **いいえ** – 製品のエントリが、ターゲット マシンの [プログラムの追加と削除] で表示されます。 • **はい** – 製品のエントリが、ターゲット マシンの [プログラムの追加と削除] で表示されません。エンドユーザーは、[プログラムの追加と削除] を使って、製品の削除や変更をしたり、サポート情報を表示したりすることができなくなります。 [プログラムの追加と削除] でボタンを非表示にする方法など、表示する製品情報の指定方法については、「[一般情報] ビューの設定」を参照して下さい。
アーキテクチャ	InstallScript InstallScript MSI	この設定により、32 ビット (x 86) および 64 ビット (x 64) Windows プラットフォームの両方、または 64 ビット (x 64) Windows プラットフォーム専用の InstallScript ベースのセットアップ ランチャーのどちらをビルドするのか指定します。次のオプションを選択できます: • **x86**—32 ビットと 64 ビットの両方の Windows プラットフォームで動作する 32 ビット セットアップ ランチャーをビルドします。 • **x64**—64 ビット Windows プラットフォーム上で動作する 64 ビット セットアップ ランチャーをビルドします。 デフォルトで、この設定は **x86** になっています。

リリースの [Setup.exe] タブ

プロジェクト: *[Setup.exe] タブは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *スイート/アドバンスト UI*

Windows Installer 再配布可能ファイルをインストールに含めるときの情報ついては、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」をご覧ください。

Setup.exe タブでは、Setup.exe ファイルの設定を構成します。また、Windows Installer 3.1 以前の再配布可能ファイルをインストールに含めるかどうかも指定することができます。Windows Installer 再配布可能ファイルについての詳細は、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」を参照してください。

テーブル 208 • Setup.exe タブの設定

設定	プロジェクトの種類	説明
セットアップランチャー	基本の MSI、InstallScript MSI	Setup.exe セットアップ ランチャーを作成するかどうかを指定します。 セットアップランチャーが必要になるシナリオについては、「セットアップランチャーの作成」をご覧ください。
実行可能ファイル名	アドバンスト UI、スイート/アドバンスト UI	InstallShield がビルド時に生成するセットアップランチャー ファイルに使用するファイル名を指定します(.exe ファイル拡張子は除きます)。この設定が空白の場合、InstallShield はデフォルト値 *Setup* を使用し、セットアップランチャーの名前は Setup.exe となります。
スイート読み込み画面の表示	アドバンスト UI、スイート/アドバンスト UI	このオプションを使って、インストール中にスイート読み込み画面を表示するかどうかを制御する事ができます。アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート 読み込み画面を非表示するには、このプロパティを **[いいえ]** に設定します。
スイート読み込み画面のメッセージ	アドバンスト UI、スイート/アドバンスト UI	このオプションを使って、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート読み込み画面にメッセージ追加することができます。
スイート読み込み画面のイメージ	アドバンスト UI、スイート/アドバンスト UI	このオプションを使って、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのスイート読み込み画面にイメージを選択します。 アプリケーションがスケーリングをサポートする場合、**[サポート ファイル]** の scale-150 および/または scale-200 フォルダー内にスケール 100 および 150 のカスタマイズされたイメージを挿入してください。 イメージの名前は、**"スイート読み込み画面のイメージ"** フィールドに入力されている **[サポート ファイル]** のフォルダーと同じでなくてはなりません。
単一 .exe ファイル名	InstallScript	インストールを実行する自己展開型の実行可能ファイルをビルドする場合、そのファイル名を指定します(拡張子も含む)。 ファイル名を指定すると、自己展開型の実行可能ファイルが作成され、このリリースの [ビルド] タブにある "リリースの場所" 設定で指定されたフォルダーの中にある Package サブフォルダーに追加されます。 この設定を空白のままにしておくと、ビルド時に、自己展開型の実行可能ファイルは作成されません。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
Setup.exe アイコン ファイル	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	Setup.exe ファイルに異なるアイコンを使用する場合、そのアイコンを含むファイルの完全修飾名を指定します。ファイルを指定するには、絶対パスまたはパス変数の相対パスを入力するか、省略記号ボタン (...) をクリックして [アイコンの変更] ダイアログ ボックスでファイルを参照します。 デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、[アイコンの変更] ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。 このフィールドを空白のままにした場合、Setup.exe ファイルにデフォルトのアイコンが使用されます。
64 ビット セットアップ ランチャー	アドバンスト UI、スイート/アドバンスト UI	64 ビット セットアップ ランチャーと 32 ビット セットアップ ランチャーのどちらをビルドするのかを指定します。 **[はい]** を選択して、64 ビット セットアップ ランチャーをビルドします。 **[いいえ]** を選択して、32 ビット セットアップ ランチャーをビルドします。詳細については、「64 ビット セットアップ EXE のビルド」を参照してください。
MSI コマンドライン引数	基本の MSI、InstallScript MSI	セットアップランチャーから Windows Installer に渡す任意のコマンドライン引数を示します。InstallShield は Setup.ini ファイルで指定された値を書き込みます。 [Startup] CmdLine=値 ***ヒント:*** *プロパティを設定する場合、この設定に* **MYPROPERTY="My Value"** *と入力します。*
セットアップ コマンドライン	InstallScript	エンド ユーザーがインストールを起動したときに Setup.exe に渡す任意のコマンドライン パラメーターを入力します。
パッケージ定義ファイルを作成する	基本の MSI、InstallScript MSI	エンド ユーザーが SMS ジョブとしてインストールを実行できるパッケージ定義ファイル (.pdf) を作成するかどうかを指定します。[はい] を選択すると、InstallShield がバージョン 2.0 の .pdf を作成します。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
必要実行レベル	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	"必要実行レベル" 設定を使って、Windows Vista 以降のプラットフォーム上でプロジェクトの Setup.exe ファイルを実行するのに必要な最低実行レベルを指定します。選択可能なオプションは、以下のとおりです。 • **管理者** – Setup.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。 非管理者は、管理者としての認証が必要になります。 • **最高権限** – Setup.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を承認する必要があります。 非管理者は、管理者権限を持たずに Setup.exe を実行します。これは、InstallScript プロジェクトと InstallScript MSI プロジェクトのデフォルト オプションです。 • **起動者** – Setup.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Setup.exe を実行することができます。Setup.exe は、資格情報または同意 (コンセント) を求める UAC メッセージを表示しません。これは、アドバンスト UI、基本の MSI、およびスイート/アドバンスト UI プロジェクトのデフォルト オプションです。 アドバンスト UI、InstallScript、InstallScript MSI、およびスイート/アドバンスト UI プロジェクト、および基本の MSI プロジェクトでは、"セットアップランチャー" 設定が [はい] に設定された場合、InstallShield はアプリケーション マニフェストを Setup.exe 起動ツールに埋め込みます。このマニフェストは選択された実行レベルを指定します。Windows Vista よりも古いバージョンのオペレーティング システムでは、必要実行レベルは適用されません。 基本の MSI プロジェクトで、"セットアップランチャー" 設定が [いいえ] に設定されている場合、InstallShield は Windows アプリケーション マニフェストを Setup.exe 起動ツールに埋め込みません。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
必要実行レベル (続き)		基本の MSI プロジェクトで必要実行レベルを昇格することの利点は、Setup.exe を実行するための権限の昇格が、必要な場合 1 度で済むという点です。またこれらの権限はインストールに含まれるすべての InstallShield 前提条件および .msi パッケージの [実行] シーケンスにも適用できるため、承認を得るためのプロンプトを複数回にわたって行う必要がありません。たとえば、すべての InstallShield 前提条件のうち 2 つが管理者権限を必要とする場合、この設定を [管理者] に変更します。そうすることにより、インストール中、Windows Installer が Setup.exe ファイルを実行する前、プロンプトはエンドユーザーに対して一度のみ表示されます。 アドバンスト UI またはスイート/アドバンスト UI プロジェクトにも似たような利点があり、Setup.exe を実行するための権限の昇格が、必要な場合 1 度で済みます。またこれらの権限はアドバンスト UI またはスイート/アドバンスト UI パッケージのすべてにも適用できるため、承認を得るためのプロンプトを複数回にわたって行う必要がありません。 ただし、権限を昇格して、インストールの終わりでもアプリケーションを起動する場合、この昇格された権限はアプリケーションに適用されますので注意してください。ほとんどの場合、Windows Vista 以降では、昇格された権限を使用したアプリケーションの実行は推奨されていません。 詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。
アンインストール順	スイート/アドバンスト UI	この設定を使って、スイート プロジェクトでパッケージのアンインストール順を指定します。選択可能なオプションは、以下のとおりです。 • **パッケージ順と同じ** – (プロジェクトで定義されている) パッケージのインストール順と同じ順番でパッケージをアンインストールします。 • **パッケージ順と逆** – (プロジェクトで定義されている) パッケージのインストール順と逆の順番でパッケージをアンインストールします。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
前提条件が昇格必要時のアドバタイズ	基本の MSI、InstallScript MSI	次のような一般的なシナリオにおいては、Windows Vista 以降のシステム上での製品のインストール中に、.msi ファイルをアドバタイズして、エンドユーザーが 2 つ目のユーザー アカウント管理 (UAC) を避けることができるようにした方がよい場合があります。 • インストールに、昇格が必要な InstallShield 前提条件が 1 つまたは複数含まれている。 • .msi ファイルも、同様に昇格を必要としている。(つまり、[一般情報] ビューの "管理者権限が必要" 設定がデフォルト値である [はい] に設定されているとき。) 作成中のインストールがこのシナリオに該当しない場合、"前提条件が昇格必要時のアドバタイズ" 設定をデフォルト値の [いいえ] のままにしておくことが推奨されます。これは、[はい] が設定された場合、2 つ目の UAC プロンプトが回避されなくなるからです。 "前提条件が昇格必要時のアドバタイズ" 設定は、Windows Vista 以降のシステム上で実行されるインストールに適用します。インストールが以前のバージョンの Windows 上で実行された場合、この設定は無視されます。有効なオプションは次のとおりです: • **アドバタイズ: サイレント** – インストールの InstallShield 前提条件が、昇格された権限で正常にインストールされたとき、.msi ファイルがアドバタイズされ、サイレント (ユーザー インターフェイスがない状態) で実行されます。これにより、資格情報または同意を求める追加の UAC プロンプトが、インストールの主要部分で必要なくなります。 • **アドバタイズ: 完全 UI**—インストールの InstallShield 前提条件が、昇格された権限で正常にインストールされた場合、.msi ファイルがアドバタイズされ、完全ユーザー インターフェイスを使って実行されます。これにより、インストールの主要部分は追加の UAC プロンプトを必要としなくなります。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
(続き)		• **アドバタイズなし** — .msi ファイルはアドバタイズされません。エンド ユーザーがインストールを実行したとき、1 つまたは複数の UAC プロンプトを表示して、InstallShield 前提条件をインストールできます。.msi ファイルも昇格を必要とする場合、インストールの主要部分が実行される前に、追加の UAC プロンプトを表示することができます。 ***メモ:*** *これらのアドバタイズ オプションが動作するためには、パッケージでアドバタイズがサポートされている必要があります。アドバタイズは瞬間的ではないので、インストールに多少の遅延が生じます。また、アドバタイズの後、インストールの主要部分が完了する前にエンドユーザーが [キャンセル] をクリックすると、予期しない動作が発生することがあります。たとえば、製品のアドバタイズ ショートカットが、メインのインストールが始まる前にデスクトップに表示されてしまうことがあります。また、この時、混乱したユーザーがメインのインストールをキャンセルすると、パッケージがアドバタイズされたまま、完全にインストールされないということも場合によって起きます。したがって、状況によっては、この設定を [アドバタイズなし] のままにすることで、2 つ目の UAC プロンプトを許可し、製品のアドバタイズを避けるた方が安全な場合もあります。* .msi ファイルがアドバタイズされず、その結果、2 つ目の UAC プロンプトが表示されることがあります。詳細については、「InstallShield 前提条件が昇格された権限で実行されるときに、製品をアドバタイズするかどうか指定する」を参照してください。
複数起動の防止	基本の MSI、InstallScript MSI	setup.exe ファイルを同時に複数起動または複数実行することを防ぐかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい** — このオプションを選択して setup.exe ファイルの複数起動または複数実行を防ぎます。 • **いいえ** — このオプションを選択して、setup.exe ファイルの複数起動または複数実行を許可します。 デフォルトで、この設定は **[いいえ]** になっています。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
MSI をローカルにキャッシュ	基本の MSI、InstallScript MSI	現在のビルドについて、.msi ファイルまたは他のインストール ファイルをターゲット システムにキャッシュするかどうか指定します。 • **[はい]** — .msi ファイルおよびその他のンストール ファイルがターゲット システム上にキャッシュされ、アプリケーションのメンテナンスや修復のときに使用されます。現在のリリースの "キャッシュ パス" 設定は、ファイルをキャッシュする場所を指定します。 • **[いいえ]** — .msi ファイルおよびその他のインストール ファイルはターゲット システム上にキャッシュされません。 ***メモ:*** *この設定は、ターゲットシステム上で .msi ファイルが* Setup.exe *ファイルと同じフォルダーに存在しないリリースに対して有効です。*
キャッシュ パス	基本の MSI、InstallScript MSI	キャッシュされた .msi ファイルおよび他のキャッシュされたインストール ファイルを保存するエンド ユーザーのシステム上の場所を指定します。**C:¥CachedFiles** などハードコード化された値を入力することもできますが、一覧から選択したインストール先変数を使ってファイルをパスにキャッシュすることをお勧めします。デフォルト値は **[LocalAppDataFolder]ダウンロードされたインストール** です。 ***メモ:*** *この設定は、現在のリリースの "MSI をローカルにキャッシュ" 設定が [はい] に設定されている場合にのみ使用されます。*
メディアのパスワード	InstallScript	インストールの実行にエンドユーザーが入力しなければならないパスワードを入力します。インストールをパスワードで保護しない場合、この設定を空白のままにします。
パスワード ダイアログを表示	InstallScript	この設定は "メディア パスワード" 設定でヌル値以外のパスワードを指定した場合のみ適用されます。 *SHOW_PASSWORD_DIALOG* のシステム変数をゼロ以外の値に設定するには (OnCheckMediaPassword イベント ハンドラー関数のデフォルト コードでパスワードを確認するコードが実行されます)、[はい] を選択します。 [いいえ] を選択した場合、スクリプトで独自のコードを入力して、エンドユーザーにパスワード入力を求め、**FeatureValidate** を呼び出してパスワードを確認する必要があります。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
小さい初期化ダイアログ	基本の MSI、InstallScript、InstallScript MSI	エンドユーザーがこのリリースを実行した時に小さい初期化ダイアログを表示するかどうかを指定します。 インストールの初期化中、デフォルトで、初期化ダイアログが表示されます。標準初期ダイアログはウェルカム ダイアログと同じサイズで、初期化テキスト、進行状況バー、および [キャンセル] ボタンが含まれます。小さな初期化ダイアログには、初期化テキスト、進捗状況、および [キャンセル] ボタンが含まれています。ただし、サイズは小さく、ビットマップ画像は含まれません。背景色は、エンドユーザーのシステムのウィンドウ色と同じになります。
Init ダイアログの製品名:	InstallScript	セットアップ初期化ダイアログに表示する製品名を入力します (オプション)。この設定を空白のままにすると、[一般情報] ビューで指定された製品名が使用されます。
最短初期化時間	基本の MSI、InstallScript、InstallScript MSI	エンドユーザーがこのリリースを実行した時に、インストールが初期化ダイアログを表示する最短時間 (秒) を指定します。 InstallShield は、この設定で指定した値を Setup.ini ファイルの SplashTime キー名の値として使用します。 インストールの初期化中、デフォルトで、初期化ダイアログが表示されます。スプラッシュ画面が使用されている場合、それもこのときに表示されます。この設定で最短初期化時間を指定すると、初期化ダイアログとスプラッシュ画面が少なくとも指定された秒数の間表示されます。初期化の時間が指定された時間を超えた場合、ダイアログとスプラッシュ画面は、インストールが初期化を完了するまで続けて表示されます。初期化が指定された時間以内に終了した場合、インストールは指定された時間が経過するまでダイアログとスプラッシュ画面を表示し、それからインストールを続行します。
ランチャをパスワードで保護	アドバンスト UI、基本の MSI、InstallScript MSI、スイート/アドバンスト UI	インストールをパスワードで保護するには、[はい] を選択してから "起動ツールのパスワード" 設定でパスワードを入力します。 ***プロジェクト:*** *基本の MSI および InstallScript MSI プロジェクトでは、この設定は "セットアップランチャー" 設定で [はい] が選択されている場合のみ使用できます。*

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
ランチャ パスワード	アドバンスト UI、基本の MSI、InstallScript MSI、スイート/アドバンスト UI	インストールを保護するパスワードを入力します。パスワードは大文字と小文字を区別します。 この設定は、"起動ツールをパスワードで保護" 設定で [はい] が選択されている場合のみ利用できます。 ***プロジェクト:*** *基本の MSI および InstallScript MSI プロジェクトでは、この設定は "セットアップランチャー" 設定で [はい] が選択されている場合のみ使用できます。*
カスタム バージョンのプロパティを使用する	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	Setup.exe のデフォルトの著作権情報とファイルの説明、およびファイル バージョンを独自の著作権情報、ファイルの説明、およびファイル バージョンでオーバーライドするかどうかを指定します。**[はい]** を選択した場合、**"起動ツールの著作権"** 設定**"ファイルの説明"**、および **"ファイル バージョン"** 設定に独自の情報を入力します。 製品バージョンおよびファイル バージョンが、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
ランチャの著作権	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	Setup.exe のデフォルトの著作権情報を製品の著作権情報でオーバーライドするには、製品の著作権情報を入力します。このとき、**"カスタム バージョンのプロパティを使用する"** 設定で **[はい]** も選択してください。 著作権情報が、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
ファイルの説明	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	Setup.exe のデフォルトのファイルの説明をオーバーライドするには、適切な説明を入力します。このとき、**"カスタム バージョンのプロパティを使用する"** 設定で **[はい]** も選択してください。 説明が、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
ファイル バージョン	基本の MSI、InstallScript、InstallScript MSI	Setup.exe のデフォルトファイル バージョンを独自のファイル バージョンでオーバーライドするには、適切なファイルバージョンを入力します。このとき、**"カスタム バージョンのプロパティを使用する"** 設定で **[はい]** も選択してください。 ファイル バージョンが、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
バージョン	アドバンスト UI、スイート/アドバンスト UI	Setup.exe のデフォルトの製品バージョンおよびファイル バージョンを独自のバージョン番号でオーバーライドするには、適切なバージョン番号を入力します。このとき、"カスタム バージョンのプロパティを使用する" 設定で [はい] も選択してください。 製品バージョンおよびファイル バージョンが、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 ファイル バージョンは、常に 4 つのフィールドで構成されます。製品バージョンに 4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、製品バージョンとして 1.1 を指定すると、Setup.exe のバージョン リソースで使用されるファイル バージョンは 1.1.0.0 となります。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
会社	アドバンスト UI、スイート/アドバンスト UI	Setup.exe のデフォルトの会社名をオーバーライドするには、適切な名前を入力します。このとき、"カスタム バージョンのプロパティを使用する" 設定で [はい] も選択してください。 会社名が、セットアップ起動プログラムの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
InstallShield 前提条件の場所	基本の MSI、InstallScript、InstallScript MSI	[再配布可能ファイル] ビュー (基本の MSI プロジェクトと InstallScript MSI プロジェクト) または [前提条件] ビュー (InstallScript プロジェクト) で選択された InstallShield 前提条件がある場所を指定します。 ***プロジェクト:*** *選択可能なオプションは、使用しているプロジェクトの種類によって異なります。* • **[個々の選択に従う]** – [再配布可能ファイル] ビューまたは [前提条件] ビューで各 InstallShield 前提条件のプロパティに指定した場所を使用します。 このオプションは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。 • **Web からダウンロードする**–プロジェクトに含まれるすべての 前提条件ファイルを、必要に応じて各前提条件の InstallShield 前提条件 (.prq) ファイルで指定された URL からダウンロードします。このオプションは、[再配布可能ファイル] ビューまたは [前提条件] ビューで各 InstallShield 前提条件のプロパティに指定した場所をオーバーライドします。 このオプションは、インストールがインターネットからダウンロードされるパッケージサイズとダウンロード時間を最小化したい場合に推奨されます。InstallShield 前提条件は、適切なバージョンが既にターゲット マシンにある場合はダウンロードされません。 このオプションは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。 • **Setup.exe から抽出する** – InstallShield 前提条件ファイルを Setup.exe に圧縮して、必要に応じて実行時に抽出します。このオプションは、[再配布可能ファイル] ビューで各 InstallShield 前提条件のプロパティに指定した場所をオーバーライドします。 このオプションは、インストール全体が Setup.exe に完全に含まれていなければならない場合に選択します。[Web からダウンロードする] オプションを選択すると、インストールが小さくなりダウンロード時間も短縮されますが、[Setup.exe から抽出する] オプションは完全に独立したインストールを提供します。 このオプションは、基本の MSI および InstallScript MSI プロジェクトで使用できます。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
InstallShield 前提条件の場所 (続き)		• **[ソース メディアからコピーする]**— InstallShield 前提条件ファイルをソース メディアに格納します。このオプションは、[再配布可能ファイル] ビューで各 InstallShield 前提条件のプロパティに指定した場所をオーバーライドします。 このオプションは、CD、DVD またはローカル ネットワークなどの固定メディアからインストールが非圧縮で実行される場合に使用します。 このオプションは、基本の MSI および InstallScript MSI プロジェクトで使用できます。 • **メディアに含める** — リリースの構成に従って、InstallShield 前提条件ファイルを、ソース メディアまたは Setup.exe に格納します。このオプションは、[前提条件] ビューで各 InstallShield 前提条件のプロパティに指定した場所をオーバーライドします。 このオプションは、InstallScript プロジェクトで使用できます。 InstallShield 前提条件が別の前提条件の依存ファイルとしてプロジェクトに追加される場合、前提条件依存ファイルの場所は、それを必要とする前提条件の場所設定に従います。 ***ヒント•*** *[Setup.exe から抽出する] オプション、[ソース メディアからコピーする] オプション、または [メディアに含める] オプションを選択してから、コンピューター上で利用できない InstallShield 前提条件を含むリリースをビルドした場合、前提条件が必要とする各ファイルについて 1 つまたは複数のビルド エラーが生成されます。これらのビルドエラーを回避するには、[再配布可能ファイル] ビューを使ってインターネットからコンピューターへ InstallShield 前提条件のダウンロードを行うか、リリースをビルドする前にプロジェクトから削除します。* 詳細については、「リリース レベルでの InstallShield 前提条件のランタイムの場所を指定する」を参照してください。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
パッケージの場所	アドバンスト UI、スイート/アドバンスト UI	[パッケージ] ビューで構成したパッケージのランタイムの場所を指定します。選択可能なオプションは以下のとおりです: • **[ソース メディアからコピーする]** – パッケージをソース メディアに格納します。このオプションは、[パッケージ] ビューで個別のパッケージの "場所" 設定に指定されたランタイムの場所をオーバーライドします。 アドバンスト UI またはスイート/アドバンスト UI インストールを CD、DVD またはローカル ネットワークなどの固定メディアから非圧縮で実行する場合は、このオプションを選択します。 • **Setup.exe から抽出する** – パッケージを Setup.exe に圧縮して、必要に応じて実行時に抽出します。このオプションは、[パッケージ] ビューで個別のパッケージの "場所" 設定に指定されたランタイムの場所をオーバーライドします。 インストール全体を Setup.exe に完全に含める場合、このオプションを選択します。**[Web からダウンロードする]** オプションを選択すると、インストールが小さくなり、ダウンロード時間も短くなります。ただし、**[Setup.exe からエンジンを抽出する]** オプションは完全に独立したインストールを提供します。 • **Web からダウンロードする** – 必要であれば、[パッケージ] ビューで各パッケージに指定された URL からプロジェクトに含まれているすべてのパッケージをダウンロードします。このオプションは、[パッケージ] ビューで個別のパッケージの "場所" 設定に指定されたランタイムの場所をオーバーライドします。 このオプションは、アドバンスト UI またはスイート/アドバンスト UI インストールがインターネットからダウンロードされる場合に、インストール サイズとダウンロード時間を最小化したい場合に推奨されます。パッケージは、適切なバージョンがターゲット システムに既存する場合はダウンロードされません。 • **個々の選択に従う** – [パッケージ] ビューで個別のパッケージの "場所" 設定に指定された場所を使用します。 詳細については、「リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する」を参照してください。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
パッケージをローカルにキャッシュする	アドバンスト UI、スイート/アドバンスト UI	ターゲット システム上で実行されるアドバンスト UI またはスイート/アドバンスト UI パッケージを、[パッケージ] ビューで各パッケージに対して定義した場所にキャッシュするかどうかを指定します。デフォルトの値は [はい] です。 非圧縮リリースをビルドする場合、この設定で [いいえ] を選択することが推奨されます。 ***メモ**• exe パッケージの場合、正しくアンインストールするには **"パッケージをローカルにキャッシュする"** 設定に [はい] を選択する必要があります。*
カスタム リターン コード	基本の MSI、InstallScript、InstallScript MSI	実行時に Setup.exe のリターン コードを独自のカスタム値で上書きするかどうかを指定します。デフォルト値は [いいえ] です。 成功時、失敗時、またはインストール中止の値を指定するには [はい] を選択します。 成功時、失敗時、またはインストール中止にサポートされている値は、0 から 32767 までです。 ***重要 •** フィールドが空白の場合、キャンセル/中止時に実際のエラーコードが返されます。*
有効期限日	基本の MSI、InstallScript MSI	エンドユーザーが特定の日付、またはその日付以降に Setup.exe を実行できなくするには、この設定で有効期限日を入力するか、矢印をクリックして、カレンダーから日付を選択します。 有効期限日を削除するには、この設定で [削除] ボタンをクリックします。 この設定で有効期限日を指定した場合、"有効期限日" 設定を使って、エンド ユーザーが有効期限日またはその日付以降に Setup.exe を起動しようとしたときに表示するメッセージを指定します。
有効期限のメッセージ	基本の MSI、InstallScript MSI	"有効期限日" 設定で構成した有効期限日またはその日付以降にエンドユーザーが Setup.exe を起動しようとしたときに表示するメッセージを入力します。

テーブル 208 • Setup.exe タブの設定 (続き.)

設定	プロジェクトの種類	説明
デバッグ ログを常に作成する	アドバンスト UI、スイート/アドバンスト UI	アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのデバッグ ログを作成する場合は **[はい]** を選択します。 オートメーション インターフェイスで CreateDebugLog メソッドを使って **"デバッグログを常に作成する"** 設定を指定することもできます。[CreateDebugLog] を [True] に設定して、デバッグ ログを作成します。デフォルト値は False です。
長いファイル名	アドバンスト UI、スイート/アドバンスト UI	デフォルトの**ログ ファイル名**は、InstallShield.log です。 **ログファイル名を**カスタマイズして、 • 任意の名前を指定することもできます。 • インストーラーの場所への絶対パスまたは相対パスを含みます。例: C:¥abc¥xyz.log または ..¥xyz.logフォルダーが指定された場所に存在しない場合は、それが作成されます。 • 任意の環境変数を含みます。例、%TEMP%¥xyz.log ***メモ•*** *インストーラーが指定されたディレクトリにアクセスできない場合、提供された名前のインストーラー場所に設定されます。*
セットアップ ファイル名	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	InstallShield がビルド時に生成するセットアップランチャー ファイルに使用するファイル名を指定します (.exe ファイル拡張子は除きます)。 この設定が空白の場合、InstallShield は "セットアップ" のデフォルト値を使用し、セットアップランチャー ファイルは Setup.exe と呼ばれます。
最小限を使用してインストール	アドバンスト UI、スイート/アドバンスト UI	スイート/アドバンスト UI インストーラーのインストールが標準モードか、パッシブ UIモードで実行するのかを指定します。パッシブ モードは、最小限のユーザー インターフェイスを使ってインストールする方法です。 • この設定で [**はい**] が選択されている場合、スイート/アドバンスト UI インストールがパッシブ モードで実行します。 • この設定で [**いいえ**] が選択されている場合、スイート/アドバンスト UI インストールが通常モードで実行します。デフォルトで、この設定は **[いいえ]** になっています。

リリースの [署名] タブ

プロジェクト• *[署名] タブは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

[署名] タブを使って、InstallShield がファイルに署名するときに使用するデジタル署名に関する情報 (証明機関より付与されたデジタル証明書ファイルを含む) を指定します。[署名] タブでまた、ビルド時にデジタル署名をするインストール内のファイルを指定することもできます。

テーブル 209 ・ [署名] タブの設定

設定	プロジェクトの種類	説明
Setup.exe ファイルの署名	アドバンスト UI、スイート/アドバンスト UI	アドバンスト UI またはスイート/アドバンスト UI インストールに署名するかどうかを指定します。
署名の種類	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	ビルドが生成したファイルにデジタル署名行う方法を指定します。署名ツール構成の引数を指定します: ・ **標準** ー InstallShield デフォルト署名ツールを使って、ビルドが生成したファイルに署名を行います。 ・ **カスタム** ー 独自のカスタマイズ署名ツールを使って、ビルドが生成したファイルに署名を行います。このオプションを設定すると、**"パス"** および **"引数"** 設定が有効になります。 ***メモ・*** *デフォルトで、この設定は* ***[標準]*** *になっています。*
パス	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	ビルドが生成したファイルにデジタル署名行うために使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
引数	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	署名ツール構成のコマンドライン引数を指定します。たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。 sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename] [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクト内で定義されているパス変数、環境変数、またはプロパティ名を使用することができます。 *メモ• "引数" 設定でプロパティ名を使用するには、角括弧 ([]) で囲む必要があります。*
証明書 URL	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンド ユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
デジタル証明書情報	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	リリースに署名を行うために使用するデジタル証明書を指定するには、この設定で省略記号ボタン (...) をクリックします。[証明書の選択] ダイアログ ボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。 InstallShield では、パブリック キー証明書 (.cer) ファイルを使って EV トークン パスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。.cer ファイルは通常、USB eToken プロバイダー (たとえば、SafeNet 認証クライアント) と関連付けられた EV 認証クライアント ツールからパブリック キーをエクスポートして作成されます。.cer ファイルが指定された場合、InstallShield は Extended Validation 証明書 (EV) プロパティを構成するための追加オプションを表示します。詳細については、次を参照してください。 • InstallShield における拡張検証 (EV) 証明書情報の構成 • [証明書の選択] ダイアログ ボックス. .pfx ファイルを指定するか、テスト ストアから証明書を選択すると、次が表示されます: • **証明書の拇印** – この読み取り専用設定は、証明書の拇印を表示します。 • **発行元** – この読み取り専用設定は、証明書の発行元情報を表示します。 • **有効期限日** – この読み取り専用設定は、証明書の有効期限日を表示します。 証明書ファイル (EV がエクスポートした .cer ファイル) を指定すると、次の設定が表示されます: • **暗号化サービス プロバイダー** – この設定を使って、暗号化サービス プロバイダー (CSP) を指定します。 • **コンテナ名** – この設定を使って、暗号化サービス プロバイダー (CSP) に関連付けられた秘密キーのコンテナ名を指定します。 • **トークン パスワード** – この設定を使って、暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。 ***メモ:*** *証明書ファイル (EV がエクスポートした .cer ファイル) を指定した後に表示される設定を使用する際は、以下の情報に注意してください:* • *暗号化サービス プロバイダー (CSP) 名とコンテナ名の両方は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
証明書パスワード	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	使用する .pfx にパスワードがある場合、それを入力します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。
出力ファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	署名を行うファイルを指定します。選択可能なオプションは以下のとおりです: • **メディア ヘッダー** – メディア ヘッダー ファイル (Data1.hdr) のみに署名する場合、このオプションを選択します。 このオプションは、InstallScript プロジェクトで使用できます。 • **なし** – インストールに署名を行わない場合、このオプションを選択します。 このオプションは、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクトで使用できます。 • **Setup.exe** – Setup.exe ファイルに署名する場合、このオプションを選択します。 このオプションは、基本の MSI、InstallScript、および InstallScript MSI プロジェクトで使用できます。 • **Setup.exe とメディア ヘッダー** – Setup.exe ファイルとメディア ヘッダー ファイル (Data1.hdr) に署名する場合は、このオプションを選択します。 このオプションは、InstallScript プロジェクトで使用できます。 • **[ソース メディアからコピーする]** – Setup.exe ファイルと Windows Installer パッケージ (.msi) に署名する場合、このオプションを選択します。 このオプションは、基本の MSI および InstallScript MSI プロジェクトで使用できます。 • **Windows Installer パッケージ** – Windows Installer パッケージ (.msi または .msm) に署名する場合は、このオプションを選択します。 このオプションは、基本の MSI プロジェクト、InstallScript MSI プロジェクト、およびマージ モジュール プロジェクトで使用できます。

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
64 ビット署名を使用	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	64 ビット署名フレームワークを使用してパッケージにデジタル署名を行うかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい** – 64 ビット署名フレームワークを有効にして、パッケージにデジタル署名を行います。 • **いいえ** – 32 ビット署署名フレームワークを有効にして、パッケージにデジタル署名を行います。 デフォルトで、この設定は **[いいえ]** になっています。
署名の説明	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	"署名出力ファイル" 設定で指定されたファイルに使用する署名の説明を指定します。ここで指定した説明は、UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示されます。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。 この設定を空白のままに残すと、InstallShield は UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示される説明として、ファイル名を拡張子なしで使用します。"パッケージ" 設定およびそのサブ設定で [ファイルに署名] を選択して、パッケージ内のファイルに署名を行った場合、InstallShield はビルド時に署名されるパッケージ内のファイルの UAC ダイアログ ボックスにこの説明を使用しません。
パッケージ内のファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	**[サポートファイル]** ビューで構成された InstallShield サポートファイル (英語、言語非依存、および詳細ファイル) を含む、インストール プロセス中にのみ使用されるすべてのファイルにデジタル署名するかどうかを指定します。 **[はい]** を選択する場合、**"含めるパターンとファイル"** 設定と **"除外するパターンとファイル"** 設定を使って、署名を行うファイルを示します。 ***Windows ロゴ*** *• インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
既に署名されているファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	プロジェクトにデジタル署名されているファイルが既に存在する場合、InstallShield で、既存のデジタル署名を [署名] タブで指定したデジタル署名で置き換えるかどうかを指定します。これは、"含めるパターンとファイル" 設定と "除外するパターンとファイル" 設定で指定された要件を満たすファイルにのみ適用されますので注意してください。 • 既にファイルに含められている既存のデジタル署名情報の代わりに、[署名] タブで指定するデジタル署名情報を使ってファイルに署名をする場合は、[はい] を選択します。 • 既に署名されているファイルについて、既存のデジタル署名情報をそのまま残す場合は、[いいえ] を選択ます。 デフォルト値は [いいえ] です。
元の場所にあるファイルに署名する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	InstallShield で署名をするとき、元のファイルにも署名をするか、またはリリースに組み込まれるファイルのみ署名するかを指定します。 • 各ファイルの一時コピーに署名して、その署名された一時コピーを使ってリリースをビルドする場合、[いいえ] を選択します。ここで [いいえ] を選択すると、元のファイルは変更も署名もされませんので注意してください。 • InstallShield で元のファイルに署名する場合、[はい] を選択します。 デフォルト値は [いいえ] です。
含めるパターンとファイル	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビルド時にデジタル署名するファイルおよびファイル パターンを指定するには、以下の手順の 1 つを行います。 • たとえば *.dll のようなファイル パターンだけでなく、現在プロジェクトに含まれているすべてのスタティック ファイルの一覧から、1 つ以上のファイル名またはファイル パターンを選択するには、この設定で省略記号ボタン (...) をクリックします。[ファイルを参照] ダイアログ ボックスが開き、1 つ以上のパターンおよびファイルを選択できます。アイテムの選択が完了すると、"次のパターンとファイルを含める" 設定の下に 1 つ以上の新しい "含める" 設定が追加されます。 • ファイル名またはパターンを手入力するには、この設定の [追加] ボタンをクリックします。InstallShield が "次のパターンとファイルを含める" 設定の下に新しい "含める" 設定を追加します。この新しい設定を使って、ファイル名またはパターンを指定できます。

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
含める	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビルド時にデジタル署名するファイルまたはファイル パターンを指定します。次の事項に注意してください。 • ワイルドカード文字の指定には、アスタリスク (*) を使用します。 たとえば、すべての .exe ファイルに署名する場合、*.exe のように指定します。**.exe** ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイル含め、特定のパターンに一致するすべてのファイルに署名を行う場合、特に便利です。 • 署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、"含める" 設定および "除外する" 設定に ***.exe** を指定すると、InstallShield は .exe ファイルに署名を行いません。 ファイルまたはファイル パターンを削除するには、この設定で [削除] ボタンをクリックします。 別のファイルまたはファイル パターンを追加するには、"含めるパターンとファイル" 設定を使います。
除外するパターンとファイル	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビルド時にデジタル署名しないファイルおよびファイル パターンを指定するには、以下の手順の 1 つを行います。 • たとえば *.dll のようなファイル パターンだけでなく、現在プロジェクトに含まれているすべてのスタティック ファイルの一覧から、1 つ以上のファイル名またはファイル パターンを選択するには、この設定で省略記号ボタン (...) をクリックします。[ファイルを参照] ダイアログ ボックスが開き、1 つ以上のパターンおよびファイルを選択できます。アイテムの選択が完了すると、"次のパターンとファイルを除外する" 設定の下に 1 つ以上の新しい "除外する" 設定が追加されます。 • ファイル名またはパターンを手入力するには、この設定の [追加] ボタンをクリックします。InstallShield が "除外するパターンとファイル" 設定の下に新しい "除外する" 設定を追加します。この新しい設定を使って、ファイル名またはパターンを指定できます。

テーブル 209 • [署名] タブの設定 (続き.)

設定	プロジェクトの種類	説明
除外する	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	ビルド時にデジタル署名しないファイルまたはファイル パターンを指定します。次の事項に注意してください。 • ワイルドカード文字の指定には、アスタリスク (*) を使用します。 たとえば、.drv ファイルはどれも署名しない場合 ***.drv** のように指定します。 ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイルが含まれていて、特定のパターンに一致するすべてのファイルに署名を行わない場合、特に便利です。 • 署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、"含める" 設定および "除外する" 設定に ***.exe** を指定すると、InstallShield は .exe ファイルに署名を行いません。 ファイルまたはファイル パターンを削除するには、この設定で [削除] ボタンをクリックします。 別のファイルまたはファイル パターンを追加するには、"除外するパターンとファイル" 設定を使います。

リリースの [インターネット] タブ

プロジェクト• *[インターネット] タブは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

[インターネット] タブでは、Web 関連の情報を指定します。

テーブル 210 • [インターネット] タブの設定

設定	プロジェクトの種類	説明
Web タイプ	基本の MSI、InstallScript MSI	Web インストール パッケージの構成を選択します。選択可能なオプションは以下のとおりです: • **単一実行可能ファイル** – このリリースを単一の自己展開型 Setup.exe ファイルとしてビルドします。インストール パッケージは自己完結型のため、この Web タイプは、多くの Web または FTP サイトからダウンロードされるパッケージに理想的です。ただし、完全なインストール パッケージがダウンロードされるため、ダウンロード時間が以下で説明されている Web タイプからのインストールよりも長くなってしまうことに注意してください。 • **ダウンローダー** – リリースを Setup.exe と MSI パッケージの組み合わせとしてビルドします。エンド ユーザーが Setup.exe をダウンロードして起動すると、すべてのファイルを含んだ .msi データベースが順にダウンロードされ実行されます。 • **Web からインストール** – Setup.exe、.msi データベースおよび外部キャビネット ファイル (.cab) を組み合わせてこのリリースをビルドします。ユーザーが Setup.exe をダウンロードして実行すると、.msi データベースがダウンロードされ実行されます。このとき、ユーザーのセットアップの種類と選択された機能に基づいて、必要な .cab ファイルのみがダウンロードされインストールされます。これにより、ダウンロード時間を最小限に抑えることができます。 ダウンローダー タイプと Web からインストールするタイプの場合、アプリケーションのメンテナンスおよび修復には、インストールのソースとして "ファイルの URL" 設定で指定された URL が使われます。
ファイルの URL	基本の MSI、InstallScript MSI	データ キャビネット ファイルをダウンロードするディレクトリを指定します。この設定では、**http://www.yourcompany.com/download** 形式の URL を使用できます。 この設定は、現在のリリースの "Web タイプ" 設定が [Web からインストールする] または [ダウンローダ] に設定されている場合にのみ使用されます。
IFTW .cab サイズ (KB)	基本の MSI、InstallScript MSI	Web メディア タイプ用にビルドされたキャビネット ファイル (.cab) のサイズをキロバイトで指定します。コンポーネントごとに別々の .cab ファイルをビルドするには、0 (ゼロ) を指定します。 この設定は、現在のリリースの "Web タイプ" 設定の値が [Web からインストール] に設定されている場合にのみ使用されます。

テーブル 210 • [インターネット] タブの設定 (続き.)

設定	プロジェクトの種類	説明
One-Click Install の生成	基本の MSI、InstallScript、InstallScript MSI	One-Click Install を作成するかどうかを指定します。One-Click Install は、最初のユーザー インターフェイスが HTML ページのインストール プログラムです。エンド ユーザーが Web ページにアクセスして [インストール] ボタンをクリックすると、インストール ファイルがターゲット システムにダウンロードされ、起動します。 ***プロジェクト:*** *基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合: エンド ユーザーのシステムにどのファイルがダウンロードされるかは、現在のリリースの "Web タイプ" 設定で選択されたオプションによって異なります。One-Click Install 設定で [はい] を選択した場合、このリリースに対する "One-Click HTML ベース名" 設定および "One-Click .cab ベース名" 設定のファイル名を必ず指定してください。この設定は、リリースの "メディアのフォーマット" 設定で Web を選択した場合にのみ使用されます。* *InstallScript プロジェクトの場合: この設定で [はい] を選択すると、適切なコマンドラインを使って* Setup.exe *ファイルのダウンロードと起動を行う外部セットアップ プレーヤー* (Setup.ocx *ファイル*) *がインストールに含められます。詳細については、「InstallScript プロジェクトの One-Click Install インストール」を参照してください。*
One-Click HTML ベース名	基本の MSI、InstallScript MSI	InstallShield がビルド時に生成する HTML ファイルのベース ファイル名を指定します。指定したベース名の末尾には .htm ファイル名拡張子が付けられ、生成されたファイルは現在のリリースの場所の Disk1 フォルダーに作成されます。 この設定は、"One-Click Install の生成" 設定の値が [はい] の場合にのみ有効です。
One-Click .cab ベース名	基本の MSI、InstallScript MSI	ビルド プロセスによって One-Click Install インストールに生成されるキャビネット ファイル (.cab) のベース ファイル名を指定します。 ここで指定された名前に .cab ファイル名拡張子が追加されます。生成されたファイルは、現在のリリースの Disk1 フォルダーに作成されます。 この設定で指定した名前は、インストールにデジタル署名を行なった場合、実行時にデジタル証明書に表示されます。 この設定は、"One-Click Install の生成" 設定の値が [はい] の場合にのみ有効です。
デフォルト Web ページの作成	InstallScript	InstallShield で、デフォルトの Web ページ (.htm ファイル) を作成し、Disk1 フォルダーに配置するかどうかを指定します。

テーブル 210 • [インターネット] タブの設定 (続き.)

設定	プロジェクトの種類	説明
Web ページ URL	InstallScript	次のいずれかを実行して、[ビルド] メニューで [Web から実行] コマンドをクリックしたときに起動される URL を示します。 • ビルド時に InstallShield によって作成され、Disk1 フィールドに配置される Setup.htm ファイルを起動する場合、このボックスを空白のままにします。 • 別の Web ページを起動する場合、このボックスに URL を入力するか、または参照ボタンをクリックして、Web ファイルを選択します。 ***ヒント•*** *URL を入力する場合、ページ名を含めないでください。また、末尾にスラッシュ (/) を使用することも避けてください。たとえば、Setup.htm ファイルの完全 URL が http://www.mypages.com/setup/Setup.htm の場合、Web ページ URL を http://www.mypages.com/setup のように入力します。* *[ビルド] メニューで [Web から実行] コマンドをクリックしたとき、適切な URL (上記例の http://www.mypages.com/setup/Setup.htm) が起動されます。*

リリースの [イベント] タブ

プロジェクト• *[イベント タブは、次のプロジェクト タイプで使用できます:*

- *アドバンスト UI*
- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *スイート/アドバンスト UI*

その他にプロジェクトによって異なる情報がある場合は、その内容が記載されています。

[イベント] タブを使って、InstallShield がビルド前、ビルド中、またはビルド後に行う様々なタスクの設定を構成することができます。

[イベント] タブの設定は、次のメイン カテゴリに分かれています:

- コマンド
- パブリッシュ
- 配布

- 仮想マシン (この領域は InstallShield Premier Edition で使用できます。)

コマンドの設定

[イベント] タブの [コマンド] 領域を使って、ビルド プロセスの様々な段階で実行するコマンドを指定できます。

テーブル 211 • [イベント] タブにあるコマンドの設定

設定	プロジェクトの種類	説明
ビルド前のイベント	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	 ***エディション:*** *この設定は、InstallShield Premier Edition で使用できます。* InstallShield がリリースをビルドを開始する前に実行するコマンドを指定します。このイベントは InstallShield がリリース フォルダとログ ファイルを作成した後、リリースのビルドを開始する前に実行します。 複数のコマンドを指定するには、この設定で省略記号ボタン (...) をクリックして [ビルド前のイベント] ダイアログ ボックスを開きます。このダイアログ ボックスで 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

テーブル 211 • [イベント] タブにあるコマンドの設定 (続き.)

設定	プロジェクトの種類	説明
ファイルの準備イベント	基本の MSI、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	 ***エディション•*** *この設定は、InstallShield Premier Edition で使用できます。* InstallShield ビルド中に実行するコマンドを指定します。このイベントは、InstallShield がリリース フォルダーを作成した後、InstallShield ビルドによって生成された DLL が .msi パッケージまたは Setup.exe ファイルにストリームされる前に実行されます。 複数のコマンドを指定するには、この設定で省略記号ボタン (...) をクリックして **[ファイルの準備イベント]** ダイアログ ボックスを開きます。このダイアログ ボックスで 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳しくは、次を参照してください: • ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する • ファイルの準備イベントを使って、署名されていない InstallShield ファイルに署名を行う

テーブル 211 • [イベント] タブにあるコマンドの設定 (続き.)

設定	プロジェクトの種類	説明
圧縮前のイベント	基本の MSI、InstallScript MSI	 ***エディション*** *この設定は、InstallShield Premier Edition で使用できます。* 製品のデータ ファイルを .cab ファイルに格納する場合、InstallShield が .msi パッケージと .cab ファイルをビルドした後に実行するコマンドを指定します。このイベントは .cab ファイルが .msi パッケージにストリームされた後、.msi パッケージにデジタル署名が行われて Setup.exe ファイルにストリームされる前に発生します。 複数のコマンドを指定するには、この設定で省略記号ボタン (...) をクリックして [圧縮前のイベント] ダイアログ ボックスを開きます。このダイアログ ボックスで 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。
ファイルの準備でバッチファイルを実行する	InstallScript、InstallScript オブジェクト	 ***エディション*** *この設定は、InstallShield Premier Edition で使用できます。* InstallShield のビルド中、バッチ ファイル (.bat または .cmd) または実行可能ファイル (.exe) を起動するには、ファイルへのパスを入力するか、省略記号ボタン (...) をクリックしてファイルを参照します。パスの一部として、山かっこ (<>) で括ったパス変数を使用できます。 これにファイルを指定すると、プロジェクトのビルド変数と同じ名前 (山かっこを含む) と値が環境変数に設定されます。また InstallShield は、環境変数 *<ISMEDIADIR>* も設定します。この変数の値は、その中にリリースの Disk Images フォルダー、Log Files フォルダー、および Report Files フォルダーが作成されるフォルダーへのパスです。バッチ ファイルの環境変数の値は、変数名をパーセント記号 (%) で囲んで参照することができます。例、 set PATH = %*<ISPROJECTDIR>*%%PATH% ファイルが終了すると、設定された環境変数は削除されます。

テーブル 211 • [イベント] タブにあるコマンドの設定 (続き.)

設定	プロジェクトの種類	説明
バッチ ファイルの実行	InstallScript、InstallScript オブジェクト	リリースがビルドされたあと、バッチ ファイル (.bat) または実行可能ファイル (.exe) を起動するには、ファイルへのパスを入力するか、省略記号ボタン(...)をクリックしてファイルを参照します。パスの一部として、山かっこ (<>) で括ったパス変数を使用できます。 これにファイルを指定すると、プロジェクトのビルド変数と同じ名前 (山かっこを含む) と値が環境変数に設定されます。また InstallShield は、環境変数 *<ISMEDIADIR>*, も設定します。この変数の値は、その中にリリースの Disk Images フォルダー、Log Files フォルダー、および Report Files フォルダーが作成されるフォルダーへのパスです。バッチ ファイルの環境変数の値は、変数名をパーセント記号 (%) で囲んで参照することができます。例、 set PATH = %*<ISPROJECTDIR>*%;%PATH% ファイルが終了すると、設定された環境変数は削除されます。
ビルド後のイベント	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	 ***エディション:*** *この設定は、InstallShield Premier Edition で使用できます。* InstallShield がリリースをビルドおよび署名した後に実行するコマンドを指定します。 複数のコマンドを指定するには、この設定で省略記号ボタン (...) をクリックして [ビルド後のイベント] ダイアログ ボックスを開きます。このダイアログ ボックスで 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

パブリッシュの設定

プロジェクトの種類によって異なる "パブリッシュ" 設定が表示されます。また、InstallShield の単体バージョンか、AdminStudio に含まれている バージョンかによっても異なります。

InstallShield の単体バージョン (AdminStudio なし) における "パブリッシュ" 設定

InstallShield の単体バージョン (AdminStudio なし) の場合、マージ モジュール プロジェクトで [イベント] タブの [パブリッシュ] 領域にこの設定があります。これらの設定を使って、リリースのビルドが成功した後、マージ モジュールを Merge Modules フォルダーを使って提供するか、またはリポジトリを使って提供するかを指定します。

テーブル 212 • InstallShield (AdminStudio なし) バージョンの [イベント] タブにある "パブリッシュ" 設定

設定	プロジェクトの種類	説明
マージモジュールのパブリッシュ	マージ モジュール	選択したリリースのビルドが成功した後、マージ モジュールを Merge Modules フォルダーを使って提供するか、またはリポジトリを使って提供するかを指定します。選択可能なオプションは以下のとおりです: • **リポジトリにパブリッシュ** – リリースのビルドが成功したとき、マージ モジュールが "リポジトリの名前" 設定で指定されたリポジトリにパブリッシュされます。このオプションは、1 つ以上のリポジトリが構成されている場合のみ選択可能です。(リポジトリを構成するには、[ツール] メニューで、[オプション] をクリックし、[リポジトリ] タブをクリックします。リポジトリは、このタブで構成できます。) • **Merge Modules フォルダーにコピー** – リリースが正常にビルドされたとき、常に Merge Modules フォルダーを新しいマージ モジュールで更新します。 • **パブリッシュしない** – リリースがビルドされたとき、マージ モジュールはリポジトリにパブリッシュされません。また Merge Modules フォルダーにもコピーされません。 詳細については、「リポジトリを使用してプロジェクトの要素を共有する」を参照してください。
リポジトリ名	マージ モジュール	リリースが正常にビルドされたとき、ビルドされたマージ モジュールをパブリッシュするリポジトリを選択します。 このオプションは、1 つ以上のリポジトリが構成されている場合のみ選択可能です。(リポジトリを構成するには、[ツール] メニューで、[オプション] をクリックし、[リポジトリ] タブをクリックします。リポジトリは、このタブで構成できます。)
表示名	マージ モジュール	リポジトリにパブリッシュするマージ モジュールの表示名を指定します。 マージ モジュールがリポジトリにパブリッシュされたあとで、[再配布可能ファイル] ビューでこのマージ モジュールを選択すると、その表示名が詳細ペインに表示されます。

テーブル 212 • InstallShield (AdminStudio なし) バージョンの [イベント] タブにある "パブリッシュ" 設定 (続き.)

設定	プロジェクトの種類	説明
発行者	マージ モジュール	このマージ モジュールをリポジトリにパブリッシュする担当者の名前を入力してください。[再配布可能ファイル] ビューで、このマージ モジュールを選択すると、この名前が説明ペインに表示されます。 マージ モジュールがリポジトリにパブリッシュされたあとで、[再配布可能ファイル] ビューでこのマージ モジュールを選択すると、その発行者名が詳細ペインに表示されます。
説明	マージ モジュール	このマージ モジュールについて、表示する説明を入力します。 マージ モジュールがリポジトリにパブリッシュされたあとで、[再配布可能ファイル] ビューでこのマージ モジュールを選択すると、その説明が詳細ペインに表示されます。

AdminStudio に含まれている InstallShield バージョンにおける "パブリッシュ" 設定

AdminStudio の一部として出荷されるバージョンの InstallShield の場合、基本の MSI プロジェクトで [イベント] タブの [パブリッシュ] 領域にこの設定があります。これらの設定を使って、ビルド中の Windows Installer パッケージをソフトウェア リポジトリにパブリッシュするかどうかなどの情報を指定します。

テーブル 213 • AdminStudio に含まれている InstallShield バージョンの [イベント] タブにある "パブリッシュ" 設定

設定	プロジェクトの種類	説明
AdminStudio リポジトリのパブリッシュ	基本の MSI	ビルド中の Windows Installer パッケージをソフトウェア リポジトリにパブリッシュするかどうかを指定します。 ソフトウェア リポジトリを使って Windows Installer パッケージを管理する場合、.msi ファイルおよびすべての関連ファイルとサブフォルダーが、そのパッケージのアプリケーション カタログで指定されているソフトウェア リポジトリの場所にあるサブフォルダーに格納されます。
グループ	基本の MSI	"AdminStudio リポジトリにパブリッシュ" 設定で [はい] を選択して、Windows Installer パッケージをアプリケーション カタログ内の 1 つ以上のグループに関連付ける場合、この設定で省略記号ボタン (...) をクリックします。[アプリケーション マネージャー グループの選択] ダイアログ ボックスが開きます。Windows Installer パッケージを含める各グループのチェック ボックスを選択します。

テーブル 213 • AdminStudio に含まれている InstallShield バージョンの [イベント] タブにある "パブリッシュ" 設定 (続き.)

設定	プロジェクトの種類	説明
更新オプション	基本の MSI	選択されたリリースでビルドされる Windows Installer をアプリケーション カタログにインポートする方法を指定します。選択可能なオプションは以下のとおりです: • **新しいパッケージ バージョン** – アプリケーション カタログで、新規のビルドは新しいパッケージとして処理されます。 • **既存のバージョンを上書き** – アプリケーション カタログで、新しいビルドが既存バージョンを上書きします。 • **新しいパッケージ履歴バージョン** – 新しいビルドが、アプリケーション カタログに存在するパッケージの新しいバージョンとして処理されます。 • **存在する場合、無視する** – 新しいビルドは、アプリケーション カタログに既存しない場合のみインポートされます。

配布

[イベント] タブにある [配布] 領域を使って、ビルド時にリリースをフォルダーまたは FTP サイトに自動的に配布するための設定を構成します。

テーブル 214 • [イベント] タブにある配布の設定

設定	プロジェクトの種類	説明
フォルダーにコピー	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リリースを特定のフォルダーへ自動的に配布するようにする場合、その場所を指定します。コピーされたフォルダーと同じ名前の既存のフォルダーは上書きされますが、フォルダーは削除されません。この設定でパスを入力するか、省略記号ボタン (...) をクリックして場所を参照して、フォルダーのパスを指定します。 選択されたリリースのメディア形式がネットワーク イメージの場合 (ディスク イメージ フォルダーは 1 つのみ作成されます)、フォルダー自体ではなくディスク イメージ フォルダーの内容がコピーされます。自己展開型実行可能ファイルを作成することを選択した場合、ディスク イメージ フォルダーではなく実行可能ファイルがコピーされます。 ***プロジェクト•*** *InstallScript および InstallScript Object プロジェクトの場合、リリースがビルドされるたびに、InstallShield が、自動的にそのリリースを [イベント] タブで指定したフォルダーにコピーします。* *その他のプロジェクトの種類の場合、[リリース] ビューでリリースを右クリックして、[配布] をクリックすると、リリースに関連するファイルがすべて指定されたフォルダーにコピーされます。* • *基本の MSI* • *InstallScript MSI* • *マージ モジュール* *基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトで、ビルドするたびに、ビルド エンジンがリリースを指定のフォルダーにコピーするように設定する場合、"ビルド後、配布する" 設定で [はい] を選択します。* ***メモ•*** *この設定でフォルダーを指定し、[イベント] タブで FTP ロケーションも指定した場合、リリースは FTP ロケーションにのみコピーされます。*

テーブル 214 • [イベント] タブにある配布の設定 (続き.)

設定	プロジェクトの種類	説明
FTP ロケーション	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	リリースを特定の FTP サーバーに自動的に配布できるようにするには、その場所の FTP URL を指定します。プロジェクトの各リリースに、異なる FTP ロケーションを指定することができます。 ***メモ•*** *FTP デフォルト フォルダー以外のパスにリリースを配布する場合は、ダブル スラッシュ (//) を使います。たとえば、リリースを、FTP の URL が* **ftp://ftp.mydomain.com** *である* **myproduct** *という名前のルートレベル フォルダーへ配布する場合、FTP の場所に* **ftp://ftp.mydomain.com//myproduct** *と入力します。* ***プロジェクト•*** *InstallScript および InstallScript Object プロジェクトの場合、リリースがビルドされるたびに、InstallShield が、自動的にそのリリースを [イベント] タブで指定した FTP の場所にコピーします。* *その他のプロジェクトの種類の場合、[リリース] ビューでリリースを右クリックして、[配布] をクリックすると、リリースに関連するファイルがすべて指定された FTP ロケーションにコピーされます。* • *基本の MSI* • *InstallScript MSI* • *マージ モジュール* *基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトで、ビルドするたびに、ビルド エンジンがリリースを指定の場所にコピーするように設定する場合、"ビルド後、配布する" 設定で [はい] を選択します。*
FTP サイト ユーザー名	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	指定する FTP ロケーションへのアップロードにユーザー名が必要な場合、ユーザー名を入力します。 ***プロジェクト•*** *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで、リリースの 1 つにユーザー名を入力すると、そのユーザー名が同じプロジェクトおよび他の基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの他のリリースにも使用されます。*

テーブル 214 • [イベント] タブにある配布の設定 (続き.)

設定	プロジェクトの種類	説明
FTP サイト パスワード	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	指定する FTP ロケーションへのアップロードにパスワードが必要な場合、パスワードを入力します。 ***プロジェクト•*** *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで、リリースの 1 つにパスワードを入力すると、そのパスワードが同プロジェクトおよび他の基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの他のリリースにも使用されます。*
ビルドの後、配布する	基本の MSI、InstallScript MSI、マージ モジュール	ビルドする度に、ビルド エンジンが自動的に指定した場所へリリースを配布するようにするかどうかを指定します。 ***ヒント•*** *ビルドの実行ごとではなく、オンデマンドでリリースを素早く配布するには、リリースを右クリックして、[配布] をクリックします。[配布] コマンドは、リリースが少なくとも 1 度ビルドされている場合のみ利用することができます。*

仮想マシンの設定

エディション• *[仮想マシン] 領域は、InstallShield Premier Edition で使用できます。*

[イベント] タブの [仮想マシン] 領域を使って、ビルド時、またはオンデマンドで選択されたリリースを仮想マシン (VM) に配布するときの設定を構成します。詳細については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 215 • [イベント] タブにある仮想マシンの設定

設定	プロジェクトの種類	説明
構成	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	この設定には、選択されたリリースを VM に配布するときに使用される VM 構成設定グループの名前が表示されます。 既存の VM 構成を指定するには、この設定で選択します。 新しい VM 構成を作成するには、この設定で省略記号ボタン (...) をクリックします。 既存の VM 構成を編集するには、その構成を選択してから省略記号ボタン (...) をクリックします。 指定されたリリースで選択された VM 構成の詳細 (インストールを配布する VM のインストール先など) をオーバーライドするには、この設定の下にあるサブ設定を構成します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。 VM 構成を他のマシンと共有する方法については、「リリースの配布用の仮想マシン設定を共有する」を参照してください。
スナップショット	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	オプションで、リリースの配布に使用するスナップショットの名前を指定することもできます。 "構成" 設定で選択されている VM 構成にスナップショットが指定されている場合、そのスナップショットの名前がこの設定のデフォルト値として取り扱われ、灰色テキストで表示されます。選択されたリリースのスナップショットをオーバーライドするには、この設定に入力します。 VM 構成にスナップショットが指定されていない場合、InstallShield は特定のスナップショットには戻りません。InstallShield は特定のスナップショットに戻さずに VM の電源をオンにして、VM にインストールをコピーします。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 215 • [イベント] タブにある仮想マシンの設定 (続き.)

設定	プロジェクトの種類	説明
ユーザー名	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	"構成" 設定で選択されている VM 構成にユーザー名が指定されている場合、そのユーザー名前がこの設定のデフォルト値として取り扱われ、灰色テキストで表示されます。選択されたリリースのユーザー名をオーバーライドするには、この設定に入力します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
パスワード	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	"構成" 設定で選択されている VM 構成にパスワードが指定されている場合、そのパスワードがこの設定のデフォルト値として取り扱われ、灰色テキストで表示されます。選択されたリリースのパスワードをオーバーライドするには、この設定に入力します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
インストール先パス	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	インストール先パスは、リリースの配布先となる VM 上の場所を識別します。パスの最後のサブフォルダーとして、InstallShield が VM 上に作成するパスを使用できます。その他のパスは既存していなくてはなりません。 "構成" 設定で選択されている VM 構成にインストール先パスが指定されている場合、そのインストール先パスがこの設定のデフォルト値として取り扱われ、灰色テキストで表示されます。選択されたリリースのインストール先パスをオーバーライドするには、この設定に入力します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
ビルドの後、配布する	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	ビルドが成功するたびに選択されたリリースを、InstallShield で自動的に配布するかどうかを指定します。VM にリリースを配布するには、以下のタスクが必要です: 1. (指定されている場合) VM を指定されたスナップショットに戻す 2. VM の電源をオンにする。 3. テスト用にビルドされたリリースを VM にコピーする。 ビルド時に VM へ自動的にリリースを配布する代わりに、リリースをオンデマンドで配布することも可能です。そのためには、リリースを右クリックしてから [VM に配布] をクリックします。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

リリースの [Appx/MSIX] タブ

プロジェクト: *[Appx/MSIX] タブは、その後基本の MSI プロジェクトで使用できます。*

Windows アプリ パッケージ (.appx/。) の作成は、[Appx/MSIX] という名前のリリースごとに表示されるタブを使って有効化します。ここで、Windows アプリ パッケージのビルド プロセスに影響するいくつかの主要な設定を指定できます。

[Appx/MSIX] タブの設定は、次のメイン カテゴリに分かれています:

- 全般
- アプリ インストーラーの設定
- パッケージ識別子のオーバーライド
- 表示プロパティ

[全般] の設定

[Appx/MSIX] タブにある [全般] 領域を使って、Windows アプリ パッケージ (.appx) をリリースのビルドの一部としてビルドするかどうかを指定し、デスクトップ拡張またはサーバー拡張を含めるかどうかを選択するなど、ビルド プロセスに影響を及ぼすその他の主要な設定を指定します。

テーブル 216 : [Appx/MSIX] タブにある全般設定

設定	プロジェクトの種類	説明

Windows アプリ パッケージのビルド	基本の MSI	Windows アプリ パッケージ (.appx/.msix) をこのリリースのビルドの一部としてビルドするかどうかを指定します。
		
		選択可能なオプションは以下のとおりです:
		
		• **はい**— Windows アプリ パッケージ (.appx/.msix) をビルドします。**[はい]** を選択して、次の処理を行います:
		  ◦ アプリを Windows ストア経由で配布する。
		  ◦ Windows Server Nano をターゲットにする。
		  ◦ UWP (ユニバーサル Windows プラットフォーム) を活用する Appx/MSIX をサイドロードします。
		• **いいえ**— Windows アプリ パッケージをビルドしません。**[いいえ]** を選択した場合、.msi のみがビルドされます。

テーブル 216 • [Appx/MSIX] タブにある全般設定 (続き.)

設定	プロジェクトの種類	説明
パッケージの種類	基本の MSI	パッケージの種類を選択します。この設定は、Windows デスクトップ アプリケーションを新しい Windows アプリ パッケージ形式に変換します。選択可能なドロップダウン オプション: • APPX– Windows アプリ パッケージのビルド • MSIX– MSIX は、Windows プラットフォームの次世代ソフトウェア デプロイメント モデルで、MSI、AppX、および App–V の長所を単一パッケージに取り込みます。MSIX は、従来型の Win 32 アプリをコンテナーで実行する機能を持つため、アプリケーションの分離、デプロイメントの簡易化、完全にクリーンなインストール、およびシームレスなソフトウェア アップデートのすべてを同時に実現します。 • 両方– .appx および . msix パッケージの両方をビルドします。
配布方法	基本の MSI	Windows アプリ パッケージの配布方法を選択します。この設定は、Windows ストア ポリシー関連、またはサイドローディング関連の要件で問題の可能性がある場合に発生する一連の警告を指示します。選択可能なドロップダウン オプション: • **直接配布 (サイドロード)**– Windows アプリ パッケージを Windows ストアからのダウンロードなしで配布します。サイドロード アプリは、Windows ストアによって認証されていません。サイドロード アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。パッケージがサイドロードされる場合、カスタマが独自に署名することを希望しない限り、一般的に署名が必要です。 • **Windows ストア**– Windowsアプリ パッケージを Windows ストアを通して配布します。このオプションが選択されているとき、サイドロードでは有効なビルドであっても、Windows ストア ポリシーで拒絶されることが確認されているアイテムには、追加の警告が発生します。パッケージが Windows ストアを通して配布される場合、そのコンテンツを制限する追加ポリシーが適用されます。この種類のアプリは、パッケージが Microsoft によって署名されるため、Microsoft にアップロードする前に署名する必要がありません。

テーブル 216 • [Appx/MSIX] タブにある全般設定 (続き.)

設定	プロジェクトの種類	説明
デスクトップ拡張を含める	基本の MSI	コア Windows アプリ パッケージ (.appx/.msix Windows アプリ) フォーマットの一部として「フルトラスト」アプリや、その他の デスクトップ ブリッジ拡張を含めるかどうかを選択します。デスクトップ アプリを有効にして UWP (ユニバーサル Windows プラットフォーム) API を活用する場合、デスクトップ拡張を有効にする必要があります。 **[いいえ]** を選択した場合、使用不可能なアイテムには警告が発生します。 **[はい]** を選択した場合、含まれるアプリとデスクトップ拡張がサポートするテスト済みの Windows バージョン (最小および最大) を指定するための、追加のサブ設定を必要に応じて構成することができます。 ***ヒント:*** *InstallShield を使って Windows アプリ パッケージを作成するとき、デスクトップ拡張またはサーバー拡張のどちらかを含めることができますが、両方を選ぶことはできません。Windows アプリ パッケージにはデスクトップおよびサーバー拡張の両方を含めることが可能ですが、デスクトップまたはサーバーのいずれかをターゲットとすることが一般的です。*
最小バージョン (デスクトップ拡張を含めるの下)	基本の MSI	オプションで、アプリの最小サポート対象 Windows バージョン番号を指定します。この設定を空白のまま残した場合、Windows 10 Anniversary Update を最小バージョン番号とします。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1

テーブル 216 • [Appx/MSIX] タブにある全般設定 (続き.)

設定	プロジェクトの種類	説明
テスト済みバージョン (デスクトップ拡張を含めるの下)	基本の MSI	オプションで、アプリのテスト済み最大サポート対象 Windows バージョン番号を指定します。この設定を空白のまま残すと、規定値である最小バージョン (または、"最小バージョン" が指定されていない場合は Windows 10 Anniversary Update の最小バージョン) に設定されます。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1
サーバー拡張を含める	基本の MSI	コア Windows アプリ パッケージ (.appx/.msix Windows アプリ) フォーマットの一部として Windows サーバー拡張を含めるかどうか、つまり Windows サーバー アプリを作成するかどうかを選択します。Windows Server Nano のインストール パッケージ フォーマットをターゲットにしてサービス、VMI プロバイダー、イベント ソース、またはパフォーマンス カウンターを配置する場合、サーバー拡張を有効にする必要があります。 **[いいえ]** を選択した場合、使用不可能なアイテムには警告が発生します。 **[はい]** を選択した場合、含まれるアプリがサポートするテスト済みの Windows Server オペレーティング システム (最小および最大) を指定するための、追加のサブ設定を必要に応じて構成することができます。 ***ヒント:*** *InstallShield を使って Windows アプリ パッケージを作成するとき、デスクトップ拡張またはサーバー拡張のどちらかを含めることができますが、両方を選ぶことはできません。Windows アプリ パッケージにはデスクトップおよびサーバー拡張の両方を含めることが可能ですが、デスクトップまたはサーバーのいずれかをターゲットとすることが一般的です。*

テーブル 216 • [Appx/MSIX] タブにある全般設定 (続き.)

設定	プロジェクトの種類	説明
MSIX Core 拡張を含める	基本の MSI	MSIX Core を使って、Windows の以前のバージョンに MSIX アプリをインストールすることができます (アプリがその Windows バージョンに対応するようにビルドされている場合)。 MSIX Core は、現在で MSIX をサポートしない次の Windows バージョンにビルドされます: • Windows 7 SP1– 6.1.7601.0 • Windows 8.1 Update 1– 6.3.9600.0 • Windows 10 2015 LTSB (1507)– 10.0.10240.0 • Windows 10 2016 LTSB (1607)– 10.0.14393.0 • Windows Server 2008 R2 – 6.1.7601.0 • Windows Server 2012– 6.2.9200.0 • Windows Server 2012 R2 – 6.3.9600.0 • Windows Server 2016 – 10.0.14393.0 デフォルトで、**[いいえ]** オプションが選択されています。選択可能オプションは、以下のとおりです: • **[いいえ]** – このオプションを選択すると、MSIX Core のエントリがマニフェスト ファイルに追加されません。 • **[はい]** – このオプションを選択すると、MSIX Core エントリが、指定の最小 Windows バージョンと、テスト済みの最大 Windows バージョンと共にマニフェスト ファイルに追加されます。 ***ヒント:*** *MSIX Core パッケージには、古いバージョンのパッケージ上に MSIX 前提条件がインストールされている必要があります。*
最小バージョン (サーバー拡張を含めるの下)	基本の MSI	オプションで、アプリの最小サポート対象 Windows バージョン番号を指定します。この設定を空白のまま残した場合、Windows 7 SP1 Update (6.1。7601.0) を最小バージョン番号とします。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 6553 の値): *majorversion.minorversion.buildnumber.revisionnumber*

テーブル 216 • [Appx/MSIX] タブにある全般設定 (続き.)

設定	プロジェクトの種類	説明
テスト済みバージョン (サーバー拡張を含めるの下)	基本の MSI	オプションで、アプリのテスト済み最大サポート対象 Windows バージョン番号を指定します。この設定を空白に残すと、デフォルトでこの設定は最大 MSIX Core サポート対象バージョン番号 10.0.14393.0 (Windows 10 2016 LTSB – 1607) と一致します。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): majorversion.minorversion.buildnumber.revisionnumber
Framework の依存関係	基本の MSI	製品で使用されているのと一致する C++ ランタイムのリテール ビルドまたはデバッグ ビルドを選択します。 ***メモ•*** *この設定によって、サイドローディング シナリオで依存関係パッケージがインストールされることはありません。これらのシナリオの場合、依存関係は手動で配布されている可能性があります。また、C++ ランタイムのデバッグ バージョンは、内部テストで役立ちますが、Windows ストアにアップロードすることができません。ローカル テストまたはサイドローディングの場合、Windows 10 SDK に含まれる Framework パッケージも配布する必要があるかもしれません。*
依存関係の結合	基本の MSI	***エディション•*** *この設定は、InstallShield Premier Edition で使用できます。* 結果となるパッケージに結合させる Windows アプリ パッケージの一覧を指定します。省略記号ボタン (...) をクリックして、Windows アプリ パッケージ (.appx/.msix Windows アプリ) を参照するか、この文字列に 1 つ以上のパスをセミコロン区切りで直接入力します。 VFS からのレジストリ データおよびファイルがこのパッケージにコピーされますが、重複するデータはすべてパッケージから省略されます。

アプリ インストーラーの設定

[Appx/MSIX] タブにある [アプリ インストーラーの設定] 領域を使って、アプリ インストーラー ファイルを生成し、パッケージのアップデート機能を構成します。

アプリ インストーラー ファイルは XML ドキュメント (*.appinstaller) で、パッケージのプロパティを定義する一連の属性、およびそれが展開される場所が含まれています。アプリ インストーラー ファイルは Windows 10 バージョン 1709 で導入されました。ここにアプリ packageの場所、およびその更新方法が指定されています。このメソッドを使用してアプリの配布を行う場合、実際のアプリパッケージの代わりにアプリ インストーラー ファイルをユーザーと共有する必要があります。ユーザーは、そのアプリ インストーラー ファイルをクリックします。こ

の時点で、見慣れたアプリ インストーラー UI が表示され、ユーザーに対してインストールの手順が案内されます。ユーザーがこの手順に従ってアプリケーションをインストールすると、そのアプリケーションが アプリ インストーラー ファイルに関連付けられます。

後程、アプリケーションにアップデートを行った場合、アプリ インストーラー ファイル (.appinstaller) を更新するだけで良くなります。このファイルを更新すると、次の [アップデートの設定] セクションの構成に従ってアプリケーションの新バージョンが自動的にユーザーにプッシュされます。

詳細は、「App Installer File Schema (アプリ インストーラー ファイルのスキーマ)」を参照してください。

テーブル 217 • [Appx/MSIX] タブにあるアプリ インストーラーの設定

設定	プロジェクトの種類	説明
AppInstaller ファイル URL	基本の MSI	アプリ インストーラー ファイルの場所を指定します。 アプリ インストーラー ファイルは、HTTP エンドポイント、UNC 共有フォルダー、ローカル ファイル共有、などの共有場所でホストすることが可能で、インストールされる MSIX パッケージを検索するためのパスが含まれています。ユーザーはこのアプリを共有場所または Web からインストールして、定期的な新しいアップデートのチェックを有効化します。
AppInstaller バージョン	基本の MSI	アプリ インストーラー ファイルのバージョンを指定します。4 つの部分から成るバージョン文字列 (Major.Minor.Build.Revision)。例: 1.0.0.60325
メイン パッケージ URL	基本の MSI	アプリ パッケージの場所への URI を指定します。このオプションが空白の場合、[**AppInstaller ファイル URL**] に指定された URI が適用されます。MainPackage 要素のその他の情報 (名前、発行元、およびバージョンを含む) は、[このページの [**パッケージ識別子のオーバーライド**] を使って変更することができます。**[パッケージ ID のオーバーライド]** で提供されている情報は、 MSIX アプリケーション パッケージのマニュフェストに含まれている情報と一致する必要があるため、これらは MSIX パッケージとアプリ インストーラー ファイルの両方でビルド中に考慮されます。
アップデートの設定	基本の MSI	**[アップデートの設定]** カテゴリーを使って、アプリ インストーラー ファイルを使った MSIX パッケージの自動更新機能を構成することができます。
アプリ起動時のアップデートの設定	基本の MSI	アプリ起動中に、このパッケージのアップデート設定を構成するかどうかを指定します。OnLaunch 中のこの動作は UI を表示し、HoursBetweenUpdateChecks、ShowPrompt、および UpdateBlocksActivation 子属性サポートを使って、次で説明されているようにさらに詳細に構成することができます。

設定	プロジェクトの種類	説明
アップデート確認の間隔 (時間)	基本の MSI	HoursBetweenUpdateChecks オプションを使って、デプロイメント サービスがアプリ インストーラー ファイルへのアップデートをチェックする頻度を指定します。 HoursBetweenUpdateChecks が 0 に設定された場合、デプロイメント サービスはアプリケーションが起動される度にアップデートをチェックします。その他の値の場合、HoursBetweenUpdateChecks で指定されている最後の時間数内にチェックされていない場合のみ、アプリケーションが起動されたときにデプロイメント サービスがアップデートのチェックを行います。 たとえば、HoursBetweenUpdateChecks が 10 に設定されている場合、過去 10 時間以内にアップデートがチェックされていない場合のみ、アプリケーションが起動されたときにデプロイメント サービスがアップデートをチェックします。Windows デプロイメント サービスは、新しいアップデートの使用が可能な場合にそれを自動的にダウンロードし、アプリケーションが閉じた後にそのアップデートをインストールします。
プロンプトを表示	基本の MSI	デプロイメントが、ユーザーに対してアップデートについての情報を表示するプロンプトを表示するかどうかを指定します。Windows 10 バージョン 1903 以降で使用可能です。 最新アップデートについては、Microsoft ドキュメンテーションを参照してください。
アップデートがアクティベーションをブロック	基本の MSI	**[はい]** を選択すると、アップデートが適用されるまでユーザーがアプリケーションを起動しないように阻止します。このオプションは、ユーザーがアップデートを選択するか、アプリを閉じることできるようにします。オプションを表示するための UI が必要なため、ShowPrompt オプションが選択されている場合のみ使用することができます。このオプションを使って、いくつかの重要な修正と共にアプリを起動する必要があるクリティカル アップデートを配布するのに役立ちます。
自動バックグラウンド タスク	基本の MSI	アプリが起動されているかどうかに関わらず、バックグラウンドでアップデートをチェックするかどうかを指定します。ユーザーがアプリを起動するかどうかにかかわらず、8 時間おきにチェックが行われます。この種類のアップデートで、ユーザー インターフェイスを表示することはできません。アップデート処理がトリガされる前に、インタラクティブ UI を表示するには、[アプリ起動時のアップデート設定] を選択します。Windows 10 バージョン 1803 以降で使用可能です。

設定	プロジェクトの種類	説明
任意のバージョンからのアップデートを強制する	基本の MSI	アプリのバージョン番号を要件に基づいて増加または減少させるかどうかを指定します。この要素が無い場合、アプリは上位バージョンにのみアップデートが可能で、Windows デプロイメント サービスは古いバージョンへのダウングレードを許可しません。このオプションは、古いバージョンのアプリ パッケージ バージョンにロールバック処理が必要な時に便利です。Windows 10 バージョン 1809 以降から使用可能です。

パッケージ識別子のオーバーライドの設定

[Appx/MSIX] タブの [パッケージ ID のオーバーライド] 領域を使って、Windows に対してパッケージを一意に識別します。これらのフィールドは、通常エンド ユーザーには表示されません。また、設定がオーバーライドされるため、既定でフィールドは拡張されません。

メモ: *アーキテクチャも、Windowsアプリ パッケージの特徴の 1 つです。[製品構成] または [一般情報] にある ″テンプレート概要″ 設定を使って、Windows アプリ パッケージのアーキテクチャを指定します。*

テーブル 218 · [Appx/MSIX] タブにあるパッケー識別子のオーバーライドの設定

設定	プロジェクトの種類	説明
名前	基本の MSI	Windows に対して Windows アプリ パッケージを識別する一意の英数字文字列を指定します。この設定が空白の場合、InstallShield によって [製品構成] または [一般情報] にある **″製品名″** 設定に基づいた名前が生成されます。

テーブル 218 • [Appx/MSIX] タブにあるパッケー識別子のオーバーライドの設定

設定	プロジェクトの種類	説明
発行者	基本の MSI	[署名] タブで証明書の実際の値が指定されている場合に、これをオーバーライドするには、パッケージの署名に使用する証明書の Sサブジェクト名 (所有者の名前) を指定します。例: CN=Revenera、O=Revenera、L=Itasca、S=Illinois C=US Windows ストアを通してパッケージが配布される場合、Microsoft にアップロードする前に署名を行う必要はないので、証明書の署名をスキップしても構いません。パッケージがサイドロードされるとき、カスタマーによるパッケージの使用方法によって、これに署名が必要な場合と不要な場合があります: • カスタマーがパッケージの署名を自身で行いたい場合、証明書と一致する ***"パブリッシャー"*** オーバーライド設定で定義された特定のパブリッシャーが必要です。 • カスタマーがパッケージの配布のみを行いたい場合、信頼された Authenticode 証明書で署名されている必要があります。その要件は、信頼された UAC プロンプトの要件と同様です。 • サイドロードを行うには、Windows の "サイドロード" 設定が有効でなくてはなりません。Windows 10 Anniversary Update では、既定でサイドロードが有効化されています。 ***ヒント:*** *[リリース] ビューの [署名] タブにある* ***"デジタル証明書情報"、"証明書パスワード"*** *、および* ***"出力ファイルの署名"*** *設定もまた、Windowsアプリ パッケージに署名を行うかどうか、およびその方法を制御し、ビルドされたパッケージに影響を与えます:*
バージョン	基本の MSI	証明書が準拠する Windows アプリ パッケージのバージョン番号を入力します。何も指定しなかった場合、[一般情報] または [製品構成] の ***"製品バージョン"*** 設定からバージョン番号がコピーされます。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1

表示プロパティ

[Appx/MSIX] タブの [表示プロパティ] 領域にある 4 つのフィールドを使って、エンド ユーザーに対してパッケージを識別します。

テーブル 219 • [Appx/MSIX] タブにある表示プロパティの設定

設定	プロジェクトの種類	説明
表示名	基本の MSI	パッケージの名前として、ユーザーが認識できる文字列を指定します。通常、アプリケーションの名前を使用します。何も指定しなかった場合、[一般情報] または [製品構成] の **"製品名"** から表示名がコピーされます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ•*** *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージをビルドします。*
発行元の表示名	基本の MSI	発行元の名前として、ユーザーが認識できる文字列を指定します。例: Revenera 何も指定しなかった場合、[一般情報] または [製品構成] の **"発行元"** 設定から表示名がコピーされます。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ•*** *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージをビルドします。*

テーブル 219 • [Appx/MSIX] タブにある表示プロパティの設定 (続き.)

設定	プロジェクトの種類	説明
説明	基本の MSI	オプションで、エンド ユーザーがパッケージを識別するのに役立つ説明を入力します。 この設定に値を入力すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。新しい値を入力する代わりに、この設定で省略記号ボタン (...) をクリックして既存の文字列を選択することもできます。詳細については、「InstallShield で文字列エントリを使用する」を参照してください。 ***メモ•*** *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージをビルドします。*
Logo	基本の MSI	パッケージを表すイメージを指定します。通常 50x50 のロゴを使用します。ロゴが指定されなかった場合、[一般情報] の **"表示アイコン"** 設定から作成されます。

リリースの [アップデート] タブ

プロジェクト• *[更新] タブは、その後スイート/アドバンスト UI プロジェクトで使用できます。*

その他にプロジェクトによって異なる情報がある場合は、その内容が記載されています。

[アップデート] タブを使って、スイート/アドバンストUI ランチャーのアップデート設定を構成します。i

[アップデート] タブの設定は、次のメイン カテゴリに分かれています:

- 全般

  アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートの **[全般]** 領域をサポート:

  - ダウンロード可能なアップデートのサポートを有効化する
  - ダウンロード可能なアップデートのより新しいバージョンを構成する
  - 確認メッセージを構成する
  - ダウンロード可能なアップデートのサポートを追加する
- アプリのアップデートを確認する

全般

[アップデート] タブの **[全般]** 領域を使って、ダウンロード可能なアップデートの設定をサポートするための設定を構成します。

テーブル 220 · [アップデート] タブにある [全般] 設定

設定	プロジェクトの種類	説明
アップデートの有効化	スイート/アドバンスト UI	アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのダウンロード可能なアップデート サポートを有効化します。
ユーザーに確認する	スイート/アドバンスト UI	エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するかどちらかのオプションを受け取ります。
強制的にインストール	スイート/アドバンスト UI	エンド ユーザーはスイート パッケージ (使用可能な場合) の新しいバージョンを強制的にダウンロードおよびインストールすることができます。
確認メッセージ	スイート/アドバンスト UI	エンド ユーザーがより新しいバージョンをダウンロードするか、スキップして現在のインストール処理を続行するためのメッセージを入力します。
アップデートURL	スイート/アドバンスト UI	アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーのアップデートをダウンロード可能にしたい場合、ファイル名を含む絶対パスを指定します。 アップデートの要件を含む、アドバンスト UI またはスイート/アドバンスト UI インストールのアップデートに関する詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」をご覧ください。

アプリのアップデートを確認する

[アップデート] タブにある **[アプリのアップデート確認する]** 領域を使って、アプリケーションのアップデートを定期的に確認するための追加設定を構成できます。

テーブル 221 · [アップデート] タブのアップデートを確認する "アプリのアップデートを確認する確認する" 設定

設定	設定	説明
アプリのアップデートを定期的に確認する	スイート/アドバンスト UI	この設定を使って、アップデートを定期的に確認するための追加の設定を構成するかどうかを指定できます。選択可能なオプションは以下のとおりです: · このオプションを **[はい]** に設定すると、**"アップデート確認の頻度"** 設定が有効化されます。 · このオプションを **[いいえ]** に設定すると、**"アップデート確認の頻度"** 設定が無効化されます。既定で、この設定は **[いいえ]** となっています。
アップデート確認の頻度	スイート/アドバンスト UI	更新プログラムの確認頻度を決定付ける、定期的な間隔を指定します。この設定は、**"アプリのアップデートを定期的に確認する"** 設定 **[はい]** に設定されている場合に有効です。選択可能なオプションは以下のとおりです: · **毎日** – 更新プログラムを毎日確認するには、このオプションを選択します。 · **毎時** – 更新プログラムを毎時確認するには、このオプションを選択します。 · **毎週** – 更新プログラムを毎週確認するには、このオプションを選択します。 ***メモ·** 既定で、この設定は **[毎日]** になっています。*
確認する間隔	スイート/アドバンスト UI	更新プログラムを確認する時間間隔または曜日と週数を指定します。この設定は **"アップデート確認の頻度"** 設定で選択したオプションによって異なります。 · **"アップデート確認の頻度"** 設定が **[毎時]** に設定設定されている場合、正確な時間数をドロップダウン リストから選択します。 · **"アップデート確認の頻度"** 設定が **[毎週]** に設定設定されている場合、省略記号ボタン (...) をクリックしてから、曜日と週数を指定する適切なチェックボックスを選択します。 ***メモ· "アップデート確認の頻度"** 設定が **[毎日]** に設定設定されている場合、この設定は無効です。*

テーブル 221 • [アップデート] タブのアップデートを確認する ”アプリのアップデートを確認する確認する” 設定

設定	設定	説明
アップデート ダイアログのタイトル	スイート/アドバンスト UI	アプリケーションのアップデート ダイアログ ボックスのタイトルを指定します。
アップデート ダイアログのテキスト	スイート/アドバンスト UI	アプリケーションのアップデート ダイアログ ボックスのテキストを指定します。

MSIX リリースの [全般] タブ

プロジェクト• *[全般] タブは、次のプロジェクト タイプで使用できます:*

- *MSIX*
- *MSIX 変更パッケージ*

[全般] タブには、以下の設定があります。これらの設定を利用して、作成したリリースで他の設定をオーバーライドすることができます。

テーブル 222 • MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
全般	MSIX、MSIX 変更パッケージ	[全般] タブにある [設定] ビューを使って、デスクトップ拡張またはサーバー拡張および署名の詳細などを含むビルド プロセスに影響を与える重要な設定を指定します。
リリースの場所	MSIX、MSIX 変更パッケージ	リリースのビルドを開始するトップレベルのディレクトリのパスを入力するか、省略記号ボタン (...) をクリックして場所を参照します。
パッケージのアーキテクチャ	MSIX、MSIX 変更パッケージ	パッケージに含まれているコードのアーキテクチャを説明します。実行可能なコードを含むパッケージは、この属性を含める必要があります。 使用可能なコードは次の通りです: • **x86** • **x64** • **ARM** • **ARM64** • **ニュートラル**

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
配布方法	MSIX、MSIX 変更パッケージ	MSIX パッケージの配布方法を選択します。この設定は、Windows ストア ポリシー関連、またはサイドローディング関連の要件で問題の可能性がある場合に発生する一連の警告を指示します。選択可能なドロップダウン オプション: • **直接配布 (サイドロード)** – MSIX パッケージを Windows ストアからのダウンロードなしで配布します。サイドロード アプリは、Windows ストアによって認証されていません。サイドロード アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。パッケージがサイドロードされる場合、カスタマが独自に署名することを希望しない限り、一般的に署名が必要です。 • **Windows ストア** – MSIX パッケージを Windows ストアを通して配布します。このオプションが選択されているとき、サイドロードでは有効なビルドであっても、Windows ストア ポリシーで拒絶されることが確認されているアイテムには、追加の警告が発生します。パッケージが Windows ストアを通して配布される場合、そのコンテンツを制限する追加ポリシーが適用されます。この種類のアプリは、パッケージが Microsoft によって署名されるため、Microsoft にアップロードする前に署名する必要がありません。
Universal拡張を含める	MSIX、MSIX 変更パッケージ	コア UWP アプリ パッケージ フォーマットの一部として Universal デバイス拡張を含めるかどうかを選択します。アプリを有効にして UWP (ユニバーサル Windows プラットフォーム) API を活用する場合、Universal拡張を有効にする必要があります。 選択した場合、 • **[いいえ]** – 使用不可能なアイテムには警告が発生します。 • **[はい]** – 含まれるアプリと Universal 拡張がサポートする最小 Windows バージョンおよびテスト済みの最大 Windows バージョンを指定するための、追加のサブ設定を必要に応じて構成することができます。

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
デスクトップ拡張を含める	MSIX、MSIX 変更パッケージ	コア UWP アプリ パッケージ フォーマットの一部として「フルトラスト」アプリや、その他の デスクトップ ブリッジ拡張を含めるかどうかを選択します。デスクトップ アプリを有効にして UWP (ユニバーサル Windows プラットフォーム) API を活用する場合、デスクトップ拡張を有効にする必要があります。 選択した場合、 ・ [いいえ] – 使用不可能なアイテムには警告が発生します。 ・ [はい] – 含まれるアプリとデスクトップ拡張がサポートするテスト済みの Windows バージョン (最小および最大) を指定するための、追加のサブ設定を必要に応じて構成することができます。 ***重要**・InstallShield を使って UWP アプリ パッケージを作成するとき、デスクトップ拡張またはサーバー拡張のどちらかを含めることができますが、両方を選ぶことはできません。UWP アプリ パッケージにはデスクトップおよびサーバー拡張の両方を含めることが可能ですが、デスクトップまたはサーバーのいずれかをターゲットとすることが一般的です。*
最小バージョン	MSIX、MSIX 変更パッケージ	オプションで、アプリの最小サポート対象 Windows バージョン番号を指定します。この設定を空白のまま残した場合、Windows 10 Anniversary Update を最小バージョン番号とします。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): majorversion.minorversion.buildnumber.revisionnumber。例、 ・ 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: ・ 14.0.25.1

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
テスト済みバージョン	MSIX、MSIX 変更パッケージ	オプションで、アプリのテスト済み最大サポート対象 Windows バージョン番号を指定します。この設定を空白のまま残すと、規定値である最小バージョン (または、"最小バージョン" が指定されていない場合は Windows 10 Anniversary Update の最小バージョン) に設定されます。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): majorversion.minorversion.buildnumber.revisionnumber。例、 ・ 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: ・ 14.0.25.1
サーバー拡張を含める	MSIX、MSIX 変更パッケージ	コア UWP アプリ パッケージ フォーマットの一部として Windows サーバー拡張を含めるかどうか、つまり Windows サーバー アプリを作成するかどうかを選択します。Windows Server Nano のインストール パッケージ フォーマットをターゲットにしてサービス、VMI プロバイダー、イベント ソース、またはパフォーマンス カウンターを配置する場合、サーバー拡張を有効にする必要があります。 選択した場合、 ・ [いいえ] – 使用不可能なアイテムには警告が発生します。 ・ [はい] – 含まれるアプリがサポートするテスト済みの Windows Server オペレーティング システム (最小および最大) を指定するための、追加のサブ設定を必要に応じて構成することができます。 ***重要・****InstallShield を使って UWP アプリ パッケージを作成するとき、デスクトップ拡張またはサーバー拡張のどちらかを含めることができますが、両方を選ぶことはできません。UWP アプリ パッケージにはデスクトップおよびサーバー拡張の両方を含めることが可能ですが、デスクトップまたはサーバーのいずれかをターゲットとすることが一般的です。*

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
MSIX Core 拡張を含める	MSIX、MSIX 変更パッケージ	MSIX Core を使って、Windows の以前のバージョンに MSIX アプリをインストールすることができます (アプリがその Windows バージョンに対応するようにビルドされている場合)。 MSIX Core は、現在で MSIX をサポートしない次の Windows バージョンにビルドされます: ・ Windows 7 SP1– 6.1.7601.0 ・ Windows 8.1 Update 1– 6.3.9600.0 ・ Windows 10 2015 LTSB (1507)– 10.0.10240.0 ・ Windows 10 2016 LTSB (1607)– 10.0.14393.0 ・ Windows Server 2008 R2 – 6.1.7601.0 ・ Windows Server 2012– 6.2.9200.0 ・ Windows Server 2012 R2 – 6.3.9600.0 ・ Windows Server 2016 – 10.0.14393.0 デフォルトで、**[いいえ]** オプションが選択されています。選択可能オプションは、以下のとおりです: ・ **[いいえ]** – このオプションを選択すると、MSIX Core のエントリがマニフェスト ファイルに追加されません。 ・ **[はい]** – このオプションを選択すると、MSIX Core エントリが、指定の最小 Windows バージョンと、テスト済みの最大 Windows バージョンと共にマニフェスト ファイルに追加されます。 ***ヒント:*** *MSIX Core パッケージには、古いバージョンのパッケージ上に MSIX 前提条件がインストールされている必要があります。*
最小バージョン	MSIX、MSIX 変更パッケージ	オプションで、アプリの最小サポート対象 Windows バージョン番号を指定します。この設定を空白のまま残した場合、Windows 7 SP1 Update (6.1。7601.0) を最小バージョン番号とします。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 6553 の値): *majorversion.minorversion.buildnumber.revisionnumber*

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
テスト済みバージョン	MSIX、MSIX 変更パッケージ	オプションで、アプリのテスト済み最大サポート対象 Windows バージョン番号を指定します。この設定を空白に残すと、デフォルトでこの設定は最大 MSIX Core サポート対象バージョン番号 10.0.14393.0 (Windows 10 2016 LTSB – 1607) と一致します。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*
Framework の依存関係	MSIX、MSIX 変更パッケージ	製品で使用されているのと一致する C++ ランタイムのリテール ビルドまたはデバッグ ビルドを選択します。 ***重要**・この設定によって、サイドローディング シナリオで依存関係パッケージがインストールされることはありません。これらのシナリオの場合、依存関係は手動で配布されている可能性があります。また、C++ ランタイムのデバッグ バージョンは、内部テストで役立ちますが、Windows ストアにアップロードすることができません。ローカル テストまたはサイドローディングの場合、Windows 10 SDK に含まれる Framework パッケージも配布する必要があるかもしれません。* Framework 依存関係をカスタマイズするには、次の手順に従います: 次の場所にインストールされている、Msix.xml ファイルを使用することができます。 *InstallShield Program Files フォルダー*¥System 次はその一例です: <VC140 Name="Microsoft.VCLibs.140.00.UWPDesktop" MinVersion="14.0.24217.0" Publisher="CN=Microsoft Corporation, O=Microsoft Corporation, L=Redmond, S=Washington, C=US">Microsoft Visual C++ 2015 Runtime</VC140>
署名	MSIX、MSIX 変更パッケージ	[署名] カテゴリーを使って、(証明機関より付与されたデジタル証明書ファイルを含み) InstallShield がファイルに署名するときに使用するデジタル署名に関する情報を指定します。
MSIX パッケージの署名	MSIX、MSIX 変更パッケージ	MSIX パッケージに署名を行うかどうかを指定します。
証明書 URL	MSIX、MSIX 変更パッケージ	完全修飾 URL を入力します (例、http://www.mydomain.com)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。

テーブル 222 • MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
デジタル証明書情報	MSIX、MSIX 変更パッケージ	リリースに署名を行うために使用するデジタル証明書を指定するには、この設定で省略記号ボタン (...) をクリックします。[証明書の選択] ダイアログ ボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。 InstallShield では、パブリック キー証明書 (.cer) ファイルを使って EV トークン パスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。.cer ファイルは通常、USB eToken プロバイダー (たとえば、SafeNet 認証クライアント) と関連付けられた EV 認証クライアント ツールからパブリック キーをエクスポートして作成されます。.cer ファイルが指定された場合、InstallShield は Extended Validation 証明書 (EV) プロパティを構成するための追加オプションを表示します。詳細については、次を参照してください。 • InstallShield における拡張検証 (EV) 証明書情報の構成 • [証明書の選択] ダイアログ ボックス .pfx ファイルを指定するか、テスト ストアから証明書を選択すると、次が表示されます: • **証明書の拇印** – この読み取り専用設定は、証明書の拇印を表示します。 • **発行元** – この読み取り専用設定は、証明書の発行元情報を表示します。 • **有効期限日** – この読み取り専用設定は、証明書の有効期限日を表示します。 証明書ファイル (EV がエクスポートした .cer ファイル) を指定すると、次の設定が表示されます: • **暗号化サービス プロバイダー** – この設定を使って、暗号化サービス プロバイダー (CSP) を指定します。 • **コンテナ名** – この設定を使って、暗号化サービス プロバイダー (CSP) に関連付けられた秘密キーのコンテナ名を指定します。 • **トークン パスワード** – この設定を使って、暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。

テーブル 222 • MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
デジタル証明書情報 (続き)	MSIX、MSIX 変更パッケージ	 ***メモ:*** *証明書ファイル (EV がエクスポートした .cer ファイル) を指定した後に表示される設定を使用する際は、以下の情報に注意してください:* • *暗号化サービス プロバイダー (CSP) 名とコンテナ名の両方は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*
証明書の拇印	MSIX、MSIX 変更パッケージ	この読み取り専用設定は、証明書の拇印を表示します。
発行元	MSIX、MSIX 変更パッケージ	この読み取り専用設定は、証明書の発行元情報を表示します。
有効期限日	MSIX、MSIX 変更パッケージ	この読み取り専用設定は、証明書の有効期限日を表示します。
暗号化サービス プロバイダー	MSIX、MSIX 変更パッケージ	暗号化サービス プロバイダー (CSP) 名を指定します。 ***メモ:*** *暗号化サービス プロバイダー (CSP) 名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*
コンテナ名	MSIX、MSIX 変更パッケージ	暗号化サービス プロバイダー (CSP) に関連付けられた秘密キー コンテナ名を指定します。 ***メモ:*** *コンテナ名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
トークン パスワード	MSIX、MSIX 変更パッケージ	暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。 ***メモ・** EV トークンのパスワードを指定する際には、以下の情報に注意してください:* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*
証明書パスワード	MSIX、MSIX 変更パッケージ	使用する .pfx にパスワードがある場合、それを入力します。InstallShield はパスワードを暗号化してプロジェクト ファイル (.isl) に保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ***重要・** ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。*
64 ビット署名を使用	MSIX、MSIX 変更パッケージ	64 ビット署名フレームワークを使用してパッケージにデジタル署名を行うかどうかを指定します。選択可能なオプションは以下のとおりです: • **はい** – 64 ビット署名フレームワークを有効にして、パッケージにデジタル署名を行います。 • **いいえ** – 32 ビット署署名フレームワークを有効にして、パッケージにデジタル署名を行います。 デフォルトで、この設定は [いいえ] になっています。

テーブル 222 • MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
署名の説明	MSIX、MSIX 変更パッケージ	指定されたファイルに使用する署名の説明を指定します。 署名出力ファイル設定ここで指定した説明は、UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示されます。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。 この設定を空白のままに残すと、InstallShield は UAC (ユーザー アカウント制御) ボックスの [プログラム名:] ラベルの右側に表示される説明として、ファイル名を拡張子なしで使用します。"パッケージ" 設定およびそのサブ設定で [ファイルに署名] を選択して、パッケージ内のファイルに署名を行った場合、InstallShield はビルド時に署名されるパッケージ内のファイルの UAC ダイアログ ボックスにこの説明を使用しません。
パッケージ識別子のオーバーライド	MSIX、MSIX 変更パッケージ	[全般] タブの [パッケージ ID のオーバーライド] 領域を使って、Windows に対してパッケージを一意に識別します。これらのフィールドは、通常エンド ユーザーには表示されません。
名前	MSIX、MSIX 変更パッケージ	Windows に対して UWP アプリ パッケージを識別する一意の英数字文字列を指定します。この設定が空白の場合、InstallShield によって [一般情報] にある **"パッケージ名"** 設定に基づいた名前が生成されます。

テーブル 222・MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
発行者	MSIX、MSIX 変更パッケージ	[全般] タブで証明書の実際の値が指定されている場合に、これをオーバーライドするには、パッケージの署名に使用する証明書のサブジェクト名 (所有者の名前) を指定します。例: CN=Revenera、O=Revenera、L=Itasca、S=Illinois C=US Windows ストアを通してパッケージが配布される場合、Microsoft にアップロードする前に署名を行う必要はないので、証明書の署名をスキップしても構いません。パッケージがサイドロードされるとき、カスタマーによるパッケージの使用方法によって、これに署名が必要な場合と不要な場合があります: ・ カスタマーがパッケージの署名を自身で行いたい場合、証明書と一致する **"発行元"** オーバーライド設定で定義された特定の発行元が必要です。 ・ カスタマーがパッケージの配布のみを行いたい場合、信頼された Authenticode 証明書で署名されている必要があります。その要件は、信頼された UAC プロンプトの要件と同様です。 ・ サイドロードを行うには、Windows の "サイドロード" 設定が有効でなくてはなりません。Windows 10 Anniversary Update では、既定でサイドロードが有効化されています。 [リリース] ビューの [全般] タブにある次の設定もまた、UWP アプリ パッケージに署名を行うかどうか、およびその方法を制御し、ビルドされたパッケージに影響を与えます: ・ デジタル証明書情報 ・ 証明書パスワード ・ 出力ファイルに署名する
発行元の表示名	MSIX、MSIX 変更パッケージ	発行元の名前として、ユーザーが認識できる文字列を指定します。例: Revenera 顧客が独自にパッケージに署名を行う場合で、何も指定されていないとき、表示名は [全般] タブにある **"パブリッシャー"** 設定からコピーされます。

テーブル 222 • MSIX リリースの [全般] タブ

設定	プロジェクトの種類	説明
バージョン	MSIX、MSIX 変更パッケージ	証明書が準拠する UWP アプリ パッケージのバージョン番号を入力します。何も指定しなかった場合、[一般情報] の **"製品バージョン"** 設定からバージョン番号がコピーされます。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): majorversion.minorversion.buildnumber.revisionnumber。例、 • 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: • 14.0.25.1

MSIX リリースの [アプリ インストーラー] タブ

プロジェクト• *[アプリ インストーラー] タブは、の その後MSIX プロジェクトで使用できます。*

Windows アプリ パッケージの作成は、[アプリ インストーラー] という名前のタブでリリースごとに有効化されます。ここで、Windows アプリ パッケージのビルド プロセスに影響するいくつかの主要な設定を指定できます。

テーブル 223 • MSIX リリースの [アプリ インストーラー] タブの設定

設定	プロジェクトの種類	説明
アプリ インストーラーの設定	MSIX	[アプリ インストーラー] タブにある [アプリ インストーラーの設定] 領域を使って、アプリ インストーラー ファイルを生成し、パッケージのアップデート機能を構成します。 アプリ インストーラー ファイルは XML ドキュメント (*.appinstaller) で、パッケージのプロパティを定義する一連の属性、およびそれが展開される場所が含まれています。アプリ インストーラー ファイルは Windows 10 バージョン 1709 で導入されました。ここにアプリ package の場所、およびその更新方法が指定されています。このメソッドを使用してアプリの配布を行う場合、実際のアプリパッケージの代わりにアプリ インストーラー ファイルをユーザーと共有する必要があります。ユーザーは、そのアプリ インストーラー ファイルをクリックします。この時点で、見慣れたアプリ インストーラー UI が表示され、ユーザーに対してインストールの手順が案内されます。ユーザーがこの手順に従ってアプリケーションをインストールすると、そのアプリケーションが アプリ インストーラー ファイルに関連付けられます。 後程、アプリケーションにアップデートを行った場合、アプリ インストーラー ファイル (.appinstaller) を更新するだけで良くなります。このファイルを更新すると、次の [アップデートの設定] セクションの構成に従ってアプリケーションの新バージョンが自動的にユーザーにプッシュされます。 詳細は、「App Installer File Schema (アプリ インストーラー ファイルのスキーマ)」を参照してください。
AppInstaller ファイル URL	MSIX	アプリ インストーラー ファイルの場所を指定します。 アプリ インストーラー ファイルは、HTTP エンドポイント、UNC 共有フォルダー、ローカル ファイル共有、などの共有場所でホストすることが可能で、インストールされる MSIX パッケージを検索するためのパスが含まれています。ユーザーはこのアプリを共有場所または Web からインストールして、定期的な新しいアップデートのチェックを有効化します。
AppInstaller バージョン	MSIX	アプリ インストーラー ファイルのバージョンを指定します。4 つの部分から成るバージョン文字列 (Major.Minor.Build.Revision)。例: 1.0.0.60325

設定	プロジェクトの種類	説明
メイン パッケージ URL	MSIX	アプリ パッケージの場所への URI を指定します。このオプションが空白の場合、[**AppInstaller ファイル URL**] に指定された URI が適用されます。MainPackage 要素のその他の情報 (名前、発行元、およびバージョンを含む) は、[このページの [**パッケージ識別子のオーバーライド**] を使って変更することができます。**[パッケージ ID のオーバーライド]** で提供されている情報は、 MSIX アプリケーション パッケージのマニュフェストに含まれている情報と一致する必要があるため、これらは MSIX パッケージとアプリ インストーラー ファイルの両方でビルド中に考慮されます。
アップデートの設定	MSIX	**[アップデートの設定]** カテゴリーを使って、アプリ インストーラー ファイルを使った MSIX パッケージの自動更新機能を構成することができます。
アプリ起動時のアップデートの設定	MSIX	アプリ起動中に、このパッケージのアップデート設定を構成するかどうかを指定します。OnLaunch 中のこの動作は UI を表示し、HoursBetweenUpdateChecks、ShowPrompt、および UpdateBlocksActivation 子属性サポートを使って、次で説明されているようにさらに詳細に構成することができます。
アップデート確認の間隔 (時間)	MSIX	HoursBetweenUpdateChecks オプションを使って、デプロイメント サービスがアプリ インストーラー ファイルへのアップデートをチェックする頻度を指定します。 HoursBetweenUpdateChecks が 0 に設定された場合、デプロイメント サービスはアプリケーションが起動される度にアップデートをチェックします。その他の値の場合、HoursBetweenUpdateChecks で指定されている最後の時間数内にチェックされていない場合のみ、アプリケーションが起動されたときにデプロイメント サービスがアップデートのチェックを行います。 たとえば、HoursBetweenUpdateChecks が 10 に設定されている場合、過去 10 時間以内にアップデートがチェックされていない場合のみ、アプリケーションが起動されたときにデプロイメント サービスがアップデートをチェックします。Windows デプロイメント サービスは、新しいアップデートの使用が可能な場合にそれを自動的にダウンロードし、アプリケーションが閉じた後にそのアップデートをインストールします。
プロンプトを表示	MSIX	デプロイメントが、ユーザーに対してアップデートについての情報を表示するプロンプトを表示するかどうかを指定します。Windows 10 バージョン 1903 以降で使用可能です。 最新アップデートについては、Microsoft ドキュメンテーションを参照してください。

設定	プロジェクトの種類	説明
アップデートがアクティベーションをブロック	MSIX	**[はい]** を選択すると、アップデートが適用されるまでユーザーがアプリケーションを起動しないように阻止します。このオプションは、ユーザーがアップデートを選択するか、アプリを閉じることできるようにします。オプションを表示するための UI が必要なため、ShowPrompt オプションが選択されている場合のみ使用することができます。このオプションを使って、いくつかの重要な修正と共にアプリを起動する必要があるクリティカル アップデートを配布するのに役立ちます。
自動バックグラウンド タスク	MSIX	アプリが起動されているかどうかに関わらず、バックグラウンドでアップデートをチェックするかどうかを指定します。ユーザーがアプリを起動するかどうかにかかわらず、8 時間おきにチェックが行われます。この種類のアップデートで、ユーザー インターフェイスを表示することはできません。アップデート処理がトリガされる前に、インタラクティブ UI を表示するには、[アプリ起動時のアップデート設定] を選択します。Windows 10 バージョン 1803 以降で使用可能です。
任意のバージョンからのアップデートを強制する	MSIX	アプリのバージョン番号を要件に基づいて増加または減少させるかどうかを指定します。この要素が無い場合、アプリは上位バージョンにのみアップデートが可能で、Windows デプロイメント サービスは古いバージョンへのダウングレードを許可しません。このオプションは、古いバージョンのアプリ パッケージ バージョンにロールバック処理が必要な時に便利です。Windows 10 バージョン 1809 以降から使用可能です。

連鎖 .msi パッケージの設定

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

Windows Installer 4.5 は、トランザクション処理を使った複数パッケージのインストールをサポートします。パッケージを連鎖させることによって、単一のトランザクションとして処理します。トランザクションに含まれる 1 つまたは複数のパッケージが正しくインストールされなかった場合、またはエンド ユーザーがインストールをキャンセルした場合、Windows Installer は全てのパッケージについてロールバックを開始して、システムを以前の状態に復元します。

[リリース] ビューにある [連鎖 .msi パッケージ] 領域では、インストールに連鎖するパッケージとして含める .msi パッケージを識別します。また、この領域で、連鎖するパッケージに渡すプロパティなどの設定も構成します。

メイン インストールの以前のバージョンと共に、またはターゲット マシンに既存していて、ターゲット マシン上に連鎖 .msi パッケージの古いバージョンが既にインストール済みであった場合、インストール中に連鎖 MSI パッケージをマイナー バージョンにアップグレードすることができます。

Windows Installer の連鎖機能では、**MsiInstallProduct** を通してインストールできるパッケージはすべてサポートされていますが、InstallScript MSI パッケージの連鎖はサポートされていません。

Windows Installer 4.0 以前のバージョンは、連鎖 .msi パッケージをインストールしません。

ヒント: *Windows Installer 4.5 再配布可能ファイルの追加方法については、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」をご覧ください。*

次のテーブルは、連鎖 .msi パッケージの設定について説明します。

テーブル 224 • 連鎖 .msi パッケージの設定

設定	説明
インストール (実行時のパス)	Windows Installer パッケージ (.msi ファイル) の名前を指定するか、[参照] ボタンをクリックして、パッケージを参照します。 [参照] ボタンをクリックした場合、製品コードが自動的に挿入されます。また InstallShield は、.msi パッケージ (および、.msi パッケージが圧縮されていない場合は非圧縮ファイル) を製品のメイン .msi パッケージにストリームするかどうかを問い合わせるプロンプトを表示します。 • ファイルのストリームを指定した場合、InstallShield は .msi パッケージの名前を **"インストール (実行時のパス)"** 設定に追加します。また、自動的にファイルの名前 (圧縮された .msi パッケージの場合) またはエントリ *.*(非圧縮の .msi パッケージの場合) が **[ストリームされたファイル]** ボックスに追加されます。*.* エントリの場合、.msi パッケージと同じフォルダーにあるすべてのファイル、および、そのサブフォルダーとその中のファイルが .msi パッケージにストリームされます。 この設定に使用する値は、一時抽出パスに相対しています。つまり、ストリームされたサポート ファイルを **Support¥File.ext** として追加し、それを連鎖パッケージの中で **[SourceDir]Support¥File.ext** として参照できます。 • ファイルのストリームを行わないことを指定した場合、InstallShield は次のパスを **"インストール (実行時のパス)"** 設定に追加します: [SourceDir]FileName.msi このパスは、必要に応じて変更が可能です。たとえば次のようなパスを使用し、また **[インストール後にストリームされたファイルを削除する]** チェック ボックスをクリアしたい場合があります: [LocalAppDataFolder]{ProductCodeGUID}¥FileName.msi この例では、.msi パッケージはローカル システムにキャッシュされ、それはメンテナンスで使用することができます。 ***メモ:*** *基本の MSI プロジェクトでマイナー アップグレードが可能です。* ***ヒント:*** *Windows Installer では、.msi パッケージのファイル サイズに制限があるため、多数のファイルやサイズの大きいファイルを .msi パッケージにストリームしなくない場合があります。*

テーブル 224 • 連鎖 .msi パッケージの設定 (続き.)

設定	説明
製品コード	この設定は、メインのインストールに連鎖される .msi パッケージの製品コードが示します。製品コードは、連鎖パッケージをアンインストールするために必要に応じて使用されます。 **"インストール (実行時のパス)"** 設定の [参照] ボタンを使用して、.msi パッケージを選択した場合、この設定の製品コードは自働的に挿入されます。
UI レベル	連鎖 .msi パッケージに使用するユーザー インターフェイス (UI) レベルを指定します。有効なオプションは次のとおりです: • **基本の UI (/qb)** – 小さいビルトイン進行状況ダイアログを表示します。 • **完全 UI (/qf)** – .msi パッケージで提供されているモーダルとモードレス ダイアログを表示します。 • **UI なし (/qn)** – インストールをサイレントで実行します。 • **簡易 UI (/qb)** – フルサイズの進行状況ダイアログなど、モードレス ダイアログのみ表示します。
インストール条件	連鎖 .msi パッケージに使用するインストール条件を指定します。実行時に条件が True と評価されたとき、連鎖パッケージの製品が既にインストールされていない場合、連鎖 .msi パッケージが Windows Installer によって起動されます。 デフォルトの条件は次の通りです: **インストールされていません** 必要に応じて、これを変更します。たとえば、連鎖 .msi パッケージをマイナー アップグレードに追加し、初回のインストールおよびアップグレードで連鎖 .msi パッケージを起動する場合、次のような条件を追加することを検討してください: **Not Installed OR REINSTALL><"機能名"** この例では、**FeatureName** は、メインのインストールがアップグレード モードで実行されたときに更新される機能の名前です。これは、修復時にも更新されます。 ***メモ:*** *実行時に、インストールの条件が評価されます。条件が False に評価された場合、インストールは、削除の条件を評価します。(メイン製品のアンインストール時にも) インストール条件が常に True に評価された場合、削除条件は一切評価されず、連鎖 .msi パッケージもアンインストールされません。したがって、常に True と評価されるインストール条件の指定を避けた方が良い場合があります。* 条件構文に関する情報は、Windows Installer ヘルプ ライブラリの「Conditional Statement Syntax」をご覧ください。

テーブル 224 • 連鎖 .msi パッケージの設定 (続き.)

設定	説明
インストールのプロパティ	インストール時に、1 つまたは複数のプロパティをメインのインストールからこの連鎖されている .msi パッケージに渡す場合、そのプロパティと適切な値を入力します。 たとえば、連鎖するパッケージの特定の機能をメインのインストールの INSTALLDIR 場所にあるサブフォルダーにインストールする場合、この設定で次の文字列を入力することができます: **INSTALLDIR="[INSTALLDIR]Subfolder¥" ADDLOCAL=Feature1** ***メモ:*** *指定するプロパティ値を非表示または詳細なログ ファイルに表示されないようにするには、指定するプロパティを* ***[動作とロジック]*** *の* ***[プロパティ マネージャー]*** *ビューを使って、MsiHiddenProperties プロパティの値として設定します。*
削除条件	連鎖 .msi パッケージに使用するアンインストール条件を指定します。インストール条件が False に評価され、削除条件が True に評価された場合、連鎖されたパッケージの製品が (存在する場合) Windows Installer によって削除されます。 デフォルトの条件は次の通りです: **REMOVE="ALL"** 条件構文に関する情報は、Windows Installer ヘルプ ライブラリの「Conditional Statement Syntax」をご覧ください。
削除のプロパティ	アンインストール時に、1 つまたは複数のプロパティをメインのインストールからこの連鎖 .msi パッケージに渡す場合、そのプロパティと適切な値を入力します。 通常、この設定は空白のままです。
リリース フラグ	特定のビルドにのみ含める連鎖 .msi パッケージに、1 つまたは複数のリリース フラグを割り当てることができます。たとえば、連鎖 .msi パッケージを必要とする特別なアドオンを含んだ製品の特別なエディションにのみ含める連鎖 .msi パッケージがある場合、その連鎖パッケージにフラグを付けることで、必要な場合にのみそれがビルドに含まれるようにします。 連鎖 .msi パッケージに リリース フラグを割り当てるには、それをこの設定で入力します。入力した文字列には、文字または数字を混在させることができます。1 つの連鎖パッケージに複数のフラグを付けるには、コンマを使ってフラグを区切ります。

テーブル 224 • 連鎖 .msi パッケージの設定 (続き.)

設定	説明
ストリームされたファイル	このボックスでは、メインのインストールのパッケージにストリームされる連鎖 .msi パッケージのファイルが示されます。 [ファイルの追加] と [フォルダーの追加] ボタンを使って、複数のファイルやフォルダー全体をストリームされたファイルの一覧に追加することができます。 代わりの抽出場所を指定するには、エントリを選択し、再度クリックして、エントリの **"実行時のパス"** の値を編集します。エントリの文字列が強調表示され、編集が可能になります。デフォルトで、ファイルは一時フォルダーに抽出されます。 このボックスのエントリを削除する場合、それを選択して、[削除] ボタンをクリックします。
インストールの後でストリームされたファイルを削除する	インストールの後でストリームされたファイルを削除する場合、このチェック ボックスを選択します。ストリームされたファイルをターゲット システムに残す場合、このチェック ボックスをクリアします。 一般的に、抽出したパッケージをローカルにキャッシュするために、ファイルを一時的ではない場所 (例、**[LocalAppDataFolder][PackageCode]¥*.***) に抽出した場合のみ、ファイルをシステムに残すようにすることをお勧めします。 1 つ以上のストリームされたファイルを含む連鎖 .msi パッケージに対して、このチェック ボックスはデフォルトで選択されています。ストリームされたファイルを含まない連鎖 .msi パッケージに対して、このチェック ボックスは無効になっています。

[パッチのデザイン] ビュー

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[アップグレード] ビューでアップグレードアイテムを追加すると、パッチのデザイン ビューを利用して、パッチとして配布することができるようにそのアップグレードのためのパッチ構成を作成することができます。パッチは、製品の以前のバージョンを持っているターゲット システムに適用できます。

[パッチのデザイン] ビューでは複数パッチ構成を作成することも可能です。各パッチ構成にはパッチをビルドするために必要な設定とデータが含まれます。各パッチ構成には少なくとも、最新セットアップが 1 つおよび以前のセットアップが 1 つ含まれている必要があります。最小 3 つの設定のみでパッチを作成することが可能です。

パッチの構成

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチを作成するには、まず [パッチのデザイン] ビューで [パッチのデザイン] エクスプローラーを右クリックしてから、[新規のパッチの構成を追加] をクリックします。パッチの構成は、少なくとも、最新セットアップ 1 つと 1 つまたは複数の以前のセットアップからなります。最新および以前のセットアップを指定する前に、次のタブでパッチの構成のパッチのプロパティを全般的に指定することができます。

- 共通
- 識別
- デジタル署名
- シーケンス
- 詳細

[共通] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成をクリックすると、複数のタブが表示されます。パッチの構成の [共通] タブは、よく利用されるパッチの構成を表示します。

パッチ出力の場所

パッチ ファイルをビルドする場所を指定、または既存フォルダーを検索します。InstallShield は、出力先に次のサブディレクトリを追加します。

¥*PatchConfigName*¥Patch

メモ: *InstallShield は出力ディレクトリに Interim (中間) と呼ばれるフォルダーも追加します。Interim フォルダーはパッチをビルドしたあとに次のファイルを含みます。*

- **.log – パッチ作成プロセスからの出力が含まれています。通常トラブルシューティングに役立つ情報が含まれています。*
- **.pcp – パッチ作成のプロパティ ファイル。パッチパッケージを生成するために必要なすべての設定が保存されています。*

ランチャの設定

ここでは、次のようなランチャー設定を構成することができます。

テーブル 225・構成可能なランチャー設定

設定	説明
Update.exe の作成	現在のパッチに Update.exe アップデート ランチャを作成する場合、このチェック ボックスを選択します。 Update.exe アップデート ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。
Windows Installer 3.1 エンジンを含む	このチェック ボックスを選択して Windows Installer 3.1 エンジン をパッチパッケージに含めます。
.NET Framework を含める	このチェック ボックスを選択して、.NET Framework をパッチパッケージに含めます。

パッチ作成キャッシュ

ここでは、次の設定を行うことができます。

テーブル 226・パッチ作成キャッシュの設定

設定	説明
Enable	このチェック ボックスを選択した場合、パフォーマンス（速度）を向上させるために InstallShield は後に続くビルドで使用する中間ファイルを作成します。これらの中間ファイルは無期限にシステム上に残ります。ディスク容量が限られている場合はこのチェック ボックスを選択しないで下さい。

大きなファイルのパッチの最適化

このチェック ボックスを選択して、インストール プロジェクトで 4 MB よりも大きい全アプリケーションファイル用に小さな bit レベルのパッチを作成します。

パッチのアンインストールを許可する（Windows Installer 3.0 が必要）

アプリケーション全体と他のパッチをアンインストールして再度インストールし直す手間を掛けずにパッチをアンインストールできるようにしたいとき、このチェック ボックスを選択します。パッチのアンインストールは一定の条件下のみで作動しますのでご注意ください。たとえば、バージョン 3.0 以前の Windows Installer はアプリケーションからパッチのみを削除することができません。さらに詳しい情報は、Windows Installerヘルプの「パッチの削除」を参照してください。

プロジェクト: *InstallScript MSI プロジェクトで、InstallScript ユーザー インターフェイス（UI）のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合、アンインストール可能なパッチ機能は使用できません。*

この関数は、InstallScript UI スタイルが (埋め込み UI ハンドラーとして InstallScript エンジンを使用する) 新しいスタイルである InstallScript MSI プロジェクトに適用します。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

パッチのビルド

適切な設定をすべて構成したあと、このボタンをクリックしてパッチをビルドします。

メモ: *[パッチのデザイン] エクスプローラーからパッチをビルドすることもできます。適切なパッチ構成を右クリックして、パッチのビルド を選択します。*

[識別] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成をクリックすると、複数のタブが表示されます。[識別] タブには、表示文字列の設定があります。表示文字列は、[プログラムの追加と削除] で表示されるパッチに関する情報に使用されます。メタデータは、ターゲットマシンの適用されたパッチを検索しカタログする Windows Installer 3.0 以降の API によっても使用されます。

ヒント: *[パッチのデザイン] ビューでパッチ構成の最新セットアップを変更するたびに、その最新セットアップからの [プログラムの追加と削除] 情報が [識別] タブにある設定用の値として使用されます。必要に応じて、[識別] タブの値をオーバーライドすることもできます。*

テーブル 227 ・ [識別] タブの設定

設定	説明
表示名	パッチの名前を指定します。
サポート URL	エンドユーザーにテクニカルサポートを提供する URL を指定します。
説明	パッチの簡単な説明を指定します。
製造元の名前	アプリケーションの製品メーカーの名前を指定します。
ターゲット製品名	アプリケーションまたはターゲット アプリケーション スイートの名前を指定します。
分類	アップグレードのカテゴリーを指定します。たとえば、クリティカルアップデート、ホットフィックス、および、サービスパック等。

[デジタル署名] タブ

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成をクリックすると、複数のタブが表示されます。[デジタル署名] タブで、パッチにデジタル署名をするときの設定を指定することができます。

メモ- パッチのファイル (アプリケーションの実行可能ファイルなど) にデジタル署名を行う場合、[リリース] ビューの [署名] タブで署名するファイルを指定することができます。

テーブル 228 - [デジタル署名] タブに設定

設定	説明
パッチパッケージに署名する	パッチ パッケージにデジタル署名を行う場合、このチェック ボックスを選択します。
アップデート ランチャーに署名する	Update.exe パッケージにデジタル署名を行う場合、このチェック ボックスを選択します。
64 ビット署名を使用	64 ビット署名フレームワークのみを使ってパッチ パッケージにデジタル署名を行う場合、このチェックボックスを選択します。
署名の種類	パッチパッケージと Update.exe ファイルにデジタル署名を付ける方法を指定します。署名ツール構成の引数を指定します: - **標準** ー InstallShield デフォルト署名ツールを使って、パッチ パッケージおよび Update.exe ファイルにデジタル署名を行います。 - **カスタム** ー 独自のカスタマイズ署名ツールを使って、パッチ パッケージおよび Update.exe ファイルに署名を行います。このオプションを設定すると、**"パス"** および **"引数"** 設定が有効になります。 *メモ- デフォルトで、この設定は **[標準]** になっています。*
証明書 URL	完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。

テーブル 228・[デジタル署名] タブに設定 (続き.)

設定	説明
デジタル証明書情報	リリースに署名を行うために使用するデジタル証明書を指定するには、この設定の横にある [参照] ボタンをクリックします。[証明書の選択] ダイアログ ボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。 InstallShield では、パブリック キー証明書 (.cer) ファイルを使って EV トークン パスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。.cer ファイルは通常、USB eToken プロバイダー (たとえば、SafeNet 認証クライアント) と関連付けられた EV 認証クライアント ツールからパブリック キーをエクスポートして作成されます。.cer ファイルが指定された場合、InstallShield は Extended Validation 証明書 (EV) プロパティを構成するための追加オプションを表示します。詳細については、次を参照してください。 ・ InstallShield における拡張検証 (EV) 証明書情報の構成 ・ [証明書の選択] ダイアログ ボックス .pfx ファイルを指定するか、テスト ストアから証明書を選択すると、次が表示されます: ・ **証明書の拇印** – この読み取り専用設定は、証明書の拇印を表示します。 ・ **発行元** – この読み取り専用設定は、証明書の発行元情報を表示します。 ・ **有効期限日** – この読み取り専用設定は、証明書の有効期限日を表示します。 証明書ファイル (EV がエクスポートした .cer ファイル) を指定すると、次の設定が表示されます: ・ **暗号化サービス プロバイダー** – この設定を使って、暗号化サービス プロバイダー (CSP) を指定します。 ・ **コンテナ名** – この設定を使って、暗号化サービス プロバイダー (CSP) に関連付けられた秘密キーのコンテナ名を指定します。 ・ **トークン パスワード** – この設定を使って、暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。 ***メモ:*** *証明書ファイル (EV がエクスポートした .cer ファイル) を指定した後に表示される設定を使用する際は、以下の情報を考慮してください:* ・ *暗号化サービス プロバイダー (CSP) 名とコンテナ名の両方は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。* ・ *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* ・ *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*

テーブル 228 ・[デジタル署名] タブに設定 (続き.)

設定	説明
パスワード	使用する .pfx にパスワードがある場合、それを入力します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。
パス	パッチ パッケージおよび Update.exe ファイルにデジタル署名を行うために使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。
引数	署名ツール構成のコマンドライン引数を指定して、パッチ パッケージおよび Update.exe ファイルにデジタル署名を行います。たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。 sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename] [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクト内で定義されている任意のパス変数または環境変数を使用することができます。
署名の説明	(必要な場合) パッチ パッケージおよび Update.exe ファイルに使用する署名の説明を指定します。ここで指定した説明は、UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示されます。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。 この設定を空白のままに残すと、InstallShield は UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示される説明として、ファイル名を拡張子なしで使用します。

[シーケンス] タブ

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成をクリックすると、複数のタブが表示されます。[シーケンス] タブで、パッチがターゲットマシンに提供された順番に関係なく、どの順番で Windows Installer バージョン 3.0 以上がインストールされた製品にパッチを適用するかを指定します。バージョン 3.0 以前の Windows Installer の場合、パッチ シーケンスは無視され、すべてのパッチは、ターゲットマシンに提供された順番で製品に適用されます。

デフォルト パッチ シーケンスを使用する

製品にパッチにデフォルトのシーケンスを使用したいとき、このチェック ボックスを選択します。このデフォルトのシーケンスを参照して、もう使われていないパッチ、入れ替わったパッチ、または製品に既に適用済みのパッチを見ることができます。

このチェック ボックスをクリアすると、カスタム シークエンスを追加することができます。このチェック ボックスをクリアにしたままで、シーケンスを追加をしないと、シーケンス情報はパッチに組み込まれません。

[詳細] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューでパッチの構成をクリックすると、複数のタブが表示されます。パッチの構成の詳細タブには、パッチの構成のためのビルド設定がすべて表示されます。

パッチの設定

ここでは、次のようなパッチのための設定を構成することができます。

テーブル 229 ・ パッチ構成の設定

プロパティ	説明
パッチ出力の場所	パッチ ファイルをビルドする場所を指定、または既存フォルダーを検索します。InstallShield は、出力先に次のサブディレクトリを追加します。 ¥*PatchConfigName*¥Patch. ***メモ:*** *InstallShield は出力ディレクトリに Interim (中間) と呼ばれるフォルダーも追加します。Interim フォルダーはパッチをビルドしたあとに次のファイルを含みます。* - **.log – パッチ作成プロセスからの出力が含まれています。通常トラブルシューティングに役立つ情報が含まれています。* - **.pcp – パッチ作成のプロパティ ファイル。パッチパッケージを生成するために必要なすべての設定が保存されています。*
パス GUID 生成	パッチのビルド時、毎回、新しい GUID を生成するかどうかを指定します。[いいえ] を選択すると、"パッチ GUID" プロパティにリストされた GUID が使用されます。デフォルトの設定は [はい] です。

テーブル 229・パッチ構成の設定（続き）

プロパティ	説明
パッチ GUID	パッチ GUID はパッチ パッケージを一意に識別するのに使用されます。″パッチ GUID の生成″ プロパティに [はい] が設定されていると、InstallShield は、パッチがビルドされるごとに、このプロパティに新しい GUID を自動生成します。
最小 Windows Installer バージョン	パッチに必要な Windows Installer の最小バージョンを指定します。たとえば、Windows Installer 2.0 を最小バージョンとする場合、**200** と入力します。Windows Installer 3.0 を最小バージョンとする場合、**300** と入力します。Windows Installer 3.1 を最小バージョンとする場合、**301** と入力します。 ***メモ・** Windows Installer の以前のバージョンを指定すると、Windows Installer エンジンの最新バージョンでのみサポートされている機能を利用できなくなります。*

ビルドの設定

ここでは、次のようなパッチのためのビルド設定を構成することができます。

テーブル 230・パッチ用のビルドの設定

プロパティ	説明
ファイル全体として含める	パッチ パッケージに含まれている各ファイルに対してファイル全体を含めるかどうかを指定する。 ***メモ・** 最新セットアップ の設定で、ファイルとごとにこの設定をオーバーライドすることができます。*
ビルド時に検証	パッチのビルド時、毎回、アップグレードおよびパッチの検証を利用するかどうかを指定します。
パッチ コンポーネントを非圧縮のままにする	すべての .msp ファイル情報を一時フォルダーに残しておく場合、[はい] を選択します。トラブルシューティングをするとき、この情報はとても役に立ちます。この情報を保存しない場合は、[いいえ] を選択します。
パッチ作成キャッシュを有効にする	この [はい] を選択すると、パフォーマンス速度を向上させるために、InstallShield は後に続くビルドで使用する中間ファイルを作成します。これらの中間ファイルは無期限にシステム上に残ります。ディスク容量が限られている場合は、[いいえ] を選択してください。パッチ キャッシュ フォルダーで、中間ファイルが作成される場所を指定します。
パッチ キャッシュ フォルダー	パッチ キャッシングからのテンポラリ ファイルが保存されるディレクトリを指定します。これらの中間ファイルは無期限にシステム上に残ります。

テーブル 230・パッチ用のビルドの設定 (続き.)

プロパティ	説明
MsiPatchOldAssembly テーブルの生成	MsiPatchOldAssemblyFile および MsiPatchOldAssemblyName テーブルのエントリを自動的に生成するかどうかを指定します。これらのエントリを利用すると Windows Installer 3.0 以降で実行されるパッチ パッケージが元のインストール ソースを実行時に要求することなく、GAC (グローバル アセンブリ キャッシュ) のアセンブリをパッチすることができます。詳細については、「グローバル アセンブリ キャッシュのアセンブリをパッチする」を参照してください。

実行時の設定

ここでは、次のようなパッチのための実行時の設定を構成することができます。

テーブル 231・パッチの実行時設定

プロパティ	説明
異なる製品コードを許可する	製品コードが元と最新のインストールで異なる場合、パッチ作成実行可能ファイルにビルド時のプロンプトを抑制させるかどうかを指定します。メジャー アップグレードでは、必ず選択してください。
異なる製品バージョンを許可する	製品バージョンがオリジナルと最新のインストールで異なる場合、パッチ作成実行可能ファイルにビルドタイムのプロンプトを抑制させるかどうかを指定します。
置換するパッチ GUID のリスト	現在のパッチで、1 つまたは複数の以前のインストール済みのパッチを置換するには、このプロパティにそれらのパッチのパッチ GUID をカンマで区切って設定します。例: **{C86838C9-DEDC-4451-B96F-94AFB9460F15},{C8633E5B-AC44-45d8-B487-C68B3B1F60D6}** [パッチのデザイン] ビューにパッチの構成が複数あっても、このプロパティの設定は通常、*必要ありません*。ただし パッチが以前の パッチに追加されたファイルを上書きしない場合、このプロパティの設定が必要な場合もあります。 ***メモ:*** *このプロパティには、パッチのインストールの部分のみに影響します。ファイルをオリジナルのバージョンに戻すことはしません。*
ターゲット製品コードのリスト	このパッチでターゲットする製品コードを、コンマで区切って、指定します。この設定にアスタリスク (*) を入力すると、InstallShield がビルド時に、パッチの構成で以前のイメージとしてリストされたインストールの製品コードで置き換えます。

テーブル 231・パッチの実行時設定 (続き.)

プロパティ	説明
Update.exe の作成	現在のパッチに Update.exe アップデート ランチャを作成するかどうかを指定します。この設定で **[はい]** を選択すると、Update.exe アップデート ランチャーを作成することができます。 Update.exe アップデート ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。
アップデート ランチャー名	InstallShield がビルド時に生成するアップデート ランチャー ファイルに使用するファイル名を指定します(.exe ファイル拡張子は除きます)。 この設定が空白の場合、InstallShield は [アップデート] のデフォルト値を使用し、アップデート ランチャー ファイルは Update.exe と呼ばれます。 ***メモ・"ランチャー名の更新" 設定を有効にするには、"Update.exe の作成" 設定が [はい] に設定されていることを確認します。***
64 ビット	現在のパッチに Update.exe アップデート ランチャーを作成するかどうかを指定します。次のオプションを選択できます: ・ **はい**—これを選択すると、64 ビットバージョンの update.exe アップデート ランチャーが作成されます。 ・ **いいえ**—これを選択すると、32 ビットバージョンのupdate.exe アップデートランチャーが作成されます。 デフォルトで、この設定は **[いいえ]** になっています。 ***メモ・"ランチャー名の更新" 設定を有効にするには、"Update.exe の作成" 設定が [はい] に設定されていることを確認します。***
MSI コマンドライン引数	Setup.ini に書き込む Windows Installer の引数を指定します。 [Startup] CmdLine=値 このプロパティのデフォルト値は、次のとおりです。 REINSTALLMODE=omus REINSTALL=ALL REINSTALL プロパティは、実行される再インストールのタイプを指定する文字を含む文字列です。詳細については、Windows Installer ライブラリの REINSTALLMODE Property」を参照してください。
ランチャをパスワードで保護	パッチをパスワードで保護するには、[はい] を選択してから "ランチャ パスワード" 設定でパスワードを入力します。パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。 この設定は Update.exe ファイルを使用するパッチのみに適用します。

テーブル 231・パッチの実行時設定 (続き.)

プロパティ	説明
ランチャ パスワード	アプリケーションを保護するパスワードを入力します。パスワード保護をアクティブにするには、"ランチャをパスワードで保護" 設定で [はい] を選択する必要があります。 パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。 この設定は Update.exe ファイルを使用するパッチのみに適用します。
RTM バージョンをターゲットにするマイナー アップデート (MSI 3.1が必要)	現在のマイナーアップグレード パッチを適用する前に、マイナー アップグレード パッチで製品の RTM (Release to Manufacturing) バージョン (または、インストールされている場合、製品の最新メジャー アップグレード) までのすべてのパッチを実質的に削除する場合、このプロパティで [はい] を選択します。このオプションを選択すると、すべてのパッチが、シーケンシングデータがあるないにかかわらず、削除されます。追加のベースライン バージョンをターゲットにする必要はないので、結果的にパッチのペイロードを増やすことになります。すべてのエンドユーザーは、中間のパッチを適用することなく、正常にこのパッチを適用することができます。 このオプションを選択すると、場合により、RTM (または、存在する場合、製品の最新メジャー アップグレード) の後に作成された以前の各マイナー アップグレードをターゲットするのに必要な情報をパッチに含める必要があります。 たとえば、SP 2 のマイナー アップグレード パッチを作成していて、このプロパティで [いいえ] を選択した場合、パッチは SP 1 のマイナー アップグレードをターゲットにする必要があります。またオプションで、他のベースライン (RTM など) をターゲットにすることもできます。これにより、ペイロードが増加します。RTM バージョンをターゲットにしない場合、RTM バージョンを持っているが SP 1 のマイナー アップパッチがないエンドユーザーは、SP 2 の前に SP 1 をインストールしなければなりません。 Windows Installer の 3.1 より以前の バージョンはこの設定を無視します。

テーブル 231・パッチの実行時設定 (続き.)

プロパティ	説明
OptimizedInstallMode プロパティ (MSI 3.1 が必要)	Windows Installer 3.1 を利用してパッチ処理を最適化し、現在のパッチを適用する時間を短縮する場合は [はい] を選択します。選択されたテーブルのリストのみをパッチする場合、Windows Installer 3.1 はその他のテーブルに関連付けられたアクションを無視します。テーブル一覧を含む詳細は、Windows Installer ヘルプライブラリの「Patch Optimization」を参照してください。 このプロパティで [はい] を選択した場合、予定通りにパッチが動作することを慎重にテストすることがマイクロソフトによって推奨されています。 現在のパッチに対して Windows Installer 3.1 によるパッチ処理の最適化を行なわない場合、[いいえ] を選択します。パッチがターゲット マシンに適用されたとき、Windows Installer はアプリケーションの完全修復を実行します。 Windows Installer の 3.1 より以前のバージョンはこのプロパティを無視します。パッチが選択したテーブルのリストのみを変更する場合、Windows Installer 3.0 はパッチを最適化します。Windows Installer 3.0 によるパッチの最適化を避けるためには、DisableFlyWeightPatching ポリシーを利用しなくてはなりません。さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「DisableFlyWeightPatching」を参照してください。 Windows Installer 3.0 より前のバージョンは、パッチが適用されたときにインストールの完全修復を実行します。

Windows Installer エンジン

ここでは、Windows Installer エンジンのための次のような設定を構成することができます。

テーブル 232・パッチ用の Windows Installer の設定

プロパティ	説明
MSI 3.1 エンジンを含める	Windows Installer エンジンをパッチパッケージに含めるかどうか指定します。
エンジンの場所	次のオプションから選択してください。 ・ **Web からエンジンをダウンロードする** – Windows Installer エンジンのインストールが必要な場合、実行時にダウンロードされます。 ・ **Update.exe からエンジンを抽出する** – ビルドは Windows Installer エンジンを Update.exe ファイルにストリームし、顧客に配布する単一ファイルを生成します。エンジンは、実行時に Update.exe ファイルから抽出され必要に応じてインストールされます。 ・ **ソース メディアからコピーする** – ビルドは、Windows Installer エンジンを Update.exe ファイルと同じディレクトリにコピーします。

テーブル 232 • パッチ用の Windows Installer の設定 (続き.)

プロパティ	説明
Windows Installer 3.1 エンジン URL	エンジン場所の URL を指定します。Update.exe ファイルは、この場所を、実行時に、エンジンをダウンロードする場所として使います。便宜上、デフォルトの URL の場所は Revenera が運営するライブ サイトとなっています。 ***メモ:*** *3.1 エンジンは、デフォルトの URL (http://www.installengine.com/Msiengine30 からダウンロードすることができます。*

Microsoft .NET Framework

ここでは、Microsoft .NET Framework エンジンのための次のような設定を構成することができます。

テーブル 233 • パッチ用の Microsoft .NET Framework の設定

プロパティ	説明
ビルドに含める	パッチに Microsoft .NET Framework を含めるかどうかを指定します。[はい] を選択すると、最新リリースに含まれている .NET Framework と同じバージョンが、この新しいパッチにも含まれます。
エンジンの場所	次のオプションから選択してください。 • **Web からエンジンをダウンロードする** – .NET Framework エンジンのインストールが必要な場合、実行時にダウンロードされます。 • **Update.exe からエンジンを抽出する** – ビルドは.NET Framework を Update.exe ファイルにストリームし、顧客に配布する単一のファイルを生成します。エンジンは、実行時に Update.exe ファイルから抽出され必要に応じてインストールされます。 • **ソース メディアからコピーする** – ビルドは、.NET Framework を Update.exe ファイルと同じディレクトリにコピーします。
エンジン URL	.NET Framework の場所の URL を指定します。Update.exe ファイルは、この場所を、実行時に、.NET Framework をダウンロードする場所として使います。便宜上、デフォルトの URL の場所は Revenera が運営するライブ サイトとなっています。

アップデート起動ツールの設定

この領域では、Update.exe 起動ツールにおける次の設定を構成することができます:

テーブル 234 • パッチ用のアップデート起動ツールの設定

プロパティ	説明
会社名	Update.exe のデフォルトの会社名を独自の会社名でオーバーライドするには、ここに会社名を入力します。 会社名が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
製品名	Update.exe のデフォルトの製品名を独自の製品名でオーバーライドするには、ここに製品名を入力します。 製品名が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
製品バージョン	Update.exe のデフォルトの製品バージョンを独自の製品バージョンでオーバーライドするには、ここに製品バージョンを入力します。 製品バージョンが、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
説明	Update.exe のデフォルトの説明を独自の説明でオーバーライドするには、適切な説明を入力します。 説明が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックしたときに表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 234・パッチ用のアップデート起動ツールの設定 (続き.)

プロパティ	説明
著作権情報	Update.exe のデフォルトの著作権情報を製品の著作権情報でオーバーライドするには、製品の著作権情報を入力します。 著作権情報が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
必要実行レベル	"必要実行レベル" 設定を使って、Windows Vista 以降のプラットフォーム上でプロジェクトの Update.exe ファイルを実行するのに必要な最低実行レベルを指定します。選択可能なオプションは、以下のとおりです。 ・ **管理者** — Update.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。非管理者は、管理者としての認証が必要になります。 ・ **最高権限** — Update.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を承認する必要があります。非管理者は、管理者権限を持たずに Setup.exe を実行します。 ・ **起動者** — Update.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Update.exe を実行することができます。Update.exe は、資格情報または同意 (コンセント) を求める UAC メッセージを表示しません。 ・ **前のセットアップ マニフェストを使用** — Update.exe マニフェストは前のセットアップで指定された必要実行レベルと同じレベルを使用します。デフォルトでは、これが設定されています。 InstallScript MSI プロジェクトと基本の MSI プロジェクトの場合、[Update.exe の作成] チェック ボックスを選択すると、InstallShield が Update.exe 起動ツールにアプリケーション マニフェストを埋め込みます。このマニフェストは選択された実行レベルを指定します。Windows Vista よりも古いバージョンのオペレーティング システムでは、必要実行レベルは適用されません。 基本の MSI プロジェクトで [Update.exe を作成する] チェック ボックスがクリアされている場合、InstallShield は Update.exe 起動ツールに Windows アプリケーション マニフェストを埋め込みません。 詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。

テーブル 234・パッチ用のアップデート起動ツールの設定 (続き.)

プロパティ	説明
アイコン	Update.exe ファイルに独自のアイコンを使用する場合、そのアイコンを含むファイルの完全修飾名を指定します。ファイルを指定するには、絶対パスまたはパス変数の相対パスを入力するか、省略記号ボタン (...) をクリックして [アイコンの変更] ダイアログ ボックスでファイルを参照します。 デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、[アイコンの変更] ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。 このフィールドを空白のままにした場合、Update.exe ファイルにデフォルトのアイコンが使用されます。

最新セットアップ

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチ構成の [最新セットアップ] 項目で、インストールの最新バージョンの設定を指定します。

最新のセットアップ パス

最新セットアップは、パッチを作成するインストールの最新バージョンでなければなりません。デフォルト設定 <ISLatestRelease> は、ビルドする最新の完全リリースに解決します。

このボックスで指定したリリースは、非圧縮状態であることを確認してください。前回の製品バージョンが圧縮の場合、できるかぎり管理インストールで作成します。

ファイル全体として含める

ビルドにファイルのビットレベルの差分を実行させるかわりに、パッチに完全な形で含めるファイルを選択します。

***メモ・** すべてのファイルを完全な形で含めるには、パッチ構成の ファイルを完全な形で含める プロパティを [はい] に設定します。*

以前のセットアップ

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチを作成するとき、[パッチのデザイン] ビューの各パッチ構成には少なくとも、最新セットアップが 1 つおよび以前のセットアップが 1 つ含まれている必要があります。パッチが製品の以前のバージョンを複数ターゲットするとき、各ターゲットイメージにそれ自身の以前のセットアップ項目があることを確認してください。

たとえば、会社がバージョン 1.0 および 1.1 のアプリケーションをサポートしていて、両方のアプリケーションを 2.0 にアップグレードする必要があるという状況を想定します。両方のバージョンをアップデートするシングルパッチを作成するには、最新のセットアップに以前のセットアップとして V1.0 と V1.1 を追加します。最新セットアップには V2.0 が入ります。

[パッチのデザイン] ビューで以前のセットアップアイテムをクリックすると、次のようなタブが利用できます。

- 共通
- 詳細

[共通] タブ

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューで以前のセットアップ項目をクリックすると、いくつかの異なるタブが表示されます。[共通] タブには、基本的な以前のセットアップ構成の設定が表示されます。

テーブル 235・ 以前のセットアップ構成の共通設定

オプション	説明
前のセットアップ	最新セットアップへアップデートにパッチの適用が必要なインストール (.msi ファイルまたは Setup.exe ファイル) を指定します。パッチ作成のプロセスには非圧縮状態のリリースが必要なため、以前のバージョンを圧縮解除する場所を指定するよう求めるプロンプトが必要に応じて表示されます。 ***注意・** パスを指定する際に以下の点に注意してください。ローカルまたはネットワークドライブから複数ディスクにわたるセットアップを圧縮解除することはできません。別の方法でセットアップの圧縮解除を行う必要があります。*

テーブル 235 • 以前のセットアップ構成の共通設定 (続き.)

オプション	説明
不足しているファイルを無視する	2つの個別のバージョンに含まれるファイルがより新しいバージョンで検出されない場合、新しいバージョンで変更されていない限り、ファイルはパッチパッケージから除外されます。ファイルを無視することの利点は、必要な変更されたファイルのみがパッケージに追加される点です。検出されないソースファイルを無視すると選択した場合、セットアップの完全イメージは不要です。

[詳細] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューで以前のセットアップ項目をクリックすると、いくつかの異なるタブが表示されます。詳細 タブには、次のトランスフォーム フィルター設定のほかに [共通] タブでアクセス可能な設定が含まれています。

テーブル 236 • 以前のセットアップ構成の詳細設定

オプション	説明
バージョンの関係	パッチのために設定を指定する製品の最新バージョンと以前のバージョンの間の必要関係を説明する条件を選択します。InstallShield は選択された条件が満たされたときのみ指定された以前のセットアップと最新のセットアップ間のトランスフォームを作成します。

テーブル 236・以前のセットアップ構成の詳細設定 (続き.)

オプション	説明
チェックするバージョン	バージョン関係 で選択されたオプションに従って選択された以前のセットアップと最新のセットアップのトランスフォームを作成すべきかどうかを判断するときに使用されるバージョン番号の箇所を 3 つの部分から選択します (複数可)。選択可能なオプションは以下のとおりです: • **メジャー、マイナー、アップグレードバージョンを確認する** — InstallShield に以前のセットアップのバージョン番号と最新セットアップのバージョン番号を 3 つすべての部分を使用して比較させる場合、このオプションを選択します。 • **メジャー、マイナー バージョンを確認する** — InstallShield に以前のセットアップのバージョン番号と最新セットアップのバージョン番号を最初の 2 つの部分を使用して比較させる場合、このオプションを選択します。バージョン番号の 3 番目の部分は無視されます。 • **メジャー バージョンのみ確認する** — InstallShield に以前のセットアップのバージョン番号と最新セットアップのバージョン番号を最初の部分を使用して比較させる場合、このオプションを選択します。バージョン番号の 2 番目と 3 番目の部分は無視されます。 • **バージョンをチェックしない** — InstallShield に以前のセットアップと最新セットアップのバージョン番号の関係を問わずにトランスフォームを作成させる場合、このオプションを選択します。 たとえば、このプロパティに [メジャー、マイナー、アップグレードバージョンを確認する] を選択して、"バージョン関係" プロパティに "以前のセットアップ バージョン <= 最新のセットアップバージョン" を選択すると、InstallShield は、以前のセットアップ バージョン番号は 1.2.3 で、最新セットアップのバージョン番号が 1.2.4 の場合、以前のセットアップと最新セットアップの間の差分にトランスフォームを作成します。ただし、以前のセットアップにバージョン番号が 1.2.0 で、最新セットアップのバージョン番号が 1.1.0 の場合、InstallShield はトランスフォームを作成しません。バージョン番号が 1.2.3 の場合、1 はメジャーバージョン番号を表し、2 はマイナー バージョン番号を表し、3 はアップグレード バージョン番号を表します。
製品コードの一致	以前のセットアップと最新のセットアップが同じ製品コードを持っているときのみトランスフォームを作成する場合、[はい] を選択します。[いいえ] を選択すると、InstallShield は、製品コードに関係なく、以前と最新のセットアップ間のトランスフォームを作成します。
アップグレード コードの一致	以前のセットアップと最新のセットアップが同じアップグレード コードを持っているときのみトランスフォームを作成する場合、[はい] を選択します。[いいえ] を選択すると、InstallShield は、アップグレード コードに関係なく、以前と最新のセットアップ間のトランスフォームを作成します。
言語の一致	以前のセットアップと最新のセットアップが同じ言語を持っているときのみトランスフォームを作成する場合、[はい] を選択します。[いいえ] を選択すると、InstallShield は、言語に関係なく、以前と最新のセットアップ間のトランスフォームを作成します。

追加の外部ファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

追加外部ファイルをパッチすると、以前のインストールの一部として出荷されなかったアプリケーションファイルのバージョンに対するバイナリファイルパッチを作成することができます。

たとえば、アプリケーション ファイルの 1 つが他のインストールによりアップデートされている可能性がある場合、そのファイル バージョンをリファレンスして、実行時にファイル レベルのパッチが支障なく適用されるようにします。

外部ファイル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

[パッチのデザイン] ビューで [外部ファイル] 領域に追加した外部ファイルをクリックすると、ファイルの構成に必要な 2 つの設定が表示されます。

最初のボックスでは、外部ファイルと関連付けるファイルキーを参照するか、そのファイルの File テーブルキーを入力します。

2 番目のボックスでは、最初のボックスで指定したファイルキーに関連付ける外部ファイルを参照します。

[追加ツール] ビュー

プロジェクト: *[追加ツール] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

その他のツールを使うと、プロジェクトに追加機能を提供できます。このビューのすべてのアイテムがすべてのプロジェクトの種類に適用するわけではありません。アイテムが利用できるプロジェクトの種類を表示するには、次のリンクをクリックします。

依存関係スキャナーいぞんかんけいスキャナー

プロジェクト: *[依存関係スキャナー] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

[依存関係スキャン] ビューには、場合によってプロジェクトに追加する必要がある依存関係を識別できる 3 つの異なるスキャン ツールが提供されています。これらのスキャナーは、Visual Basic プロジェクト スキャナー、既にプロジェクトに含まれているファイルをスキャンするスタティックスキャナー、およびあらゆる依存関係について実行中のアプリケーションをスキャンするダイナミックスキャナーです。

MSI デバッガー

プロジェクト: *[MSI デバッガー] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *MSI データベース*
- *トランスフォーム*

ユーザー インターフェイスシーケンスのアクションやダイアログでブレークポイントを設定し、各ステップの Windows Installer プロパティを監視および変更することにより、セットアップをデバッグします。

ダイレクト エディター

プロジェクト: *ダイレクト エディターは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

ダイレクト エディターでは、プロジェクト ファイル (.ism) またはデータベース ファイル (.msi、.msm、または .mst) に含まれるすべてのテーブルを参照できます。プロジェクトの種類によっては、このビューでは、Windows Installer データベース フォーマットに詳しい上級開発者向けに、高度な機能も提供されています。[ダイレクト エディター] の [参照の追跡] ペインを使って、テーブル レコードの関係性を簡単に確認することもできます。

依存関係スキャナー

プロジェクト: *[依存関係スキャナー] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

あるファイルが、他のファイルの関数に依存してタスクを実行することがよくあります。ただし、インストール プロジェクトにアプリケーションファイルを含める際に、*「依存関係」*と呼ばれる他のファイルに気が付かない場合があります。InstallShieldでは、これらのファイルを見つけて作業を行う次のスキャン ウィザードが用意されています。スキャナーは、依存関係スキャナー ビューでアクセスできます。

テーブル 237 ・ 依存関係スキャナー

スキャンのオプション	説明
スタティック スキャンの実行	スタティック スキャン ウィザードは、プロジェクトのポータブル実行可能ファイル (例、.exe、.ocx、.com、.tlb、.hlp、および .chm) を探し、必要な依存関係を検出します。
ダイナミック スキャンの実行	ダイナミック スキャン ウィザードは、実行可能ファイルが実行中にシステムを監視して、実行可能ファイルで必要となる可能性がある .dll または .ocx ファイルを確認します。

これらのスキャナーのどれか 1 つを使って基本の MSI または InstallScript MSI プロジェクトに追加されるファイルは、セットアップベストプラクティス に従って追加されます。

スタティックおよびダイナミック スキャン ウィザードを使用する場合、InstallShield を使ってスタティックまたはダイナミック スキャンを実行するとき必ず自動的に選択または除外されるファイルを指定することができます。詳細については、「依存関係スキャナーでファイルをフィルターする」を参照してください。

MSI デバッガー

プロジェクト: *[MSI デバッガー] ビューは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *MSI データベース*
- *トランスフォーム*

MSI デバッガーでリリースをデバッグするとき、パッケージの [ユーザー インターフェイス] と [実行] シーケンスの手順を実行しながら Windows Installer のプロパティを表示して設定することができます。

タスク ***次の手順に従って、MSI デバッガーでのセットアップ操作を開始します。***

1. リリースをビルドします。ただし、リリースをデバッグするときには 1 つ重要な制約があります。Setup.exe 内で圧縮されたパッケージをデバッグすることはできません。
2. [**MSI デバッガー**] ビューに移動します。MSI デバッガーには最初に [ユーザー インターフェイス] シーケンスと [実行] シーケンスにおける標準およびカスタム アクションがすべて表示され、次にプロジェクトにおけるダイアログがすべて表示されます。またツールバーの **[デバッグ]** ボタンデバッグをクリックするか、**[ビルド]** メニューの **[デバッグ]** を選択して MSI デバッガーを起動します。
3. アクションまたはダイアログにブレークポイントを設定します。
4. MSI デバッガーを起動します。

ブレークポイントに到達するまで各アクションおよびダイアログが実行され、ブレークポイントに到達した時点で実行が停止します。ここで、ウォッチ ウィンドウと変数 ウィンドウにプロパティを表示し、設定することができます。最後に、残りのアクションを順番に実行するか、または、デバッガーを中止します。

メモ: *MSI デバッガーでは、システム変更の実行をインストールスクリプトの実行時まで遅らせる遅延カスタム アクションをサポートするようになりました。*

MSI デバッガーと InstallScript デバッガーは目的がまったく異なる点にご注意ください。セットアップ パッケージは InstallScript デバッガーではデバッグできません。また、InstallScript カスタム アクションは MSI デバッガーではデバッグできません。

ヒント: *InstallShield が次のタイプのカスタム アクションの 1 つにステップインすると(F11 キーを押すとステップインできます)、クライアントマシンにインストールされている登録済みデバッガーを起動するオプションが提供されます。2 つのオプションから 1 つを選ぶダイアログが表示されます。登録済みのデバッガーを起動するには、[はい]を選択します。次のアクションに進むには [いいえ] を選択します。対応のカスタム アクションは次のとおりです。*

- *MSI DLL*
- *標準 DLL*

ダイレクト エディター

プロジェクト: *ダイレクト エディターは、次のプロジェクト タイプで使用できます:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

次の情報には、プロジェクト固有の詳細が含まれています。

ダイレクト エディターでは、プロジェクト ファイル (.ism) またはデータベース ファイル (.msi、.msm、または .mst) に含まれるすべてのテーブルを参照できます。プロジェクトの種類によっては、このビューでは、Windows Installer データベース フォーマットに詳しい上級開発者向けに、高度な機能も提供されています。

Windows Installer ベースのプロジェクトでは、ダイレクト エディターは 2 つの異なるモードで実行します:

- **プロジェクト編集モード** – プロジェクト ファイルに含まれるテーブルを編集できるモード。ダイレクト エディターで追加された変更は、InstallShield がビルド時に作成する Windows Installer パッケージに組み込まれます。

  InstallShield のその他のビューで追加された変更は、ダイレクト エディターに反映されます。また、ダイレクト エディターで追加された変更も、対応する他のビューがある場合は、そこに反映されます。たとえば、[機能] ビューを使ってプロジェクトに新しい機能を追加すると、その機能がダイレクト エディターの Feature テーブルに自動的に追加されます。[ダイレクト エディター] ビューを使って機能を追加すると、それに対応して [機能] ビューが更新されます。

  Windows Installer ベースのプロジェクト タイプである、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、プロジェクト編集モードを使用します。

- **ダイレクト編集モード** – Windows Installer データベース (.msi、.msm、または .mst ファイル) のテーブルを編集できるモード。ダイレクト エディターで追加した変更を保存するとき、InstallShield が Windows Installer データベースを更新します。

  ダイレクト編集モードでは、その他の標準 InstallShield ビューを利用できません。これは、標準 InstallShield ビューが、Windows Installer データベースを直接編集するときには使用できないビルド時の機能を必要とするためです。

  Windows Installer ベースのプロジェクト タイプである、MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、ダイレクト編集モードを使用します。

ダイレクト エディターを任意のプロジェクト タイプで使用する

ダイレクト エディターには、プロジェクトまたはデータベース内の各テーブルをリスト表示する [テーブル] エクスプローラーがあります。このエクスプローラーでテーブルを選択すると、右側のペインに次の要素が表示されます:

- ボタン行とその他のコントロール
- スプレッドシート形式のテーブル

テーブルの各行は、プロジェクトまたはデータベース内のレコードを意味します。各列のヘッダーに括弧付きで挿入されている表示は、列に入力できるデータの種類とサイズを示します。たとえば、*S255* は文字列の制限文字数が 255 文字であることを示し、*I2* は、2 バイトの整数を示します。

次の表は、ダイレクト エディター内でテーブルを選択したときに表示されるすべてのボタンと、その他のコントロールについて説明します。

テーブル 238 • [ダイレクト エディター] ビューのコントロール

コントロールの名前	アイコン	説明
新しいレコード		選択したテーブルに新しい行を追加できる、[テーブルにレコードを追加] ダイアログ ボックスが表示されます。
選択したレコードを編集		選択した行のデータを編集できる、[テーブルのレコードを編集] ダイアログ ボックスが表示されます。
選択したレコードを削除		選択した行 (複数可) を削除します。
選択したレコードを切り取る		選択中の行を削除して、クリップボードに保存します。
選択したレコードのコピー		選択中の行をクリップボードにコピーします。
レコードの貼り付け		クリップボードに保存されている行を挿入します。
文字列の検索		文字列のインスタンスを検索できる、[検索] ダイアログ ボックスを表示します。このダイアログ ボックスを使って、選択したテーブル内、またはすべてのテーブル内を検索するなどの条件を指定できます。
次を検索		指定した文字列の次の位置を検索します。
検索/置換		文字列のインスタンスを検索して、それを新しい文字列と置換できる、[置換] ボックスを表示します。このダイアログ ボックスを使って、選択したテーブル内、またはすべてのテーブル内を検索するなどの条件を指定できます。
参照の追跡を表示		[ダイレクト エディター] の下にある [参照の追跡] ペインの表示を切り替えます。[参照の追跡] ペインを使って、テーブル レコードの関係性を簡単に確認できます。詳細については、「テーブル レコード参照の追跡」を参照してください。
検索グリッド		この検索ボックスで指定した文字列に従って、[ダイレクト エディター] ビューに表示されるレコードをダイナミックにフィルターします。このボックスに文字列を入力すると、それを含まない行すべてが非表示となります。
ダイレクト エディター ヘルプ		[ダイレクトエディター] ビューのヘルプを表示します。

Windows Installer ベースのプロジェクトで、プロジェクト編集モードを使ったダイレクト エディターでの作業

基本の MSI、DIM、InstallScript MSI、マージ モジュール、および QuickPatch プロジェクトでは、上級ユーザーはダイレクト エディターを使って、次のようなタスクを処理できます:

- プロジェクト ファイル (.ism) 内のすべてのテーブルを参照する。
- テーブルにレコードを追加/削除する。
- レコードまたはフィールドを切り取り、コピー、および貼り付ける。
- テーブル内の個別のフィールドを編集する。
- カスタム テーブルを追加する。
- 表示されているテーブル レコードをフィルターして、特定の文字列を含むプロパティを非表示にする。
- 1 つのテーブル、またはすべてのテーブルで特定の文字列を検索して、必要に応じてそれを置換する。
- テーブル内の列のサイズを変更、およびその順序を変更する。

ヒント: 標準の Windows Installer テーブルが選択されているときに F1 キーを押すと、Windows Installer ヘルプが開き、特定のテーブルについての情報が表示されます。

Windows Installer ベースのプロジェクトで、プロジェクト編集モードでダイレクト エディターを使用するとき、次の詳細にご注意ください。

- File テーブルは、スタティック データのみを表示します。ダイナミック リンク ファイルなどの追加情報が、InstallShield によってビルド時に作成される Windows Installer データベースの File テーブルに追加される場合があります。
- Windows Installer データベースで対応するテーブルとは異なり、.ism ファイルの Binary、Icon、および Patch テーブルは、バイナリ データを格納しません。代わりに、ビルドのソース パスへのリンクが格納されます。
- 標準テーブルと InstallShield テーブルの列属性は、プロジェクト編集モードでは変更できません。ただし、カスタムテーブルの列属性は編集できます。ダイレクト編集モードの場合、列属性は、標準テーブル、InstallShield テーブル、およびカスタム テーブルで編集できます。
- Directory テーブルでは、ローカライズ可能なプロパティは使用できません。

Windows Installer ベースのプロジェクトで、ダイレクト編集モードを使ったダイレクト エディターでの作業

MSI データベース、MSM データベース、およびトランスフォーム プロジェクトでは、ダイレクト エディターを使って、ダイレクト編集モードで使用できる本質的にすべてのタスクを実行できます。ただし、.ism ファイルのテーブルを使った処理ではなく、Windows Installer データベース (.msi、.msm、または .mst) で直接作業を行います。

InstallScript および InstallScript オブジェクト プロジェクトでダイレクト エディターを使用する

InstallScript および InstallScript オブジェクト プロジェクトでは、ダイレクト エディターで、.ism ファイルに含まれるすべてのテーブルを参照できますが、これらのプロジェクト タイプでは、InstallShield のその他のビューを使って変更を行うことが推奨されます。

InstallScript および InstallScript オブジェクト プロジェクトでは、Windows Installer ベースのプロジェクトで使用できるテーブルと同じテーブルの多くを使用しますが、共通テーブルの多くは異なる効果を持ちます。さらに、InstallShield は、InstallScript と InstallScript オブジェクト プロジェクトのカスタム テーブルを無視します。つまり、カスタム テーブルは実行時に使用できません。

ダイレクト エディター テーブルの詳細

InstallShield のダイレクト エディターには、テーブル、スキーマ情報、および検証エラーの詳細を確認できるいくつかの機能が提供されています。これにより、ダイレクト エディターを使って高度な問題を識別および解決するためにトラブルシューティングを行うセットアップ作成者またはパッケージ作成者の生産性が飛躍的に高まります。これらの機能について、次に説明します:

- Directory テーブルの擬似列に、解決されたターゲット ディレクトリ パスが表示される
- 列ヘッダー スキーマ情報のツールヒント
- テーブル レコード参照の追跡
- 破損した参照のインジケータ

Directory テーブルの擬似列に、解決されたターゲット ディレクトリ パスが表示される

InstallShield の Directory テーブルには、読み取り専用で灰色表示された擬似列が含まれており、ここに各行のディレクトリ場所の解決済みパスが表示されます。擬似列はテーブル列と同様に動作しますが、実際にはテーブルに格納されません。擬似列に表示されるテキストは並べ替えることができますが、その値を挿入、更新、または削除することはできません。

列ヘッダー スキーマ情報のツールヒント

InstallShield は列ヘッダー上に、列のスキーマ情報を含むツールヒントを表示します。

例

Directory テーブルには、次のスキーマ ツールヒントが表示されます:

テーブル 239 ・

列ヘッダー	ツールヒント
Directory (s72)	Directory: char(72), required
Directory_Parent (S72)	Directory_Parent: char(72), nullable
DefaultDir (l255)	DefaultDir: char(72), localizable, required
ISDescription (S255)	ISDescription: char(255), nullable
ISAttributes (I4)	ISAttributes: long integer, nullable
ISFolderName (S255)	ISFolderName: char(255), nullable

Binary テーブルには、次のスキーマ ツールヒントが表示されます:

テーブル 240 ・

列ヘッダー	ツールヒント
Name (s72)	Name: char(72), required
Data (V0)	Data: stream, nullable
ISBuildSourcePath (S255)	ISBuildSourcePath: char(255), nullable

CustomAction テーブルには、次のスキーマ ツールヒントが表示されます:

テーブル 241 ・

列ヘッダー	ツールヒント
Action (s72)	Action: char(72), required
Type (i2)	Type: short integer, required
Source (S64)	Source: char(64), nullable
Target (S0)	Target: char(0), nullable
ExtendedType (I4)	Extended Type: long integer, nullable
ISComments (S255)	ISComments: char(255), nullable

テーブル レコード参照の追跡

ダイレクトダイレクト エディターには、テーブル レコード間の関連性を簡単に比較できる [参照の追跡] ペインがあります。ダイレクト エディター上部にある **[参照の追跡を表示]** ボタンを使って、ペインの表示/非表示を切り替えることができます。

各レコードは、1 つ以上のレコードを参照、または 1 つ以上のレコードによって参照されている可能性があります。レコードが強調表示されている場合、別のレコードを参照しているか、別のレコードから参照されていることを示し、[参照の追跡] ペインには、参照が存在するテーブルを表示する **[参照テーブル]** セクション、および実際のレコードの参照を表示する追加セクションが含まれます。レコードの参照セクションには、参照の方向を示す矢印アイコンが表示されます。

- 右向きの緑色矢印は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードを**参照する**ことを示します。
- 左向きの青色矢印は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードによって**参照されている**ことを示します。
- 両方向を指す 2 つの矢印 (右向きの緑色矢印と左向きの青色矢印) は、ダイレクト エディター テーブルで選択されたレコードが [参照の追跡] ペインに表示されているレコードを**参照する**、*および***参照されている**ことを示します。

*メモ: 複数のダイレクト エディター レコードが選択されている場合、フォーカスされているレコードのみの参照が表示されます。さらに、複数のテーブルが **[参照テーブル]** セクションに表示された場合、ダイレクト エディター テーブルで選択されたレコードが参照する、または複数のテーブルのレコードによって参照されることを示します。関連するレコードの参照を表示するには、**[参照テーブル]** セクションにある任意のテーブルをクリックします。*

ヒント: [参照の追跡] ペインでは、セル内をダブルクリックしてレコードの参照間を簡単に移動することができます。

破損した参照のインジケータ

ダイレクト エディターのテーブル レコードが、もう存在しない外部キー レコードを参照している場合があります。InstallShield は、このような破損した参照を持つセルに注意が向くように、赤い背景色を使用します。

たとえば、Component テーブルの **Directory_** 列が、Directory テーブルで見つからないディレクトリ名を参照する場合、**Directory_** 列が赤い背景色で表示されます。

*メモ: ダイレクト エディターの破損した参照インジケータは、[オプション] ダイアログ ボックスの [プリファレンス] タブにある **[参照の整合性を維持]** チェックボックスには関連していません。**"参照の整合性を維持"** 設定は、プライマリ キーを変更したときに、外部キーを更新することを目的とし、破損した参照インジケータは、親の無いレコードを簡単に識別できるよう、破損した参照を表示します。このため、破損した参照は **"参照の整合性を維持"** 設定で行った選択に関係なく表示されます。*

Windows Installer テーブル リファレンス

Windows Installer テーブルについての情報は、MSDN ライブラリの「Database Tables」を参照してください。

InstallShieldテーブル リファレンス

次のテーブルは、アプリケーションに特定の機能性を追加するいくつかの InstallShield テーブルをリストしたものです。特定のテーブルについての詳細は、リンクをクリックしてください。

テーブル 242 ・ InstallShieldテーブルの名前と説明

テーブル名	プロジェクトの種類	説明
ISAlias	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	このテーブルには、InstallShield の最新バージョンを使って InstallShield Professional プロジェクトが Windows Installer ベースのプロジェクト、または InstallScript プロジェクトに変換されたときに使用されたエイリアス情報が含まれます。

テーブル 242 • InstallShieldテーブルの名 前と説明 (続き.)

テーブル名	プロジェクトの種類	説明
ISComCatalogAttribute	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、COM+ カタログの属性についての情報が含まれています。各エントリには、属性が所属する ISComCatalogObject テーブルエントリへの外部キーがあります。
ISComCatalogCollection	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、COM+ カタログコレクションについての情報が含まれています。COM+ カタログコレクションは、COM+ カタログオブジェクトの同じタイプのフォルダーです。たとえば、[コンポーネント サービス] ビューの COM+ アプリケーション項目、[コンポーネント] 項目および [レガシ コンポーネント] 項目は、COM+ カタログ コレクションです。各エントリには、コレクションが所属する ISComCatalogObject テーブル エントリへの外部キーがあります。
ISComCatalogCollectionObject	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、COM+ カタログオブジェクトが所属する COM+ カタログコレクションについての情報が含まれています。
ISComCatalogObject	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、COM+ カタログ オブジェクトについての情報が含まれています。COM+ カタログオブジェクトは、COM+ カタログコレクション内に含まれる項目です。各エントリには表示名があります。
ISComPlusApplication	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、COM+ アプリケーションについての情報が含まれています。COM+ アプリケーションは、追加情報を持つ COM+ カタログ オブジェクトです。このテーブルには、アプリケーション固有の情報と、基本情報を持つ ISComCatalogObject テーブル エントリの外部キーがあります。
ISCustomActionReference	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、インストールに含まれるカスタム アクションそれぞれの動作についての情報が含まれています。

テーブル 242 • InstallShieldテーブルの名前と説明 (続き.)

テーブル名	プロジェクトの種類	説明
ISIISItem	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、[IIS の構成] ビューで構成された各 IIS 要素 (Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、および Web サービス拡張) に関する情報が含まれています。
ISIISProperty	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	このテーブルには、[IIS の構成] ビューで構成された設定の値が含まれます。
ISProductConfigurationInstance	基本の MSI	このテーブルは、同じマシンで製品のインスタンスを同コンテキストで複数インストールするのためのサポートを提供します。
ISSelfReg	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	COM サーバーの自己登録方法として ISSelfReg を選択した場合で、プロジェクトに自己登録としてマークされているファイル (またはダイナミックリンク) が含まれている場合、InstallShieldはこれらのファイルについての情報を自動的に .msi データベースの ISSelfReg テーブルへ追加します。

ISAlias テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

InstallShield Professional で作成されたプロジェクトを InstallShield 2025 にアップグレードする場合、InstallShield は、データを ISAlias テーブルに追加することがあります。

Windows Installer では、識別子にスペースを含めることはできません。また、指定された文字だけを含める必要があります。このため、InstallShield Professional で使用していたコンポーネント名、機能名、またはスクリプト定義フォルダー名の一部が、Windows Installer ベースのプロジェクトへアップグレードしたプロジェクトでは有効ではない場合があります。この問題を解決するため、InstallShield ではエイリアスを使用して、古い名前と新しい識別子が一致するようにしています。(これらの名前はスクリプトで文字列として使用される可能性があるため、InstallShield は用語変更を行いません。)

メモ: *InstallScript プロジェクトにアップグレードするときに、機能名およびスクリプト定義のフォルダーに無効な文字のみを以下に示します。*

*¥ / : * ? " < > |*

コンポーネント名には、これらのキャラクタおよび単一の引用符 (') は無効です。

ISAlias には、次の情報が含まれています。

テーブル 243 : ISAlias テーブル情報

列の名前	情報
エイリアス	InstallShield Professional で使用される名前文字列。
Identifier	InstallShield でエイリアスを参照するために使用される文字列。
テーブル	エイリアスを使用するテーブルの名前 (例、機能)。このテーブルはダイレクト エディターで提供されています。

ISCOMCatalogAttribute テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISCOMCatalogAttribute テーブルには、COM+ アプリケーションについての情報が含まれています。各エントリには、属性が所属する ISComCatalogObject テーブルエントリへの外部キーがあります。

列

ISCatalogObject_(プライマリキー)

ISCatalogObject テーブルへの外部キー

ItemName (プライマリキー)

カタログオブジェクトの名前付き属性

ItemValue

ItemName で定義された属性に関連付けられた値

ISCOMCatalogCollection テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISCOMCatalogCollection テーブルには、COM+ アプリケーションについての情報が含まれています。COM+ カタログコレクションは、COM+ カタログオブジェクトの同じタイプのフォルダーです。たとえば、[コンポーネントサービス] ビューの COM+ アプリケーション、[コンポーネント] 項目および [レガシ コンポーネント] 項目は、COM+ カタログコレクションです。各エントリには、コレクションが所属する ISComCatalogObject テーブル エントリへの外部キーがあります。

列

ISCatalogCollection (プライマリキー)

ISCatalogCollection テーブルの一意のキー

ISCatalogObject_

ISCatalogObject テーブルへの外部キー。これは、カタログ コレクションが所属するカタログ オブジェクトです。

名前

カタログ コレクション名。

ISCOMCatalogCollectionObject テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISCOMCatalogCollectionObject テーブルには、COM+ カタログオブジェクトが属する COM+ カタログコレクションに関する情報が含まれています。

列

ISCatalogCollection_(プライマリキー)

ISCatalogCollection テーブルの一意のキー

ISCatalogObject_(プライマリキー)

ISCatalogObject テーブルへの外部キー。これはカタログ コレクションに含まれるカタログオブジェクトです。

ISCOMCatalogObject テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISCOMCatalogObject テーブルには、COM+ アプリケーションについての情報が含まれています。COM+ カタログオブジェクトは、COM+ カタログコレクション内に含まれる項目です。各エントリには表示名があります。

列

ISCatalogObject (プライマリキー)

ISCatalogObject テーブルの一意のキー。

DisplayName

カタログ オブジェクトの表示名。

ISCOMPlusApplication テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISCOMPlusApplication テーブルには、COM+ アプリケーションについての情報が含まれています。COM+ アプリケーションは、追加情報を持つ COM+ カタログ オブジェクトです。このテーブルには、アプリケーション固有の情報と基本情報を持つ ISComCatalogObject テーブル エントリの外部キーがあります。

列

ISCatalogObject_(プライマリキー)

ISComCatalogObject テーブルへの外部キー。これは、ルート レベルのアプリケーション オブジェクトです。このアプリケーションに関連付けられている他のデータはすべて ISComCatalogObject テーブルから派生させることができます。

ComputerName

ComponentService オブジェクトがリモートマシンからロードされた場合、コンピューター名をここに保存します。ローカル コンピューターには NULL を使用できます。

Component_

コンポーネント テーブルへの外部キー。

ISCustomActionReference テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISCustomActionReference テーブルには、プロジェクトに含まれるカスタム アクションそれぞれの正しい動作に関する情報が含まれています。

テーブル 244 : ISCustomActionReference テーブルの情報

列	Type	キー	Null 使用可能	説明
Action_	Identifier	はい	いいえ	CustomAction テーブルのエントリを参照する識別子。
説明	テキスト	いいえ	いいえ	カスタム アクションの動作を説明するテキスト。 InstallShield は、基本の MSI、InstallScript MSI、またはマージ モジュール プロジェクトのビルドするとき、この列に ISCAReferenceFilePath 列で参照されるファイル内のテキストを挿入します。 ダイレクト編集モードで作業中の場合、[カスタム アクションとシーケンス] ビュー (または [カスタム アクション] ビュー) でヘルプ ファイルを選択するとすぐに InstallShield がファイルのコンテンツをこの列にストリームします。
FileType	テキスト	いいえ	いいえ	ヘルプ ファイルのファイルの種類。
ISCAReference FilePath	テキスト	いいえ	いいえ	カスタム アクションの動作が記述されたファイルの名前を含む完全パス。 この列は InstallShield プロジェクト ファイル (.ism) に含まれますが、.msi ファイルにはありません。

ISIISItem テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ISIISItem テーブルは、[IIS の構成] ビューで構成された IIS の設定に関する情報を含みます。

テーブル 245 • ISIISItem テーブルの情報

列	Type	キー	Null 使用可能	説明
ISIISItem	文字列	はい	いいえ	テーブルのプライマリ キー。このアイテムは、Web サイトまたはその他の IIS アイテムが作成されたことを示します。
ISIISItem_Parent	文字列	いいえ	はい	ISIISItem 列への外部キー。このアイテムは、現在のアイテムを含む Web サイトまたはその他の IIS アイテムを識別します。
DisplayName	ローカライズ可能文字列	いいえ	はい	Web サイトまたはその他の IIS アイテムの翻訳された名前 (オプション)。
Type	整数	いいえ	いいえ	IIS アイテムの種類を識別します。選択可能なオプションは以下のとおりです: • 1 – Web サイト • 2 – アプリケーション • 3 – 仮想ディレクトリ • 4 – アプリケーション プール • 5 – Web サービス拡張
Component_	文字列	いいえ	はい	IIS アイテムを含むコンポーネントの名前。

ISIISProperty テーブル

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ISIISProperty テーブルは、[IIS の構成] ビューで構成された IIS の設定に関する情報を含みます。

メモ: *[IIS 構成] ビューで Web サイト、アプリケーション、または仮想ディレクトリの詳細設定を構成すると、ISIISProperty テーブルの MetaData* フィールドが自動的に構成されます。特定の設定についてのヘルプは、MSDN ヘルプ ライブラリの「IS Metabase Properties」を参照してください。*

テーブル 246 ・ ISIISProperty テーブルの情報

列	Type	キー	Null 使用可能	説明
ISIISProperty	文字列	はい	いいえ	テーブルのプライマリ キー。
ISIISItem	文字列	はい	いいえ	テーブルのプライマリ キー。ISIISItem テーブルへの外部キー。このアイテムは、このプロパティが属する Web サイトまたはその他の IIS アイテムを識別します。
スキーマ	文字列	いいえ	はい	この列は将来利用するために予約されています。
FriendlyName	文字列	いいえ	はい	この列は将来利用するために予約されています。
MetaDataProp	整数	いいえ	はい	METADATA_RECORD 構造体の dwMDIdentifier フィールドと同一です。
MetaDataType	整数	いいえ	はい	METADATA_RECORD 構造体の dwMDDataType フィールドと同一です。 ***メモ:*** *InstallShield は、次の値をサポートします:* - *1 (DWORD_METADATA)* - *2 (STRING_METADATA)* - *5 (MULTISZ_METADATA)*

テーブル 246 • ISIISProperty テーブルの情報 (続き.)

列	Type	キー	Null 使用可能	説明
MetaDataUserType	整数	いいえ	はい	METADATA_RECORD 構造体の dwMDUserType フィールドと同一です。Windows SDK ヘッダー IISCnfg.h で定義されている、使用可能なオプションは以下のとおりです: • 1 (IIS_MD_UT_SERVER) • 2 (IIS_MD_UT_FILE) • 100 (IIS_MD_UT_WAM) • 101 (ASP_MD_UT_APP)
MetaDataAttributes	整数	いいえ	はい	METADATA_RECORD 構造体の pbMDAttributes フィールドと同一です。
MetaDataValue	文字列	いいえ	はい	METADATA_RECORD 構造体の pbMDData フィールドと同一です。
Order	整数	いいえ	はい	プロパティを IIS アイテムに適用する順序。
ISAttributes	整数	いいえ	はい	IIS プロパティが標準プロパティであるか、詳細プロパティであるかを識別します。詳細プロパティは、[IIS の構成] ビューの "その他の IIS プロパティ" 設定で構成することができます。 選択可能なオプションは以下のとおりです: • 0 (標準プロパティ) • 1 (詳細プロパティ) その他すべてのオプションは、将来利用するために予約されています。

ISProductConfigurationInstance テーブル

プロジェクト: *この情報は、基本の MSI プロジェクトに適用します。*

ISProductConfigurationInstance テーブルは、同じマシンで製品のインスタンスを同コンテキストで複数インストールするのためのサポートを提供します。このテーブルは、インストールがサポートする各インスタンスのプロパティ値を個々に定義します。詳細については、「インスタンスのプロパティを設定します」を参照してください。

***注意 -** エンド ユーザーが製品の複数インスタンスを同じマシン上に同じコンテキストでインストールできるインストールを作成するには、インストール開発者の洗練されたオーサリング能力と献身的な取り組みが必要です。この機能の使用は、上級インストール開発者にのみお勧めします。*

テーブル 247 - ISProductConfigurationInstance テーブルの情報

列	Type	キー	Null 使用可能	コメント
ISProductConfiguration_	テキスト	はい	いいえ	このインスタンスが適用される製品構成を識別します。
InstanceId	整数	はい	いいえ	このインスタンスのインスタンス番号を識別します。この値は InstanceId のプロパティに格納されます。
プロパティ	Identifier	はい	いいえ	このインスタンスに作成するプロパティを識別します。
値	テキスト	はい	いいえ	このインスタンスのプロパティの値を指定します。

ISSelfReg テーブル

***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

COM サーバーの自己登録方法として ISSelfReg を選択した場合で、プロジェクトに自己登録としてマークされているファイル (またはダイナミックリンク) が含まれている場合、InstallShieldはこれらのファイルについての情報を自動的に .msi データベースの ISSelfReg テーブルへ追加します。

テーブル 248 - ISSelfReg テーブルの情報

列の名前	説明
FileKey	File テーブルに入る外部キー。自己登録するファイル (.dll、.ocx、.exe、.tlb、.olb) を識別します。
コスト	後で使用するために予約されています。
Order	自己登録の発生順を示す負の数以外の数。Order の値が 1 のファイルが最初に登録され、値が 2 のファイルが次、という具合に順番に登録されます。Order の値が 0 のすべてのファイルが最後に登録されます。

テーブル 248 • ISSelfReg テーブルの情報 (続き.)

列の名前	説明
CmdLine	自己登録の .exe ファイルの場合、自己登録されたときに実行可能ファイルに渡すコマンドラインを格納します。このフィールドはデフォルトでは空です。InstallShield ランタイム エンジンは .exe ファイルを /regserver 引数を使って登録し、/unregserver 引数を使って登録解除します。 登録および登録解除中に別の引数を渡すよう指定するには、/hello|/goodbye のように引数を垂直の線 (|) で区切ります。

InstallShield標準 Windows Installer テーブルの列

InstallShieldでは、機能の数が増えたのに伴い、標準 Windows Installer テーブルが拡張しました。変更があったテーブル、追加された列、および新しいフィールドに配置するデータの説明を以下に示します。

テーブル 249 • 標準 MSI テーブルへの追加

テーブル名	列の名前	説明
AdminExecuteSequence	ISComments	このシーケンスに関する作成者コメント。
AdminUISequence	ISComments	このシーケンスに関する作成者コメント。
AdvtExecuteSequence	ISComments	このシーケンスに関する作成者コメント。
AdvtUISequence	ISComments	このシーケンスに関する作成者コメント。
バイナリ	ISBuildSourcePath	ICO または EXE ファイルへの完全パス。
コンポーネント	ISAttributes	コンポーネントの InstallShield カスタムプロパティを保存するために使用します。現在は ExtractAtBuild でのみ使用されています。
コンポーネント	ISComments	ユーザー コメント。
Control	ISWindowStyle	このコントロールに適用する非 MSI ウィンドウスタイルを指定する 32 ビット言語。
Control	ISControlId	コントロールのコントロール ID を表すのに使用した数。ダイアログのエクスポートで使用。
CustomAction	ISComments	このカスタム アクションのコメントを入力します。
ダイアログ	ISComments	このダイアログに関する作成者のコメント。

テーブル 249 • 標準 MSI テーブルへの追加 (続き.)

テーブル名	列の名前	説明
ダイアログ	ISWindowStyle	このコントロールに適用する非 MSI ウィンドウスタイルを指定する 32 ビット言語。スクリプトベースのセットアップでのみ使用されます。
ダイアログ	ISResourceId	ダイアログのエクスポートで使用したダイアログ ID を表す数。
Directory	ISDescription	フォルダーの説明。
機能	ISReleaseFlags	この機能が特定のリリースにビルドされるかどうかを指定するリリース フラグ。
機能	ISComments	コメント。
File	ISBuildSourcePath	完全パス。パス変数は使用する可能性があるため、パスではなくテキストでのカテゴリ指定。
File	ISAttributes	このフィールドには次の属性が含まれています: • OverrideSystemAttributes (0x04) • OverrideSystemSize (0x08) • OverrideSystemVersion (0x10) • OverrideSystemLanguage (0x20) UseSystemSettings(0x1) 属性は、現在使用されていません。
アイコン	ISBuildSourcePath	ICO または EXE ファイルへの完全パス。
アイコン	ISIconIndex	抽出するアイコン インデックス (オプション)。
InstallExecuteSequence	ISComments	このシーケンスに関する作成者コメント。
InstallUISequence	ISComments	このシーケンスに関する作成者コメント。
パッチ	ISBuildSourcePath	パッチ見出しへの完全パス。
ProgID	ISAttributes	ExtractIcon など、コンポーネントの InstallShield カスタムプロパティを保存するために使用します。
プロパティ	ISComments	ユーザー コメント。

テーブル 249 • 標準 MSI テーブルへの追加 (続き.)

テーブル名	列の名前	説明
レジストリ	ISAttributes	レジストリ項目の InstallShield カスタム プロパティを保存するために使用します。現在は Automatic でのみ使用されています。
ショートカット	ISComments	このショートカットに関する作成者コメント。

QuickPatch プロジェクト

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトは、規模の小さい単一のアップグレードをエンド ユーザーに配布したいインストール作成者へお勧めする特殊な Windows Installer ベースのプロジェクトです。QuickPatch プロジェクトごとに、1 セットのビューがあります。

[パッチの設定] ビュー

パッチの設定 ビューには、他のビューへのリンクがあります。

- **[一般情報] ビュー** – アップグレードおよびオリジナル インストールについての情報を入力します。現在のプロジェクトがパッチするリリースを指定し、プロジェクトに含めるカスタム アクションを選択します。
- **[ファイル] ビュー**–パッチにファイルを追加してファイルのパッチ オプションを指定します。
- **[レジストリ] ビュー** – 変更、または削除するレジストリの値を指定します。
- **[パス変数] ビュー** – 変数ベースのファイルリンクを使って、開発システム間のパッチの移動を容易にします。

[追加ツール] ビュー

[追加ツール] ビューには、ダイレクト エディター へのリンクがあります。ダイレクト エディターを使ってデータベースのテーブルを直接編集します。

[パッチの設定] ビュー

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[パッチの設定] ビューは、QuickPatch プロジェクトを作成または開いたときに利用できます。このビューを利用して、QuickPatch を構成します。

一般情報

アップグレードおよびオリジナル インストールについての情報を入力します。現在のプロジェクトがパッチするリリースを指定し、プロジェクトに含めるカスタム アクションを選択します。

ファイル

パッチにファイルを追加して、ファイル パッチ オプションを指定します。

レジストリ

変更、または削除するレジストリの値を指定します。

パス変数

変数ベースのファイルリンクを使って、開発システム間のパッチの移動を容易にします。

[一般情報] ビュー

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビューには、QuickPatch プロジェクトに関する基本的な情報が含まれています。このビューでは製品プロパティ、ビルド設定、パッチ履歴、およびカスタム アクションを表示し、構成することができます。

製品のプロパティ

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビューで [製品のプロパティ] をクリックすると、以下が表示されます。

テーブル 250 • 製品のプロパティの設定

設定	説明
元のセットアップ パス	ここでは、製品名とオリジナル インストール イメージへのパスが表示されます。パス ボックスは読み取り専用です。
製品バージョン	[オリジナル セットアップのバージョン] ボックスには、オリジナル インストールの製品バージョンが表示されます。このボックスは読み取り専用です。[QuickPatch バージョン] ボックスで、QuickPatch プロジェクトの製品バージョンを変更することができます。このボックスへの入力はオプションです。
FlexNet Connect の統合	FlexNet Connect がオリジナル インストールで有効にされていて、FlexNet Connect が製品を識別するために使用するカスタム バージョン番号を指定する場合 (Windows Installer が使用する標準バージョン スキームを使わずにという意味で)、このボックスで作成中の QuickPatch に新しいカスタム バージョン番号を入力します。

ビルドの設定

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビューで [ビルドの設定] をクリックすると、以下のタブが表示されます。

- 共通
- 識別
- デジタル署名
- 詳細

[共通] タブ

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビューで [ビルドの設定] をクリックすると、いくつかのタブが表示されます。[共通] タブには、QuickPatch のためのよく利用されるビルドの設定があります。

ビルドの場所

パッチ ファイルをビルドする場所を指定、または既存フォルダーを検索します。

ランチャの設定

ここでは、次のようなランチャー設定を構成することができます。

テーブル 251 • 構成可能なランチャー設定

設定	説明
Update.exe の作成	現在の QuickPatch パッケージに Update.exe アップデート ランチャを作成する場合、このチェック ボックスを選択します。 Update.exe アップデート ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。
Windows Installer 3.1 エンジンを含める	このチェック ボックスを選択して Windows Installer 3.1 エンジン をパッチパッケージに含めます。
.NET Framework を含める	このチェック ボックスを選択して、.NET Framework をパッチパッケージに含めます。

パッチのアンインストール

アプリケーション全体と他の QuickPatch パッケージをアンインストールして再度インストールし直す手間を掛けずにパッチをアンインストールできるようにする場合、**[パッチのアンインストールを許可する (Windows Installer 3.0 が必要)]** チェック ボックスを選択します。QuickPatch パッケージのアンインストールは一定の条件下のみで

動作しますのでご注意ください。たとえば、バージョン 3.0 よりも前の Windows Installer はアプリケーションから QuickPatch のみを削除することができません。さらに詳しい情報は、Windows Installerヘルプの「パッチの削除」を参照してください。

[識別] タブ

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトの [一般情報] ビューで [ビルドの設定] をクリックすると、いくつかのタブが表示されます。[識別] タブには、表示文字列の設定があります。表示文字列は、[プログラムの追加と削除] で表示されるパッチに関する情報に使用されます。メタデータは、ターゲットマシンの適用されたパッチを検索しカタログする Windows Installer 3.0 以降の API によっても使用されます。

メモ: *パッチのメタデータはパッチ (.msp) ファイルに直接格納され、.msi パッケージには格納されません。.msp ファイルのコンテンツはローカライズされません。したがって、文字列エントリは、以下のメタデータの設定には使用できません。*

テーブル 252 : [識別] タブの設定

設定	説明
表示名	パッチの名前を指定します。
サポート URL	エンドユーザーにテクニカルサポートを提供する URL を指定します。
説明	パッチの簡単な説明を指定します。
製造元の名前	アプリケーションの製品メーカーの名前を指定します。
ターゲット製品名	アプリケーションまたはターゲット アプリケーション スイートの名前を指定します。
分類	アップグレードのカテゴリーを指定します。たとえば、クリティカルアップデート、ホットフィックス、および、サービスパック等。

[デジタル署名] タブ

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトの [一般情報] ビューで [ビルドの設定] をクリックすると、いくつかのタブが表示されます。[デジタル署名] タブで、パッチにデジタル署名をするときの設定を指定することができます。

メモ: *QuickPatch プロジェクトでは、パッチ パッケージと* Update.exe *ファイルにデジタル署名することができます。QuickPatch パッケージの各ファイル (アプリケーションの実行可能ファイルなど) にデジタル署名を行う場*

合、それらを手動で署名してから、プロジェクトに追加しなければなりません。Windows SDK に含まれている SignTool.exe *を使って、手動でファイルに署名を行うことができます。*

テーブル 253 ・[デジタル署名] タブの設定

設定	説明
パッチパッケージに署名する	QuickPatch パッケージにデジタル署名を行う場合、このチェック ボックスを選択します。
アップデート ランチャーに署名する	Update.exe パッケージにデジタル署名を行う場合、このチェック ボックスを選択します。
64 ビット署名を使用	64 ビット署名フレームワークのみを使って QuickPatch パッケージにデジタル署名を行う場合、このチェックボックスを選択します。
署名の種類	パッチパッケージと Update.exe ファイルにデジタル署名を付ける方法を指定します。署名ツール構成の引数を指定します: • **標準** ー InstallShield デフォルト署名ツールを使って、パッチ パッケージおよび Update.exe ファイルにデジタル署名を行います。 • **カスタム** ー 独自のカスタマイズ署名ツールを使って、パッチ パッケージおよび Update.exe ファイルに署名を行います。このオプションを設定すると、**"パス"** および **"引数"** 設定が有効になります。 ***メモ・** デフォルトで、この設定は **[標準]** になっています。*
証明書 URL	完全修飾 URL を入力します (例、**http://www.mydomain.com**)。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。

テーブル 253 • [デジタル署名] タブの設定 (続き.)

設定	説明
デジタル証明書情報	リリースに署名を行うために使用するデジタル証明書を指定するには、この設定の横にある [参照] ボタンをクリックします。[証明書の選択] ダイアログ ボックスが開いて、.pfx ファイル、証明書ファイル (EV がエクスポートした .cer ファイル) の場所を指定するか、証明書を含む証明書ストアについての情報を指定することができます。 InstallShield では、パブリック キー証明書 (.cer) ファイルを使って EV トークン パスワードを暗号化してプロジェクト ファイルに保存するオプションが提供されています。.cer ファイルは通常、USB eToken プロバイダー (たとえば、SafeNet 認証クライアント) と関連付けられた EV 認証クライアント ツールからパブリック キーをエクスポートして作成されます。.cer ファイルが指定された場合、InstallShield は Extended Validation 証明書 (EV) プロパティを構成するための追加オプションを表示します。詳細については、次を参照してください。 • InstallShield における拡張検証 (EV) 証明書情報の構成 • [証明書の選択] ダイアログ ボックス .pfx ファイルを指定するか、テスト ストアから証明書を選択すると、次が表示されます: • **証明書の拇印** – この読み取り専用設定は、証明書の拇印を表示します。 • **発行元** – この読み取り専用設定は、証明書の発行元情報を表示します。 • **有効期限日** – この読み取り専用設定は、証明書の有効期限日を表示します。 証明書ファイル (EV がエクスポートした .cer ファイル) を指定すると、次の設定が表示されます: • **暗号化サービス プロバイダー** – この設定を使って、暗号化サービス プロバイダー (CSP) を指定します。 • **コンテナ名** – この設定を使って、暗号化サービス プロバイダー (CSP) に関連付けられた秘密キーのコンテナ名を指定します。 • **トークン パスワード** – この設定を使って、暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを指定します。 ***メモ:*** *証明書ファイル (EV がエクスポートした .cer ファイル) を指定した後に表示される設定を使用する際は、以下の情報を考慮してください:* • *暗号化サービス プロバイダー (CSP) 名とコンテナ名の両方は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*

テーブル 253 • [デジタル署名] タブの設定 (続き.)

設定	説明
パスワード	使用する .pfx にパスワードがある場合、それを入力します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ストアにパスワード付きでインポートされた証明書をプロジェクトで使用するように構成すると、ビルド時、InstallShield がプロジェクトのファイルに署名を行うときに、Windows がパスワードをプロンプトします。Windows が使用する強力なキー保護のため、InstallShield がパスワードを暗号化サービス プロバイダーに提供することはできません。
パス	パッチ パッケージおよび Update.exe ファイルにデジタル署名を行うために使用する署名ツールの場所を指定します。署名ツールの場所を指定するには、この設定で省略記号ボタン (...) をクリックします。
引数	署名ツール構成のコマンドライン引数を指定して、パッチ パッケージおよび Update.exe ファイルにデジタル署名を行います。たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。 sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename] [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクト内で定義されている任意のパス変数または環境変数を使用することができます。
署名の説明	(必要な場合) パッチ パッケージおよび Update.exe ファイルに使用する署名の説明を指定します。ここで指定した説明は、UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示されます。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。 この設定を空白のままに残すと、InstallShield は UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示される説明として、ファイル名を拡張子なしで使用します。

[詳細] タブ

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビューで [ビルドの設定] をクリックすると、いくつかのタブが表示されます。[詳細] タブには、QuickPatch 用に構成することができるビルドの設定がすべてあります。

全般

[全般] 領域では、次の設定を行います。

テーブル 254 • 全般設定

プロパティ	説明
ビルドの場所	パッチ ファイルをビルドする場所を指定、または既存フォルダーを検索します。
プロジェクト ファイルの形式	プロジェクト ファイル (.ism) に使用するファイル形式を選択します。選択可能なオプションは以下のとおりです: • **バイナリ** – ism ファイルをデータベース ファイルとして保存するには、このオプションを選択します。プロジェクトを開いたり保存したりする際のスピードは、この形式が最も早いです。これは、現在のプロジェクト タイプのデフォルト形式です。現在のプロジェクトにこのプロジェクト ファイル形式を選択すると、Windows Installer データベース エディターで .ism ファイルを更新できます。また、Windows Installer API またはそのオートメーション インターフェイスを使って、.ism ファイルを更新することもできます。 • **XML** – .ism ファイルを階層構造を持つテキスト ベースの形式で保存するには、このオプションを選択します。このプロジェクト ファイル形式は、ソース コード管理システムでの使用に最適です。XML ツールを使用して、プロジェクト ファイルを更新できます。 ***メモ:*** *どちらのプロジェクト ファイル形式でも、コマンドラインからリリースをビルドできます。*
Update.exe の作成	現在の QuickPatch パッケージに Update.exe アップデート ランチャを作成するかどうかを指定します。この設定で **[はい]** を選択すると、Update.exe アップデート ランチャーを作成することができます。 Update.exe アップデート ランチャーが必要になる場合についての詳細は、「パッチ時の考慮事項」を参照してください。
アップデート ランチャー名	InstallShield がビルド時に生成するアップデート ランチャー ファイルに使用するファイル名を指定します(.exe ファイル拡張子は除きます)。 この設定が空白の場合、InstallShield は [アップデート] のデフォルト値を使用し、アップデート ランチャー ファイルは Update.exe と呼ばれます。 ***メモ:*** *"ランチャー名の更新" 設定を有効にするには、"Update.exe の作成" 設定も [はい] に設定されていることを確認します。*

テーブル 254 • 全般設定 (続き.)

プロパティ	説明
64 ビット	現在の QuickPatch パッケージに 64 ビット バージョンの Update.exe アップデート ランチャーを作成するかどうかを指定します。次のオプションを選択できます: • **はい**—これを選択すると、64 ビットバージョンの update.exe アップデート ランチャーが作成されます。 • **いいえ**—これを選択すると、32 ビットバージョンのupdate.exe アップデート ランチャーが作成されます。 デフォルトで、この設定は **[いいえ]** になっています。 ***メモ: "ランチャー名の更新"** 設定を有効にするには、**"Update.exe の作成"** 設定も **[はい]** に設定されていることを確認します。*
置換するパッチ GUID のリスト	現在の QuickPatch で、1 つまたは複数の以前のインストール済みのパッチを置換するには、このプロパティにそれらのパッチのパッチ GUID をカンマで区切って設定します。例: **{C86838C9-DEDC-4451-B96F-94AFB9460F15},{C8633E5B-AC44-45d8-B487-C68B3B1F60D6}** [履歴] ビューに QuickPatch プロジェクトが複数あっても、このプロパティの設定は通常、*必要ありません*。ただし QuickPatch プロジェクトが以前の QuickPatch プロジェクトに追加されたファイルを上書きしない場合、このプロパティの設定が必要な場合もあります。 置換するパッチの GUID が分からない場合、このプロパティをクリックしてから、省略記号 (...) ボタンをクリックして参照します。パッチ (.msp または .exe ファイル) を選択すると、対応する GUID がこのプロパティに InstallShield によって追加されます。
新規 UpdatedImage フォルダーを作成	この設定を利用して、既存の UpdatedImage フォルダーと共に QuickPatch パッケージをビルドすることが可能になります。このオプションを利用するには、このパッケージを少なくとも 1 度ビルドしている必要があります (UpdatedImage フォルダーが必要な為) 。このオプションを [いいえ] に設定すると、既存の UpdatedImage フォルダーからパッケージをビルドします。この方法で UpdatedImage フォルダーにある MSI パッケージを微調整して、その .msi データを QuickPatch プロジェクトで利用することができます。このオプションを [はい] (デフォルト値) にすると、パッケージをビルドする度に UpdatedImage フォルダーに .msi ファイルが再生成されます。
MsiPatchOldAssembly テーブルの生成	MsiPatchOldAssemblyFile および MsiPatchOldAssemblyName テーブルのエントリを自動的に生成するかどうかを指定します。これらのエントリを利用すると Windows Installer 3.0 以降で実行されるパッチ パッケージが元のインストール ソースを実行時に要求することなく、GAC (グローバル アセンブリ キャッシュ) のアセンブリをパッチすることができます。詳細については、「グローバル アセンブリ キャッシュのアセンブリをパッチする」を参照してください。

テーブル 254 • 全般設定 (続き.)

プロパティ	説明
パッチ シーケンスのエントリ作成	QuickPatch にパッチ シーケンシングを使用するかどうかを指定します。パッチ シーケンスを参照して、もう使われていないパッチ、入れ替わったパッチ、または製品に既に適用済みのパッチを見ることができます。シーケンスは、パッチがターゲットマシンに提供された順番に関係なく、どの順番で Windows Installer バージョン 3.0 以上がインストールされた製品にパッチを適用するかを指定します。バージョン 3.0 以前の Windows Installer の場合、パッチ シーケンスは無視され、すべてのパッチは、ターゲットマシンに提供された順番で製品に適用されます。
QuickPatch の簡素化	QuickPatch パッケージの簡素化を行い、最もシンプルなパッケージをビルドするかどうかを指定します。デフォルトの値は [はい] です。 QuickPatch の簡素化を行うと、通常の QuickPatch パッケージに比べて新しいサブ機能とカスタム アクションの数が少ない QuickPatch パッケージを生成できます。 たとえば、QuickPatch プロジェクトに新しいファイルとレジストリ エントリが含まれていて、が QuickPatch の簡素化を行わなかった場合、そのファイルとレジストリ エントリ用に新しいサブ機能が作成されます。さらに、特定の Windows Installer パッチ要件に対応するため、1 つまたは複数のビルド済み カスタム アクションが追加されます。これに対し、InstallShield が QuickPatch の簡素化を行う場合、ファイルまたはレジストリ エントリは既存する機能に追加されるため、特別なビルド済み InstallShield カスタム アクションは必要ありません。 ***メモ:*** *次のシナリオにおいて、InstallShield が QuickPatch パッケージの作成処理を簡素化することはできません。* • *QuickPatch パッケージがインストール済みのファイルを削除する。* • *QuickPatch パッケージがレジストリ キーを削除する、またはその名前を変更する。* • *QuickPatch パッケージが、簡素化されていない通常の QuickPatch イメージをターゲットとする。つまり、[一般情報] ビューの [履歴] 領域で、簡素化を行わなかった QuickPatch のチェック ボックスを選択した場合、QuickPatch の簡素化はできないということです。1 つまたは複数の簡素化されていない QuickPatch イメージをターゲットとする簡素化された QuickPatch のビルドを試みると、ビルド警告が表示され、簡素化は行われません。*
ランチャをパスワードで保護	QuickPatch パッチをパスワードで保護するには、[はい] を選択してから "ランチャ パスワード" 設定でパスワードを入力します。QuickPatch パッチをパスワードで保護すると、QuickPatch パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。 この設定は Update.exe ファイルを使用する QuickPatch パッチのみに適用します。

テーブル 254 • 全般設定 (続き.)

プロパティ	説明
ランチャ パスワード	アプリケーションを保護するパスワードを入力します。パスワード保護をアクティブにするには、"ランチャをパスワードで保護" 設定で [はい] を選択する必要があります。 パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。 この設定は Update.exe ファイルを使用するパッチのみに適用します。

Windows Installer エンジン

ここでは、Windows Installer エンジンのための次のような設定を構成することができます。

テーブル 255 • Windows Installer エンジンの設定

プロパティ	説明
Windows Installer 3.1 エンジンを含める	Windows Installer エンジンをパッチパッケージに含めるかどうか指定します。
エンジンの場所	次のオプションから選択してください。 • **Web からエンジンをダウンロードする** — Windows Installer エンジンのインストールが必要な場合、実行時にダウンロードされます。 • **Update.exe からエンジンを抽出する** — ビルドは Windows Installer エンジンを Update.exe ファイルにストリームし、顧客に配布する単一ファイルを生成します。エンジンは、実行時に Update.exe ファイルから抽出され必要に応じてインストールされます。 • **ソース メディアからコピーする**—ビルドは、Windows Installer エンジンを Update.exe ファイルと同じディレクトリにコピーします。
Windows Installer 3.1 エンジン URL	エンジン場所の URL を指定します。Update.exe ファイルは、この場所を、実行時に、エンジンをダウンロードする場所として使います。便宜上、デフォルトの URL の場所は Revenera が運営するライブ サイトとなっています。

Microsoft .NET Framework

ここでは、Microsoft .NET Framework エンジンのための次のような設定を構成することができます。

テーブル 256 • Microsoft .NET Framework の設定

プロパティ	説明
ビルドに含める	パッチに Microsoft .NET Framework を含めるかどうかを指定します。

テーブル 256 • Microsoft .NET Framework の設定 (続き.)

プロパティ	説明
エンジンの場所	次のオプションから選択してください。 • **Web からエンジンをダウンロードする** — .NET Framework エンジンのインストールが必要な場合、実行時にダウンロードされます。 • **Update.exe からエンジンを抽出する**—ビルドは.NET Framework を Update.exe ファイルにストリームし、顧客に配布するシングルファイルを生成します。エンジンは、実行時に Update.exe ファイルから抽出され必要に応じてインストールされます。 • **ソース メディアからコピーする** — ビルドは、.NET Framework を Update.exe ファイルと同じディレクトリにコピーします。
エンジン URL	.NET Framework の場所の URL を指定します。Update.exe ファイルは、この場所を、実行時に、.NET Framework をダウンロードする場所として使います。便宜上、デフォルトの URL の場所は Revenera が運営するライブ サイトとなっています。

アップデート起動ツールの設定

この領域では、Update.exe 起動ツールにおける次の設定を構成することができます:

テーブル 257 • パッチ用のアップデート起動ツールの設定

プロパティ	説明
会社名	Update.exe のデフォルトの会社名を独自の会社名でオーバーライドするには、ここに会社名を入力します。 会社名が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
製品名	Update.exe のデフォルトの製品名を独自の製品名でオーバーライドするには、ここに製品名を入力します。 製品名が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 257 • パッチ用のアップデート起動ツールの設定 (続き.)

プロパティ	説明
製品バージョン	Update.exe のデフォルトの製品バージョンを独自の製品バージョンでオーバーライドするには、ここに製品バージョンを入力します。 製品バージョンが、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
説明	Update.exe のデフォルトの説明を独自の説明でオーバーライドするには、適切な説明を入力します。 説明が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックしたときに表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
著作権情報	Update.exe のデフォルトの著作権情報を製品の著作権情報でオーバーライドするには、製品の著作権情報を入力します。 著作権情報が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 257・パッチ用のアップデート起動ツールの設定 (続き.)

プロパティ	説明
必要実行レベル	"必要実行レベル" 設定を使って、Windows Vista 以降のプラットフォーム上でプロジェクトの Update.exe ファイルを実行するのに必要な最低実行レベルを指定します。選択可能なオプションは、以下のとおりです。 • **管理者** — Update.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。 非管理者は、管理者としての認証が必要になります。 • **最高権限** — Update.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を承認する必要があります。 非管理者は、管理者権限を持たずに Setup.exe を実行します。 • **起動者** — Update.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Update.exe を実行することができます。Update.exe は、資格情報または同意 (コンセント) を求める UAC メッセージを表示しません。 • **前のセットアップ マニフェストを使用** — Update.exe マニフェストは前のセットアップで指定された必要実行レベルと同じレベルを使用します。デフォルトでは、これが設定されています。 InstallScript MSI プロジェクトと基本の MSI プロジェクトの場合、[Update.exe の作成] チェック ボックスを選択すると、InstallShield が Update.exe 起動ツールにアプリケーション マニフェストを埋め込みます。このマニフェストは選択された実行レベルを指定します。Windows Vista よりも古いバージョンのオペレーティング システムでは、必要実行レベルは適用されません。 基本の MSI プロジェクトで [Update.exe を作成する] チェック ボックスがクリアされている場合、InstallShield は Update.exe 起動ツールに Windows アプリケーション マニフェストを埋め込みません。 詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。
アイコン	Update.exe ファイルに独自のアイコンを使用する場合、そのアイコンを含むファイルの完全修飾名を指定します。ファイルを指定するには、絶対パスまたはパス変数の相対パスを入力するか、省略記号ボタン (...) をクリックして [アイコンの変更] ダイアログ ボックスでファイルを参照します。 デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、[アイコンの変更] ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。 このフィールドを空白のままにした場合、Update.exe ファイルにデフォルトのアイコンが使用されます。

履歴

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビュー内の [一般情報] エクスプローラーにある [履歴] 項目は QuickPatch プロジェクトの概要です。すべての関連リリースの一覧を表示し、現在の QuickPatch プロジェクトでどのリリースをパッチするのかを指定することができます。

カスタム アクション

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[一般情報] ビュー内の [一般情報] エクスプローラーにある [カスタム アクション] 項目は、パッチを作成しているオリジナル インストール プロジェクトで定義されているカスタム アクションのリストです。現在の QuickPatch プロジェクトで実行するカスタム アクションを指定することができます。

[ファイル] ビュー

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[ファイル] ビューで、QuickPatch のファイルを管理をしたり、オリジナル インストールに使われているファイルのバージョン番号、言語および他の情報を見たりすることができます。[パッチするファイル] および [元のセットアップ ファイル] エクスプローラー内のすべてのファイルはキーファイル順にリストされています。

[元のセットアップ ファイル] エクスプローラーの中のファイルをクリックすると、ファイル名、インストール先、バージョン番号、および他の関連する情報が右のペインに表示されます。このエクスプローラーから [パッチするファイル] エクスプローラーにファイルをドラッグ アンド ドロップすることができます。[パッチするファイル] エクスプローラーでは、QuickPatch が元のインストールに適用されたときに追加、変更または削除されるファイルがすべて表示されます。

新規ファイルの設定

プロジェクト: *この情報は、QuickPatch プロジェクトに適用します。*

[ファイル] ビュー内の [パッチするファイル] エクスプローラーで新規ファイルをクリックすると、右のペインで次のような設定を構成することができます。

ファイルの設定

テーブル 258 • [ファイルの設定] 領域にある設定

設定	説明
パッチセットアップに追加するファイルを指定します。	オリジナル インストールに含まれるファイルに変更を行うとき、パッチが適用されるときにターゲット システムにインストールするファイルの最新バージョンを指定します。
自己登録	追加するファイルが自己登録 DLL または OCX ファイルの場合、このチェック ボックスを選択します。 ***メモ•*** *このオプションは .exe または .tlb ファイルの自己登録を行いません。*
COM 情報の抽出	ファイルが COM サーバーの場合で、パッチのビルド時にファイルから COM 情報を自動的に抽出するには、このチェック ボックスを選択します。

統合の設定

ファイルの保存先、および、このファイルと関連付ける機能を選択します。

メモ• *このインストール状態には機能の関連付けが必要です。QuickPatch プロジェクトへ新規データを追加する場合、それを機能と関連付ける必要があります。新規データは対応する機能がインストールされている場合のみインストールされます。*

変更/削除されたファイルの設定

プロジェクト• *この情報は、QuickPatch プロジェクトに適用します。*

[ファイル] ビュー内の [パッチするファイル] エクスプローラーでオリジナル インストールに追加したファイルをクリックすると、次のような設定を構成することができます。

アップデートされたファイル

テーブル 259 ・［アップデートされたファイル］領域の設定

設定	説明
ファイルの最新バージョンを指定する	オリジナル インストールに含まれるファイルに変更を行うとき、パッチが適用されるときにターゲット システムにインストールするファイルの最新バージョンを指定します。
ファイル全体として含める	選択したファイルを、バイト レベルのファイルの差分だけではなく、1 つのファイルとしてパッチに含める場合、このチェック ボックスの選択を考慮してください。その場合、InstallShield は Windows Installer が選択されたファイルの実際のバージョン番号をすべて無視し、またターゲット システムのファイルを QuickPatch パッケージに含まれるバージョンでアップデートするかしないかを決定するときに、選択されたファイルがバージョン 1.0.0.0 であると想定するようにパッチを構成します。 ターゲット システムのファイルを QuickPatch パッケージに含まれるバージョン指定がされていない対応する新しいファイルで常に上書きする場合、バージョンが指定されていないファイルでこのチェック ボックスを選択すると便利です。Windows Installer ファイルの上書き規則では、バージョンが指定されているファイルが、バージョンが指定されていないファイルに優先します。したがって、このチェック ボックスを選択すると、Windows Installer はバージョンが指定されていないファイルを QuickPatch パッケージに含まれるファイルでアップデートします。これは、QuickPatch に含まれるファイルのバージョンが指定されていると判断するためです。 ***重要 ・** Windows Installer ファイルの上書き規則によると、ターゲット システム上のファイルの方がインストールされるバージョンより新しい場合でも、常に最新バージョンのファイルが保持されます。このため、ターゲット システム上のファイルのバージョンが指定されている場合、一部の状況において、ターゲット システム上のファイルがアップデートされない可能性があるため、このチェック ボックスの選択は避けたほうが良い場合があります。たとえば、ターゲット システム上のファイル バージョンが 1.1.0.0 で、QuickPatch プロジェクトのファイル バージョンが 2.0.0.0 の時にこのチェック ボックスを選択すると、実行時、ターゲット システム上のファイルはアップデートされません。これは、QuickPatch パッケージのファイルの実際のバージョン番号がターゲット システム上のファイルのバージョン番号よりも新しいにも関わらず、Windows Installer が QuickPatch のファイル バージョンをターゲット システム上のファイル バージョンよりも古い 1.0.0.0 と判断したためです。*

テーブル 259 • [アップデートされたファイル] 領域の設定 (続き.)

設定	説明
COM 情報の抽出	ファイルが COM サーバーの場合で、パッチのビルド時にファイルから COM 情報を自動的に抽出するには、このチェック ボックスを選択します。このチェック ボックスは、ベース .msi パッケージの Class または TypeLib テーブルに COM 情報を含むパッチ ファイルのデフォルトとして選択されています。 ***メモ•*** *パッチを行う既存のファイルを指定するとき、InstallShield は自動的に、そのファイルが自己登録型であるかどうかを検出します。元にファイルが自己登録型である場合、パッチに含まれるファイルも自己登録型に設定されます。*

元のファイルの情報

[オリジナル ファイル情報] では、オリジナル インストールのオリジナル ファイルについての情報を見ることができます。

コンポーネントの情報

[コンポーネント情報] では、コンポーネント名、キーファイル等のファイル コンポーネント情報を見ることができます。

このボックスをチェックして、ファイルをセットアップからパッチで削除する

パッチが適用されるとき、ターゲット マシンからファイルを削除する場合、このチェック ボックスを選択します。

[レジストリ] ビュー

プロジェクト• *この情報は、QuickPatch プロジェクトに適用します。*

QuickPatch プロジェクトの [レジストリ] ビューでは、現在ソースマシンに存在する項目をビジュアル表示、またはインストール先/ターゲット システム上でパッチプロジェクトがビルドされ、適用された後にパッチを行う項目をビジュアル表示します。[インストール先コンピューターの [レジストリ] ビュー] ペインおよび [インストール先コンピューターのレジストリ データ] ペインは、元のインストールのレジストリ エントリが予め挿入されています。

インストール先コンピューターのレジストリ データペインに表示される各項目は、値タイプとデータの現在の状況を示すアイコンの隣に表示されます。変更されたデータ値のアイコンは水色(ターコイズ)の鉛筆があります。新規データ値のアイコンは右上に赤い星印と共に表示されます。ターゲット システムから削除されるレジストリ値のアイコンは、赤い X 印が付いています。以下のサンプルアイコンと説明を参照してください。

テーブル 260 • QuickPatch プロジェクトの [レジストリ] ビューのアイコン

アイコン	説明
	このアイコンは、QuickPatch が適用されるときターゲット システムに追加される新規 DWORD 値を示します。
	このアイコンは、QuickPatch が適用されるときターゲット システムで更新される変更済みの DWORD 値を示します。
	このアイコンは、以前のインストールからターゲット システム上にある既存 DWORD 値を示します。
	このアイコンは、ターゲット システム上に存在し、QuickPatch プロジェクトが適用されるとき削除される DWORD 値を示します。

InstallShield 前提条件エディター リファレンス

プロジェクト: *次のプロジェクト タイプは、InstallShield 前提条件のサポートを含みます:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

これら全てのプロジェクト タイプは、InstallShield 前提条件タイプのセットアップ前提条件のサポートを含みます。基本の MSI プロジェクトは、機能前提条件タイプのサポートを含みます。

InstallShield には、InstallShield 前提条件を、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、パッケージとして含めることができるサポートも含まれています。詳細については、「InstallShield前提条件 (.prq) をアドバンスト UI またはスイート/アドバンスト UI プロジェクトに含める」を参照してください。

InstallShield 前提条件 は、製品が必要とする製品、またはテクノロジ フレームワークのためのインストールです。InstallShield で提供されている InstallShield 前提条件の一例として、Java Runtime Environment (JRE) および SQL Server Express Edition があります。任意の既存の InstallShield 前提条件をインストール プロジェクトに追加して、その設定の多くを構成することができます。また、独自の InstallShield 前提条件を作成して、それをプロジェクトに追加することもできます。

タスク ***InstallShield 前提条件エディターを開くには、以下の手順に従います:***

[ツール] メニューで **[前提条件エディター]** をクリックします。

タスク ***InstallShield 前提条件エディターに既に存在する前提条件を開くには、以下の手順に従います:***

1. **[ファイル]** メニューで、**[開く]** をクリックします。**[開く]** ダイアログ ボックスが開きます。
2. ファイルを選択して **[開く]** をクリックします。

次のタブは、InstallShield 前提条件エディターに関連付けられています。

- プロパティ
- Windows パッケージ マネージャー
- 条件
- 含めるファイル
- 実行するアプリケーション
- 動作
- 依存関係

[プロパティ] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件ファイル エディターの [プロパティ] タブで、InstallShield 前提条件についての一般情報を設定します。

テーブル 1 ・ [プロパティ] タブの設定

設定	説明
InstallShield 前提条件の一意の ID	InstallShield 前提条件の一意のファイル識別子を入力するか、デフォルト値のままにしておきます。これは前提条件アプリケーションの名前、コンポーネント、または GUID 名前、どちらでも構いません。 ***メモ:*** *新しい InstallShield 前提条件を作成するために InstallShield 前提条件ファイル エディターを開くたびに、新しい GUID が生成され、このボックスにリストされます。*
前提条件ファイルをダウンロードする際、.prq をダウンロードするための別のロケーション	必要に応じ、.prq ファイルの代わりの URL を入力します。例: **http://www.mywebsite.com/MyPrq.prq** 詳細については、「.prq ファイルの代替 URL を指定する」を参照してください。 ***メモ:*** *InstallShield 前提条件は、FTP URL をサポートしません。*
説明	前提条件ファイルの概要を入力します。この説明は [再配布可能ファイル] ビューで前提条件ファイルを選択したときに、概要タイトルの下に表示されます。

[Windows パッケージ マネージャー] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*

- *InstallScript MSI*

InstallShield 前提条件エディターの [Windows Package Manager] タブを使って、製品に前提条件としてインストールしたい Windows Package Manager (WPM) パッケージを指定することができます。

Windows Package Manager は、インストール、アップグレード、構成、および Microsoft パブリック リポジトリからソフトウェアを使用する処理を自動化するためのシステム、または一連のツールです。Window Package Manager クライアント winget ツールは、レポジトリ内のパッケージを検索して、そのパッケージをパッケージ マニフェスト情報に基づいてインストールします。

メモ: *Windows Package Manger 前提条件を、アドバンスト UI および スイート/アドバンスト UI プロジェクトでインポートすることはできません。*

WPM 前提条件のランタイムは、Windows 10 以降のワークステーションでサポートされています。その他のオペレーティング システムの場合、WPM 前提条件はランタイム中にスキップされます。

テーブル 2 · [Windows Package Manager] タブにある設定

設定	説明
Windows Package Manager からインストール	Windows Package Manager から前提条件をインストールする場合は、これを選択してください。このチェックボックスを選択すると、[Windows Package Manager] および [動作] 以外の、Windows Package Manager 前提条件に適用されないその他すべてのタブが使用できなくなります。また、[動作] タブの下にある "再起動" 設定を無効になります。 ***メモ:*** *メモ: 管理者権限が必要な Windows Package Manager 前提条件には、[動作] タブの下にある "管理者権限" 設定の構成が必要な場合があります。*
パッケージ ID	パッケージ ID を取得するには、Microsoft リポジトリのパッケージ マニュフェストを検索するか、Windows Package Manager クライアント winget を **search** または **show** オプションと共に実行してください。
バージョン	Windows Package Manager からインストールするパッケージのバージョンを指定してください。最新版か、特定のバージョンを指定します。 デフォルトで、Windows Package Manager クライアント winget はレポジトリで使用可能な最新のパッケージをインストールします。WPM リポジトリで複数のバージョンが使用可能で、特定のバージョンをインストールしたい場合は、Windows Package Manager リポジトリで使用可能な特定のバージョンを指定することができます。
追加のパラメーター	**Windows パッケージ マネージャー**からパッケージをインストール中に渡す、任意の追加パラメーターを指定することが出来ます。 たとえば **—accept-source-agreements** パラメータを使って、ソース使用許諾契約書に同意し、前提条件が問題なく正しくインストールされるようにします。

WPM パッケージのインストール中、InstallShield は WPM クライアント winget ツールがターゲット マシン上で使用可能かどうかをチェックします。Winget クライアント ツールが使用できない場合、InstallShield はターゲット マシンに Windows Package Manager 前提条件をインストールします。そして、ビルド中に InstallShield は Settings.xml の設定に基づいて Windows Package Manager.prq をバンドルします。Windows Package Manager.prq のコンテキスト メニューから **[選択したアイテムをダウンロードする…]** オプションを選択して、**[再配布可能ファイル]** ビューから Windows Package Manager.prq ファイルをダウンロードする必要があります。このファイルをダウンロードしなかった場合、前提条件ファイルの場所として **[ソースメディアからコピーする]** または **[Setup.exe から抽出する]**. を選択したときにビルドが失敗する可能性があります。

InstallShield と共にインストールされている Settings.xml ファイルを検出します。Settings.xml は、使用している InstallShield の言語バージョンに応じて、次のいずれかの場所にインストールされています:

- **英語**—*InstallShield Program Files フォルダー*¥Support¥0409
- **日本語**—*InstallShield Program Files フォルダー*¥Support¥0411

<WPMRedistributable> 要素とその子要素を検索します。デフォルトでは次のように表示されています:

<WPMRedistributable Name="Windows Package Manager.prq" />

<WPMRedistributable Name=""> が空白の場合、InstallShield はビルド中に Windows Package Manager.prq をバンドルせず、実行時にインストールを行いません。それでも、WPM クライアント winget ツールがターゲットマシンで使用可能な状態にしてください。これが使用できない場合、実行時に WPM パッケージのインストールがスキップされてしまします。

InstallShield は、InstallShield サーバーの最新の WPM winget クライアント ツールで頻繁に **Windows Package Manager** 前提条件を更新します。**"ビルドの場所"** として **[Web からダウンロードする]** を使用して、実行時にターゲット マシン上で最新の WPM winget クライアント ツールが更新されるようにしてください。

[条件] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件ファイルの [条件] タブを利用して、ターゲット マシンに InstallShield 前提条件がインストールされているかどうかを判断するインストール条件を定義することができます。これを行わなかった場合、ターゲット システムは InstallShield 前提条件が適切にインストールされなかったと見なして作動するため、問題が発生します。また、オペレーティング システム、レジストリ、またはファイル要件を指定するインストール条件を作成することも可能です。

実行時に条件が True と評価すると、条件が満たされます。InstallShield 前提条件は、以下が当てはまる場合にターゲット マシンにインストールされます:

- ターゲット マシンがすべてのオペレーティング システム条件、および [条件] タブにリストされているその他すべての条件を満たしている。
- 機能前提条件のみ (つまり、メイン インストールに含まれる 1 つまたは複数の機能に関連付けられている InstallShield 前提条件) — 機能前提条件を含む機能は、必ずインストールされなくてはなりません。したがっ

て、機能の条件がターゲット システム上で満たされていない場合、またはエンド ユーザーが機能のインストールを行わないことを選択した場合、その機能はインストールされません。その結果、インストールされる別の機能にも機能前提条件が関連付けられていない限り、関連付けられた機能前提条件はいずれもインストールされません。

メモ: *Windows サービス パックなどの前提条件のインストールがサポートされていないため、オペレーティング システム バージョン情報に変更がないものと想定されます。したがって、オペレーティング システム条件は前提条件が正しくインストールされているかどうかを検証する際に考慮されません。*

このタブの [追加] をクリックすると、[前提条件] ダイアログ ボックスが開きます。ダイアログは、[条件] タブで既存の条件を選択し、[変更] ボタンをクリックしたときも開きます。

[含めるファイル] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

新しい InstallShield 前提条件を作成する時、InstallShield 前提条件と共に含む必要があるインストール ファイルを指定しなくてはなりません。既存する InstallShield 前提条件に必要なファイルのリストを変更することもできます。InstallShield 前提条件エディターの [含めるファイル] タブでファイルを指定します。

このタブの [追加] ボタンまたは [複数ファイルの追加] ボタンをクリックして InstallShield 前提条件ファイルを指定する、または [変更] ボタンをクリックしてそのファイルを変更すると、[新規ファイル] ダイアログ ボックスが開きます。

[ファイル URL の設定] ボタンをクリックすると、**[ファイル URL の設定]** ダイアログ ボックスが開きます。

メモ: *InstallShield 前提条件は、FTP URL をサポートしません。*

[実行するアプリケーション] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件エディターの [実行するアプリケーション] タブでは、ターゲット マシン上への InstallShield 前提条件のインストール方法を指定することができます。

テーブル 3 · [実行するアプリケーション] タブの設定

設定	説明
起動するアプリケーションを指定する	InstallShield 前提条件がインストールされている場合にターゲット マシンで起動するファイル 、通常 Setup.exe セットアップランチャーまたは .msi パッケージまたは App パッケージ (msix、msibundle、appx、または appxbundle) を選択します。[含めるファイル] タブで指定されたファイルのみがこの一覧に含まれます。 ***プロジェクト·** .msi ファイルを指定し、[動作] タブで進行状況の表示を指定した場合、Setup.exe セットアップ ランチャーは進行状況メッセージをキャプチャして、実行時に MsiExec.exe の代わりに Windows Installer API を使って .msi パッケージを起動します。* その他の種類のファイルを指定した場合、または進行状況が表示されない .msi ファイルを指定した場合、Setup.exe セットアップランチャーは実行時に Open 動詞 (.msi と .exe ファイルの場合)、またはデフォルトの動詞 (その他の種類のファイル) のどちらかを使ってファイルを実行します。 ***メモ·** ドロップダウン ボックスから App パッケージ (msix、msixbundle、appx、または appxbundle) を選択すると、App パッケージのコンテンツに基づいて [条件] タブの下に 2 つの条件が追加されます。App パッケージの ID と発行元の詳細に基づいて App パッケージを検出する条件、および Windows オペレーティング システムバージョンおよびビルド番号をチェックするもう 1 つの条件。* App パッケージの場合、**[実行するアプリケーション]** タブの下にある残りの構成は適用しません。
Windows Installer エンジンおよび/または .NET Framework を先にインストールする必要がある。	Windows Installer エンジンまたは .NET Framework (Dotnetfx.exe)、あるいはその両方を、この InstallShield 前提条件がインストールされる前にインストールする必要がある場合、このチェック ボックスを選択します。 ***メモ·** このチェック ボックスを選択しても Windows Installer エンジンまたは .NET Framework をインストールに追加しません。これらがインストールに含まれている場合、InstallShield 前提条件がインストールされる前にこれらをインストールしなくてはならないことを指定するだけです。Windows Installer エンジンまたは .NET Framework をインストールに含める場合、それらをプロジェクトに追加する必要があります。詳しくは、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」または「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」をご覧ください。*

テーブル 3 - [実行するアプリケーション] タブの設定 (続き.)

設定	説明
アプリケーションのコマンドラインを指定する	必要に応じて、[アプリケーションのコマンドラインを指定する] ボックスで **[起動するアプリケーションを指定する]** リストで選択したファイルのコマンドラインを入力します。このボックスにファイル名を含めないでください。 指定可能なコマンドライン パラメーターについての詳細は、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。 ***メモ-*** *前提条件が非表示として構成されている場合 (つまり、前提条件がインストールの必要がある前提条件の一部としてセットアップ前提条件のランタイム ダイアログに含まれていない場合)、標準コマンドライン パラメーターではなく、サイレント コマンドライン パラメーターを使って前提条件が起動されます。したがって、[動作] タブにある **[前提条件をインストール一覧に表示しない]** チェック ボックスが選択される可能性がある場合は、**[セットアップがサイレントモードで実行中にアプリケーションのコマンドラインを指定する]** ボックスにコマンドライン パラメーターを指定してください。*
セットアップがサイレントモードで実行されているときのアプリケーションのコマンドラインを指定する	必要に応じ、適切なコマンドラインを入力します。このボックスにファイル名を含めないでください。 指定可能なコマンドライン パラメーターについての詳細は、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。 ***メモ-*** *InstallShield 前提条件を含むインストールを /s コマンドライン パラメーターを使って起動すると、前提条件のインストールは自動的にサイレントで実行されません。必要に応じて、[実行するアプリケーション] タブの "**セットアップがサイレントモードで実行中にアプリケーションのコマンドラインを指定する**" 設定で、InstallShield 前提条件用の有効なサイレント コマンドライン パラメーターを指定してください。*
再起動が必要な場合、アプリケーションが戻す戻りコード(十進法) で指定する	選択された InstallShield 前提条件となるアプリケーションがインストール後にターゲットマシンの再起動を要求する場合、このボックスに戻りコードを入力します。 ***ヒント-*** *複数のリターンコードが存在する場合、各コードはカンマで区切ってリストしてください。* InstallShield 前提条件として起動するファイルの戻りコードが分からない場合は、ファイルの作成者に問い合わせてください。 再起動が必要な InstallShield 前提条件の詳細は、「InstallShield 前提条件のインストールでターゲットマシンを再起動する」を参照してください。

[動作] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

InstallShield 前提条件エディターの [動作] タブでは、特定の状況下における動作を指定することができます。

テーブル 4 : [動作] タブの設定

設定	説明
前提条件は管理者権限を必要とする	管理者権限が InstallShield 前提条件に必要な場合、または InstallShield 前提条件がマシンごとにインストールされる必要がある場合、このチェック ボックスを選択します。このチェック ボックスはデフォルトで選択されます。 ***メモ:*** *管理者権限が必要な InstallShield 前提条件では、Windows Vista 以降のシステムでインストールが実行される前に、エンドユーザーからの資格情報または同意 (コンセント) が必要な場合があります。詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。*
ユーザーは前提条件をオプションでスキップすることができる	エンド ユーザーが InstallShield 前提条件をインストールするかどうかを選択するためのメッセージ ボックスを表示する場合は、このチェック ボックスを選択します。 エンド ユーザーが InstallShield 前提条件のインストールの要否を選択できないようにするには、このチェック ボックスをクリアします。

テーブル 4 · [動作] タブの設定 (続き.)

設定	説明
前提条件をインストール一覧に表示しない	ターゲット システムに 1 つまたは複数のセットアップ前提条件のインストールが必要な場合、実行時、メイン インストールが実行する前にセットアップ前提条件ダイアログが表示されます。エンド ユーザーがこのダイアログで [インストール] ボタンをクリックすると、セットアップ前提条件がインストールされます。 そのダイアログで表示されるセットアップ前提条件の一覧に現在のセットアップ前提条件を含むには、このチェック ボックスをクリアします。 一覧に現在のセットアップ前提条件を含まない場合は、このチェック ボックスを選択します。セットアップ前提条件はセットアップ前提条件一覧には表示されませんが、必要に応じてターゲット システムにインストールされます。 ***メモ· [前提条件をインストール一覧に表示しない]*** *チェック ボックスが選択またはクリアされているかに関わらず、機能前提条件が実行時にセットアップ前提条件ダイアログに表示されることはありません。つまり、InstallShield 前提条件をプロジェクトに追加して、それを機能に関連付けた場合、実行時、メイン インストールが実行する前に表示されるセットアップ前提条件ダイアログで、機能前提条件は一覧に表示されません。* *セットアップ前提条件、または機能前提条件が非表示としてマークされている場合、標準のコマンドライン パラメーターではなく、サイレント コマンドライン パラメーターを使って起動されます。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。*

テーブル 4 - [動作] タブの設定 (続き.)

設定	説明
前提条件のウィンドウに進行状況を表示する (未加工の MSI ファイルのみ)	実行時に、現在の InstallShield 前提条件のインストール進行状況を表示するには、このチェック ボックスを選択します。InstallShield 前提条件のインストールが .msi ファイルのときだけ進行状況の表示が可能です。InstallShield 前提条件のインストールによって起動されるファイルが Setup.exe ファイルの場合、進行状況を表示することはできません。さらに、InstallShield 前提条件を含むインストールは基本の MSI インストールまたは InstallScript MSI インストールでなくてはなりません。 InstallShield 前提条件のインストールが Setup.exe ファイルの場合、この設定は無視されます。また、InstallShield 前提条件を含むインストールが InstallScript インストールの場合、この設定は無視されます。 ***メモ:*** *進行状況を表示するためにこのチェック ボックスを選択した場合、一部のコマンドライン パラメーターのみがサポートされます。コマンドライン パラメーターは、InstallShield 前提条件エディターの [実行するアプリケーション] タブで指定されます。詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。* *前提条件の進行状況の表示を指定すると、デフォルトで、前提条件の .msi パッケージのユーザー インターフェイスは表示されませんので、ご注意ください。この動作をオーバーライドするには、[実行するアプリケーション] タブで、コマンドライン パラメーターとして /qf を指定できます。*

テーブル 4 • [動作] タブの設定 (続き.)

設定	説明
前提条件をインストールした後も、その前提条件のインストールがまだ条件で要求された場合:	InstallShield 前提条件のインストールを実行済みであるのにも関わらず、InstallShield 前提条件のインストールが必要であると条件が表示される場合、次のどれかが当てはまる可能性があります。 • InstallShield 前提条件が正しくインストールされなかった。 • InstallShield 前提条件に不正確な条件が指定されていた。 たとえば、ターゲット マシンに特定のファイルが存在しない場合に InstallShield 前提条件をインストールする必要があると条件付けたとします。InstallShield 前提条件がインストールされた後にもそのファイルが不足している場合、その条件は適切ではなかったと考えられます。 InstallShield 前提条件のインストールが必要であると条件が表示し続けた場合の動作を指定してください。選択可能なオプションは、以下のとおりです。 • **セットアップを終了する** — アプリケーションをインストールしない場合は、このオプションを選択します。 • **セットアップを続行するかどうかを確認する** — エンド ユーザーに対してメインのインストールを続行するかどうかを指定するようにプロンプトするメッセージ ボックスを表示する場合、このオプションを選択します。 • **セットアップを続行する** — 基本的に、InstallShield 前提条件の満たされていない条件を無視して、次の InstallShield 前提条件がある場合はそのインストールを続ける、またはメインのインストールを続行する場合は、このオプションを選択します。

テーブル 4 • [動作] タブの設定 (続き.)

設定	説明
前提条件が再起動を必要とする場合	「InstallShield 前提条件のインストールでターゲットマシンを再起動する」でも説明されているように、InstallShield 前提条件をインストールするためにターゲット マシンを再起動する必要がある場合があります。 InstallShield 前提条件が、ターゲット マシンの再起動を必要とする場合の動作を指定します。 • **終了し、再起動時に再開** – InstallShield 前提条件のインストールを終了し、InstallShield 前提条件がターゲット マシンを再起動してインストールを再開する場合は、このオプションを選択します。 • **記録する。マシンが再起動された場合、再開を失敗して、インストールの後に再起動する** – 必要に応じてターゲット マシンが再起動することを記録し、メインのインストールの終わりに再起動をスケジュールし、InstallShield 前提条件のインストールのあとでターゲット マシンが再起動した場合終了するようにする場合、このオプションを選択します。したがって、再起動が必要であるように見える場合で、メインのインストールが終了するまで (または次の InstallShield 前提条件が再起動をトリガーするまで) 延期したいときは、このオプションを選択します。 実行時に、再起動がまだ保留中のとき、メイン インストールの Windows Installer プロパティ ISSCHEDULEREBOOT の値は 1 に設定されています。 • **無視する。ただし、マシンが再起動されるときインストールの継続を中止する** – インストールを継続するが、ターゲット マシンが再起動されるときに中止する場合は、このオプションを選択します。つまり再起動が必要であると検出されたが、これをスキップするには、このオプション を選択します。 • **再起動が検出されなかったときのみユーザーにプロンプトする。ただし、マシンは常に再起動し、再起動時に再開** – ターゲット マシンが再起動を必要としないと検出された場合のみ、ターゲット マシンの再起動をエンド ユーザーにプロンプトする場合は、このオプションを選択します。エンドユーザーがこれを承諾した場合にターゲットマシンは再起動され、インストールが続行します。 • **何も検出されなかったときもユーザーにマシンを再起動するようにプロンプトし、再起動時に再開** – ターゲット マシンが再起動を必要としないと検出された場合でも、ターゲット マシンの再起動をエンド ユーザーにプロンプトする場合は、このオプションを選択します。エンドユーザーがこれを承諾した場合にターゲットマシンは再起動され、インストールが続行します。 • **マシンを再起動し、再起動時に再開** – ターゲット マシンを再起動してインストールを継続する場合は、このオプションを選択します。

[依存関係] タブ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

新しい InstallShield 前提条件を作成する、または既存の InstallShield 前提条件を変更するとき、この InstallShield 前提条件が依存する他の 前提条件 (.prq ファイル) を指定することができます。依存関係は、InstallShield 前提条件エディターの依存関係タブで指定します。

このタブの [追加] して依存関係を追加すると、[新しい依存関係] ダイアログ ボックスが開きます。

エラーと警告

InstallShield ヘルプ ライブラリの次のトピックは、インストール作成作業中に起こる可能性のあるエラーおよび警告についての情報を提供します。

テーブル 1 ・ エラーおよび警告トピックのリスト

トピック	説明
ビルド エラーと警告	ビルド処理中に発生する可能性があるエラーと警告をリストします。
メディア ビルド エラーと警告	InstallScript コンパイラ エラーと警告をリストします。
MSI/MSM 変換エラー	MSI/MSM オープン ウィザードを使ってインストール パッケージ (.msi ファイル) またはマージ モジュール (.msm ファイル) を InstallShield プロジェクト (.ism ファイル) へ変換する際に発生する可能性があるエラーをリストします。
Windows Installer 実行時のエラー	実行時に発生する可能性がある Windows Installer エラーをリストします。このエラーは、一部の種類の機能をサポートするために InstallShield プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションに関連しています。
Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト)	InstallScript インストールで Setup.exe の実行時に発生する可能性があるエラーをリストします。
Setup.exe 戻り値および実行時のエラー (基本の MSI および InstallScript MSI プロジェクト)	基本の MSI インストールおよび InstallScript MSI インストールで Setup.exe の実行時に発生する可能性があるエラーをリストします。
Setup.exe 戻り値および実行時のエラー (アドバンスト UI およびスイート/アドバンスト UI)	アドバンスト UI またはスイート/アドバンスト UI の Setup.exe インストールが実行されるときに発生する可能性のあるリターン コード、エラー、およびログ記録されたステータス コードをリストします。
Visual Studio プロジェクトのインポート エラーと警告	次のいずれかの処理を行ったときに発生する可能性のあるエラーと警告をリストします: • Visual Studio セットアップ プロジェクト (.vdproj) を InstallShield 基本の MSI プロジェクト (.ism) に変換。 • Visual Studio マージ モジュール プロジェクト (.vdproj) を InstallShield マージ モジュール プロジェクト (.ism) に変換。 • Visual Studio セットアップまたはマージ モジュール プロジェクト (.vdproj) を InstallShield 基本の MSI またはマージ モジュール プロジェクト (.ism) にインポート。
アップグレード エラー (InstallShield Professional からのアップグレード)	InstallShield Professional を使って作成したプロジェクトをアップグレードしたときに起こる可能性のあるエラーをリストします。

テーブル 1 • エラーおよび警告トピックのリスト (続き.)

トピック	説明
アップグレードの警告 (InstallShield Professional からのアップグレード)	InstallShield Professional を使って作成したプロジェクトをアップグレードしたときに起こる可能性のある警告をリストします。
アップグレード エラーと警告 (InstallShield—Windows Installer Edition からのアップグレード)	InstallShield—Windows Installer Edition を使って作成したプロジェクト (.ism file) をアップグレードしたときに起こる可能性のあるエラーと警告をリストします。
Windows Installer ランタイムエラーの HRESULT 値	Windows Installer エラー 1904 および 1905 と共に提供される InstallShield カスタム HRESULT コードについて説明します。
DIFxAPI エラー (InstallScript プロジェクト)	DIFx ドライバー関数が呼び出されたときに戻される可能性のある DIFxAPI エラーそれぞれについて説明します。
"文字列 PRODUCT_NAME が文字列テーブルで見つかりません" エラー	このエラーが発生する状況、および問題の修正手順について説明します。
InstallScript エラー情報	InstallScript エラーおよび警告について説明します。
仮想化変換のエラーと警告	仮想アプリケーションを作成するとき、ビルド プロセス中に発生する可能性のあるエラーと警告をリストします。

エラーおよび警告についての詳細記事は、ナレッジ ベース に掲載されています。

Windows Installer エラー メッセージ (Windows Installer ヘルプ ライブラリ)

ビルド エラーと警告

次のテーブルは、ビルド処理中に発生する可能性のあるエラー及び警告の一覧です。

メモ: *InstallShield のほとんどのビルド エラーと警告についての詳細 (解決策を含む) は、ナレッジ ベースを参照してください。*

仮想アプリケーションのビルド中に発生する可能性があるエラーと警告の解決法については、「仮想化変換のエラーと警告」を参照してください。

テーブル 2 • ビルド エラーと警告

エラー/警告番号	メッセージ	トラブルシューティング情報
-1031	32 ビットのオペレーティング システム上で 64 ビット署名を使用することはません。	64 ビット署名では、64 ビットのオペレーティング システムを使用してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-32000	ビルドがユーザーによってキャンセルされました。	このエラーが発生するのは、ビルド プロセスの途中でビルドが終了した場合のみです。
9571	ビジュアル資産の横長タイル イメージを設定する必要があります : '%1'	ビジュアル資産に大きいタイル イメージが指定されている場合、パッケージをインストールするためには、横長タイル イメージを指定する必要があります。
9570	バックグラウンド タスク宣言に、少なくとも 1 つ以上のサポートされているタスク タイプを選択する必要があります。	バックグラウンド タスクの宣言には、1 つ以上のサポートされているタスクを選択し、[宣言] ビューからバックグラウンド タスクの宣言に [サポートされているタスク] を追加する必要があります。
9569	少なくとも 1 つ以上のサポートされているファイル タイプまたはデータ形式が必要です。共有ターゲット宣言で [すべてのファイル タイプをサポート] または 1 つ以上のファイル タイプまたはデータ形式を指定して明示的に追加してください。	[共有ターゲットの宣言] には少なくとも 1 つ以上のサポート対象ファイル タイプまたはデータを指定しなくてはなりません。共有ターゲット宣言で [すべてのファイル タイプをサポート] または 1 つ以上のファイル タイプまたはデータ形式を指定して明示的に追加してください。
9568	指定されたマネージド URL '%1' は、[Web アカウント プロバイダー宣言] で指定された ベース URL と重複しています。	マネージド URL の 1 つが、[Web アカウント プロバイダー宣言] で指定されたベース URL と重複しており、[Web アカウント プロバイダー宣言] で一意のマネージド URL を指定する必要があります。
9567	エントリ ポイント、実行可能ファイル、またはランタイム型が予定プロバイダーの宣言で指定されている場合、開始ページ '%1' を指定することはできません。	[予約プロバイダー宣言]で [開始ページ] が指定されている場合、[エントリ ポイント]、[実行可能ファイル]、または [ランタイム タイプ] を指定することはできません。[起動アクション] で [エントリ ポイント] または [スタートページ] エントリのどちらかを指定してください。
9566	宣言要素に定義されている実行可能ファイル '%1' がパッケージ内に存在しません。	宣言に指定されているカスタム実行可能ファイルがパッケージ内で見つからなかった場合、正しい実行可能ファイルで再定義するか、実行可能ファイルをパッケージに追加する必要があります。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
9565	ターゲット実行可能ファイル 'アプリケーション名' は実行時に昇格が必要です。デスクトップ ブリッジは、管理者権限または昇格された権限を必要とするパッケージ済み実行可能ファイルをサポートしません。	アプリケーションが UIAccess=true または level="highestAvailable|requireAdministrator" を UAC マニフェストの requestedExecutionLevel 要素内で指定する場合、UWP への変換は現在サポートされていません。 パッケージマニフェスト内の次のアプリ機能のフラグを設定すると、アプリケーションが昇格を要求できます。 <rescap:Capability Name="allowElevation"/> このフラグを設定する場合、InstallShield [ダイレクト エディター] ビューの 'ISVirtualPackage' テーブルに移動して、次のエントリを作成してください。 区切り記号として空白スペースを使って複数のアプリ機能を指定できます。 • **Name**: AppxCapabilityList • **Value**: rescap:allowElevation
9564	モジュール [COM モジュール名] には COM データが含まれています。この COM サーバーは、この AppX パッケージの外部ではアクティベートすることができません。	指定された COM サーバーは、そのすべてのレジストリ エントリおよび拡張ポイントがそのプライベート ハイブ内にあり、システムに公開されないため、この AppX パッケージの外部でアクティベートすることができません。アプリケーションが独自の目的だけに COM を使用する場合、このエラーを無視してアプリケーションのプライベート ハイブ (Registry.dat) 内の COM エントリのみを使って、アプリをサポートすることができます。同じ Appx/MSIX パッケージ内のすべてのバイナリは、そのレジストリにアクセスすることができますが、システム上のその他のアプリは、アプリのプライベート ハイブ内を参照することができません。
9563	依存関係 [依存パッケージ名] からレジストリ データを統合できませんでした。	指定の依存パッケージから registry.dat ファイルの抽出に失敗したため、レジストリ データの結合中にエラーが発生しました。
9562	依存関係 [依存パッケージ名] からファイルを統合できませんでした。	MSIX/Appx パッケージをビルドするために依存パッケージを結合中にエラーが発生しました。 指定の依存パッケージがソースの場所に存在すること、およびパッケージが有効であることを確認してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
9561	NT サービス [サービス名] はサポートされていない方法で開始するように設定されています。	追加されたサービスには有効な開始タイプがありません。有効な開始タイプは、MANUAL/AUTO 開始です。このため、デフォルトで AUTO 開始タイプとなります。 追加したサービスを確認してください。 すべてのサービスにはパス、開始アカウント、種類、エラー制御、開始の種類などの適切なプロパティが必要です。
9560	NT サービス [サービス名] は、サポートされていないスタートアップ ユーザー アクション エラーの制御オプション (SERVICE_ERROR_CRITICAL) を使用しています。	追加されたサービスには適切なエラー制御 (NORMAL または IGNORE) がありません。 追加したサービスを確認してください。 すべてのサービスにはパス、開始アカウント、種類、エラー制御、開始の種類などの適切なプロパティが必要です。
9559	NT サービス [サービス名] は、サポートされていないスタートアップ アカウント[アカウント] を使用します。適切な規定アカウントが使用できないため、このサービスをスキップします。	追加されたサービスには、次のどれかの開始アカウントが含まれていません: • LocalSystem • NetworkService • LocalService 追加したサービスを確認してください。 すべてのサービスにはパス、開始アカウント、種類、エラー制御、開始の種類などの適切なプロパティが必要です。
9558	この MSI には [サービスの数] NT サービスが含まれていますが、パッケージにサーバー ターゲットが含まれていないため、ビルドされた UWP アプリ パッケージに NT サービスは含まれません。	MSI の NT サービスが有効化されていますが、UWP アプリ パッケージのサーバー ターゲット サポートが有効化されていません。 [サーバー拡張] を有効化し、次のバージョン詳細に更新して問題を解決することができます: リリース > リリース名 > Appx/MSIX > サーバー拡張を含める
9557	resources.pri を作成できません。resources.pri 無しでは、スケーリングされたロゴを使用することはできません。	システムに Windows 10 SDK がインストール済みかどうかを確認して、インストールされていない場合は、これをインストールします。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
9556	Windows 10 SDK が見つかりません。一部の機能は使用できません。	システムに Windows 10 SDK がインストール済みかどうかを確認して、インストールされていない場合は、これをインストールします。
9554	プロトコル ハンドラー [名前] には 'open' 動詞が含まれていません。	これは警告メッセージです。
9553	拡張子 [拡張子] のファイル タイプの関連付けに 'open' 動詞が含まれていませんが、別の動詞が含まれています。	このファイル関連付けで 'open' 動詞を追加します。
9552	Msix App ファイル エントリの書き込み中に予期しないエラーが発生しました。	MSIX パッケージにペイロード ファイル/フォルダーを追加中にエラーが発生しました。 ビルド時、システム上にソース ファイルが存在するかどうかを確認します。
9551	.Msix ライターの初期化中に予期しないエラーが発生しました。	たとえば出力場所に MSIX パッケージのフォルダーを作成中、またはアプリケーション パッケージのプロセッサ アーキテクチャ情報が不足しているなど、MSIX パッケージ ライターの初期化エラーが発生しました。
9513	[ファイル名 (行番号)] にスキーマ検証エラー	スキーマ ファイルに対して MSIX/Appx マニフェスト xml の検証中にエラーが発生しました。 必須の値やサポート対象の値など、マニフェスト エントリの構成については、https://docs.microsoft.com/en-us/uwp/schemas/ を参照してください。 MSIX プロジェクトの対応するビューで、値を訂正する必要があります。 たとえば、追加された宣言に指定された値を再チェックして、値がスキーマと一致するかどうかを確認します。
9512	ファイル [ファイル名] をパッケージに追加できません。	MSIX パッケージに指定のファイルを追加中にエラーが発生しました。指定のファイルがシステムに存在し、十分なディスク空き容量があることを確認します。
9510	Windows Installer パッケージをビルド中に警告が発生しました。	このエラーは、[パッチのデザイン] ビューの最新のイメージが以前のイメージを 1 つも含んでいない場合に発生します。最新のイメージはパッチを作成するのに以前のイメージを最低 1 つ含まなくてはなりません。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–7394	ビルドイベント – '% 1 ' が終了コード '% 2' で失敗しました。	このエラーは、**[イベント]** タブの **[ビルド前のイベント]**、**[ファイルの準備イベント]**、**[圧縮前のイベント]**、または **[ビルド後のイベント]** 設定で指定されたコマンドがビルド時間中に実行されなかった場合にのみ発生します。 この問題を解決するには、**[イベント]** タブのこれらの設定のすべてのコマンドが有効であることを確認し、必要に応じて更新します。
–7391	SQL スクリプトの生成中、レコードの取得処理中にエラーが発生しました。	このエラーは、マシンに ODBC ドライバー 17 バージョンが搭載されていない場合に発生します。このエラーを解決するは、ODBC ドライバー 17 バージョンをインストールしてみてください。
–7387	インストーラーで [アップデートの確認] が構成されているため、リリースの ”アップデート ダイアログのテキスト” を空白のまま残すことはできません。	このエラーは、スイート/アドバンスト UI プロジェクトで定期的なアップデートの構成中に、アプリケーションのアップデート ダイアログ ボックスにテキストが入力されていない場合に発生します。 このエラーを解決するには、**[メディア]** ビューの **[リリース]** エクスプローラーに表示される **[アプリのアップデートを確認する]** 領域の下で定義されている **”アップデート ダイアログのテキスト”** 設定でアプリケーションのアップデート ダイアログ ボックスのテキストを指定してください。
–7386	[アップデートの確認] が構成されているため、リリースの ” アップデート ダイアログのタイトル” 設定を空白のまま残すことはできません。	このエラーは、スイート/アドバンスト UI プロジェクトで定期的なアップデートの構成中に、アプリケーションのアップデート ダイアログ ボックスにタイトルが入力されていない場合に発生します。 このエラーを解決するには、**[メディア]** ビューの **[リリース]** エクスプローラーに表示される **[アプリのアップデートを確認する]** 領域の下で定義されている **”アップデート ダイアログのタイトル”** 設定でアプリケーションのアップデート ダイアログ ボックスのタイトルを指定してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7385	インストールで [アップデートの確認] が構成されているため、[一般情報] の "プログラムの追加と削除" 設定は [はい] に設定します。	このエラーは、アドバンスト UI または スイート/アドバンスト UI インストーラーでアップデーをの確認が構成されている場合に、アドバンスト UI または スイート/アドバンスト UI 製品のエントリーがターゲット システムのコントロール パネルにある [プログラムの追加と削除] で使用できなかった場合に発生します。 このエラーを解決するには、**[一般情報]** ビューの **[プログラムの追加と削除]** 領域で定義されている **"[プログラムの追加と削除] エントリの表示"** 設定を [はい] に設定して下さい。
-7382	[パッケージ] ビューのパッケージ %1 に対する "WPM パッケージ ID 値が空白です。有効なパッケージ ID を入力してください。	このエラーは、スイート/アドバンスト UI プロジェクトに有効な Windows パッケージ マネージャー (WPM) パッケージ ID が含まれていない時に発生します。 このエラーを解決するには、**[編成]** ビューの **[パッケージ]** エクスプローラーに表示される **[新しい WPM パッケージ]** の **[共通]** タブ内で、**[全般]** 領域の下に定義される **"WPM パッケージ ID"** 設定に有効な Windows パッケージ マネージャー (WPM) パッケージ ID を指定してください。 Windows パッケージ マネージャー (WPM) パッケージ ID を取得するには、Microsoft リポジトリのパッケージ マニュフェストを検索するか、Windows パッケージ マネージャー クライアント winget コマンドライン ツールを **search** または **show** オプションで実行してください。
-7355	文字列 %2 の値 %4 は、テーブル %1 列 %3 の検証基準を満たしていません。	この警告は、ローカライズされた文字列が文字列エディター テーブル内の列の検証基準を満たしていない時に発生します。この警告を解決するには、文字列エディター内のフラグされた値を更新してください。
-7354	文字列 %2 の値 %4 は、テーブル %1 列 %3 では使用できません。	このエラーは、ローカライズされた文字列が文字列エディター テーブル内の名前付き列に有効な値が含まれていないときに発生します。このエラーを解決するには、文字列エディター内のフラグされた値を更新してください。
-7350	ISWSEWrap テーブルをビルド中にエラーが発生しました。ファイル %1 が見つかりませんでした。	このエラーは、プロジェクトに WiseScript カスタム アクションが含まれているが、[カスタム アクションとシーケンス] ビューでそれに指定された実行可能ファイルが見つからなかった場合に発生します。このエラーを解決するには、このカスタム アクションの "WiseScript 実行可能ファイル" 設定のパスを変更します。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7348	ファイル %sを保存中にエラー 0x%08x が発生	このエラー メッセージで %s は、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでビルド時に保存する必要があるファイルの名前とパスを示します。このエラーを解決するには、そのファイルが使用可能であり、読み取り専用またはロックされていないかを確認してください。
-7347	InstallShield では、.spc および .pvk ファイルを使った署名がサポートされていません。プロジェクト内のデジタル証明書情報を .pfx 証明書または証明書ストアにある証明書への参照と入れ換えてください。	このエラーは、.spc および .pvk ファイルを使ったデジタル署名がサポートされている InstallShield の以前のバージョンで、これらのファイルをプロジェクトに追加していた場合、そのプロジェクトを InstallShield 2025 で開いてリリースまたはパッチをビルドしようとしたときに発生します。 このエラーを解決するには、.spc の参照をリリースまたはパッチ構成から参照を削除します。これを、.pfx 証明書または証明書ストアにある証明書への参照と入れ換えることができます。詳しくは、次を参照してください: • デジタル署名とセキュリティ • ビルド時にリリースとそのファイルにデジタル署名を行う • パッチ パッケージに署名する • QuickPatch パッケージに署名する
-7346	リリースに含まれるファイルが SHA-1 証明書を使って署名されています。Windows は、2016 年 1 月以降に SHA-1 証明書を使って署名されているファイルを信頼しません。	この警告は、ビルド時に SHA-1 証明書を使ってリリースにデジタル署名を行うようにプロジェクトが構成されている場合に発生します。 SHA-1 はセキュリティの脆弱性があるため、SHA-256 の使用が推奨されます。したがって、InstallShield プロジェクトに含まれる任意の SHA-1 証明書は、SHA-256 証明書と置き換えることが推奨されます。 InstallShield でリリースに署名するための SHA-1 証明書を SHA-256 証明書に置き換えるには、[リリース] ビューの [署名] タブを使って、現在の証明書への参照を SHA-256 証明書と置き換えます。 詳細については、「デジタル署名とセキュリティ」を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7337	機能 [1] に関連付けられたパッケージがありません。	スイート/アドバンスト UI プロジェクトまたはアドバンスト UI プロジェクトでリリースをビルドしたとき、プロジェクトにパッケージを含まない機能が含まれている場合、このビルド警告が表示されます。 このビルド警告を解決するには、少なくとも 1 つ以上のパッケージが機能と関連付けられていることを確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける」を参照してください。 ビルド警告で識別された機能に、あえてパッケージを含まない場合は、このビルド警告を無視することができます。
-7336	アドバンスト UI プロジェクトでは、イベントがサポートされていません。	InstallShield Professional Edition を使って、[イベント] ビューに 1 つ以上のアクションを含むスイート/アドバンスト UI プロジェクトを開くと、InstallShield はプロジェクトをアドバンスト UI プロジェクトとして開いて、ビルド時にこのビルド エラーを表示します。アドバンスト UI プロジェクトでは、イベントがサポートされていません。 このビルド エラーを解決するには、プロジェクトからアクションを削除するか、InstallShield Premier Edition を使ってプロジェクトを編集してからリリースをビルドしてください。 詳細については、「スイート/アドバンスト UI インストールの動作を拡張するアクションを使用する」を参照してください。
-7335	プロジェクトには未定義のアクション '%1' への参照が含まれています。	スイート/アドバンスト UI プロジェクトにアクションを追加し、プロジェクト内のパッケージにアクションを関連付けてから、.issuite プロジェクト ファイルをテキスト エディターで編集してプロジェクトからこのアクションの一部を削除した場合、ビルド時にこのエラーが発生します。 このエラーを解決するには、ビルド エラーで参照されたアクションを .issuite ファイルから削除してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7334	パッケージ '%1' には Windows の機能への参照が含まれていますが、このエディションの InstallShield ではサポートされていません。これらの参照は、ビルドに含まれません。	InstallShield Professional Edition を使って、1 つ以上の Windows の役割または機能を有効化するサポートを含むスイート/アドバンスト UI プロジェクトを開くと、InstallShield はプロジェクトをアドバンスト UI プロジェクトとして開いて、ビルド時にこのビルド エラーを表示します。アドバンスト UI プロジェクトは、Windows の役割または機能を有効化するサポートを含みません。 このビルド エラーを解決するには、プロジェクトから Windows の役割と機能を削除するか、InstallShield Premier Edition を使ってプロジェクトを編集してからリリースをビルドしてください。 詳細については、「スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する」を参照してください。
-7329	IA64 %1 %2 を x64 パッケージに含めています。	このビルド警告は、以下の条件が当てはまる場合に発生します: • x64 が "テンプレートの概要" 設定で指定されています。 • プロジェクトには、実行時にインストールがターゲット システムに転送できる Itanium 64 ビット ファイルが含まれています。 x64 ターゲット システムで Itanium 64 ビット ファイルを実行することはできません。 要件によっては、IA64 ファイルを x64 ファイルに置換した方が良い場合があります。その他の状況下では、このビルド警告を無視することができます。たとえば、コンポーネントが x64 システムにインストールされないように防ぐ条件がファイルのコンポーネントに含まれている場合、このビルド警告を無視することができます。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7328	x64 %1 %2 を IA64 パッケージに含めています。	このビルド警告は、以下の条件が当てはまる場合に発生します: • IA64 が "テンプレートの概要" 設定で指定されています。 • プロジェクトには、実行時にインストールがターゲット システムに転送できる x64 ビット ファイルが含まれています。 Itanium 64 ターゲット システムで x64 ビット ファイルを実行することはできません。 要件によっては、x64 ファイルを IA64 ファイルに置換した方が良い場合があります。その他の状況下では、このビルド警告を無視することができます。たとえば、コンポーネントが IA64 システムにインストールされないように防ぐ条件がファイルのコンポーネントに含まれている場合、このビルド警告を無視することができます。
-7327	64 ビットの %1 %2 が 32 ビット限定のパッケージに含まれています。	このビルド警告は、以下の条件が当てはまる場合に発生します: • Intel が "テンプレートの概要" 設定で指定されている。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトには、実行時にインストールがターゲット システムに転送できる x64 ビット ファイルが含まれています。 x86 ターゲット システムで x64 ファイルを実行することはできません。 要件によっては、x64 ファイルを x86 ファイルに置換した方が良い場合があります。その他の状況下では、このビルド警告を無視することができます。たとえば、コンポーネントが x86 システムにインストールされないように防ぐ条件がファイルのコンポーネントに含まれている場合、このビルド警告を無視することができます。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7326	32 ビットの %1 %2 が 64 ビット限定のパッケージに含まれています。	このビルド警告は、以下の条件が当てはまる場合に発生します: • x64 アーキテクチャが "テンプレートの概要" 設定で指定されている。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトには、実行時にインストールがターゲット システムに転送できる x86 ファイルが含まれています。 多くの場合、x64 ターゲット システム上で x86 ファイルを実行することができます。ただし、x64 システムが WOW64 (32-bit Windows-on-Windows) をサポートしない場合、x86 ファイルを実行することができないこともあります。 要件によっては、x86 ファイルを x64 ファイルに置換した方が良い場合があります。その他の状況下では、このビルド警告を無視することができます。たとえば、コンポーネントが x64 システムにインストールされないように防ぐ条件がファイルのコンポーネントに含まれている場合、このビルド警告を無視することができます。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
-7325	ファイル %1 のアーキテクチャを検証中に予期しないエラーが発生しました。	このビルド エラーは、アーキテクチャの検証を行うときにプロジェクト内のファイルが 32 ビットと 64 ビットのどちらであるのかを InstallShield が判断できない場合に発生します。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7324	%1 カスタム アクション %2 が、ネイティブ コード %3 の関数を呼び出せません。	このビルド エラーは、プロジェクトに標準 DLL ファイルのマネージ コードまたは .NET Core カスタム アクションが含まれている場合に発生します。このエラーを解決するには、カスタムアクションを DLL ファイルで関数を呼び出す標準の DLL カスタム アクションと置換します。そうでない場合、DLL ファイルを Visual Basic .NET または C# などのマネージ コードで書かれた .NET アセンブリで置換します。 詳細は、次を参照してください: • マネージ コード ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す. • .NET Core ベースのマネージ アセンブリ内のパブリック メソッドを呼び出す
-7323	%1 カスタム アクション %2 が、マネージ コード %3 の関数を呼び出せません。	このビルド エラーは、プロジェクトで 。NET アセンブリ ファイルに標準 DLL カスタム アクションが含まれている場合に発生します。このエラーを解決するには、カスタム アクションを .NET アセンブリで パブリック メソッドを呼び出す マネージ コード カスタム アクションと置換します。そうでない場合、.NET アセンブリを C++ などのネイティブ コードで書かれた標準 DLL で置換します。 詳細については、「標準 DLLI ファイルの関数を呼び出す」を参照してください。
-7322	IA64 の %1 カスタム アクション %2 を x64 パッケージで実行することはできません。	このビルド警告は、以下の条件が当てはまる場合に発生します: • x64 が ″テンプレートの概要″ 設定で指定されています。 • プロジェクトには Itanium 64 ビット システムをターゲットとするカスタム アクション ファイルが含まれています。 x64 ターゲット システムで IA64 カスタム アクションを実行することはできません。 詳細については、「″テンプレート概要″ プロパティを使用する」を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7321	x64 の %1 カスタム アクション %2 を IA64 パッケージで実行することはできません。	このビルド警告は、以下の条件が当てはまる場合に発生します: • IA64 が "テンプレートの概要" 設定で指定されています。 • プロジェクトには x64 ビット システムをターゲットとするカスタム アクション ファイルが含まれています。 IA64 ターゲット システムで x64 カスタム アクションを実行することはできません。 詳細については、「"テンプレート概要" プロパティを使用する」を参照してください。
-7320	64 ビットの %1 カスタム アクション %2 を 32 ビット限定のパッケージに含めることはできません。	このビルド エラーは、以下の条件が当てはまる場合に発生します: • Intel が "テンプレートの概要" 設定で指定されている。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトに 64 ビット カスタム アクション ファイルが含まれている。 64 ビット DLL ファイルを 32 ビット ターゲット システムで実行することはできません。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
-7319	32 ビットの %1 カスタム アクション %2 を 64 ビット限定のパッケージに含めることはできません。	このビルド エラーは、以下の条件が当てはまる場合に発生します: • x64 が "テンプレートの概要" 設定で指定されています。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトに 32 ビット カスタム アクション ファイルが含まれている。 32 ビット DLL ファイルを WOW64 (32-bit Windows-on-Windows) をサポートしない 64 ビット ターゲット システムで実行することはできません。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7318	InstallScript カスタム アクション %1 を 64 ビット限定のパッケージに含めることはできません。	このビルド エラーは、以下の条件が当てはまる場合に発生します: • x64 が "テンプレートの概要" 設定で指定されています。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトに InstallScript カスタム アクションが含まれている。 InstallScript カスタム アクションを WOW64 (32-bit Windows-on-Windows) をサポートしない 64 ビット ターゲット システムで実行することはできません。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
-7317	64 ビットの VBScript/JScript カスタム アクション %1 を 32 ビット限定のパッケージに含めることはできません。	このビルド エラーは、以下の条件が当てはまる場合に発生します: • Intel が "テンプレートの概要" 設定で指定されている。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトに 64 ビット VBScript または JScript カスタム アクション ファイルが含まれている。 64 ビット スクリプト ファイルを 32 ビット ターゲット システムで実行することはできません。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7316	32 ビットの VBScript/JScript カスタム アクション %1 を 64 ビット限定のパッケージに含めることはできません。	このビルド エラーは、以下の条件が当てはまる場合に発生します: • x64 が "テンプレートの概要" 設定で指定されています。 • "プラットフォームの検証" 設定に [厳密な検証をする] が選択されている。 • プロジェクトに 32 ビット VBScript または JScript カスタム アクション ファイルが含まれている。 32 ビット スクリプト ファイルを WOW64 (32-bit Windows-on-Windows) をサポートしない 64 ビット ターゲット システムで実行することはできません。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
-7315	カスタム アクション %1 のアーキテクチャを検証中に予期しないエラーが発生しました。	このビルド エラーは、アーキテクチャの検証を行うときにプロジェクト内のカスタム アクション ファイルが 32 ビットと 64 ビットのどちらであるのかを InstallShield が判断できない場合に発生します。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
-7314	カスタム アクションのアーキテクチャを検証中に予期しないエラーが発生しました。	このビルド エラーは、アーキテクチャの検証を行うときにプロジェクト内のファイルが 32 ビットと 64 ビットのどちらであるのかを InstallShield が判断できない場合に発生します。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
-7313	アドバンスト UI プロジェクトでは、Windows アプリ パッケージはサポートされていません。	Windows アプリ パッケージを含むスイート/アドバンスト UI プロジェクトを InstallShield Professional Edition で開くと、アドバンスト UI プロジェクトとして開き、ビルド時にこのビルド エラーが表示されます。アドバンスト UI プロジェクトでは、Windows アプリ パッケージはサポートされていません。 このビルド エラーを解決するには、プロジェクトから Windows アプリ パッケージを削除するか、InstallShield Premier Edition を使ってプロジェクトを編集してからリリースをビルドしてください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7312	証明書ファイル '%1' が、パッケージ '%2' に選択されていますが、このパッケージのファイルと共に含まれていません。"証明書ファイル" 設定からこのファイルを削除するか、ファイルを Windows アプリ パッケージに追加してください。	Windows アプリ パッケージをサポートする Windows のバージョンは、パッケージのソースを信頼しない限り、サイドロード パッケージをインストールしません。たとえば Visual Studio で生成されたカスタム証明書を使用する場合、スイート/アドバンスト UI インストールではターゲット システムの証明書ストアにカスタム証明書を追加することで、Windows がパッケージのソースを信頼するように設定できます。スイート/アドバンスト UI インストールは、.cer ファイルをサイドロード パッケージに含めて、[パッケージ] ビューで Windows アプリ パッケージの "証明書ファイル" 設定にそれを指定した場合、この処理を行います。 .cer ファイルを指定して、プロジェクトにそのファイルを含めなかった場合、ビルド時にビルド警告が発生する場合があります。.cer ファイルを追加するには、[パッケージ] ビューで Windows アプリ パッケージを見つけて、ビュー内でそのノードを展開します。[パッケージ ファイル] ノードを選択してから右側のペインを右クリックして、[追加] をクリックします。追加するファイルを参照して選択します。
-7311	サイドロード アプリケーション パッケージ '%1' のマニフェスト ファイルをロードできません。	このビルド エラーは、サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールをビルドしようとしたときに、InstallShield が Windows アプリ パッケージのマニフェスト ファイルを読み取ることができない場合に発生します。これは、Windows XP または Windows Server 2003 システム上でビルドを行うときに発生する場合があります。サイドローディング Windows アプリ パッケージを含むスイート/アドバンスト UI インストールを作成およびビルドするには、InstallShield または Standalone Build が搭載されているマシン上に Windows Vista 以降または Windows Server 2008 以降が必要です。 このエラーを解決するには、[パッケージ] ビューから Windows アプリパッケージを削除するか、リリースを Windows Vista 以降または Windows Server 2008 以降でビルドします。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7308	スイート/アドバンスト UI インストールは ARM プロセッサ専用にビルドされたサイドローディング Windows アプリ パッケージをサポートしません。	このエラーは、スイート/アドバンスト プロジェクトでは、ARM プロセッサ アーキテクチャをターゲットとする Windows アプリ パッケージを含めたときに発生します。InstallShield には、インストールにこの種類のパッケージを含むサポートが含まれていません。 このエラーを解決するには、[パッケージ] ビューを使ってエラーメッセージにリストされているパッケージをプロジェクトから削除し、これを x86 または x64、あるいはニュートラルをターゲットとするパッケージと置換することを考慮してください。
-7295	アドバンスト UI プロジェクトでは、.exe プライマリ パッケージはサポートされていません。	InstallShield では、.exe パッケージを、スイート/アドバンスト UI (InstallShield Premier Edition で利用可能) に、プライマリ パッケージとして追加することができますが、アドバンスト UI (InstallShield Professional Edition で利用可能) には、プライマリ パッケージとして追加することはできません。 InstallShield Professional Edition で、Premier Edition で作成されたプロジェクトを開き、プロジェクトに .exe パッケージがプライマリ パッケージとして含まれていた場合、ビルド エラーが発生します。 このビルド エラーを解決するには、InstallShield Premier Edition を使用してリリースをビルドするか、プロジェクトの [パッケージ] ビューを使って、.exe パッケージをプロジェクトから削除します。 アドバンスト UI とスイート/アドバンスト UI プロジェクトの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7294	アドバンスト UI プロジェクトには、1 言語のみ含めることができます。	InstallShield Premier Edition で提供されているスイート/アドバンスト UI プロジェクトには、複数言語がサポートされていますが、InstallShield Professional Edition で提供されているアドバンスト UI プロジェクトでは、1 言語のみサポートされています。 InstallShield Professional Edition で、Premier Edition で作成されたプロジェクトを開き、プロジェクトに 1 つ以上の言語が含まれていた場合、ビルド エラーが発生します。 このビルド エラーを解決するには、InstallShield Premier Edition を使用してリリースをビルドするか、プロジェクトの [一般情報] ビューにある "セットアップ言語" 設定を使って、1 言語を残してすべて削除します。また、必要に応じて、[リリース] ビューの [ビルド] タブにある "UI 言語" 設定からも。1 言語以外のすべての言語を削除します。 アドバンスト UI とスイート/アドバンスト UI プロジェクトの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。
-7293	アドバンスト UI プロジェクトには、プライマリ パッケージを 1 つのみ含めることができます。	InstallShield Premier Edition で提供されているスイート/アドバンスト UI プロジェクトには、複数のプライマリ パッケージがサポートされていますが、InstallShield Professional Edition で提供されているアドバンスト UI プロジェクトでは、1 つのみサポートされています。 InstallShield Professional Edition で、Premier Edition で作成されたプロジェクトを開き、プロジェクトに 1 つ以上のプライマリ パッケージが含まれていた場合、ビルド エラーが発生します。 このビルド エラーを解決するには、InstallShield Premier Edition を使ってリリースをビルドするか、プロジェクトの [パッケージ] ビューから、1 つ以外のプライマリ パッケージをすべて削除します。 アドバンスト UI とスイート/アドバンスト UI プロジェクトの違い関する説明は、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」をご覧ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7292	アップデート URL が設定されていますが、リリースは、デジタル署名されるように構成されていません。	このビルド警告は、URL が、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [リリース] ビューにある [アップデート] タブの "アップデート URL" 設定で指定されていて、リリースがデジタル署名されるように構成されていないときに発生します。 この警告を解決するには、[リリース] ビューの [署名] タブで、リリースのデジタル署名情報を指定します。 アップデートの詳しい情報については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。
-7291	アップデート URL が設定されていますが、プロジェクトにファイルがソース メディアにあるパッケージが含まれています。	このビルド警告は、URL が、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [リリース] ビューにある [アップデート] タブの "アップデート URL" 設定で指定されていて、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内に、ソース メディアに格納されるように構成されているパッケージが 1 つ以上存在した場合に発生します。 ベースのアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーをビルドする場合、この警告は無視できます。 以前のベース セットアップ ランチャーがターゲット システムでっ実行された時にダウンロードされて起動される、アップデート アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーをビルドする場合、プロジェクトで、パッケージのランタイムの場所を、セットアップ ランチャーからの抽出か、Web からのダウンロードに変更します。アップデート セットアップ ランチャーは、ソース メディアに格納されているパッケージには依存できません。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトの特定のパッケージに対してランタイムの場所を指定する」を参照してください。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7256	ソフトウェア識別タグ ファイルにデジタル署名するためには、.NET Framework 3.5 のインストールが必要です。	プロジェクトにソフトウェア識別タグを含めるように構成した場合、ビルドするリリースに .pfx ファイルを使用してリリースにデジタル署名を行うと、InstallShield はビルド時に作成するタグにデジタル署名を行います。ビルド マシンに .NET Framework 3.5 がインストールされていない場合、このビルド警告が表示されます。 このビルド警告を解決するためには、ビルド マシンに .NET Framework 3.5 をインストールしてください。
-7255	ソフトウェア識別タグファイルのデジタル署名には、.pfx ファイルのみ使用できます。	プロジェクトにソフトウェア識別タグを含めるように構成した場合、ビルドするリリースに証明書ストアにある証明書を使ったデジタル署名を構成すると、InstallShield がこのビルド警告を生成します。 このビルド警告を解決するためには、証明書ストアにある証明書を .pfx ファイルに切り替えてください。デジタル署名付きのタグが不要な場合、この警告は無視できます。
-7246	パッケージ "%1" に、条件の設定の 1 つにアスタリスク (*) がプレースホルダーとして使用されている MSI パッケージ条件があります。このプレースホルダーは、適切な .msi パッケージが見つからなかったため、置き換えることができませんでした。	アドバンスト UI またはスイート/アドバンスト UI インストールで、条件で使用されているアスタリスクを、パッケージからの適切な情報 (製品コードは製品バージョンなど) で置き換えるには、パッケージがビルド時に存在している必要があります。 このエラーを解決するには、パッケージ ファイルがソースの場所にあることを確認します。ダイナミック ファイル リンクがパッケージに使用されている場合、ダイナミック リンクのフィルターによって、ビルドからパッケージ ファイルを除外されないことを確認してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–7244	プロジェクトに、プライマリ パッケージが含まれていません。必要に応じて、[パッケージ] ビューで、パッケージを追加し、″パッケージの種類″ 設定でプライマリを選択します。	このエラーは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、プライマリ パッケージとして構成されていないパッケージが含まれていないときに発生します。 プロジェクトにパッケージが含まれていない場合に、このエラーを解決するには、[パッケージ] ビューを使って、プロジェクトにパッケージを追加します。詳細については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」を参照してください。 プロジェクトに既に依存パッケージをあり、それをプライマリ パッケージにするには、[パッケージエ ビューの ″パッケージの種類″ 設定を使用します。詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。
–7243	[パッケージ] ビューのパッケージ '%1' に対する ″検出条件″ 設定が空です。この種類の条件は、ターゲット システム上にパッケージがインストール済みであるかどうかを評価します。	アドバンスト UI またはスイート/アドバンスト UI プロジェクトの .exe タイプのパッケージの場合、″検出条件″ 設定を空白のままに残すことはできません。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。
–7240	パッケージ [1] が、どの機能にも関連付けられていません。	アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューに含まれている各パッケージは、1 つ以上の機能に関連付ける必要があります。そうでない場合、ビルド時にパッケージがリリースに含まれません。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージを機能に関連付ける」を参照してください。
–7239	パッケージ '%2' の [パッケージ] ビューで、操作の ″ターゲット″ 設定で選択されたターゲット ファイル '%1' がプロジェクトで見つかりませんでした。	このエラーは、アドバンスト UI またはスイート/アドバンスト UI インストール内のあるパッケージのターゲット ファイルがビルド時に見つからなかったときに発生します。 このエラーを解決するには、ターゲット ファイルがソースの場所にあることを確認します。ダイナミック ファイル リンクがパッケージに使用されている場合、ダイナミック リンクのフィルターによって、ビルドからパッケージ ファイルを除外されないことを確認してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–7236	[一般情報] ビューの %1　設定の値が無効であるため、InstallShield は、ソフトウェア識別タグを作成できませんでした。以下の文字が含まれていないことを確認してください: ¥ / : * ? " < > |	InstallShield は、タグファイル（*TagCreatorID_UniqueID*.swidtag）の名前の一部として、入力済みの "一意な ID" および "タグ作成者 ID" 設定の値を使用します。したがって、入力する ID に、ファイル名には使えない文字は使用できません。 この問題を解決するためには、警告メッセージで通知された設定に有効な値を入力してください。 詳細については、「製品のソフトウェア識別タグを含める」を参照してください。
–7235	[一般情報] ビューの %1　設定が空白であるため、InstallShield は、ソフトウェア識別タグを作成できませんでした。	ソフトウェア識別タグを作成するには、[一般情報] ビューにあるいくつかの設定に値が必要です。この警告を解決するには、警告メッセージで通知された設定に適切な値を入力するか、または "ソフトウェア識別タグの使用" 設定で [いいえ] を選択します。 詳細については、「製品のソフトウェア識別タグを含める」を参照してください。
–7234	テンプレート ファイル Swidtag.xml を開くことができなかったため、InstallShield はソフトウェア識別タグを作成できませんでした。	InstallShield はタグ ファイルをビルドする際に、以下のいずれかの場所にインストールされている Swidtag.xml ファイルを使用します: *InstallShield Program Files フォルダー*¥Support¥0409 *InstallShield Program Files フォルダー*¥Support¥0411 この問題を解決するには、ロックが解除されている Swidtag.xml ファイルが存在することを確認してください。 詳細については、「製品のソフトウェア識別タグを含める」を参照してください。
–7233	[パッケージ] ビューでパッケージ %1 に無効な URL (%2) が指定されました。	このビルド エラーは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [パッケージ] ビューにあるパッケージに [Web からダウンロード] が設定されていて、"ロケーション" 設定の下にある "URL" 設定が空白、または無効な URL が指定されている場合に発生します。このエラーを解決するには、以下のいずれかの文字列で始まる URL を入力します: • http:// • https:// • ftp://

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7225	プロジェクトに、パッケージが含まれていません。	このビルド エラーは、パッケージが含まれていないアドバンスト UI またはスイート/アドバンスト UI プロジェクトをビルドしようとしたときに発生します。 このエラーを解決するには、[パッケージ] ビューを使って、プロジェクトにパッケージを追加します。詳細については、「パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加するときのガイドライン」を参照してください。
-7223	[モード] 条件または [検出] 条件は、Setup.exe が正しく実行されるために必須です。	このビルド エラーは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに、.exe パッケージが含まれていて、それに対して、適切な条件が構成されていないときに発生します。 プロジェクト内のパッケージへの検出条件の追加については、次を参照してください: • アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド • アドバンスト UI またはスイート/アドバンスト UI プロジェクトで条件を定義するときのガイドライン モード条件の作成については、「アドバンスト UI またはスイート/アドバンスト UI インストールのインストール モードまたはメンテナンス モードをトリガする」を参照してください。
-7222	メディアにサポート言語が含まれていません。メディアに最低 1 つのサポート言語を選択しなくてはなりません。	このエラーは、メディアにサポート言語が選択されていない場合のみ発生します。この問題を解決するには、[リリース] ビューの ″言語″ 設定で、最低 1 つの言語が選択されていることを確認してください。
-7220	DIM リファレンスを追加中に、エラーが発生しました。	このエラーは、DIM リファレンスが、[DIM リファレンス] ビューで示されているように、プロジェクトに含まれているにもかかわらず、その DIM プロジェクト ファイル (.dim) が見つからない時に発生します。原因としては、.dim ファイルが別の場所に移動された、または名前が変更されたにもかかわらず、インストール プロジェクトで、その DIM リファレンスを含むパスまたはファイル名が更新されていないなどが考えられます。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–7218	DIM のビルド中に競合が検出されました。テーブル %1 行 %2 のデータが以下の列のデータと異なります: %3	この警告は、開いているインストール プロジェクトと、その DIM リファレンスの 1 つの間でデータの競合が発生した時表示されます。この警告メッセージで、%1 は、競合を含むテーブルの名前を表し、%2 は、該当する行を表し、%3 は、次のフォーマットで表示されます: **列名 ("インストール ユニットの値" <> "インストール プロジェクトの値")** DIM リファレンスのテーブル データは、DIM プロジェクトの [ダイレクト エディター] ビューにあります。インストールのテーブル データは、インストール プロジェクトの [ダイレクト エディター] ビューにあります。 データの競合が発生した場合、[DIM リファレンス] ビューの [ビルド オプション] タブにある "競合の解決" 設定で選択された値に応じて、インストール プロジェクトのデータまたは DIM データが使用されます。 • **ベース プロジェクト値を使用する** – この値が選択された場合、DIM 内にある値の代わりに、インストール プロジェクト内にある値が使用されます。 • **DIM プロジェクト値を使用する** – この値が選択された場合、インストール プロジェクト内にある値の代わりに、DIM プロジェクト内にある値が使用されます。 適切な値が使用されている場合、この警告メッセージは無視できます。上書きに使用される値を使用する場合、そのプロジェクト内の値を訂正することもできますし、"競合の解決" 設定の値を変更することもできます。ただし、この DIM リファレンスで他の競合が発生している場合、"競合の解決" 設定の値を変更すると、ビルド時に他の競合に対して設定される値も変わる可能性があるので注意してください。 詳細については、「ビルド時に発生する基本の MSI プロジェクトと DIM リファレンス間の競合を解決する」を参照してください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7216	このプロジェクトには、現在使用されていない InstallScript オブジェクトが含まれています。これらのオブジェクトは、プロジェクトから削除する必要があります。可能な限り、これらのオブジェクトの代わりに前提条件を使用しなくてはなりません。	InstallScript オブジェクトに代わって、InstallShield 前提条件が推奨されます。InstallScript オブジェクトの代わりに、InstallShield 前提条件が推奨されます。InstallShield 前提条件エディターを使って、独自の InstallShield 前提条件を作成できます。これらの InstallShield 前提条件は、InstallScript、InstallScript MSI、および基本の MSI プロジェクトで共有することができます。詳しくは、次を参照してください: • InstallShield 前提条件を定義する • InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加する
-7215	ビルド イベントをスキップしています。ビルド イベントは、InstallShield Premier Edition でのみサポートされています。	InstallShield では、ビルド プロセスの様々な段階で実行するコマンドを指定できます。このビルド イベント機能は、InstallShield Premier Edition で提供されています。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。 InstallShield Professional Edition を使ってビルド イベントを含むリリースをビルドしようとすると、InstallShield はビルド イベントの設定を無視してこの警告を生成します。 この警告を解決するには、リリースのビルドに InstallShield Premier Edition を使ってください。
-7214	テーブル %1 のロード中にエラーが発生しました。	このエラーは、エラー メッセージに記述されているテーブルがプロジェクトから不足している、またはテーブルの列が不足している場合に発生します。 このエラーを解決するためには、[ダイレクト エディター] ビューを開いて、エラー メッセージに記述されているテーブルを探します。それが存在する場合、テーブル内の列と新しいプロジェクトに含まれている列とを比べて、足りないテーブルを追加します。テーブルが存在しない場合、新しいプロジェクトを .idt ファイルとしてエクスポートしてから、その .idt ファイルを既存プロジェクトにインポートします。詳細については、次を参照してください。 • ダイレクト エディターからテーブルをエクスポートする • ダイレクト エディターを使ってテーブルをインポートする

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7211	圧縮済みネットワーク イメージ Setup.exe をビルドしています。その他のビルドの種類は、InstallShield の評価モードでは使用できません。	InstallShield を評価モードで使用している (つまり、アクティベートしていない) 場合、圧縮済み Setup.exe ファイルをビルドできますが、その他のリリースの種類をビルドすることはできません。Windows Installer ベースのリリースを作成すると、.msi データベースが常に Setup.exe ファイル内に組み込まれます。 評価モードで非圧縮リリースをビルドしようとすると、非圧縮オプションは無視されて、InstallShield によってビルド警告が表示されます。 InstallShield のアクティベーションを行うと、評価モードの制限が解除されます。
-7209	現在の Windows Installer パッケージ名には Unicode 文字が含まれているため、圧縮済みネットワーク イメージ Setup.exe をビルドする際にランタイム エラーが発生する可能性があります。	このビルド警告は、以下の条件が当てはまる場合に発生します: • .msi パッケージの名前に、ビルド マシンの ANSI コード ページでサポートされていない Unicode 文字が含まれている。 • リリースにセットアップランチャー (Setup.exe file) が含まれている。 この警告は、Windows Installer パッケージが Setup.exe からターゲット システム上の一時保管場所に抽出された場合に、エラー 1152 が発生する可能性があることを通知します。 この警告を解決するには、マシンの ANSI コード ページでサポートされていない Unicode 文字を含まないように .msi パッケージの名前を変更します。そのためには、[リリース] ビューにある製品構成の [全般] タブにある "MSI パッケージ ファイル名" 設定を使用します。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7208	このパッケージの現在の場所は、Windows Installer が .cab ファイルを抽出しようとしたときにランタイム エラーの原因となる Unicode 文字を含みます。	この警告は、以下の条件が当てはまる場合に発生します: • InstallShield がビルドしているリリースのパスにビルド マシンの ANSI コード ページでサポートされていない Unicode 文字が含まれている。 • 製品のプログラム ファイルが .cab ファイルに圧縮されるようにリリースを構成している。 エンド ユーザーがビルド マシンの ANSI コード ページでサポートされていない Unicode 文字を含まないパスの場所からリリースを起動する場合は、この警告を無視できます。 エンド ユーザーがターゲット システムの ANSI コード ページでサポートされていない Unicode 文字を含むパスからインストールを起動しようとすると、Windows Installer が .cab ファイルを抽出しようとしたときにランタイム エラー 1311 が発生します。
-7207	1 つのパッケージに MsiLockPermissionsEx テーブルと LockPermissions テーブルの両方を含めることはできません。	1 つのインストールに MsiLockPermissionsEx テーブルと LockPermissions テーブルの両方を含めることはできません。サービスに ″アクセス許可″ 設定およびその下にある設定を使うと、パッケージの MsiLockPermissionsEx テーブルを構成していることになります。[一般情報] ビューの ″ロックダウンの設定方法″ 設定で [従来型の Windows Installer 処理] を選択した場合に、プロジェクト内の 1 つ以上のファイル、フォルダー、またはレジストリ キーにアクセス許可を設定すると、パッケージの LockPermissions テーブルを構成していることになります。 詳細については、「ロックダウン環境におけるファイル、フォルダー、レジストリ キー、および Windows サービスのセキュリティ保護」を参照してください。
-7201	言語 ‘%1’ がプロジェクトで見つかりませんが、このリリースには含まれています。	このビルド警告は、ある言語が [リリース] ビルドの [ビルド] タブにある ″UI 言語″ 設定で選択されていて、それが、[一般情報] ビルドの ″セットアップ言語″ 設定で選択されていないときに発生します。 このビルド警告は、リリースに、指定されたランタイム言語が含まれないことをアラートします。指定されたランタイム言語をリリースに含めない場合、この警告は無視できます。 指定されたランタイム言語をリリースに含める場合、その言語を ″セットアップ言語″ 設定に追加します。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7187	%1 テキスト変更の "検索する文字列" 設定が構成されていません。このテキスト変更は実行されません。	[テキスト ファイルの変更] ビューで変更セットに変更アイテムが追加されていますが、"検索する文字列" 設定の値が空白です。この設定には必ず値が必要です。そうでない場合、実行時にテキスト ファイルの変更が行われません。 詳細については、「テキスト ファイルの変更を指定する」を参照してください。
-7186	%1 テキスト変更セットの "含めるファイル" 設定が構成されていません。このテキスト変更セットに関連付けられたテキスト変更は、いずれも実行されません。	[テキスト ファイルの変更] ビューで変更セットが追加されていますが、"含めるファイル" 設定の値が空白です。この設定には必ず値が必要です。そうでない場合、実行時にテキスト ファイルの変更が行われません。 詳細については、「テキスト 変更セットを作成する」を参照してください。
-7185	%1 トランスレーション (文字列 ID %2) には、コードページ %3 で使用できない文字が含まれています。	このエラーは、以下の両方の条件が当てはまるときに発生します: • 指定言語に入力された文字列 ID 値に、その言語のコードページでは使用できない文字が含まれている。 • 言語固有 (ANSI) のデータベースをビルドしている。 このエラーを解決するには、適切なコードページから文字を使用するように、[文字列エディター] ビューで文字列 ID の値を編集します。 ターゲット言語のコードページでは使用できない文字を使う必要がある場合、"UTF-8 データベースのビルド" 設定で [はい] を選択します。ただし、Windows Installer は UTF-8 データベースを完全にサポートしないため、一部の状況においてユーザー インターフェイスに問題が生じる場合があります。詳細については、"UTF-8 データベースのビルド" 設定の説明を参照してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7184	%1 列 (%2 テーブル) に、コードページ %3 で使用できない文字が含まれています: "%4"	このエラーは、以下の両方の条件が当てはまるときに発生します: • ソース プロジェクトのデータベースのテーブルに、1 つまたは複数のターゲット言語のコードページで使用できない文字が含まれている。 • 言語固有 (ANSI) のデータベースをビルドしている。 このエラーを解決するには、適切なコードページからの文字を使用するように、エラー メッセージで説明されたデータを変更してください。たとえば、エラー メッセージが Shortcut テーブルについて記述している場合、その文字列を [ショートカット] ビューで変更してみてください。 ターゲット言語のコードページでは使用できない文字を使う必要がある場合、"UTF-8 データベースのビルド" 設定で [はい] を選択します。ただし、Windows Installer は UTF-8 データベースを完全にサポートしないため、一部の状況においてユーザー インターフェイスに問題が生じる場合があります。詳細については、"UTF-8 データベースのビルド" 設定の説明を参照してください。
-7174	"%1" に、プロジェクトの文字列テーブルに含まれていない文字列テーブル エントリ '%2' への参照が含まれています。	この警告メッセージで、%1 はプロジェクトに含まれる InstallScript ファイル (.rul) への完全修飾パスを示し、%2 はその .rul ファイルで使用されているが、プロジェクトの文字列エントリにはリストされていない ID を示します。 InstallShield は、プロジェクトの InstallScript ファイルに含まれているが、[文字列エディター] ビューには含まれていない各文字列 ID のビルド時に、この警告を表示します。 その文字列 ID を含む InstallScript コードの行を参照するには、ビルド時に [出力] ウィンドウの [タスク] タブ上に表示される警告メッセージをダブルクリックします。文字列 ID は、文字列定数演算子 (@) で始まります。 このビルド問題を解決するためには、文字列 ID が [文字列エディター] ビューで定義されていることを確認してください。また、InstallScript コードの文字列 ID のスペルと [文字列エディター] ビュー ID のスペルとを確実に一致させます。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7143	コンポーネント %1 は 64 ビット フォルダーにインストールされますが、64 ビット コンポーネントとマークされていません。これにより、このコンポーネントのファイルに正しくないインストール パスが与えられる可能性があります。	コンポーネントが 64 ビットとして構成されていない場合、Windows Installer はそのコンポーネントのファイルを適切な 64 ビットの場所にインストールしない可能性があります。 コンポーネントが 64 ビットであると指定するには、コンポーネントの "64 ビット コンポーネント" 設定で [はい] を選択します。 "64 ビット コンポーネント" 設定、および追加のコンポーネントの設定に関する詳細については、「コンポーネントの設定」を参照してください。
-7138	現在ビルド中のリリースの [Msi コマンドライン] が、REINSTALLMODE を含んでいます。チェックサムやバージョンにかかわらず、すべてのファイルを強制的に再インストールする 'a' オプションがパラメーターに含まれている可能性があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7137	%2 カスタム アクションの [カスタム アクション] ビューにある "ヘルプ ファイルのパス" 設定で選択された %1 ファイルにはテキストが含まれていません。	警告で通知されたカスタム アクションの正常な動作を説明するファイルを作成します。.txt、.htm、.rtf ファイルなど、テキストベースのファイルを指定してください。次に、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) で、カスタム アクションを選択し、"ヘルプ ファイルのパス" 設定で作成したファイルを選択します。 詳細については、「カスタム アクションの動作をドキュメント化する」を参照してください。
-7135	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7128	setup.exe.manifest の埋め込みエラー。	このエラーは、Setup.exe ランチャでのアプリケーション マニフェストの埋め込みに問題があるときに発生します。アプリケーション マニフェストでは、Windows Vista 以降のプラットフォーム上でインストール (セットアップランチャー、InstallShield 前提条件および .msi ファイル) を実行するためのインストールの Setup.exe ファイルによって必要とされる最低特権レベルを指定します。 このエラーは、InstallShield Supportフォルダーからテンプレート マニフェスト ファイルが不足している場合にも発生します。また、Setup.exe テンプレートが InstallShield Redist¥Language Independent¥i386 フォルダーから不足している場合にも発生します。 このエラーを解決するには、InstallShield の修復を実行してください。
-7125	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7124	MSBuild を起動できませんでした。.NET Framework %1 が、システムの '%2' にインストールされていることを確認してください。Visual Studio 内から InstallShield を実行するとき、Visual Studio 2005 以降が必要です。	MSBuild を使って、InstallShield プロジェクトを含む Visual Studio ソリューションをビルドするには、開発マシンに .NET Framework 3.5 以降が必要です。また、Visual Studio 2005 以降を使用しなくてはなりません。
-7123	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7122	%1 のソース ファイルに情報を書き込む際に、エラーが発生しました。ファイルが書き込み可能であり、ISTemplate.manifest ファイルが読み取り専用になっていないことを確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7121	マニフェスト ファイル %1 に、少なくとも 1 つのファイル要素にある COM 情報が含まれていません。マニフェストに関連付けられているすべての COM モジュールが自己登録であること、および自己登録が各 COM モジュールで失敗しないことを確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7120	テーブル %1 のビルド中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7119	コンポーネント %1 のキーファイルを見つけることができませんでした。アドバタイズされたショートカット %2 はこのコンポーネントのキーファイルをポイントします。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7118	InstallScript カスタム アクション %1 が、InstallScript MSI プロジェクトの InstallUISequence でシーケンスできませんでした。	[ユーザー インターフェイス] シーケンスは、InstallScript MSI インストールで実行されません。イベント ドリブン型の InstallScript コードは、ユーザー インターフェイスを処理します。この問題を解決するには、以下のいずれかを実行します。 • [実行] シーケンスの InstallScript カスタム アクションをスケジュールします。 • InstallScript カスタム アクションの代わりに、InstallScript イベント ドリブン型コードを使用します。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7117	標準 dll から呼び出され、Binary テーブルに格納されているカスタム アクション %1 が、遅延/ロールバック/コミット実行を行うことができませんでした。	標準 dll を使用して、遅延実行、ロールバック実行、またはコミット実行のカスタム アクションを呼び出そうとすると、このエラーが発生します。非 Windows Installer .dll (MyFunc(MSIHANDLE) エントリ ポイントを持たないファイル) を呼び出したとき、InstallShield は次を行います。 • 標準 Windows Installer エントリ ポイントを使用して、ラッパー .dll を作成します。 • Binary テーブルに、.dll および InstallShield ラッパー .dll を格納します。 カスタム アクションが呼び出されたとき、Windows Installer は InstallShield ラッパー .dll を呼び出します。そのあと、ラッパー .dll は、.dll を Binary テーブルから抽出し、指定した関数を呼び出します。ただし、InstallShield ラッパー .dll は、遅延/ロールバック/コミットのカスタム アクションが実行されている間、Binary テーブルのアクセス権がないため、.dll の抽出および呼び出しはできません。 このエラーを解決するには、以下のいずれかのオプションを試みることができます。 • 標準 Windows Installer エントリ ポイントがある新しい .dll を作成して、それを直接呼び出します。 • .dll を自分で作成したくない場合、InstallScript コードを使用して、同関数を実行します。 • [遅延]、[ロールバック]、または [コミット] モードの代わりに、[即時] モードでカスタム アクションを実行するようにスケジュールします。
-7116	カスタム アクション DLL %1 がリリースに見つかりませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7113	%1 の言語サポートは、この Edition には含まれていません。	このエラーは、ひとつのプロジェクトでデフォルトの数を超える数の言語を有効化した場合、Premier Edition 以外では 2 言語以上を使ってビルドしたときに発生します。(最初のエラーで停止するよう設定されていない限り) ビルドは成功しますが、デフォルトの数を超えるの言語はビルドされません。制限数以内の最初の 2 つの言語はビルドされます。%1 はこのリリースに含まれていない言語の名前で置換されます。このエラーを解決するには、リリースのビルドに InstallShield Premier Edition を使ってください。
-7108	%s は、大きすぎて CAB に保存できません (最大 2GB)。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7107	マージ モジュール %1 は、圧縮ビルドからのインストールをサイレントで失敗します。圧縮ポストビルドを利用すると適切に動作する可能性があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7104	コンポーネント %1 のインストール先が GlobalAssemblyCache です。コンポーネントに、キーファイルが必要です。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7101	このコンポーネント %1 は複数の機能で共有されています。このために、MSI のオンデマンド インストール テクノロジを使用する機能に問題が起こる可能性があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7098	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7097	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7096	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7095	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7092	バイナリ エントリ ISSetup.dll 不足	このエラーは、InstallScript カスタム アクションを含むプロジェクトをビルドするとき、InstallScript エンジン (ISSetup.dll) が Binary テーブルから削除されている場合に発生します。このエラーを解決するには、**ISSetup.dll** と呼ばれる新しいエントリを手動で Binary テーブルに追加してから、次の DLL を参照します: *InstallShield Program Files フォルダー*¥Redist¥Language Independent¥i386¥ISSetup.dll
-7088	%1 はロードに失敗しました。関連付けられたカスタム オブジェクトをビルドするには、このファイルをロードする必要があります。	ファイル %1 が InstallShield Program Files フォルダー内の System サブフォルダにあることを確認します。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7087	Windows Installer オブジェクトの作成でエラーが発生したため、カスタム オブジェクトをインストール メディアに含めることができません。カスタム オブジェクトを含めるには、Windows Installer をビルド システムにインストールする必要があります。	Windows Installer がビルド システムに適切にインストールされていることを確認してください。
-7086	デバイス ドライバー インストール フレームワークは、現在利用できません。デバイス ドライバー機能は、アップデートが提供されるまで完全には機能しません。詳細はテクニカル サポートまでお問い合わせください。	デバイス ドライバー ウィザードを使用してインストールと一緒にデバイス ドライバー パッケージに含めたプロジェクトのリリースをビルドするとこのエラーが発生します。デバイス ドライバーをインストールするのに必要な再配布可能ファイルは、InstallShield へのアップデートで利用可能になります。アップデートの取得方法は、再配布可能ファイルが利用可能になったとき、ナレッジ ベースのビルド エラーに関する項目で見ることができます。再配布可能ファイルを受け取る前にビルドエラーを解決するには、[コンポーネント] ビューでデバイス ドライバー コンポーネントを削除してから、リリースを再ビルドします。
-7084	VBScript カスタム アクション %1 は有効な VBS ファイルをポイントしていません。	この警告メッセージは、インストールに VBScript カスタム アクションを追加するとき、カスタム アクションに指定されたファイルが VBScript ではないときに表示されます。このエラーを解決するには、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) で指定されたカスタム アクションに対して適切なタイプのファイルを選択します。
-7083	スクリプト '%s' がどの接続にも関連付けられていません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7082	Sql Script を生成中にエラーが起きました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7080	デバイス ドライバー パッケージの "ビルド時にスキャン" プロパティが設定されているかどうかを確認するため、コンポーネントをテスト中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7079	デバイス ドライバー パッケージへのパスを取得中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7078	デバイス ドライバー パッケージ %1 をスキャンすることができませんでした。ファイルが見当たりません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7077	デバイス ドライバー パッケージをスキャン中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7065	MSI 3.1 エンジンが見つかりませんでした。Microsoft がこのエンジンを配布している場合、[ツール] メニューの [再配布可能ファイル ダウンローダー] を利用して、これダウンロードすることができます。ビルド済みのインストールには 3.0 エンジンが含まれています。	このメッセージは、Windows Installer エンジンの バージョン 3.1 を含めるように選択されている場合で、ビルド マシンにそれが見つからないときにビルド エラーとして表示されます。 このメッセージは、Windows Installer エンジンのバージョン 3.1 または 2.0 の適切なバージョンを含めるオプションが選択されている場合で、バージョン 3.1 がビルド マシンに見つからないときにビルド警告として表示されます。
-7064	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7063	コンポーネント %2 のファイル %1 を Windows Fonts フォルダーにインストール中ですが、Font テーブルに対応するレコードがありません。Font はターゲット システムに適切に登録されません。	この警告を解決するには、[ファイルとフォルダー] ビューを開いて、メッセージの中で指摘のあったフォントを削除し、そのあと Fonts フォルダーにもう一度そのフォントを追加します。InstallShield は Font テーブルに対応するレコードを自動的に追加します。
-7062	クライアント マシンから COM+ に関する設定をリフレッシュ中です。クライアント マシンから COM+ の設定をリフレッシュ中に、エラーが発生しました。	このビルドエラーは、例外エラーが、[コンポーネント サービス] ビューで選択された COM+ アプリケーションで起きたときに発生します。COM+ 設定をリフレッシュするオプションが、ビルド プロセスの最中に提供されます。ビルドエラー -7062 は、このオプションを選択して、リフレッシュ プロセスが失敗したとき発生します。このエラーを解決するには、[コンポーネント サービス] ビューを開いて COM+ アプリケーションを選択し、インストール タブでビルド時にクライアント マシンから COM+ 設定をリフレッシュするチェック ボックスをクリアします。
-7061	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7060	ライセンス要件に従い、InstallShield では MSI 3.0 Beta からMSI 3.0 エンジン再配布可能ファイルを提供する必要があります。instmsi.exe を MSI 3.0 Beta フォルダーから ¥redist¥Language Independent¥i386¥MSI3.0¥instmsiw.exe へコピーしてください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7059	"最新のパッチ バージョン" プロパティをアップデートすることができません。従って、このプロジェクトからの将来のパッチ ビルドは適切に配列されない可能性があります。プロジェクトが読み取り専用になっていないことを必ず確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7058	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7036	プライマリ アプリケーション アセンブリのロード中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-7025	COM+ コンポーネント サーバー ファイル '%1' が、システムで見つかりません。コンポーネント '%2' はターゲット システムにインストールされません。COM+ アプリケーションがインストールされた後に、コンポーネントを手動でインストールする必要があります。	%1 は、不足している COM+ DLL を参照し、%2 は ProgID を参照します。この警告を解決するためには、システムの適切な場所に DLL を追加するか、プロジェクトから COM+ コンポーネントを削除します。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7024	J# をビルドに含むには、.NET も含める必要があります。	J# には、.NET Framework 再頒布可能パッケージ バージョンが必要ですが、インストールに含まれていません。エンドユーザーのシステム上に .NET Framework があることが確実に分かっている場合は、この警告メッセージを無視してください。.NET Framework を持っていない可能性がある場合、それをインストールに追加します。.NET Framework がエンドユーザーのシステム上にない場合、J# をインストールの一部としてインストールする際、問題が起きる可能性があります。 詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。
-7023	内部ビルド エラー。	このエラーは、 MFC 7.1 等のマージ モジュールが適切に構成されていない、または破損しているときに発生します。プロジェクトにマージ モジュールが複数存在するときに、どちらがエラーの原因になっているか分からない場合、まず最初にその問題になっているマージ モジュールを特定する必要があります。方法として、まず、プロジェクトからマージ モジュールのどちらか 1 つを削除し、リリースをビルドしてみます。エラーが続けて起こるようであれば、その問題になっているマージ モジュールを特定したことになります。そうでない場合、別のマージ モジュールを削除して、またビルドします。エラーの原因になっているマージ モジュールの特定ができた時点で、再インストールしてみます。エラーがそれでも解決されないようであれば、作成者に問い合わせて最新のバージョンを取得してください。
-7015	setup.inx をメディア フォルダーにコピー中、エラーが起きました。予期せぬエラー	このエラーは、コンパイル済みスクリプト ファイル Setup.inx が ¥Media¥¥Disk Images¥Disk 1 フォルダー内のメディア フォルダーにコピーできないときに起こります。ディスク容量が足りないか、ファイルがロックされている可能性があります。
-7014	setup.inx をメディア フォルダーにコピー中、エラーが起きました。スクリプトは読み取り専用です。	このエラーは、コンパイル済みスクリプト ファイル Setup.inx が読み取り専用になっているときに発生します。このエラーを解決するには、Setup.inx ファイルを見つけます。このファイルは通常 ¥Media¥¥Disk Images¥Disk 1 フォルダー内にあります。このファイルを右クリックして、[プロパティ] を選択します。[読み取り専用] チェック ボックスをクリアにします。
-7012	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-7011	%1 が GlobalAssemblyCache へインストールされるためにはストロング ネームが必要です。	ストロング ネームが使われている場合のみ、Global Assembly Cache にアセンブリをインストールすることができます。ストロング ネームには、アセンブリの簡易テキスト名、バージョン番号、カルチャ情報(利用可能な場合)、パブリック キーおよびデジタル署名が使われます。Global Assembly Cache でストロング ネームのアセンブリを使って、複数の異なるバージョンの .dll ファイルを同名で持つアセンブリを複数バージョン持つことができます。Global Assembly Cache に保存されているアセンブリはコンピューター上で異なるアプリケーション間で共有することができます。このエラー メッセージを解決するとき、2 つのオプションがあります。 • アセンブリを他のアプリケーションと共有することが明示的に要求されていない場合、アセンブリのターゲット インストール先を Global Assembly Cache 以外の場所に変えます。COM interop がアセンブリにアクセスできるようにするためには、アセンブリを Global Assembly Cache にインストールする必要はありませんので注意してください。 • Global Assembly Cache にインストールするためにストロング ネームが使われているアセンブリを選択します。アセンブリは、1 度作成されると、ストロング ネームで署名できませんので注意してください。アセンブリは、作成時のみストロング ネームで署名できます。
-7001	前回のビルド中に .msi ファイルが部分的にのみビルドされました。スクリプトを .msi ファイルにストリーミングするには、.msi ファイルを完成させる必要があります。[ビルド] メニューから [ビルド <リリース名>] を選択して、完全な .msi ファイルをビルドします。	ビルドをキャンセルすると、部分的にビルドされた .msi ファイルは終了します。部分的な .msi ファイルにスクリプトをコンパイルしようとするとこのエラーが発生します。完全な .msi ファイルをビルドするには、[ビルド] メニューから [ビルド <リリース名>] を選択してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6654	ファイル %1 はコンポーネント %2 のキーファイルではありません。製品とともにインストールされる標準 DLL の機能を呼び出すときに、そのアクションが遅延実行としてスケジュールされる場合、呼び出される DLL はコンポーネントのキーファイルでなければなりません。	呼び出した DLL をコンポーネントのキーファイルに設定します。
-6653	インストールの機能 %1 には、1600 以上のコンポーネントが含まれています。1 機能につき、含めることのできる最大コンポーネント数は 1600 です。	詳細については、「ICE47」および「FeatureComponents テーブル」を参照してください。
-6652	InstallUtilLib.dll と共に利用する _isconfig.xml を作成できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6651	ビルドしているセットアップに 32,767 を超えるファイルがあります。セットアップ パッケージを適当な MSI スキーマに自動的に切り替えます。	この警告は、インストール パッケージに含まれるファイルが 32,767 ファイルを超える場合に表示されます。InstallShield は自動的に大きなパッケージ スキーマを適用します。 ***注意** • パッケージで大きなパッケージ スキーマを使う場合、パッチに問題があります。パッチをビルドする場合、エラー「プライマリ トランスフォームを生成できません。」がトリガーされます。*
-6647	ファイルを %1 から %2 へ移動できません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6646	マージ モジュール %2 の %1 プロパティが NULL に設定されています。プロパティを NULL にすることはできません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6645	%1 はロードに失敗しました。エラー: %2 エラーの説明: %3	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6641	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6640	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6639	マージ モジュール %1 は次のマージ モジュールのどれか 1 つをセットアップへ含むことを必要とします: %2	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6638	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6637	*"C:¥RegTest¥SingleSlash!"=dword: 2* のインポート中、コンポーネント NewComponent1 の無効レジストリ データ (イタリック体の情報が無効データを示します)。	ここに示した無効データは円記号を 2 つではなく 1 つしか使わなかった例です。別の例では *dword* ではなく *DWORD* を入力しています。
-6636	キーファイル %1 および %2 が File テーブルで見つかりました。大文字小文字に関わらず、キーが同一名の場合、ファイルがターゲット システムにインストールされたとき予期しない結果が招くことがあります。これは、キャビネット ファイルに圧縮されたファイルがファイル キーを使って名前が付けられるためです。この問題を解決するためには、圧縮セットアップまたはマージ モジュールをビルドする場合、ファイルキーの 1 つをキャビネット ファイルで一意になるように変更します。[ダイレクト エディター] ビューでファイル キー名を変更することができます。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6635	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6634	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6633	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6632	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6631	InstallScript MSI プロジェクトに、複数インスタンス サポートがありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6630	InstanceId プロパティが Property テーブルにありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6629	InstanceId プロパティの値が数値ではありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6628	InstanceId プロパティの値が、ISProductConfigurationInstance テーブルの値と重複しています。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6627	コンポーネント %1 は、非ファイル データを含みますが、条件として InstanceId が含まれていません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6626	インスタンス トランスフォーム %1 をサブストレージに追加することができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6625	インスタンス トランスフォーム %1 を作成することができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6624	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6623	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6622	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6620	キー同期用の参照パッケージ %1 を開くことができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6619	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6618	動的に作成されたコンポーネントへのキーパスの設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6617	ファイル '%1' を同期マップへ追加中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6616	コンポーネント %1 へのキー パスの設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6615	ファイル %1 の File.FileName を更新することができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6614	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6613	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6612	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6611	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6610	ダイナミック リンク ファイル %1 が変更されたファイルかどうかを判断する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6609	ダイナミック リンク ファイル %1 が新しいファイルかどうかを判断する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6608	%1 が圧縮済みビットマップ ファイルの為、インストール中にスプラッシュ画面は表示されません。スプラッシュ画面を表示するには、非圧縮ビットマップ ファイルを指定しなくてはなりません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6607	%1 がビットマップ ファイルではない為、インストール中にスプラッシュ画面は表示されません。スプラッシュ画面を表示するには、有効なビットマップ ファイルを指定しなくてはなりません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6606	このセットアップは、マイナー アップグレード アイテム '%1' が指定するセットアップと同じ パッケージ コードを持ちます。パッケージ コードは一意でなくてはなりません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6605	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6604	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6603	アップグレードされたイメージ %1 を開く際に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6602	アップグレードされたイメージ %1 を閉じる際に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6601	アップグレード イメージ内のプロパティ %1 を設定しようとしてエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6600	アップグレードされたイメージからプロパティ %1 を読み出す際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6599	アップグレードされたイメージに標準 QuickPatch アクションを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6598	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6597	ISQuickPatchHelper.dll をアップグレードされたイメージのバイナリ テーブルに追加する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6596	アップグレード イメージ内のアクション %1 をシーケンスしようとしてエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6595	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6594	アップグレードされたイメージのベースアクション %1 用のシーケンス番号を決定する際にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6593	%1 のファイル クラス ラッパー初期化中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6592	ファイル %1 の処理方法を判断中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6591	ファイル %1 をパッチするのに必要な設定を行なう際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6590	%1 の RemoveFile テーブル エントリを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6589	コンポーネント '%1' を指定された機能に付加する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6588	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6587	%1 にフォーマットされたファイル名を生成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6586	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6585	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6584	新規コンポーネントの作成を試行中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6583	コンポーネント %1 をロードする際に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6582	コンポーネント %1 を保存中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6581	コンポーネント '%1' を機能 %2 に追加中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6580	機能 %1 のサブ機能を生成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6579	コンポーネント %2 のテーブル '%1' でデータを同期中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6578	アップグレードされたイメージの作成時において、コンポーネント '%1' からのショートカットの同期中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6577	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6576	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6575	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6574	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6573	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6572	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6571	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6570	プロパティ ‘%1’ をパッチ構成プロパティ ファイルへ書き込み中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6569	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6568	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6567	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6566	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6565	イメージ ファミリ ‘%1’ をパッチ構成プロパティ ファイルへ書き込み中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6564	ターゲット イメージ ‘%1’ をパッチ構成プロパティ ファイルから削除中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6563	ターゲット イメージ ‘%1’ をパッチ構成プロパティ ファイルへ書き込み中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6562	アップグレードされたイメージ %1 をパッチ構成プロパティファイルへ書き込み中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6561	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6560	保存されたパッケージ コード %1 が、参照されたパッケージ %3 の %2 と一致しません。参照された Quick Patch の MSI パッケージを変更すると、その MSI パッケージを参照する全ての Quick Patch が無効になります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6558	InstallExecuteSequence テーブルのカスタム アクション %1 はインストールされたファイルから実行されます。カスタム アクションを問題なく実行するには、ソース ファイルがローカルにインストールされたことを確認する条件が必要な場合があります。	インストールされたファイルから実行されるカスタム アクションを問題なくに実行するには、条件ステートメントを使って確実にファイルをローカルにインストールしなくてはなりません。 • カスタム アクションが RemoveFiles の前にシーケンスされている場合 – コンポーネントがローカルにインストールされている場合のみアクションを実行します。(?ComponentName=3) • カスタム アクションが RemoveFiles とInstalLFiles の間にシーケンスされている場合 – コンポーネントがローカルにインストールされている場合のみアクションを実行します。アンインストールではアクションを実行しません。(?ComponentName=3) AND NOT($ComponentName=2) • カスタム アクションが InstallFiles の後にシーケンスされている場合 – コンポーネントがローカルにインストールされる場合のみアクションを実行します。($ComponentName=3) ***メモ:*** *ComponentName はソースファイルと関連付けられているコンポーネントです。*
-6557	InstallShield プログラム ファイル フォルダー¥Support¥NetCF.ini ファイルから指定されたプラットフォーム/プロセッサの .NET Compact Framework が見つかりませんでした。	.NET Compact Framework が正しい場所にあるか確認してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6556	セットアップへ .NET Compact Framework .cab ファイルを含む際にエラーが発生しました。	このエラーは次の理由で発生した可能性があります: • InstallShield の プログラム ファイル フォルダー¥Support¥.NetCR.ini ファイルで指定された場所に .NET Compact Framework ファイルが見つかりませんでした。 • CEDefault.ico ファイルが InstallShield の プログラム ファイル フォルダー¥Program¥04xx フォルダーにありません。 • ISCEInstall テーブルへの書き込みエラー。
-6555	InstallShield Program File Folder¥Support¥CFAppMgr.ini ファイルを一時保管場所にコピーすることができません。	CFAppMgr.ini ファイルが InstallShield の プログラム ファイル フォルダー¥Support フォルダーにあることを確認してください。このファイルがない場合、InstallShield セットアップを修復モードで起動してください。一時ファイルをすべて削除してシステム上の空きスペースを増やしてください。
-6553	プロセッサの .NET Compact Framework .cab ファイルが見つかりませんでした。	InstallShield の プログラム ファイル フォルダー¥Support¥NetCF.ini ファイルを開き、このプロセッサの .NET CF 選択を構成して問題を解決してください。
-6552	コンポーネント %2 のファイル キー %1 は既に使用中であるため、以前のパッケージと同期することができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6551	NewComponent1 コンポーネントのレジストリレコードである Registry1 の名前列と値列が空です。Windows Installer では、空の文字列にデフォルト値の *HKEY_LOCAL_MACHINE¥New Key* #1 が設定されます。"デフォルト" 値を "設定しない" に設定する場合、[レジストリ] ビューで " 存在しない場合はインストールする、存在する場合はアンインストールする" をそのキーに設定します。キーがターゲットマシン上に既に存在する場合は、(デフォルト) 値は変更されません。	この警告は、プロジェクトに追加されて、デフォルトのキーが手動で設定されなかったすべてのレジストリで表示されます。"デフォルト" 値を "設定しない" に設定するには、[レジストリ] ビューで "存在しない場合はインストールする、存在する場合はアンインストールする" をそのキーに設定する必要があります。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6550	%1 はグローバル アセンブリ キャッシュにインストールできる有効な .NET アセンブリではありません (%1 には完全パスを差し替える)。	有効なファイルのみグローバル アセンブリ キャッシュにインストールされます。インストール用に選択したファイルが有効なアセンブリであることを確認してください。
-6549	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6548	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6547	ファイル '%1' は、セットアップに既に存在します。新規ファイルを挿入する代わりに既存ファイルを変更してください。そうしなければ、このファイルが適切にアンインストールされない可能性があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6546	新規ファイル '%1' の競合をテスト中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6545	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6544	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6543	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6542	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6541	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6540	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6539	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6538	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6537	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6536	%1 を開く際にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6535	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6534	コンポーネント %1 をクローン中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6533	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6532	ファイル %1 に更新されたコンポーネント名を設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6530	アップグレード項目 [2] のセットアップ [1] 処理エラー。これは有効なセットアップではありません。	このエラーは InstallShield の [アップグレード] ビューにマイナー アップグレード エントリが存在し、指定された以前のインストールが指定の場所に存在しない場合に発生します。解決方法は、ナレッジベース記事 Q108525を参照してください。
-6529	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6528	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6528	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6527	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6526	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6525	InstallExecuteSequence テーブルのカスタム アクション %1 はインストールされたファイルから実行されます。%2 アクションの後にシーケンスしなければなりません。カスタム アクションが適切にシーケンスされ、ベース アクションがシーケンスに存在することを確認してください。	このエラーは、セットアップの最中にインストールされたファイルから実行するカスタム アクションをオーサリングしたときに発生します。カスタム アクションはシーケンスで InstallFiles アクションの後に配置しなくてはなりません。さもなければ、Windows Installer がカスタム アクションを実行したときにファイルがターゲット システムに存在しません。唯一の例外は、このカスタム アクションをアンインストールで実行する場合です。その場合、ファイルは既にインストールされているのでカスタム アクションを InstallFiles の後に行う必要がありません。
-6524	InstallExecuteSequence テーブルのカスタムアクション %1 が据え置きになり、%2 と %3 の間に順序付ける必要があります。カスタムアクションが適切に順序付けられ、シーケンスに基本アクションがあることを確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6523	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6522	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6521	機能 %1 に必要な属性を設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6520	コンポーネント %1 の COM データ削除中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6519	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6518	ファイル %1 を SelfReg テーブルから削除中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6517	ファイル %1 を SelfReg テーブルへ追加中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6506	現在ビルド中のリリースの Msi コマンドライン属性が REINSTALL または REINSTALLMODE パラメーターを設定しています。これは、セットアップが常に再開モードで実行される原因となります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6504	最新のイメージ %1 は以前のイメージを含みません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6501	ISDLLWrapper テーブルをビルド中にエラーが発生しました。コンポーネント %1 が Component テーブルにありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6500	ISDLLWrapper テーブルをビルド中にエラーが発生しました。ファイル %1 が File テーブルにありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6499	シャローフォルダー構造設定は複数ディスクリリースでは利用しないで下さい。	[リリース] ビューの [ビルド] タブにある "シャロー フォルダー構造" 設定を構成できます。エラー -6499 が表示された場合、"シャロー フォルダー構造" 設定で [いいえ] を選択して、標準のビルド フォルダー構造が作成されるようにします。
-6497	パッチ作成警告: %1。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6496	バージョンが付いていないファイル '%1' のファイルハッシュが、以前のバージョンのそれと異なります。このファイルは更新されません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6495	ファイル '%1' のバージョンについて、3 番目のバージョン要素以降のみが異なります。バージョン文字列の最初の 3 つの要素が異なる必要があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6494	ファイル '%1' の新しいファイルは、セットアップに存在するファイルと同じバージョンを持ちます。このファイルは更新されません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6493	ファイル '%1' の新しいファイルは、セットアップに存在するファイルとよりも低い番号のバージョンを持ちます。このファイルは更新されません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6492	ファイル '%1' に関連機能が選択されていません。新しいファイルを追加するためには、この設定が必須です。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6491	新しいファイルパス '%1' は ファイル '%2' には存在しません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6490	グローバルアセンブリキャッシュをインストールするようコンポーネント %1 が構成されているのでジャストインタイムコンパイルを設定できません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6489	グローバルアセンブリキャッシュをインストールするようコンポーネント %1 が構成されているのでインストーラークラスを無視できません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6487	最新のイメージ %1 は以前のイメージを含みません。	このエラーは、[パッチのデザイン] ビューの最新のイメージが以前のイメージを 1 つも含んでいない場合に発生します。最新のイメージはパッチを作成するのに以前のイメージを最低 1 つ含まなくてはなりません。
-6486	MSI.DLL のバージョンが 2.0 より前なので、CAB ファイルのデジタル署名ができません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6485	MSI.DLL のバージョンが 2.0 より前なので MsiFileHash テーブルをビルドできません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6584	外部ファイルキー '%1' は、アップグレードされたイメージ '%2' に存在しません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6483	外部ファイル参照 '%2' が存在しません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6482	外部ファイルキーを検証中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6481	ターゲット イメージ ‘%1’ がありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6480	アップグレードしたイメージ ‘%1’ がありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6479	ディレクトリ ID '%1' のターゲットパスを解決することができません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6478	FON ファイルはフォント タイトルを持っている必要があります。コンポーネント %2 のフォント %1。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6477	キー '%1' および値 '%2' のレジストリ エントリの削除を追加中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6476	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6475	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6474	プロジェクトに追加されたレジストリ値の変更を処理中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6473	キー '%2' から値 '%1' を削除するための準備中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6472	すべての機能について、キー '%2' から 値 '%1' を削除中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6471	機能 '%3' の値 '%1' をキー '%2' から削除中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6470	プロジェクトに追加されたレジストリキーの変更を処理中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6469	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6468	すべての機能のレジストリキー '%1' を '%2' に変更するための設定中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6467	機能 '%3' のレジストリキー '%1' を '%2' に変更するための設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6466	デスクトップセットアップへの CE セットアップ情報入力エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6465	.inf ファイルからの CE セットアップメディア作成エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6464	CE セットアップメディアをビルドする為の .inf ファイルを作成中、CE セットアップ DLL 情報の追加エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6463	CE レジストリ エントリの追加エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6462	CE ファイル関連付けの追加エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6461	CE ショートカットの追加エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6460	CE セットアップファイルの追加エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6459	CE セットアップメディアをビルドする為の .inf ファイルへの情報入力エラー。	このエラーについての詳細は、<製品構成>¥<リリース>¥CEApps¥<CE プロジェクト名> に作成されるログファイルを参照してください。
-6458	アップグレードされたイメージ '%1' 用の [追加ファイル] 参照を作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6457	ターゲットイメージで '%1' で '検索されないファイルを無視する' が設定されています。アップグレードされたイメージで 'すべてのファイルを含める オプション を利用する場合、これは設定できません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6456	すべてのファイルをパッチに含むために必要な設定を作成中に予期しないエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6455	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6454	'%1' のファイルパッチすべてを作成することができません。ファイルの名前を変更する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6453	ファイル '%1' の名前を変更して元のファイル名 ‘'%2' に戻す際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6452	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6451	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6449	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6448	ファイル '%1' をアップグレードされたイメージから削除する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6447	ファイル '%1' のファイルテーブルエントリを削除中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6446	ファイル '%1' のアップグレードされたイメージ用のコンポーネントキーを削除中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6445	ファイル '%1' をアップグレードされたイメージへ追加中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6444	ファイルキー '%1' の新しいファイルエントリを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6443	ファイルキー '%1' の新しいコンポーネントを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6442	ルート機能 '%1' の下に新しい機能を作成することができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6441	機能 '%1' およびコンポーネント '%2' の機能コンポーネント エントリを作成することができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6440	新規ファイルの設定を作成するときに、機能 '%s' が見つかりません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6434	ISLatestRelease が指定したセットアップ '%1' が圧縮ビルドされています。この変数を利用するには、セットアップを非圧縮でビルドする必要があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6433	プロパティ %1 を設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6432	構成データプロパティを作成中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6431	イメージファミリー %1 の構成データを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6430	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6429	ターゲットイメージ %1 の構成データを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6428	アップグレードされたイメージ %1 の構成データを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6427	'%1' によって参照されたセットアップが終了しません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6426	変数を解決する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6425	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6424	プロジェクトはマイナーアップグレードアイテムを含みますが、現在のリリースはセットアップランチャーを使用しません。このマイナーアップグレードを適切に作動させるためには、アップグレードコマンドラインを手動で指定しなくてはなりません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6423	プロジェクトには、セットアップ %1 を参照するマイナー アップグレード アイテムが含まれています。ただし、参照されているセットアップはメジャー アップグレードによってのみアップグレードが可能です。メジャーアップグレードアイテム、または自動アップグレードアイテムを利用してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6422	%1 にメジャーアップグレード設定を追加中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6421	セットアップ %1 がメジャーアップグレードかどうかを判断中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6420	アップグレードアイテムセットアップ [MSI 名].msi が見つかりません。アップグレード設定を作成できません。	セットアップが存在すること、またそれが .msi ファイルであること、そして名前が正しいことを確認してください。
-6419	アップグレード項目 %1 を処理中に予期しないエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6418	アップグレードする項目を処理中に予期しないエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6415	パッチパッケージを作成中にエラーが発生しました。ログファイル '%1' の内容を出力ウィンドウに書き込み中です。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6414	バージョンが付いていないファイル %1 のファイルハッシュを入力中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6413	パッチ作成処理が、エラーコード %1 を戻しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6412	パッチパッケージを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6411	%1 のファイルバージョンを設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6410	ファイル %s をリリースの保存場所にコピーする際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6409	ファイル %1 のソースパスをビルド中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6408	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6407	アップデートされたイメージの PackageCode を設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6406	アップデートされたイメージで製品バージョンを設定中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6405	アップグレードされたイメージを準備中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6404	パッチ構成プロパティファイルの準備中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6403	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6402	ビルド用のリリースフォルダーを準備中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6401	オリジナル セットアップファイルをリリースフォルダーにコピーできません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6400	パッチ構成プロパティ テンプレートをリリースフォルダーにコピーできませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–6309	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6308	mscorsn.dll がシステム上にありません。[ツール] の [オプション] を使って mscorsn.dll へのパスを設定することができます。ファイルは Microsoft .NET Framework の再配布可能ファイルの一部です。%1。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6307	'%1' へ単一または複数のファイルを抽出できません。ファイルパスが、オペレーティング システムの制限より長くなっています。現在のリリースのビルド場所を短いパスに変更してこの問題を解決してください。	現在のリリースのビルド場所を短いパスへ変更してください。
–6306	IIS 仮想ディレクトリの構成用に指定したポート番号が無効です。	[IIS 構成] ビューで仮想ディレクトリに指定した TCP ポート番号が有効であることを確認してください。詳細については、「TCP ポート番号とサイト番号の構成」を参照してください。
–6305	アセンブリ "%1" を次の理由でロードできませんでした: "%2"	依存関係をロードできること、またアセンブリがアセンブリバインディング規則に従ってアクセス可能であることを確認してください。詳細については、.NET Framework SDK ヘルプまたは MSDN に掲載されている「How the Runtime Locates Assemblies」を参照してください。
–6304	CUB ファイル "%1" の検証エラー。	CUB ファイルが有効であることを確認してください。
–6303	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6302	InstallShield は Visual Studio .NET プロジェクト出力 "%1" をコンポーネント "%2" から解決できませんでした。プロジェクトを Visual Studio .NET でビルドする、もしくは Visual Studio .NET コマンドライン (devenv /build ConfigName [/project ProjName] [/projectconfig ConfigName] SolutionName) を使用してビルドしなくてはなりません。	プロジェクトを Microsoft Visual Studio または Visual Studio コマンドラインを使ってビルドします。
-6274	Setup.exe が次の Disk1 の場所に見つかりませんでした: %1。に見つかりません。ビルドテーブルのみを実行している場合は、リリースの前の完全ビルドが完了しているか確認してください。	このエラーは、セットアップが Setup.exe からスタンプを取得しようとしたにもかかわらず、Setup.exe が Disk1 フォルダーに存在しなかった場合に発生します。これは完全ビルドを途中でキャンセルしてから [ビルドテーブルのみ] オプションを実行した場合に発生します。
-6273	VB スクリプトカスタムアクションがプロジェクトにありますが、VBScriptRuntime マージモジュールが組み込まれていません。	[再配布可能ファイル] ビューを使用して、VBScriptRuntime マージ モジュールをプロジェクトに追加します。
-6271	ファイル %1 が見つかりません。このファイルの MsiFileHash テーブルをビルド中にエラーが発生しました。指定した場所にファイルが存在することを確認します。	指定した場所に識別されたファイルが存在することを確認します。 ファイルに定義済みフォルダー VSSolutionFolder が含まれる場合は、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。
-6270	アイコンテーブルのレコード %1 が %2 文字制限を超えています。そのためビルドでデータベースを維持できません。	レコードが制限文字数内であることを確認してください。
-6269	ディレクトリ %1 を %2 へコピー中にエラーが発生しました。ソース ディレクトリのパスが正しいことを確認してください。	ソース ディレクトリのパスを確認してください。
-6268	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6267	cab ファイル %1 を %2 の場所に抽出中にエラーが発生しました。	この場所が正確か確認してください。
-6266	%1 の CAB 抽出エンジンを作成および初期化中にエラーが発生しました	このエラーは、1 つ以上のファイルが登録されていない場合に発生することがあります。すべてのファイルが登録されていることを確認してください。
-6265	機能 '%1' をビルド中にエラーが発生しました。この機能はプロジェクトの別の機能と同じ名前で、大文字小文字が違います。InstallScript MSI プロジェクトでは機能名を重複できません。機能の 1 つの名前を変更してこのエラーを修正してください。	機能の 1 つを削除するか名前を変更してください。
-6264	%1 テーブルのレコードは文字列 ID '%2' を列 '%3' に使用していますが、この文字列は空白で、列はヌル不可能です。	列で使用する文字列を指定してください。
-6263	使用可能な InstallScript カスタム アクションの最大数 (1,000) を超えました。いくつかの InstallScript カスタム アクションを削除してから、セットアップを再ビルドしてください。	このエラーは、基本の MSI、DIM、InstallScript MSI、またはマージ モジュール プロジェクトで InstallScript カスタム アクションの数が 1,000 を超えた場合に表示されます。[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) を使って InstallScript カスタム アクションを削除し、プロジェクトに含めることが可能な最大数以下に抑えます。
-6262	Directory テーブルにエントリ %1 が含まれています。この識別子は予約済みです。Directory テーブルでこれを定義すると、実行時に予期せぬ結果が生じます。ダイレクト エディターを使用してこのディレクトリ識別子を変更してください。	ダイレクト エディターの Directory テーブルに移動して、ディレクトリ識別子 を変更してください。
-6260	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6259	証明書の詳細がみつかりません。MSIX/APPX パッケージをインストールするには、デジタル署名が必要です。	MSIX/APPX パッケージをインストールするには、パッケージをデジタル署名する必要があります。デジタル署名がない場合はインストールが失敗します。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6258	ファイル「%1」からデジタル署名情報を抽出中にエラーが発生しました。IDE に提供されたデジタル署名情報が正しいか確認してください。	デジタル署名情報が正しいことを確認してください。[リリース] ビューまたはリリース ウィザードの [デジタル署名] パネルでデジタル署名情報にアクセスできます。
-6257	PRODUCT_NAME が文字列テーブルで定義されていることをビルドエンジンが検出しました。実行時に IDE で定義された製品名値ではなく、この値が製品名として表示されます。	この警告メッセージは、ID PRODUCT_NAME が [文字列エディター] ビューで定義されているときに発生します。この文字列テーブル ID は [一般情報] ビューで、または [リリース] ビューの製品構成に設定された製品名の値を上書きします。
-6255	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6254	ファイル %1 の MsiFileHash テーブルをビルド中にエラーが発生しました	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6253	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6252	内部ビルドエラー	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6251	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6250	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6249	コンポーネント %1 はグローバルアセンブリキャッシュへをインストールするように設定されていますが、単一または複数のサブディレクトリを含みます。	このエラーは、サブディレクトリを含む .NET プロジェクト出力を使用する時に発生します。InstallShield は、Global Assembly Cache へのこれらのファイルのインストールを現在サポートとしていません。このエラーを回避するには、コンポーネントのディレクトリを変更 (たとえば INSTALLDIR へ) するか静的にファイルを追加して設定します。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6248	コンポーネント %2 の依存ファイル %1 が見つかりませんでした。	リリースの場所が *<ISProjectDataFolder>* または *<ISProjectFolder>* の場合、[テーブルのビルドとファイルのリフレッシュ] を利用してリリースをビルドします。詳細については、.NET Framework SDK ヘルプまたは MSDN に掲載されている「How the Run time Locates Assemblies」を参照してください。
-6247	.NET Framework 再配布可能ファイル %1 がシステムに見つかりません。%2	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6246	dotnetfx.exe をビルド済みイメージに追加することができませんでした。.NET 1.1 を含む場合、コア言語が選択されていることを確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6245	1 つ以上のプロジェクトに、.NET Framework を必要とする .NET 属性が含まれています。リリースに .NET Framework を含めることをお勧めします。	プロジェクトに.NET Framework 再配布可能ファイルを追加することを検討してみてください。.NET Framework は、実行時にターゲット システムに存在しないとき、インストール時にインストールされます。詳細については、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」を参照してください。
-6244	1 つ以上のプロジェクトに、.NET Framework を必要とする .NET 属性が含まれています。ただし .NET Framework を検出することができません。%2	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6243	InstallUtilLib.dll がシステムに見つかりません。このファイルはインストーラーカスタム アクションに必要です。[ツール] の [オプション] を使って InstallUtilLib.dll へのパスを設定することができます。ファイルは Microsoft .NET Framework の再配布可能ファイルの一部です。%1	[ツール] メニューから [オプション] を選択し、.NET タブをクリックしてパスを InstallUtilLib.dll に設定します。
-6242	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6241	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6240	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6239	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6238	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6237	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6236	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6235	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6234	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6233	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6232	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6231	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6230	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6229	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6228	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6227	コンポーネント %1 の .NET スキャンに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–6226	Visual Studio .NET ソリューションを開いています。	ソリューションが指定の場所に存在するか確認してください。ソリューション情報を見つけるには、ダイレクト エディターの InstallShield テーブルに移動して、ISDotNetSolution の値を見つけます。 さらに、Visual Studio がビルド マシンにインストールされていることを確認してください。
–6225	Visual Studio .NET プロジェクト出力 "%1" を解決しています	プロジェクトがソリューションに存在するか確認してください。ソリューション情報を見つけるには、ダイレクト エディターの InstallShield テーブルに移動して、ISDotNetSolution の値を見つけます。さらに、そのプロジェクトに指定された出力があるか確認してください。
–6224	機能 %2 のマージ モジュール %1 を処理しています	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6223	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6222	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6221	コンポーネント %2 からの Visual Studio .NET プロジェクト出力 "%1" を解決できませんでした。	プロジェクトがソリューションに存在するか確認してください。ソリューション情報を見つけるには、ダイレクト エディターの InstallShield テーブルに移動して、ISDotNetSolution の値を見つけます。さらに、そのプロジェクトに指定された出力があるか確認してください。
–6220	動的に取得された %1 データは、コンポーネント %2 に関連付けられているスタティック データと競合しています。ダイナミック データで上書きします。	このエラーは、MsiAssembly と MsiAssemblyName テーブルに手動で入力されたエントリが、スキャンの結果得られた .NET アセンブリのプロパティと競合する場合に発生します。このエラーを除去するには、コンポーネントの [ビルド時に .NET をスキャン] プロパティを [なし] に設定するか、MsiAssembly と MsiAssemblyName テーブルに手動追加された値を削除します。
–6219	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6218	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–6217	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6216	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6215	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6214	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6213	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6212	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6211	コンポーネント %1 のインストール先は GlobalAssemblyCache ですが、キーファイルが .NET アセンブリではありません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6210	コンポーネント %1 の COM .NET Interop 情報をビルド中にエラーが発生しました。	InstallShield は、.NET.dll の regasm.exe を呼び出して .reg ファイルを作成します。これによって .reg ファイルが作成されます。その後 .reg ファイルがインポートされます。これらの手順のいずれかが失敗すると、エラー -6210 が発生します。コンポーネントのキーファイルが .NET .dll で、.NET アセンブリに [エクスポートするタイプ] があることを確認してください。
-6209	Regasm.exe がシステムに見つかりません。このファイルではコンポーネントの .NET COM Interop 情報を取得する必要があります。[ツール] の [オプション] を使って Regasm.exe へのパスを設定することができます。ファイルは Microsoft .NET Framework の再配布可能ファイルの一部です。%1	[ツール] メニューから [オプション] を選択し、.NET タブをクリックしてパスを Regasm.exe に設定します。
-6208	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6207	セットアップ ファイル %s をビルド中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6205	コンポーネント %1 をビルド中にエラーが発生しました。コンポーネントのインストール先ディレクトリ %2 が directory テーブルにありません。コンポーネントのインストール先を有効な場所に変更してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6204	コンポーネント %2 に %1 をインポート中にエラーが発生しました。指定の場所にファイルが存在することと、ファイルが有効な REG ファイルであるかどうかを確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6203	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6202	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6201	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6200	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6199	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6197	%1 を %2 へストリーム中に、エラーが発生しました。.msi パッケージが、現在別のプロセスで使用されていないことを確認してください。	.msi パッケージが現在別のプロセスで使用されていないことを確認してください。
-6196	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6195	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6194	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6193	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6192	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6191	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6190	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6189	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6188	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6187	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6186	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6185	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6184	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6183	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6182	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6181	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6180	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6179	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6178	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6177	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6176	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6175	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6174	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6173	ファイル %1 の名前を %2に変更中にエラーが発生しました。	interm フォルダーを含む製品構成のドライブに十分な空き容量があるか確認してください (たとえば、¥MySetups¥Your Project Name-27¥Product Configuration 1¥Interm)。
-6172	CAB ファイルを作成中にエラーが発生しました増分の圧縮ビルドには、プロジェクトに確実に最低 1 つのファイルを含む、またはメディアを再ビルド行ってください。	プロジェクトに最低 1 つのファイルを含むか、メディアの再ビルドを行ってください。
-6171	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6170	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6169	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6168	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6167	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6166	概要情報ストリームの Word Count Summary プロパティを更新中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6165	テーブル %1 をエクスポート中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6164	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6163	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6162	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6161	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6160	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6159	ソースプロジェクトとターゲットデータベースで、テーブル列のデータ型が一致しないものがあります。	ダイレクト エディターを使って、指定の列とテーブルでデータ型が正しいことを確認してください。
-6158	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6157	テーブル中のターゲットフィールドが、データを収めるには小さすぎます。	ダイレクト エディターを使って、割り当てられたテーブルフィールドにデータが収まるようにしてください。
-6156	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6155	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6154	ビルドがユーザーによってキャンセルされました。	このエラーが発生するのは、ビルド プロセスの途中でビルドが終了した場合のみです。
-6153	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6152	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6151	ターゲットデータベースを保存できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6150	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6149	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6148	指定されたテーブルにレコードを挿入できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6147	テーブルの指定ターゲットフィールドを、更新できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6146	指定されたテーブルで、新規レコードを作成できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6145	テーブルの指定列から値を取り出せませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6144	テーブル %1 のデータベースビューを開くことができませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6143	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6142	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6141	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6140	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6139	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6138	Setup.rul のコンパイルエラーが発生しました。	Setup.rul はプロジェクトに必須です。このファイルがプロジェクトに収められているか確認してください。
-6137	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6136	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6135	指定された機能が、セットアップの種類と関連付けられていませんでした。	指定された機能を、セットアップの種類と関連付けてください。
-6134	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6133	現在の製品構成で指定されたプロパティを更新中にエラーが発生しました。	製品構成の指定プロパティを調べて、それが有効であるか確認してください。
-6132	Storages テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
-6131	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6130	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6129	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6128	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6127	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6126	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6125	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6124	未定義の例外。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6123	未定義の例外。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6122	ビルドにはコンポーネントが含まれていません。	ビルドには、少なくとも 1 つのコンポーネントを含める必要があります。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6121	ビルドしたマージ モジュールをモジュールフォルダーにコピーする際に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6120	リソースリンカが、指定された DLL のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6119	リソースリンカが見つかりませんでした。	[ツール] メニューから [オプション] ダイアログを開いて、場所にリソースのリンカを設定します。
-6118	リソース コンパイラが、DLL のリンクに必要な指定 RES ファイルのビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6117	リソースコンパイラが見つかりませんでした。	[ツール] メニューから [オプション] ダイアログを開いて、場所にリソースのリンカを設定します。
-6116	次のプロジェクトからの rc ファイルのエクスポートに失敗しました: %1。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6115	ISSetup.dll からバージョン情報を取得することができません。	このエラーは、次の場所にある ISSetup.dll ファイルが破損している場合、またはこれがビルド マシンから削除されている場合に発生することがあります。このファイルは必ず次の場所に配置します。 InstallShield Program Files フォルダー¥redist¥Language Independent¥i386 このエラーを解決するには、InstallShield の修復を実行してください。
-6114	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6113	増分ビルド中に、エラーが発生しました。	[テーブルのみをビルドする] および [テーブルをビルドしてファイルを更新する] を選択する前に、前のメディアがビルドされたか確認し、また前のメディアが削除されていないことを確認してください。
-6112	Setup.exe にストリームしているファイルの削除でエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6111	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6110	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6109	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6108	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6107	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6106	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6105	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6104	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6103	ファイル %1 が見つかりませんでした。	指定した位置にファイルが存在すること確認してください。 ファイルに定義済みフォルダー VSSolutionFolder が含まれる場合は、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。
-6102	"%1" に一致するダイナミック ファイルを検索中にエラーが発生しました。	指定したダイナミックファイルのフラグが有効であることを確認してください。
-6101	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6100	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6099	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6098	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6097	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6096	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6095	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6094	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6093	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6092	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6091	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6090	コントロール '%2' に指定されたスクロール可能テキスト ファイル '%1' が存在しません。	指定位置 %1 に置かれたテキスト ファイルに .rtf ファイルの完全なパスがあることを確認してください。これが空白である場合、コントロール %2 の ISBuildSourcePath 列が空白になっています。スクロール可能テキストコントロールについては、Control テーブル中の ISBuildSourcePath を有効なファイルにしてください。
-6089	コントロール:'%1' の テキスト列と ISBuildSourcePath 列の両方にパス情報が含まれています。テキスト列を使用します。	コントロール %1 に、Control テーブルの ISBuildSourcePath 列 と Text 列 の両方のエントリを指定することはできません。ダイレクト エディターを起動し、ISBuildSourcePath 列か Text 列 から不要な情報を削除してください。
-6088	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6087	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6086	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6085	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6084	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6083	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6082	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6081	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6080	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6079	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6078	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6077	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6076	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6075	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6074	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6073	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6072	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6071	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6070	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6069	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6068	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6067	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6066	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6065	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6064	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6063	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6062	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6061	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6060	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6059	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6058	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6057	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6056	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6055	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6054	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6053	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6052	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6051	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6050	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6049	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6048	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6047	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6046	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6045	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6044	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6043	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6042	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6041	内部ビルド エラー。	このビルド エラーは、プロジェクトに複数言語のサポートが含まれていて、未使用のディレクトリを .msi パッケージの Directory テーブルから削除するようにリリースが構成されている場合に発生します。 問題を解決するために、次の操作を実行します。 1. **[メディア]** の下にあるビュー リストで、**[リリース]** をクリックします。 2. **[リリース]** エクスプローラーで、このエラーが発生したときにビルド中であったリリースをクリックします。 3. **[ビルド]** タブをクリックします。 4. **"未使用のディレクトリを保持する"** 設定で、**[はい]** を選択します。 詳細については、「リリースの [ビルド] タブ」を参照してください。
-6040	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6039	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6038	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6037	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6036	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6035	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6034	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6033	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6032	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6031	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6030	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6029	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6028	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6027	InstallShield は Visual Studio .NET 出力ファイル用の File テーブル内の ISBuildSourcePath 列の値を変換できません。InstallShield が Visual Studio .NET 内で実行されていません。 *または* InstallShield は Visual Studio .NET 出力ファイル用の File テーブル内の ISBuildSourcePath 列の値を変換できません。開いている Visual Studio .NET ソリューションの名前が .ism プロジェクトに格納されているソリューション名と異なります。	この警告を避けるには、Microsoft Visual Studio からアップグレードを実行します。 *または* .ism プロジェクトを InstallShield で作成する際、それを同一のソリューション内で開きます。
-6026	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6025	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6024	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6023	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6022	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6021	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6020	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6019	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6018	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6017	ビルドは COM 情報を抽出できませんでした。管理者として実行していることを確認してください。	InstallShield で COM 情報を抽出するためには、管理者権限が必要です。詳細については、「InstallShield を管理者権限を使って、または管理者権限を持たずに起動する違い」を参照してください。
-6016	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6015	次のファイルの短いファイル名を作成中にエラーが発生しました: %1。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6014	コンポーネントのショートカットがアイコンリソースを参照していないため、無効です。	これを解決するには、アイコンファイルとアイコンインデックスを、ショートカットに指定してください。
-6013	指定されたコンポーネントを作成するためのコンポーネント条件が無効です。	[コンポーネント] ビューから、指定されたコンポーネントの [条件] プロパティを修正してください。
-6012	言語 %1 のコードページを設定できませんでした。	指定した言語用のコードページをインストールしてください。
-6011	文字列テーブルを開けませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6009	Internet Explorer の Web メディアの作成中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6008	Internet Explorer の Web メディアの作成中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-6007	指定された非圧縮サポート ファイルを Disk1 へコピー中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6006	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6005	パッケージ定義ファイルを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6004	Msi エンジンのデジタル証明書の Setup.exe へのストリーム中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6003	"%1" の Setup.exe へのストリーム中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6002	オブジェクトのカスタム ビルド セットアップを実行中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6001	ISObjectProperty の前処理中に、エラーが発生しました。すべての項目に有効な IncludeInBuild タグが付いているか、確認してください。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-6000	補足マージ モジュール情報の収集中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5099	ビルド レポートを作成中にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5098	Directory テーブルに循環参照が含まれています。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5097	_Validation テーブルにカスタムレコードを書き込めませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5096	標準テーブルリストを読み出せませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5095	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–5094	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5093	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5092	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5091	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5090	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5089	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5088	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5087	最初のエラー発生時に中止しました。	ユーザーが、[ツール] の [オプション] メニューで [最初のエラーの発生時にビルドプロセスを停止] を選択しています。エラーが発生したため、ビルドを中止しました。
–5086	警告をエラーとして処理します。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5085	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5084	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5083	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5082	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5081	Setup.ini に書き込めませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5080	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–5079	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5078	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5077	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5076	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5075	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5074	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5073	バイナリキー %1 がありません。	ダイレクト エディターで Binary テーブルを開き、該当するエントリが存在することを確認します。
–5072	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5071	標準 DLL 関数構文エラー。	関数が何も返さない場合、void が指定されているか確認してください。カスタム アクションの "関数署名" フィールドで省略記号ボタン (...) クリックして、署名を検証します。
–5070	標準 DLL 関数名構文エラー。	カスタム アクションの "関数署名" フィールドで省略記号ボタン (...) クリックして、署名を検証します。
–5069	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5068	標準 DLL 関数パラメーター構文エラー。	カスタム アクションの "関数署名" フィールドで省略記号ボタン (...) クリックして、署名を検証します。
–5066	標準 DLL 関数戻り値の構文エラー。	カスタム アクションの "関数署名" フィールドで省略記号ボタン (...) クリックして、署名を検証します。
–5065	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5064	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–5063	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5062	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5061	指定されたファイル名は、既に存在します。	コンポーネントの "ソースの場所" プロパティを使うと、このエラーは発生しません。
–5060	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5059	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5058	マージ モジュールの依存関係を取得できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5057	マージ モジュールカタログを取得できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5056	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5054	ファイル "%1" のサイズを判別できませんでした。	指定したファイルが存在することを確認してください。
–5053	ファイル "%1" が見つかりませんでした。	指定したファイルが存在することを確認してください。
–5052	ボリューム %1 の空き容量を判別できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5051	ファイル "%1" から読み取り専用の属性を削除できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5050	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5049	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5048	ファイル "%1" を作成できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-5047	ディレクトリ %1 を作成できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5046	以前のビルドレポートを保存できませんでした。	このビルド エラーは、リリースが存在するが、対応するリリース フォルダー内の Report Files フォルダーが存在しない時に発生します。この状況は一般的に、Report Files フォルダーを手動で削除しているが、ディスク イメージ フォルダーを削除していない場合に発生します。この状況はまた、プロジェクトを InstallShield の以前のバージョンからアップグレードしている場合にも発生します。 このエラーを解決するには、[リリース フォルダーを開く] ボタンをクリックします。現在のリリース フォルダーが開きます。そのフォルダー内のすべてのファイルを削除します。その後、リリースを再ビルドします。
-5045	以前のビルドログを保存できませんでした。	このビルド エラーは、リリースが存在するが、対応するリリース フォルダー内の Log Files フォルダーが存在しない時に発生します。この状況は一般的に、Log Files フォルダーを手動で削除しているが、ディスク イメージ フォルダーを削除していない場合に発生します。この状況はまた、プロジェクトを InstallShield の以前のバージョンからアップグレードしている場合にも発生します。 このエラーを解決するには、[リリース フォルダーを開く] ボタンをクリックします。現在のリリース フォルダーが開きます。そのフォルダー内のすべてのファイルを削除します。その後、リリースを再ビルドします。
-5044	ディレクトリ %1 を削除できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5043	ボリュームが、存在しません。	[リリース] ビューを調べて、指定したボリュームが存在することを確認してください。
-5042	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5041	コンポーネントまたは機能のインストール先の指定に、文字列 ID "%1" が使用されています。	インストール先の指定に、文字列 ID は使えません。この文字列 IDを使っているコンポーネントまたは機能を検索し、ディレクトリ識別子を使ってインストール先を変更してください。
-5040	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
–5039	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5038	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5037	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5036	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5033	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5032	ListView テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5031	ListBox テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5030	ComboBox テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5029	CheckBox テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5028	RadioButton テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5027	Control テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5026	Dialog テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
–5025	パッケージを保存できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5024	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
–5023	File テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-5022	Feature テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
-5021	Component テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
-5020	Directory テーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、テーブル中のデータが正しいかどうかを確認してください。
-5019	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5018	論理ディスクに、機能が 1 つも含まれていません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5017	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5016	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5015	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5014	アイコン "%1" をビルド中にエラーが発生しました。指定されたアイコンキーがアイコンテーブルに見つかりませんでした。	このエラーの詳細については、ナレッジベース記事 Q107116 を参照してください。
-5013	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5012	%1 の Feature_ 値が無効です。	ダイレクト エディターを使って、指定したテーブルを開き、テーブル中のデータが有効であることを確認してください。
-5011	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5010	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5009	スキーマ概要ストリームは、200 以上である必要があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-5008	概要ストリームのテンプレートで Intel64 を指定する必要があります。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5007	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5006	"%1" を保存できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5005	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5004	プロジェクトを書き込みアクセスで開けませんでした。	プロジェクトが IDE で既に開かれていないか確認してください。
-5003	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5002	内部ビルド エラー。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-5001	setup.ini をコピーできませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4701	One-Click Install Web ページをビルド中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4371	InstallScript のコンパイルで警告が発生しました。	InstallScript デバッガーを使って、InstallScript を順に追って、問題点を見つけてください。
-4370	InstallScript のコンパイルでエラーが発生しました。	InstallScript デバッガーを使って、InstallScript を順に追って、問題点を見つけてください。
-4354	ビルドが、コンポーネント中のファイルから COM 情報を抽出できませんでした。	ファイルが自動登録されること、および自動登録プロセスに問題がないか確認してください。
-4352	指定されたコンポーネントのキーファイルがありません。ダイナミックな COM 抽出にはキーファイルが必要です。	指定したコンポーネントのキーファイルを、[コンポーネント] ビューで設定してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-4350	動的に取得された MIME テキスト/スクリプトレットは、コンポーネント %1 に関連付けられているスタティック データと競合しています。ダイナミック データで上書きします。	この警告は、コンポーネント ウィザードを使って COM サーバー .dll ファイルを追加した場合に発生します。コンポーネント ウィザードは静的な COM 抽出を行い、後でコンポーネントの "ビルド時に COM 抽出" 設定を [はい] に変更します。動的に抽出されたデータは、.ism ファイルのスタティックデータと競合します。COM 抽出設定が正確であることを確認してください。
-4349	動的に抽出された COM コンポーネントの %1 情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4348	動的に抽出された COM コンポーネントの %1 情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4347	動的に抽出された COM コンポーネントの情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4346	動的に抽出された COM コンポーネントの情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4345	動的に抽出された COM コンポーネントの情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4344	動的に抽出された COM コンポーネントの情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4343	動的に抽出された COM コンポーネントの情報のビルドに失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4340	COM 抽出モジュールの初期化に失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-4327	文字列 ID “%1” を InstallScript 文字列ファイルに書き込めませんでした。	このエラーは、InstallScript を含むプロジェクトに関連しています (InstallScript MSI プロジェクトか、InstallScript カスタム アクションを持つ基本の MSI プロジェクト)。ビルド時に InstallShield はプロジェクト内のすべての文字列エントリを含む .ini ファイルを作成します。エラーメッセージで、“%1” は書き込みに失敗した .ini ファイル内の行番号を表します。行番号が 0 の場合、.ini ファイルに書き込む、または作成する一般エラーが発生したことを示します。 。これは、ディスク容量不足を表していることがあります。
-4303	指定されたコンポーネントでファイルキーを同期させる際、予期しないエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4302	コンポーネント %2 でファイル キー %1 を同期させる際に、競合が発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4301	キー同期用の .msi ファイルが見つかりませんでした。	指定の MSI ファイルが存在することを確認してください。
-4093	ソース %1 をターゲット %2 にコピーできません	ファイルが存在すること、ターゲット先に十分なディスク領域があること、その場所にコピーできることを確認してください。
-4092	MSI データベース %1 を開く際に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-4075	ファイルが見つかりません。機能 ‘%2’ のモジュール ‘%1’ をマージ中にエラーが発生しました。	マージ モジュール検索パスにマージ モジュールが存在することを確認してください。 このエラーは、InstallScript MSI オブジェクトを含むインストール プロジェクトが InstallShield 2025 にアップグレードされた場合に発生することがあります。InstallShield 2025 では現在 InstallScript MSI オブジェクトを使用できません。この場合、[再配布可能ファイル] ビューを使って、プロジェクトからそのオブジェクトを削除します。InstallScript MSI オブジェクトを InstallShield 2025 を使ってマージ モジュール プロジェクトに変換してアップグレードすることができます。変換後、マージ モジュールをインストール プロジェクトに追加します。
-4072	%1 の依存関係の取得中に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

テーブル 2 - ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-4006	ファイルを削除できません。	指定ファイルが存在し、読み取り専用になっていないことを確認してください。
-3876	Summary Stream の不正なテンプレート %1 を無視しています。	InstallShield は、"テンプレートの概要" 設定に無効なエントリが含まれているときに、この警告を生成します。有効な値を入力する方法については、「"テンプレート概要" プロパティを使用する」を参照してください。
-3851	指定されたテーブルのビルドで、エラーが発生しました。	ダイレクト エディターを使って、指定したテーブルを開き、テーブル中のデータが有効であることを確認してください。
-3713	関数ブロックは "モジュール::関数" の形式である必要があります。	関数ブロックを "モジュール::関数" の形式に変更してください。
-3204	アイコン用の指定ファイルから、指定されたインデックスでアイコンを抽出できません。	アイコンを収めたファイルが存在すること、およびファイル中に該当するインデックス番号があることを確認してください。
-3138	指定したパス構成がプロジェクトに存在しません。パス構成が存在すること、大文字と小文字の区別を含んでスペルに誤りが無いことを確認してください。	IPWiProject オブジェクトにある BuildPatchConfiguration メソッドを使うとき、このエラーが発生する場合があります。
-3114	アプリケーションパス %1 は、関連コンポーネント %2 の宛先フォルダーを含んでいません。	このエラーは、アプリケーション パスのエントリが [コンポーネント] ビューの [詳細設定] セクションに作成されてもアプリケーション パス が指定されなかった場合に発生します。これを訂正するには、アプリケーション パスのエントリを設定します。 アプリケーション パスはレジストリ エントリでもあるため、次のようなレジストリ パスが作成されていてパス エントリがない場合、このエラーが発生する可能性があります。 HKEY_LOCAL_MACHINE¥Microsoft¥Windows¥CurrentVersion¥App Paths これを訂正するには、パスエントリを設定します。
-3028	指定されたテーブルの文字列 ID の置換中に、エラーが発生しました。	指定されたテーブルで、必要な文字列 ID が空欄にされています。文字列 ID を指定してください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-3016	Binary テーブル %1 をパッケージへ追加する際にエラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-2200	ファイルを上書きできません。	指定のファイルが読み取り専用でないことを確認してください。
-1531	ディスクに指定されたサイズが、この機能には小さすぎます。	リリース ウィザードで、[メディアのフォーマット] 設定を大きなサイズに変更してください。
-1530	ディスクに指定されたサイズが、このファイルには小さすぎます。	リリース ウィザードで、[メディアのフォーマット] 設定を大きなサイズに変更してください。
-1527	プロジェクトにファイルが含まれていません。	この警告は、プロジェクト中に 1 つもファイルが入っていない場合に発生します。[ファイルとフォルダー] ビューを使ってプロジェクトにファイルを追加すると、このエラーは発生しません。
-1505	CAB ファイルを MSI パッケージに追加できませんでした。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-1501	"%1" を "%2" に圧縮できませんでした。	IsCmdBld.exe が必ず相対ディレクトリで実行されるようにしてください。それにはエラーメッセージで指定されたファイルを見つけてそのディレクトリで IsCmdBld.exe を起動します。
-1119	指定した CUB ファイルが存在しません。	CUB ファイル名のつづりが正しいか確認してください。
-1027	%1 の署名に失敗しました。	現在のリリースに提供されているデジタル書名情報 (デジタルキーおよびパスワード) を確認してください。
-1024	ファイルが見つかりません。Binary テーブルにファイルをストリームできません。	指定したファイルが存在することを確認してください。
-1015	ソース '%1' をターゲット %2 にコピーできません。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-1014	ディレクトリの名前を変更できません。	Windows エクスプローラーまたは DOS プロンプトがリリース出力フォルダー (Disk1) のサブフォルダーまたは Interm フォルダーをポイント中かロックしている可能性があります。現在のディレクトリを変更してください。Disk1 フォルダー内の開かれているファイルを閉じてください。Msidb.exe が開いている場合は、閉じてください。

テーブル 2 • ビルド エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティング情報
-1013	指定されたファイルは、他のプログラムによって使われています。	現在このファイルを使っているアプリケーションを閉じて、ビルドプロセスを再実行してください。
-1009	ディスク容量が不足しているか、またはターゲットドライブが見つかりません。	ビルドターゲット上のディスク領域を増やすか、ビルドのターゲットを新規に選んでください。ターゲットドライブを指定できない場合は、新しいターゲットを選択するか、あるいはターゲットドライブのアクセス許可が正しく設定されているか確認してください。
-1007	ソース %1 をターゲット %2 にコピーできません	ファイルが存在すること、ターゲット先に十分なディスク領域があること、その場所にコピーできることを確認してください。
-1005	%1 を %2 に圧縮できませんでした(%1 はファイルへの完全パス、%2 は.cab ファイル名を表します)。	このエラーは、圧縮セットアップをビルドしていて空き容量が足りなくなった場合や、一時的スペースにセットアップをビルドしていた場合などに発生することがあります。また、CAB API に伴う内部エラーが原因の場合もあります。詳細については、テクニカル サポートにお問い合わせください。
-1001	MSI データベースを開く際に、エラーが発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-1000	製品構成が無効です。ディレクトリの作成に失敗しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。

メディア ビルド エラーと警告

ビルドが生成される前に発生するメディアビルドのエラーは、エラーメッセージダイアログ ボックスに表示されます。ビルドが生成されたときに発生するエラーは、[出力] ウィンドウに表示され、ログファイルに記録されます。

テーブル 3 • メディアビルドエラーと警告

エラー/警告番号	メッセージ
118	ファイル #number をもつ CAB ファイルが見つかりませんでした。システムで指定されたパスが見つかりません。
129	メディアが小さすぎます。ディスク番号のファイル名に収まりません。

テーブル 3 • メディアビルドエラーと警告 (続き.)

エラー/警告番号	メッセージ
-5000	ファイルグループ 'ファイルグループ名' が存在しないファイルへのリンクを含んでいます。 - 'ファイル名'
-5006	'EnterPassword' ダイアログリソースがリソースライブラリ filename に存在しません。
-5020	ビルドしたメディアファイルを URL サイトにアップロードできませんでした。URL を解析することができませんでした。
-5021	ビルドしたメディアファイルをフォルダー 'パス' にコピーできませんでした。
-5032	自己展開型実行可能ファイルの署名に失敗しました。
-5050	フォントデータをプロジェクトファイルから読み取ることができませんでした。'Font' テーブルが プロジェクトに存在していない可能性があります。
-5051	次のフォントデータファイルを作成することができませんでした: ‘データ ファイル パスと名前’
-7040	メディアには機能が含まれていません。
-7041	CABEngine のインスタンスを作成することができませんでした。CABEngine コンポーネントが適切にインストールまたは登録されていない可能性があります。InstallShield を再インストールする必要があるかもしれません。
-7043	差分ビルド用の次のメディアを開くことができませんでした - ″メディアヘッダー ファイルへのパス″
-7044	[プロジェクトの設定] プロパティシートで製品バージョンが指定されていません。
-7045	ファイルが見つかりませんでした - ″InstallShield プログラム ファイル フォルダー ¥Redistributable¥Compressed Files¥Language Independent¥OS Independent¥Corecomp.ini″
-7126	十分な空きディスク容量がありません。
-7127	ディスクイメージディレクトリは別のプログラムが使用しています。
-7275	証明書が無効です。
-7276	証明書の期限が切れています。
-7380	ビルドがユーザーによってキャンセルされました。
-9008	オブジェクトプロパティの保存ができません。

エラー 118

メッセージ

ファイル #*number* をもつ CAB ファイルが見つかりませんでした。システムで指定されたパスが見つかりません。

トラブルシューティング情報

このエラーは既に存在するメディアをビルドしようとして、以前作成されたキャビネット ファイルの1つが既に存在しないとき発生します。

このエラーが発生した場合、そのメディアを再ビルドします。

エラー 129

メッセージ

メディアが小さすぎます。ディスク*番号*の*ファイル名*に収まりません。

トラブルシューティング情報

このエラーが発生する理由として、以下の原因が考えられます。

- 1 つのディスクでのファイルの合計サイズが、選択されたディスクサイズよりも大きい。
- インストールでは、[サポート ファイル] ビューに配置されているファイル (Setup.bmp や License.txt など) や、Data1.cab を含む特定のファイルが、最初のディスクに常駐される必要があります。また、次のファイルを含めた特定のファイルも Data1.cab に圧縮される必要があります。
  - セットアップに含めたオブジェクト用ファイル
  - インストールがサポートする各言語のリソースファイル
  - 選択したダイアログ ボックス スキン用のファイル

インストールがサイズの大きいスキン、セットアップファイルまたはオブジェクト (または、多数のサポート ファイル、オブジェクト、言語)を含む場合、最初のディスク用のファイルの合計サイズはリリース ウィザードの [メディア タイプ] パネルで選択したディスクサイズよりも大きくても構いません。

このエラーを防ぎ、ビルドを正常に完了するには、プロジェクトからスキンやサポート ファイルまたはオブジェクトを削除し、サポート ファイルのサイズを縮小するか、サイズの小さいファイルを持つスキンまたはさらに大きいディスクサイズを選択してください。

ディスク 1 以外のディスクがエラーメッセージによって指定された場合、コンポーネントの中のファイルは圧縮された後も、ファイルによっては大きすぎてディスクに収まらない場合があります。このような場合、大きいファイルを削除するか、サイズを縮小するか、さらに大きいディスクサイズを選択する必要があります。

- ドライブに十分な空き容量がない。

インストールのキャビネット ファイルを作成する時、リリースビルダーは、システムの一時フォルダーとビルド ロケーションに空き容量があることを必要とします。このエラーを防ぐには、システムの一時フォルダーが常駐するドライブと、ビルドの場所を指定したドライブに十分な空き容量があることを確認してください。

システムの一時フォルダーは、DOS プロンプトで "set" という語を入力して検出できます。一時フォルダーは、TEMP 環境変数の値によって識別されます。ビルドの場所を見つけるには、リリース ウィザードの [一般オプション] パネルの [詳細] ダイアログ ボックス、または [リリース] ビューの "リリースの場所" プロパティを見ます。

- 十分な仮想メモリがない。

Windows NT では、このエラーは、ハードディスクの空き容量が足りないのではなく、仮想メモリの空き容量が足りないことが原因の場合があります。さらに詳しい情報は、ナレッジベース記事 Q102757 を参照してください。

警告 -5000

メッセージ

コンポーネント '*コンポーネント名*' が存在しないファイルへのリンクを含んでいます。 - '*ファイル名*'

トラブルシューティング情報

示されたコンポーネントが存在しないファイルへのリンクを含んでいます。コンポーネントからそのファイルを削除するか、または、そのファイルが名前変更または移動されている場合、示された名前とロケーションに復元します。

警告 -5006

メッセージ

'EnterPassword' ダイアログリソースがリソースライブラリ *filename* に存在しません。

トラブルシューティング情報

セットアップ初期化中に [パスワード ダイアログを表示する] オプションが、リリース ウィザードの [パスワード] パネルまたは [リリース] ビューの "パスワード ダイアログの表示" プロパティで選択されていますが、1 つまたは 複数の _Isres.dll のコピーがこのダイアログ ボックス用のリソース テンプレートを含んでいません。この警告は、リソーステンプレートを含んでいない _Isres.dll の各コピーに対して別々の形で表示されます。この問題を解決するためには、InstallShield Premier Edition が必要です。

警告 -5020

メッセージ

ビルドしたメディアファイルを URL サイトにアップロードできませんでした。URL を解析することができませんでした。

トラブルシューティング情報

[リリース] ビューの "FTP ホスト アドレス" プロパティで指定した URL が存在することを確認してください。

警告 -5021

メッセージ

ビルドしたメディアファイルをフォルダー 'パス' にコピーできませんでした。

トラブルシューティング情報

[ビルドしたメディアファイルをフォルダーにコピーする] オプションが リリース ウィザードの [ポストビルド オプション] パネル または [リリース] ビューの "フォルダーにコピー" プロパティで選択されています。このオプションで指定したフォルダーに、ビルドしたメディア ファイルをコピーすることができませんでした。指定したパスが存在し、書き込み可能なのを確認してください。

警告 -5032

メッセージ

自己展開型実行可能ファイルの署名に失敗しました。

トラブルシューティング情報

ビルドで、リリース ウィザードの [デジタル署名] パネルまたは [リリース] ビューの "メディアに署名" プロパティで指定された自己展開型実行可能ファイルの署名に失敗しました。エラーの原因としては、プライベートキーファイルが無効または存在していないことが考えられます。

警告 -5050

メッセージ

フォントデータをプロジェクトファイルから読み取ることができませんでした。'Font' テーブルが プロジェクトに存在していない可能性があります。

トラブルシューティング情報

ダイレクト エディターを使用して、Font テーブルがプロジェクトに含められているか確認してください。テーブルが含まれていなく、かつソースコントロールソフトウェアを使用してもそのテーブルを含んだバージョンにロールバックできない場合、一度プロジェクトからフォントファイルを削除し再度それらをプロジェクトに含めてみてください。

警告 -5051

メッセージ

次のフォントデータファイルを作成することができませんでした: *'データ ファイル パスと名前'*

トラブルシューティング情報

指定されたパスが存在し書き込み可能であること、および同名の読み取り専用ファイルが既に同じロケーションに存在していないことを確認してください。

エラー –7040

メッセージ

メディアには機能が含まれていません。

説明

指定したメディアに機能が含まれていません。すべてのメディアビルドには最低1つの機能が含まれていなければなりません。

トラブルシューティング情報

[機能] ビューで、必ず最低1 つの機能が作成されていることを確認してください。

エラー –7041

メッセージ

CABEngine のインスタンスを作成することができませんでした。CABEngine コンポーネントが適切にインストールまたは登録されていない可能性があります。InstallShield を再インストールする必要があるかもしれません。

トラブルシューティング情報

この問題を解決するには、Windows システム フォルダーの中の Regsvr32.exe を利用するか、または以下のようにDOS コマンドを利用して、手作業で MediaBuild40.dll ファイルを(ビルドマシンの共通ファイルフォルダーの中のInstallShield¥MediaBuild フォルダーの中に) 登録します。

Regsvr32.exe "C:¥Program Files¥Common Files¥InstallShield¥MediaBuild¥MediaBuild40.dll"

エラー –7043

メッセージ

差分ビルド用の次のメディアを開くことができませんでした – *"メディアヘッダーファイルへのパス"*

トラブルシューティング情報

このメッセージは、リリース ウィザードの [アップデート] パネルにある [差分メディア] リスト ボックスで指定したメディア ヘッダー ファイルにのことを指しています。このエラーは、以下の条件によって発生します。

- ファイルが存在しません。
- ファイルが破損している。
- ファイルが互換性がないメディア。たとえば、InstallShield Professional 5.x でビルドされたメディア等。

エラー -7044

メッセージ

製品バージョンが [製品のプロパティ] ビューで指定されていません。

トラブルシューティング情報

[一般情報] ビューの "製品バージョン" 設定にヌル以外の値を指定なくてはなりません。

エラー -7045

メッセージ

ファイルが見つかりませんでした - "パス¥Corecomp.ini"

トラブルシューティング情報

Corecomp.ini が指定されたロケーションに見つかりませんでした。Corecomp.ini を移動した場合、または名前を変更した場合、そのコピーを Corecomp.ini というファイル名前で指定された場所に配置します。この方法で Corecomp.ini を復元できない場合、InstallShield の修復を実行します。

エラー -7126

メッセージ

セットアップをビルドすることができません。

説明

セットアップビルドに必要なディスク空き容量が不足しています。

トラブルシューティング情報

メディアウィザードを使用するには、システムの一時フォルダー内に空き容量が必要です。ビルドの場所およびこのドライブに十分な空き容量があることを確認してください。

エラー -7127

メッセージ

ディスクイメージディレクトリは別のプログラムが使用しています。

トラブルシューティング情報

このエラーが発生する理由として、以下の原因が考えられます。

- ビルドに使用するディスクイメージまたは Disk1 フォルダーが Windows Explorer または DOS で開かれている可能性があります。Windows Explorer または DOS ウィンドウで別のフォルダーを開くか、ウィンドウを閉じてメディアビルダーを再実行します。

- ディスクイメージまたは Disk1 フォルダーは読み取り専用です。Windows Explorer でフォルダーを右クリックして、[プロパティ] をクリックします。フォルダーが読み取り専用となっている場合、読み取り専用チェック ボックスをクリアしてからメディアビルダーを再実行します。
- メディアがビルドされているフォルダーの下に bldblack という名前のフォルダーが存在します。その名前のフォルダーを削除し、メディアビルダーを再実行します。

エラー -7275

メッセージ

証明書が無効です。

トラブルシューティング情報

プロジェクトで指定されたデジタル証明書が無効です。有効な証明書を指定して再ビルドしてください。

エラー -7276

メッセージ

証明書の期限が切れています。

トラブルシューティング情報

デジタル証明書の期限が切れています。現在の証明書を指定して再ビルドしてください。

エラー -7380

メッセージ

ビルドがユーザーによってキャンセルされました。

説明

ビルドがユーザーによってキャンセルされました。

トラブルシューティング情報

ユーザーが [キャンセル] ボタンをクリックしてビルド プロセスをキャンセルしました。

エラー -9008

メッセージ

オブジェクトプロパティの保存ができません。

トラブルシューティング情報

このエラーはプロジェクト内のオブジェクトが破損されているときに(たとえば、Isrt.dll が存在しない等)、発生します。

MSI/MSM 変換エラー

以下の表に、インストールパッケージ (.msi ファイル) またはマージ モジュール (.msm ファイル) を MSI/MSM オープンウィザードで変換する際に生じる各エラーについて、トラブルシューティングのヒントを示します。

このウィザードは、Windows Installer データベースにアクセスしてエラーを戻します。これらのエラーの詳細については、ビルド エラーと警告を参照してください。ソースの .msi または .msm パッケージが無効だと、ドキュメントに記載のない内部エラーが表示されることがあります。

ヒント: *MSI/MSM オープンウィザードを起動する前に、Microsoft Windows Installer プラットフォーム SDK の一部である Orca または Msival2.exe でパッケージを検証してください。*

テーブル 4 · MSI と MSM の変換エラー

エラー番号	メッセージ	トラブルシューティングのヒント
-6000	ファイルを開けません	.msi または .msm へのパスが、無効んも可能性があります。 ファイルは、他のプログラムによって使われている可能性があります。 ファイルが破損しているか、有効な Windows Installer データベースではない可能性があります。
-6001	データベース体系がサポートされていません	MSI/MSM オープンウィザードでは、スキーマ 30 以降と互換性のあるデータベースだけがサポートされています。.msi または .msm ファイルのスキーマを判断するには、ファイルを Orca で開き、**表示** メニューで **概要情報**を選択します。**概要情報の編集**ダイアログ ボックスの**スキーマ** フィールドを変更するだけで、このエラーを回避できることもあります。
-7000	レコードを作成できませんでした	レコードがすでに存在している場合、ウィザードは、ファイルで検出された複製エントリを作成できません。
-7001 **-7002** **-7004**	外部キーが無効です	.msi または .msm データベース内の外部キーによって参照されるレコードが無効か、または消失しているために、ウィザードはレコードを作成できませんでした。

テーブル 4 ・ MSI と MSM の変換エラー (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
-7005	文字列のプロパティを設定できません	フィールドがデータベースファイル内の複製であったために、MSI/MSM オープンウィザードは、文字列値を設定できませんでした。このエラーは、前の段階でレコード作成に失敗したことが原因である可能性があります (エラー -7000, -7001, -7002, -7004)。
-7006	整数のプロパティを設定できません	フィールドがデータベース内の複製であったために、MSI/MSM オープンウィザードは、整数値を設定できませんでした。このエラーは、前の段階でレコード作成に失敗したことが原因である可能性があります (エラー -7000, -7001, -7002, -7004)。
-7008	プロパティを設定できません	フィールドがデータベース内の複製であったために、MSI/MSM オープンウィザードは、値を設定できませんでした。このエラーは、前の段階でレコード作成に失敗したことが原因である可能性があります (エラー -7000, -7001, -7002, -7004)。
-7009	バイナリファイルを作成できません	ウィザードは、ファイルからバイナリ ファイル、アイコン、または、.rtf ファイルを抽出して、それを新しいプロジェクトの場所にコピーすることができませんでした。 抽出ディレクトリにある既存のファイルを削除します。 十分な空き容量があることと、抽出ディレクトリに適切な書き込み権限があることを確認してください。
-7010	cab 抽出用の出力パスが存在しません	ウィザードは、指定されたパスを使用して .cab ファイルを抽出することができませんでした。 .cab ファイルの出力パスをタイプする代わりに、**[参照]** をクリックしてパスを選択してから、もう一度ウィザードを起動します。
-7011	一時 .cab ファイルを削除できません	インストールパッケージまたはマージ モジュールから一時 .cab ファイル抽出した後に、そのファイルを削除できません。
1001	ソース MSI データベースパスが無効です	.msi データベースへのパスが有効であることを確認してください。
1017	プロセスが中断されました	変換プロセスを完了してください。

テーブル 4 ・ MSI と MSM の変換エラー (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
100001	予期せぬエラー	このエラーメッセージはそのエラーに関連する関数とテーブル名を表示します。
100002	ファイルをキャビネットから抽出できません	ソースが有効な .msi または .msm データベースであることを確認してください。
200001	出力パスが指定されませんでした	この警告は、コンバーターがデータベース テーブルまたはキャビネット ファイルからバイナリ ファイルを抽出しなければならないときに、MSI/MSM オープン ウィザードの **[ファイルの場所]** パネルにある [データ ファイルの場所] ボックスで、無効な出力パスが指定された場合に発生します。

内部エラー

次は内部エラーです。これらは、.msi または .msm データベースのインポート中に発生すべきではありません。MSI/MSM オープンウィザードを起動する前に、Microsoft Windows Installer プラットフォーム SDK の一部である Orca または Msival2.exe でパッケージを検証してください。

エラーメッセージには、変換が失敗したときのデータベーステーブル、フィールドおよび値が表示されます。

テーブル 5 ・ 内部エラー

エラー番号	メッセージ	トラブルシューティングのヒント
1002	無効なターゲットデータベースパス	デフォルトのプロジェクトの場所が有効であることを確認してください。パスを確認するには、**ツール** メニューで、**オプション**をクリックします。デフォルトのプロジェクト場所へのパスが **[ファイルの場所]** タブで指定されています。
1003	ソースの MSI/MSM データベースを開けませんでした	データベースがロックされていないか、また別のアプリケーションで開かれていないか確認してください。
1004	ターゲットデータベースを開けませんでした	プロジェクトファイルがロックされていないか、また別のアプリケーションで開かれていないか確認してください。 デフォルトのプロジェクトの場所が有効であることを確認してください。パスを確認するには、**ツール** メニューで、**オプション**をクリックします。デフォルトのプロジェクト場所へのパスが **[ファイルの場所]** タブで指定されています。

テーブル 5 • 内部エラー (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
1006	ソースまたはターゲット MSI/MSM データベースのテーブルリストを取得できませんでした	ソースが有効な .msi または .msm データベースであることを確認してください。
1007	フィールド取得に失敗しました	これは通常、MsiRecordGetXXX API のエラーが原因です。ソースが有効な Windows Installer データベースか確認してください。
1009	MsiCreateRecord エラー	プロジェクトファイルがロックされていないか、また別のアプリケーションで開かれていないか確認してください。
1010	フィールドのアップデートに失敗しました	これは通常、MsiRecordSetXXX API の失敗が原因です。 プロジェクトファイルがロックされていないか、また別のアプリケーションで開かれていないか確認してください。
1011	レコードをテーブルに挿入できませんでした	プロジェクトファイルがロックされていないか、また別のアプリケーションで開かれていないか確認してください。
1012	バイナリデータをソース MSI/MSM データベースから抽出できませんでした	抽出ディレクトリにある既存のファイルを削除します。 十分なディスク容量があること、また MSI/MSM オープン ウィザードの [**ファイルの場所**] パネルにある "**データファイルの場所**" フィールドに指定されている出力パスに対して適切な書き込み権限があることを確認してください。
1013	ターゲットデータベースへのファイルのストリームに失敗しました	プロジェクトファイルがロックされていないか、また別のアプリケーションで開かれていないか確認してください。
1014	MsiDatabaseCommit エラー	プロジェクトファイルがロックされていないか、また別のアプリケーションで開かれていないか確認してください。
5001	フィールド値を解決できませんでした	これは通常、予期せぬデータベーススキーマが原因で発生します。トラブルシューティングのエラー エラー –6001 を参照してください。

テーブル 5 ・ 内部エラー (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
5002	ソースとターゲットの MSI データベースのフィールド タイプに互換性がありません	ソースが有効な .msi または .msm データベースであることを確認してください。

Windows Installer 実行時のエラー

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *QuickPatch*
- *トランスフォーム*

次のテーブルは、インストールの実行時に発生する可能性のある Windows Installer エラーの一覧です。このエラーは、異なる機能をサポートするために InstallShield プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションに関連しています。カスタム アクションの一覧は、「InstallShield カスタム アクション リファレンス」を参照してください。

***ヒント・** 一部のランタイム エラーについての詳細 (解決策を含む) は、ナレッジ ベースを参照してください。*

テーブル 6 ・ Windows Installer ランタイム エラー

エラー番号	メッセージ	トラブルシューティング情報
27500	このセットアップには、IIS 仮想ルートの設定に IIS (Internet Information Server) が必要です。IIS がインストールされていることを確認してください。	このエラーは、プロジェクトに [IIS 構成] ビューで構成された IIS データが含まれている場合に発生します。 このエラーが発生した場合、ターゲット システムに IIS がインストールされていることを確認してください。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27501	このセットアップでは、 IIS Virtual Roots の設定に管理者権限が必要です。	このエラーは、プロジェクトに [IIS 構成] ビューで構成された IIS データが含まれている場合に発生します。 このエラーは、IIS 仮想ディレクトリの設定に必要な管理者権限がない状態で、IIS 仮想ディレクトリを設定するインストールを実行したときに発生します。
27502	[2] [3] に接続できませんでした。[4]	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、インストールが指定されたデータベース サーバーに接続できなかった場合に発生します。
27503	指定されたデータベース サーバーからバージョン情報を取得中にエラーが発生しました。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、[SQL スクリプト] ビューで、SQL 接続の [要件] タブを使ってデータベース サーバーに最小バージョン要件を指定した場合で、インストールがターゲット システム上に存在するデータベース サーバーのバージョンを判別できなかったときに発生します。
27504	指定されたデータベース サーバーがバージョン要件を満たしていません。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、ターゲットシステムが [SQL スクリプト] ビューで SQL 接続の [要件] タブで選択されたデータベース サーバーの最小バージョン要件を満たしていない場合に発生します。このシナリオでは、このタブにある **[最小要件が満たされなかったときも、インストールの続行を許可する]** チェック ボックスがクリアされている場合にのみ、このエラーが発生します。
27505	SQL スクリプト ファイルを開くことができません。	このエラーは、プロジェクトに [SQLスクリプト] ビューで構成された SQL スクリプトが含まれている場合で、ランタイムが .msi パッケージから SQL スクリプト ファイルを抽出できなかった場合に発生します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27506	SQL スクリプトを抽出中にエラーが発生しました。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL スクリプトが含まれる場合に発生します。 このエラーは、ターゲット データベース サーバーで SQL スクリプト エラーが起こった場合に発生します。このエラーを解決するためには、プロジェクトに含まれる SQL スクリプトを確認して、必要な変更を行います。
27507	SQL Server を参照するには ODBC32.dll のインストールと登録が必要です。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーを回避するためには、インストール プロジェクトに MDAC の InstallShield 前提条件を追加してください。
27508	COM+ アプリケーション [2] のインストール中にエラーが発生しました。[3]	このエラーは、プロジェクトに [コンポーネント サービス] ビューで構成された COM+ アプリケーションが含まれる場合に発生します。 このエラーは、COM+ コンポーネント DLL の依存関係が不足している場合、またはCOM+ アプリケーションの設定に誤りがある場合に発生します。このエラーを解決するためには、プロジェクトに依存関係を追加するか、プロジェクトに含まれる COM+ アプリケーションを確認して、必要な変更を加えてください。
27509	COM+ アプリケーション [2] のアンインストール中にエラーが発生しました。[3]	このエラーは、プロジェクトに [コンポーネント サービス] ビューで構成された COM+ アプリケーションが含まれる場合に発生します。 このエラーは、ランタイムがシステムから COM+ アプリケーションを削除するのに失敗した場合に発生します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27510	COM+ アプリケーション [2] のインストール中にエラーが発生しました。Microsoft(R) .NET クラス ライブラリをロードできませんでした。.NET サービス コンポーネントの登録を行うためには、Microsoft(R) .NET Framework のインストールが必要です。	このエラーは、プロジェクトに [コンポーネント サービス] ビューで構成された COM+ アプリケーションが含まれる場合に発生します。 このエラーの発生を防ぐためには、プロジェクトへの .NET Framework の追加を考慮してください。詳しくは、「.NET Framework 再配布可能ファイルをプロジェクトへ追加する」をご覧ください。 その他、プロジェクト アシスタントの [要件] ページを使って、プロジェクトに .NET Framework 要件を追加することもできます。ターゲット システムに .NET Framework の適切なバージョンが搭載されていない場合、インストールはエラー メッセージを表示して製品のインストールを行いません。
27511	SQL スクリプト ファイル [2] を実行できませんでした。接続は開いていません: [3]	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL スクリプトが含まれる場合に発生します。 このエラーは、インストールが SQL スクリプトを実行する前に接続が切断された場合に発生します。
27514	このインストールには、Microsoft SQL Server が必要です。指定されたサーバー [3] は、Microsoft SQL Server Desktop Engine または SQL Server Express です。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、ターゲット システムに Microsoft SQL Server Desktop Engine または SQL Server Express が搭載されていて、[SQL スクリプト] ビューで SQL 接続の [要件] タブにある次のチェック ボックスがクリアされている場合に発生します。 • Microsoft SQL Server Desktop Engine/SQL Server Express へのインストールを許可する • 最小要件が満たされなかったときも、インストールの続行を許可する
27515	[2] [3] からスキーマ バージョンを取得中にエラーが発生しました。データベース: [4]。[5]	この エラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL スクリプトが含まれていて、その SQL スクリプトの [全般] タブで "スキーマ バージョン" が指定されている場合に発生します。 このエラーは、ランタイムが InstallShield テーブルから現在のスキーマ バージョンを取得しようとしたときに、ターゲット データベース サーバーで SQL スクリプト エラーが発生した場合に表示されます。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27516	[2] [3] にスキーマ バージョンを書き込み中にエラーが発生しました。データベース: [4]。[5]	この エラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL スクリプトが含まれていて、その SQL スクリプトの [全般] タブで ″スキーマ バージョン″ が指定されている場合に発生します。 このエラーは、ランタイムが InstallShield テーブルに現在のスキーマ バージョンを書き込もうとしたときに、ターゲット データベース サーバーで SQL スクリプト エラーが発生した場合に表示されます。
27517	このインストールには、COM+ アプリケーションをインストールするための管理者権限が必要です。管理者としてログオンし、このインストールを再実行してください。	このエラーは、プロジェクトに [コンポーネント サービス] ビューで構成された COM+ アプリケーションが含まれる場合に発生します。 このエラーは、COM+ アプリケーションのインストールに必要な管理者権限がない状態で、COM+ アプリケーションを含むインストールを実行したときに発生します。
27519	XML ファイル [2] の更新エラー。[3]	このエラーは、プロジェクトに [XML ファイルの変更] ビューで構成された XML 参照が含まれている場合に発生します。 このエラーは、MSXML がターゲット XML ファイルの保存に失敗した場合に発生します。[3] パラメーターは、MSXML が戻すエラー コードを参照しています。エラー コード: • XML_BAD_ENCODING — XML ファイルには、[XML ファイルの変更] ビューにある XML ファイルの [詳細] タブで定義された指定のエンコードではない文字が含まれています。 • E_INVALIDARG — XML ファイルには、有効なファイル名が含まれていません。 • E_ACCESSDENIED — ファイル システム エラーが発生しました。 • E_OUTOFMEMORY — XML ファイルを保存できませんでした。 その他に発生する値は、ファイル システム エラーが発生したことを示します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27520	XML ファイル [2] を開くときにエラーが発生しました。[3]	このエラーは、プロジェクトに [XML ファイルの変更] ビューで構成された XML 参照が含まれている場合に発生します。 このエラーは、MSXML がターゲット XML ファイルのロードに失敗した場合に発生します。[3] パラメーターは、MSXML が戻すエラー コードを参照しています。エラー コード: • S_FALSE — XML ファイルのロードに失敗しました。
27521	このセットアップでは、XML ファイルの構成を行うために MSXML 3.0 以降が必要です。3.0 以降がインストールされていることを確認してください。	このエラーは、プロジェクトに [XML ファイルの変更] ビューで構成された XML 参照が含まれている場合に発生します。 このエラーの発生を防ぐためには、[再配布可能ファイル] ビューを使って、MSXML の InstallShield 前提条件またはマージ モジュールをプロジェクトに追加してください。
27524	NetApi32.DLL のロード中にエラーが発生しました。ISNetApi.dll には、NetApi32.DLL プロパティがロードされている NT ベースのオペレーティング システムが必要です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。エラーは、ターゲット システムに NetApi32.dll をロードできない場合に発生します。
27525	サーバーが見つかりません。指定のサーバーが存在することを確認してください。サーバー名を空白にすることはできません。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。エラーは、LogonInfoNameAndServer ダイアログまたは LogonInformation ダイアログで既存するドメインまたはサーバー名を入力しなかった場合に発生します。
27526	ISNetApi.dll から特定できないエラーが発生しました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。エラーは、NetApi32.dll に含まれる関数から特定できないエラーが返された場合に発生します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27527	バッファーが小さすぎます。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、ドメインで使用可能なすべてのサーバーのリストを取得するのに必要なメモリの容量をランタイムが割り当てることができなかった場合に発生します。
27528	アクセスが拒否されました。管理者権限があることを確認してください。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、構成している Windows ユーザー アカウント情報にユーザーがアクセスできない場合に発生します。
27529	コンピューターが無効です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、入力されたコンピューター名が無効な場合に発生します。
27531	未処理の例外。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、NetApi32.dll の関数呼び出しの 1 つが例外をスローした場合に発生します。
27532	このサーバーまたはドメインに対して、ユーザー名が無効です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、LogonInformation ダイアログまたは LogonInfoNameAndServer ダイアログでドメインに存在しないユーザー名を指定した場合に発生します。
27533	大文字と小文字を区別するパスワードが一致しません。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、LogonInfoCreateUser ダイアログの "パスワード" 設定と "パスワードの確認" 設定に異なるパスワードを入力した場合に発生します。
27534	リストが空白です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、LogonInfoListServers ダイアログを表示する必要があるが、サーバーが検出されなかった場合に発生します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27535	アクセス違反です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、ユーザー アカウントまたはサーバーに関する情報を取得または処理するときに、メモリー アクセス違反が行われた場合に発生します。
27536	グループの取得時にエラーが発生しました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、インストール中にグループ アカウントについての情報を取得するときにエラーが発生した場合に表示されます。
27537	ユーザーをグループに追加中にエラーが発生しました。このドメインまたはサーバーにグループが存在することを確認してください。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、ユーザー アカウントのメンバーシップをグループに追加しようとしたときにアカウントの種類が無効である場合、そのメンバーが既にグループのメンバーである場合、またはメンバーが存在しない場合に発生します。
27538	ユーザーの作成時にエラーが発生しました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、インストールでユーザー アカウントを作成したとき、またはユーザーをグループに追加するときエラーが発生した場合に表示されます。
27539	NetAPI から ERROR_NETAPI_ERROR_NOT_PRIMARY が返されました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、ユーザー アカウントを追加しようとしたとき、既存のユーザー アカウントにメンバーシップを付与するとき、またはユーザー アカウントを削除するときに、処理がドメインのプライマリ ドメイン コントローラーに対してのみ許可されている場合に発生します。
27540	指定されたユーザー名は、既に存在します。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、新しいユーザー アカウントを作成しようとしたときに、ユーザー アカウントが既に存在する場合に発生します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27541	指定されたグループは、既に存在します。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、新しいユーザー アカウントを作成しようとしたときに、グループが既に存在する場合に発生します。
27542	パスワードが無効です。パスワードがネットワークのパスワード ポリシーに準拠していることを確認してください。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。 このエラーは、新しいユーザー アカウントを作成するときに、指定したパスワードがネットワークで設定されているパスワード要件を満たしていない場合に発生します。このエラーを解決するには、適切なパスワードを入力してください。
27543	名前が無効です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、既存のユーザー アカウントにグループのメンバーシップを付与するとき、またはユーザー アカウントを削除するときに、指定したユーザー名が検出されない場合に発生します。
27544	グループが無効です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、ユーザー アカウントにグループのメンバーシップを付与するときに、指定したグループが存在しない場合に発生します。
27545	ユーザー名に空白は使用できません。またその形式は DOMAIN¥Username です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、ユーザー アカウントを構成するときに、″ユーザー名″設定に有効な情報を入力しなかった場合に発生します。
27546	ユーザー TEMP ディレクトリに INI ファイルを作成またはロード中にエラーが発生しました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、InstallShield カスタム アクションが Windows ユーザー アカウントの構成または作成に使用する構成 .ini ファイルを作成または開くときに問題があった場合に発生します。

テーブル 6 • Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27548	新しいユーザー情報を含む INI ファイルをユーザーの TEMP ディレクトリから削除中にエラーが発生しました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、InstallShield カスタム アクションが Windows ユーザー アカウントの構成または作成に使用する構成 .ini ファイルを削除するときに問題があった場合に発生します。
27549	プライマリ ドメイン コントローラー (PDC) の取得中にエラーが発生しました。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、指定のドメイン用のドメイン コントローラーが検出されなかった場合に発生します。
27550	ユーザーを作成するためには、各フィールドに値が必要です。	このエラーは、Windows ユーザー アカウントを作成または設定できる機能をプロジェクトに追加した場合に発生する可能性があります。このエラーは、LogonInfoCreateUser ダイアログの設定の 1 つが空白の場合に発生します。
27551	[2] のODBC ドライバーが検出されませんでした。このは、[2] データベース サーバーに接続するために必要です。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、SQL 接続と関連付けられている ODBC ドライバーがターゲット システムに存在しない場合に発生します。このエラーを回避する手段として、ドライバーをインストールする InstallShield 前提条件を作成して、その前提条件をプロジェクトに追加する方法があります。
27552	データベース [4] の作成中にエラーが発生しました。サーバー: [2] [3][5]	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれていて、その接続の [全般] タブにある [存在しない時、カタログを作成する] チェック ボックスが選択されている場合に発生する場合があります。 このエラーは、[SQL スクリプト] ビューで SQL 接続の [全般] タブにある "カタログ名" 設定で指定されたデータベース カタログを作成するときに、ターゲット データベース サーバーで SQL スクリプト エラーが発生した場合に表示されます。

テーブル 6 ・ Windows Installer ランタイム エラー (続き.)

エラー番号	メッセージ	トラブルシューティング情報
27553	データベース [4] に接続中にに エラーが発生しました。サー バー: [2] [3][5]	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、[SQL スクリプト] ビューで SQL 接続 の [全般] タブにある "カタログ名" 設定で指定され たデータベース カタログを接続するときに、ター ゲット データベース サーバーで SQL スクリプト エ ラーが発生した場合に表示されます。
27554	接続 [2] を開こうとしてエラー が発生しました。この接続に は、有効なデータベース メタ データが関連付けられていま せん。	このエラーは、プロジェクトに [SQL スクリプト] ビューで構成された SQL 接続が含まれている場合に発生します。 このエラーは、接続に有効なデータベース メタデー タが関連付けられていない場合に発生します。このエ ラーを回避するためには、[ダイレクト エディター] ビューで ISSQLConnectionDBServer テーブルを確認し て、必要な変更を行ってください。
27555	オブジェクト [2] にアクセス許 可を適用しようとしたときに、 エラーが発生しました。シス テム エラー: [3] ([4])	このエラーは、プロジェクトのファイル、フォルダ、 またはレジストリ キーの保護にカスタム InstallShield 処理を使用した場合に発生する場合があります。この 機能に関する詳細については、「ロックダウン環境に おけるファイル、フォルダー、レジストリ キー、お よび Windows サービスのセキュリティ保護」を参照 してください。

Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト)

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.exe は、適切に起動しなかった場合、エラー メッセージを生成しないことがあります。これらのメッセージは、多くの場合、重大なエラーが発生した際に表示されます。これらのメッセージはエンドユーザーには滅多に表示されません。

これらの戻り値は、Setup.exe を起動するかまたは Setup.exe を起動するためにバッチファイルを使用して、**CreateProcess** API を呼び出すときにキャプチャできます。

エラーメッセージは、メッセージボックス内で表示されます。すべてのエラー メッセージには番号が付いています。これらは、InstallScript のシステム エラー メッセージなので、スクリプトでメッセージを抑制することはできません。次のテーブルは、これらのエラーメッセージの説明です。

テーブル 7 • InstallScript プロジェクトのランタイム エラーおよび戻り値

エラー/戻り値	説明
3010	インストールの再開が必要ですが、インストールが再開処理を開始しません。この場合、インストールが BATCH_INSTALL をゼロ以外の値に設定して完了しましたが、**System** 関数が呼び出されませんでした。 Setup.exe は、インストールが成功して、キャンセルされなかった (つまり、インストールが abort キーワードで終了しなかった) 場合のみこの値を返します。それ以外の場合は、適切なエラー値または ISERR_SETUP_CANCELED が返されます。 この戻り値の定数は ERROR_SUCCESS_REBOOT_REQUIRED です。
1641	この戻り値は、(通常、**SdFinishReboot** 関数が内部で **System** を呼び出した結果として) **System** 関数が呼び出されたためにインストールそのものがマシンを再開した場合に発生します。 Setup.exe は、インストールが成功して、キャンセルされなかった (つまり、インストールが abort キーワードで終了しなかった) 場合のみこの値を返します。それ以外の場合は、適切なエラー値または ISERR_SETUP_CANCELED が返されます。 この戻り値の定数は ERROR_SUCCESS_REBOOT_INITIATED です。
0	インストールは exit キーワードを使って終了しました。
0x80042000	エンド ユーザーがインストールをキャンセルしたため、インストールは abort キーワードを使って終了しました。 この戻り値の定数は ISERR_SETUP_CANCELED です。
0x80040708	要求されたエンジン コンポーネントを作成することができませんでした。COM コンポーネントを作成するために必要な権限があるかどうかを確認してください。 ***メモ:*** *このエラー メッセージには、オペレーティング システムが返した詳細なエラー情報を示す、追加のエラー番号またはエラー文字列が含まれていることがあります。*
-5001	一般エラー
-5002	メディア ヘッダーの読み込みに失敗しました。
-5003	カーネルのインストールに失敗しました。
-5004	カーネルの起動に失敗しました。
-5005	CAB を開くことができませんでした。

テーブル 7 • InstallScript プロジェクトのランタイム エラーおよび戻り値 (続き.)

エラー/戻り値	説明
–5006	サポートのインストールに失敗しました。
–5007	テキスト置換の設定に失敗しました。
–5008	インストール情報の初期化に失敗しました。
–5009	インストール ドライバーの取得に失敗しました。
–5010	プロパティの初期化に失敗しました。
–5011	インストール ドライバーの実行に失敗しました。
–5012	サポートのアンインストールに失敗しました。
–5013	セットアップ起動ファイルからファイルを抽出できませんでした。
–5014	ファイルのダウンロードに失敗しました (インターネット インストール中にインストール ファイルを保存するときのみ発生します)
–5017	インストールをクローンすることができませんでした。
–6001	セットアップランチャーの起動に失敗しました。
–6002	セットアップランチャーを見つけることができませんでした。
–6003	セットアップランチャーのロードに失敗しました。
–6004	セットアップランチャーの署名を確認することができませんでした。
–6005	セットアップランチャーを適切な場所にインストールできませんでした。
–6006	セットアップランチャの抽出に失敗しました。

通常、上記のテーブル内のエラー メッセージが表示されると、次のテーブル内の HRESULT 値の 1 つも同時に表示されます。これらの HRESULT 値により、エラーの原因に関する詳しい情報を見ることができます。

テーブル 8 • Setup.exe ランタイム エラーに関連付けられている HRESULT 値

HRESULT	意味
0x80041F40	corecomp.in.i を見つけることができません。
0x80041F41	corecomp.ini は空か、破損しています。
0x80041F42	.dll ファイル inIsCoGetClassObject をロードできません。
0x80042328	メディアに署名がありません。

テーブル 8・Setup.exe ランタイム エラーに関連付けられている HRESULT 値 (続き.)

HRESULT	意味
0x80042329	証明書が無効です。
0x8004232A	デジタル署名が無効です。
0x8004232B	空き容量が足りません。
0x80042A94	指定された opType の メイン opsequence が ログに存在します。古いファイルが新しいエンジンで開かれたか、新しい (エンジン) コードが新規に導入された opType の opsequence を取得しようとした可能性があります。
0x80042A95	機能のログの完全 ID を構築または再構築することができませんでした。
0x80042A96	ログ操作が、ログ ファイルを開かずに実行されました。
0x80042A97	内部 opsequence 構造の初期化に失敗しました。
0x80042A98	Opsequence の一般エラー ::GetNext
0x80042A99	Opsequence の一般エラー ::Pop
0x80042A9A	Opsequence の一般エラー ::insert_sequencetuple
0x80042A9B	Opsequence の一般エラー ::delete_sequencetuple
0x80042A9C	プロパティの初期化に失敗しました。- LogDB または機能オブジェクトのプロパティを設定/取得した際に発生することがあります。

Setup.exe 戻り値および実行時のエラー (基本の MSI および InstallScript MSI プロジェクト)

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

プロジェクトによって異なる情報がある場合は、その内容が説明されています。

次のテーブルは、基本の MSI または InstallScript MSI プロジェクトから Setup.exe を実行したときに発生する可能性のあるエラーの一覧です。InstallScript プロジェクトで発生する可能性がある Setup.exe エラーについては、「Setup.exe 戻り値および実行時のエラー (InstallScript プロジェクト)」を参照してください。

これらの戻り値は、Setup.exe を起動するかまたは Setup.exe を起動するためにバッチファイルを使用して、**CreateProcess** API を呼び出すときにキャプチャできます。

メモ: *英語以外のシステム上でエラーが発生した場合、エラー文字列がで英語ではない場合があります。*

テーブル 9 • Setup.exe ランタイム エラーと戻り値

エラー/ステータス番号	メッセージ	トラブルシューティングのヒント
–4	コマンドラインが無効です。	 ***プロジェクト:*** *これは InstallScript MSI プロジェクトに適用します。* Setup.exe に有効なコマンドライン文字列が渡されたかどうかを確認してください。
–3	エンド ユーザー がインストールをキャンセルしたため、インストールが終了しました。	 ***プロジェクト:*** *これは InstallScript MSI プロジェクトに適用します。*
–1	一般エラー。	 ***プロジェクト:*** *これは InstallScript MSI プロジェクトに適用します。*
0	プログラムが正常に終了しました。	なし。
401	文字列値が文字列に対して十分な大きさではありません。InstallShield がテキスト文字列を文字列変数にコピーしようと試みました。テキスト文字列が、その文字列変数に宣言されている長さを超えています。	文字列変数で宣言されている長さを確認してください。長さを最大許容値に増やします。
1150	セットアップが互換性のない Windows のバージョンを検出しました。[OK] をクリックし、Windows 95、Windows NT 4.0、またはそれ以降のバージョンでセットアップを再起動してください。	Windows Installer は Windows NT 4.0 またはそれ以降、および Windows 9x 以降と互換性があります。ご使用の Windows のバージョンを確認し、必要な場合は互換性のあるバージョンにアップグレードしてください。

テーブル 9 • Setup.exe ランタイム エラーと戻り値 (続き.)

エラー/ステータス番号	メッセージ	トラブルシューティングのヒント
1151	一時ディレクトリへの書き込みエラー	一時ディレクトリへ書き込むには、環境変数 TEMP が設定されている必要があります。一時フォルダーが存在していて、セットアップを格納するのに十分なディスク容量があることを確認してください。一時フォルダーにファイルがある場合は、それらを削除して Setup.exe を再実行してください。
1152	一時ディレクトリへの <ファイル名> の抽出エラー	Temp フォルダーが書き込み可能であることを確認してください (上のエラーを参照)。Temp フォルダーが書き込み可能な場合は、セットアップ中のファイルが壊れている可能性があります。壊れているファイルがないことを確認し、Setup.exe を再実行してください。
1153	セットアップ初期化ファイルの読み取りエラー	Setup.ini ファイル は、Setup.exe と同じフォルダーに配置しなくてはなりません。そうでない場合、適切な場所に Setup.ini を移動させます。
1154	インストーラーが <パス> で見つかりません。	Windows Installer が正しくインストールされていないか、古いバージョンが使用されている可能性があります。必要に応じて再インストールします。
1155	ファイル <ファイル名> が見つかりません。	.msi ファイルがあることを確認してください。.msi ファイルがある場合は、それが Setup.exe と同じフォルダー内にあることを確認してください。.msi ファイルは、Setup.exe に圧縮する選択をした場合、表示されないことがあります。
1156	Windows Installer の内部エラー	Windows Installer が正常にインストールされていません。InstMsiW.exe (Windows NT および 2000 用) または InstMsiA.exe (Windows 9x 用) を実行して再インストールしてください。
1157	Msiexec.exe の起動に失敗しました。	ターゲット プラットフォームにWindows Installer の正しいバージョンを配布していることを確認してください。 Msiexec.exe コマンドライン引数の構文を確認してください。
1158	文字列の作成エラー	Disk1 フォルダーにある Setup.ini と言語固有の INI ファイル (0x0409.ini など) のすべての文字列が有効であることを確認してください。
1201	セットアップは、<フォルダー> に <容量> KB の空き容量が必要です。空き容量を増やして再度実行してください。	ターゲット位置のディスク容量が不充分です。セットアップをインストールするドライブに 10 MB 以上の空き容量があることを確認してください。

テーブル 9 • Setup.exe ランタイム エラーと戻り値 (続き.)

エラー/ステータス番号	メッセージ	トラブルシューティングのヒント
1202	このインストールを完了するための十分な権限がマシンのすべてのユーザーにありません。管理者としてログオンし、このインストールを再実行してください。	Windows NT 4.0 および 2000 では、このインストールを完了するために管理者特権が必要です。
1203	コマンドラインパラメーターが無効です。	Setup.exe を起動する際に使用したコマンドライン ステートメントを再確認してください。
1208	<言語> の ANSI コードページがシステムにインストールされていないので、選択された言語でセットアップを実行できません。セットアップを実行し、他の言語を選択してください。	セットアップを実行し、他の言語を選択してください。
1603	一般的な Windows Installer エンジンのエラーSetup.ini に必要なディスク容量を増やし、再度実行してください。	 ***プロジェクト:*** *これは基本の MSI プロジェクトに適用します。* Windows Installer を再インストールしてください。
1611	ファイル <ファイル名> を抽出できません。	このエラーは、Setup.exe の中で圧縮されたファイルを抽出できない場合に発生します。Temp フォルダー (または Windows または WinNT フォルダー) に十分なディスク容量があるか、またセットアップがこれらのフォルダーに書き込めるかどうか確認してください。
1614	ファイル <ファイル名> をダウンロード中にエラーが発生しました。	URL が間違っている場合は、**[キャンセル]** をクリックして正しい URL を入力してください。 URL が正しい場合は、アクティブなインターネット接続が使用可能か確認してください。
1621	ファイル <ファイル名> の署名を確認できませんでした。	ファイルはダウンロードされましたが、署名を確認できませんでした。ファイルが壊れているか、元の署名者とは別の会社によって署名されているか、署名できない可能性があります。 ターゲット システムに Wintrust.dll が存在することを確認してください。

テーブル 9・Setup.exe ランタイム エラーと戻り値 (続き.)

エラー/ステータス番号	メッセージ	トラブルシューティングのヒント
1627	次のファイルを保存できません: <ファイル名>	指定されたファイルが既に存在しないこと、またターゲット システムに十分なハードディスク容量があることを確認してください。
1628	スクリプトベースのインストールを完了できませんでした。	 ***プロジェクト:*** *これは InstallScript MSI プロジェクトに適用します。*
1629	コマンドラインが無効です。	Setup.exe を起動する際に使用したコマンドライン ステートメントを再確認してください。
1641	インストールが成功し、再起動が必要でした。	 ***プロジェクト:*** *これは InstallScript MSI プロジェクトに適用します。* Windows Installer エンジンが ERROR_SUCCESS (0) を返した場合、Setup.exe ファイルが ERROR_SUCCESS_REBOOT_INITIATED (1641) を返す場合があります。この戻り値は、(通常、**SdFinishReboot** 関数が内部で **System** を呼び出した結果として) **System** 関数が呼び出されたためにインストールそのものがマシンを再開した場合に発生します。
1670	モジュール [1] をロードできませんでした。エラー コード: [2]	[1] は、インストールがロードを試みている .dll ファイルへの完全パスです (すべての場合において、ISSetup.dll)。 [2] は、**LoadLibrary()** から戻された Windows エラーです。 このエラーは、ISSetup.dll が Disk1 フォルダーに存在しないときに発生します。これはまた、ISSetup.dll が Setup.exe に圧縮されている場合で、圧縮メディアがビルドされた場合にも発生することがあります。 このエラーを解決するには、インストールを再ビルドしてみてください。
1700	InstallScript エンジンの初期化中にエラーが発生しました。	 ***プロジェクト:*** *これは InstallScript MSI プロジェクトに適用します。* このエラーは、InstallScript エンジンのロード中に問題が発生したことを示します。

テーブル 9 • Setup.exe ランタイム エラーと戻り値 (続き.)

エラー/ステータス番号	メッセージ	トラブルシューティングのヒント
1701	InstallScript エンジン サポート ファイルを一時保管場所に抽出できませんでした。	 ***プロジェクト•*** *これは InstallScript MSI プロジェクトに適用します。* このエラーは、1 つまたは複数のファイルを ISSetup.dll ファイルから一時ディレクトリに抽出できなかったときに発生します。 このエラーを解決するため、Windows Installer ログ ファイルを作成して詳しい情報を入手してください。ログ ファイルには、対応する Windows エラー番号およびその他の診断内容といった、問題を解決するために役立つ情報が記録されます。
1919	ODBC データソースの設定エラー。アクセス可能なファイルが存在していることを確認してください。	Windows 95 を実行しているマシンに ODBC コアがありません。ODBC ドライバーを含むパッケージをインストールする場合は、事前に MDAC をインストールしてください。
3010	インストールが成功し、インストールを完成するために再起動が必要です。	 ***プロジェクト•*** *これは InstallScript MSI プロジェクトに適用します。* Windows Installer エンジンが ERROR_SUCCESS (0) を返した場合、Setup.exe ファイルが ERROR_SUCCESS_REBOOT_REQUIRED (3010) を返す場合があります。これは、インストールの再開が必要ですが、インストールが再開処理を開始しないことを示します。この場合、インストールが BATCH_INSTALL をゼロ以外の値に設定して完了しましたが、**System** 関数が呼び出されませんでした。

Setup.exe 戻り値および実行時のエラー (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

以下は、エンド ユーザーがアドバンスト UI またはスイート/アドバンスト UI Setup.exe インストールを実行したときに発生する可能性のあるエラーの一覧です。コードの種類列には、次の情報が含まれています:

- **終了コードのみ** – インストール ステータスは、アドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルによって戻される。
- **エラー コード/終了コード** – インストール ステータスは評価されたエラー条件に処理され、ステータスはアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルによって戻される。
- **ログ記録のみ** – インストール ステータスは、デバッグ ログ記録が有効な場合にログ記録される。(デバッグ ログ記録は、/debuglog コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルに渡してアドバンスト UI またはスイート/アドバンスト UI インストールを起動した場合に有効です。コマンドライン パラメーターの詳細については、「アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター」を参照してください。)

メモ: *エラー メッセージの一部は、実行中のアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルで文字列エントリが使用可能になる前に表示される場合があります。そのため、たとえば Setup.xml が不足している、別のアドバンスト UI またはスイート/アドバンスト UI インストールが既に実行中、または条件検証に失敗した場合、これらの特定のエラー メッセージは英語で表示されます。言語の選択前に発生するその他のエラーは、通常デフォルト セットアップ言語で表示されます。*

テーブル 10 ・ Setup.exe ランタイム エラーと戻り値

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x80040701	終了コードのみ	セットアップの状態についての情報を読み取りまたは初期化することができませんでした。Setup.xml セットアップ要素の SuiteId 属性が不足しています。ベンダーに問い合わせてください。	Setup.xml ファイルの SuiteID 属性の値は、プロジェクトの [一般情報] ビューで定義されたスイート GUID を示します。この GUID が何らかの原因で Setup.xml ファイルから削除された場合、このエラーが実行時に発生する可能性があります。 このエラーを解決するには、アドバンスト UI またはスイート/アドバンスト UI インストールを再ビルドしてみてください。
0x80040702	終了コードのみ	UI DLL をロードおよび初期化できませんでした。 UI リソースが不足している可能性があります。ベンダーに問い合わせてください。	この終了コードは、プロジェクトまたは Setup.xml ファイルの UIResource 定義が不足している、または無効な場合 (たとえば、プロジェクトが破損している場合) に発生します。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x80040705	終了/エラー コード	ステージする現在のファイルの MD5 のチェックに失敗しました。ファイルが破損している可能性があります。	アドバンスト UI またはスイート/アドバンスト UI インストールは MD5 を使って、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの整合性を確認します。MD5 のチェックに失敗したとき、このエラーが発生します。 このエラーを解決するには、アドバンスト UI またはスイート/アドバンスト UI インストールを再ビルドしてみてください。
0x80040706	終了コードのみ	実行中のスイート セットアップで setup.xml ファイルが見つかりませんでした。ベンダーに問い合わせてください。	このステータス コードは、Setup.xml リソースが見つからなかった場合に発生します。
0x80040707	ログ記録のみ	このセットアップを実行するのに必要なファイルの抽出に失敗しました。ベンダーに問い合わせてください。	Setup.xml ファイルの Resources セクションにリストされているリソースが実行中のアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルから抽出されなかった場合に、このステータス コードがログ記録に含まれます。ビルド エラーと [サポート ファイル] ビューで不足しているリソースを確認して、必要に応じてそれらを追加してください。
0x80040708	ログ記録のみ	このセットアップを実行するのに必要なファイルのコピーまたはダウンロードに失敗しました。ベンダーに問い合わせてください。	Setup.xml ファイルの Resources セクションにリストされているリソースがコピーまたはダウンロードされなかった場合に、このステータス コードがログ記録に含まれます。ビルド エラーと [サポート ファイル] ビューで不足しているリソースを確認して、必要に応じてそれらを追加してください。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x80040709	エラー コードのみ	ステージング中にファイルを抽出できませんでした。ファイルがストリームに存在しないか、ファイルをターゲット マシンに書き込むことができませんでした。	このエラーは、アドバンスト UI またはスイート/アドバンスト UI インストールがアドバンスト UI またはスイート/アドバンスト UI インストール Setup.exe ファイルにストリームされるファイルのステージングに失敗したときに発生します。インストールが破損しているか、ビルド時にファイルが不足しています。
0x8004070A	終了/エラー コード	セットアップ パスワードが間違っているか、入力されませんでした。セットアップを続行できません。	アドバンスト UI またはスイート/アドバンスト UI プロジェクトの [リリース] ビューに表示されるリリースの Setup.exe タブでは、セットアップランチャーをパスワードで保護するかどうかを指定できます。セットアップランチャーをパスワードで保護する場合、エンド ユーザーがインストールを実行するために入力する必要があるパスワードを指定します。 このエラーは、インストールのプレインストールの段階で、ユーザー インターフェイスからパスワードのプロンプトが行われなかった場合に発生します。この状況は、カスタム UI DLL が使用された場合で、エンド ユーザーがパスワード プロンプトを回避しようとしたときに発生します。また、エンド ユーザーがパスワード保護されたアドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインからサイレントで起動した場合で、コマンドラインにパスワードが含まれていないときにも発生します。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x8004070B	終了/エラー コード	セットアップ コマンドラインが無効です。セットアップを続行できません。	このエラーは、コマンドラインからアドバンスト UI またはスイート/アドバンスト UI Setup.exe ファイルを起動するときに無効なコマンドライン パラメーターを渡した場合に発生します。 有効なコマンドライン パラメーターの一覧は、「アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター」を参照してください。
0x8004070C	終了/エラー コード	セットアップを一時フォルダ内で開始することができません。ベンダーに問い合わせてください。	このエラーは、ターゲット システム上の一時フォルダーの空き容量が足りない場合に発生します。このエラーを解決するには、一時フォルダの空き容量を増やす方法を考慮してください。
0x8004070D	終了/エラー コード	ステージングのみの処理用に入力されたインストール先フォルダーを作成できませんでした。	/stage_only コマンドライン パラメーターを使ってアドバンスト UI またはスイート/アドバンスト UI インストールを実行すると、アドバンスト UI またはスイート/アドバンスト UI インストールはユーザーが指定したディレクトリにスイートのパッケージをコピーしようとします。ディレクトリは既存しなくてはならず、存在しない場合はインストールでこのエラーが発生します。 このエラーを解決するためには、BrowseStageFolder ウィザード ページで指定したパスが存在するを確認してください。存在しない場合は、それを作成してからインストールに戻ってください。

テーブル 10・Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x8004070E	終了/エラー コード	リモート サーバーまたはローカル パスからファイルをキャッシュしようとして、予期しないエラーが発生しました。ベンダーに問い合わせてください。	ファイルが Web サーバー上にある場合、インストールでこのエラーが発生しているマシン上で、そのファイルを使用できることを確認してください。ファイルがソース メディアにある場合、そのソース メディアが有効かつ使用可能であることを確認してください。その他に利用可能な情報にについては、デバッグ ログを確認してください。
0x8004070f	終了/エラー コード	setup.xml を解析中に無効な条件が見つかりました。詳細を確認するためのデバッグ ログを作成します。スイートを正しく起動するためには、IDE 上で条件を訂正する必要があります。	このエラーは、無効な条件が構成された場合に発生します。 このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。 このエラーを解決するには、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれる無効な条件を編集します。
0x80040710	エラーコード	中止条件で定義されたスイートの要件が満たされなかったため、セットアップが中止しました。	このエラーは、[一般情報] ビューで定義された終了条件が、ターゲット システム上で実行時に満たされなかった場合に発生します。これが予期しない動作の場合、終了条件の変更が必要な場合があります。詳しくは、「アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する」を参照してください。
0x80040711	終了/エラー コード	スイート インストールの別のインスタンスが既に実行中です。別のインスタンスが終了するまで待機してから、再試行してください。	1 つのターゲット システム上でアドバンスト UI またはスイート/アドバンスト UI インストールのインスタンスを同時に 1 つだけ実行できます。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x80040712	終了/エラー コード	ISSetup.dll をロードして InstallScript パーセルを起動するのに失敗しました。	このエラーは、プロジェクトに含まれる InstallScript パッケージの 1 つ以上のファイルが、ビルド時に InstallShield で使用できなかったときに発生することがあります。 このエラーを解決するには、InstallScript パッケージに含まれるすべてのファイルがビルド時に使用可能であることを確認してください。
0x80040713	終了/エラー コード	ISSetup.dll を呼び出して InstallScript パーセルを起動するのに失敗しました。パーセルに含まれる InstallScript セットアップが、InstallScript パーセルをサポートするバージョンの InstallShield でビルドされていることを確認してください。	このエラーは、プロジェクトに InstallShield 2012 Spring 以降でビルドされた InstallScript パッケージを含むと発生することがあります。追加要件については、「InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する」を参照してください。
0x80040714	終了/エラー コード	EXE の起動に失敗しました。このエラーは、EXE アクション、昇格された EXE パーセル、Windows Vista、2008、7、または 2008 R2 上の Windows の機能サポートを起動しようとして、EXE の起動に失敗したときに発生することがあります。デバッグ ログにエラーの詳細が含まれている可能性がありますので、そちらを確認してください。	このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x80040715	終了/エラー コード	DLL のロードに失敗しました。このエラーは、Windows 8 またはそれ以降のバージョンの Windows で DLL アクションまたは Windows の機能サポートの DLLをロードしようとしたときに発生する場合があります。デバッグ ログにエラーの詳細が含まれている可能性がありますので、そちらを確認してください。	このエラーは、プロジェクトに含まれる DLL アクションが呼び出す必要のあるファイルが不足している、またはロードできなかった場合に発生する可能性があります。また、インストールが Windows のロールまたは機能をターゲット システム上で有効にしようとするときに発生することもあります。 このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。
0x80040716	終了/エラー コード	PowerShell アクションを実行するのに必要なサポートファイルの検出またはロードに失敗しました。デバッグ ログにエラーの詳細が含まれている可能性がありますので、そちらを確認してください。	このエラーは、プロジェクトに含まれる PowerShell アクションが呼び出す必要のあるファイルが不足している、またはロードできなかった場合に発生する可能性があります。 このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。
0x80040718	終了/エラー コード	InstallScript アクションで InstallScript 関数を呼び出すのに失敗しました。デバッグ ログにエラーの詳細が含まれている可能性がありますので、そちらを確認してください。	このエラーは、インストールが InstallScript アクションで InstallScript 関数を呼び出すことができなかったために発生します。 このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x8004071A	エラー コード	スイート デバッグ ログに指定されたパスが存在しない、または無効です。/debuglog をファイルパスと共に渡すときに、有効なファイル パスを指定してください。この場合、スイート エラーは 1622 を返します。	デバッグ ログ ファイルの有効なパスを確実に指定してください。
0x8004071B	終了/エラー コード	Web 配布パーセルをインストールするために必要なリソースが見つかりませんでした。セットアップを再ビルドしてからもう一度実行してください。	このエラーは、プロジェクト内の Web 配布パッケージに必要なファイルが不足している場合に発生することがあります。 このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。
0x8004071C	終了/エラー コード	CLR アクションを呼び出すために必要なリソースが不足している、またはロードすることができませんでした。デバッグ ログにエラーの詳細が含まれている可能性がありますので、そちらを確認してください。	このエラーは、プロジェクトに含まれるマネージ コード アクションが呼び出す必要のあるファイルが不足している、またはロードできなかった場合に発生する可能性があります。 このエラーをトラブルシューティングするには、デバッグ ログ ファイルの詳細を確認してください。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x8007007f	エラー コードのみ	指定されたプロシージャが見つかりません。	このえらーは、システム エラー 127 です。 このエラーは、アドバンスト UI またはスイート/アドバンスト UI インストールで、Windows Installer の最小要件を満たさないシステムで実行されたときに発生する可能性があります。 • ターゲット システムでアドバンスト UI またはスイート/アドバンスト UI インストールを実行するには、ターゲット システムに、Windows Installer 3.1 以降が必要です。これらのシステムでこのエラーを回避するには、アドバンスト UI またはスイート/アドバンスト UI インストール全体が Windows Installer 4.5 以降がないシステムで実行されるのを防ぐ終了条件を作成できます。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールの終了条件を定義する」を参照してください。

テーブル 10 • Setup.exe ランタイム エラーと戻り値 (続き.)

ステータス コード	コードの種類	メッセージ	トラブルシューティングのヒント
0x8007007f (続き)			• スアドバンスト UI またはスイート/アドバンスト UI インストールに、トランザクション処理と共に実行されるパッケージを含まれている場合、ターゲット システムに Windows Installer 4.5 以降が必要です。これらのシステムでこのエラーを回避するには、トランザクションで各パッケージに対して対象条件を作成して、以前のバージョンの Windows Installer があるシステム上で、これらのパッケージが起動されるのを防ぎます。詳細については、「アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおける、リリースの作成およびビルド」を参照してください。 このエラーは、パッケージが InstallShield 2012 以前でビルドされているために、InstallScript パッケージのインストールで問題が生じたときにも発生することがあります。詳細については、「InstallScript パッケージをアドバンスト UI またはスイート/アドバンスト UI プロジェクトに追加する」を参照してください。 また、アドバンスト UI またはスイート/アドバンスト UI の拡張条件で問題が生じたときも、発生する可能性があります。プロジェクトに拡張条件が含まれている場合、拡張条件の DLL に、適切にエクスポートされたエントリ ポイントがあることを確認してください。

Visual Studio プロジェクトのインポート エラーと警告

次のいずれかの処理を行ったときに発生する可能性のあるエラーと警告をリストします:

- Visual Studio セットアップ プロジェクト (.vdproj) を InstallShield 基本の MSI プロジェクト (.ism) に変換。

- Visual Studio マージ モジュール プロジェクト (.vdproj) を InstallShield マージ モジュール プロジェクト (.ism) に変換。
- Visual Studio セットアップまたはマージ モジュール プロジェクト (.vdproj) を InstallShield 基本の MSI または マージ モジュール プロジェクト (.ism) にインポート。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9000	不明な例外が発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-9001	不明な COM 例外が発生しました。	このエラーの情報について ナレッジ ベースを調べるか、テクニカル サポートにご連絡ください。
-9002	プロジェクト %1 をロード中にエラーが発生しました。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9003	プロジェクト %1 を作成中にエラーが発生しました。	このエラーは、 が基本の MSI プロジェクトを Visual Studio セットアップ プロジェクトから作成できないとき、または マージ モジュール プロジェクトを Visual Studio マージ モジュール プロジェクトから作成できないときに発生します。InstallShield は、プロジェクトを Visual Studio プロジェクト ファイル (.vdproj) を含んでいるフォルダーに作成しようとします。 このエラーを解決するには、Visual Studio プロジェクト ファイルを含むフォルダーが読み取り専用になっていないかどうかを確認し、読み取り専用の場合、それを書き込み可能にします。
-9004	プロジェクト %1 を保存中にエラーが発生しました。	このエラーは、 が基本の MSI プロジェクトを Visual Studio セットアップ プロジェクトから作成できないとき、または マージ モジュール プロジェクトを Visual Studio マージ モジュール プロジェクトから作成できないときに発生します。InstallShield は、プロジェクトを Visual Studio プロジェクト ファイル (.vdproj) を含んでいるフォルダーに保存しようとします。 このエラーを解決するには、Visual Studio プロジェクト ファイルを含むフォルダーが読み取り専用になっていないかどうかを確認し、読み取り専用の場合、それを書き込み可能にします。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9005	プロジェクトを変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9006	プロジェクトの種類を判別中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9007	セクション '%1' を解析中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9008	セクション %1 (%2 の下) に開き大かっこがありません。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9009	セクション %1 (%2 の下) に閉じ大かっこがありません。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9010	無効なエントリ %1 が見つかりました (セクション %1 (%2 の下))。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9011	プロパティ %1 をセクション %2 に追加中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。

テーブル 11・Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9012	セクション %1 をセクション %2 に追加中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9013	セクション % 1が存在しません。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9014	製品のプロパティを変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9015	ファイルを変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9016	ファイルの種類 %1 が無効です。ファイル %2 は変換されませんでした。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9017	ファイル %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9018	機能を変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9019	機能 %1 を変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9020	フォルダーを変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9021	フォルダー %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9022	フォルダーの種類 %1 が無効です。フォルダー %2 は変換されませんでした。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9023	カスタム アクションを変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9024	カスタム アクション %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9025	カスタム アクションの種類 %1 が無効です。カスタム アクション %2 は変換されませんでした。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。

テーブル 11・Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9026	ソース ファイル %1 (カスタム アクション %2) のコンポーネントが見つかりませんでした。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9027	ファイルの種類を変換中にエラーが発生しました。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9028	ファイル拡張子 %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9029	ファイル拡張子の種類 %1 が無効です。ファイル拡張子 %2 は変換されませんでした。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9030	動詞 %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。
-9031	動詞の種類 %1 が無効です。動詞 %2 は変換されませんでした。	このエラーは、Visual Studio プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポートにご連絡ください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9032	ファイル 拡張子 %1 のコマンドが指定されていません。ファイル拡張子は変換されませんでした。	この警告は、Visual Studio のファイルの種類エディターで指定されたファイルの種類の Command プロパティが空白のまま残されたとき発生します。この警告は、変換処理中に InstallShield がファイル拡張子を変換できなかったことをアラートします。 この問題を解決するには、Visual Studio でファイル タイプのコマンドを指定してから、Visual Studio セットアップ プロジェクトを InstallShield プロジェクトに変換します。それ以外の方法として、InstallShield の [コンポーネント] ビュー内の [ファイルの種類] 領域にファイル拡張子を追加し構成します。
-9033	コマンド %1 (ファイル拡張子 %2) のコンポーネントが見つかりませんでした。ファイル拡張子は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9034	レジストリを変換中にエラーが発生しました。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9035	レジストリ キー %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9036	レジストリ キーの種類 %1 が無効です。レジストリ キー %2 は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9037	レジストリ値 %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9038	レジストリ値の種類 %1 が無効です。レジストリ値 %2 は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9039	ショートカットを変換中にエラーが発生しました。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9040	ショートカット %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9041	ショートカットの種類 %1 が無効です。ショートカット %2 は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9042	ショートカットのターゲット %1 が無効です。ショートカット %2 は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
–9043	起動条件を変換中にエラーが発生しました。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
–9044	起動条件 %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
–9045	起動条件の種類 %1 が無効です。起動条件 %2 は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
–9046	ロケータを変換中にエラーが発生しました。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
–9047	ロケーター %1 を変換中にエラーが発生しました。種類: %2	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
–9048	ロケータの種類 %1 が無効です。ロケータ %2 は変換されませんでした。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9049	製品のアイコンを変換中にエラーが発生しました。	このエラーは、Visual Studio セットアップまたはマージ モジュール プロジェクト ファイルが破損している場合、または InstallShield が Visual Studio セットアップ プロジェクト ファイルを正しく読み込めないときに発生する可能性があります。このエラーを解決するには、テクニカル サポート にご連絡ください。
-9050	プロジェクトに言語 '%1' は InstallShield に含まれていません。言語は変換されませんでした。	この警告は、言語のサポートを含む Visual Studio プロジェクトを変換するとき、InstallShield にその言語のビルトイン サポートがないとき発生します。 ***エディション:*** *InstallShield Premier edition には複数言語のサポートが含まれていますが、Professional Edition には含まれていません。*
-9052	ファイルの種類 '%1' は拡張子が指定されていません。ファイルの種類は変換されませんでした。	この警告は、Visual Studio のファイルの種類エディターで指定されたファイルの種類の Extensions プロパティが空白のまま残されたとき発生します。この警告は、インポートまたは変換処理中に InstallShield がファイル拡張子を構成できなかったことをアラートします。 この問題を解決するには、Visual Studio でファイルの種類に 1 つ以上のファイル拡張子を指定してから、Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートします。これ以外の方法としては、Visual Studio プロジェクトを変換またはインポートした後に InstallShield の [コンポーネント] ビューにある [詳細設定] 領域でファイル拡張子を追加または構成できます。
-9053	起動条件 '%1' で条件が指定されていません。起動条件は変換されませんでした。	この警告は、Visual Studio の起動条件エディターで指定された起動条件の Condition プロパティが空白のまま残されたとき発生します。この警告は、インポートまたは変換処理中に InstallShield が起動条件を構成できなかったことをアラートします。 この問題を解決するには、Visual Studio で条件を指定してから、Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートします。これ以外の方法としては、Visual Studio プロジェクトを変換またはインポートした後に InstallShield の [一般情報] ビューで起動条件を追加または構成できます。

テーブル 11・Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9054	ファイル検索 '%1' のファイル名が指定されていません。ファイル検索は変換されませんでした。	この警告は、Visual Studio の起動条件エディターで指定されたファイル検索の FileName プロパティが空白のまま残されたとき発生します。この警告は、インポートまたは変換処理中に InstallShield がファイル検索を構成できなかったことをアラートします。 この問題を解決するには、Visual Studio でファイル検索のファイル名を指定してから、Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートします。これ以外の方法としては、Visual Studio プロジェクトを変換またはインポートした後に InstallShield の [システム検索] ビューでファイル検索処理を追加または構成できます。
-9055	機能 '%1' は、InstallShield プロジェクトに既に存在します。機能は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定の機能が含まれていて、同じ機能が既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。
-9056	ディレクトリ '%1' は、InstallShield プロジェクトに既に存在します。フォルダー '%2' は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定のフォルダーが含まれていて、同じフォルダーが既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[ファイルとフォルダー] ビューをチェックして、警告メッセージに示されたフォルダーが InstallShield プロジェクトから不足していないかどうか確認してください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9057	ファイル キー '%1' は、InstallShield プロジェクトに既に存在します。ファイル '%2' は変換されませんでした。	この警告は、特定のファイル キーと関連付けられているファイルを含む Visual Studio プロジェクトを、同じファイル キーを持つファイルを既に含む InstallShield プロジェクトにインポートしたときに発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[コンポーネント] ビューの [ファイル] 領域をチェックして、警告メッセージに示されたファイルが InstallShield プロジェクトから不足していないかどうか確認してください。 [ダイレクト エディター] ビューには、プロジェクトに含まれる各ファイルに関連付けられたファイル キーが表示されます。ファイル キーは、.msi テーブルの File テーブルのプライマリ キーです。File テーブルは、重複するファイル キーを持つことができません。
-9058	ショートカット キー '%1' は、InstallShield プロジェクトに既に存在します。ショートカット '%2' は変換されませんでした。	この警告は、特定のショートカット キーと関連付けられているショートカットを含む Visual Studio プロジェクトを、同じショートカット キーを持つショートカットを既に含む InstallShield プロジェクトにインポートしたときに発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[ショートカット] ビューをチェックして、警告メッセージに示されたショートカットが InstallShield プロジェクトから不足していないかどうか確認してください。
-9059	拡張子 '%1' は、InstallShield プロジェクトに既に存在します。ファイルの種類 '%2' は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定の拡張子が含まれていて、同じ拡張子が既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[コンポーネント] ビューの [詳細設定] 領域をチェックして、警告メッセージに示されたファイル拡張子が InstallShield プロジェクトから不足していないかどうか確認してください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
–9060	レジストリ キー '%1' は、InstallShield プロジェクトに既に存在します。レジストリ エントリ 'キー: %2 値: %3 データ: %4' は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定のレジストリ エントリが含まれていて、同じレジストリ エントリが既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[レジストリ] ビューをチェックして、警告メッセージに示されたレジストリ エントリが InstallShield プロジェクトから不足していないかどうか確認してください。
–9061	カスタム アクション '%1' は、InstallShield プロジェクトに既に存在します。カスタム アクション %2 は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定のカスタム アクションが含まれていて、同じ名前を持つカスタム アクションが既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[カスタム アクションとシーケンス] ビューをチェックして、警告メッセージに示されたカスタム アクションが InstallShield プロジェクトから不足していないかどうか確認してください。
–9062	起動条件 '%1' は、InstallShield プロジェクトに既に存在します。起動条件 %2 は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定の起動条件が含まれていて、同じ名前を持つ起動条件が既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[一般情報] ビューの "インストール条件" 設定をチェックして、警告メッセージに示された起動条件が InstallShield プロジェクトから不足していないかどうか確認してください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
–9063	RegLocator '%1' は、InstallShield プロジェクトに既に存在します。レジストリ検索 '%2' は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定のレジストリ検索が含まれていて、同じ名前を持つレジストリ検索が既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[システム検索] ビューをチェックして、警告メッセージに示されたレジストリ検索が InstallShield プロジェクトから不足していないかどうか確認してください。
–9064	DrLocator '%1' は、InstallShield プロジェクトに既に存在します。ファイル検索 '%2' は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定のファイルまたはフォルダー検索が含まれていて、同じ名前を持つファイルまたはフォルダー検索が既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[システム検索] ビューをチェックして、警告メッセージに示されたファイルまたはフォルダー検索が InstallShield プロジェクトから不足していないかどうか確認してください。
–9065	CompLocator '%1' は、InstallShield プロジェクトに既に存在します。Windows Installer 検索 '%2' は変換されませんでした。	この警告は、Visual Studio プロジェクトに特定のファイルまたはフォルダー検索が含まれていて、同じ名前を持つファイルまたはフォルダー検索が既に含まれている InstallShield プロジェクトにその Visual Studio プロジェクトをインポートした場合に発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[システム検索] ビューをチェックして、警告メッセージに示されたファイルまたはフォルダー検索が InstallShield プロジェクトから不足していないかどうか確認してください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9070	前提条件を変換中にエラーが発生しました。	このエラーは、前提条件を含む Visual Studio セットアップ プロジェクトをインポートするときに、InstallShield が Visual Studio 前提条件を対応する InstallShield 前提条件にマップしようとして問題が起こった場合に発生します。 このエラーを解決するためには、テクニカル サポート に連絡するか、必要な再配布可能ファイルをインストールする InstallShield 前提条件を作成して、それをプロジェクトに追加する方法を考慮してください。詳細については、「InstallShield 前提条件を定義する」および「インストール プロジェクトに含まれている InstallShield 前提条件を利用する」を参照してください。
-9071	InstallShield には、'%1' に対応する InstallShield 前提条件がありません。前提条件は変換されませんでした。	この警告は、前提条件を含む Visual Studio セットアップ プロジェクトをインポートするときに、InstallShield で対応する InstallShield 前提条件がない場合に発生します。 この警告を解決するためには、必要な再配布可能ファイルをインストールする InstallShield 前提条件を作成して、それをプロジェクトに追加する方法を考慮してください。詳細については、「InstallShield 前提条件を定義する」および「インストール プロジェクトに含まれている InstallShield 前提条件を利用する」を参照してください。
-9074	プロジェクト出力を変換中にエラーが発生しました。	このエラーは、1 つ以上のプロジェクト出力を含む Visual Studio セットアップまたはマージ モジュール プロジェクトを変換するとき、InstallShield が プロジェクト出力を InstallShield プロジェクトに組み込むときに問題が発生した場合に起こります。 このエラーを解決するには、テクニカル サポート にご連絡ください。
-9075	プロジェクト出力 %1 を変換中にエラーが発生しました。種類: %2	このエラーは、1 つ以上のプロジェクト出力を含む Visual Studio セットアップまたはマージ モジュール プロジェクトを変換するとき、InstallShield が プロジェクト出力を InstallShield プロジェクトに組み込むときに問題が発生した場合に起こります。 このエラーを解決するには、テクニカル サポート にご連絡ください。

テーブル 11・Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9076	プロジェクト出力の種類 %1 が無効です。プロジェクト出力 %2 は変換されませんでした。	このエラーは、1 つ以上のプロジェクト出力を含む Visual Studio セットアップまたはマージ モジュール プロジェクトを変換するとき、プロジェクト出力の 1 つが無効な場合に起こります。 このエラーを解決するためには、Visual Studio プロジェクトのプロジェクト出力を確認して、問題をすべて解決してください。その後、Visual Studio プロジェクトを InstallShield プロジェクトに変換します。
-9077	ファイル キー '%1' は、InstallShield プロジェクトに既に存在します。プロジェクト出力 '%2' は変換されませんでした。	この警告は、特定のファイル キーと関連付けられているファイル持つプロジェクト出力を含む Visual Studio プロジェクトを、同じファイル キーを持つファイルを既に含む InstallShield プロジェクトにインポートしたときに発生します。この状況は、同じ Visual Studio プロジェクトを InstallShield プロジェクトに複数回にわたってインポートする場合に起こります。 この警告が発生した場合、[ファイルとフォルダー] ビューをチェックして、警告メッセージに示されたファイルが InstallShield プロジェクトから不足していないかどうか確認してください。 [ダイレクト エディター] ビューには、プロジェクトに含まれる各ファイルに関連付けられたファイル キーが表示されます。ファイル キーは、.msi テーブルの File テーブルのプライマリ キーです。File テーブルは、重複するファイル キーを持つことができません。
-9078	InstallShield プロジェクトが Visual Studio ソリューションにありません。プロジェクト出力 '%1' は変換されませんでした。	このエラーは、InstallShield プロジェクトが InstallShield で開かれていて、Visual Studio 内部から開かれていない場合に、プロジェクト出力を含む Visual Studio プロジェクトを InstallShield プロジェクトにインポートしようとすると発生します。 このエラーを解決するには、InstallShield プロジェクトを Visual Studio 内部から開いて、そこに Visual Studio プロジェクトをインポートします。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-9079	現在の Visual Studio ソリューション内で Visual Studio プロジェクト '%1' が見つかりませんでした。プロジェクト出力 '%2' は変換されませんでした。	プロジェクト出力含む Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield プロジェクトにインポートする場合、その Visual Studio プロジェクトは、そのプロジェクトのすべての依存関係と同じソリューション内になくてはなりません。そうでない場合、Visual Studio プロジェクトをインポートしようとしたときに、このエラーが発生します。 このエラーを解決するには、インポートする Visual Studio セットアップまたはマージ モジュール プロジェクトを含んでいるのと同じ Visual Studio ソリューションに InstallShield プロジェクトがあることを確認してください。また、そのソリューションに Visual Studio セットアップまたはマージ モジュール プロジェクトの依存関係であるその他の Visual Studio プロジェクトのすべてが含まれていることも確認してください。確認が終わってから、Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield プロジェクトにインポートできます。
-9080	プロジェクト出力グループ '%1' が Visual Studio プロジェクト '%2' 中に見つかりませんでした。プロジェクト出力 '%3' は変換されませんでした。	このエラーは、プロジェクト出力を含む Visual Studio セットアップまたはマージ モジュール プロジェクトを変換する際に、InstallShield がそのグループのプロジェクト出力を InstallShield プロジェクトに組み込むときに問題が発生した場合に起こります。 このエラーを解決するには、テクニカル サポート にご連絡ください。
-9081	ファイル '%1' は、セットアップまたはマージ モジュールから除外するように指定さています。ファイルは変換されませんでした。	この警告は、Exclude プロパティが True に設定されているファイルを含む Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield プロジェクトにインポートする場合に発生します。 指定されたファイルを InstallShield プロジェクトから除外する場合は、この警告を無視してください。 指定されたファイルを InstallShield プロジェクトに含める場合は、InstallShield の [ファイルとフォルダー] ビューを使ってこれを追加できます。詳細については、「プロジェクト ファイルに追加してターゲット システムにインストールする」を参照してください。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
–9082	プロジェクト出力 '%1' は、セットアップまたはマージ モジュールから除外するように指定さています。プロジェクト出力は変換されませんでした。	この警告は、プロジェクト出力にファイルを含む Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield プロジェクトにインポートする場合で、そのファイルの Exclude プロパティに True が選択されているときに発生します。 指定されたファイルを InstallShield プロジェクトから除外する場合は、この警告を無視してください。 指定されたファイルを InstallShield プロジェクトに含める場合は、InstallShield の [ファイルとフォルダー] ビューを使ってこれを追加できます。詳細については、「Visual Studio ソリューションにリファレンスを追加する」を参照してください。
–9083	Visual Studio のデフォルト起動条件は、InstallShield でサポートされていません。起動条件 %1 は変換されませんでした。	この警告は、デフォルト起動条件を含む Visual Studio セットアップまたはマージ モジュール プロジェクトを InstallShield プロジェクトにインポートしたときに発生します。 指定された起動条件を InstallShield プロジェクトから除外する場合は、この警告を無視してください。 Visual Studio が元のプロジェクトに追加したデフォルト起動条件をターゲット システム上でチェックする起動条件を InstallShield プロジェクトに追加するには、InstallShield の [システム検索] ビューを使います。詳細については、「システム検索の追加」を参照してください。
–9086	'%2' と名づけられたファイル検索の MinDate '%1' が無効です。MinDate プロパティは変換されませんでした。	この警告は、MinDate プロパティに無効な値を持つファイル検索を含む Visual Studio セットアップをインポートまたは変換しようとしたときに発生します。 この警告を解決するには、以下の手順に従います: 1. InstallShield で **[システム検索]** ビューを開きます。 2. ファイルを右クリックして、**[変更]** をクリックします。**システム検索ウィザード**が開きます。 3. このウィザードの 2 番目のパネルで、**[詳細]** ボタンをクリックします。**[ファイルの詳細]** ダイアログ ボックスが開きます。 4. 適切な詳細を入力します。

テーブル 11 • Visual Studio プロジェクトのインポート エラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
–9087	'%2' と名づけられたファイル検索の MaxDate '%1' が無効です。MaxDate プロパティは変換されませんでした。	この警告は、MaxDate e プロパティに無効な値を持つファイル検索を含む Visual Studio セットアップをインポートまたは変換しようとしたときに発生します。 この警告を解決するには、以下の手順に従います: 1. InstallShield で **[システム検索]** ビューを開きます。 2. ファイルを右クリックして、**[変更]** をクリックします。**システム検索ウィザード**が開きます。 3. このウィザードの 2 番目のパネルで、**[詳細]** ボタンをクリックします。**[ファイルの詳細]** ダイアログ ボックスが開きます。 4. 適切な詳細を入力します。
–9088	InstallShield では Visual Studio Web セットアップ プロジェクトを InstallShield プロジェクトに変換することはできません。	このエラーは、Visual Studio Web セットアップ プロジェクトを InstallShield にインポートまたは変換しようとしたときに発生します。InstallShield は、Visual Studio Web セットアップ プロジェクトの変換をサポートしません。この種類のエラーが発生した場合、InstallShield で新しいプロジェクトを作成し、[IIS 構成] ビューを使ってターゲット システム上の IIS Web サイトを管理する方法を考慮してください。
–9089	InstallShield では Visual Studio CAB プロジェクトを InstallShield プロジェクトに変換することはできません。	このエラーは、Visual Studio CAB セットアップ プロジェクトを InstallShield にインポートまたは変換しようとしたときに発生します。InstallShield は、Visual Studio CAB セットアップ プロジェクトの変換をサポートしません。
–9090	Visual Studio プロジェクトの種類を識別できませんでした。このため、プロジェクトは変換されませんでした。	このエラーは、InstallShield がその種類を識別できない Visual Studio プロジェクトをインポートまたは変換しようとしたときに発生します。
–10000	ユーザーにより変換がキャンセルされました。	このエラーは、変換処理を途中でキャンセルしたときに発生します。

アップグレード エラー (InstallShield Professional からのアップグレード)

このテーブルは、InstallShield Professional で作成したプロジェクト (.ism ファイル) を InstallShield にアップグレードしたときに生じる可能性があるエラーについてトラブルシューティングのヒントを示しています。

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント

エラー番号	メッセージ	トラブルシューティングのヒント
-61	InstallShield Professional プロジェクトのバージョン情報を取得できませんでした。Version 5.5 として実行します。	変換しようとしているプロジェクトが InstallShield Professional バージョン 5.5 以降のものであることを確認してください。セットアップが古いバージョンの InstallShield Professional を使って作成されている場合は、サポート Web サイトを参照してください。
-62	ファイル <FileName> 中の [Language] セクションに LanguageSupport キーがありませんでした。	InstallShield Professional セットアップがサポートしている言語を特定できませんでした。サポート言語を手動で定義してください。
-63	ファイル <FileName> 中の [Data] セクションに ProductName キーがありませんでした。	InstallShield Professional プロジェクトの名前を特定できませんでした。[概要情報ストリーム] の "サブジェクト" フィールドに、この情報を手動で入力してください。
-64	ファイル <FileName> 中の [Data] セクションに Author キーがありませんでした。	InstallShield Professional セットアップの名前を特定できませんでした。[概要情報ストリーム] の "作成者" フィールドに、この情報を手動で入力してください。
-65	ファイル <FileName> 中の [Data] セクションに Version キーがありませんでした。	セットアッププロジェクトのバージョンを特定できませんでした。[一般情報] ビューの "製品バージョン" 設定に、この情報を手動で入力してください。
-66	ファイル <FileName> 中の [Data] セクションに HomeUrl キーがありませんでした。	アプリケーションの URL を特定できませんでした。[一般情報] ビューの "発行元/製品 URL"、"サポート URL"、および "製品アップデート URL" 設定のいずれか、またはすべてに、この情報を入力してください。
-67	ファイル <FileName> 中の [Data] セクションに CompanyName キーがありませんでした。	InstallShield Professional プロジェクトファイルから会社名を読み取れませんでした。この情報を [一般情報] ビューの "発行元" 設定に手動で入力してください。
-68	ファイル <FileName> 中の [Data] セクションに InstallationGuid キーがありませんでした。	インストールの GUID を特定できませんでした。[一般情報] ビューの "製品コード" 設定に新しい GUID が作成されました。セットアップ用の元の GUID が分かる場合は、生成された新規 GUID を上書きすることができます。

テーブル 12 ・ アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
-69	ファイル <FileName> 中の [Data] セクションに CurrentFileGroupDef キーがありませんでした。	プロジェクトのファイル グループへの参照が見つかりませんでした。InstallShield Professional のファイルグループ設定を確認し、プロジェクトを保存して、再度プロジェクトをアップグレードしてみてください。
-70	ファイル <FileName> 中の [Data] セクションに CurrentComponentDef キーがありませんでした。	プロジェクトのコンポーネントへの参照が見つかりませんでした。InstallShield Professional プロジェクトのコンポーネント設定を確認し、プロジェクトを保存して、再度プロジェクトをアップグレードしてみてください。
-79	ファイル <.ipr ファイル> のセクション <SectionName> にキー Platforms が見つかりません.	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-80	ファイル <.ipr ファイル> のセクション <SectionName> にキー Password が見つかりません.	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-81	警告:著作権、所属、電子メール、および概要設定が使用されていません。 警告:Descriptions.txt、Instructions.txt および Notes.txt ファイルは使用されません。 警告:[プロジェクト] メニューの [ソースコントロール] オプションを使用して、ソースコントロールにプロジェクトを追加します。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-82	ファイル <.ipr ファイル>のセクション <SectionName> にキー RunDebug がみつかりません.	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-241	ファイル <FileName> の <FileGroupName> セクションに SELFREGISTERING キーがありませんでした	ファイルグループが自動登録かどうかを特定する際にエラーが発生しました。デフォルトでは、このファイルグループから作成された新しいコンポーネントは自動登録として設定されません。この設定は、該当するコンポーネントのプロパティページで変更できます。

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
–242	ファイル <FileName> の <FileGroupName> セクションに TARGET キーがありませんでした	1 つのファイルグループのターゲットディレクトリが特定できませんでした。このファイルグループから作成された 1 つ以上のコンポーネントは、デフォルトで <INSTALLDIR> をポイントします。この設定は、該当するコンポーネントの ″インストール先″ プロパティで変更できます。
–243	ファイル <FileName> の <FileGroupName> セクションに LANGUAGE キーがありませんでした	1 つのファイルグループのターゲット言語が特定できませんでした。このファイルグループから作成されたコンポーネントは言語非依存となります。この設定は、適切なコンポーネントの ″言語″ プロパティで変更できます。
–244	ファイル <FileName> 中の [DYNAMIC] セクションに FOLDER キーがありませんでした	1 つのファイルグループで、動的ファイルリンクへの参照が見つかりませんでした。結果として、この動的リンクにより参照されるファイルはいずれも移行されませんでした。InstallShield Professional プロジェクトを開いて動的リンク設定を確認するか、手動でこれらのファイルを新しいプロジェクトに追加できます。
–245	ファイル <FileName> 中の [DYNAMIC] セクションに INCLUDESUBDIR キーがありませんでした	1 つのファイルグループで、サブフォルダーを動的リンクの一部に含めるかどうかを特定できませんでした。結果として、新しいプロジェクトにサブフォルダーは含まれません。InstallShield Professional プロジェクトを開いて動的リンク設定を確認するか、手動でこれらのファイルを新しいプロジェクトに追加できます。
–246	ファイル <FileName> 中の [DYNAMIC] セクションに WILDCARD キーがありませんでした	1 つの動的リンクに指定したワイルドカードを読み取れませんでした。結果として、動的リンクが指定されたディレクトリに含まれているすべてのファイルがセットアップに追加されました。InstallShield Professional プロジェクトを開いて動的リンク設定を確認するか、手動で不要なファイルを新しいプロジェクトから削除できます。
–247	ファイル <Path>¥Default.fdf の < セクション名> セクションの下のキー FILETYPE が見つかりませんでした。	1 つのファイル グループが共有指定されていたかどうかを特定できませんでした。結果として、そのファイルグループから作成されたコンポーネントは共有指定されません。この設定を変更するには、該当するコンポーネントの ″共有″ プロパティを *[はい]* に設定してください。
–252	ファイル <.ipr ファイル>のセクション <SectionName> にはキー Folder が見つかりません	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
-281	ファイル <Path>¥Default.fdf を開くときにエラーが発生しました	ファイルグループへの参照が見つかりませんでした。InstallShield Professional プロジェクトのファイルグループ設定を確認し、保存してから、InstallShield 2025 でプロジェクトを開き直してください。
-282	次のファイルを開くときにエラーが発生しました: <ファイル名>	アップグレードは 1 つのファイルグループの場所を特定できませんでした。したがって、このファイルグループに関連するファイルはセットアップに追加されませんでした。InstallShield Professional プロジェクトのファイルグループ設定を確認し、保存してから、InstallShield 2025 でプロジェクトを開き直してください。
-283	ファイル <Path>¥Default.fdf 中に [FILEGROUPS] セクションが見つかりません。	ファイルグループへの参照が見つかりませんでした。InstallShield Professional プロジェクトのファイルグループ設定を確認し、保存してから、InstallShield 2025 でプロジェクトを開き直してください。
-284	ファイル <Path>¥Default.fdf 中に <SectionName> セクションが見つかりません。	ファイルグループへの参照が見つかりませんでした。結果として、このファイルグループに関連するファイルはプロジェクトに追加されませんでした。InstallShield Professional プロジェクトのファイルグループ設定を確認し、保存してから、InstallShield 2025 でプロジェクトを開き直してください。
-285	ファイル <Path>¥Default.fdf の <セクション名> セクションの下のキー FILETYPE が見つかりませんでした。	InstallShield は 1 つのファイルグループのファイルがダイナミック (動的) とスタティック (静的) のどちらでリンクされているかを特定できませんでした。結果として、そのファイル グループのファイルはセットアップに追加されませんでした。InstallShield Professional プロジェクトのファイルグループ設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-286	ファイル <ファイル名> に [DYNAMIC] セクションを見つけることができませんでした	InstallShield は 1 つのファイルグループのファイルがダイナミック (動的) とスタティック (静的) のどちらでリンクされているかを特定できませんでした。結果として、そのファイル グループのファイルはセットアップに追加されませんでした。InstallShield Professional でのファイルグループ設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-301	ファイル <Path>¥Default.cdf 中の <セクション名> セクションの下に DESCRIPTION キーが見つかりませんでした。	1 つのコンポーネントの [説明] プロパティを読み取れませんでした。コンポーネントから作成された機能の [説明] プロパティに、この情報を手動で入力してください。

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
-302	ファイル <Path>¥Default.cdf 中の <セクション名> セクションに DISPLAYTEXT キーが見つかりませんでした。	1 つのコンポーネントの表示名を読み取れませんでした。コンポーネントから作成された機能の "表示名" プロパティに、この情報を手動で入力してください。
-303	ファイル <Path>¥Default.cdf 中の <セクション名> セクションに COMMENT キーが見つかりませんでした。	1 つのコンポーネントのコメントを読み取れませんでした。該当するコンポーネントの "コメント" プロパティに、この情報を手動で入力してください。
-306	ファイル <.ipr ファイル> のセクション <セクション名> でキー <キー名> が見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-341	次のファイルを開くときにエラーが発生しました: <パス>¥Default.cdf	コンポーネントのインポート時にエラーが発生しました。この結果、機能は作成されませんでした。InstallShield Professional でのコンポーネント設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-342	ファイル <Path>¥Default.cdf 中に [COMPONENTS] セクションが見つかりません。	コンポーネントのインポート時にエラーが発生しました。この結果、機能は作成されませんでした。InstallShield Professional プロジェクトのコンポーネント設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-343	ファイル <Path>¥Default.cdf 中に <SectionName> セクションが見つかりません。	1 つのコンポーネントのインポート時にエラーが発生しました。結果として、そのコンポーネントは移行されませんでした。InstallShield Professional プロジェクトのコンポーネント設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-401	ファイル <Path>¥Default.reg を開くときにエラーが発生しました	InstallShield Professional プロジェクトに関連するレジストリ情報を読み取れませんでした。結果として、レジストリ情報は移行されませんでした。InstallShield Professional プロジェクトのレジストリ設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-402	ファイル <Path>¥Default.re 中に [DATA] セクションが見つかりません。	InstallShield Professional プロジェクトに関連するレジストリ情報を読み取れませんでした。結果として、レジストリ情報は移行されませんでした。InstallShield Professional プロジェクトのレジストリ設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
-403	ファイル <Path>¥Default.re 中に <SectionName> セクションが見つかりません。	レジストリ情報の一部を読み取れませんでした。結果として、レジストリデータは移行されませんでした。InstallShield Professional プロジェクトのレジストリ設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-404	ファイル <Path>¥Default.re 中に <SectionName> セクションが見つかりません。	レジストリ情報の一部を読み取れませんでした。結果として、レジストリデータは移行されませんでした。InstallShield Professional プロジェクトのレジストリ設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-421	ファイル <Path>¥Default.shl 中の <SectionName> セクションに TARGET キーが見つかりません。	1 つのショートカットのターゲットが特定できませんでした。該当するショートカットの "ターゲット" プロパティに、この情報を手動で入力してください。
-422	ファイル <Path>¥Default.shl 中の <SectionName> セクションに ICONFILE キーが見つかりません。	1 つのショートカットのアイコンが特定できませんでした。該当するショートカットの "アイコン ファイル" プロパティに、この情報を手動で入力してください。
-423	ファイル <Path>¥Default.shl 中の <SectionName> セクションに ICONINDEX キーが見つかりません。	1 つのショートカットのアイコンインデックスが特定できませんでした。該当するショートカットの "アイコンインデックス" プロパティに、この情報を手動で入力してください。
-424	ファイル <Path>¥Default.shl 中の <SectionName> セクションに DISPLAYTEXT キーが見つかりません。	1 つのショートカットのショートカットテキストが特定できませんでした。該当するショートカットの "表示名" プロパティに、この情報を手動で入力してください。
-425	ファイル <Path>¥Default.shl 中の <SectionName> セクションに PARAMETERS キーが見つかりません。	1 つのショートカットの引数が特定できませんでした。該当するショートカットの "引数" プロパティに、この情報を手動で入力してください。
-426	ファイル <Path>¥Default.shl 中の <SectionName> セクションに SHORTCUTKEY キーが見つかりません。	1 つのショートカットのショートカットキー特定できませんでした。該当するショートカットの "ホットキー" プロパティに、この情報を手動で入力してください。
-427	ファイル <Path>¥Default.shl 中の <SectionName> セクションに STARTINY キーが見つかりません。	1 つのショートカットの先頭ディレクトリが特定できませんでした。該当するショートカットの "作業ディレクトリ" プロパティに、この情報を手動で入力してください。

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
–428	ファイル <Path>¥Default.shl 中の <SectionName> セクションに COMMENTS キーが見つかりません。	1 つのショートカットのコメントを読み取れませんでした。該当するショートカットの "コメント" プロパティに、この情報を手動で入力してください。
–429	ファイル <Path>¥Default.shl 中の <SectionName> セクションに RUN キーが見つかりません。	1 つのショートカットの "実行" プロパティを読み取れませんでした。このプロパティはデフォルトでは、該当するショートカットの "実行" プロパティで [標準] に設定されます。
–430	ファイル <.ipr ファイル> のセクション <セクション名> で Typeunder キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
–431	ファイル <.ipr ファイル> のセクション <セクション名> で Uninstall キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
–432	ファイル <.ipr ファイル> のセクション <セクション名> で DisplayText キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
–433	ファイル <.ipr ファイル> のセクション <セクション名> で Shared キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
–434	ファイル <.ipr ファイル> のセクション <セクション名> で Replace キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
–435	ファイル <.ipr ファイル> のセクション <セクション名> で InternetShortcut キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
–440	ファイル <.ipr ファイル> のセクション <セクション名> で EngineBinding キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.

テーブル 12 • アップグレードエラーのトラブルシューティングに関するヒント (続き.)

エラー番号	メッセージ	トラブルシューティングのヒント
-441	ファイル <.ipr ファイル> のセクション <セクション名> で UpdateMode キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-442	ファイル <.ipr ファイル> のセクション <セクション名> で UpdateService キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-443	ファイル <.ipr ファイル> のセクション <セクション名> で LogFile キーが見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-461	<Path>¥Default.shl ファイルを開けません	InstallShield Professional プロジェクトに関連するシェルオブジェクト情報を読み取れませんでした。結果として、ショートカットは作成されませんでした。InstallShield Professional プロジェクトでシェルオブジェクト設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-462	<Path>¥Default.shl ファイルで [DATA] セクションが見つかりません	InstallShield Professional プロジェクトに関連するシェルオブジェクト情報を読み取れませんでした。結果として、ショートカットは作成されませんでした。InstallShield Professional プロジェクトでシェルオブジェクト設定を確認し、保存してから、InstallShield でプロジェクトを開き直してください。
-463	ファイル <Path>¥Default.shl 中の [DATA] セクションに <キー名> キーが見つかりません。	1 つのショートカットを移行できませんでした。ショートカットエクスプローラーで、このショートカットを手動で作成してください。
-464	<Path>¥Default.shl ファイルで <SectionName> セクションが見つかりませんでした	1 つのショートカットを移行できませんでした。ショートカットエクスプローラーで、このショートカットを手動で作成してください。
-471	<.ipr file> ファイルで <SectionName> セクションが見つかりませんでした	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.
-500	ファイル <.ipr ファイル> のセクション <セクション名> でキー <キー名> が見つかりませんでした。	InstallShield Professional 7.0以前のバージョンで作成されたプロジェクトの場合、この警告メッセージは無視してください.

アップグレードの警告 (InstallShield Professional からのアップグレード)

InstallShield Professional と InstallShield 2025 のアーキテクチャの違いにより、インストール プロジェクトをアップグレードすると、マニュアルでの調整が必要になることがあります。

プロジェクト内の設定をマニュアルで調整する必要がある場合、プロジェクトをアップグレードした後に、出力ウィンドウに警告が表示されます。必要措置に関しては、関連ヘルプトピックを参照してください。(続き.)

テーブル 13 ・ アップグレードの警告

警告番号	メッセージ
-289	動的にリンクされたファイルのファイルリンクが存在しません
-305	InstallShield Professional コンポーネントの ビルドに含める プロパティを直接移行できませんでした
-480	共有属性は、コンポーネントのキーファイルにのみ設定できます。
-481	共有属性は、コンポーネントのキーファイルにのみ設定できます。動的にリンクされたファイルはキーファイルとして使用できません。
-486	ショートカットアイコンは設計時にリンクする必要があります。
-487	InstallShield 2025 では、ファイルのインストールはファイルのバージョン規則に従って行われます。
-488	スクリプトに廃止されたイベントが見つかりました。
-495	ODBC コアはデフォルトでインストールされていません。セットアップ プロジェクトでこの機能が必要な場合は、[再配布可能ファイル] ビューに移動して、MDAC マージ モジュールを選択してください。

更新の警告 289: 動的にリンクされたファイルのファイルリンクが存在しません

InstallShield Professional では、ソースファイルへのダイナミックリンクによって、パスやファイル名をハードコード化しなくても、フォルダーおよびサブフォルダーの全内容をセットアップに追加できます。ソースフォルダーにファイルを追加したり、ファイルを削除したりすると、含まれるファイルのリストがリリースのビルド時にダイナミック (動的) に更新されました。

InstallShield 2025 へのアップグレード

InstallShield Professional を InstallShield 2025 にアップグレードして、存在しないダイナミックファイルリンクを使用してファイルを含めると、アップグレード警告 289 が表示されます。この警告は、ファイルリンクは作成されているが、ファイルが存在しないことを知らせます。

更新の警告 -305: InstallShield Professional コンポーネントの "ビルドに含める" プロパティを直接移行できませんでした

InstallShield Professional プロジェクトのコンポーネントで " ビルドに含める" プロパティが [いいえ] に設定されている場合、対応する機能は "リリース フラグ" プロパティが EXCLUDEFROMBUILD に設定されています。セットアップから機能を除外するには、[リリース] ビューか、リリース ウィザードの [フィルタリング設定] パネルでリリースフラグを手動設定 する必要があります。

メモ: *[リリース] ビューのリリースフラグはセットアップに含める機能を指定するため、[リリース] ビューで指定するフラグは、EXCLUDEFROMBUILD 以外にする必要があります。*

更新の警告 -480: ファイル グループ中のファイルの共有 .dll 参照カウントのインクリメント

InstallShield Professional バージョン 6.x 以前では、ファイルグループの共有プロパティを設定すると、そのファイルグループの *すべて*のファイルの共有 DLL 参照カウントがインストール時にインクリメントされていました。アップグレード時に、この設定は、InstallShield 2025 でこのファイルグループ用に作成されたコンポーネントの共有属性に移行されます。

共有 .dll 参照カウントの動作

InstallShield 2025では、コンポーネントの共有属性を設定すると、そのコンポーネントのキーファイル*のみ*の共有 .dll 参照カウントがインクリメントされます。 では、コンポーネントの共有属性を設定すると、そのコンポーネントのキーファイルのみの共有 .dll 参照カウントがインクリメントされます。1 つのコンポーネントにつき定義できるのは 1 つのキーファイルだけです。複数のファイルの共有参照カウントをインクリメントするには、各ファイルに独自のコンポーネントが必要で、またそのファイルがそのコンポーネントのキーファイルである必要があります。

メモ: *.exe、.dll および .ocx ファイルについては、コンポーネントウィザードを使用して、大量のファイルのコンポーネントを素早く作成できます。*

InstallShield 2025 では、ファイルが共有指定されていなくてもファイルの共有 .dll 参照カウントが存在すれば、そのファイルをインストールすると参照カウントは自動的にインクリメントされ、アンインストールするとデクリメントされます。

さらに、コンポーネントの参照カウントは、コンポーネント GUID (または "コンポーネントコード" プロパティ) に基づいて維持されます。したがって、コンポーネントの共有プロパティは、以前のインストールにファイルの共有がある可能性がある場合や、別のコンポーネント GUID を持つコンポーネントがこのファイルをインストールした場合にのみ役立ちます。

更新の警告 -481: ファイルグループ内のダイナミックリンクファイルの共有 DLL 参照カウントのインクリメント

InstallShield Professional バージョン 6.x 以前では、ダイナミックファイルリンクを持つファイルグループの共有プロパティを設定すると、ファイルリンクタイプに関わらず、そのファイルグループの *すべて*のファイルの共有 DLL 参照カウントがインストール時にインクリメントされていました。アップグレード時に、この設定は、InstallShield 2025 でこのファイルグループ用に作成されたコンポーネントの共有属性に移行されます。

共有 .dll 参照カウントの動作

では、コンポーネントの共有属性を設定すると、そのコンポーネントのキーファイル*のみ*の共有 .dll 参照カウントがインクリメントされます。動的にリンクされたファイルはキーファイルとして使用できないため、コンポーネントのファイルの共有参照カウントはインクリメントされません。

コンポーネントは、1 つのキーファイルしか使用できません。したがって、複数のファイルの共有参照カウントをインクリメントするには、各ファイルに独自のコンポーネントが必要で、またそのファイルがそのコンポーネントのキーファイルである必要があります。各ファイルについて、参照カウントで共有 DLL 参照カウントをインクリメントするようにするには、スタティック (静的) にファイルをリンクする必要があります。

メモ: EXE、DLL および OCX ファイルについては、コンポーネントウィザードを使用して、大量のファイルのコンポーネントを素早く作成できます。

InstallShield 2025 では、ファイルが共有指定されていなくてもファイルの共有 DLL 参照カウントが存在すれば、そのファイルをインストールすると参照カウントは自動的にインクリメントされ、アンインストールするとデクリメントされます。

さらに、コンポーネントの参照カウントは、コンポーネント GUID (または "コンポーネントコード" プロパティ) に基づいて維持されます。したがって、コンポーネントの共有プロパティは、以前のインストールにファイルの共有がある可能性がある場合や、別のコンポーネント GUID を持つコンポーネントがこのファイルをインストールした場合にのみ役立ちます。

更新の警告 -486: ショートカット アイコンの指定

InstallShield Professional でのショートカットの作成

InstallShield Professional バージョン 6.x および以前のバージョンでは、ショートカットアイコンはターゲットマシンの視点から作成されていました。たとえば、ショートカットのアイコンが <TARGETDIR>¥MyIcon.ico にインストールされている場合、これがショートカットアイコンに指定された修飾パスでした。実行時には、ショートカット作成プロセスは、ターゲットマシンのアイコンにバインドしました。

ショートカットの作成

InstallShield 2025 で、ショートカットアイコンは設計時にリンクする必要があります。ターゲット システムにアイコンファイルとしてパスを指定する代わりに、プロジェクトファイルでビルドマシンにアイコンファイルとしてパスを指定する必要があります。そうすると、ビルドエンジンはインストール時にアイコンに個別の*ストリーム*を作成します。そして、ランタイムのアイコン作成プロセス中に、インストールはそのストリームを使用します。つまり、セットアップがショートカットに使用しているアイコンをインストールする必要はなくなりました。

InstallShield Professional プロジェクトをアップグレードするとき、InstallShield 2025 はターゲットアイコンのパスをセットアッププロジェクトに存在するファイルと一致させようとします。一致できない場合、この警告が表示されます。

ショートカットアイコンのパスの修正

ヒント: *ショートカットアイコンのパスを修正するには、次の操作を実行します。*

1. IDE で **[ショートカット]** ビューに移動して、警告メッセージに指定されたショートカットをクリックします。
2. **アイコンファイル** プロパティフィールドで、アイコンファイルを参照します。

メモ: *ショートカット アイコンは、InstallShield 2025 で必須です。アイコンが指定されていない場合、ビルドエンジンはショートカットのターゲットファイルの最初のアイコンインデックスをショートカットアイコンに使用します。*

更新の警告 –487: インストール時のファイルの上書き

ファイル グループの "上書き" プロパティに基づいたファイルのインストール

InstallShield Professional バージョン 6.x およびそれ以前のバージョンでは、個別のファイルグループに対して上書きプロパティを設定することができました。あるファイルグループは日付に基づいてファイルをインストールし、別のファイルグループはファイルバージョンがターゲットマシンのファイルバージョンより古い場合にのみファイルをインストールするという設定が可能でした。

ファイルのバージョン規則に基づいたファイルのインストール

InstallShield 2025 では、ファイルのバージョン規則という方法を使用して、ターゲットマシンにインストールされる必要のあるファイルを決定します。コンポーネントレベルでファイルのバージョン規則をオーバーライドする唯一のオプションは、上書きしない プロパティを [はい] に設定することです。

ファイルのバージョン規則の代替規則

ほとんどのアプリケーションにとってファイルのバージョン規則はデフォルトのもので十分ですが、アプリケーションの要件によっては、使用規則を管理して、ファイルを上書きすべきか決定する必要がある場合があるかもしれません。このため、ファイルのバージョン規則には個別で使用したり、相互に使用してファイルのバージョン規則を変更する代替規則があります。

REINSTALLMODE

ファイルのバージョン規則を上書きする REINSTALLMODE のテクニックは、カスタムのバージョン規則を確立するためのグローバルなメカニズムです。このテクニックを使用すると、インストールのすべてのファイルのバージョン規則が影響を受けます。

タスク ***REINSTALLMODE プロパティを使用するには、次の操作を実行します。***

1. **プロパティ マネージャー**を開きます。
2. REINSTALLMODE という名前のプロパティを作成します。
3. REINSTALLMODE プロパティに指定されたオプションコードに基づいて、このプロパティの値を設定します。

インストーラーはインストール時にこのプロパティを読み取り、その設定を使用してファイルのバージョン指定を上書きします。この設定は、再インストール時やアプリケーション修復時にもインストーラーによって使用されます。

コンパニオン ファイル

ファイルのバージョン規則のオーバーライドにコンパニオン ファイルのテクニックを使用すると、コンパニオン ファイルのインストールに基づいてファイルをインストールするようにファイルを構成できます。このテクニックは、いくつかのファイルのインストール状態をバインドした方がよい場合に役立ちます。

たとえば、バージョン指定されていないユーザーデータファイルをバージョン指定されている .exe ファイルにバインドさせることができます。.exe がインストールされていない場合、ユーザーデータはインストールされません。

更新の警告 -488: スクリプト ファイルに古い形式のイベントがあります

スクリプト ファイルに古い形式のイベントが見つかりました。ほとんどの InstallShield Professional スクリプトイベントは InstallShield 2025 でサポートされていますが、利用できないものもいくつかあります。

同等の機能を持つイベント

- **OnInstallingFile**
- **OnUninstallingFile**

これらの 2 つのイベントは、**OnInstallFilesActionBefore** および **OnInstallFilesActionAfter** に置き換えられました。これらのイベントは Install、Uninstall、および Maintenance で呼び出されます。

OnInstallFilesActionBefore イベントは、インストールがコンポーネントの [アクション] 状態に基づいて、そのコンポーネントに含まれるファイルをターゲットマシンに追加、またはターゲットマシンから削除する準備を行っている間に呼び出されます。アンインストール時、ファイルへのショートカットの削除、ファイルの COM データの登録解除など、ソースファイルを必要とするすべての関連アクションは既に実行されています。

OnInstallFilesActionAfter イベントは、インストールがファイルをターゲットマシンに追加またはターゲットマシンから削除し終わったときに呼び出されます。インストール時には、NT サービスの登録や ODBC エントリの作成などのファイルに関連するインストールアクションはまだ実行されていません。

メモ: ***OnInstallFilesActionBefore*** *イベントおよび* ***OnInstallFilesActionAfter*** *イベントを使用するときには、Windows Installer API 関数の* ***MsiGetComponentState*** *を使用してコンポーネントのアクション状態を指定することができます。*

部分的にサポートされるイベント

OnSelfRegistrationError イベントは、**XCopyFile** 関数を使用してソース メディアにアプリケーション ファイルをコピーする場合に呼び出されます。

サポートされないイベント

以下のイベントは InstallShield 2025 でサポートされません:

- **OnFileReadOnly**
- **OnFileLocked**
- **OnNextDisk**
- **OnRemovingSharedFile**
- **OnFileError**
- **OnInternetError**
- **OnMD5Error**

更新の警告 495: ODBC コアがデフォルトでインストールされていません

ODBC コアはデフォルトでインストールされていません。これがセットアップに必須の場合、[再配布可能ファイル] ビューから MDAC マージ モジュールを選択してください。

ODBC インストールのための MDAC の必要条件

InstallShield Professional では ODBC コアおよび ODBC ドライバーをインストールするオブジェクトを含んでいますが、InstallShield 2025 ではこの機能を直接サポートしていません。Microsoft では ODBC を MDAC のサブセットとしてインストールすることを必要としているため、Windows Installer は ODBC コアの直接インストールをサポートしていません。

アップグレード エラーと警告 (InstallShield—Windows Installer Edition からのアップグレード)

この表は、InstallShield—Windows Installer Edition で作成したプロジェクト (.ism ファイル) を InstallShield にアップグレードしたときに生じる可能性があるエラー、および警告についてトラブルシューティングのヒントを示しています。

テーブル 14 ・ アップグレードエラーと警告

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-100	このエラーは、コンポーネントのキーパスに設定されたレジストリ値を持つプロジェクトをアップグレードしているときに表示されます。	レジストリ値をコンポーネントのキーパスとして設定することはできません。このエラーを回避するには、このキーパスをコンポーネントから削除してもう一度アップグレードを試みてください。

テーブル 14 • アップグレードエラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-201	プロジェクトに重複したファイル名をもつコンポーネントが含まれている場合、このエラーが表示されます。アップグレードはこれらのファイルの1つだけを保持し、残りを削除します。	この問題を回避するには、同一のコンポーネントの中に複製ファイル名がないようにしてください。
-2007 -2011 -2121 -2122	このエラーは、アップグレードで古いテーブル構造から情報を読み取れなかったり、新しいテーブル構造に情報を書き込めない場合に生じます。	ファイルがアップグレードされてから、エラーのあった部分に移動して設定を再挿入する必要があります。
-6020	基本の MSI プロジェクトでは、スクリプティングビルボードがサポートされなくなりました。このプロパティはアップグレードされません。	プロジェクトをアップグレードした後にこのプロパティのサポートを追加するには、プロジェクトを InstallScript MSI プロジェクトに変換して、ビルボードを [サポート ファイル/ビルボード] ビューに追加してください。
-6021	パス変数にはアポストロフィを含めることはできません。この文字は無効な文字です。名前が変更されます。	セットアップを正常にビルドするには、プロジェクトをアップグレードした後に、このパス変数を参照しているすべての参照を更新してください。
-6022	カスタム アクションで指定された MSI ファイルを開くことはできません。ソースプロパティは変換されません。	プロジェクトを正常にビルドするには、プロジェクトをアップグレードした後に、カスタム アクションの ″ソース″ プロパティに MSI ファイルの製品コードを入力してください。
-6023	システムには必要なコードページがインストールされません。関連のある文字列テーブルの値が正しくアップグレードされません。	プロジェクトをアップグレードする前に、適切なコードページをインストールしてください。
-6026	このエラーは、InstallShield の定義済みパス変数と同じユーザー定義のパス変数が InstallShield—Windows Installer Edition にある場合に発生します。	正しくアップグレードする為には、InsallShield—Windows Installer Edition プロジェクトのユーザー定義のパス変数を削除してから、プロジェクトを InstallShield にアップグレードしてください。アップグレード後、プロジェクトを InstallShield で開き、ユーザー定義パス変数に相当する新しいパス変数を作成します。InstallShield プロジェクトのすべてのリファレンスをアップデートします。

テーブル 14 • アップグレードエラーと警告 (続き.)

エラー/警告番号	メッセージ	トラブルシューティングのヒント
-6031	プロパティ名 "sOldPropertyName" に無効な文字が含まれています。名前は sNewPropertyName" に変更されます。プロジェクトを正常にビルドするには、プロジェクトをアップグレードした後に、このプロパティ名のリファレンスをすべて更新してください。	プロジェクトをアップグレードした後、古いプロパティ名のリファレンスすべてを新しいプロパティ名に変更します。

Windows Installer ランタイムエラーの HRESULT 値

ISRegSrv.dll は、ターゲット システムで自動登録ファイルを登録および登録解除するカスタム アクションから呼び出される InstallShield DLL です。この DLL は、特定の理由で失敗すると、Windows Installer ランタイムエラーの 1904 または 1905 を表示します。

- 1904: モジュール [2] の登録に失敗しました。HRESULT [3].
- 1905: モジュール [2] の登録解除に失敗しました。HRESULT [3].

[3] は、InstallShield のカスタム HRESULT である場合があります。

テーブル 15 • Windows Installer ランタイムエラーの HRESULT 値

HRESULT	説明
-2147220475	自動登録ファイルのファイル拡張子が無効です。
-2147220474	実行可能ファイルのファイル拡張子ですが実際は実行不可能です。
-2147220473	汎用登録エラー。
-2147220472	汎用登録解除エラー。

Windows Installer エラー メッセージ (Windows Installer ヘルプ ライブラリ)

DIFxAPI エラー (InstallScript プロジェクト)

このテーブルは、DIFx ドライバー関数を呼び出して戻すことができる DIFxAPI エラーのリストです。DIFx ドライバー関数からの戻り値が、Win32 エラー (正の戻り値) の場合、ISERR_WIN_BASE がエラーに追加されて、エラーが ISERR_SUCCESS 以下であることを示します。Win32 エラーに関する最新ドキュメントは、MSDN Library を参照してください。

テーブル 16 • DIFxAPI エラー (InstallScript)

戻り値	説明
ISERR_ISERVICE_NOT_ENABLED	DIFx サポートが有効になっていないか、または、Disable(SERVICE_DIFX_*)) を使用して無効にされたかを示します。DIFx サポートの有効化に関する情報については、「デバイス ドライバーのインストール」を参照してください。
ISERR_WIN_BASE + ERROR_INVALID_FUNCTION	difxapi.dll が (in SUPPORTDIR) に見つからなかった、または、エクスポートされた関数が見つからなかったことを示します。DIFx サポートがプロジェクトで有効にしたことを確認してください。
ERROR_DEPENDENT_APPLICATIONS_EXIST	関数は、ドライバー パッケージと指定されたアプリケーションの間の関連付けを削除しましたが、他のアプリケーションがドライバー パッケージに関連付けられているため関数は、ドライバー パッケージをアンインストールしませんでした。
ISERR_WIN_BASE + ERROR_ALREADY_EXISTS	ドライバー パッケージは、既にプレインストールされており、再度プレインストールされませんでした。
ISERR_WIN_BASE + ERROR_INSUFFICIENT_BUFFER	pDestInfPath バッファーが小さすぎるため、要求された .inf ファイル パスを取得できませんでした。
ISERR_WIN_BASE + ERROR_FILE_NOT_FOUND	指定された .inf ファイルが見つかりませんでした。
ERROR_DRIVER_PACKAGE_NOT_IN_STORE	.inf ファイルが、DriverPackageInfPath によって指定された .inf ファイルに対応するドライバー ストアの中に存在しません。
ISERR_WIN_BASE + ERROR_NO_MORE_ITEMS	このエラー コードは、DRIVER_PACKAGE_ONLY_IF_DEVICE_PRESENT 定数が指定されているときのみ適用しますが、DRIVER_PACKAGE_FORCE 定数が指定されていません。この場合、関数は、指定されたドライバー パッケージをプレインストールしませんでした。これは、指定されたドライバー パッケージはデバイス ツリーの中のデバイスと一致しましたが、各デバイスに既にインストールされているドライバーは、指定されたドライバー パッケージよりもそのデバイスとより適合性が高いからです。これは、存在している、および、存在していないデバイスに適用します。

テーブル 16 • DIFxAPI エラー (InstallScript) (続き.)

戻り値	説明
ERROR_NO_SUCH_DEVINST	このエラー コードは、DRIVER_PACKAGE_ONLY_IF_DEVICE_PRESENT 定数が指定されているときのみ適用します。ドライバー パッケージがデバイス ツリーのデバイスに一致しなかったため、関数は、指定されたドライバー パッケージをプレインストールしませんでした。これは、存在している、および、存在していないデバイスに適用します。
ERROR_NO_DEVICE_ID	ドライバー パッケージは、現在のプラットフォームがサポートするハードウェア ID または 互換性 IDを指定しません。
ERROR_MISSING_FILE	.inf ファイルによって参照されたファイルが見つからなかったため、関数は、ドライバーをプレインストールしませんでした。
ISERR_WIN_BASE + ERROR_CANNOT_MAKE	関数は、ドライバーをプレインストールしませんでした。
TRUST_E_NOSIGNATURE	ドライバー パッケージは、署名されていません。
CERT_E_EXPIRED	ドライバー パッケージの署名に使用された証明書は、期限が切れています。
CERT_E_WRONG_USAGE	ドライバー パッケージの証明書は、要求された使用法に有効ではありません。ドライバー パッケージが、有効 WHQL 署名を持たないため、以下の状況が発生した場合、関数は、このエラー コードを戻します。 • ドライバーの署名ダイアログ ボックスに応答して、ユーザーは、ドライバー パッケージをインストールしないことを選択しました。 • DRIVER_PACKAGE_SILENT 定数が指定されています。
CRYPT_E_FILE_ERROR	ドライバー パッケージのカタログが見つからなかった、または、関数がドライバー パッケージの署名を確認しようとしたとき他のエラーが発生した可能性があります。
ERROR_INVALID_CATALOG_DATA	ドライバー パッケージのカタログ ファイルが無効、または、見つかりませんでした。
ERROR_SIGNATURE_OSATTRIBUTE_MISMATCH	証明書は、現在の Windows バージョンに対して有効でないか、期限が切れています。
ISERR_WIN_BASE + ERROR_SHARING_VIOLATION	ドライバー ストアのドライバー パッケージのコンポーネントがスレッドまたはプロセスによってロックされています。これは、呼びだされているスレッドまたはプロセス以外のプロセスまたはスレッドが現在同じドライバー パッケージをアクセス中のとき発生します。
ISERR_WIN_BASE + ERROR_ACCESS_DENIED	管理者グループのメンバーのみ、この機能にアクセスすることができます。

テーブル 16 • DIFxAPI エラー (InstallScript) (続き.)

戻り値	説明
ISERR_WIN_BASE + ERROR_BAD_ENVIRONMENT	この呼び出しが実行された Windows バージョンは、この操作をサポートしていません。
ISERR_WIN_BASE + ERROR_INVALID_PARAMETER	指定されたパラメーターが有効ではありません。
ISERR_WIN_BASE + ERROR_INVALID_NAME	指定された .inf ファイル パスが有効ではありません。
ISERR_WIN_BASE + ERROR_FILENAME_EXCED_RANGE	指定された .inf ファイル パスの文字の長さが、サポートされている最大の長さを超えています。
ISERR_WIN_BASE + ERROR_CANT_ACCESS_FILE	指定された .inf ファイルがシステム .inf ファイル ディレクトリにないため、ドライバー パッケージ ファイルをプレインストールできませんでした。
ISERR_WIN_BASE + ERROR_OUTOFMEMORY	使用可能なシステム メモリが、ドライバー パッケージのプレインストールに足りませんでした。
ERROR_UNSUPPORTED_TYPE	ドライバー パッケージ タイプがサポートされていません。
ERROR_IN_WOW64	32 ビット バージョン DirectX は、Win64 システム上で動作しません。64 ビット バージョンの DIFxAPI が必要です。
ERROR_INSTALL_FAILURE	インストールが失敗しました。
ISERR_WIN_BASE + ERROR_INVALID_FUNCTION	ドライバー パッケージは、Plug and Play (PnP) 関数ドライバー用ではありません。

″文字列 PRODUCT_NAME が文字列テーブルで見つかりません″ エラー

InstallScript MSI プロジェクトを InstallScript プロジェクトに変換する最中、セットアッププロジェクトが実行できなかった場合にこのエラーが発生します。*SHELL_OBJECT_FOLDER = @PRODUCT_NAME;* がスクリプトに含まれているかどうかを確認してください。

メモ: *SHELL_OBJECT_FOLDER を OnFirstUIBefore で設定する必要はありません。スクリプトに以下の行が含まれている場合、これを削除してください:*

SHELL_OBJECT_FOLDER = @PRODUCT_NAME;″

InstallScript エラー情報

InstallScript コンパイラは [出力] ウィンドウにエラーメッセージを表示します。通常メイン ウィンドウの下に配置されるこのウィンドウには、[コンパイル] タブがあります。出力ウィンドウが存在しない場合、InstallShield の [表示] メニューで、[出力] をクリックします。

セットアップスクリプトに、エラーを識別する次の種類のエラーメッセージが表示されます。

テーブル 17 ・ エラーメッセージの種類

エラーメッセージの種類	範囲
警告	C7501 から C7503
構文とコンパイラー エラー	C8001 から C8522
致命的なエラー	F8501 から F8519
内部エラー	C9001

InstallScript 構文またはコンパイラ エラー

構文エラーがある場合、ファイル Setup.inx を作成できず、コンパイラによって以下のエラーメッセージの 1 つが表示されます: 構文エラーが 1 つしかない場合でも、複数のメッセージがコンパイラで生成されることがあります。最初のエラーメッセージは構文エラーによるものです。その他のエラーメッセージは、エラーのある部分以降のステートメントをコンパイラが解決できないためにトリガーされます。

ヒント: *スクリプトで複数の構文エラーが生成される場合、最初の 1 つを修正してから再コンパイルします。最初のエラーを修正することにより、その他のエラーの 1 つまたは複数を解決できることがあります。*

テーブル 18 ・ InstallScript 構文またはコンパイラ エラー

エラー番号	メッセージ
C8001	複数のメインプログラムが定義されています
C8002	関数名が必要です
C8003	プロトタイプ宣言をしていない関数です
C8004	ID がすでに宣言されています
C8005	関数が DLL 関数として定義されています
C8006	関数名の後に '(' がありません
C8007	コンマが必要です
C8008	ID が必要です

テーブル 18 • InstallScript 構文またはコンパイラ エラー (続き.)

エラー番号	メッセージ
C8009	パラメーターが多すぎます
C8010	右の括弧がありません
C8011	関数の最初に 'begin' がありません
C8012	セミコロンが必要です
C8013	予期しないソースファイルの終わりです
C8014	ID がすでに定義されています
C8015	既に定義されているメンバー名です
C8016	ラベルが定義されていません
C8017	typedef (struct)名が必要です
C8018	typedef オブジェクトはこのコンテキストでは規則違反です
C8019	タイプ宣言が必要です
C8020	パラメーターリストが必要です
C8021	'typedef' の後に 'begin' がありません
C8022	コンマまたはセミコロンが必要です
C8023	右の括弧が必要です
C8024	無効な配列 / 文字列のサイズです
C8025	未定義の ID
C8026	識別子の使用方法が無効です
C8027	オペランドがありません
C8028	オペレータが解読できません
C8031	関数が呼び出されません
C8032	メンバー参照がありません
C8033	実体のない配列です
C8034	式がありません

テーブル 18 • InstallScript 構文またはコンパイラ エラー (続き.)

エラー番号	メッセージ
C8035	ステートメントラベルが必要です
C8036	引数が多すぎます
C8037	変数が必要です
C8038	数値が必要です
C8039	ストリング値が要求されました
C8040	引数リストが不完全です
C8042	文字列あるいは配列が必要です
C8043	typedef ポインター型が必要です
C8044	typedef タイプが必要です
C8045	メンバー名が見つかりません
C8046	数値の変数が必要です
C8047	変数のアドレスのみを取ることができます
C8048	マクロ名がありません
C8049	#if/#elif に式がありません
C8050	#if/#elif の式が無効です
C8051	文字列リテラルの終わりがありません
C8052	拡張マクロのテキストが大きすぎます
C8053	識別子が #define マクロではありません
C8054	インクルードファイルの名前がありません
C8055	#elif の前に #if がありません
C8057	#else の前に #if がありません
C8058	#endif の前に #if がありません
C8059	プリプロセッサコマンドが認識されません
C8062	定数オペランドが必要です

テーブル 18 • InstallScript 構文またはコンパイラ エラー (続き.)

エラー番号	メッセージ
C8063	入力された行が長すぎます
C8064	コメントが終了されていません
C8065	文字列リテラルが 255 文字を超えています
C8066	ファイルの終わりに #endif ステートメントがありません
C8067	整数定数が長すぎます
C8068	認識されない文字が見つかりました
C8069	プリプロセッサコマンドは行の最初に置いてください
C8070	文字列定数が必要です
C8071	コロンが必要です
C8072	'elseif' を 'else' の後に置くことはできません
C8073	キーワード 'then' がありません
C8074	'else' 句は既に存在しています
C8075	'default' ラベルは既に存在しています
C8076	ステートメントに複数の case ラベルがあります
C8077	ラベルがすでに定義されています
C8078	ラベルが、この for ループで無効です
C8079	ステートメントが無効です
C8080	関数の引数がありません
C8081	代入結果が無効です
C8082	switch の後に '(' がありません
C8083	switch の後に ')' がありません
C8084	switch の後に 'case' がありません
C8085	'=' がありません
C8086	'to' または 'downto' がありません

テーブル 18 • InstallScript 構文またはコンパイラ エラー (続き.)

エラー番号	メッセージ
C8087	プログラムから値を返すことができません
C8088	if ステートメント内部にありません
C8089	for ステートメント内部にありません
C8090	repeat ステートメント内部にありません
C8091	while ステートメント内部にありません
C8092	関数が呼び出されましたが定義されていません
C8093	typedef 中の文字列のサイズが必要です
C8097	構文エラー
C8098	DLL 名の後に '.name' がありません
C8099	DLL 関数名が必要です。
C8100	この DLL には関数が定義されていません
C8101	この関数に DLL を指定する必要があります
C8112	typedef にそれ自体が含まれています
C8113	外部変数をローカル変数にすることはできません
C8114	プリプロセッサ ユーザー定義エラー
C8126	'テキスト': 文字列変数が必要です
C8127	try/catch/endcatch 内でラベルが不正です
C8128	try/catch/endcatch 内で goto が不正です
C8522	アクセスが拒否されました。

エラー C8001

メッセージ

'プログラム': 複数のメインプログラムが定義されています

説明

メインプログラムブロックの後にキーワードプログラムが検出されました。

トラブルシューティングのヒント

スクリプトはキーワードプログラムで始まり、キーワード endprogram で終わるメイン プログラム ブロックをひとつのみ含むことができます。メインプログラムブロックをひとつだけ利用するスクリプトに再構築します。

スクリプトに複数ソースファイルが含まれていて、これらのソースファイルの一つ以上がメインプログラムブロックを含む場合にはこのエラーメッセージが発生します。

エラー C8002

メッセージ

'テキスト': 関数名が必要です

説明

テキスト が指定した場所に、コンパイラが関数名を必要としています。

トラブルシューティング情報

パラメーターリストの前に不足している、あるいはスペルミスした関数名を確認します。

以前の行でセミコロンが抜けていないかどうかをチェックします。

エラー C8003

メッセージ

'名前' : プロトタイプ宣言をしていない関数です

説明

*名前*が指定した関数がプロトタイプ ステートメントで宣言されていません。

トラブルシューティング情報

関数が宣言されていない場合、現在のブロック (メインプログラムまたは関数定義) の前にプロトタイプステートメントを挿入します。

関数が現在のブロック以前に宣言されている場合、プロトタイプステートメントの関数宣言と関数定義の関数ヘッダーとを比較します。関数名のスペルがどちらも同じであること、そしてパラメーターの数と種類が一致することを確認してください。

関数がスクリプトの後の方で宣言される場合、宣言を現在のブロックの前に移動させます。

エラー C8004

メッセージ

'名前' : 識別子が既に宣言されています

説明

名前 で指定された識別子は、ブロック内で既に宣言されています。識別子は、同じブロックで一度だけ宣言することができます。

トラブルシューティング情報

エラーが発生したメインプログラムブロック、または関数ブロックを調べて、最初の識別子の宣言を見つけ出します。2 番目の宣言が単なる重複であった場合は、それを削除します。2 番目の宣言が最初の宣言とは別の変数である場合、2 番目の宣言の識別子名を変更し、それが参照するスクリプト内のすべてのステートメントをアップデートします。

以前の変数の宣言が見つからなかった場合、InstallScript で予約されている識別子名を宣言している可能性があります。予約語はスクリプト エディター内で色がついた構文です。

エラー C8005

メッセージ

名前' : 関数が DLL 関数として定義されています

説明

*名前*が指定した関数の関数定義が無効です。これは関数がプロトタイプ ステートメントで DLL 関数として宣言されているためです。

トラブルシューティング情報

関数が DLL 関数でない場合、DLL ファイル名を関数プロトタイプから削除してください。

ユーザー定義関数の名前が DLL 関数名と競合する場合、ユーザー定義関数の名前を変更してその名前を参照するスクリプト内のすべてのステートメントをアップデートします。ユーザー定義関数をプロトタイプステートメントで必ず宣言してください。

エラー C8006

メッセージ

'テキスト': 関数名の後に '(' がありません

説明

コンパイラが関数定義の関数名の後に開き丸カッコを必要とする場所に *テキスト* が示した文字 (単数または複数) がありました。

トラブルシューティング情報

関数のパラメーター リストが、かっこで括られているかを確認します。関数にパラメーターがない場合、括弧は空白となります。

関数名とそのパラメーターリストの間にはスペースのみが入っていることを確認します。

エラーC8007

メッセージ

'テキスト': コンマが必要です

説明

テキスト が指定した文字（単数または複数）が、コンマが必要な場所にありました。

トラブルシューティング情報

このエラーが関数宣言の中で発生した場合、パラメーター宣言を分割するために必要なコンマが不足しています。

このエラーが関数呼び出しの中で発生した場合、パラメーターを分割するために必要なコンマが不足しています。

エラー C8008

メッセージ

'テキスト': 識別子が必要です

説明

テキスト が指定した文字（単数または複数）が、識別子が必要な場所にありました。

トラブルシューティング情報

このエラーが変数宣言の中で発生した場合、データ型の後に識別子があるか、不要な句読点がないか、そして識別子が利用可能な文字のみを含んでいるかを確認します。

このエラーが関数呼び出しの中で発生した場合、各引数ペアがコンマひとつのみで区切られているかを確認します。

エラー C8009

メッセージ

'テキスト': パラメーターが多すぎます

説明

テキスト が指定した文字（単数または複数）が、関数に宣言された数以上のパラメーターを含む関数呼び出しパラメーターリストにありました。

トラブルシューティング情報

関数呼び出しに所属しない引数（単数または複数）を識別するには、関数呼び出しの引数の種類とその数と、その関数に宣言されたパラメーターの種類と数とを比較します。

エラー C8010

メッセージ

'テキスト': パラメーターが多すぎます

説明

テキスト が指定した文字 (1 つまたは複数) が、括弧の右側が必要な位置にあります。

トラブルシューティング情報

エラーは通常、関数定義内の関数ヘッダーにあるパラメーターリストで括弧の右側が足りない場合にトリガーされます。

エラー C8011

メッセージ

'end': 関数の最初に 'begin' がありません

説明

関数宣言にキーワード begin がありません。

トラブルシューティング情報

関数の最初のステートメントの前にキーワード begin を挿入してください。ローカル変数宣言は関数ヘッダーとキーワード begin の間に配置しなくてはなりません。関数ヘッダーそのものをセミコロンで終了させることはできません。

エラー C8012

メッセージ

'テキスト': セミコロンが必要です

説明

テキスト が指定した文字 (単数または複数) が、セミコロンが必要な場所にありました。

トラブルシューティング情報

セミコロンを挿入して、エラー場所の直前にあるステートメントを終了してください。

エラー C8013

メッセージ

予期しないソースファイルの終わりです

説明

現在のプログラムまたは関数ブロックを閉じるのに必要なキーワードが見つからないまま、コンパイラがソースファイルの終わりに到達しました。

トラブルシューティングのヒント

- スクリプトに関数定義が含まれる場合、各関数定義がキーワード end とセミコロンで終了されているか確認してください。
- typedef ブロックが end ステートメントで閉じられているか確認してください。
- フロー制御ブロックが正しく閉じられているか確認してください。
- #if ステートメント、#ifdef ステートメント、そして#ifndef ステートメントすべてに対応する #endif ステートメントがあることを確信してください。.

エラー C8014

メッセージ

'名前' : 識別子が既に定義されています

説明

name で指定された識別子は、構造内で既に宣言されています。

トラブルシューティング情報

スクリプトとエラー場所の前に含まれているスクリプト何れかを調査して、識別子の最初の宣言を見つけます。2 番目の宣言が単なる重複であった場合は、それを削除します。2 番目の宣言を最初の宣言とは別の識別子とする場合、2 番目の宣言で識別子名を変更し、それが参照するスクリプト内のすべてのステートメントをアップデートします。

エラー C8015

メッセージ

'名前' : メンバーが既に定義されています

説明

name で識別されたメンバーは、構造内で既に宣言されています。

トラブルシューティング情報

エラーが発生した構造を調べてメンバーの最初の宣言を見つけます。2 番目の宣言が単なる重複であった場合は、それを削除します。2 番目の宣言を最初の宣言とは別のメンバーとする場合、2 番目の宣言でメンバー名を変更し、それが参照するスクリプト内のすべてのステートメントをアップデートします。

エラー C8016

メッセージ

'名前' : 未定義のラベルです:

説明

name が指定した識別子が、定義されていないラベルを参照する goto ステートメントで利用されています。

トラブルシューティング情報

エラーはエラー場所の直ぐ上に見つかりました。名前を参照する goto ステートメントを検出します。goto ステートメントがプログラムブロック内にある場合、プログラムブロック内で定義されたラベルを参照しなくてはなりません。goto ステートメントが関数定義にある場合、その関数定義で定義されたラベルを参照しなくてはなりません。フロー制御を行うか、またその場所でラベルを定義するかを決定します。

エラー C8017

メッセージ

'テキスト': typedef (struct) 名が必要です

説明

テキスト が指定した文字 (単数または複数) が、typedef 宣言が必要な場所にありました。

トラブルシューティング情報

データ型が宣言された変数へ指定されたことを確認します。データ型の予約語のスペルが正しいこと、そしてすべて大文字であることを確認してください。

そのデータ型に複数の変数が宣言されている場合、各識別子名がその他の句読点ではなくコンマで分割されていることを確認してください。

データ宣言がセミコロンで終了されていることを確認します。

パラメーターとして構造へポインターを宣言する関数プロトタイプでエラーが発生した場合、関数プロトタイプの以前に構造が宣言されていることを確認します。

エラー C8018

メッセージ

'text'; typedef オブジェクトはこのコンテキストでは規則違反です

説明

text が指定する構造タイプが不正な場所にあります。

トラブルシューティング情報

このエラーは、構造タイプをパラメーターの一つとして宣言する関数プロトタイプ内で発生します。これは受け入れられません。その代わりに、パラメーターを構造へのポインターとして宣言します。パラメーターは次に示したように、キーワード POINTER を使った構造タイプの名前に続きます:

```
typedef RECT
 begin
    SHORT sX;
    SHORT sY;
 end;

 RECT Rectangle;
 RECT POINTER pRect;

 prototype SizeRectangle(RECT POINTER);
```

エラー C8019

メッセージ

'テキスト' : タイプ宣言が必要です

説明

テキスト が指定した文字（単数または複数）が、データ宣言が必要な場所にありました。

トラブルシューティング情報

一般的にこのエラーは、エラー場所の前にある行で、データ宣言の最後にセミコロンが不足している場合に発生します。また、関数ヘッダーがセミコロンで終了されている場合にも関数定義内で起こります。

エラー C8020

メッセージ

'テキスト': パラメーターリストが必要です

説明

テキスト が指定した文字（単数または複数）が、対応するパラメーターリストがないプロトタイプ宣言の直後にありました。

トラブルシューティング情報

エラー場所の上にある関数のパラメーターリストを指定します。関数がパラメーターを持たない場合、関数名の後に中身が空白の括弧を配置して空白リストを指定します。

エラー C8021

メッセージ

'テキスト' : 'typedef' の後に 'begin' がありません

説明

テキスト が指定した文字 (単数または複数) が、typedef ステートメント内のキーワード begin が必要な場所にありました。typedef ステートメントでは、キーワード begin の後に構造名が続かなくてはなりません。

トラブルシューティング情報

構造名の後、句読点を挟まずにキーワード begin を挿入します。

エラー C8022

メッセージ

'テキスト' : コンマまたはセミコロンが必要です

説明

テキスト が指定した文字 (単数または複数) が、コンマまたはセミコロンが必要な場所にありました。

トラブルシューティング情報

このエラーがデータ宣言の近くで発生した場合、識別子が有効であること、およびデータ宣言がセミコロンで終了していることを確認してください。複数の識別子が同じステートメント内で宣言されている場合、それらがコンマで区切られていることを確認します。

エラー C8023

メッセージ

'text': 右の括弧が必要です

説明

テキスト が指定した文字 (単数または複数) が、括弧の右側が必要な位置にあります。

トラブルシューティング情報

このエラーは、配列への参照で右側の閉じ括弧が不足している場合に発生します。

エラー C8024

メッセージ

'テキスト' : 無効な配列 / 文字列のサイズです

説明

テキスト が指定した文字 (単数または複数) は、文字列または配列の宣言には利用できません。

トラブルシューティング情報

サイズインジケーターが定数の場合、スペルが正しいことを確認してください。

配列のサイズが 0 よりも大きいことを確認してください。

エラー C8025

メッセージ

'テキスト' : 未定義の識別子

説明

*テキスト*によって指定された識別子が宣言されていません。

トラブルシューティング情報

スクリプト内のすべての識別子は、メインプログラムブロック内、または関数ブロック内で参照される前に宣言する必要があります。識別子が宣言されていることを確認してください。宣言されている場合、宣言で使われたスペルがエラー場所の識別子のスペルと一致するかを確認してください。

メモ: *スクリプトはサポートされていない関数を呼び出す可能性があります。詳細については、「サポートされていない関数」を参照してください。*

変換後のコンパイル時の " 'INSTALLDIR' : 未定義の識別子"

エラー C8025 は、識別子 (この場合 INSTALLDIR) が宣言されていないことを意味します。スクリプト内のすべての識別子は、メインプログラムブロック内、または関数ブロック内で参照される前に宣言する必要があります。

メモ: *エラー C8025 のエラーメッセージそのものは特定のエラーではありません。定義されていないものすべてに発行されます。*

(InstallScript MSI から InstallScript プロジェクトへの) 変換処理中に行われる変更の一部として、様々な IDE 要素にある INSTALLDIR が TARGETDIR に変更されます。エラー C8025 は TARGETDIR の値がスクリプトで設定されていない場合に発生します。

したがって、TARGETDIR の値は **OnFirstUIBefore** イベント ハンドラー関数の中でデフォルトで設定されるようにスクリプトで設定する必要があります。

次のコードを利用して TARGETDIR を設定することができます。

```
if ( ALLUSERS ) then
    TARGETDIR = PROGRAMFILES ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
else
    TARGETDIR = FOLDER_APPDATA ^ IFX_COMPANY_NAME ^ IFX_PRODUCT_NAME;
```

エラー C8026

メッセージ

'テキスト' : 識別子の使用方法が無効です

説明

ステートメント内で、*テキスト* が指定した識別子が不適切に利用されています。

トラブルシューティング情報

このエラーメッセージは、たとえば、演算の中で構造名の代わりに構造タイプ名を利用するなど、様々なエラーによってトリガーされます。エラー場所のステートメントを調べて、ステートメントにあるすべての演算子が適切に利用されていることを確認してください。

エラー C8027

メッセージ

オペランドがありません

説明

コンパイラは演算子の後に必要なオペランドを検出できませんでした。

トラブルシューティング情報

このエラーは、ソースファイルが演算子で終了されている不完全なステートメントにコンパイラが遭遇した場合に表示される場合があります。

エラー C8028

メッセージ

'テキスト' : 未解決の演算子

説明

テキスト が指定した文字（単数または複数）が、解決できない式の後にあります。

トラブルシューティング情報

このエラーは、式にオペランドが足りない場合に発生します。式にある各演算子を調べて、各演算子が必要なオペランドを含んでいるとを確認してください。

エラー C8031

メッセージ

'テキスト' : 関数が呼び出されません

説明

テキスト が指定する文字 (単数または複数) が、関数名の後にありました。ここには引数リストの開き括弧が必要です。

このエラーは、関数呼び出しの結果が変数に割り当てられるステートメントで発生します。メッセージは関数名に続く引数リストの中のエラーによってトリガーされます。

トラブルシューティング情報

開き括弧を使って、引数リストが関数名と区切られていることを確認してください。

引数リストが指定されていない場合、関数名の後に挿入します。パラメーターを必要としない関数を呼び出すときは、関数名の後に中が空白の括弧を配置します。

エラー C8032

メッセージ

'テキスト' : メンバー参照がありません

説明

テキスト が指定した文字 (単数または複数) が、メンバー名が必要な場所にありました。

トラブルシューティング情報

エラー場所にあるコード行は、構造のメンバーではなく構造を参照します。どの構造メンバーを参照し、そのメンバー名を挿入するのかを決定します。必ずメンバー演算子を使って、構造名とメンバー名を区切ってください。

エラー C8033

メッセージ

'テキスト' : 実体のない配列です

説明

テキスト が指定した文字 (単数または複数) の前にある識別子は、添え字無し配列変数です。InstallScript では完全な配列構造への参照を受け付けません。各配列の要素を参照する必要があります。

トラブルシューティング情報

どの配列要素を参照するのか決定し、添え字を使って配列内の位置を指示します。添え字は角括弧内、配列名の直後に配置しなくてはなりません。

エラー C8034

メッセージ

'テキスト' : 式がありません

説明

テキスト が指定した文字 (単数または複数) が、式が必要な場所にありました。

トラブルシューティング情報

このエラーは一般的には、代入する値が不足している場合に代入ステートメントで発生します代入演算子の右側へ式を挿入します。

定数を参照するステートメントでエラーが発生した場合、その定数の定義に定数名と値の両方が含まれていることを確認してください。

エラー C8035

メッセージ

'テキスト' : ステートメントラベルが必要です

説明

テキスト が指定した文字 (単数または複数) が、ラベルが必要な場所であるキーワード goto の後にありました。

トラブルシューティング情報

キーワード goto の後にラベルを追加する、または goto ステートメントを削除してください。

エラー C8036

メッセージ

'名前' : 引数が多すぎます

説明

name が指定した関数が、多くの引数で呼び出されました。

トラブルシューティング情報

関数呼び出しの引数と、その関数の宣言にあるパラメーターリストと比較してください。宣言で定義されていない関数呼び出しの引数 (単数または複数) を削除してください。

エラー C8037

メッセージ

'テキスト' : 変数が必要です

説明

テキスト が指定した定数、リテラル値、または構造メンバーが、変数を必要とするパラメーター位置で引数として渡されました。

トラブルシューティング情報

パラメーターに必要な種類の変数を宣言し、その変数に定数またはリテラル値を代入します。そして、定数またはリテラル値の代わりに変数を渡します。

InstallShield では参照を使って構造メンバーを渡すことはできません。同じ種類の変数を構造メンバーとして宣言し、新しい変数へ構造メンバーの値を代入してから、変数を代わりに渡します。関数が戻ったとき、構造メンバーへ変数の値を代入します。

エラー C8038

メッセージ

'テキスト' : 数値が必要です

説明

テキスト が指定した数値以外の引数が、数値引数を必要とするパラメーター位置に渡されました。

トラブルシューティング情報

渡される引数すべてが、該当するパラメーター位置に適切なデータ型であることを確認してください。また、必要な引数がすべて含まれていることを確認してください。

エラー C8039

メッセージ

'値' : 文字列値が必要です

説明

値 が指定した番号が、文字列データを必要とする式、ステートメント、あるいは関数引数にありました。

トラブルシューティング情報

文字演算で、すべてのオペランドと演算子が文字列データと互換性を持っていることを確認してください。

代入ステートメントで、文字列変数へ割り当てる値がそれ自体文字列であることを確認してください。

関数呼び出しで、文字列を必要とするパラメーター位置に文字列以外のデータを渡していないことを確認してください。

エラー C8040

メッセージ

'名前' : 引数リストが不完全です

説明

name が指定した関数への呼び出しの中で、1 つまたは複数の引数が不足しています。

トラブルシューティング情報

不足している引数を認識するためには、関数呼び出し内の引数の種類とその数と、関数へのパラメーターの種類とその数とを比較してください。

すべての引数が存在する場合、それらがコンマで区切られているかを確認してください。

エラー C8042

メッセージ

'[' :文字列あるいは配列が必要です

説明

角括弧の左側にある識別子は文字列または配列タイプでなくてはなりません。

トラブルシューティング情報

このエラーは、文字列または配列以外の変数のインデックス値を指定するときに発生します。角括弧の左側にある識別子を調べてください。それが文字列または配列ではない場合、インデックス値を削除するか、または変数を文字列または配列として宣言します。

エラー C8043

メッセージ

'テキスト' : typedef インター型が必要です

説明

テキスト が指定する文字 (単数または複数) は、構造へのポインターとのみ利用できます。

トラブルシューティング情報

このエラーの一般的な原因として、構造名とメンバー名との間にある構造ポインター演算子の誤った利用が挙げられます。構造ポインター演算子は、構造へのポインターとともに参照されるレコードのメンバーを指定する場合のみ利用することができます。構造ポインター演算子をメンバー演算子と入れ替えてください。

エラー C8044

メッセージ

'テキスト' : typedef 型が必要です

説明

テキスト が指定した文字（単数または複数）は、typedef ステートメント内で宣言された構造とのみ利用することができます。

トラブルシューティング情報

このエラーの一般的な原因として、構造へのポインターとメンバー名との間に利用しているメンバー演算子の誤った利用が挙げられます。メンバー演算子は構造名とともに参照される構造のメンバーを指定する場合のみ利用することができます。メンバー演算子を構造ポインター演算子と入れ替えてください。

エラー C8045

メッセージ

'名前' : メンバー名が見つかりません

説明

name が指定したメンバーが、現在参照している構造に存在しません。

トラブルシューティング情報

指定したメンバーの構造定数を確認してください。

メンバー名のスペルを確認してください。

エラー C8046

メッセージ

数値の変数が必要です

説明

数値変数を必要とするパラメーター位置に、数値以外の引数が渡されました。

トラブルシューティング情報

引数を数値変数に差し替えてください。

エラー C8047

メッセージ

'&' : 変数のアドレスのみを取ることができます

説明

コンパイラは定数またはリテラル上でアドレス演算子を検出しました。

トラブルシューティング情報

アドレス演算子 (&) は、変数とのみ利用できます。

エラー C8048

メッセージ

マクロ名がありません

説明

エラー場所の #define ステートメントにマクロ名がありません。

トラブルシューティング情報

#define プリプロセッサ命令の後、マクロ値の前にマクロ名を挿入してください。

エラー C8049

メッセージ

#if/#elif に式がありません

説明

#if または #elif プリプロセッサ命令の後に、コンパイルのフローを決定する有効な式がありません。

トラブルシューティング情報

このエラーメッセージは、#if または #elif キーワードの後で式が指定されなかった場合にトリガーされます。これらの命令のいづれかの後に無効な式が続いた場合にもトリガーされます。

エラー C8050

メッセージ

'text' : #if/#elif の式が無効です

説明

テキスト が指定した文字 (単数または複数) は #if または #elif プリプロセッサコマンドに続く式の中では無効です。

トラブルシューティング情報

このエラーメッセージは、#if または #elif キーワードの後の式が無効の場合にトリガーされます。

エラー C8051

メッセージ

文字列リテラルの終わりがありません

説明

#define ステートメントにある文字列リテラルに終わりの引用符がありません。

トラブルシューティング情報

エラーが検出された行の上にある #define ステートメントを探します。文字列定数は、リテラルの初めと終わりを引用符で括ってください。数値定数は引用符で括らないでください。

エラー C8052

メッセージ

拡張マクロのテキストが大きすぎます

説明

#define ステートメントの識別子用に指定したテキストが 256 文字を超えています。

トラブルシューティング情報

その識別子に指定した文字列を短くするか、テキストを複数の短い文字列に分割してから、それぞれを #define ステートメントで指定してください。その後プログラムで文字列を連結します。

エラー C8053

メッセージ

'名前' : 識別子が #define マクロではありません

説明

#undef ステートメントで利用され、*name* が指定した識別子が定義されていません。

トラブルシューティング情報

名前 が指定した識別子のスペルを確認してください。

エラー C8054

メッセージ

インクルードファイルの名前がありません

説明

プリプロセッサ命令 #include が、含めるファイルの名前を指定していません。

トラブルシューティング情報

含めるソースファイルの名前を指定する、またはプリプロセッサ命令を削除してください。

エラー C8055

メッセージ

#elif の前に #if がありません

説明

コンパイラは、#if 命令がない #elif 命令を検出しました。

トラブルシューティング情報

エラー場所の前にあるステートメントを調べて、#if 命令で始まる条件コンパイル命令を構築してください。

エラー C8057

メッセージ

#else の前に #if がありません

説明

コンパイラは、#if 命令がない #else 命令を検出しました。

トラブルシューティング情報

エラー場所の前にあるステートメントを調べて、#if 命令で始まる条件コンパイル命令を構築してください。

エラー C8058

メッセージ

#endif の前に #if がありません

説明

コンパイラは、#if 命令がない #endif 命令を検出しました。

トラブルシューティング情報

エラー場所の前にあるステートメントを調べて、#if 命令で始まる条件コンパイル命令を構築してください。

エラー C8059

メッセージ

'テキスト' : プリプロセッサ コマンドが認識されません

説明

テキスト が指定した文字 (単数または複数) で、記号 # に続くものは、プリプロセッサ コマンドとして認識されません。

トラブルシューティング情報

プリプロセッサコマンドが InstallShield スクリプトで有効であり、スペルが正しいことを確認してください。

エラー C8062

メッセージ

'テキスト' : 定数オペランドが必要です

説明

テキスト が指定した文字 (単数または複数) が、定数が必要なステートメント位置にありました。

トラブルシューティング情報

このエラーメッセージは、キーワード case の後に定数またはリテラルが続かない場合に switch...endswitch ブロックでトリガーされます。

エラー C8063

メッセージ

入力された行が長すぎます

説明

ソースコード行が 1,024 文字を超えています。

トラブルシューティング情報

行を複数に分割してください。

エラー C8064

メッセージ

コメントが終了されていません

説明

エラー場所で始まるコメントが終了されていません。

トラブルシューティング情報

コメントの終了場所を探してください。コメント終了文字を挿入するか、またはコメント終了文字が誤って入力されている場合は訂正してください。

エラー C8065

メッセージ

文字列リテラルが 255 文字を超えています

説明

エラー場所の文字列リテラル は 255文字を超えています。

トラブルシューティング情報

文字列リテラルを複数文字列リテラルに分割し、連結させてください。

エラー C8066

メッセージ

ファイルの終わりに #endif ステートメントがありません

説明

スクリプトで使われる #if 命令に、対応する #endif 命令がありません。

トラブルシューティング情報

スクリプトのプリプロセッサ命令を調べてください。すべての #if 命令に、対応する #endif があることを確認してください。

エラー C8067

メッセージ

'数値' : 整数定数が長すぎます

説明

数値 が指定した値が数値変数には長すぎます

トラブルシューティング情報

NUMBER データに有効な範囲は −2,147,483,648 から +2,147,483,647 です。値を +2,147,483,647 から +4,294,967,295 の範囲に指定すると、その値は超過して負の数値結果となります。値が 4,294,967,299 よりも大きい場合、このエラーメッセージが表示されます。

エラー C8068

メッセージ

'文字' : 認識されない文字が見つかりました

説明

char が指定した文字は、コンパイラが評価できませんでした。

トラブルシューティング情報

たとえば “$” や “{,” など、特定の文字はコンパイラによって認識されません。

エラー C8069

メッセージ

'#' : プリプロセッサコマンドは行の最初に置いてください

説明

シンボル # で認識されるプリプロセッサ命令が、不適切な位置にあります。プリプロセッサ命令は、それが含まれる行で最初のキーワードでなくてはなりません。

トラブルシューティング情報

シンボル # はプリプロセッサ命令を認識するために予約されています。InstallScript ではその他の有効な利用法はありません。エラーを生成した行を調査してください。識別子の中に シンボル # がある場合、その識別子とそれが参照するスクリプトにあるすべての名前を変更してください。

エラー C8070

メッセージ

'テキスト' : 文字列定数が必要です

説明

テキスト が指定した文字 (単数または複数) が、文字列定数が必要なステートメント位置にありました。

トラブルシューティング情報

このエラーメッセージは、キーワード case の後に文字列定数が続かない場合に switch...endswitch ブロックでトリガーされます。switch...endswitch ブロックでは、case ステートメントで指定された定数は switch ステートメントで指定された変数、または式結果のデータ型と同じでなくてはなりません。

エラー C8071

メッセージ

コロンが必要です

説明

コンパイラはステートメント内にコロンを必要とします。

トラブルシューティング情報

このエラーは、キーワード case の後の定数に続いてコロンがない場合に switch ステートメントで発生します。その case の定数とステートメントの間にコロンを挿入してください。

エラー C8072

メッセージ

'elseif' : 'elseif' を 'else' の後に置くことはできません

説明

キーワード elseif が、if...endif ブロックの else ステートメントの後に検出されました。

トラブルシューティング情報

キーワード elseif は、else ステートメントの後に if...endif ブロックで利用することはできません。if...endif を再構築してこのエラーを回避してください。

エラー C8073

メッセージ

'テキスト' : キーワード 'then' がありません

説明

テキスト が指定した文字 (単数または複数) が、キーワード then が必要な if ブロックで検出されました。

トラブルシューティング情報

エラー場所の直前にある if ステートメントを探してください。キーワード if に続く条件式の後にキーワード then を挿入してください。

エラー C8074

メッセージ

'else' : 'else' 句は既に存在しています

説明

if...endif ブロックで 2 つの else ステートメントが検出されました。

トラブルシューティング情報

if...endif ブロックは else ブランチを 1 つだけ含むことができます。複数条件ブランチを必要とする場合、elseif ステートメントを含む if...endif ブロック、または switch...endswitch ブロックを代わりに利用してください。

エラー C8075

メッセージ

'テキスト' : 'default' ラベルは既に存在しています

説明

switch...endswitch ステートメントで 2 番目のデフォルトの case が検出されました。

トラブルシューティング情報

switch...endswitch ステートメントはデフォルトの case を指定するキーワードのデフォルトのインスタンスをひとつのみ含むことができます。switch....endswitc ステートメントを再構築して、デフォルトの case のひとつを削除します。

InstallShield のバージョンの中には、switch ブロックで複数 case ラベルが検出された時にこのエラーメッセージが表示されるものもあります。

エラー C8076

メッセージ

'case' : ステートメントに複数の case ラベルがあります

説明

コンパイラは、ステートメントまたは最後に定義した case を実行するためのステートメントの場所にキーワード case を検出しました。

トラブルシューティング情報

文字、あるいは *テキスト* が指定した文字の直前にある case ステートメントが不完全です。終了セミコロンが不足している可能性があります。

InstallShield のバージョンの中には、switch ブロックで複数のデフォルトのステートメントが検出された時にこのエラーメッセージが表示されるものもあります。

エラー C8077

メッセージ

'名前' : ラベルが既に定義されています

説明

名前 が示すラベルは、メインプログラムブロックまたは関数で既に使用されています。

トラブルシューティング情報

プログラムブロック、または関数ブロック内の各ラベルは固有でなくてはなりません。

エラー C8078

メッセージ

ラベルが、この for ループで無効です

説明

for ステートメント内でラベルを定義することはできません。

トラブルシューティングのヒント

エラー場所にあるステートメントを調査します。ラベルが for ステートメント内で定義されないように、コードを変更します。

エラー C8079

メッセージ

'テキスト' : ステートメントが無効です

説明

テキスト が示した場所にあるステートメントは、現在の内容では受け付けられない、または無効なプロセスを指定します。

トラブルシューティングのヒント

エラー場所にあるステートメントを調査します。プロセスが有効であることを確認します。無効ステートメントの例は、代入演算子の左側にある定数または算術演算子を使った代入ステートメント、そしてラベルの真下にある end ステートメントです。

エラー場所にあるステートメントが有効な場合、エラー場所の上部にあるステートメントブロック (レコード宣言、コントロールステートメント、および関数定義など) を調べます。各ステートメントブロックが正しいキーワードで終了されているか確認します。

エラー C8080

メッセージ

'テキスト' : 関数の引数がありません

説明

テキスト が指定した文字 (単数または複数) が、引数リストが必要な関数呼び出しの後に検出されました。

トラブルシューティング情報

関数にはどの引数が必要なのかを決定し、括弧内に完全な引数リストを含むよう、呼び出しを修正してください。

エラー C8081

メッセージ

'テキスト' : 代入結果が無効です

説明

テキスト が指定した文字 (単数または複数) は、代入ステートメントが指定した値を代入することはできません。

トラブルシューティング情報

このエラーは値を構造へ代入した場合に発生します。InstallScript は構造への直接代入を受け付けません。各構造のメンバーへそれぞれ値を代入しなくてはなりません。

エラー C8082

メッセージ

'テキスト' : switch の後に '(' がありません

説明

テキスト が指定した文字 (単数または複数) が、switch...endswitch ブロック内の左側の括弧が必要な位置で検出されました。

トラブルシューティング情報

このエラーが発生した switch ステートメントを調査してください。キーワード switch に続く式は括弧で括らなくてはなりません。

エラー C8083

メッセージ

'テキスト': switch の後に ')' がありません

説明

テキスト が指定した文字 (単数または複数) が、switch...endswitch ブロック内の右側の括弧が必要な位置で検出されました。

トラブルシューティング情報

このエラーが発生した switch ステートメントを調査してください。キーワード switch に続く式は括弧で括らなくてはなりません。

エラー C8084

メッセージ

'テキスト' : switch の後に 'case' がありません

説明

テキスト が指定した文字 (単数または複数) が、case ステートメントが必要な switch...endswitch ブロックで検出されました。

トラブルシューティング情報

このエラーは switch ステートメントがセミコロンで終了している場合、または case ステートメントが直後に続かない場合に発生します。

エラー C8085

メッセージ

'text' : '=' がありません

説明

テキスト が指定した文字（単数または複数）が、for...endfor ブロックの等号が必要な場所に検出されました。

トラブルシューティング情報

このエラーは、キーワード for に続く式に等号が不足する場合に発生します。エラーが発生した for ステートメントを調べ、キーワード for に続く式の構造が正しいことを確認してください。

エラー C8086

メッセージ

'テキスト' : 'to' または 'downto' がありません

説明

テキスト が指定した文字（単数または複数）が、for...endfor ブロックの等号が必要な場所に検出されました。

トラブルシューティング情報

このエラーはキーワード to または downto が for ステートメントで不足している場合に発生します。

エラー C8087

メッセージ

プログラムから値を返すことができません

説明

メインプログラムブロックには return ステートメントが含まれます。プログラムから値を戻すことはできません。

トラブルシューティング情報

return ステートメントをプログラムブロックから削除してください。

エラー C8088

メッセージ

'endif' : if ステートメント内部にありません

説明

キーワード endif が、その前の if ステートメントと一致しません。

トラブルシューティング情報

エラー場所にある if...endif ブロックを調査します。キーワード endif に対応する if ステートメントが存在するかを確認してください。このエラーは、以前に if...endif ブロックでエラーがあった時にトリガーされることがしばしばあります。

エラー C8089

メッセージ

'endfor' : for ステートメント内部にありません

説明

キーワード endfor が、その前の for ステートメントと一致しません。

トラブルシューティング情報

エラー場所にある for...endfor ブロックを調査します。キーワード endfor に対応する for ステートメントが存在するかを確認してください。このエラーは、以前に for...endfor ブロックでエラーがあった時にトリガーされることがしばしばあります。

エラー C8090

メッセージ

'until' : repeat ステートメント内部にありません

説明

キーワード until が、その前の repeat ステートメントと一致しません。

トラブルシューティング情報

エラー場所にある repeat...until ブロックを調査します。キーワード until に対応する repeat ステートメントが存在するかを確認してください。このエラーは、以前に repeat...until ブロックでエラーがあった時にトリガーされることがしばしばあります。

エラー C8091

メッセージ

'endwhile' : while ステートメント内部にありません

説明

キーワード endwhile が、その前の while ステートメントと一致しません。

トラブルシューティング情報

エラー場所にある while...endwhile ブロックを調査します。キーワード endwhile に対応する while ステートメントが存在するかを確認してください。このエラーは、以前に while...endwhile ブロックでエラーがあった時にトリガーされることがしばしばあります。

エラー C8092

メッセージ

'名前' : 関数が呼び出されましたが定義されていません

説明

名前 が指定する関数はスクリプト本体でプロトタイプ化され呼び出されましたが、定義されていません。

トラブルシューティング情報

このエラーは、関数名がプロトタイプ、関数呼び出し、並びに関数定義のスペルが同一でないときに発生します。また、関数定義が (#include ステートメントと共に) 含まれてるべきファイル内に存在するように見えて、実際は含まれていない場合にも発生します。

エラー C8093

メッセージ

'名前' : typedef 中の文字列のサイズが必要です

説明

*名前*が指定する識別子は typedef ブロックでメンバーとして定義された文字列ですが、サイズ指定は含まれていません。

トラブルシューティング情報

InstallScript のオートサイズ機能は typedef ステートメントでは動作しないので、構造内ですべての STRING 宣言のサイズを指定してください。

エラー C8097

メッセージ

'テキスト' : 構文エラー

説明

テキスト が指定した文字（単数または複数）が、区分できない構文エラーをトリガーしました。

トラブルシューティング情報

このエラーは構文エラーでトリガーされますが、これには特定のエラーメッセージがありません。

エラー C8098

メッセージ

'テキスト' : DLL 名の後に '.name' がありません

説明

テキスト が指定した文字（単数または複数）は、DLL ファイルの関数が必要な DLL ファイル名の後に検出されました。

トラブルシューティング情報

DLL から関数を呼び出すとき、DLL ファイル名ではなく関数名を指定してください。

エラー C8099

メッセージ

'テキスト' : DLL 関数名が必要です。

説明

テキスト が指定した文字（単数または複数）は、DLL ファイルの関数が必要な DLL ファイル名の後に検出されました。

トラブルシューティング情報

このエラーは、プロトタイプステートメントでファイル名が宣言されていない DLL ファイルを使って関数名が指定されたときに発生します。この関数がプロトタイプ化されていることを確認して下さい。もしそうであれば、関数名のスペルを確認してください。

エラー C8100

メッセージ

'名前' : この DLL には関数が定義されていません

説明

名前 が指定する関数は、その前にある名前の DLL 用にプロトタイプ化されていません。

トラブルシューティング情報

このエラーは複数の DLL から関数をプロトタイプ化し、その中の関数を間違った DLL名で呼び出したときに発生します。

エラー C8101

メッセージ

'名前' : この関数に DLL を指定する必要があります

説明

名前 が指定した関数は、2 つの異なる DLL から 2 回プロトタイプ化されています。

トラブルシューティング情報

複数の DLL から同じ名前を使って関数を呼び出すとき、次の例の様に、DLL 名を接頭辞として関数名を修飾し、その後にピリオドを続けます:

```
    prototype MYDLL.UsefulFunction(INT, POINTER);
    prototype YOURDLL.UsefulFunction(INT, POINTER);

    export prototype MyFunc(HWND);

function MyFunc(hMSI)
    NUMBER nValue;
    POINTER psvString;
begin
    MYDLL.UsefulFunction(nValue, psvString);
end;
```

エラー C8112

メッセージ

'名前' : typedef にそれ自体が含まれています

説明

名前 が指定したメンバーのデータ型は無効です。typedef にはそのもののインスタンスであるメンバーを含むことはできません。

トラブルシューティング情報

typedefs を再構築してこのエラーを回避してください。

エラー C8113

メッセージ

'external' : 外部変数をローカル変数にすることはできません

説明

キーワード external が、関数ブロック内のデータ宣言で検出されました。

トラブルシューティング情報

データは関数にローカルで、外部宣言をすることはできません。

エラー C8114

メッセージ

'テキスト' :プリプロセッサ ユーザー定義エラー

説明

このメッセージは #error 命令によってトリガーされます。

エラー C8115

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

メッセージ

'関数名': プロトタイプによるリターン タイプの不一致

説明

このメッセージは、スクリプト定義関数(関数によって戻される値のデータ型)が、関数定義と関数宣言で同じでない場合に表示されます。きちんと指定されない戻り値の型は、NUMBER と認識されてしまうことに注意します。

エラー C8126

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

メッセージ

'テキスト': 文字列変数が必要です

説明

text が指定した文字（単数または複数）が、文字列宣言が必要な場所にありました。

トラブルシューティング情報

文字配列（つまり、文字列）以外の配列 を宣言するには、角括弧ではなく丸括弧を使用します。たとえば、次のとおりです:

```
NUMBER nArray(3);
```

エラー C8127

メッセージ

try/catch/endcatch 内でラベルが不正です

説明

try...catch...endcatch ステートメント内でラベルを定義することはできません。

トラブルシューティングのヒント

エラー場所にあるステートメントを調査します。ラベルが try...catch...endcatch ステートメント内で定義されないように、コードを変更します。

エラー C8128

メッセージ

try/catch/endcatch 内で goto が不正です

説明

try...catch...endcatch ステートメント内で goto ステートメントを使用することはできません。

トラブルシューティングのヒント

エラー場所にあるステートメントを調査します。ラベルが try...catch...endcatch ステートメント内で goto ステートメントが使用されないように、コードを変更します。

エラー C8522

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

メッセージ

アクセスが拒否されました。

説明

InstallShield は、指定されたファイルを開くことができませんでした。

トラブルシューティング情報

このエラーメッセージは、指定されたファイルが読み取り専用であることが原因で起こる場合があります。Windows エクスプローラーでファイルを見つけ、そのプロパティを確認してください。

InstallScript の致命的エラー

致命的なエラーは、コンパイルを続行できなくするエラーです。これらのエラーは、インクルードファイルの欠如、ファイル名の不正指定、ディスクの I/O エラー、システムリソース不足などが原因です。また、コンパイラには、インクルードファイルの制限などのビルトイン制限があり、これを超えると致命的エラーが発生します。

テーブル 19 • InstallScript の致命的エラー

エラー	メッセージ
F8501	アクション ファイルの I/O エラー
F8502	デバッグ ファイルの I/O エラー
F8503	スクリプト入力ファイルを開けません
F8504	.inx 出力ファイルを開くことができません
F8505	.dbg 出力ファイルを開けません
F8506	メモリ不足
F8508	行内のマクロ拡張が多すぎます
F8509	インクルードファイル名の区切り文字が無効です
F8510	インクルードファイル名の区切り文字がありません
F8511	インクルードファイルを開けません。
F8512	インクルードファイルとパスが大きすぎます
F8513	ネストされた #include ファイルが多すぎます
F8514	#if ステートメントのネストの数が多すぎます

テーブル 19 • InstallScript の致命的エラー (続き.)

エラー	メッセージ
F8515	マクロ拡張バッファーがオーバーフローしています
F8516	最大エラー数に達しました
F8517	インクルードファイルが多すぎます
F8518	リンク ファイルの I/O エラー
F8519	ユーザー定義の致命的エラー

エラー F8501

メッセージ

アクション ファイルの I/O エラー

説明

InstallShield がコンパイル中に作成された中間ファイルにアクセスしようとした際にエラーが発生しました。

トラブルシューティング情報

セットアップをコンパイルする度に、InstallShield はファイル拡張子 .dbg、.ino、.inx、および .obs をもつ中間ファイルを作成します。コンパイルの後、これらのファイルはスクリプトフォルダーに残ります。このメッセージはエラーがアクションファイルへのアクセスの際に発生したときに表示されます。このファイルの拡張子は .obs です。可能性のある原因は:

- ディスクがいっぱいです。不要なファイルを削除して有効ディスクスペースを増やしてから、再度コンパイルしてください。
- システムリソース (ファイルハンドラー並びに/またはメモリ) が不足しています。開いているその他のアプリケーションを閉じて、再度コンパイルしてください。ディスク容量またはシステムリソースが充分にあるシステムでエラーが発生した場合、ファイルアクセス失敗の他の考えられる理由を追及してください:
  - アクションファイルが破損している可能性があります。スクリプトフォルダーから削除または名前の変更を行った後、再度コンパイルしてください。
  - アクションファイルがファイルサーバーの共有フォルダーにある場合、そのフォルダーを上書きすることができるネットワーク権限を持っているかどうか確認してください。
  - 以前のコンパイルのアクションファイルが存在する場合、そのファイル属性が読み取り専用、システム、または非表示に変更されていないか確認してください。ファイルが別のアプリケーション又はユーザーによって開かれていないかも確認してください。

エラー F8502

メッセージ

デバッグ ファイルの I/O エラー

説明

InstallShield がコンパイル中に作成された中間ファイルにアクセスしようとした際にエラーが発生しました。

トラブルシューティング情報

セットアップをコンパイルする度に、InstallShield はファイル拡張子 .dbg、.ino、.inx、および .obs をもつ中間ファイルを作成します。コンパイルの後、これらのファイルはスクリプトフォルダーに残ります。このメッセージはエラーがデバッグファイルへのアクセスの際に発生したときに表示されます。このファイルの拡張子は .dbg です。可能性のある原因は:

- ディスクがいっぱいです。不要なファイルを削除して有効ディスクスペースを増やしてから、再度コンパイルしてください。
- システムリソース (ファイルハンドラー並びに/またはメモリ) が不足しています。開いているその他のアプリケーションを閉じて、再度コンパイルしてください。ディスク容量またはシステムリソースが充分にあるシステムでエラーが発生した場合、ファイルアクセス失敗の他の考えられる理由を追及してください:
  - デバッグファイルが破損している可能性があります。スクリプトフォルダーから削除または名前の変更を行った後、再度コンパイルしてください。
  - デバッグファイルがファイルサーバーの共有フォルダーにある場合、そのフォルダーを上書きすることができるネットワーク権限を持っているかどうか確認してください。
  - 以前のコンパイルの .dbg ファイルが存在する場合、そのファイル属性が読み取り専用、システム、または非表示に変更されていないか確認してください。ファイルが別のアプリケーション又はユーザーによって開かれていないかも確認してください。

エラー F8503

メッセージ

スクリプト入力ファイルを開けません

説明

コンパイラは指定されたスクリプト ファイルを開くことができませんでした。

トラブルシューティング情報

ファイルの場所と名前を確認してください。ファイル名のスペルを間違えたか、不正確なパスを指定した可能性があります。

エラー F8504

メッセージ

.inx 出力ファイルを開けません

説明

コンパイラはコンパイル済みスクリプト用のファイルを作成することができませんでした。

トラブルシューティング情報

このエラーは、InstallShield が .inx ファイルを作成、または再作成できなかったときに生成されます。通常、このエラーは次の問題が原因で発生します:

- ディスクがいっぱいです。不要なファイルを削除して有効ディスクスペースを増やしてから、再度コンパイルしてください。
- システムリソース (ファイルハンドラー並びに/またはメモリ) が不足しています。開いているその他のアプリケーションを閉じて、再度コンパイルしてください。ディスク容量またはシステムリソースが充分にあるシステムでエラーが発生した場合、ファイルアクセス失敗の他の考えられる理由を追及してください:
  - ターゲットファイルがファイルサーバーの共有フォルダーにある場合、そのフォルダーを上書きすることができるネットワーク権限を持っているかどうか確認してください。
  - 以前のコンパイルの .inx ファイルが存在する場合、そのファイル属性が読み取り専用、システム、または非表示に変更されていないか確認してください。ファイルが別のアプリケーション又はユーザーによって開かれていないかも確認してください。

エラー F8505

メッセージ

.dbg 出力ファイルを開けません

説明

コンパイラはデバッグファイルを作成することができませんでした。このファイルにはデバッガーを使ってスクリプトを実行するのに必要な情報が含まれています。

トラブルシューティング情報

このエラーは、次の問題が原因で発生します:

- ディスクがいっぱいです。不要なファイルを削除して有効ディスクスペースを増やしてから、再度コンパイルしてください。
- システムリソース (ファイルハンドラー並びに/またはメモリ) が不足しています。開いているその他のアプリケーションを閉じて、再度コンパイルしてください。

ディスク容量またはシステムリソースが充分にあるシステムでエラーが発生した場合、ファイルアクセス失敗の他の考えられる理由を追及してください:

- デバッグファイルがファイルサーバーの共有フォルダーにある場合、そのフォルダーを上書きすることができるネットワーク権限を持っているかどうか確認してください。
- 以前のコンパイルの .dbg ファイルが存在する場合、そのファイル属性が読み取り専用、システム、または非表示に変更されていないか確認してください。ファイルが別のアプリケーション又はユーザーによって開かれていないかも確認してください。

エラー F8506

メッセージ

メモリ不足です

説明

コンパイラは、コンパイルを完了するために充分なメモリを割り当てることができませんでした。

トラブルシューティング情報

開いているその他のアプリケーションを閉じて、再度コンパイルしてください。問題が解決しない場合、Windows を再起動してから InstallShield をロードして再度コンパイルを行ってください。

エラー F8508

メッセージ

行内のマクロ拡張が多すぎます

説明

指定した行のマクロの数が多すぎます。一行あたりの最大マクロ数は 100 です。

トラブルシューティング情報

行を複数に分割してマクロの数を減らしてください。

エラー F8509

メッセージ

インクルードファイル名の区切り文字が無効です

説明

プリプロセッサ コマンド #include の後に指定されたファイル名に、始めの引用符、または山括弧（<) が不足しています。

トラブルシューティング情報

プリプロセッサ命令 #include の後に指定されたファイル名は、引用符 (“ファイル名”) または山括弧 (<ファイル名>) で括ります。ファイル名の前に引用符または山かっこを挿入してください。始まりの引用符を挿入した場合、ファイル名の後には終わりの引用符が続かなくてはなりません。 始まりの山かっこを挿入した場合、ファイル名の後には終わりの山かっこ (>) が続かなくてはなりません。

エラー F8510

メッセージ

インクルードファイル名の区切り文字がありません

説明

プリプロセッサ命令 #include の後に指定されたファイル名に、終わりの引用符、または山括弧 (<) が不足しています。

トラブルシューティング情報

プリプロセッサ命令 #include の後に指定されたファイル名は、引用符 (“ファイル名”) または山括弧 (<ファイル名>) で括ります。ファイル名の後に引用符または山かっこを挿入してください。ファイル名の始まりに引用符が付いている場合、ファイル名の後に引用符を挿入しなくてはなりません。 ファイル名の始まりに山かっこがある場合、終わりの山かっこ (>)を挿入しなくてはなりません。

エラー F8511

メッセージ

インクルードファイルを開くことができません

説明

コンパイラが、スクリプト内にあるファイルを見つける、または開くことができませんでした。

トラブルシューティングのヒント

このエラーは、次の問題が原因で発生します。

- #include ステートメントのファイル名のスペルミス
- #include ステートメントのファイル名が完全なパスなしで指定されているが、このファイルはソースディレクトリでも見つからない。
- 探しているファイルが配置されているネットワークドライブのファイルを開く権限がない。
- ファイルが破損しています。

エラー F8512

メッセージ

インクルードファイルとパスが大きすぎます

説明

include ステートメントで指定したパスが、制限の長さ 256 文字を超えています。

トラブルシューティング情報

含まれたファイルを、長さが 256 文字未満のパスを使って参照することのできるフォルダーへ移動させてください。

エラー F8513

メッセージ

ネストされた #include ファイルが多すぎます

説明

InstallScript では、インクルードファイルのネストは最大 8 階層まで可能です。制限を超えています。

トラブルシューティング情報

ネスト階層の数を減らします。

エラー F8514

メッセージ

#if ステートメントのネストの数が多すぎます

説明

InstallScript では、#if ステートメントのネストは最大 10 階層まで可能です。制限を超えています。

トラブルシューティング情報

ネスト階層の数を減らします。

エラー F8515

メッセージ

マクロ拡張バッファーがオーバーフローしています

説明

エラー行にあるマクロの拡張が、制限の 1024 文字を超えています。

トラブルシューティング情報

コードを再構築してこのエラーを回避してください。

エラー F8516

メッセージ

最大エラー数に達しました

説明

[設定] ダイアログ ボックスの [コンパイル/リンク] タブで指定したエラーの最大数に達しました。

トラブルシューティング情報

[設定] ダイアログ ボックスの [コンパイル/リンク] タブについての詳細は、「[コンパイル/リンク] タブ」を参照してください。

エラー F8517

メッセージ

インクルードファイルが多すぎます

説明

インクルードの最大数を超えました。

トラブルシューティング情報

セットアップスクリプトは 100 以上のソースファイルを含むことができません。複数のソースファイルを組み合わせて全体の数を 100 に減らしてください。

エラー F8518

メッセージ

リンク ファイルの I/O エラー

説明

InstallShield がリンクファイルへ書き込む際にエラーが発生しました。

トラブルシューティング情報

セットアップをコンパイルする度に、InstallShield はファイル拡張子 .dbg、.ino、.inx、および .obs をもつ中間ファイルを作成します。コンパイルの後、これらのファイルはスクリプトフォルダーに残ります。このメッセージは、リンクファイル (.ino) へのアクセスの際にエラーが発生したとき表示されます。

このエラーには次の原因が考えられます:

- ディスクがいっぱいです。不要なファイルを削除して有効ディスクスペースを増やしてから、再度コンパイルしてください。
- リンクファイルが破損している。スクリプトフォルダーから削除または名前の変更を行った後、再度コンパイルしてください。

エラー F8519

メッセージ

ユーザー定義の致命的エラー

説明

このメッセージは #error 命令によってトリガーされます。

InstallScript 内部エラー

構文にエラーがある場合、内部エラーが発生する可能性があります。

C9001 内部エラー

エラー C9001

メッセージ

'テキスト' : 内部エラー

説明

テキスト が指定した文字 (単数または複数) が、区分できない構文エラーをトリガーしました。

トラブルシューティング情報

このエラーは構文エラーでトリガーされますが、これには特定のエラーメッセージがありません。

InstallScript 警告

コンパイラは、ランタイム エラーや効率悪化の原因となる、変則的なスクリプトを知らせる警告メッセージを出します。

注意 *- [コンパイラ設定] ダイアログ で [エラーとして警告] が選択されない限り、または指定された警告数を超えたとき以外、警告がスクリプトのコンパイルを妨げることはありません。デフォルトの最大値は 50 です。*

テーブル 20 - InstallScript 警告

警告番号	メッセージ
C7501	マクロの再定義
C7502	文字列/アレイサイズが推奨する制限を超えています
C7503	関数が定義されていますが呼び出されていません
C7505	typedef 定義が前のものと異なっています

警告 C7501

メッセージ

'名前' : マクロの再定義

説明

名前 が指定する識別子は、#define ステートメントで既に利用されています。

トラブルシューティング情報

2 番目の定義が単なる重複であった場合は、それを削除します。2 番目の定義が最初の定義とは異なる場合、2 番目の定義の識別子名を変更し、それが参照するスクリプト内のすべてのステートメントをアップデートします。

警告 C7502

メッセージ

'サイズ' :文字列/配列サイズが推奨する制限を超えています

説明

*サイズ*が指定した数値は、InstallScript の推奨最大値を超えた文字列または配列のサイズを示します。

トラブルシューティング情報

予期しない影響を避けるため、配列サイズ縮小します。

Warning C7503

メッセージ

'name' : function defined but never called

説明

名前 が指定する関数はセットアップでは呼び出されません。

トラブルシューティング情報

セットアップサイズを縮小するには、スクリプトから利用しない関数を削除します。

Warning C7505

Message

'name' : typedef definition differs from previous

説明

名前 が指定するデータ構造は、複数ライブラリ ファイル (.obl ファイル) で定義されています。

トラブルシューティング情報

定義が重複しているだけならば、片方を削除してください。別の定義が必要な場合、データ構造の名前を変更して固有の名前を付け、それを参照するスクリプト内のすべてのステートメントを更新します。

仮想化変換のエラーと警告

Windows Installer パッケージを仮想アプリケーションに変換するとき、エラーや警告が発生することがあります。一部のエラーと警告は、すべての種類のパッケージの仮想化に対して共通で、他は、パッケージを作成している仮想化ソリューションに対して固有です。

エラー -9000: 不明な例外

次のテーブルは、このメッセージの説明です:

テーブル 1 • エラー -9000: 不明な例外

分類	説明
種類:	エラー
メッセージ:	不明な例外が発生しました。
原因:	これは予期しない内部エラーです。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9001: 不明な COM

次のテーブルは、このメッセージの説明です:

テーブル 2 • エラー -9001: 不明な COM

分類	説明
種類:	エラー
メッセージ:	内部エラー。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9002: パッケージを開くときにエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 3 • エラー -9002: パッケージを開くときにエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	パッケージを開く際に、エラーが発生しました。

テーブル 3 ・ エラー -9002: パッケージを開くときにエラーが発生しました

分類	説明
原因:	これは、Windows Installer パッケージの読み込み中に発生した予期しない内部エラーです。
解決方法:	ユーザーがパッケージにアクセス可能であることを確認してください。エラーが引き続き発生し、パッケージがネットワーク共有にある場合、パッケージをローカルにコピーして（ネットワークの接続に関する問題を避けるため）、再試行してください。 これによって問題が解決されなかった場合、事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9003: パッケージを保存するときにエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 4 ・ エラー -9003: パッケージを保存するときにエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	パッケージを保存中に、エラーが発生しました。
原因:	これは、Citrix プロファイルを保存しようとしたときに発生した予期しない内部エラーです。
解決方法:	ユーザーにプロファイルをビルド中の場所へのアクセスがあることを確認してください。 これによって問題が解決されなかった場合、事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9004: ユーザーによってプロセスがキャンセルされました

次のテーブルは、このメッセージの説明です:

テーブル 5 ・ エラー -9004: ユーザーによってプロセスがキャンセルされました

分類	説明
種類:	エラー
メッセージ:	処理がユーザーによってキャンセルされました。
原因:	ユーザーが **Cancel** ボタンをクリックして、ビルドを停止しました。
解決方法:	ビルドの再度開始します。

エラー -9005: 一時フォルダーの作成中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 6 • エラー -9005: 一時フォルダーの作成中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	一時フォルダーを作成中にエラーが発生しました
原因:	このエラーは、ユーザーが C:¥TMP への書き込みの権限がないとき、またはドライブの容量が足りないとき発生します。
解決方法:	C:¥TMP への書き込みアクセスを取得し、ドライブのディスク領域を解放してから、プロファイルを再ビルドしてください。

エラー -9006: パッケージの圧縮中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 7 • エラー -9006: パッケージの圧縮中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	パッケージ 'PackageName' を圧縮解除中にエラーが発生しました。
原因:	このエラーは、パッケージが圧縮の Windows Installer パッケージ (.msi) の発生します。エラーは、InstallShield が管理インストールを実行して、ファイルを抽出しようとしたとき生成されました。
解決方法:	このエラーが発生したとき、Windows Installer からエラー コードが戻されます。Windows Installer ヘルプ ライブラリでエラー コードを調べ、問題の原因を判別してください。 Windows Installer からエラー コードが戻されなかった場合、このエラーは、適切に作成されなかったパッケージが原因によるものです。Windows Installer パッケージで、AdminExecuteSequence テーブルが定義されているかどうかを確認してください。このテーブルがない場合、パッケージは圧縮できません。

エラー -9007: 拡張子を持つファイルが見つかりませんでした

次のテーブルは、このメッセージの説明です:

テーブル 8 • エラー -9007: 拡張子を持つファイルが見つかりませんでした

分類	説明
種類:	エラー
メッセージ:	拡張子 'ComponentKeyName' を持つファイルが見つかりませんでした。
原因:	このエラーは、ファイル拡張子の処理中に発生した予期しないエラーです。
解決方法:	ファイル拡張子の実行可能ファイルが存在し、そのコンポーネントのキー ファイルとして設定されていることを確認してください。

エラー -9008: アイコンの抽出中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 9 • エラー -9008: アイコンの抽出中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	アイコン 'IconKeyName' を抽出中にエラーが発生しました
原因:	これは、Icon テーブル内にあるアイコンの抽出中に発生した予期しないエラーです。
解決方法:	Icon テーブルの Icon エントリが有効であることを確認してください。必要に応じて、有効なアイコンで置き換えてください。

エラー -9009: 不明のプロバイダー

次のテーブルは、このメッセージの説明です:

テーブル 10 • エラー -9009: 不明のプロバイダー

分類	説明
種類:	エラー
メッセージ:	指定されたプロバイダーは知られていない 'ProviderName' です。
原因:	これは予期しない内部エラーです。

テーブル 10 • エラー -9009: 不明のプロバイダー

分類	説明
解決方法:	このエラーは、ダイレクト エディターと使って変更した無効なデータが原因によるものです。ビルド中の **リリース**を削除して、新しいリリースを作成して再ビルドしてください。

エラー -9010: ターゲット ファイル名が正しくありません

次のテーブルは、このメッセージの説明です:

テーブル 11 • エラー -9010: ターゲット ファイル名が正しくありません

分類	説明
種類:	エラー
メッセージ:	ターゲット ファイルの名前が無効です。'FileName'
原因:	これは予期しない内部エラーです。
解決方法:	このエラーは、ダイレクト エディターと使って変更した無効なデータが原因によるものです。Citrix アシスタント / ThinAppn アシスタントの **[プロファイル情報]** ページにある **[名前]** フィールドを確認し、名前にファイル名に無効な文字が使用されていないことを確認してください。

エラー -9011: MSI テーブルの読み込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 12 • エラー -9011: MSI テーブルの読み込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	MSI テーブル 'TableName' を読み取り中に予期しないエラーが発生しました
原因:	このエラーは、指定さえた Windows Installer テーブルの処理中に発生した予期しないエラーです。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9012: メソッドで予期しないエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 13 • エラー -9012: メソッドで予期しないエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	メソッド 'MethodName' で予期しないエラーが発生しました
原因:	これは予期しない内部エラーです。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9013: タイプ ライブラリが見つかりませんでした

次のテーブルは、このメッセージの説明です:

テーブル 14 • エラー -9013: タイプ ライブラリが見つかりませんでした

分類	説明
種類:	エラー
メッセージ:	タイプライブラリが見つかりませんでした: 'TypeLibraryName'
原因:	このエラーは、COM 情報の抽出時にタイプ ライブラリ ファイルが見つからなかったとき発生します。
解決方法:	プロファイルのビルド時にタイプ ライブラリ ファイルが適切な場所にあるかどうかを確認してください。 これによって問題が解決されなかった場合、事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9014: ShellExecute が失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 15 • エラー -9014: ShellExecute が失敗しました

分類	説明
種類:	エラー
メッセージ:	ShellExecute が失敗しました: 'CommandLine'

テーブル 15・エラー -9014: ShellExecute が失敗しました

分類	説明
原因:	指定したコマンドラインがプロセスを起動できなかったとき、このエラーが発生します。
解決方法:	表示されている実行可能ファイルの名前が有効なファイルであり、ユーザーにそれを実行する権限があることを確認してください。 これによって問題が解決されなかった場合、事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9015: ドライバーの完全パスを判別できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 16・エラー -9015: ドライバーの完全パスを判別できませんでした

分類	説明
種類:	警告
メッセージ:	ドライバー 'DriverName' の完全パスを判別できません
原因:	このエラーは、ODBCDataSource テーブルで参照されたドライバーがパッケージでインストールされないとき発生します。
解決方法:	このエラーは、次のいずれかの方法で解決できます: **Windows Installer パッケージを編集する** 1. パッケージを InstallShield のダイレクト編集モードを使って編集します。 2. ISVirtualPackage テーブルに移動します。 3. 次のようにエントリを作成して、不足しているドライバーの完全パスを識別します: **名前:<ドライバー名> 説明** **値: ドライバーへのパス** **ドライバーを手動でインストールする** 不足しているドライバーをマシンにインストールして、Citrix プロファイルを再ビルドします。

警告 -9016: テーブルのコンテンツが無視されました

次のテーブルは、このメッセージの説明です:

テーブル 17 • 警告 -9016: テーブルのコンテンツが無視されました

分類	説明
種類:	警告
メッセージ:	テーブル 'TableName' のコンテンツは無視されます。
原因:	このエラー メッセージによって、Citrix 変換の既知の制限が識別されます。
解決方法:	テーブルのコンテンツが必須の場合、アプリケーションをリパッケージして、Citrix プロファイルを再ビルドします。

警告 -9017: .NET 1.x アセンブリはサポートされていません

次のテーブルは、このメッセージの説明です:

テーブル 18 • 警告 -9017: .NET 1.x アセンブリはサポートされていません

分類	説明
種類:	警告
メッセージ:	アセンブリ 'AssemblyName' は .NET 1.x アセンブリであるため、正しく変換されません。.NET 2.0/3.0 アセンブリのみが現在サポートされています。このパッケージをまず最初に再パッケージすることを検討してみてください。
原因:	このエラーは、.NET 1.x アセンブリを含むパッケージの変換が試みられたときに発生します。.NET 2.0/3.0 アセンブリのみが現在サポートされています。
解決方法:	アプリケーションをリパッケージして、Citrix プロファイルを再ビルドします。

警告 -9018: カスタム アクションが無視されました

次のテーブルは、このメッセージの説明です:

テーブル 19 • 警告 -9018: カスタム アクションが無視されました

分類	説明
種類:	警告
メッセージ:	カスタム アクション 'CustomActionName' は無視されます。

テーブル 19 • 警告 -9018: カスタム アクションが無視されました

分類	説明
原因:	Windows Installer パッケージを Citrix プロファイルに変換するとき、すべてのカスタム アクションは無視されます。Windows Installer パッケージのカスタム アクションが作成したターゲット マシンへの変更は、Citrix プロファイルに伝達されません。 ***メモ:*** *システムを変更しない、またはインストールの実行に使用されないカスタム アクション (InstallShield エディター の定義済みカスタム アクションや タイプ 19 カスタム アクションなど) が実行されたとき、メッセージは表示されません。タイプ 51 カスタム アクション (書式付きテキスト文字列からプロパティを設定します) が実行された場合、そのカスタム アクションは自動的に解決されます。タイプ 35 カスタム アクションが実行された場合、そのカスタム アクションが* Directory *テーブルで参照されているときのみ解決されます。*
解決方法:	解決方法は、カスタム アクションの目的によって異なります: • カスタム アクションが単に値の入力や他の種類の軽度の変更を自動的に行う場合、この警告は無視することができます。 • カスタム アクションの実行によってインストールの動作が変更される可能性がある場合 (Property の設定など)、この問題を解決する必要が場合によってあります。 この問題を解決するには、まず、変換済みのパッケージを Citrix XenApp 上で起動してみます。アプリケーション エラーが発生した場合、このアプリケーションはリパッケージが必要です。 ***Windows Installer パッケージをリパッケージして、カスタム アクション機能をキャプチャするには、以下の手順に従います:*** 1. リパッケージ ウィザード を使って、このアプリケーションをリパッケージします。アプリケーションのインストール中、リパッケージ ウィザード はシステム変更を監視します。データはこのあと リパッケージャー プロジェクトに変換されます。 2. リパッケージャー プロジェクトをビルドして、訂正した Windows Installer パッケージを生成します。この新しい Windows Installer パッケージにはカスタム アクションが含まれていませんが、(リパッケージ プロセスの結果として) 元のカスタム アクションによって実行された機能を含みます。

警告 -9019: 条件付きコンポーネント

次のテーブルは、このメッセージの説明です:

テーブル 20 • 警告 -9019: 条件付きコンポーネント

分類	説明
種類:	警告

テーブル 20・警告 -9019: 条件付きコンポーネント

分類	説明
メッセージ:	正しく変換されない可能性がある 1 つ以上の条件付きコンポーネントが存在します
原因:	この警告は、条件付のコンポーネントを変換しようとしたとき、コンポーネントの条件が評価されなかったために発生します。結果として、コンポーネントが仮想パッケージに含まれます。
解決方法:	コンポーネントを仮想パッケージに含めない場合、.msi パッケージまたはプロジェクト ファイルを変更して、コンポーネントを除外します。それから、仮想アプリケーションを再ビルドします。仮想パッケージにこのコンポーネントを含める場合、この警告は無視できます。

エラー -9020: ディレクトリの親がヌルです

次のテーブルは、このメッセージの説明です:

テーブル 21・エラー -9020: ディレクトリの親がヌルです

分類	説明
種類:	エラー
メッセージ:	ディレクトリ 'DirectoryName' にヌルの親があります。このディレクトリは無視されます。
原因:	このエラーは、ディレクトリ テーブルのエントリ (TARGETDIR 以外) がヌルのとき発生します。
解決方法:	ThinApp で仮想化されたアプリケーションを評価して、適切に動作するか確認します。適切に動作しない場合、パッケージをリパッケージすることをお勧めします。

エラー -9021: COM データを抽出できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 22 • エラー -9021: COM データを抽出できませんでした

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'FileName' のデータをサポートしていません。
原因:	この Windows Installer パッケージの TypeLib または SelfReg テーブルに、でアプリケーションデータに変換できない COM データを含むエントリがあります。 COM 抽出できないファイルによっては、COM テーブルのマップを無効にした状態で Windows Installer パッケージをリパッケージすることにより、このアプリケーションを Citrix XenApp 分離環境で適切に実行することが可能です。 COM データは Windows Registry に格納されます。従って、Windows Installer パッケージをリパッケージすると、キャプチャ プロセスで、Registry への変更がすべてキャプチャされるため、データがすべて取得可能になります。COM 抽出の必要はありません。
解決方法:	この問題を解決するには、COM テーブルのマップを無効にした状態で Windows Installer パッケージをリパッケージする必要があります。

エラー -9022: Complus テーブル

次のテーブルは、このメッセージの説明です:

テーブル 23 • エラー -9022: Complus テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'Complus' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに Complus テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、Complus テーブルは読み込まれません。
解決方法:	Complus テーブルには、COM+ アプリケーションをインストールするために必要な情報が含まれています。Citrix XenApp では、COM+ アプリケーションとの交信がサポートされていますが、COM+ サービスの*インストール*はサポートされていません。したがって、このアプリケーションは Citrix XenApp で展開することはできません。

エラー -9024: FileSFPCatalog

次のテーブルは、このメッセージの説明です:

テーブル 24 • エラー -9024: FileSFPCatalog

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'FileSFPCatalog' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **FileSFPCatalog** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**FileSFPCatalog** テーブルは読み込まれません。
解決方法:	**FileSFPCatalog** テーブルは、指定されたファイルをシステム ファイルの保護によって使用されているカタログ ファイルに関連付けます。このファイルがアプリケーションの機能にとって必要な場合、リパッケージャー を使って、アプリケーションをリパッケージする必要があります。

警告 -9026: LaunchCondition テーブル

次のテーブルは、このメッセージの説明です:

テーブル 25 • 警告 -9026: LaunchCondition テーブル

分類	説明
種類:	警告
メッセージ:	変換プロセスは MSI テーブル 'LaunchCondition' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに LaunchCondition テーブルが含まれているとき、この警告が発生します。変換プロセス中に、**LaunchCondition** テーブルは読み込まれません。

テーブル 25 • 警告 –9026: LaunchCondition テーブル

分類	説明
解決方法:	**LaunchCondition** テーブルには、インストールを開始するためにすべて満たされる必要がある条件の一覧が含まれています。たとえば、アプリケーションで Windows XP の実行が必要な場合、Windows XP が **LaunchCondition** テーブルに含まれます。このテーブルは読み込まれないため、検証は実行されません。従って、Windows XP 以外のオペレーティング システムを実行しているユーザーがこの Citrix プロファイルを起動したとき、アプリケーションが適切に機能しない場合があります。 この問題を解決するには、以下のいずれかのタスクを実行します: • **オプション 1: [プロファイルの要件] ページで要件を設定する** – 起動条件にオペレーティング システム、Service Pack、言語要件のみ含まれている場合、このパッケージを Citrix アシスタントで開いて、**[プロファイルの要件]** ページでそれらのオペレーティング システムと言語要件を設定します。そのあと、このアプリケーションを Citrix プロファイルとして展開します。 • **オプション 2: 起動条件が一致したかどうかを判別する** – テーブルに一覧表示されている起動条件を確認して、組織内にあるデスクトップがこれらの要件に一致するかどうかを判別します。すべてのデスクトップが要件に一致する場合、このアプリケーションを Citrix プロファイルとして展開します。 デスクトップが要件に一致しない場合 (Internet Explorer 7.0 の代わりに 6.0 があるなど)、要件に一致するようにデスクトップをアップグレードしてから、このアプリケーションを Citrix プロファイルとして展開します。

警告 –9027: LockPermissions テーブル

次のテーブルは、このメッセージの説明です:

テーブル 26 • 警告 –9027: LockPermissions テーブル

分類	説明
種類	警告
メッセージ:	変換プロセスは MSI テーブル 'LockPermissions' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに LockPermissions テーブルが含まれているとき、この警告が発生します。変換プロセス中に、LockPermissions テーブルは読み込まれません。
解決方法:	**LockPermissions** テーブルは、アプリケーションの個々の箇所 (ファイル、レジストリ キー、ロックダウンされた環境で作成されたフォルダー) をセキュリティで保護するために使用されます。 Citrix は、ファイル、レジストリ、または作成されたフォルダーの権限をサポートしません。アプリケーションの ACL (アクセス制御リスト) の権限は変更することができません。このアプリケーションを分離環境で実行するとき、ユーザーにはすべての権限があるため、この警告により問題が発生することはありません。

エラー -9028: MoveFile テーブル

次のテーブルは、このメッセージの説明です:

テーブル 27 • エラー -9028: MoveFile テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'MoveFile' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに MoveFile テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、MoveFile テーブルは読み込まれません。
解決方法:	この MoveFile テーブルには、指定されたソース ディレクトリから指定されたインストール先ディレクトリに移動またはコピーされるファイルの一覧が含まれています。このテーブルは読み込まれないため、この問題を解決するには、次のいずれかを実行する必要があります: • **オプション 1: Windows Installer パッケージを編集する**— InstallShield で Windows Installer パッケージを開いき、指定されたディレクトリに追加のファイルをインストールして、**MoveFile** テーブルを使用する必要がなくなるようにします。 • **オプション 2: アプリケーションをリパッケージする** — リパッケージ ウィザード を利用して、このアプリケーションをリパッケージし、リパッケージャー プロジェクトをビルドして、訂正した Windows Installer パッケージを生成します。 • **オプション 3: 起動前スクリプトを作成する** — アプリケーションの起動時に **MoveFile** テーブルで識別されるファイルの移動操作を実行する起動前スクリプトを作成します。

エラー -9029: MsiDriverPackages テーブル

次のテーブルは、このメッセージの説明です:

テーブル 28 • エラー -9029: MsiDriverPackages テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'MsiDriverPackages' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **MsiDriverPackages** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**MsiDriverPackages** テーブルは読み込まれません。

テーブル 28 • エラー -9029: MsiDriverPackages テーブル

分類	説明
解決方法:	**MsiDriverPackages** テーブルには、アプリケーション内の 1 つのドライバー パッケージにつき 1 つのレコードが含まれています。 Citrix XenApp では、ドライバーは、どの種類もサポートされていません。たとえば、プリンターをインストールするとき、プリンター ソフトウェアを分離環境からインストールすることができますが、プリンター ドライバーはインストールできません。 このため、この問題を解決するには、必要なプリンター ドライバーを分離環境の外、つまりユーザーのデスクトップ マシンにインストールする必要があります。

警告 -9030: ODBCTranslator テーブル

次のテーブルは、このメッセージの説明です:

テーブル 29 • 警告 -9030: ODBCTranslator テーブル

分類	説明
種類:	警告
メッセージ:	変換プロセスは MSI テーブル 'ODBCTranslator' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **ODBCTranslator** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ODBCTranslator** テーブルは読み込まれません。
解決方法:	**ODBCTranslator** テーブルには、インストールに属する ODBC トランスレーターの一覧が含まれています。ODBC トランスレーターは、ある形式の未処理データを特定の種類のデータベースで使用できる別の形式に変換します。 **ODBCTranslator** テーブルを無視することにより、アプリケーションが適切に動作しなくなることはありません。これにより問題が発生した場合、リパッケージャー を使って、アプリケーションをリパッケージする必要があります。

警告 -9031: RemoveFile テーブル

次のテーブルは、このメッセージの説明です:

テーブル 30 • 警告 -9031: RemoveFile テーブル

分類	説明
種類:	警告
メッセージ:	変換プロセスは MSI テーブル 'RemoveFile' のデータをサポートしていません。

テーブル 30 • 警告 –9031: RemoveFile テーブル

分類	説明
原因:	変換中の Windows Installer パッケージに **RemoveFile** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**RemoveFile** テーブルは読み込まれません。この警告は、アンインストールではなくインストール中にアプリケーション インストールがファイルを削除したとき表示されます。
解決方法:	**RemoveFile** テーブルには、削除されるファイルの一覧が含まれています。ファイルの削除要件がクリーン アップ ステップのみの場合、アプリケーションの機能に対して影響はないため、この問題を解決する必要はありません。 ただし、**RemoveFile** テーブルに含まれているファイルの存在によってアプリケーションが適切に機能しなくなる場合、ファイルが分離環境で表示されなくなるように、ファイルの分離オプションを **Ignore** に設定する必要があります。**Ignore** オプションは、分離環境が常に (分離環境の外にあるファイルを無視して) システム上にあるファイルを探すように指示します。

警告 –9032: RemoveIniFile テーブル

次のテーブルは、このメッセージの説明です:

テーブル 31 • 警告 –9032: RemoveIniFile テーブル

分類	説明
種類:	警告
メッセージ:	変換プロセスは MSI テーブル 'RemoveIniFile' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **RemoveIniFile** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**RemoveIniFile** テーブルは読み込まれません。
解決方法:	**RemoveIniFile** テーブルには、アプリケーションが .ini ファイルから削除する必要がある情報が含まれています。このエントリの削除がアプリケーションの機能にとって必要な場合、リパッケージャー を使って、アプリケーションをリパッケージする必要があります。

警告 –9033: RemoveRegistry テーブル

次のテーブルは、このメッセージの説明です:

テーブル 32 • 警告 –9033: RemoveRegistry テーブル

分類	説明
種類:	警告
メッセージ:	変換プロセスは MSI テーブル 'RemoveRegistry' のデータをサポートしていません。

テーブル 32 • 警告 –9033: RemoveRegistry テーブル

分類	説明
原因:	変換中の Windows Installer パッケージに **RemoveRegistry** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**RemoveRegistry** テーブルは読み込まれません。
解決方法:	**RemoveRegistry** テーブルには、アプリケーションがシステム レジストリから削除する必要があるレジストリ情報が含まれています。この削除要件がクリーン アップ ステップのみの場合、アプリケーションの機能に対して影響はないため、この問題を解決する必要はありません。 ただし、**RemoveRegistry** テーブルに含まれているレジストリ キーの存在によってアプリケーションが適切に機能しなくなる場合、レジストリ キーが分離環境で表示されなくなるように、レジストリ キーの分離オプションを **Ignore** に設定する必要があります。**Ignore** オプションは、分離環境が常に (分離環境の外にあるレジストリ キーを無視して) システム上にあるレジストリ キーを探すように指示します。

エラー –9036: ISCEInstall テーブル

次のテーブルは、このメッセージの説明です:

テーブル 33 • エラー –9036: ISCEInstall テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'ISCEInstall' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **ISCEInstall** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ISCEInstall** テーブルは読み込まれません。
解決方法:	**ISCEInstall** テーブルは、Windows Mobile アプリケーションをインストールするために使用されます。モバイル アプリケーションを Citrix プロファイルに変換することができません。

エラー –9037: ISComPlusApplication テーブル

次のテーブルは、このメッセージの説明です:

テーブル 34 • エラー –9037: ISComPlusApplication テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'ISComPlusApplication' のデータをサポートしていません。

テーブル 34 • エラー –9037: ISComPlusApplication テーブル

分類	説明
原因:	変換中の Windows Installer パッケージに **ISComPlusApplication** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ISComPlusApplication** テーブルは読み込まれません。
解決方法:	**ISComPlusApplication** テーブルには、COM+ アプリケーションについての情報が含まれています。Citrix XenApp では、COM+ アプリケーションとの交信がサポートされていますが、COM+ サービスの*インストール*はサポートされていません。したがって、このアプリケーションは Citrix XenApp で展開することはできません。

エラー –9038: ISPalmApp テーブル

次のテーブルは、このメッセージの説明です:

テーブル 35 • エラー –9038: ISPalmApp テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'ISPalmApp' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **ISPalmApp** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ISPalmApp** テーブルは読み込まれません。
解決方法:	**ISPalmApp** テーブルは、Palm モバイル アプリケーションをインストールするために使用されます。モバイル アプリケーションを Citrix プロファイルに変換することができません。

エラー –9039: ISSQLScriptFile テーブル

次のテーブルは、このメッセージの説明です:

テーブル 36 • エラー –9039: ISSQLScriptFile テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'ISSQLScriptFile' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **ISSQLScriptFile** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ISSQLScriptFile** テーブルは読み込まれません。

テーブル 36 • エラー –9039: ISSQLScriptFile テーブル

分類	説明
解決方法:	**ISSQLScriptFile** テーブルには、SQL スクリプトの一覧が含まれています。Windows Installer パッケージがインストールされたとき、SQL スクリプトを実行して、データベースを更新することができます。Citrix プロファイルとして実行されているアプリケーションは、データベースを更新できません。 この問題を解決するには、変換済みの Citrix プロファイルを使用する前に、次のいずれかの方法を使ってデータベースを更新します: • スクリプトを利用して、データベースを手動で更新する。 • データベースへのアクセスがあるネットワーク上のマシンの 1 つで Windows Installer インストールを実行して、データベースを更新する。。

エラー –9040: ISVRoot テーブル

次のテーブルは、このメッセージの説明です:

テーブル 37 • エラー –9040: ISVRoot テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'ISVRoot' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **ISVRoot** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ISVRoot** テーブルは読み込まれません。
解決方法:	**ISVRoot** テーブルは Web サイトをインストールします。分離環境で Citrix プロファイルとして実行されているアプリケーションは、Web サイトを作成できません。したがって、インストール時に Web サイトを作成するアプリケーションに Citrix プロファイルを作成することはサポートされていません。

エラー -9041: ISXmlFile テーブル

次のテーブルは、このメッセージの説明です:

テーブル 38 • エラー -9041: ISXmlFile テーブル

分類	説明
種類:	エラー
メッセージ:	変換プロセスは MSI テーブル 'ISXmlFile' のデータをサポートしていません。
原因:	変換中の Windows Installer パッケージに **ISXmlFile** テーブルが含まれているとき、このエラーが発生します。変換プロセス中に、**ISXmlFile** テーブルは読み込まれません。
解決方法:	**ISXmlFile** テーブルは XML ファイルを変更します。このアプリケーションが適切に実行されるために XML ファイルの変更が必要な場合、リパッケージャー を使って、このアプリケーションをリパッケージする必要があります。

エラー -9051: パッケージの圧縮中がキャンセルされました

次のテーブルは、このメッセージの説明です:

テーブル 39 • エラー -9051: パッケージの圧縮中がキャンセルされました

分類	説明
種類:	エラー
メッセージ:	パッケージの圧縮解除がユーザーによってキャンセルされました
原因:	ユーザーによって、圧縮された MSI パッケージの圧縮解除プロセスがキャンセルされました。

エラー -9100: パッケージ オブジェクトのインスタンスの作成が失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 40 • エラー -9100: パッケージ オブジェクトのインスタンスの作成が失敗しました

分類	説明
種類:	エラー
メッセージ:	Citrix パッケージ オブジェクトのインスタンスを作成できませんでした
原因:	予期しない内部エラー。

テーブル 40 • エラー -9100: パッケージ オブジェクトのインスタンスの作成が失敗しました

分類	説明
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9101: パッケージ オブジェクトの作成操作が失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 41 • エラー -9101: パッケージ オブジェクトの作成操作が失敗しました

分類	説明
種類:	エラー
メッセージ:	Citrix パッケージ オブジェクトを作成できませんでした: 'ObjectName'
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

Error -9102: ヘッダー情報の書き込みに失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 42 • Error -9102: ヘッダー情報の書き込みに失敗しました

分類	説明
種類:	エラー
メッセージ:	パッケージのヘッダー情報の書き込みに失敗しました。
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9103: Citrix の最終処理が失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 43 • エラー -9103: Citrix の最終処理が失敗しました

分類	説明
種類:	エラー
メッセージ:	Citrix の最終処理が失敗しました
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9104: Citrix の保存に失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 44 • エラー -9104: Citrix の保存に失敗しました

分類	説明
種類:	エラー
メッセージ:	Citrix の保存に失敗しました
原因:	予期しない内部エラー。このエラーは、プロファイルをデジタル署名するとき発生することがあります。
解決方法:	Citrix プロファイルをデジタル署名するオプションを選択解除して、プロファイルを再ビルドします。

エラー -9105: Citrix ライターの初期化中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 45 • エラー -9105: Citrix ライターの初期化中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix ライターの初期化中に予期しないエラーが発生しました
原因:	予期しない内部エラー。

テーブル 45 • エラー –9105: Citrix ライターの初期化中にエラーが発生しました

分類	説明
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー –9106: Citrix パッケージの初期化中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 46 • エラー –9106: Citrix パッケージの初期化中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix パッケージの初期化中に予期しないエラーが発生しました
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー –9107: Citrix ファイル エントリの書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 47 • エラー –9107: Citrix ファイル エントリの書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix ファイル エントリの書き込み中に予期しないエラーが発生しました
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9108: ソース ファイル パスの判別中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 48 • エラー -9108: ソース ファイル パスの判別中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	'FileName' のソース ファイル パスの判別中に予期しないエラーが発生しました
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9109: Citrix フォルダー エントリの書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 49 • エラー -9109: Citrix フォルダー エントリの書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix フォルダー エントリの書き込み中に予期しないエラーが発生しました
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9110: Citrix レジストリ エントリの書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 50 • エラー -9110: Citrix レジストリ エントリの書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix レジストリ エントリの書き込み中に予期しないエラーが発生しました

テーブル 50・エラー –9110: Citrix レジストリ エントリの書き込み中にエラーが発生しました

分類	説明
原因:	予期しない内部エラー。
解決方法:	最初に、製品が適切にインストールされていることを確認してください。そのあと、事前の調査を行い、InstallShield のテクニカル サポートまでお問い合わせください。

エラー –9113: Citrix INI ファイル エントリの書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 51・エラー –9113: Citrix INI ファイル エントリの書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix INI ファイル エントリの書き込み中に予期しないエラーが発生しました
原因:	予期しない内部エラー。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー –9114: Citrix ショートカットの書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 52・エラー –9114: Citrix ショートカットの書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix ショートカットの書き込み中に予期しないエラーが発生しました
原因:	ショートカットをプロファイルに書き込む中に、致命的なエラーが発生しました。
解決方法:	ショートカットが有効なファイルをポイントすることを確認してください。問題を効率よく解決するために、ショートカットを 1 つだけ保持して、再ビルドを試みてください。

エラー -9115: Citrix プロファイルを保存するときにエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 53 • エラー -9115: Citrix プロファイルを保存するときにエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix プロファイルの保存中に予期しないエラーが発生しました
原因:	Citrix プロファイルを保存中に、致命的なエラーが発生しました。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9116: 空の Citrix プロファイルを作成するときにエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 54 • エラー -9116: 空の Citrix プロファイルを作成するときにエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	空の Citrix プロファイルの作成中に予期しないエラーが発生しました
原因:	InstallShield では、Citrix プロファイルの内部インストールを新規作成することができません。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9117: 中間フォルダーの作成中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 55 • エラー -9117: 中間フォルダーの作成中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	中間フォルダーの作成中に予期しないエラーが発生しました

テーブル 55・エラー -9117: 中間フォルダーの作成中にエラーが発生しました

分類	説明
原因:	InstallShield では、このビルドに使用する中間フォルダーを作成することができません。このエラーは、ユーザーが C:¥TMP への書き込みの権限がないとき発生することがあります。
解決方法:	C:¥TMP への書き込みアクセスを取得し、プロファイルを再ビルドしてください。

エラー -9118: Citrix プロファイルの初期化中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 56・エラー -9118: Citrix プロファイルの初期化中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix プロファイルの初期化中に予期しないエラーが発生しました
原因:	新しいプロファイルの初期値が設定できませんでした。
解決方法:	パッケージ名、説明、バージョン、セキュリティに関する設定を確認して、原因を判別します。

エラー -9119: Citrix プロファイルにデフォルト ターゲットを作成するときにエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 57・エラー -9119: Citrix プロファイルにデフォルト ターゲットを作成するときにエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix プロファイルでデフォルト ターゲットを作成中に予期しないエラーが発生しました
原因:	新しいプロファイルで初期ターゲットを作成できませんでした。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9120: プロファイルからファイルを削除中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 58 • エラー -9120: プロファイルからファイルを削除中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	プロファイルからファイル 'FileName' を削除中に予期しないエラーが発生しました
原因:	指定されたファイルをプロファイルから削除できませんでした。
解決方法:	ファイルが存在し、かつ、ユーザーにこのファイルへの権限があることを確認してください。 これによって問題が解決されなかった場合、事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9121: ファイルを Citrix プロファイルにコピーできませんでした

次のテーブルは、このメッセージの説明です:

テーブル 59 • エラー -9121: ファイルを Citrix プロファイルにコピーできませんでした

分類	説明
種類:	エラー
メッセージ:	ファイルを Citrix プロファイルにコピーできませんでした。エラー: 'Name' ファイル: 'Name'
原因:	指定されたファイルをプロファイルにコピーできませんでした。
解決方法:	ファイルが存在し、かつ、ユーザーにこのファイルへの権限があることを確認してください。 また、このエラーが発生したとき、Windows Installer からエラー コードが戻されます。Windows Installer ヘルプ ライブラリでエラー コードを調べ、問題の原因を判別してください。

エラー -9122: ターゲットが Citrix プロファイルに存在しません

次のテーブルは、このメッセージの説明です:

テーブル 60 • エラー -9122: ターゲットが Citrix プロファイルに存在しません

分類	説明
種類:	警告
メッセージ:	ショートカット 'ShortcutName' が Citrix プロファイルに存在しません。ショートカットを除外します。

テーブル 60・エラー –9122: ターゲットが Citrix プロファイルに存在しません

分類	説明
原因:	ショートカットがポイントするファイルがパッケージに含まれていません。
解決方法:	Citrix Assistant **Profile Shortcuts** ページで選択されているアイテムをクリアしてショートカットを除外し、プロファイルを再ビルドします。

エラー –9124: このプロファイルに作成されたショートカットがありません

次のテーブルは、このメッセージの説明です:

テーブル 61・エラー –9124: このプロファイルに作成されたショートカットがありません

分類	説明
種類:	エラー
メッセージ:	このプロファイルにショートカットは作成されませんでした。
原因:	Citrix プロファイルには、有効なショートカットが最低 1 つ含まれている必要があります。
解決方法:	Citrix Assistant **Profile Shortcuts** ページでショートカットを追加して、プロファイルを再ビルドします。

エラー –9125: Citrix ファイルの種類の関連付けの書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 62・エラー –9125: Citrix ファイルの種類の関連付けの書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Citrix のファイルの種類の関連付けの書き込み中に予期しないエラーが発生しました
原因:	ファイルの種類の関連付けを書き込むことができません。
解決方法:	まず事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

エラー -9126: 証明書を使ってプロファイルを署名できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 63 • エラー -9126: 証明書を使ってプロファイルを署名できませんでした

分類	説明
種類:	エラー
メッセージ:	指定された証明書を使ったプロファイルの署名に失敗しました
原因:	使用されている証明書が無効です。
解決方法:	有効な証明書を取得して、プロファイルを再ビルドします。

エラー -9127: スクリプトの実行を作成できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 64 • エラー -9127: スクリプトの実行を作成できませんでした

分類	説明
種類:	エラー
メッセージ:	"ScriptName" のスクリプト実行を作成することができませんでした
原因:	指定されたスクリプトに無効なデータが含まれています。
解決方法:	Citrix Assistant **Profile Shortcuts** ページで、プロファイルからスクリプトを削除してから再度追加し、プロファイルを再ビルドします。 これによって問題が解決されなかった場合、事前の調査を行ってから、InstallShield のテクニカル サポートまでお問い合わせください。

警告 -9128: ショートカットが重複しています

次のテーブルは、このメッセージの説明です:

テーブル 65 • 警告 -9128: ショートカットが重複しています

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカットは、既にプロファイル内に存在します。重複するショートカットを除外します。

テーブル 65 • 警告 -9128: ショートカットが重複しています

分類	説明
原因:	このプロファイルに、異なるスタート メニューの場所またはパッケージの他の場所を参照する複数のショートカットが定義されています。
解決方法:	これらのショートカットは必要ありません。Citrix Assistant **Profile Shortcuts** ページで、ショートカットを選択解除して、プロファイルを再ビルドします。

警告 -9129: ショートカットの名前が重複しています

次のテーブルは、このメッセージの説明です:

テーブル 66 • 警告 -9129: ショートカットの名前が重複しています

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカットは、既にプロファイル内に存在しますが、使用するコマンド ライン パラメーターが異なります。新しい一意のショートカット 'NewShortcutName(1)' が、プロファイル内に作成されます。
原因:	このプロファイルに、名前が同じで、異なるコマンドライン パラメーターを持つ 2 つのショートカットが定義されています。
解決方法:	Citrix アシスタント **[プロファイルのショートカット]** ページで、ショートカットの 1 つの名前を変更して、プロファイルを再ビルドします。

警告 -9130: ショートカットのターゲットが重複しています

次のテーブルは、このメッセージの説明です:

テーブル 67 • 警告 -9130: ショートカットのターゲットが重複しています

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカットは、既にプロファイル内に存在しますが、使用するコマンド ライン パラメーターが異なります。新しい一意のショートカット 'NewShortcutName(1)' が、プロファイル内に作成されます。
原因:	このプロファイルに、名前が同じで、異なるターゲットを持つ 2 つのショートカットが定義されています。
解決方法:	Citrix アシスタント **[プロファイルのショートカット]** ページで、ショートカットの 1 つの名前を変更して、プロファイルを再ビルドします。

警告 -9131: インストーラーの変数を解決できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 68 • 警告 -9131: インストーラーの変数を解決できませんでした

分類	説明
種類:	警告
メッセージ:	文字列 'StringName' 内のインストーラー変数を解決することができません
原因:	ビルド時に Windows Installer の変数の一部が解決できませんでした。アプリケーションで特定の値が必要な場合、これによりエラーが発生することがあります。
解決方法:	このアプリケーションをリパッケージして、プロファイルを再ビルドするか、Windows Installer パッケージで定数値を使用します。

警告 -9132: 16 色ショートカット アイコンが見つかりませんでした

次のテーブルは、このメッセージの説明です:

テーブル 69 • 警告 -9132: 16 色ショートカット アイコンが見つかりませんでした

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカット用の 16 色アイコンがありません。ショートカット アイコンがパブリッシュされたときに、歪んで表示される可能性があります。
原因:	このショートカットに使用されているアイコンに、16 色の画像が含まれていません。現在、Citrix では 16 色以上の画像がサポートされていないため、このアイコンが Citrix XenApp で表示およびパブリッシュされたとき、歪んで表示される可能性があります。
解決方法:	ショートカットを変更して、異なるアイコンを使用するか、16 色の画像を現在使用されているアイコンに追加します。

警告 -9133: ショートカット アイコンが見つかりませんでした

次のテーブルは、このメッセージの説明です:

テーブル 70 • 警告 -9133: ショートカット アイコンが見つかりませんでした

分類	説明
種類:	警告

テーブル 70・警告 -9133: ショートカット アイコンが見つかりませんでした

分類	説明
メッセージ:	'ShortcutName' ショートカット用のアイコンがありません。Windows アプリケーションの汎用アイコンが使用されます。
原因:	このショートカットのアイコンが見つからなかったとき、汎用的な Windows アプリケーション アイコンが使用されます。このエラーは、使用中のファイルが壊れているか、そのファイルから画像を抽出できないとき、発生することがあります。
解決方法:	ショートカットを変更して、異なるアイコンを使用します。

警告 -9134: 実行可能ファイルからアイコンを抽出できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 71・警告 -9134: 実行可能ファイルからアイコンを抽出できませんでした

分類	説明
種類:	警告
メッセージ:	実行可能ファイル 'filename' からアイコンを抽出できませんでした。実行可能ファイルが壊れていないことを確認してください。
原因:	アイコン ファイルが壊れている可能性があります。
解決方法:	ショートカットを変更して、異なるアイコンを使用します。

エラー -9135: ショートカットのターゲットが 16 ビットです

次のテーブルは、このメッセージの説明です:

テーブル 72・エラー -9135: ショートカットのターゲットが 16 ビットです

分類	説明
種類:	エラー
メッセージ:	ショートカット 'ShortcutName' のターゲットは 16 ビットです。このアプリケーションは、Citrix 分離環境で適切に機能しない可能性があります。
原因:	このショートカットがポイントするファイルが 16 ビット アプリケーションです。
解決方法:	ファイルをより新しい 32 ビット バージョンで置換します。Citrix 環境でアプリケーションが適切に動作するかどうかもテストすることもできます。

警告 -9136: 一部のファイルが圧縮されていない可能性があります

次のテーブルは、このメッセージの説明です:

テーブル 73 • 警告 -9136: 一部のファイルが圧縮されていない可能性があります

分類	説明
種類:	警告
メッセージ:	このパッケージにはデフォルト インストール レベルを 0 とする機能が含まれているため、一部のファイルが圧縮解除されない可能性があります。
原因:	圧縮された Windows Installer パッケージがインストールされるとき、ビルド エンジンは管理インストールを実行して、それを圧縮解除します。管理インストールでは、ファイルを含む機能のデフォルト インストール レベルが **0** のとき、そのファイルを圧縮解除しません。InstallShield では、これらの機能に含まれているコンポーネントにファイルがある場合、これらのファイルが Citrix プロファイルにコピーされようとしたとき、それらがソースの場所に存在しないため、エラーが発生します。
解決方法:	この問題を解決するには、Windows Installer パッケージを編集して、その機能のデフォルト インストール レベルを **0** 以外の値に設定します。

警告 -9137: 対象となるディレクトリが見つかりませんでした

次のテーブルは、このメッセージの説明です:

テーブル 74 • 警告 -9137: 対象となるディレクトリが見つかりませんでした

分類	説明
種類:	警告
メッセージ:	'FileName' ファイルのインストール先ディレクトリが見つかりません。変換プロセスを続行する前に、このアプリケーションのリパッケージを検討してください。
原因:	これは内部エラーです。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

Warning -9138: DuplicateFile テーブル エントリを無視しています

次のテーブルは、このメッセージの説明です:

テーブル 75 • Warning -9138: DuplicateFile テーブル エントリを無視しています

分類	説明
種類:	警告
メッセージ:	DestFolder に使用されているプロパティを解決できないため、DuplicateFile テーブルを無視しています: 'INVALIDPATH'
原因:	変換中の Windows Installer パッケージの DuplicateFile に 1 つ以上のエントリが含まれていて、DuplicateFile テーブルに含まれるエントリの 1 つの DestFolder 列に使用されているプロパティが解決できないときに、このエラーが発生することがあります。たとえば、重複するファイルのインストール先がカスタム アクションで設定されるとき、そのインストール先は変換中に解決されません。
解決方法:	**DuplicateFile** テーブルには、インストール中に、元のファイルと異なるディレクトリ、または別の名前を持つ同じディレクトリに複製が必要なファイルの一覧が含まれています。このテーブルのインストール先は解決されないため、この問題を解決するには、次のいずれかを実行する必要があります: • **オプション 1: Windows Installer パッケージを編集する**— InstallShield で Windows Installer パッケージを開いて、そのファイルのコピーを追加し、**DuplicateFile** テーブル内の問題のあるエントリを使用しないようにします。 • **オプション 2: アプリケーションをリパッケージする** — リパッケージ ウィザード を利用して、このアプリケーションをリパッケージし、リパッケージャー プロジェクトをビルドして、訂正した Windows Installer パッケージを生成します。 • **オプション 3: 起動前スクリプトを作成する**—アプリケーションの起動時に **DuplicateFile** テーブル内の問題のあるエントリに対してファイルのコピー操作を実行する起動前スクリプトを作成します。

警告 -9150: Windows Installer パッケージをビルド中に警告が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 76 • 警告 -9150: Windows Installer パッケージをビルド中に警告が発生しました

分類	説明
種類:	警告
メッセージ:	Windows Installer パッケージのビルド警告:*警告番号とメッセージ*。
原因:	この警告は、InstallShield で、作成中の App-V アプリケーションについて、新しい基本の MSI プロジェクトとリリースをビルドしたとき、ビルド警告が発生したことを意味します。

テーブル 76 • 警告 -9150: Windows Installer パッケージをビルド中に警告が発生しました (続き.)

分類	説明
解決方法:	この警告は、場合によって、InstallShield で App-V アプリケーションをビルドしたときに App-VPackage サブフォルダーに作成された基本の MSI プロジェクトで解決する必要があります。ただし、基本の MSI プロジェクトは、App-V プロジェクトを再ビルドするたびに上書きされます。

エラー -9151: Windows Installer パッケージをビルド中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 77 • エラー -9150: Windows Installer パッケージをビルド中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	Windows Installer パッケージのビルドエラー:*エラー番号とメッセージ*。
原因:	このエラーは、InstallShield で、作成中の App-V アプリケーションについて、新しい基本の MSI プロジェクトとリリースをビルドしたとき、ビルド エラーが発生したことを意味します。
解決方法:	このエラーは、場合によって、InstallShield で App-V アプリケーションをビルドしたときに App-VPackage サブフォルダーに作成された基本の MSI プロジェクトで解決する必要があります。ただし、基本の MSI プロジェクトは、App-V プロジェクトを再ビルドするたびに上書きされます。

Error -9200: ThinApp のインストールが必要です

次のテーブルは、このメッセージの説明です:

テーブル 78 • Error -9200: ThinApp のインストールが必要です

分類	説明
種類:	エラー
メッセージ:	ThinApp アプリケーションを適切にビルドするには、ThinApp のライセンス付きバージョンまたはデモバージョンがこのマシンにインストールされている必要があります。(www.vmware.com)
原因:	ThinApp がインストールされていません。
解決方法:	ThinApp をインストールします。

警告 -9201: ショートカット ファイルの拡張子は".exe" である必要があります

次のテーブルは、このメッセージの説明です:

テーブル 79 • 警告 -9201: ショートカット ファイルの拡張子は".exe" である必要があります

分類	説明
種類:	警告
メッセージ:	ショートカット 'ShortcutName' のターゲットの拡張子が、.exe ではありません。ショートカットを除外します。
原因:	ファイル名に .exe という拡張子がないショートカットは除外されます。
解決方法:	必要な操作はありません。

エラー -9202: アプリケーションが作成されませんでした

次のテーブルは、このメッセージの説明です:

テーブル 80 • エラー -9202: アプリケーションが作成されませんでした

分類	説明
種類:	エラー
メッセージ:	アプリケーションは作成されませんでした。
原因:	Windows Installer パッケージにショートカットがないか、すべてのショートカットが除外された可能性があります。
解決方法:	ソース Windows Installer .msi パッケージには、.exe ファイルへのショートカットが最低 1 つ必要です。

Error -9203: ThinApp ツールがありません

次のテーブルは、このメッセージの説明です:

テーブル 81 • Error -9203: ThinApp ツールがありません

分類	説明
種類:	エラー
メッセージ:	ThinApp: 'ToolName' は見つかりませんでした。

テーブル 81 • Error -9203: ThinApp ツールがありません

分類	説明
原因:	ThinApp アプリケーションをビルドするときに必要な ThinApp ツールの 1 つが見つかりませんでした。
解決方法:	ThinApp を再インストールします。

エラー -9204: ショートカットが重複しています

次のテーブルは、このメッセージの説明です:

テーブル 82 • エラー -9204: ショートカットが重複しています

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカットは既に存在します。重複するショートカットを除外します。
原因:	ソース パッケージに、同じ .exe ターゲットをポイントするショートカットが 2 つあります。
解決方法:	必要な操作はありません。

エラー -9205: 同じ名前のショートカットが既に存在しますが、パラメーターが異なります

次のテーブルは、このメッセージの説明です:

テーブル 83 • エラー -9205: 同じ名前のショートカットが既に存在しますが、コマンドライン パラメーターが異なります

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカットは既に存在しますが、コマンド ライン パラメーターが異なります。新しい一意のショートカットが作成されます。
原因:	パッケージ内の 2 つ同名のショートカットで、異なる引数が使用されています。
解決方法:	必要な操作はありません。

エラー -9206: 同じ名前のショートカットが既に存在しますが、ターゲットが異なります

次のテーブルは、このメッセージの説明です:

テーブル 84 • エラー -9206: 同じ名前のショートカットが既に存在しますが、ターゲットが異なります

分類	説明
種類:	警告
メッセージ:	'ShortcutName' ショートカットは既に存在しますが、ターゲットが異なります。新しい一意のショートカットが作成されます。
原因:	パッケージ内の 2 つ同名のショートカットが、異なるターゲットをポイントしています。
解決方法:	必要な操作はありません。

エラー -9207: ビルド プロセス中にエラーが発生しました (vregtool.exe)

次のテーブルは、このメッセージの説明です:

テーブル 85 • エラー -9207: ビルド プロセス中にエラーが発生しました (vregtool.exe)

分類	説明
種類:	エラー
メッセージ:	ThinApp (vregtool.exe) のビルド中に、エラーが発生しました。
原因:	ThinApp ビルド プロセスの vregtool.exe の実行中に、予期しないエラーが発生しました。
解決方法:	このエラーが発生した直前に表示された進行状況メッセージによって、このエラーの原因が明らかになる場合があります。 ビルド フォルダー階層構造にロックされたファイル/フォルダーがないことを確認してください。

エラー -9208: ビルド プロセス中にエラーが発生しました (vftool.exe)

次のテーブルは、このメッセージの説明です:

テーブル 86 • エラー -9208: ビルド プロセス中にエラーが発生しました (vftool.exe)

分類	説明
種類:	エラー
メッセージ:	ThinApp (vftool.exe) のビルド中に、エラーが発生しました

テーブル 86・エラー -9208: ビルド プロセス中にエラーが発生しました (vftool.exe)

分類	説明
原因:	ThinApp ビルド プロセスの vftool.exe の実行中に、予期しないエラーが発生しました。
解決方法:	このエラーが発生した直前に表示された進行状況メッセージによって、このエラーの原因が明らかになる場合があります。 ビルド フォルダー階層構造にロックされたファイル/フォルダーがないことを確認してください。

エラー -9209: ビルド プロセス中にエラーが発生しました (tlink.exe)

次のテーブルは、このメッセージの説明です:

テーブル 87・エラー -9209: ビルド プロセス中にエラーが発生しました (tlink.exe)

分類	説明
種類:	エラー
メッセージ:	ThinApp (tlink.exe) のビルド中にエラーが発生しました
原因:	ThinApp ビルド プロセスの tlink.exe の実行中に、予期しないエラーが発生しました。
解決方法:	このエラーが発生した直前に表示された進行状況メッセージによって、このエラーの原因が明らかになる場合があります。 ビルド フォルダー階層構造にロックされたファイル/フォルダーがないことを確認してください。

エラー -9300: AdviseFile 操作で、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 88・エラー -9300: AdviseFile 操作で、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	規則 'RuleName' の AdviseFile 操作で、未処理の例外が発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9301: AdviseRegistry 操作で、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 89 • エラー -9301: AdviseRegistry 操作で、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	規則 'RuleName' の AdviseRegistry 操作で、未処理の例外が発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9302: コマンド操作で、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 90 • エラー -9302: コマンド操作で、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	コマンド操作で、未処理の例外が発生しました。説明: 'CommandActionName'
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9303: ファイルの変更操作で、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 91 • エラー -9303: ファイルの変更操作で、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	ファイルの変更操作で、未処理の例外が発生しました。説明: 'FileName'
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー –9304: レジストリの変更操作で、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 92 • エラー –9304: レジストリの変更操作で、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	レジストリの変更操作で、未処理の例外が発生しました。説明: 'RegistryName'
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー –9305: 作成操作で、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 93 • エラー –9305: 作成操作で、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	作成操作で、未処理の例外が発生しました。説明: 'CreateName'
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー –9306: 規則エンジンの実行時に、未処理の例外が発生しました

次のテーブルは、このメッセージの説明です:

テーブル 94 • エラー –9306: 規則エンジンの実行時に、未処理の例外が発生しました

分類	説明
種類:	エラー
メッセージ:	規則エンジンの実行中に未処理の例外が発生しました。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9401: App-V ライターの初期化中に、エラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 95 • エラー -9401: App-V ライターの初期化中に、エラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V ライターの初期化中に、予期しないエラーが発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9402: App-V パッケージの初期化中に、エラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 96 • エラー -9402: App-V パッケージの初期化中に、エラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V パッケージの初期化中に、予期しないエラーが発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9403: App-V ファイル エントリを書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 97 • エラー -9403: App-V ファイル エントリを書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V エントリの書き込み中に、予期しないエラーが発生しました。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9404: App-V フォルダー エントリを書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 98 • エラー -9404: App-V フォルダー エントリを書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V フォルダー エントリの書き込み中に予期しないエラーが発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9405: App-V レジストリ エントリを書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 99 • エラー -9405: App-V レジストリ エントリを書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V レジストリ エントリの書き込み中に、予期しないエラーが発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9406: App-V INI ファイル エントリを書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 100 • エラー -9406: App-V INI ファイル エントリを書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V INI ファイル エントリの書き込み中に予期しないエラーが発生しました。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9407: App-V ショートカットを書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 101 • エラー -9407: App-V ショートカットを書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V ショートカットの書き込み中に、予期しないエラーが発生しました。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9408: App-V ファイル型のデータを書き込み中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 102 • エラー -9408: App-V ファイル型のデータを書き込み中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V ファイル型のデータの書き込み中に、予期しないエラーが発生しました。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9409: App-V データの保存中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 103 • エラー -9409: App-V データの保存中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V データの保存中に、予期しないエラーが発生しました
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9410: ソース ファイル パスの判別中にエラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 104 • エラー -9410: ソース ファイル パスの判別中にエラーが発生しました

分類	説明
種類:	エラー
メッセージ:	'FileName' のソース ファイル パスの判別中に、予期しないエラーが発生しました。
原因:	ファイルのインストール場所 (ランタイム プロパティによって判別されます) が、App-V 仮想コンバーターによって判別できませんでした。
解決方法:	InstallShield でファイルを検索し、既知のディレクトリを指定します。

エラー -9411: OSD ファイルのテンプレートを抽出できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 105 • エラー -9411: OSD ファイルのテンプレートを抽出できませんでした

分類	説明
種類:	エラー
メッセージ:	Microsoft App-V OSD ファイルのテンプレートを抽出できませんでした。OSD ファイルの生成は正しく実行されません。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9412: OSD ファイルを保存できませんでした

次のテーブルは、このメッセージの説明です:

テーブル 106 • エラー -9412: OSD ファイルを保存できませんでした

分類	説明
種類:	エラー
メッセージ:	Microsoft App-V OSD ファイルを保存できませんでした。OSD ファイルの生成は正しく実行されません。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9413: App-V OSD の保存

次のテーブルは、このメッセージの説明です:

テーブル 107 • エラー -4313: App-V OSD の保存

分類	説明
種類:	エラー
メッセージ:	Microsoft App-V OSD ファイルを保存できませんでした。OSD ファイルの生成は正しく実行されません。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

警告 -9414: プライマリ アプリケーションの依存関係として指定されたローカル App-V アプリケーション

次のテーブルは、このメッセージの説明です:

テーブル 108 • 警告 -9414: プライマリ アプリケーションの依存関係として指定されたローカル App-V アプリケーション

分類	説明
種類:	警告
メッセージ:	ローカルの App-V アプリケーションがプライマリ アプリケーションの依存関係として指定されました。別の場所に移動された場合、プライマリ アプリケーションは正しく実行されなくなる可能性があります。
原因:	ユーザーによって、ローカル ドライブまたはマップされたドライブにある依存する App-V アプリケーションが指定されました。これは、依存アプリケーションの OSD ファイル内にある CODEBASE タグの HREF 属性を検証することで判別されます。
解決方法:	依存関係のアプリケーションは、非 FILE プロトコルを使用するポータブル メカニズム、または、ネットワーク URL を使って参照します。

エラー -9415: 依存関係のアプリケーションが見つかりませんでした

次のテーブルは、このメッセージの説明です:

テーブル 109 • エラー -9415: 依存関係のアプリケーションが見つかりませんでした

分類	説明
種類:	エラー
メッセージ:	依存関係のアプリケーションが見つかりませんでした: 'ApplicationName'

テーブル 109 • エラー -9415: 依存関係のアプリケーションが見つかりませんでした

分類	説明
原因:	指定された App-V の依存アプリケーション ファイルが見つかりませんでした。
解決方法:	指定された App-V の依存アプリケーションのパスを確認します。

警告 -9416: プライマリ アプリケーション ディレクトリが無効です

次のテーブルは、このメッセージの説明です:

テーブル 110 • 警告 -9416: プライマリ アプリケーション ディレクトリが無効です

分類	説明
種類:	エラー
メッセージ:	指定されたプライマリ アプリケーション ディレクトリ、'DirectoryName'、が存在しません。
原因:	これは、プライマリ アプリケーション ディレクトリが指定された後、Windows Installer パッケージで指定されたディレクトリが変更されたときに発生する可能性があります。
解決方法:	InstallShield で提供されている参照フォルダーを使って、有効なプライマリ アプリケーション ディレクトリを指定します。

エラー -9417: 依存アプリケーションの OSD ファイルに、無効な HREF の値が含まれています。

次のテーブルは、このメッセージの説明です:

テーブル 111 • エラー -9417: 依存アプリケーションの OSD ファイルに、無効な HREF の値が含まれています。

分類	説明
種類:	エラー
メッセージ:	依存アプリケーションの OSD ファイル内にある CODEBASE タグの HREF フィールドに無効な値が含まれています: 'HREF_Field_Value'
原因:	依存アプリケーションの OSD ファイル内の CODEBASE タグに、空または存在しない HREF 属性が存在する可能性があります。
解決方法:	依存アプリケーションの OSD ファイル内にある CODEBASE タグに有効な HREF 属性がエントリされていることを確認してください。

エラー -9418: サイドバイサイド アセンブリをプライベート化中にエラー

次のテーブルは、このメッセージの説明です:

テーブル 112 • エラー -9418: サイドバイサイド アセンブリをプライベート化中にエラー

分類	説明
種類:	エラー
メッセージ:	サイドバイサイド アセンブリをプライベート化中に、エラーが発生しました。
原因:	App-V パッケージを変換するとき、App-V ランタイムが win32 Sxs アセンブリ キャッシュにインストールされたファイルを見つけられるように、それらをプライベート化する必要があります。このエラーは、このプロセスで予期しない失敗が発生したとき発生します。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー -9419: ウォーターマークを挿入中に、エラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 113 • エラー -9419: ウォーターマークを挿入中に、エラーが発生しました

分類	説明
種類:	エラー
メッセージ:	App-V パッケージに評価版ウォーターマークを挿入すル際、エラーが発生しました。
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

エラー: -9424: App-V 5.x パッケージをビルドする

次のテーブルは、このメッセージの説明です:

テーブル 114 • エラー: -9424: App-V 5.x パッケージをビルドする

分類	説明
種類:	エラー
メッセージ:	Microsoft App-V 5 への変換処理をスキップしています。App-V 5 パッケージのビルドに必要な最小オペレーティング システム要件は、Windows 8 または Windows Server 2012 です。
解決方法:	InstallShield がサポートする任意の Windows バージョンで App-V 5.x パッケージの設定を構成できますが、InstallShield 内部から App-V 5.x パッケージをビルドするには Windows 8 または Windows Server 2012 が必要です。

警告 -9500: ショートカットが不足しています

次のテーブルは、このメッセージの説明です:

テーブル 115 • 警告 -9500: ショートカットが不足しています

分類	説明
種類:	警告
メッセージ:	ショートカットのターゲット 'FileName' が存在しません。ショートカットを除外します。
原因:	プロジェクトにあるショートカットのターゲット ファイルが存在しません。
解決方法:	このアプリケーションをリパッケージして、仮想パッケージを再ビルドします。

エラー -10000: ユーザーによってプロセスがキャンセルされました

次のテーブルは、このメッセージの説明です:

テーブル 116 • エラー -10000: ユーザーによってプロセスがキャンセルされました

分類	説明
種類:	エラー
メッセージ:	処理がユーザーによってキャンセルされました。
原因:	ユーザーが **Cancel** をクリックして、プロファイルの変換プロセスをキャンセルしました。

警告 -10001: スイート ファイルが不足しています

次のテーブルは、このメッセージの説明です:

テーブル 117 • 警告 -10001: スイート ファイルが不足しています

分類	説明
種類:	警告
メッセージ:	スイート MSI ファイル 'FileName' が見つからないため、このファイルは変換プロセスから除外されます。
原因:	スイートの変換の一部である MSI ファイルが見つかりませんでした。
解決方法:	スイートの変換プロセスのインプット ファイルが存在することを確認してください。

警告 -10002: スイート ファイルが重複しています

次のテーブルは、このメッセージの説明です:

テーブル 118 • 警告 -10002: スイート ファイルが重複しています

分類	説明
種類:	警告
メッセージ:	スイート MSI ファイル 'FileName' が、メインの MSI ファイルと同じであるため、このファイルは変換プロセスから除外されます。
原因:	スイートの変換が試みられたとき、Windows Installer ファイル (.msi) と指定された追加の Windows Installer ファイルの 1 つが同じであることが検出されました
解決方法:	スイートの変換プロセスの一部として、一意の Windows Installer ファイルを指定してください。

警告 -10003: アプリケーション ファイルが不足しています

次のテーブルは、このメッセージの説明です:

テーブル 119 • 警告 -10003: アプリケーション ファイルが不足しています

分類	説明
種類:	エラー
メッセージ:	アプリケーション ファイル 'ApplicationName' が見つかりません
原因:	インストールによって参照されるファイルが、App-V 仮想コンバーターによって見つかりませんでした。ファイルの参照がインストールによって壊された可能性があります。
解決方法:	InstallShield を使って、インストール内でそのファイルを見つけるか、そのリンクを修正または削除します。

警告 -10004: INI ファイルが不足しています

次のテーブルは、このメッセージの説明です:

テーブル 120 • 警告 -10004: INI ファイルが不足しています

分類	説明
種類:	エラー
メッセージ:	INI ファイル 'INI_File_Name' が見つかりません。

テーブル 120 • 警告 -10004: INI ファイルが不足しています

分類	説明
解決方法:	InstallShield テクニカル サポートをお問い合わせください。

修正 11000: Excluding TCPIP レジストリ エントリを除外しています

次のテーブルは、このメッセージの説明です:

テーブル 121 • 修正 11000: Excluding TCPIP レジストリ エントリを除外しています

分類	説明
種類:	修正
メッセージ:	TCPIP レジストリ エントリを Citrix プロファイルから除外します。
アクション:	すべての TCPIP レジストリ エントリが Citrix プロファイルから除外されます。

致命的エラー 11001: VMware が失敗しました

次のテーブルは、このメッセージの説明です:

テーブル 122 • 致命的エラー 11001: VMware が失敗しました

分類	説明
種類:	致命的
メッセージ:	VMWare は仮想化できません。
原因:	仮想化の対象のアプリケーションが VMware のとき、変換は失敗します。
アクション:	エラー メッセージ「VMware は仮想化できません」が表示されます。

警告 11003: コントロール パネル アプレット – Citrix

次のテーブルは、このメッセージの説明です:

テーブル 123 • 警告 11003: コントロール パネル アプレット – Citrix

分類	説明
種類:	警告
メッセージ:	コントロール パネル アプレット [AppletName] はコントロール パネルに表示されません。
アクション:	アプリケーションにコントロール パネルのアプレットが含まれているとき警告が表示されます。

修正 11004: コントロール パネル アプレット – ThinApp

次のテーブルは、このメッセージの説明です:

テーブル 124 • 修正 11004: コントロール パネル アプレット – ThinApp

分類	説明
種類:	修正
メッセージ:	'DirectoryPath' に保存されているコントロール パネル アプレットにショートカットを生成しています
アクション:	ThinApp のコントロール パネル アプレット用に、デフォルトのショートカットが作成されます。

致命的エラー 11005: QuickTime 7.4.1 が原因で、致命的エラーが発生しました

次のテーブルは、このメッセージの説明です:

テーブル 125 • エラー 11005: QuickTime 7.4.1 が原因で、致命的エラーが発生しました

分類	説明
種類:	致命的エラー
メッセージ:	QuickTime 7.4.1 は、仮想パッケージから実行したときエラーが発生することが知られています。代わりに QuickTime 7.4.5 を使用してください。
原因:	QuickTime 7.4.1 は、正しく仮想化できません。
解決方法:	QuickTime 7.4.5 を取得して、再度変換プロセスを実行してください。

修正 11006: Adobe Distiller によって AdobePDFSettings が除外されました

次のテーブルは、このメッセージの説明です:

テーブル 126 • 修正 11006: Adobe Distiller によって AdobePDFSettings が除外されました

分類	説明
種類:	修正
メッセージ:	レジストリ キー Software¥Adobe¥Acrobat Distiller¥AdobePDFSettings を除外しています。Adobe Distiller は、初回の使用でこれらの設定を再作成します。
アクション:	AdobePDFSettings レジストリの設定が除外されます。

修正 11007: URL ショートカットの除外

次のテーブルは、このメッセージの説明です:

テーブル 127 • 修正 11007: Adobe Distiller によって AdobePDFSettings が除外されました

分類	説明
種類:	修正
メッセージ:	.URL ファイルへのショートカットを除外します。App-V は、これらのファイルを適切に起動しません。
アクション:	.URL ファイルへのショートカットが除外されました。

テクニカル サポートにお問い合わせの前に

InstallShield のテクニカル サポートに問い合わせをする前に、以下の手順を実行して、発生している問題を明らかにしてください:

- **パッケージを確認する** – 変換中のパッケージに問題があるかどうかを判別するために、ファイルを 1 つだけ含む簡単なパッケージの Citrix プロファイルを試行でビルドします。
- **マシンと OS を確認する** – 特定のマシンまたはオペレーティング システム上にある構成に問題があるかどうかを判別するために、別のマシンまたはオペレーティング システムで Citrix プロファイルを試行でビルドします。
- **個々のファイルを確認する** – このエラーが特定のアイテムについてのみ発生するかどうかを判別するために、特定のアイテムを削除または除外して、ビルド エラーが無くなるかどうかを確認します。

変換前および後の操作が必要なアプリケーションの機能

一部のアプリケーションの機能は、Citrix プロファイルの作成時に無視されます。このため、仮想アプリケーションが動作するために、追加の変換前または変換後の操作をいくつか実行する必要があります。

次のテーブルは、手動で追加の変換プロセスが必要なアプリケーションの機能の一覧です:

テーブル 128 • プロファイルの変換時に無視されるアプリケーション機能

Windows Installer 機能	手動の変換ステップ
ユーザー定義のカスタム アクション	Windows Installer パッケージを Citrix プロファイルに変換するとき、すべてのカスタム アクションは無視されます。ユーザー定義のカスタム アクションの場合、警告メッセージが表示されます。Windows Installer パッケージのカスタム アクションが作成したターゲット マシンへの変更は、Citrix プロファイルに伝達されません。 解決方法は、カスタム アクションの目的によって異なります: • カスタム アクションが単に値の入力や他の種類の軽度の変更を自動的に行う場合、この警告は無視することができます。 • カスタム アクションの実行によってインストールの動作が変更される可能性がある場合 (Property の設定など)、この問題を解決する必要が場合によってあります。 この問題を解決するには、まず、変換済みのパッケージを Citrix XenApp 上で起動してみます。アプリケーション エラーが発生した場合、次のステップを実行して、このアプリケーションをリパッケージする必要があります。 ***ユーザー定義のカスタム アクションを使ってパッケージを正常に変換するには、以下の手順に従います:*** 1. リパッケージ ウィザード を利用して、このアプリケーションをリパッケージします。 アプリケーションのインストール中、リパッケージ ウィザードはシステム変更を監視します。データはこのあと リパッケージャー プロジェクトに変換されます。 2. リパッケージャー プロジェクトをビルドして、訂正した Windows Installer パッケージを生成します。この新しい Windows Installer パッケージにはカスタム アクションが含まれていませんが、(リパッケージ プロセスの結果として) 元のカスタム アクションによって実行された機能を含みます。
サービス	Citrix XenApp では、サービスは、どの種類もサポートされていません。このため、この問題を解決するには、必要なサービスを分離環境の外、つまりユーザーのデスクトップ マシンにインストールする必要があります。 ***サービスを使ってパッケージを正常に変換するには、以下の手順に従います:*** 1. あるアプリケーションとサービスが同一の Windows Installer パッケージにバンドルされている場合、そのサービスのみを含む別の Windows Installer パッケージを作成します。 2. そのサービスを各ユーザー デスクトップ マシンにインストールします。このアプリケーションの Citrix プロファイルを、サービスが既にインストールされているマシン上の分離環境で実行できます。

テーブル 128 • プロファイルの変換時に無視されるアプリケーション機能

Windows Installer 機能	手動の変換ステップ
COM+	Citrix XenApp では、COM+ アプリケーションとの交信がサポートされていますが、COM+ サービスの*インストール*はサポートされていません。従って、COM+ サービスを含むアプリケーションは Citrix XenApp で展開することはできません。

オートメーション インターフェイス

InstallShield ヘルプ ライブラリは、InstallShield インターフェイスを使用してインストールの全要素を作成し、リリースをビルドする方法を説明します。InstallShield では、上級開発者向けに COM インターフェイスを公開しています。これを使用して、プログラム (Visual Basic 実行可能ファイルなど) やスクリプト (Windows Scripting Host の VBScript ファイルなど) から多くの同じタスクを実行できます。オートメーション インターフェイスを通して、メソッドの呼び出し、プロパティの設定、コレクションへのアクセスを行い、InstallShield インターフェイスの場合と同様にプロジェクトを開き、その機能やコンポーネント データを変更することができます。

オートメーションインターフェイスで最高レベルのオブジェクトは ISWiProject オブジェクト です。このインターフェイスにアクセスするには、まずこのオブジェクトを作成する必要があります。次いで、そのオブジェクト、メソッド、プロパティを使用して、プロジェクトを変更できます。

メモ: *オートメーション インターフェイスは実行時ではなく設計時にプロジェクトに影響を与えます。*

基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、およびマージ モジュール プロジェクトにおけるオートメーション インターフェイス サポートについての情報は、次の章を参照してください:

- オートメーション オブジェクト
- オートメーション コレクション

アドバンスト UI およびスイート/アドバンスト UI プロジェクトにおけるオートメーション インターフェイス サポートについての情報は、次の章を参照してください:

- » アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション オブジェクト
- » アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション コレクション

オートメーション オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

各 InstallShield オートメーション オブジェクトは、機能、コンポーネント、ファイルなどのインストール要素を表します。このセクションでは、各 InstallShield オートメーション オブジェクトが説明されています。

オートメーションインターフェイスで最高レベルのオブジェクトは ISWiProject オブジェクト です。このインターフェイスにアクセスするには、まずこのオブジェクトを作成する必要があります。次いで、そのオブジェクト、メソッド、プロパティを使用して、プロジェクトを変更できます。

メモ: *オートメーション オブジェクトのセブセクションで提供されている例は、VBScript ではなく Visual Basic サンプル コードを使用しています。*

ISWiProject オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

メモ: *オートメーション オブジェクトのセブセクションで提供されている例は、VBScript ではなく Visual Basic サンプル コードを使用しています。*

ISWiProject には、InstallShield プロジェクト ファイル (.ism または .dim) を開いたり閉じたり、保存するためのインターフェイスが含まれます。InstallShield をシステムにインストールすると、次の VB や VBScript 構文を使って ISWiProject のコピーのインスタンスを作成できます。

```
Set m_ISWiProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
```

インストール プロジェクトを開くには、以下の手順で OpenProject メソッドを呼び出します。

```
' .ism ファイルへのパスのビルド
strFile = "C:¥InstallShield 2021 Projects¥Test.ism"
m_ISWiProject.OpenProject strFile
```

重要: *オートメーション インターフェイスを 64 ビット マシンで使用する場合、これを 32 ビット実行可能ファイルを使ってロードする必要があります。これを怠った場合、ISWiProject またはその他のオブジェクトのインスタンスを作成したときに、エラーが発生する可能性があります。詳細については、「64 ビット システム上でオートメーション インターフェイスを使用する」を参照してください。*

ビルドのステータスイベント

ISWiProject オブジェクトでは次のステータスイベントが発生します。

- Public Event ProgressIncrement(ByVal lIncrement As Long, pbCancel As Boolean)
- Public Event ProgressMax(ByVal lMax As Long, pbCancel As Boolean)
- Public Event StatusMessage(ByVal sMessage As String, pbCancel As Boolean)

これらのイベントは、ビルドプロセス中に発生してステータスのアップデートを提供します。ビルドを中止するには pbCancel を設定します。これらのイベントの使い方をデモンストレーションするサンプル コードは、「ビルドのステータス イベント」を参照してください。

機能プロパティと属性の多くは、このオブジェクトのメソッド、プロパティ、コレクションを通してオートメーション インターフェイスで使用できます。以下のテーブルでは、ISWiProject Object のメンバーがリストされています。いくつかのメソッド、プロパティ、コレクションは、特定のプロジェクトタイプにのみ使用できます。

メンバー

テーブル 1 • ISWiProject オブジェクトのメンバー

名前	プロジェクト	種類	説明
ActiveLanguage	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト	読み書きプロパティ	プロジェクトのデフォルト言語を取得またはは設定します。1 つの言語を言語 ID に指定します。 ISWiShortcut の DisplayName などのローカライズ可能プロパティを取得または設定すると、オートメーション インターフェイスによりデフォルトの言語が使用されます。InstallShield では、ローカライズ可能なプロパティには文字列エントリを使用しなくてはなりません。プロパティが文字列 ID を持つ場合、プロパティに入力する文字列は、デフォルト言語の文字列エントリの値となります。
AddAdvancedFile	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	メソッド	指定された名前を持つアドバンス ファイルを(オプションで)現在のプロジェクトの指定されたディスク イメージに作成します。
AddAutomaticUpgradeEntry	基本の MSI、InstallScript MSI	メソッド	新しい ISWiAutomaticUpgradeEntry 項目を作成し、そのオブジェクトのハンドルを返します。
AddChainPackage メソッド	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトに ISWiChainPackage オブジェクトを作成します。
AddComponent	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在のプロジェクトに ISWiComponent オブジェクトを作成します。
AddCustomAction	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトに ISWiCustomAction オブジェクトを作成します。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AddFeature	基本の MSI、InstallScript、InstallScript MSI	メソッド	現在のプロジェクトに ISWiFeature オブジェクトを作成します。
AddLanguage	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在のプロジェクトに言語を追加します。
AddPatchConfig	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトに ISWiPatchConfig オブジェクトを作成します。
AddPathVariable	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在のプロジェクトにパス変数を追加します。
AddProductConfig	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトに ISWiProductConfig オブジェクトを作成します。
AddProperty	基本の MSI、InstallScript MSI	メソッド	新しい ISWiProperty 項目を作成し、そのオブジェクトのハンドルを返します。
AddSetupFile	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	メソッド	指定された名前で現在のプロジェクトにサポート ファイルを作成します。

テーブル 1・ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AddSetupType	InstallScript、InstallScript オブジェクト	メソッド	指定された名前で現在のプロジェクトにセットアップの種類を作成します。
AddSqlServerConnection	基本の MSI、DIM、InstallScript、InstallScript MSI	メソッド	現在のプロジェクトに新しいサーバー接続を追加します。
AddUpgradeTableEntry	基本の MSI、InstallScript MSI	メソッド	新しい ISWiUpgradeTableEntry 項目を作成し、そのオブジェクトのハンドルを返します。
AdminExecuteSequence	基本の MSI、InstallScript MSI	読み取り専用のプロパティ	現在のプロジェクトの [管理]-[実行] シーケンスを表わす ISWiSequence コレクションを返します。
AdminUISequence	基本の MSI、InstallScript MSI	読み取り専用のプロパティ	現在のプロジェクトの [管理]-[ユーザー インターフェイス] シーケンスを表す ISWiSequence コレクションを返します。
AdvtExecuteSequence	基本の MSI、InstallScript MSI	読み取り専用のプロパティ	現在のプロジェクトの [アドバタイズ]-[実行] シーケンスを表わす ISWiSequence コレクションを返します。
AdvtUISequence	基本の MSI、InstallScript MSI	読み取り専用のプロパティ	現在のプロジェクトの [アドバタイズ]-[ユーザー インターフェイス] シーケンスを表す ISWiSequence コレクションを返します。(入力規則ICE78 では、[アドバタイズ]-[ユーザー インターフェイス] シーケンスが空になっている必要があるとしています。詳細については、「ICE」を参照してください。)
BuildPatchConfiguration	基本の MSI、InstallScript MSI	メソッド	このメソッドを呼び出すと、strPatchConfiguration 変数が認識したパッチ構成がビルドされます。このアクションは、[パッチのデザイン] ビューでパッチ構成を右クリックしてから 'パッチのビルド' を選択した場合と同じです。
BuildPCPFile	基本の MSI、InstallScript MSI	メソッド	.pcp ファイルに基づいてパッチパッケージをビルドします。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
CloseProject	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	InstallShield インストール プロジェクトを閉じます。
CompanyName	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	会社名を指定します。 ***プロジェクト:*** *基本の MSI および InstallScript MSI プロジェクトでは、このプロパティは [一般情報] ビューの ″発行元″ 設定に対応します。* *InstallScript プロジェクトでは、このプロパティは [一般情報] ビューの ″会社名″ 設定に対応します。*
CompanyPhone	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	エンド ユーザー向けのテクニカル サポート電話番号を指定します。この電話番号は、[プログラムの追加と削除] の [サポート情報] ダイアログに表示されます。 このプロパティは、[一般情報] ビューの ″サポート電話番号″ 設定に対応します。
CompanyURL	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	エンド ユーザーがテクニカル サポートを参照できる URL を指定します。この URL は、[プログラムの追加と削除] の [サポート情報] ダイアログ内でハイパーリンクとして使用されます。 このプロパティは、[一般情報] ビューの ″発行元/製品 URL″ 設定に対応します。
CreatePatch	基本の MSI、InstallScript MSI	メソッド	 ***注意 •*** *CreatePatch メソッドは最新の開発では避けられています。その代わりに BuildPCPFile および BuildPatchConfiguration メソッドが利用されます。*

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
CreateProject	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	新しいプロジェクト ファイル (.ism または .dim) を作成します。
DeleteAdvancedFile	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	メソッド	現在のプロジェクトから指定のアドバンスファイルを削除します。
DeleteChainPackage メソッド	基本の MSI、InstallScript MSI、	メソッド	現在のプロジェクトから ISWiChainPackage オブジェクトを削除します。
DeleteComponent	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	プロジェクトから指定したコンポーネントを削除します。
DeleteCustomAction	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトの ISWiCustomAction オブジェクトを削除します。
DeleteMergeModule	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトからマージ モジュールを削除します。
DeletePatchConfig	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトから ISWiPatchConfig オブジェクトを削除します。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DeletePathVariable	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在のプロジェクトからパス変数を削除します。
DeleteProductConfig	基本の MSI、InstallScript MSI	メソッド	現在のプロジェクトから ISWiProductConfig オブジェクトを削除します。
DeleteProperty	基本の MSI、InstallScript MSI	メソッド	プロパティを Property テーブルから削除します。
DeleteSetupFile	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	メソッド	現在のプロジェクトから指定のサポート ファイルを削除します。
DeleteSetupType	InstallScript、InstallScript オブジェクト	メソッド	現在のプロジェクトから指定のセットアップの種類を削除します。
DeleteSQLConnection	基本の MSI、DIM、InstallScript、InstallScript MSI	メソッド	現在のプロジェクトから SQL Server 接続を削除します。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DeselectLanguage	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在のプロジェクトから言語の選択を解除します。
DotNetFrameworkPath	基本の MSI、InstallScript MSI	読み書きプロパティ	Microsoft .NET Framework へのパスを指定します。このパラメーターはオプションです。
EnableSwidtag	基本の MSI	読み書きプロパティ	インストールのソフトウェア識別タグの作成機能を有効または無効にします。次は、タグの作成を無効にするサンプル コードです: pProject.EnableSwidtag = False ソフトウェアのタグ機能に関する情報は、「製品のソフトウェア識別タグを含める」を参照してください。
ExportProject	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	このオプションを使うと、プロジェクト ファイルは ソース コードの統合 により適した XML 表現を使用します。
ExportStrings	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	文字列エントリをテキスト ファイルにエクスポートします。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ForceUpgrade	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	プロジェクトを InstallShield の以前のバージョンから現在のバージョンにアップグレードします。
GenerateGUID	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	文字列として返される新しい一意の GUID を生成します。
ImportProject	基本の MSI	メソッド	.isv ファイル (ソース コードの統合プログラムで保持されるテキスト ファイル) を InstallShield .ism プロジェクト ファイルに変換します。 ***メモ*** *.isv ファイル形式は InstallShield for Windows Installer のバージョン 2.x と InstallShield Developer のバージョン 8.x 以前でしか使用されていません。*
ImportStrings	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	文字列エントリをテキスト ファイルからインポートします。
INSTALLDIR	基本の MSI、InstallScript MSI	読み書きプロパティ	プロジェクトに INSTALLDIR プロパティを設定します。通常の値は [ProgramFilesFolder]CompanyName¥ProductName です。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
InstallExecuteSequence	基本の MSI、InstallScript MSI	読み取り専用のプロパティ	現在のプロジェクトの [インストール]-[実行] シーケンスを表わす ISWiSequence コレクションを返します。
InstallUISequence	基本の MSI、InstallScript MSI	読み取り専用のプロパティ	現在のプロジェクトの [インストール]-[ユーザー インターフェイス] シーケンスを表す ISWiSequence コレクションを返します。
ISWiAdvancedFiles	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	コレクション	プロジェクトのすべてのアドバンスファイルを (ISWiAdvancedFile オブジェクトとして)含めます。
ISWiAutomaticUpgradeEntries	基本の MSI、InstallScript MSI	コレクション	ISWiAutomaticUpgradeEntry アイテムのコレクションを戻します。このコレクションは、IDE で定義づけられた自動アップグレードのリストを表示します。
ISWiComponents	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コレクション	このプロジェクトのすべてのコンポーネントを格納します。
ISWiCustomActions	基本の MSI、InstallScript MSI	コレクション	このプロジェクトのすべてのカスタム アクションを格納します。
ISWiFeatures	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	コレクション	すべての機能の名前とそのコンポーネントが含まれます。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ISWiLanguages	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コレクション	現在のプロジェクトに含まれるすべての言語を含みます。
ISWiPatchConfigs	基本の MSI、InstallScript MSI	コレクション	このプロジェクトのすべてのパッチ構成が含まれます。
ISWiPathVariables	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コレクション	現在のプロジェクトに関連付けられているすべてのパス変数を含みます。
ISWiProductConfigs	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コレクション	このプロジェクトのすべての製品構成が含まれます。
ISWiProperties	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コレクション	このプロジェクトのすべての Windows Installer プロパティが含まれます。
ISWiSetupFiles	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	コレクション	プロジェクトのすべてのサポート ファイルを (ISWiSetupFile オブジェクトとして)含めます。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ISWiSetupTypes	InstallScript、InstallScript オブジェクト	コレクション	プロジェクトのすべてのセットアップの種類を(ISWiSetupType オブジェクトとして)含めます。
ISWiSISProperties	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コレクション	このプロジェクトのすべての概要情報ストリームプロパティが含まれます。
ISWiSQLConnections	基本の MSI、DIM、InstallScript、InstallScript MSI	コレクション	現在のプロジェクトのすべての SQL Server 接続が含まれています。
ISWiUpgradeTableEntries	基本の MSI、InstallScript MSI	コレクション	ISWiUpgradeTableEntry アイテムのコレクションを戻します。このコレクションは、IDE で定義づけられたスタティック メジャー アップグレード アイテムのリストを表示します。
MergeModuleSearchPath	基本の MSI、InstallScript MSI	読み書きプロパティ	プロジェクトで参照させるマージ モジュール (.msm) ファイルを含むカンマで区切ったフォルダーを指定します(複数指定可)。 ***メモ:*** *ISWiRelease.Build メソッド を使う予定の場合、このプロパティを設定してください。*
MinimumTargetDotNetVersion	基本の MSI、InstallScript MSI	読み書きプロパティ	ターゲット システム上でインストールに必要な .NET Framework の最小バージョンを指定します。このパラメーターはオプションです。デフォルトでは、InstallShield インターフェイスがサポートする .NET Framework の最新バージョンが設定されています。
MinimumTargetMSIVersion	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	ターゲット システム上でインストールに必要な Windows Installer の最小バージョンを指定します。このパラメーターはオプションです。デフォルトでは、InstallShield インターフェイスがサポートする Windows Installer の最新バージョンに設定されています。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ModuleID と ModuleIDGUID	マージ モジュール	読み書き文字列プロパティ	これらのプロパティは、マージ モジュール プロジェクトでのみ使用されます。各マージ モジュールは、ModuleSignature テーブルにレコードがあります。ModuleSignature テーブルの ModuleID フィールドは 2 つのセクション (ModuleID.ModuleIDGUID.) で構成されます。 たとえば、IDID.019880FB_04F2_4D4B_99C9_9A0A12185229 の ModuleID 値について、ISWiProject オブジェクトの ModuleID および ModuleIDGUID プロパティが次の処理を有効にします: • ModuleID プロパティは、*IDID* を取得または設定します。 • ModuleIDGUID プロパティは、*019880FB_04F2_4D4B_99C9_9A0A12185229* を取得または設定します。
OnUpgrade	基本の MSI、InstallScript MSI	読み書きプロパティ	[アップグレード] ビューの [共通] タブにある "スモール/マイナー アップグレード" 設定オプションにの値を取得または設定します。このプロパティを使って、エンド ユーザーがアップグレードを続行する前にプロンプトするかどうかを指定します。この機能には Setup.exe セットアップ ランチャーが必要です。選択可能なオプションは以下のとおりです: • **eouPrompt (1)**—プロンプトする • **eouUpgrade (2)**—ユーザーに確認のプロンプトを表示せずに、アップグレードをインストールする • **eouNoOp (0)**—すべてのプロンプトを無効にするインストールはアップグレードのシナリオを検出して、インストールをアップグレード モードで実行します。
OpenProject	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	新しい InstallShield インストールプロジェクト ファイル (.ism または .dim) を開きます。 このメソッドは、InstallShield ユーザー インターフェイスでプロジェクトが開かれていると失敗します。
PackageCode	基本の MSI、InstallScript MSI	読み書きプロパティ	パッケージ コード を文字列 GUID として設定します。GUID は、{12345678-1234-1234-1234-1234567890AB} のように中括弧で囲む必要があります。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ProductCode	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	読み書きプロパティ	製品コード を文字列 GUID として取得または設定します。GUID は、{12345678-1234-1234-1234-1234567890AB} のように中括弧で囲む必要があります。 このプロパティは、マージ モジュールプロジェクトには使用できません。
ProductName	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	製品名の取得または設定を行います。
ProductVersion	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	プロジェクトのバージョン番号を文字列として格納します。
RemoveFeature	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	メソッド	プロジェクトから機能を削除します。
SaveProject	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	InstallShield インストール プロジェクトを保存します。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SelfRegistrationMethod	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	自己登録と指定されたファイルに使用する自己登録方法を指定します。有効なオプションは次のとおりです: • **esrISSelfReg (0)**—すべての自己登録ファイルに InstallShield 自己登録テーブル (ISSelfReg) を使用します。 • **esrSelfReg (1)**—すべての自己登録ファイルに Windows Installer 自己登録テーブル (SelfReg) を使用します。
SkipUpgradeValidators	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	通常ビルドの終わりで実行されるアップグレード検証ツールを、スキップすることができます。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SwidtagProperty	基本の MSI	読み書きプロパティ	[一般情報] ビューのソフトウェア識別タグ設定の値を取得または設定します。このプロパティは、最初のパラメーターで次のオプションの 1 つを受け取ります: • **eswtpUniqueId (0)** – このパラメーターは、[一般情報] ビューにある "一意な ID" 設定の値を指定します。 • **eswtpTagCreatorName (2)** – このパラメーターは、[一般情報] ビューにある "タグ作成者" 設定の値を指定します。 • **eswtpTagCreatorRegid (3)** – このパラメーターは、[一般情報] ビューにある "タグ作成者 ID" 設定の値を指定します。 • **eswtpSfwCreatorName (4)** – このパラメーターは、[一般情報] ビューにある "ソフトウェア作成者" 設定の値を指定します。 • **eswtpSfwCreatorRegid (5)** – このパラメーターは、[一般情報] ビューにある "ソフトウェア作成者 ID" 設定の値を指定します。 • **eswtpSfwLicensorName (6)** – このパラメーターは、[一般情報] ビューにある "ソフトウェア ライセンサー" 設定の値を指定します。 • **eswtpSfwLicensorRegid (7)** – このパラメーターは、[一般情報] ビューにある "ソフトウェア ライセンサー ID" 設定の値を指定します。 • **eswtpEntitlementRequired (8)** – このパラメーターは、ソフトウェアが正しく運用されていることを判断するために、ユーザーが製品に対応するソフトウェアエンタイトルメントを所有している必要があるかどうかを指定します。 次は、このプロパティを使ったサンプル コードです: pProject.SwidtagProperty(eswtpTagCreatorRegid) = "regid.1986-12.com.mycompany"
UpgradeCode	基本の MSI、InstallScript MSI	読み書きプロパティ	アップグレードコードプロパティを文字列 GUID として取得あるいは設定します。GUID は、{12345678-1234-1234-1234-1234567890AB} のように中括弧で囲む必要があります。 このプロパティは、マージ モジュールプロジェクトには使用できません。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
UseXMLProjectFormat	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージモジュール	読み書きブール値プロパティ	製品ファイルで XML ファイル形式を使用するかどうかを指定します。

AddAdvancedFile メソッド

プロジェクト: *AddAdvancedFile メソッドは、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

AddAdvancedFile メソッドは、指定された名前を持つアドバンスファイル オブジェクトを(オプションで)現在のプロジェクトの指定されたディスクイメージに作成します。

構文

AddAdvancedFile (FileName As String, Optional DiskType As ISWiDiskType = edtDisk1) As ISWiAdvancedFile

パラメーター

テーブル 2 · AddAdvancedFile メソッド パラメーター

パラメーター	説明
FileName	追加するアドバンスファイルの完全修飾ファイル名を渡します。
DiskType	オプションのパラメーターでは、アドバンスファイルの追加先となる Advanced Files フォルダーを指定できます。次の値のうち 1 つを指定します。 • edtDisk1(1)—アドバンス ファイルを Disk1 フォルダーに追加します。このオプションのパラメーターのデフォルト値として設定されています。 • edtLastDisk(-1)—アドバンス ファイルを [最後のディスク] フォルダーに追加します。 • edtOther(0)—アドバンス ファイルを [その他] フォルダーに追加します。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pAdvancedFile As ISWiAdvancedFile

Set pAdvancedFile = pProj.AddAdvancedFile ("C:¥My Files¥MyLastDiskFile.ext", edtLastDisk)

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

AddAutomaticUpgradeEntry メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddAutomaticUpgradeEntry メソッドを呼び出すと、新しい IswiAutomaticUpgradeEntry 項目が作成され、そのオブジェクトに対するハンドルが返されます。

構文

AddAutomaticUpgradeEntry()

例

次の Visual Basic 行では、このメソッドを示します。

```
'################################################
'# このサンプルは有効なパスなしには適切に実行しません
'################################################
set pProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProject.OpenProject("C:¥Documents and Settings¥Testlab¥My Documents¥MySetups¥your project name-21.ism")


'################################################
'新しいエントリを作成するには

dim pAutoEntry
set pAutoEntry = pProject.AddAutomaticUpgradeEntry()

pAutoEntry.Name ="AutoEntry"
pAutoEntry.ReleaseFlags = "AutoFlag"
pAutoEntry.UpgradedSetupPath = "AutoPath"

'################################################
'既存エントリの検索と置換をおこなうには

set pAutoEntry = nothing
set pAutoEntry = pProject.IswiAutoUpgradeEntries("AutoEntry")
pAutoEntry.Delete

pProject.SaveProject
pProject.CloseProject
```

次に適用：

- ISWiProject

AddChainPackage メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddChainPackage メソッドを呼び出して、現在のプロジェクトに連鎖 .msi パッケージを作成および追加します。

構文

AddChainPackage (ChainPackageKey As String) As ISWiChainPackage

パラメーター

テーブル 3 · AddChainPackage メソッド パラメーター

パラメーター	説明
ChainPackageKey	作成および追加したい連鎖 .msi パッケージの名前。

次に適用 :

- ISWiProject

AddComponent メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

AddComponent メソッドは、現在のプロジェクトに指定された名前でコンポーネント オブジェクトを作成します。

コンポーネントが追加されると、GUID が自動的に作成されます (ISWiComponent の GUID プロパティを参照してください)。コンポーネントを追加した後で、そのコンポーネントの他のプロパティを設定し、機能に関連付けできます。

構文

AddComponent (ComponentKey As String) As ISWiComponent

パラメーター

テーブル 4 · AddComponent メソッド パラメーター

パラメーター	説明
ComponentKey	コンポーネントキーは、[セットアップのデザイン] ビューに表示されるコンポーネントの名前と同一です。この文字列には、アルファベット、ピリオド (.) およびアンダースコア (_) を使用できますが、名前の先頭は文字またはアンダースコアにする必要があります。プロジェクトにあるコンポーネント名に、ComponentKey と同じ名前は付けられません。 コンポーネント名では、大文字と小文字が区別されます。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim pComponent As ISWiComponent
Set pComponent = m_pProject.AddComponent ("New_Component")
```

次に適用 :

- ISWiProject

AddCustomAction メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

AddCustomAction メソッドを呼び出して、現在のプロジェクトにカスタム アクションを追加します。カスタム アクションをシーケンスに挿入するには、InsertCustomAction メソッドを使用します。

構文

AddCustomAction (CustomActionKey As String) as ISWiCustomAction

パラメーター

テーブル 5 · AddCustomAction メソッド パラメーター

パラメーター	説明
CustomActionKey	作成するカスタム アクションの文字列名です。

次に適用 :

- ISWiProject

AddFeature メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

AddFeature メソッドを使うと、指定した名前で機能を作成できます。AddFeature は、ISWiProject のメンバーです。新しい機能が、現在のプロジェクトに最上位機能として追加されましたが、このメソッドが ISWiFeature オブジェクトのメンバーのとき、新しい機能は現在の機能のサブ機能として追加されます。

機能には、新機能を InstallShield に追加すると、その新機能に与えられるデフォルトと同じプロパティが作成されます。

機能を追加した後で、その機能の他のプロパティを設定し、コンポーネントと関連付けできます。

構文

AddFeature (FeatureKey As String) As ISWiFeature

パラメーター

テーブル 6 • AddFeature メソッド パラメーター

パラメーター	説明
FeatureKey	機能キーは、[セットアップのデザイン] および [機能] ビューに表示される機能の名前と同一です。その名前は、新しい ISWiFeature オブジェクトのインデックスになります。この文字列には、アルファベット、ピリオド (.) およびアンダースコア (_) を使用できますが、名前の先頭は文字またはアンダースコアにする必要があります。プロジェクトにある機能名に、FeatureKey と同じ名前は付けられません。 機能名では、大文字と小文字が区別されます。

例

次の Visual Basic コードは AddFeature の使用例で、機能とサブ機能を追加して、このサブ機能を既存のコンポーネントと関連付けています。

```
Dim pProject As ISWiProject

Set pProject = New ISWiProject
pProject.OpenProject "C:¥MySetups¥SampleApp.ism"

Dim pFeat1, pFeat2 As ISWiFeature
Dim sFeat1Name, sFeat2Name As String

sFeat1Name = "ParentFeature"
sFeat2Name = "SubFeature"

' 機能 ParentFeature を追加し、次に SubFeature を ParentFeature の下に追加
Set pFeat1 = pProject.AddFeature(sFeat1Name)
Set pFeat2 = pFeat1.AddFeature(sFeat2Name)

Dim pComp
Dim sCompName As String

' 既存コンポーネント NewComponent1 を SubFeature に追加
sCompName = "NewComponent1"
Set pComp = pProject.ISWiComponents.Item(sCompName)
pFeat2.AttachComponent pComp

pProject.SaveProject
pProject.CloseProject
```

次に適用：

- ISWiProject
- ISWiFeature

AddLanguage メソッド

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

AddLanguage メソッドを呼び出して、現在のプロジェクトに言語を追加します。このメソッドの使用方法は、InstallShield の [一般情報] ビューで言語を追加する方法に似ています。ISWiLanguage オブジェクトを使って、新しい言語のプロパティを設定できます。

構文

AddLanguage (LangId As String)

パラメーター

テーブル 7 : AddLanguage メソッド パラメーター

パラメーター	説明
LangId	現在のプロジェクトに追加する言語 ID を文字列で渡します。例: oProject.AddLanguage "1035"

次に適用 :

- ISWiProject

AddPatchConfig Method

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddPatchConfig メソッドを呼び出して、パッチ構成をプロジェクトに追加します。このメソッドは、IDE の [パッチのデザイン] ビューでパッチ構成を追加する方法に似ています。新しいパッチ構成のプロパティは、オブジェクトを使って設定することができます。

構文

AddPatchConfig (PatchConfigKey As String) As ISWiPatchConfig

パラメーター

テーブル 8 · AddPatchCofig メソッドのパラメーター

パラメーター	説明
PatchConfigKey	新しいパッチ構成の名前を保持します。

次に適用 :

- ISWiProject

AddPathVariable メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

AddPathVariable メソッドを呼び出して、現在のプロジェクトにパス変数を追加します。このメソッドの使用は、InstallShield の [パス変数] ビューでパス変数を追加するのに類似しています。新しいパス変数のプロパティは、ISWiPathVariable オブジェクトを使って設定することができます。

***重要·** 定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

構文

AddPathVariable (sName As String) As ISWiPathVariable

パラメーター

テーブル 9 ・ AddPathVariable メソッド パラメーター

パラメーター	説明
sName	パス変数の名前を指定します。

次に適用 :

- ISWiProject

AddProductConfig メソッド

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

AddProductConfig メソッドを呼び出して、製品構成をプロジェクトに追加できます。このメソッドは、リリース ウィザード か、IDE の [リリース] ビューを使って製品構成を追加するのに似ています。新しい製品構成のプロパティは、ISWiProductConfig オブジェクトを使って設定することができます。

構文

AddProductConfig (ProductConfigKey As String) As ISWiProductConfig

パラメーター

テーブル 10 ・ AddProductConfig メソッド パラメーター

パラメーター	説明
ProductConfigKey	新しい製品構成の名前を保持します。

次に適用 :

- ISWiProject

AddProperty メソッド

***プロジェクト・** AddProperty メソッドは、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

AddProperty メソッドは、現在のプロジェクトに指定された名前で Windows Installer を作成します。

構文

AddProperty (Name As String) As ISWiProperty

パラメーター

テーブル 11・AddProperty メソッド パラメーター

パラメーター	説明
名前	追加する Windows Installer プロパティの名前を渡します。

例

次の Visual Basic 行では、AddProperty メソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"

Dim pProperty As ISWiProperty
Set pProperty = pProj.AddProperty ("MYPROP")
pProperty.Value = "MyPropValue"

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiProject

AddSetupFile メソッド

プロジェクト: *AddSetupFile メソッドは、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

AddSetupFile メソッドは、現在のプロジェクトに指定された名前でサポート ファイルを作成します。

構文

AddSetupFile (FileName As String) As ISWiSetupFile

パラメーター

テーブル 12 • AddSetupFile メソッドのパラメーター

パラメーター	説明
FileName	追加するサポート ファイルの完全修飾ファイル名を渡します。

例

次の Visual Basic 行では、AddSetupFile メソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pSetupFile As ISWiSetupFile

Set pSetupFile = pProj.AddSetupFile ("C:¥My Files¥MySupportFile.ext")

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

AddSetupType メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

AddSetupType メソッドは、現在のプロジェクトに指定された名前でセットアップタイプを作成します。

構文

AddSetupType (Name As String) As ISWiSetupType

パラメーター

テーブル 13 • AddSetupType メソッド パラメーター

パラメーター	説明
名前	追加するセットアップの種類の名前を渡します。表示名として使用する文字列を指定します。 オートメーション インターフェイスは、このパラメーターで指定した名前を基に新しいセットアップの種類の有効なプライマリ キーを生成します。

例

次の Visual Basic 行では、AddSetupFile メソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pSetupType As ISWiSetupType

Set pSetupType = pProj.AddSetupType ("クライアント セットアップ – スペース節約")

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiProject

AddSQLServerConnection メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

AddSQLServerConnection メソッドを呼んで、プロジェクトに新しいサーバー接続を追加します。

構文

Function AddSQLServerConnection(strConnectionName As String) As ISWiSQLConnection

パラメーター

テーブル 14 • AddSQLServerConnection メソッドのパラメーター

パラメーター	説明
ConnectionName	確立する新しい接続の名前を指定します。

戻り値

このメソッドは ISWiSQLConnection オブジェクトを戻します。

次に適用：

- ISWiProject

AddUpgradeTableEntry メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddUpgradeTableEntry メソッドを呼び出すと、新しい ISWiUpgradeTableEntry 項目が作成され、そのオブジェクトに対するハンドルが返されます。

構文

AddUpgradeTableEntry()

例

次の Visual Basic 行では、このメソッドを示します。

```
'################################################
'# このサンプルは有効なパスなしには適切に実行しません
'################################################
set pProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProject.OpenProject("C:¥Documents and Settings¥Testlab¥My Documents¥MySetups¥your project name-21.ism")


'################################################
'新しいエントリを作成するには

dim pTableEntry
set pTableEntry = pProject.AddUpgradeTableEntry()

pTableEntry.UpgradeCode ="{12345678-1234-1234-1234-123456789012}"
```

```
pTableEntry.VersionMin = "1.0"
pTableEntry.VersionMax = "2.0"
pTableEntry.Language = "0"
pTableEntry.Remove = ""
pTableEntry.ActionProperty = "ACTIONPROP_AUTOMATION"
pTableEntry.Attributes = 5

'###############################################
'既存エントリの検索と置換をおこなうには

set pTableEntry =pProject.IswiUpgradeTableEntries("{12345678-1234-1234-1234-123456789012}")
pTableEntry.Delete

pProject.SaveProject
pProject.CloseProject
```

次に適用：

- ISWiProject

BuildPatchConfiguration メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

BuildPatchConfiguration を呼び出して、strPatchConfiguration パラメーターによって識別されたパッチ設定をビルドします。このアクションは、[パッチのデザイン] ビューでパッチ構成を右クリックしてから 'パッチのビルド' を選択した場合と同じです。

構文

BuildPatchConfiguration (strPatchConfigName as string)

パラメーター

テーブル 15 • BuildPatchConfiguration メソッド パラメーター

パラメーター	説明
strPatchConfigName	プロジェクトへの完全修飾パスを入力します。

Properties

テーブル 16 • BuildPatchConfiguration メソッドに関連づけられたプロパティ

プロパティ	説明
PatchConfigErrorCount	特定のビルドに関連するエラー カウントを含みます。
PatchConfigWarningCount	特定のビルドに関連する警告カウントを含みます。

例

次の Visual Basic 行では、このメソッドを示します。

```
'プロジェクトオブジェクトの作成
dim pProject
set pProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")


'OpenProject メソッドの呼び出し
'########################################################################
'# このサンプルは有効なパスなしには適切に実行しません
'########################################################################
'pProject.OpenProject("¥Your project.ism")

pProject.OpenProject("C:¥Documents and Settings¥Testlab¥My Documents¥MySetups¥your project name-21.ism")

'OpenProject メソッドの呼び出し
'###############################################################################
'# このサンプルは有効なパッチ構成名なしには適切に実行しません
'###############################################################################

pProject.BuildPatchConfiguration "Config1"

MsgBox "Patch Congifuration Config1 build with " & _
    pProject.PatchConfigErrorCount & " Errors and " & _
    pProject.PatchConfigWarningCount & "警告"
```

次に適用 :

- ISWiProject

BuildPCPFile メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

BuildPCPFile メソッドでは、パッチ作成プロパティ (.pcp) ファイルの設定に基づいて、パッチパッケージ (.msp) ファイルが作成されます。

パッチを作成するには、プロジェクトが開いている必要はありません。

構文

BuildPCPFile (strPCPFile As String, strPatchFile As String, strLogFile As String, bCreateUpdateExe As Boolean) As Long

パラメーター

テーブル 17 : BuildPCPFile メソッド パラメーター

パラメーター	説明
strPCPFile	.pcp ファイルの完全なパスを指定します。[パッチのデザイン] ビューでは、.pcp ファイルの作成や変更を行うことができます。
strPatchFile	作成する .msp ファイルの完全なパスを指定します。同じ名前のファイルは、いずれも上書きされます。指定したディレクトリが存在しない場合、ファイルは作成されません。
strLogFile	このメソッドの状態やパッチパッケージ作成の成果を記録する、.log ファイルの完全なパスを指定します。ファイルが既に存在する場合、そのファイルに現在の状態が追加されます。
bCreateUpdateExe	このオプションを True に設定して、パッチを適用する実行可能ファイルでパッケージをラップします。

戻り値

BuildPCPFile() は、Windows Installer 関数 UiCreatePatchPackage が戻した結果を戻します。

BuildPCPFile メソッドでは、パッチ作成プロパティ (.pcp) ファイルの設定に基づいて、パッチパッケージ (.msp) ファイルが作成されます。

パッチを作成するには、プロジェクトが開いている必要はありません。

次に適用 :

- ISWiProject

CloseProject メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

OpenProject で開いたプロジェクト ファイル (.ism または .dim) を閉じるには、CloseProject を呼び出します。

構文

CloseProject ()

戻り値

CloseProject は常に 0 を返します。

次に適用：

- ISWiProject

CreatePatch メソッド

注意： *このメソッドは新しい開発には使用不可能となりました。新しい開発には、BuildPCPFile メソッドと BuildPatchConfiguration メソッドが使用されます。既存のビルドスクリプトとの下位互換性を容易にするため、CreatePatch メソッドは BuildPCPFile メソッドを呼び出します。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

構文

CreatePatch()

次に適用：

- ISWiProject

CreateProject メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

CreateProject メソッドは、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、またはマージ モジュール プロジェクトで新しい InstallShield プロジェクト ファイル (.ism) を作成します。DIM プロジェクトには、新しい InstallShield DIM プロジェクト ファイル (.dim) を作成します。

構文

CreateProject (strISWiProjectFile As String, ByVal projectType As ISWiProjectType)

パラメーター

テーブル 18 • CreateProject メソッド パラメーター

パラメーター	説明
strISWiProjectFile	作成する .ism または .dim ファイルへの完全修飾パスを入力します。完全なファイル名を含みます。
projectType	作成するプロジェクトの種類を指定します。利用可能な値は、以下のとおりです。 • **eptMsi (1)**—基本の MSI プロジェクトの作成。 • **eptScript (-1)**—InstallScript MSI プロジェクトの作成。 • **eptPro (9)**—InstallScript プロジェクトの作成。 • **eptProObj (10)**—InstallScript オブジェクト プロジェクトの作成。 • **eptMsm (2)**—マージ モジュール プロジェクトの作成。 • **eptDim (15)**—DIM プロジェクトの作成。

例

次のサンプル コードは、TestProject.ism と名付けられた基本の MSI プロジェクトを作成します。

```
Dim pProject As ISWiAutoAutomation Interface Version.ISWiProject
Set pProject = CreateObject("ISWiAutoAutomation Interface Version.ISWiProject")

Dim sProjectName As String
sProjectName = "C:¥InstallShield 2021 Projects¥TestProject.ism"
```

```
pProject.CreateProject sProjectName, eptMsi

pProject.OpenProject sProjectName, False

pProject.CloseProject
```

次に適用：

- ISWiProject

DeleteAdvancedFile メソッド

プロジェクト：この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

DeleteAdvancedFile メソッドは、現在のプロジェクトから指定の アドバンスファイルを削除します。

構文

DeleteAdvancedFile (pAdvancedFile As ISWiAdvancedFile) As Long

パラメーター

テーブル 19 • DeleteAdvancedFile メソッド パラメーター

パラメーター	説明
pAdvancedFile	削除するアドバンス ファイル オブジェクトを指定します。

例

次の Visual Basic 行では、DeleteAdvancedFile メソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pAdvancedFile As ISWiAdvancedFile

Set pAdvancedFile = pProj.ISWiAdvancedFiles.Item(1)
pProj.DeleteAdvancedFile pAdvancedFile

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiProject

DeleteChainPackage メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

DeleteChainPackage メソッドを呼び出して、現在のプロジェクトから連鎖 .msi パッケージを削除します。

構文

DeleteChainPackage (ChainPackage As ISWiChainPackage)

パラメーター

テーブル 20 - DeleteChainPackage メソッド パラメーター

パラメーター	説明
ChainPackage	現在のプロジェクトから削除する ISWiChainPackage オブジェクト オブジェクトを渡します。

次に適用：

- ISWiProject

DeleteComponent メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

DeleteComponent を呼び出すと、プロジェクト内のコンポーネントを削除できます。[セットアップのデザイン] ビューにあるコンポーネントを削除するのと同様、DeleteComponent も共に機能からコンポーネントを削除し、永久にプロジェクトから削除します。

構文

DeleteComponent (Component As ISWiComponent)

パラメーター

テーブル 21 • DeleteComponent メソッド パラメーター

パラメーター	説明
コンポーネント	メソッドが削除する ISWiComponent オブジェクトを指定します。

例

以下の行を使用すると、機能 FeatureName1 と関連のあるすべてのコンポーネントを削除できます:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"

Dim pFeat As ISWiFeature
Dim pComp As ISWiComponent

Set pFeat = pProj.ISWiFeatures.Item("FeatureName1")

For Each pComp In pFeat.ISWiComponents
    pProj.DeleteComponent pComp
Next

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

DeleteCustomAction メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

DeleteCustomAction メソッドを呼び出して、現在のプロジェクトからカスタム アクションを完全に削除できます。

構文

DeleteCustomAction (CustomAction As ISWiCustomAction)

パラメーター

テーブル 22 • DeleteCustomAction メソッド パラメーター

パラメーター	説明
CustomAction	現在のプロジェクトから削除する ISWiCustomAction オブジェクトを渡します。

次に適用：

- ISWiProject

DeleteMergeModule メソッド

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

DeleteMergeModule メソッドを呼び出して、現在のプロジェクトからマージ モジュールを完全に削除できます。

構文

DeleteMergeModule (FileName As String)

パラメーター

テーブル 23 • DeleteMergeModule メソッド パラメーター

パラメーター	説明
FileName	プロジェクトから削除するマージ モジュール (.msm) ファイルの完全なファイル名を渡します。

次に適用：

- ISWiProject オブジェクト

DeletePatchConfig メソッド

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

DeletePatchConfig メソッドを呼び出して、プロジェクトからパッチ構成を削除できます。このメソッドは、IDE の [パッチのデザイン] ビューでパッチ構成を削除する方法に似ています。

構文

DeletePatchConfig (PatchConfig As ISWiPatchConfig)

パラメーター

テーブル 24 · DeletePatchConfig メソッドのパラメーター

パラメーター	説明
PatchConfigKey	削除するパッチ構成の名前を渡します。

次に適用 :

- ISWiProject

DeletePathVariable メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

DeletePathVariable メソッドを呼び出して、プロジェクトからパス変数を削除します。このメソッドの使用は、InstallShield の [パス変数] ビューからパス変数を削除するのに類似しています。

***重要·** 定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

構文

DeletePathVariable (pPathVar ISWiPathVariable) As Long

パラメーター

テーブル 25 • DeletePathVariable メソッド パラメーター

パラメーター	説明
pPathVar	削除するパス変数の ISWiPathVariable オブジェクトを渡します。

次に適用 :

- ISWiProject

DeleteProductConfig メソッド

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*
- *マージ モジュール*

DeleteProductConfig メソッドを呼び出して、プロジェクトから製品構成を削除できます。このメソッドは、IDE の [リリース] ビューで製品構成を削除する方法に似ています。

構文

DeleteProductConfig (ProductConfig As ISWiProductConfig)

パラメーター

テーブル 26 • DeleteProductConfig メソッド パラメーター

パラメーター	説明
ProductConfigKey	削除する製品構成の名前を渡します。

次に適用 :

- ISWiProject

DeleteProperty メソッド

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

DeleteProperty メソッドを呼び出して、Property テーブルからプロパティを削除します。成功すると関数は 0 を返します。

構文

Public Function DeleteProperty(ByVal pProperty As ISWiProperty) As Long

パラメーター

テーブル 27 • DeleteProperty メソッド パラメーター

パラメーター	説明
プロパティ	削除するプロパティの名前を渡します。

例

```
Dim proj As IswiAutoAutomation Interface Version.ISWiProject
Set proj = New IswiAutoAutomation Interface Version.ISWiProject
proj.OpenProject "c:¥mysetups¥your project name-3.ism"
Dim pProp As IswiAutoAutomation Interface Version.ISWiProperty
Set pProp = proj.ISWiProperties.Item("MyProperty")
proj.DeleteProperty pProp
```

次に適用：

- ISWiProject

DeleteSetupFile メソッド

プロジェクト: *DeleteSetupFile メソッドは、次のプロジェクトの種類に適用します。*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

DeleteSetupFile メソッドは、現在のプロジェクトから指定の サポート ファイルの選択を解除します。

構文

DeleteSetupFile (pSetupFile As ISWiSetupFile) As Long

パラメーター

テーブル 28 • DeleteSetupFile メソッド パラメーター

パラメーター	説明
pSetupFile	削除するサポート ファイルオブジェクト

例

次の Visual Basic 行では、DeleteSetupFile メソッドを示します:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pSetupFile As ISWiSetupFile

Set pSetupFile = pProj.ISWiSetupFiles.Item("MySupportFile.abc")
pProj.DeleteSetupFile pSetupFile

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

DeleteSetupType メソッド

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

DeleteSetupType メソッドは、現在のプロジェクトから指定の セットアップタイプの選択を解除します。

構文

DeleteSetupType (pSetupType As ISWiSetupType) As Long

パラメーター

テーブル 29 • DeleteSetupType メソッド パラメーター

パラメーター	説明
pSetupType	削除するセットアップの種類オブジェクト

例

次の Visual Basic 行では、DeleteSetupType メソッドを示します:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pSetupType As ISWiSetupType

Set pSetupType = pProj.ISWiSetupTypes.Item(1)
pProj.DeleteSetupType pSetupType

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

DeleteSQLConnection メソッド

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

DeleteSQLConnection メソッドを呼んで、SQL Server 接続をプロジェクトから削除します。

構文

Function DeleteSQLConnection(Connection As ISWiSQLConnection) As Long

パラメーター

テーブル 30 ・ DeleteSQLConnection メソッド パラメーター

パラメーター	説明
Connection	プロジェクトから削除したい接続を指定します。

次に適用：

- ISWiProject

DeselectLanguage メソッド

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

DeselectLanguage メソッドを呼び出して、現在のプロジェクトから言語を選択解除できます。

構文

DeselectLanguage (LangId As String)

パラメーター

テーブル 31 ・ DeselectLanguage メソッド パラメーター

パラメーター	説明
LangId	oProject.DeleteLanguage"1036" のように、文字列で選択解除する言語 ID を渡します。

次に適用：

- ISWiProject

ExportProject メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ExportProject を呼び出して、ソースコード管理プログラムでチェックする前に .ism ファイルを XML ファイルに変換します。

*メモ: プロジェクトファイルフォーマットを XML またはバイナリ形式に変換しても、プロジェクトファイルの拡張子は *.ism のまま変更されません。*

.isv ファイル形式は InstallShield for Windows Installer のバージョン 2.x と InstallShield Developer のバージョン 8.x 以前でしか使用されていません。

構文

ExportProject (strISWiTextProjectFile As String, Optional strISWiProjectFile As String) As Long

パラメーター

テーブル 32 : ExportProject メソッド パラメーター

パラメーター	説明
strISWiTextProjectFile	.isv ファイルへの完全なパスを入力します。
strISWiProjectFile	.ism ファイルへの完全なパスを入力します。このパラメーターは、.ism ファイルが既に開いている場合のみ使用できるオプションです。

次に適用 :

- ISWiProject

ExportStrings メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ExportStrings を呼び出して、特定言語の文字列エントリの全部または一部をタブ区切りの Unicode テキストファイル (.txt) にコピーします。この .txt ファイルを、翻訳されたテキストで更新することができる翻訳者に渡すことができます。ファイルが翻訳された後、ImportStrings メソッド を呼び出して、翻訳済み文字列エントリをプロジェクトにインポートできます。

構文

ExportStrings (strfile As String, dTimeStamp As Date, sLang As String) As Boolean

パラメーター

テーブル 33 ・ ExportStrings メソッドのパラメーター

パラメーター	説明
strfile	エクスポートされた文字列を入れる、テキストファイルの完全なパスを入力します。存在する場合、オートメーション インターフェイスがそれを上書きします。 ExportStrings が Unicode ファイルを作成します。
dTimeStamp	文字列エントリには、最後に更新された日付と時刻が含まれます。このタイムスタンプは、新しい文字列エントリ、または最後に翻訳を行った日付以降に変更された文字列エントリのみを翻訳者に渡したい場合に便利です。 特定の日付以降に更新された文字列エントリのみをエクスポートするには、エクスポートする最も古い文字列エントリの日付を指定します。オートメーション インターフェイスは、指定された日付、またはそれ以降に変更された文字列エントリをエクスポートします。 指定された言語のすべてのエントリをエクスポートするには、数字 0 を入力します。
sLang	テキスト ファイルにエクスポートする文字列エントリを含む言語の ID を指定します。

次に適用 :

- ISWiProject

ForceUpgrade メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ForceUpgrade メソッドは、プロジェクトを InstallShield の以前のバージョンから現在のバージョンにアップグレードします。

構文

ForceUpgrade (ISWiProjectFile As String)

パラメーター

テーブル 34 ・ ForceUpgrade メソッドのパラメーター

パラメーター	説明
ISWiProjectFile	アップグレードするプロジェクト ファイル (.ism) の完全修飾パスを入力します。完全なファイル名を含みます。

次に適用 :

- ISWiProject

GenerateGUID メソッド

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

GenerateGUID メソッドは、文字列として返される新しい一意の GUID を生成します。

次に適用 :

- ISWiProject

ImportProject メソッド

メモ: *このメソッドは InstallShield for Windows Installer のバージョン 2.x と InstallShield Developer のバージョン 8.x 以前でしか使用されていません。*

ImportProject を呼び出して、.isv ファイルを .ism ファイルに変換できます。ソース コード管理プログラムからプロジェクトにアクセスすると、InstallShield がそれをテキスト (.isv) ファイルとして保存します。ただし、OpenProject および ISCmdBld.exe の両方で、標準の InstallShield プロジェクト (.ism) ファイルが必要になります。

後で、ExportProject を呼び出して、ソース管理プログラムにチェックインする前に、.ism ファイルを .isv ファイルに変換します。

ImportProject は、プロジェクトがオートメーション インターフェイスまたは InstallShield で既に開かれている場合、失敗します。

構文

ImportProject (strISWiProjectFile As String, strISWiTextProjectFile As String) As Long

パラメーター

テーブル 35 : ImportProject メソッド パラメーター

パラメーター	説明
strISWiProjectFile	結果の .ism ファイルへの完全なパスを入力します。
strISWiTextProjectFile	.isv ファイルへの完全なパスを入力します。

Error Codes

テーブル 36 : ImportProject メソッドのエラー コード

コード	説明
1007	プロジェクト ファイルは既に開かれています。
1009	プロジェクトをインポートできませんでした。
1012	.isv ファイルがありません。

次に適用 :

- ISWiProject

ImportStrings メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ImportStrings を呼び出して、タブ区切りテキスト ファイル (.txt) の内容をプロジェクトの特定の言語にインポートできます。このメソッドは、エントリをテキストファイルにエクスポートし、それらを翻訳する際に役立ちます。

構文

ImportStrings (strfile As String, sLang As Integer, Optional Enum eiType, Optional strLogFile As String) As Boolean

パラメーター

テーブル 37 • ImportStrings メソッド パラメーター

パラメーター	説明
strfile	インポートする文字列エントリを含むテキスト ファイルへの完全修飾パスを入力します。
sLang	テキスト ファイルにエクスポートする文字列エントリを含む言語の ID を指定します。
eiType	文字列エントリのインポート中に競合を処理する方法について指定します: • eiIgnore (0)—文字列エントリの値が、プロジェクト内の指定した言語に既に含まれている場合は、置換しない。 • eiOverwrite (1)—プロジェクトの既存の文字列エントリを、テキスト ファイルの文字列エントリと置換します。このオプションのパラメーターのデフォルト値として設定されています。
strLogFile	文字列エントリの状態やインポート結果を記録する log ファイルの完全修飾パスを指定します。log ファイルには、文字列エントリをインポートする際の競合や、不適切にフォーマットされて使用できないエントリの一覧が記録されます。ログファイルが既に存在する場合、上書きされます。 ***メモ:*** *このパラメーターはオプションです。*

次に適用 :

- ISWiProject

OpenProject メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクトを開いてオートメーション インターフェイスを使って変更するには、.ism または .dim ファイルへの完全修飾パスを使って OpenProject を呼び出します。このメソッドは、InstallShield にプロジェクトをロードするのと同じです。

メモ: *ISWiProject.OpenProject メソッドは、InstallShield プロジェクトが開かれていると失敗します。*

構文

OpenProject (strISWiProjectFile As String, Optional ByVal bReadOnly As Boolean) As Long

パラメーター

テーブル 38 • OpenProject メソッドのパラメーター

パラメーター	説明
strISWiProjectFile	プロジェクト ファイル (.ism または .dim) への完全修飾パスを入力します。
bReadOnly	書き込み権限を使ってプロジェクトを開くかどうかを指定します。True を指定すると、プロジェクトは読み取り専用ファイルとして開かれます。このオプションのパラメーター値は、デフォルトで False に設定されています。

戻り値とエラー

テーブル 39 • OpenProject メソッドの戻り値とエラー

戻り値	説明
0	プロジェクトが正常に開かれました
1	プロジェクトは開きましたが、他のプロセスによりロックされています。
2	プロジェクトを開いて書き込みアクセス権を得ようとしましたが、プロジェクトは読み取り専用モードで開かれました。
3	マージ モジュール カタログの作成中に問題が起きました。([オプション] ダイアログ ボックスにある [ファイルの場所] タブまたは Standalone Build に -o を使って) マージ モジュール検索パスが正しいことを確認してください。また、マージ モジュール検索パスにあるマージ モジュールが破損していないこと、および自己登録 IsMMUpdater2.dll ではないことも確認してください。
エラーコード 1100	ファイルを開けません: %1.ファイルが有効な InstallShield プロジェクトであることを確認してください。
エラーコード 1101	ファイルを開けません: %1.プロジェクトは以前のバージョンの InstallShield Developer、InstallShield – Windows Installer または InstallShield Express を使って作成されました。
エラーコード 1102	ファイルを開けません: %1.プロジェクトは InstallShield Professional または InstallShield Express 2.xx を使って作成されました。
エラーコード 1103	ファイルを開けません: %1.プロジェクトは InstallShield Developer のより最新バージョンで作成されたか、プロジェクトが InstallShield、InstallShield DevStudio または InstallShield Developer のより最新バージョンに更新されています。

次に適用：

- ISWiProject

RemoveFeature メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

RemoveFeature メソッドを使用すると、現在のプロジェクトから完全に機能を削除できます。

構文

RemoveFeature (Feature As ISWiFeature) As Long

パラメーター

テーブル 40 • RemoveFeature メソッド パラメーター

パラメーター	説明
機能	ISWiFeatures の中で削除する項目。

例

以下のコードでは、プロジェクトを開いて NewFeature1 という名前の機能を削除してから、プロジェクトを保存して閉じます:

```
Dim pProj As ISWiProject
Dim pFeat As ISWiFeature

Set pProj = New ISWiProject
pProj.OpenProject "C:¥Test¥SampleProject.ism"

Set pFeat = pProj.ISWiFeatures.Item("NewFeature1")
pProj.RemoveFeature pFeat

pProj.SaveProject
pProj.CloseProject
```

RemoveFeature が ISWiFeature のメンバーの場合は、このメソッドを使用して現在の機能を削除することはできません。ただし、次の例のように、RemoveFeature を呼び出して NewFeature1 の下の最初のサブ機能を削除する場合など、サブ機能を削除することは可能です。

```
Set pFeat = pProj.ISWiFeatures.Item("NewFeature1")
pProj.RemoveFeature pFeat.Features(1)
```

次に適用：

- ISWiProject
- ISWiFeature

SaveProject メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト ファイルに加えた任意の変更をオートメーションインターフェイスで保存するには、SaveProject を呼び出します。

構文

SaveProject ()

戻り値とエラー

テーブル 41 ・ SaveProject メソッドの値

戻り値	説明
ERROR_SUCCESS (0)	プロジェクトが保存されました。
エラーコード 1104	ファイルを保存できません: %1プロジェクトが読取専用で開かれているか、もしくは別の処理で使用中です。

次に適用：

- ISWiProject

ISWiAdvancedFile オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ISWiAdvancedFile は、プロジェクト内のアドバンスト ファイルを表します。アドバンス ファイルは InstallShield の [サポート ファイル] ビュー (または [サポート ファイル/ビルボード] ビュー) の既存ファイルでも、AddAdvancedFile を呼び出して追加したファイルでもかまいません。

プロジェクトの ISWiAdvancedFiles コレクションを使用して、特定のアドバンス ファイルにアクセスします。ISWiAdvancedFiles の Item プロパティには、インデックス番号かファイルキーのいずれかを指定します。

メンバー

テーブル 42 • ISWiAdvancedFile オブジェクトのメンバー

名前	種類	説明
名前	読み取り専用プロパティ	アドバンスファイルのファイルキーを格納します。ファイルキーの詳細については、上記の ISWiFile の説明を参照してください。
BuildSourcePath	読み書きプロパティ	アドバンスファイルの完全修飾ソースファイル名を格納します。あらゆるパス変数は、実際のパスへ解決されます。
Disk	読み書きプロパティ	列挙された整数プロパティでは、アドバンスファイルの追加先となる Advanced Files フォルダーを指定できます。次の値のうち 1 つを指定します。 • **edtDisk1 (1)**—Disk 1 フォルダーを指定します。 • **edtLastDisk (-1)**—最後のディスクフォルダーを指定します。 • **edtOther (0)**—その他のフォルダーを指定します。

例

ファイル キーは、[ダイレクト エディター] ビューの ISDisk1 テーブルの最初のエントリとしてのみ IDE で表示されます。ファイル キーの代わりに、次の例のようなコードを記述して、特定のファイル名の存在を確認することができます。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pAdvancedFile As ISWiAdvancedFile

For Each pAdvancedFile In pProj.ISWiAdvancedFiles
    If UCase(pAdvancedFile.BuildSourcePath) = "C:¥MY FILES¥MYADVANCEDFILE.EXT" Then
        Exit For
    End If
Next

pProj.SaveProject
pProj.CloseProject
```

ファイル C:¥My Files¥MyAdvancedFile.ext がプロジェクトのセットアップ ファイル中に見つかると For ループが終了しますが、この時 pAdvancedFile には C:¥My Files¥MyAdvancedFile.ext オブジェクトがコピーされています。

次に適用：

- ISWiProject

ISWiAutomaticUpgradeEntry オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiAutomaticUpgradeEntry は、InstallShield インターフェイスの [アップグレード] ビューの自動アップグレード アイテムを表します。

メンバー

テーブル 43 ・ ISWiAutomaticUpgradeEntry オブジェクトのメンバー

名前	種類	説明
Delete	メソッド	プロジェクトファイルから AutomaticUpgradeEntry を削除します。
名前	読み書きプロパティ	IDE で表示されるアップグレードの名前を設定します。
UpgradedSetupPath	読み書きプロパティ	アップグレードする MSI パッケージへのパスを設定します。ビルドエンジンは、このセットアップをアップグレードするのに必要な設定を行います。
ReleaseFlag	読み書きプロパティ	特定のリリース フラグが設定されているときだけセットアップにビルドされるよう、このアップグレード エントリを構成します。

次に適用：

- ISWiProject

Delete メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

プロジェクト ファイルから ISWiAutomaticUpgradeEntry を削除するには、このメソッドを呼び出します。

構文

Delete()

次に適用：

- ISWiAutomaticUpgradeEntry

ISWiChainPackageData オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiChainPackageData は、InstallShield の [リリース] ビューに含まれる連鎖 .msi パッケージに関連する一連のデータを意味します。ISWiChainPackageDatas コレクション でアイテムを特定して、連鎖 .msi パッケージに関連するデータを取り込むことができます。

メンバー

テーブル 44 • ISWiChainPackageData オブジェクト メソッド

プロパティ	種類	説明
FilePath	読み取り専用プロパティ	連鎖パッケージ データからファイルまたはフォルダの名前を取得します。
ISBuildSourcePath	読み取り専用プロパティ	連鎖パッケージ データからファイルまたはフォルダのパスおよびファイル名を取得します。
オプション	読み書きプロパティ	インストール後にストリームされたファイルを削除するオプションを取得または設定します。選択可能なオプション: • ecpoLeaveBehindOnSuccess (0)—この値を使って、ストリームされたファイルの削除を有効化します。 • ecpoLeaveBehindOnSuccess (1)—この値を使って、ストリームされたファイルの削除を無効化します。 既定では、オプション プロパティが ecpoLeaveBehindOnSuccess (0) に設定されています。
パッケージ	読み取り専用プロパティ	パッケージの一意キーを取得します。

次に適用：

- ISWiChainPackage オブジェクト

ISWiChainPackage オブジェクト

プロジェクト: *この情報は、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiChainPackageData は、InstallShield の［リリース］ビューに含まれる連鎖 .msi パッケージを意味します。ISWiChainPackages コレクション でアイテムを特定して、連鎖 .msi パッケージを取得することができます。

メンバー

テーブル 45 · ISWiChainPackage オブジェクト メンバー

プロパティ	種類	説明
AddChainPackageData	メソッド	ストリームされたファイルにファイルまたはフォルダー全体を追加します。
DeleteChainPackageData	メソッド	ストリームされたファイルの一覧からファイルまたはフォルダーのエントリを削除します。
DisplayName	読み書きプロパティ	連鎖 .msi パッケージの名前を取得または設定します。
InstallCondition	読み書きプロパティ	連鎖 .msi パッケージに使用するインストール条件を取得または設定します。 実行時に条件が True と評価されたとき、連鎖パッケージの製品が既にインストールされていない場合のみ、連鎖 .msi パッケージが Windows Installer によって起動されます。
InstallProperties	読み書きプロパティ	インストール中にメイン インストールから連鎖 .msi パッケージに 1 つまたは複数のプロパティを渡す場合、それらのプロパティおよび対応する値を取得または設定します。
ISReleaseFlags	読み書きプロパティ	特定のビルドにのみ含める連鎖 .msi パッケージに、1 つまたは複数のリリース フラグを割り当てます。

テーブル 45 • ISWiChainPackage オブジェクト メンバー (続き.)

プロパティ	種類	説明
オプション	読み書きプロパティ	連鎖 .msi パッケージに使用するユーザー インターフェイス (UI) レベルを取得または設定します。選択可能なオプションは以下のとおりです: • ecoUIBasic (0)—このオプションを使って、組み込み進行状況ダイアログ(小) を表示します。 • ecoUINone (1)—このオプションを使って、インストールをサイレント モードで実行します。 • ecoUIReduced (2) — フルサイズの進行状況ダイアログなど、モードレス ダイアログのみ表示します。 • ecoUIFull (3) — .msi パッケージで提供されているモーダルとモードレス ダイアログを表示します。
パッケージ	メソッド	連鎖 .msi パッケージに追加する .msi パッケージのパスとファイル名を取得または設定します。
ProductCode	読み書きプロパティ	製品コードの 文字列 GUID を取得または設定します。 ***メモ•*** *パッケージ プロパティは、自動的に製品 コードに追加されます。*
RemoveCondition	読み書きプロパティ	連鎖 .msi パッケージに使用するアンインストール条件を取得または設定します。InstallCondition が False に評価され、RemoveCondition が True に評価された場合、連鎖されたパッケージの製品が (存在する場合) Windows Installer によって削除されます。
RemoveProperties	読み書きプロパティ	アンインストール中にメイン インストールから連鎖 .msi パッケージに 1 つまたは複数のプロパティを渡す場合、それらのプロパティおよび対応する値を取得または設定します。
SourcePath	読み取り専用のプロパティ	Windows Installerパッケージ (.msi ファイル) の名前を取得します。

次に適用 :

- ISWiProject

AddChainPackageData メソッド

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddChainPackageData メソッドを呼び出して、ストリームされたファイル一覧にファイルまたはフォルダー全体を追加します。

構文

AddChainPackageData (ISBuildSourcePath As String) As ISWiChainPackageData

パラメーター

テーブル 46 - AddChainPackageData メソッド

パラメーター	説明
ISBuildSourcePath	ストリームされたファイル一覧に追加するファイルまたはフォルダーのパスを渡します。

次に適用：

- ISWiChainPackage オブジェクト

DeleteChainPackageData メソッド

***プロジェクト-** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

DeleteChainPackageData メソッドを呼び出して、ストリームされたファイル一覧からファイルまたはフォルダー全体を削除します。

構文

DeleteChainPackageData (ChainPackageData As String) As ISWiChainPackageData

パラメーター

テーブル 47 - DeleteChainPackageData メソッド

パラメーター	説明
ChainPackageData	ストリームされたファイル一覧から削除したい ISWiChainPackageData オブジェクトを押します。

次に適用：

- ISWiChainPackage オブジェクト

ISWiCustomAction オブジェクト

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiCustomAction は、InstallShield の [カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) 内のカスタム アクションを示します。ISWiCustomActions コレクションの項目を指定して、カスタム アクションを取得できます。

メンバー

テーブル 48 • ISWiCustomAction オブジェクトのメンバー

プロパティ	種類	説明
ActionType	読み書きプロパティ	数値カスタム アクションの種類を保持します。詳細については、「カスタム アクションの種類についての概要」を参照してください。
コメント	読み書きプロパティ	カスタム アクションのコメント (オプション)。
名前	読み書きプロパティ	カスタム アクションの名前。
ソース	読み書きプロパティ	カスタム アクションのソースを指定します。異なるタイプのカスタム アクションに必要なソースプロパティについては、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する を参照してください。
SourceEx	書き込み専用のプロパティ	カスタム アクションのソースプロパティを設定します。Source プロパティと同じです。SourceEx は、Binary テーブルに格納されているソースを変更するときに新しいパスに使う既存のバイナリ レコードを置き換えます。その他のカスタム アクションでパスが使用されると、そのパスも更新されます。Source は、他のカスタム アクションがこのパスを使用する場合に新しいバイナリ レコードを作成します。カスタム アクションのパスだけが更新されます。
ターゲット	読み書きプロパティ	カスタム アクションのターゲットを指定します。異なるタイプのカスタム アクションに必要なターゲット プロパティについては、[カスタム アクションとシーケンス] ビュー (または、[カスタム アクション] ビュー) でカスタム アクションを作成する を参照してください。

次に適用 :

- ISWiProject

ISWiComponent オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiComponent は、InstallShield ユーザー インターフェイスにあるコンポーネントを表します。ISWiComponents コレクションにある項目を指定して、コンポーネントを取得します。コンポーネントは、IDE で作成されたものでも、 AddComponent で呼び出だされたものでも構いません。

コンポーネントの属性の多くは、このオブジェクトのメソッドとプロパティを通してオートメーション インターフェイスで使用できます。プロジェクトタイプに使用できるメソッド、プロパティ、コレクションについての詳細は、次のテーブルを参照してください。

メンバー

テーブル 49 • ISWiComponent オブジェクトのメンバー

名前	プロジェクト	種類	説明
AddComponentSubFolder	InstallScript、InstallScript オブジェクト	メソッド	現在の ISWiComponent にサブフォルダーを追加します。
AddDynamicFileLinking	基本の MSI、InstallScript、InstallScript MSI、マージモジュール	メソッド	現在のコンポーネントにダイナミック ファイル リンクを追加します。
AddEnvironmentVar	基本の MSI、InstallScript MSI、マージモジュール	メソッド	現在のコンポーネントに環境変数を追加します。
AddFile	All	メソッド	現在のコンポーネントにファイルを追加します。
AddRemoveFile	基本の MSI、InstallScript MSI、マージモジュール	メソッド	現在のコンポーネントにファイルまたはフォルダーの削除エントリを追加します。
Attrib64BitComponent	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	このコンポーネントを 64ビットとして登録するには、このプロパティを True にします。このプロパティを False にすると、コンポーネントが 32 ビットで登録されます。コンポーネントが 64 ビットで、32ビットコンポーネントを置換する場合は、GUID プロパティを使って新しい GUID を指定するようにしてください。 64 ビットサポートには、Windows Installer サービスのバージョン 2.0 が必要となることに注意してください。
Comments	All	読み書きプロパティ	このコンポーネントの コメント を取得または設定します。
Compressed	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール値プロパティは、コンポーネントの ″圧縮″ プロパティを取得または設定します。このプロパティは、コンポーネントのファイルを圧縮する (True) か、リリースビルダーでキャビネット ファイルにファイルを格納したときにファイルを圧縮しない (False) かを指定します。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
条件	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	コンポーネントに関連付けられているすべての条件を含みます。
Destination	All	読み書きプロパティ	コンポーネントのインストール先を取得または設定します。次の形式が有効です。 • (Windows Installer ベースのプロジェクトのみ) [INSTALLDIR] • (Windows Installer ベースのプロジェクトのみ) [ProgramFilesFolder]Subfolder • (InstallScript プロジェクトのみ) <TARGETDIR> • (InstallScript プロジェクトのみ) <WINDIR>¥Subfolder • C:¥FolderName ***重要•*** *ISWiComponent オブジェクトの Destination プロパティを設定すると、オブジェクトの SourceLocation プロパティがリセットされます。したがって、SourceLocation プロパティを設定するコードはすべて、Destination プロパティを設定するコードの後に配置する必要があります。*
DeviceDriverFlags	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティは、サポートされなくなりました。
DeviceDriverFlagsEx	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティを使用すると、デバイス ドライバーのフラグ設定や実行時のオプションをオートメーション レイヤーから直接指定することができます。
DeviceDriverSetRedist	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティは、実行時にデバイス ドライバー インストールすべてに使用される .dll を設定します。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DeviceDriverUse64BitRedist	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	このプロパティは、サポートされなくなりました。
DeviceDriverUseLocalizedRedist	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	このプロパティは、サポートされなくなりました。
差分	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール値プロパティは、コンポーネントの ″差分″ プロパティを取得または設定します。このプロパティは、差分メディアをビルドするときにメディアビルダーのデフォルトの動作を受け入れる (True) か上書きする (False) かを指定します。差分メディアは、同じファイル (同じ日時、サイズ、属性) が指定の比較メディアに存在する場合、このコンポーネントからファイルを除外します。
DotNetApplicationFile	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	アセンブリをプライベートとしてインストールする場合、この属性を、アセンブリを含むファイルの FileKey に設定します。このプロパティが NULL に設定されている場合、アセンブリはグローバルアセンブリキャッシュにインストールされます。
DotNetAssembly	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール値プロパティは、コンポーネントの ″.NET アセンブリ″ プロパティを取得または設定します。このプロパティは、コンポーネントのファイルをローカルの .NET アセンブリ (True) としてインストールするか、アセンブリとしてインストールしないか (False) を指定します。
DotNetCOMInterop	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	.NET アセンブリを COM 呼び出しから使用できるようにするには、これを True に設定します。
DotNetInstallerClass	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	アセンブリが .NET Installer ネームスペースのメソッドを実装する場合、これを True に設定します。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DotNetInstallerClassArgCommit	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティを使うと、Commit メソッドを呼び出したときの Installer クラスの引数リストを指定できます。
DotNetInstallerClassArgInstall	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティを使うと、Installer クラスのインストール実行コンテキストの引数リストを指定できます。
DotNetInstallerClassArgRollback	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティを使うと、Rollback メソッドを呼び出したときの Installer クラスの引数リストを指定できます。
DotNetInstallerClassArgUninstall	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティを使うと、Installer クラスのアンインストール実行コンテキストの引数リストを指定できます。
DotNetPrecompile	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティが True に設定されているとき、セットアップはインストール時に .NET アセンブリからネイティブイメージを作成し、それをターゲット システム上のネイティブイメージキャッシュにインストールします。これによって、アセンブリのロードと実行がスピードアップします。これはコードおよびデータ構造がジャストインタイム (JIT) コンパイルからではなく、ネイティブイメージキャッシュから復元されるためです。
DotNetScanAtBuild	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティはビルドエンジンにアセンブリの処理方法を指示します。次の値から 1 つを設定します。 • ednbNone (0)—ビルドは何も行いません。 • ednbProps (1)—ビルドは、アセンブリのプロパティをスキャンします。 • ednbDepAndProps (2)—ビルドは、アセンブリのプロパティと依存関係をスキャンします。
ExtractAtBuild	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	コンポーネントのビルド時に COM 抽出設定を取得または設定します。使用できる値は True か False です。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
FTPLocation	InstallScript、InstallScript オブジェクト	読み書きプロパティ	コンポーネントの "FTP の場所" プロパティを取得または設定します。このプロパティはコンポーネントの FTP の場所を指定します。
GUID	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	"コンポーネントコード" プロパティとも呼ばれる文字列 GUID を {12345678-1234-1234-1234-1234567890AB} のように中かっこで括って保持します。
HTTPLocation	InstallScript、InstallScript オブジェクト	読み書きプロパティ	コンポーネントの "HTTP の場所" プロパティを取得または設定します。このプロパティはコンポーネントの HTTP の場所を指定します。
ImportINIFile	All	メソッド	オートメーションレイヤーから INI ファイルをインポートします。
ImportRegFile	All	メソッド	REG ファイルの内容を、このコンポーネントのレジストリデータにマージします。
IsDeviceDriver	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	この Boolean プロパティが true の場合、このコンポーネントのキーファイルはデバイス ドライバー パッケージです。つまり、プロパティが false の場合、キーファイルはデバイス ドライバー パッケージではありません。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
IsPlatformSelected	InstallScript、InstallScript オブジェクト	読み書きプロパティ	PlatformSuiteCheck プロパティと併用してコンポーネントの "プラットフォーム スイート" プロパティを取得または設定します。このプロパティは、コンポーネント固有のプラットフォーム スイートがあればこれを指定します。 次の定数を使用してこのプロパティを設定します。 • epsMSBackOffice = 32 (&H20) • epsMSSmallBusinessServer = 256 (&H100) • epsMSSmallBusinessServerWithRestrictiveLiscenses = 512 (&H200) • epsTerminalServices = 16 (&H10) • epsWin2kAdvSvrOrWinDotNetEnterpriseServer = 128 (&H80) • epsWin2kOrWinDotNetDatacenterServer = 64 (&H40) • epsWinServer = 4 • epsWinWorkstation = 8 • epsWinXPHome = 2 • epsWinXPPro = 1 • epsArchIa64 = 1024 (&H400) • epsArchAmd64 = 2048 (&H800) • epsArchIntel32 = 4096 (&H1000) • epsWinServer2003R2 = 8192 (&H2000) 例: pFeature.IsPlatformSelected(epsWinXPHome) = True
ISWiComponentSubFolders	InstallScript、InstallScript オブジェクト	コレクション	このコンポーネントに関連するすべてのサブフォルダーを含みます。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ISWiDynamicFileLinkings	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コレクション	指定されたコンポーネントに関連付けられた、すべてのダイナミック ファイル リンクを含みます。
ISWiEnvironmentVars	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コレクション	このコンポーネントに関連するすべての環境変数を含みます。
ISWiFiles	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コレクション	このコンポーネントに関連するすべてのファイルを含みます。
ISWiFolders	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コレクション	このコンポーネントに関連するすべてのフォルダーを含みます。
ISWiRemoveFiles	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	メソッド	現在のコンポーネントと関連付けられている、すべてのファイルおよびフォルダーの削除エントリを含む ISWiRemoveFiles コレクションを返します。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明

KeyPath	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	
			メモ: *ISWiComponent.KeyPathType は、ISWiComponent.KeyPath を設定する前に設定する必要があります。この順番を逆にすると、エラーが発生します。*
			
			このコンポーネントのキーパスの役割を果たすファイルまたはレジストリ エントリを含みます。正確なタイプは、下記の KeyPathType プロパティで示されます。
			
			キーパスがキーファイルの場合、ファイルキーを含みます。コンポーネントファイル リストのキー列でこの値を参照できますが、ファイルが動的にリンクされている場合、ファイルキーが変更されることがあるので注意します。ファイル キーは、ISWiFile オブジェクトの "名前" プロパティと同じです。
			
			キーパスがレジストリ値であるとき、レジストリ キーを含む特別な形式の文字列があります。レジストリキーのパスは、角かっこで囲まれ、垂直線で値の名前と区切られています。たとえば次の例では、コンポーネント m_MyComponent のキーパスを、HKEY_LOCAL_MACHINE¥Software¥MyCompany¥Settings の値 MyName に設定します。
			
			`m_MyComponent.KeyPathType = kptRegistry`
			`m_MyComponent.KeyPath = "[HKEY_LOCAL_MACHINE¥Software¥MyCompany¥Settings]|MyName"`
			
			動的にリンクされているファイルは、キーファイルにはできません。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
KeyPathType	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	ISWiComponent.KeyPathType は、ISWiComponent.KeyPath を設定する前に*設定する必要があります*。逆の順番で実行されると、エラーが発生します。KeyPath プロパティに含まれるキーパスの正確なタイプをポイントする値を格納します。次の種類のうち 1 つを指定します。 • kptRegistry (1) • kptFile (2) • kptFolder (3) • kptODBC (4) オートメーションインターフェイスはキーパスの種類を kptRegistry または kptFile にのみ設定できますが、現在のキーパスの種類を読み取っている場合は、別の値が戻り値になる可能性もあります。
その他	InstallScript、InstallScript オブジェクト	読み書きプロパティ	コンポーネントの "その他" プロパティを取得または設定します。このプロパティはテキスト文字列をコンポーネントに関連付けます。
名前	All	読み取り専用プロパティ	コレクション内にある現在の項目のコンポーネント名
NeverOverwrite	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	インストーラーに、既存コンポーネントを置換するかどうか指示します。既存コンポーネントを残すには True を指定し、置換するには Falseを指定します。
OSFilter	InstallScript、InstallScript MSI、InstallScript オブジェクト	読み書きプロパティ	コンポーネントの "オペレーティング システム" 設定を取得または設定します。ターゲット マシンのオペレーティング システムがこの設定に指定されたオペレーティング システムの中にない場合、コンポーネントはインストールされません。デフォルトでは、コンポーネントはオペレーティング システムに依存しません。つまり、コンポーネントに特定のオペレーティング システム固有のデータがないことを意味します。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
OSFilter (続き)			次の定数を使用してこのプロパティを設定します。 • eosOSIndepedent = 0 • eosWin95 = &H10 (16) • eosWin98 = &H40 (64) • eosWinMe = &H80 (128) • eosWinNT4 = &H10000 (65536) • eosWin2000 = &H100000 (1048576) • eosWinXP = &H400000 (4194304) • eosWinServer2003 = &H800000 (8388608) • eosWinVista = &H1000000 (16777216)—これらの定数は Windows Vista および Windows Server 2008 用です。 • eosWin7 = &H2000000 (33554432)—これらの定数は Windows 7 および Windows Server 2008 R2 用です。 • eosWin8 = &H4000000 (67108864)—これらの定数は Windows 8 および Windows Server 2012 用です。 • eosWin81 = &H8000000 (134217728)—これらの定数は Windows 8.1 および Windows Server 2012 R2 用です。 • eosAll = &HFD100D0 (265355472) 複数のプラットフォームを指定できます (例、**eosWin7 Or eosWinServer2008 Or eosWinVista Or eosWinServer2003**)。 ***ヒント:*** *このプロパティを IsPlatformSelected プロパティと一緒に使用して、Windows Vista と Windows Server 2008 との違い、Windows 7 と Windows Server 2008 R2 との違い、Windows 8 と Windows Server 2012 との違い、または Windows 8.1 と Windows Server 2012 R2 の違いを区別します。*

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
OverwriteMainOptions	InstallScript、InstallScript オブジェクト	読み書きプロパティ	OverwriteSubOptionsVersion プロパティと OverwriteSubOptionsDate プロパティを合わせて、コンポーネントの "上書き" プロパティを取得または設定します。このプロパティは、コンポーネントのファイルでターゲット システムの既存のバージョンを常に上書きするか、上書きしないか、または日時スタンプやバージョン番号に基づいて条件的に上書きするかどうかを指定します。次の定数を使用してこのプロパティを設定します。 • ecomoAlways (0)—ターゲット システムのファイルは常に上書きされます。 • ecomoByDate (1)—ターゲット システムのファイルは、OverwriteSubOptionsDate プロパティで指定された日時スタンプに基づいて条件付きで上書きされます。 • ecomoByVersion (2)—ターゲット システムのファイルは、OverwriteSubOptionsVersion プロパティで指定されたバージョン番号に基づいて条件付きで上書きされます。 • ecomoByVersionThenDate (3)—ターゲット システム上のファイルは、バージョン番号に基づいて条件付きで上書きされるか、配布メディアのファイルとターゲット システムのファイルが同じバージョン番号を持つ場合やいずれのファイルにもバージョン番号がない場合は、OverwriteSubOptionsVersion プロパティと OverwriteSubOptionsDate プロパティで指定された日時スタンプに基づいて上書きされます。 • ecomoNever (4)—ターゲット システムのファイルは上書きされません。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
OverwriteSubOptionsDate	InstallScript、InstallScript オブジェクト	読み書きプロパティ	OverwriteMainOptions プロパティと OverwriteSubOptionsVersion プロパティを合わせて、コンポーネントの "上書き" プロパティを取得または設定します。このプロパティは、コンポーネントのファイルでターゲット システムの既存のバージョンを常に上書きするか、上書きしないか、または日時スタンプやバージョン番号に基づいて条件的に上書きするかどうかを指定します。OverwriteMainOptions が ecomoAlways、ecomoByVersion、ecomoNever に設定されている場合は無視されます。次の定数を使用してこのプロパティを設定します。 • ecomsoNewer (0)—配布メディアのファイルがより新しい日付の場合、ターゲット システムのファイルは上書きされます。 • ecomsoNewerOrSame (1)—配布メディアのファイルがより新しいか同じ日付の場合、ターゲット システムのファイルは上書きされます。 • ecomsoOlder (2)—配布メディアのファイルがより古い日付の場合、ターゲット システムのファイルは上書きされます。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
OverwriteSubOptionsVersion	InstallScript、InstallScript オブジェクト	読み書きプロパティ	OverwriteMainOptions プロパティと OverwriteSubOptionsDate プロパティを合わせて、コンポーネントの "上書き" プロパティを取得または設定します。このプロパティは、コンポーネントのファイルでターゲット システムの既存のバージョンを常に上書きするか、上書きしないか、または日時スタンプやバージョン番号に基づいて条件的に上書きするかどうかを指定します。OverwriteMainOptions が ecomoAlways、ecomoByDate、ecomoNever に設定されている場合は無視されます。次の定数を使用してこのプロパティを設定します。 • ecomsoNewer (0)—配布メディアのファイルのバージョン番号がより高い番号の場合、ターゲット システムのファイルは上書きされます。 • ecomsoNewerOrSame (1)—配布メディアのファイルのバージョン番号が同じか高い番号の場合、ターゲット システムのファイルは上書きされます。 • ecomsoOlder (2)—配布メディアのファイルのバージョン番号が低い番号の場合、ターゲット システムのファイルは上書きされます。
パーマネント	All	読み書きプロパティ	このプロパティは、IDE の Permanent コンポーネントプロパティに相当します。コンポーネントが削除されないようにする場合、Permanent を True とし、それ以外の場合は False とします。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PlatformSuiteCheck	InstallScript、InstallScript オブジェクト	読み書きプロパティ	IsPlatformSelected プロパティと併用してコンポーネントの "プラットフォーム スイート" プロパティを取得または設定します。このプロパティは、コンポーネント固有のプラットフォーム スイートがあればこれを指定します。次の定数を使用してこのプロパティを設定します。 • ecpscAll (0)—セットアップは、IsPlatformSelected プロパティで指定されたすべてのスイートがターゲット システムに存在する場合にのみコンポーネントのファイルをインストールします。 • ecpscAtLeastOne (1)—セットアップは、IsPlatformSelected プロパティで指定されたスイートが少なくとも 1 つ ターゲット システムに存在する場合にのみコンポーネントのファイルをインストールします。 • ecpscSuiteIndependent (2)—コンポーネントのファイルのインストールは、ターゲット システムのスイートに依存しません。
PotentiallyLocked	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール値プロパティは、コンポーネントの "ファイルのロック" プロパティを取得または設定します。このプロパティは、既にインストールされているコンポーネントのファイルをロックするかどうかを指定します。つまりセットアップ中に別のアプリケーションで使用するかどうかを指定します。
RegFileToMergeAtBuild	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	ビルド時にコンポーネントのレジストリ エントリと一緒にマージする .reg ファイルの完全パスを指定します。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
RegistrationType	基本の MSI、InstallScript MSI、マージモジュール	プロパティ	次のオプションは、このコンポーネントの登録情報のソースを決定するために使用できます。 • rgtNone (0)—InstallShield は、データを動的に抽出したり、コンポーネントを自己登録型としてマークしませんが、COM 登録の詳細設定の情報はすべて登録されます。 • rgtSelfReg (-1)—このコンポーネントのファイルの自己登録ルーチンは、インストーラーによって呼び出されます。 • rgtExtractAtBuild (-2)—リリースのビルド時に COM 登録情報を動的に抽出します。 IDE のコンポーネントビューから "登録タイプ" プロパティが削除されたため、オートメーション インターフェイスによる " 登録タイプ" プロパティの設定には次のような影響があります。 • rgtNone を使うと、コンポーネントの "ビルド時に COM抽出" プロパティが [いいえ] に設定されます。 • rgtSelfReg を使うと、コンポーネントの "ビルド時に COM 抽出" プロパティが [いいえ] に設定され、コンポーネントの各ファイルで自己登録を使用するようになります。 • rgtExtractAtBuild を使うと、コンポーネントの "ビルド時に COM 抽出" プロパティが [はい] に設定され、コンポーネントの各ファイルで自己登録を使用しないようになります。
RemoteInstallation	基本の MSI、InstallScript MSI、マージモジュール	読み書きプロパティ	次のオプションのうち 1 つを使用して、コンポーネントのファイルをローカルにインストールするか、または配布メディアに残します。 • rfsLocal (0)—コンポーネントの "インストール先" プロパティに保存されたフォルダーにファイルをインストールします。 • rfsSource (1)—配布メディアにファイルを残します。 • rfsOptional (2)—コンポーネントは、機能のプロパティに従います

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
RemoveComponentSubFolder	InstallScript、InstallScript オブジェクト	メソッド	現在の ISWiComponent からサブフォルダーを削除します。
RemoveDynamicFileLinking	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	メソッド	現在のコンポーネントからダイナミック ファイル リンクを削除します。
RemoveEnvironmentVar	基本の MSI、InstallScript MSI、マージ モジュール	メソッド	現在のコンポーネントから環境変数を削除します。
RemoveFile	All	メソッド	現在のコンポーネントからファイルを除去します。
RemoveRemoveFile	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	メソッド	現在のコンポーネントからファイルまたはフォルダーの削除エントリを削除します。ファイルまたはフォルダーの削除エントリをプロジェクトから削除するには、ISWiRemoveFile オブジェクトを渡します。
SelfRegister	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール値プロパティは、コンポーネントの "自己登録" プロパティを取得または設定します。このプロパティは、コンポーネントのファイルを自己登録にするかどうかを指定します。
SharedDLLRefCount	All	読み書きプロパティ	True を指定すると、インストール時にコンポーネントの各ファイルの参照カウントをインストーラーが増分し、False を指定すると、これを実行しません。

テーブル 49 • ISWiComponent オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SourceLocation	All	読み書きプロパティ	圧縮セットアップパッケージまたはリリースの保存場所にある、フォルダーの名前を指定します。リリースの保存場所は、リリースのビルド時にこのコンポーネントのファイルが格納される場所です。このプロパティは、InstallShield でコンポーネントの "ソースの場所" 設定と同じです。 ***重要•*** *ISWiComponent オブジェクトの Destination プロパティを設定すると、オブジェクトの SourceLocation プロパティがリセットされます。したがって、SourceLocation プロパティを設定するコードはすべて、Destination プロパティを設定するコードの後に配置する必要があります。*
Transitive	All	読み書きプロパティ	コンポーネントのインストール時に条件の再評価をインストーラーにさせるには True とし、それ以外の場合は False とします。

次に適用：

- ISWiFeature

AddComponentSubFolder メソッド

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

AddComponentSubFolder メソッドは、指定のコンポーネントサブフォルダーを現在のコンポーネント、またはコンポーネントのサブフォルダーに追加します。

構文

AddComponentSubFolder (Name As String) As ISWiComponentSubFolder

パラメーター

テーブル 50 ・ AddComponentSubFolder メソッド

パラメーター	説明
名前	追加するサブフォルダーの名前を渡します。

例

次の Visual Basic 行では、AddComponentSubFolder メソッドを示します:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pComponent As ISWiComponent
Dim pComponentSubFolder As ISWiComponentSubFolder

Set pFeature = pProj.ISWiFeatures.Item("MyFeature")
Set pComponent = pFeature.ISWiComponents.Item("MyComp")
Set pComponentSubFolder = pComponent.AddComponentSubFolder("MyCompSubFolder")

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiComponent
- ISWiComponentSubFolder

AddDynamicFileLinking メソッド

***プロジェクト・** この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

AddDynamicFileLinking を呼び出して、指定のコンポーネントにダイナミック ファイル リンクを追加します。指定したフォルダーに含まれるファイルはすべて、自動的にインストールに組み込まれます。

プロジェクト: *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトで新しいダイナミック ファイル リンクを追加すると、デフォルトでそのすべてにコンポーネントのベスト プラクティスが適用されます。詳細については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。*

構文

AddDynamicFileLinking (sSourceFolder As String) As ISWiDynamicFileLinking

パラメーター

テーブル 51 : AddDynamicFileLinking メソッドのパラメーター

パラメーター	説明
sSourceFolder	動的にリンクさせるファイルを含むフォルダーに、完全修飾パスを渡します。

例

次の Visual Basic 例は、**MyDynamicFiles** へのダイナミック リンクをコンポーネント **MyComponent** に追加します。

```
m_ISWiFeature.ISWiComponents("MyComponent").AddDynamicFileLinking "C:¥Build 141¥MyDynamicFiles"
```

MyDynamicFiles フォルダーに含まれるすべてのファイルが、プロジェクトに追加されます。フォルダーは各ビルドの前にスキャンされ、すべての新規または変更済みファイルが自動的にプロジェクトに組み込まれます。

次に適用 :

- ISWiComponent

AddEnvironmentVar メソッド

プロジェクト: *AddEnvironmentVar メソッドは、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

AddEnvironmentVar を呼び出して、コンポーネントに環境変数を追加します。

構文

AddEnvironmentVar (sName As String) As ISWiEnvironmentVar

パラメーター

テーブル 52 • AddEnvironmentVar メソッドのパラメーター

パラメーター	説明
sName	環境変数の名前を指定します。

例

以下の Visual Basic の例は、環境変数 **MyEnvironment** をコンポーネント **MyComponent** に追加します。

```
m_ISWiFeature.ISWiComponents("MyComponent").AddEnvironmentVar "MyEnvironment"
```

次に適用：

- ISWiComponent

AddFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

AddFile を呼び出してファイルを指定のコンポーネントに追加するか、InstallScript プロジェクトの場合は、コンポーネントサブフォルダーにファイルを追加します。このメソッドを使って追加したファイルは、スタティック ファイル リンクを持ちします。

構文

AddFile (FileName As String) As ISWiFile

パラメーター

テーブル 53 • AddFile メソッド パラメーター

パラメーター	説明
FileName	このコンポーネントまたはコンポーネント サブフォルダーに追加する個々のファイルの完全なファイル名を渡します。

例

以下の Visual Basic の例では、ファイル MyFile.exe がコンポーネント MyComponent に追加されます。

```
m_ISWiFeature.ISWiComponents("MyComponent").AddFile "C:¥Build 141¥MyFile.exe"
```

次に適用：

- ISWiComponent
- ISWiComponentSubFolder

AddRemoveFile メソッド

プロジェクト: *AddRemoveFile メソッドは、次のプロジェクトの種類に適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

AddRemoveFile メソッドは、現在のコンポーネントにファイルまたはフォルダーの削除エントリを追加します。このファイルとフォルダーの削除機能は、アプリケーションによって作成されるファイルの削除など、インストールが追跡を行わない処理に使用すると便利です。

構文

AddRemoveFile () As ISWiRemoveFile

次に適用：

- ISWiComponent

DeviceDriverFlagsEx プロパティ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

DeviceDriverFlagsEx プロパティを使用すると、デバイス ドライバーのフラグ設定やランタイム オプションをオートメーション レイヤーから直接指定することができます。このプロパティに次のような値を渡すことができます。OR を使用すると、複数の値を渡すことができます。

テーブル 54 • DeviceDriverFlagsEx Property の値

値	定義
0	設定なし (デフォルト)
1	デバイスの既存のドライバーをこのドライバーで置き換えるには、このプロパティの値を DeviceDriverFlagsEx プロパティに渡します。このプロパティを渡さずに、デバイスのドライバーが既に存在すると、ドライバーのインストールは実行されません。
2	ドライバーに対応するデバイスがコンピューターへ接続されていない場合、デバイスドライバーのインストール中に [デバイスをコンピューターへ接続] ダイアログが表示されないようにするには、このプロパティ値を渡します。
4	このプロパティの値を DeviceDriverFlagsEx プロパティに渡すと、インストールで、デバイスドライバーの [プログラムの追加と削除] エントリのみが作成され、インストール自体の追加エントリは作成されません。DIFxApp は Windows Vista 以降のシステム上では、この機能をサポートしません。
8	デフォルトでは、DIFxApp は、署名されていないドライバー パッケージや、ファイルが足りないドライバー パッケージをインストールすることができません。このデフォルトの動作をオーバーライドするには、この値を DeviceDriverFlagsEx プロパティに渡します。
16	デフォルトで、DIFxApp は、ドライバー パッケージをアンインストールするとき、ドライバーをインストールしたときにシステムにコピーされたバイナリ ファイルを削除しません。このデフォルトの動作をオーバーライドするには、この値を DeviceDriverFlagsEx プロパティに渡します。 この値を渡すと、DIFxApp は、バイナリ ファイルがドライバー ストア内の対応するバイナリ ファイルと同一の場合のみ、そのバイナリ ファイルをシステムから削除します。 ***注意 •*** *ドライバーのバイナリ ファイルが他のドライバー パッケージまたはアプリケーションに必要ではないことを確認できるときのみ、この値を渡してください。*

テーブル 54 • DeviceDriverFlagsEx Property の値 (続き.)

値	定義
その他の値	無効。これは致命的なインストールエラーです。DIFxApp はコンポーネントをインストールせず、このコンポーネントの前にインストールされた同じインストールパッケージに含まれるコンポーネントをアンインストールします。

次に適用：

- ISWiComponent

DeviceDriverSetRedist プロパティ

プロジェクト: DeviceDriverSetRedistプロパティは次のプロジェクトの種類に適用します：

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

DeviceDriverSetRedist プロパティは、プロジェクトのすべてのデバイス ドライバー インストールに使われるカスタム アクション ランタイム .dll を設定します。このプロパティに使用できる値は、以下のとおりです。

- eddr32Bit—このプロジェクトのデバイス ドライバーが 32 ビット マシンをターゲットし、インストールは英語のランタイム ダイアログのみを使用することを示します。
- eddf32BitLocalized—このプロジェクトのデバイス ドライバーが 32 ビット マシンをターゲットし、インストールにすべてのローカライズされたインストール ランタイム ダイアログが含まれていることを示します。
- eddf64Bit—このプロジェクトのデバイス ドライバーが Itanium 64 ビット マシンをターゲットし、インストールは英語のランタイム ダイアログのみを使用することを示します。
- eddf64BitLocalized—このプロジェクトのデバイス ドライバーが Itanium 64 ビット マシンをターゲットし、インストールにすべてのローカライズされたインストール ランタイム ダイアログが含まれていることを示します。
- eddf64BitAMD—このプロジェクトのデバイス ドライバーが AMD 64 ビット マシンをターゲットし、インストールは英語のランタイム ダイアログのみを使用することを示します。
- eddf64BitAMDLocalized—このプロジェクトのデバイス ドライバーが AMD 64 ビット マシンをターゲットし、インストールにすべてのローカライズされたインストール ランタイム ダイアログが含まれていることを示します。

英語のランタイム .dll ファイルは、ローカライズされた .dll ファイルよりも小さいため、容量的に問題がある場合や追加の言語ランタイム ダイアログが不要な場合は、(ローカライズされた値ではなく) "英語の値のみ"のオプションから選択します。

ローカライズされた値から選択すると、インストールは以下の言語のダイアログおよびテキストを含むランタイム .dll を使用します。

- アラビア語 (サウジアラビア)

- スペイン語 – モダン ソート (スペイン)
- ノルウェー語 (ブークモール) (ノルウェー)
- 中国語 (中国)
- フィンランド語 (フィンランド)
- オランダ語 (オランダ)
- 中国語(台湾)
- フランス語 (フランス)
- ポーランド語 (ポーランド)
- チェコ語 (チェコ)
- ヘブライ語 (イスラエル)
- ポルトガル語 (ブラジル)
- デンマーク語 (デンマーク)
- ハンガリー語 (ハンガリー)
- ポルトガル語 (ポルトガル)
- ドイツ語 (ドイツ)
- イタリア語 (イタリア)
- ロシア語 (ロシア)
- ギリシャ語 (ギリシャ)
- 日本語 (日本)
- スウェーデン語 (スウェーデン)
- 英語 (U.S.)
- 韓国語 (韓国)
- トルコ語 (トルコ)

次に適用 :

- ISWiComponent

ImportINIFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ImportINIFile メソッドを呼んで INI ファイルをオートメーションレイヤーからインポートします。

構文

ImportINIFile(INIFile As String, [DirectoryID As String])

パラメーター

テーブル 55 • ImportINIFile メソッド パラメーター

パラメーター	説明
INIFile	インポートする必要がある INI ファイルへの完全パスを示します。
DirectoryID	ディレクトリは、INSTALLDIR 等の有効な MSI ディレクトリ識別子でなければありません。ディレクトリは指定をしないとオプションになります。InstallShield は、コンポーネント ディレクトリを使用します。

次に適用：

- ISWiComponent

ImportRegFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ImportRegFile メソッドを呼び出して、指定したコンポーネントのレジストリデータに、レジストリ (REG) ファイルの内容をインポートします。このメソッドは、手動で IDE で REG ファイルをインポートする操作を模倣します。

構文

ImportRegFile (RegFile As String, Optional OverWrite As Boolean)

パラメーター

テーブル 56 • ImportRegFile メソッド パラメーター

パラメーター	説明
RegFile	インポートする REG ファイルの完全なパスを指定します。
OverWrite	次のオプションの中から 1 つを選択し、レジストリデータの結合時に発生する競合の処理方法をImportRegFileに指示します。 True—コンポーネントのレジストリ データに値が存在する場合、REG ファイルからの値で上書きします。 False—REG ファイルの値がコンポーネントのレジストリ データに既に存在する場合、値はインポートしません。 このオプションのパラメーター値は、デフォルトで True に設定されています。

次に適用：

- ISWiComponent

ISWiRemoveFiles メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiRemoveFiles メソッドは、現在のコンポーネントと関連付けられている、すべてのファイルおよびフォルダーの削除エントリを含む ISWiRemoveFiles コレクションを返します。

構文

ISWiRemoveFiles As ISWiRemoveFiles

パラメーター

テーブル 57 • RemoveRemoveFile メソッドのパラメーター

パラメーター	説明
pRemoveFolder	削除する ISWiRemoveFile オブジェクトを渡します。

例

次の Visual Basic コードは、コンポーネントの特定のファイル削除エントリにプロパティを設定する方法の例です。この例では、変数 m_oMyComp によって既にコンポーネントが参照されているものとします。

```
m_oMYComp.ISWiRemoveFiles("RemoveFile1").FileName = "*.txt"
```

次に適用 :

- ISWiComponent

RemoveComponentSubFolder メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

RemoveComponentSubFolder メソッドは、指定のコンポーネントサブフォルダーを現在のコンポーネント、またはコンポーネントのサブフォルダーから削除します。

構文

RemoveComponentSubFolder (pComponentSubFolder As ISWiComponentSubFolder) As Long

パラメーター

テーブル 58 • RemoveComponentSubFolder メソッドのパラメーター

パラメーター	説明
pComponentSubFolder	削除するサブフォルダー オブジェクトを渡します。

例

次の Visual Basic 行では、RemoveComponentSubFolder メソッドを示します:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
```

```
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pComponent As ISWiComponent
Dim pComponentSubFolder As ISWiComponentSubFolder

Set pFeature = pProj.ISWiFeatures.Item("MyFeature")
Set pComponent = pFeature.ISWiComponents.Item("MyComp")
Set pComponentSubFolder = pComponent.ISWiComponentSubFolders.Item("MyCompSubFolder")
pComponent.RemoveComponentSubFolder pComponentSubFolder

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiComponent
- ISWiComponentSubFolder

RemoveDynamicFileLinking メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

RemoveDynamicFileLinking を呼び出して、現在のコンポーネントからダイナミック ファイル リンクを削除します。

構文

RemoveDynamicFileLinking (pSourceFolder As ISWiDynamicFileLinking) As Long

パラメーター

テーブル 59 · RemoveDynamicFileLinking メソッドのパラメーター

パラメーター	説明
pSourceFolder	ISWiDynamicFileLinking オブジェクトを渡して、プロジェクトから削除するダイナミック ファイル リンクを指定します。

次に適用：

- ISWiComponent

RemoveEnvironmentVar メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

RemoveEnvironmentVar を呼び出して、指定したコンポーネントからファイルを削除します。

構文

RemoveEnvironmentVar (sName As ISWiRemoveEnvironmentVar) As Long

パラメーター

テーブル 60 : RemoveEnvironmentVar メソッドのパラメーター

パラメーター	説明
sName	ISWiEnvironmentVar オブジェクトを渡して、削除する環境変数を指定します。

次に適用 :

- ISWiComponent

RemoveFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

指定したコンポーネントからファイルを削除するには、RemoveFile を呼び出します。

構文

RemoveFile (pFile As ISWiFile) As Long

パラメーター

テーブル 61 • RemoveFile メソッド パラメーター

パラメーター	説明
pFile	削除する ISWiFile オブジェクトを渡します。

次に適用 :

- ISWiComponent

RemoveRemoveFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

RemoveRemoveFile メソッドは、現在のコンポーネントからファイルまたはフォルダーの削除エントリを削除します。ファイルまたはフォルダーの削除エントリをプロジェクトから削除するには、ISWiRemoveFile オブジェクトを渡します。

構文

RemoveRemoveFile (pRemoveFolder As ISWiRemoveFile) As Long

パラメーター

テーブル 62 • RemoveRemoveFile メソッドのパラメーター

パラメーター	説明
pRemoveFolder	削除する ISWiRemoveFile オブジェクトを渡します。

次に適用 :

- ISWiComponent

ISWiComponentSubFolder オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

ISWiComponentSubFolder は、現在のコンポーネントまたはコンポーネント サブフォルダーに属する コンポーネント サブフォルダー を表します。コンポーネント サブフォルダーは、InstallShield ユーザー インターフェイスの [コンポーネント] ビューにある既存のサブフォルダーでも、AddComponentSubFolder を呼び出して追加したものでもかまいません。

プロジェクトの ISWiComponentSubFolder コレクションを使用して特定のコンポーネントサブファイルにアクセスします。ISWiComponentSubFolders の Item プロパティには、インデックス番号かコンポーネントサブフォルダー名のいずれかを指定します。

メンバー

テーブル 63 · ISWiComponentSubFolder オブジェクトのメンバー

名前	種類	説明
AddComponentSubFolder	メソッド	コンポーネント サブフォルダーにサブフォルダーを追加します。
AddFile	メソッド	コンポーネント サブフォルダーにファイルを追加します。
DeleteFile	メソッド	コンポーネント サブフォルダーからファイルを削除します。
ISWiComponentSubFolders	コレクション	このコンポーネント サブフォルダーに関連するすべてのサブフォルダーを含みます。
ISWiFiles	コレクション	このコンポーネント サブフォルダーに関連するすべてのファイルを含みます。
名前	読み取り専用プロパティ	コンポーネント サブフォルダーの名前を格納します。
RemoveComponentSubFolder	メソッド	コンポーネント サブフォルダーからサブフォルダーを削除します。

例

次の Visual Basic コードは ISWiComponentSubFolder オブジェクトの使用方法を示しています:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
```

```
Dim pComponent As ISWiComponent
Dim pComponentSubFolder As ISWiComponentSubFolder

Set pFeature = pProj.ISWiFeatures.Item("MyFeature")
Set pComponent = pFeature.ISWiComponents.Item("MyComp")
Set pComponentSubFolder = pComponent.ISWiComponentSubFolders.Item("MyCompSubFolder")
pComponentSubFolder.AddFile("C:¥My Files¥MyFile.ext")

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiComponent

DeleteFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

DeleteFile を呼び出して、指定の コンポーネントサブフォルダーからファイルを削除します。.このメソッドで追加されたファイルには、IDE でファイルを追加した場合と同様のスタティックなファイルリンクが含まれます。

構文

DeleteFile (pFile As ISWiFile) As Long

パラメーター

テーブル 64 : DeleteFile メソッド パラメーター

パラメーター	説明
pFile	削除するファイルの オブジェクトを渡します。

例

次の Visual Basic の例では、MyCompSubFolder というコンポーネントサブフォルダーからファイルを削除します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pComponent As ISWiComponent
Dim pComponentSubFolder As ISWiComponentSubFolder
Dim pFile As ISWiFile

Set pFeature = pProj.ISWiFeatures.Item("MyFeature")
Set pComponent = pFeature.ISWiComponents.Item("MyComp")
Set pComponentSubFolder = pComponent.ISWiComponentSubFolders.Item("MyCompSubFolder")
```

```
Set pFile = pComponentSubFolder.ISWiFiles.Item("MyFile.ext")
pComponentSubFolder.DeleteFile pFile

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiComponentSubFolder

ISWiCondition オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiCondition は、ISWiFeature オブジェクトのサブセットです。このオブジェクトでは、プロジェクトの機能の条件付論理を取得および設定することができます。

メンバー

テーブル 65 • ISWiCondition オブジェクトのメンバー

名前	種類	説明
条件	読み書きプロパティ	この機能に関連した 条件 を取得または設定します。
レベル	読み書きプロパティ	条件が TRUE に評価された場合に、機能に割り当てられる インストールレベル を取得または設定します。

次に適用：

- ISWiFeature

ISWiDynamicFileLinking オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiDynamicFileLinking は、コンポーネントに属するダイナミック ファイル リンクを表します。ダイナミック ファイル リンクは、InstallShield ユーザー インターフェイスの既存のファイル、または AddDynamicFileLinking を呼び出して追加したファイルのどちらでもかまいません。

メンバー

テーブル 66 • ISWiDynamicFileLinking オブジェクトのメンバー

名前	プロジェクト	種類	説明
CreateBestPracticeComponents	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ダイナミック リンク ファイルにのコンポーネント作成でベスト プラクティス メソッドを使用する場合、このプロパティを True に設定します。 コンポーネント作成にディレクトリごとに 1 つのコンポーネント メソッドを使用する場合、このプロパティを True に設定します。 2 つのメソッドに関する詳細については、「ダイナミック リンクがあるファイルの適切なコンポーネント作成方法を判別する」を参照してください。
DynamicSubfolders	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	SourceFolder で提供されているすべてのサブフォルダーを含む場合、このプロパティを True に設定します。
ExcludeFiles	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	除外するファイルの種類を指定します。アスタリスク (*) に続けて拡張子を入力します。複数のエントリはカンマで区切ります。
IncludeFiles	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	含めるファイルの種類を指定します。アスタリスク (*) に続けて拡張子を入力します。複数のエントリはカンマで区切ります。
SelfRegister	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	動的ににリンクされたフォルダーに含まれるすべてのファイルを自己登録させる場合、このプロパティを True に設定します。 コンポーネントの Attrib64BitComponent プロパティが True に設定されると、ターゲット システム上でインストールは 64 ビットの自己登録を行います。詳細については、「64 ビット オペレーティング システムをターゲットにする」を参照してください。

テーブル 66 • ISWiDynamicFileLinking オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SourceFolder	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	コンテンツを動的にリンク付けするフォルダーへの完全修飾パスを指定します。

次に適用 :

- ISWiComponent

ISWiEnvironmentVar オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiEnvironmentVar は、コンポーネントに属する環境変数を表します。ファイルは、InstallShield ユーザー インターフェイスの既存のファイル、または AddEnvironmentVarを呼び出して追加したファイルのどちらでもかまいません。

メンバー

テーブル 67 • ISWiEnvironmentVar オブジェクトのメンバー

名前	種類	説明
EnvType	読み書きプロパティ	この列挙された整数プロパティは、環境変数名がシステム変数またはユーザー環境変数のどちらであるかを指定します。 • evtSystem—指定されたシステム環境変数を作成、変更または削除します。 • evtUser—ユーザー環境から環境変数を作成、変更または削除します。指定された環境変数は、インストール時にログオンしているエンドユーザー以外は使用できません。
Key	読み取り専用プロパティ	Environment テーブルのキーを格納します。
名前	読み書きプロパティ	環境変数の名前を取得、または設定します。
OnInstall	読み書きプロパティ	この列挙された整数プロパティは、関連付けられた機能がターゲット システムにインストールされたときに発生するアクションを取得または設定します。 • evoiSet—"配置" プロパティと共に既存の環境変数の値を設定します。このオプションは、ターゲット システムに環境変数がない場合、これを作成して、インストール中にこれを設定します。その環境変数がターゲット システムに存在する場合、インストール中に設定されます。 • evoiCreate—指定された環境変数が既に存在しない場合、ターゲット システムに環境変数を作成して、変数の値を設定します。 • evoiRemove—指定された環境変数をターゲット システムから削除します。
OnUninstall	読み書きプロパティ	この列挙された整数のプロパティは、関連付けられた機能がアンインストールされるときに、この環境変数を削除するかどうかを指定します。 • evouRemove—関連付けられた機能がアンインストールされる時に、環境変数または追加された値をターゲット システムから削除します。OnInstall プロパティが Create に設定されていると、OnUninstall の Remove によって環境変数全体が削除されます。OnInstall プロパティが Set に設定されていると、OnUninstall の Remove 値は、変数の値に付加された値のみを削除します。 • evouLeave — 関連付けられた機能がアンインストールされた時に、環境変数または追加された値をターゲット システムに残したままにします。

テーブル 67・ISWiEnvironmentVar オブジェクトのメンバー（続き.）

名前	種類	説明
Placement	読み書きプロパティ	この列挙された整数のプロパティは、指定された環境変数の既存値に相対して、"値" フィールドの値の配置を取得または設定します。 ・ evpAppend—指定された環境変数の既存値の最後に、Value プロパティで示された値を追加します。 ・ evpPrefix—指定された環境変数の既存値の始めに、Value プロパティで示された値を追加します。 ・ evpReplace—指定された環境変数の値を Value プロパティに示された値で置換します。
値	読み書きプロパティ	この環境変数のパスまたは値を、取得または設定します。

次に適用：

- ISWiComponent

ISWiFeature オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiFeature は、InstallShield のトップレベルの機能またはサブ機能を示します。ISWiFeatures コレクションにある項目を指定して、機能を呼び出します。

機能プロパティと属性の多くは、このオブジェクトのメソッド、プロパティ、コレクションを通してオートメーションインターフェイスで使用できます。プロジェクトタイプに使用できるメソッド、プロパティ、コレクションについての詳細は、次のテーブルを参照してください。

メンバー

テーブル 68 • ISWiFeature オブジェクトのメンバー

名前	プロジェクト	種類	説明
AddCondition	基本の MSI、InstallScript MSI	メソッド	現在の ISWiFeature に条件を追加します。
AddFeature	基本の MSI、InstallScript、InstallScript MSI	メソッド	ISWiFeature オブジェクトを、現在の ISWiFeature のサブ機能として作成します。
AddMergeModule	基本の MSI、InstallScript MSI	メソッド	マージ モジュールを現在の機能に関連付けます。
AddObject	InstallScript	メソッド	現在の ISWiFeature に InstallScript オブジェクトを追加します。
アドバタイズ	基本の MSI、InstallScript MSI	読み書きプロパティ	現在の機能のアドバタイズプロパティを取得または設定します。次の定数を使用してこのプロパティを設定します。 • eaAllow (1)—この機能のアドバタイズを有効にするには、このオプションを使用します。アドバタイズは許可されますが、セットアップ実行時のデフォルトのオプションではありません。 • eaFavor (2)—機能をデフォルトでアドバタイズするには、このオプションを使用します。エンド ユーザーは、Custom Setup ダイアログの機能のアドバタイズ オプションをいつでも変更することができます。 • eaDisallow (3)—この機能でアドバタイズを使用できないようにする場合、このオプションを使用します。エンドユーザーは、[カスタム セットアップ] ダイアログで機能をアドバタイズできません。 • eaDisableIfNotSupported (4)—アドバタイズは、Internet Explorer 4.01 以降のシステムでのみ動作します。ターゲット システムがこの条件を満たしていない場合、アドバタイズはオフにされます。ターゲット システムがアドバタイズをサポートする場合、アドバタイズは許可されます。

テーブル 68 • ISWiFeature オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AttachComponent	基本の MSI、InstallScript、InstallScript MSI	メソッド	コンポーネントを現在の機能に関連付けます。
CDRomFolder	InstallScript	読み書きプロパティ	機能の [CD-ROM フォルダー] プロパティを取得または設定します。このプロパティは、機能のチェック ボックスがリリース ウィザードの [カスタムメディアレイアウト] パネルがチェックされている場合に、機能の配置先となるディスクイメージのフォルダーを指定します。
Comments	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	この機能に関連付けられた コメント を取得または設定します。
DeleteCondition	基本の MSI、InstallScript MSI	メソッド	現在の機能に関連付けられた条件を削除します。
説明	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	この機能の説明を、取得または設定します。
DescriptionID	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	この機能の説明の文字列識別子を、取得または設定します。 ***メモ:*** *このプロパティを設定するには、文字列エントリが存在して、その文字列 ID が判明していなくてはなりません。*
Destination	基本の MSI、InstallScript MSI	読み書きプロパティ	この機能のデフォルトのインストール先を取得あるいは設定します。

テーブル 68 • ISWiFeature オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ディスプレイ	基本の MSI、InstallScript MSI	読み書きプロパティ	この機能の 表示 プロパティを取得または設定します。次の定数を使用してこのプロパティを設定します。 • edtVisibleAndCollapsed (0)— デフォルトによりサブ機能が展開されずに、[カスタム セットアップ] ダイアログに機能が表示されます。 • edtVisibleAndExpanded (1)— デフォルトによりサブ機能が展開され、[カスタム セットアップ] ダイアログに機能が表示されます。 • edtNotVisible (2) —[カスタム セットアップ] ダイアログで、機能がエンドユーザーに表示されません。
DisplayName	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	この機能の表示名を取得あるいは設定します。
DisplayNameID	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	この機能の 表示名 の文字列識別子を取得または設定します。 *メモ: このプロパティを設定するには、文字列エントリが存在して、その文字列 ID が判明していなくてはなりません。*
暗号化	InstallScript	読み書きプロパティ	機能の "暗号化" プロパティを取得または設定します。このプロパティは、機能のファイルをリリース ビルダーで暗号化するかどうかを指定します。このプロパティはブール値です。
機能	基本の MSI、InstallScript、InstallScript MSI	読み取り専用プロパティ	この機能のすぐ下にサブ機能がある場合、それらのサブ機能の ISWiFeatures コレクションがこのプロパティに含まれます。
FTPLocation	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	機能の [FTP の場所] プロパティを取得または設定します。このプロパティは機能の FTP の場所を指定します。
HTTPLocation	InstallScript、InstallScript MSI	読み書きプロパティ	機能の [HTTP の場所] プロパティを取得または設定します。このプロパティは機能の HTTP の場所を指定します。

テーブル 68 • ISWiFeature オブジェクトのメンバー（続き.）

名前	プロジェクト	種類	説明
IncludeInBuild	InstallScript	読み書きプロパティ	機能の [ビルドに含める] プロパティを取得または設定します。このプロパティは機能を配布メディアに含めるかどうかを指定します。このプロパティはブール値です。
GUID	InstallScript	読み書きプロパティ	機能の GUID プロパティを取得または設定します。GUID はセットアップのログファイルにある機能を識別します。アップデートセットアップで機能のGUIDを変更する場合、GUID は既存の機能を更新したものではなく新機能としてこれを扱います。
InstallLevel	基本の MSI、InstallScript MSI	読み書きプロパティ	この機能のインストールレベルを取得あるいは設定します。
ISWiComponents	基本の MSI、InstallScript、InstallScript MSI	コレクション	現在の機能に関連するすべてのコンポーネントを含みます。
ISWiConditions	基本の MSI、InstallScript MSI	コレクション	現在の機能に関連するすべての条件を含みます。
ISWiObjects	InstallScript	コレクション	現在の機能に関連するすべての InstallScript オブジェクトを含みます。
その他	InstallScript、InstallScript MSI	読み書きプロパティ	機能の Miscellaneous プロパティを取得または設定します。このプロパティはテキスト文字列を機能に関連付けます。
Name	基本の MSI、InstallScript、InstallScript MSI	読み取り専用プロパティ	コレクション内にある現在の要素の機能名

テーブル 68 • ISWiFeature オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
Need	InstallScript	読み書きプロパティ	機能の "ファイルの必要性" プロパティを取得または設定します。このプロパティは、エンド ユーザーが機能の選択を解除したときに警告または重大な警告を表示するか、警告を表示しないかを指定します。次の定数を使用してこのプロパティを設定します。 • efnCritical (0)—エンドユーザーがセットアップで選択解除できない機能に対して、このオプションを使用します。 • efnHighlyRecommended (1) — エンドユーザーに機能をインストールすることを通知する場合、このオプションを使用します。 • efnStandard (2) — メッセージを表示せずに、エンドユーザーが機能を選択解除できるようにする場合、このオプションを使用します。
OnInstalled	InstallScript、InstallScript MSI	読み書きプロパティ	機能の OnInstalled プロパティを取得または設定します。このプロパティは、機能をインストールした後で実行する関数を指定します。
OnInstalling	InstallScript、InstallScript MSI	読み書きプロパティ	機能の OnInstalling プロパティを取得または設定します。このプロパティは、機能をインストールする前に実行する関数を指定します。
OnUninstalled	InstallScript、InstallScript MSI	読み書きプロパティ	機能の OnUninstalled プロパティを取得または設定します。このプロパティは、機能をアンインストールした後で実行する関数を指定します。
OnUninstalling	InstallScript、InstallScript MSI	読み書きプロパティ	機能の OnUninstalling プロパティを取得または設定します。このプロパティは、機能をアンインストールする前に実行する関数を指定します。
Password	InstallScript	読み書きプロパティ	機能の [パスワード] プロパティを取得または設定します。このプロパティは機能のパスワードを指定します。
ReleaseFlags	基本の MSI、InstallScript MSI	読み書きプロパティ	この機能に関連付けられたリリース フラグを取得または設定します。複数のフラグは、コンマで区切ります。 ISWiRelease の ReleaseFlags プロパティで、プロジェクトの機能をフィルターする際に使用するリリース フラグを指定できます。

テーブル 68 • ISWiFeature オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
RemoteInstallation	基本の MSI、InstallScript MSI	読み書きプロパティ	機能の リモートインストール プロパティを取得または設定します。次の定数を使用してこのプロパティを設定します。 • efrLocal (0) — 機能のファイルがすべてターゲット システムにインストールされます。 • efrSource (1) — この機能に属するファイルが、CD-ROM やネットワーク サーバーなどのソース媒体から直接実行します。 • efrFollowParent (2) — サブ機能に、その親機能の "リモート インストール" プロパティを指定します。
RemoveComponent	基本の MSI、InstallScript、InstallScript MSI	メソッド	この機能からコンポーネントを除去します。
RemoveFeature	基本の MSI、InstallScript、InstallScript MSI	メソッド	プロジェクトから指定した機能を削除します。
RemoveMergeModule	基本の MSI、InstallScript MSI	メソッド	現在の機能からマージ モジュールを削除します。
RemoveObject	InstallScript	メソッド	現在の ISWiFeature から InstallScript オブジェクトを削除します。
Required	基本の MSI、InstallScript MSI	読み書きプロパティ	機能の必須プロパティを取得または設定します。このプロパティはブール値です。
RequiredFeatures	InstallScript、InstallScript MSI	読み書きプロパティ	現在の機能の 必要な機能 プロパティで指定されている機能の名前をリストするコンマ区切り文字列を取得または設定します。

テーブル 68 • ISWiFeature オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SetupTypeStatus	InstallScript、InstallScript MSI	読み書きプロパティ	プロパティの [引数] で指定されたセットアップの種類の機能の選択状態を取得または設定します。次の定数を使用してこのプロパティを設定します。 • estSelected (10) — 機能が選択されています。 • estDeselected (11) — 機能が選択されていません。 次の Visual Studio コード スニペットでは、Help_Files 機能が Compact セットアップ タイプに選択されていません。 Set pFeature = pProject.ISWiFeatures.Item("Help_Files") Set pSetupType = pProject.ISWiSetupTypes.Item("Compact") pFeature.SetupTypeStatus(pSetupType) = 11
StatusText	InstallScript	読み書きプロパティ	機能の "ステータス テキスト" プロパティを取得または設定します。このプロパティは、機能の転送中に進行状況インジケーターの最上行からエンドユーザーに表示されるテキストを指定します。
Visible	InstallScript	読み書きプロパティ	機能の [可視] プロパティを取得または設定します。このプロパティはインストール中に機能ダイアログ ボックスで機能を可視にするかどうかを指定します。このプロパティはブール値です。

次に適用：

- ISWiProject

AddCondition メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddCondition メソッドを呼び出して、現在の機能に条件を追加します。

構文

AddCondition (sKey As String) As ISWiCondition

パラメーター

テーブル 69 • AddCondition メソッド パラメーター

パラメーター	説明
sKey	現在の機能に追加する条件の文字列バージョンを渡します。

次に適用：

- ISWiFeature

AddMergeModule メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddMergeModule メソッドを呼び出して、マージ モジュールを現在の機能と関連付けます。このメソッドは、マージ モジュールを [再配布可能ファイル] ビュー の機能に追加する方法と同様です。

構文

AddMergeModule (FileName As String)

パラメーター

テーブル 70 • AddMergeModule メソッド パラメーター

パラメーター	説明
FileName	この機能と関連付けするマージ モジュール (.msm) ファイルの完全なファイル名を渡します。

次に適用：

- ISWiFeature

AddObject メソッド

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

AddObject メソッドは、指定のモニカと一緒に InstallScript オブジェクトを現在の機能に追加します。

構文

AddObject (Moniker As String) As ISWiObject

パラメーター

テーブル 71 • AddObject メソッド パラメーター

パラメーター	説明
Moniker	追加する InstallScript オブジェクトのモニカを渡します。InstallScript オブジェクトのモニカを取得するには、IDE で新しい Professional プロジェクトを作成して InstallScript オブジェクトを機能に追加し、[ダイレクト エディター] ビューの ISFeatureExtended テーブルに移動して Moniker 列を参照します。

例

次の Visual Basic 行では、AddObject メソッドを示します:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pObject As ISWiObject

Set pFeature = pProj.ISWiFeatures.Item(2)
Set pObject = pFeature.AddObject ("@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9")

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiFeature

AttachComponent メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

AttachComponent メソッドを呼び出して、コンポーネントを現在の機能と関連付けます。機能と関連付けするには、コンポーネントが作成されていなければなりません (IDE で手動によって作成、または AddComponent を呼び出して作成)。

構文

AttachComponent (Component As ISWiComponent)

パラメーター

テーブル 72 • AttachComponent メソッド パラメーター

パラメーター	説明
コンポーネント	この機能と関連付けするコンポーネントを示す ISWiComponent オブジェクトを指定します。

次に適用：

- ISWiFeature

DeleteCondition メソッド

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

DeleteCondition メソッドを呼び出して、現在の機能から条件を削除します。

構文

DeleteCondition (pCondition As ISWiCondition)

パラメーター

テーブル 73 • DeleteCondition メソッド パラメーター

パラメーター	説明
pCondition	ISWiCondition オブジェクトは、削除する条件を表します。

次に適用：

- ISWiFeature

RemoveComponent メソッド

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript*
- *InstallScript MSI*

指定した機能からコンポーネントを削除するには、RemoveComponent を呼び出します。RemoveComponent は、[セットアップのデザイン] ビューでコンポーネントを削除するのと同様に、機能とコンポーネントの関連付けを解除しますが、プロジェクトから完全にコンポーネントを削除する訳ではありません。

構文

RemoveComponent (pComponent As ISWiComponent) As Boolean

パラメーター

テーブル 74 • RemoveComponent メソッド パラメーター

パラメーター	説明
pComponent	削除する ISWiComponent オブジェクトを入力します。

次に適用：

- ISWiFeature

RemoveMergeModule メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

RemoveMergeModule メソッドを呼び出して、マージ モジュールを現在の機能から削除します。(プロジェクトからマージ モジュールを削除する場合は、DeleteMergeModule メソッドを呼び出します。)

構文

RemoveMergeModule (FileName As String)

パラメーター

テーブル 75 • RemoveMergeModule メソッドのパラメーター

パラメーター	説明
FileName	この機能から削除するマージ モジュール (.msm) ファイルの完全なファイル名を渡します。

次に適用：

- ISWiFeature

RemoveObject メソッド

プロジェクト: この情報は、InstallScript プロジェクトに適用します。

RemoveObject メソッドは、指定の InstallScript オブジェクト を現在の機能から削除します。

構文

RemoveObject (Object As ISWiObject) As Long

パラメーター

テーブル 76 : RemoveObject メソッドのパラメーター

パラメーター	説明
オブジェクト	削除される InstallScript オブジェクト

例

ComVisual Basic 行では、RemoveObject メソッドを示します:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pObject As ISWiObject

Set pFeature = pProj.ISWiFeatures.Item("Help_Files")
Set pObject = pFeature.ISWiObjects.Item(1)
pFeature.RemoveObject pObject

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiFeature

ISWiFile オブジェクト

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiFile は、コンポーネントに属するファイルか、InstallScript プロジェクトの場合はプロジェクトのコンポーネント サブフォルダーを表します。ファイルは、InstallShield ユーザー インターフェイスの既存のファイル、または AddFile を呼び出して追加したファイルのどちらでも構いません。

機能プロパティと属性の多くは、このオブジェクトのメソッド、プロパティ、コレクションを通してオートメーション インターフェイスで使用できます。いくつかのメソッド、プロパティ、コレクションは、特定のプロジェクトの種類でのみ使用できます。

コンポーネントの ISWiFiles コレクションを使用して特定のファイルにアクセスします。ISWiFiles の Item プロパティには、インデックス番号かファイルキーのいずれかを指定します。ファイル キーは、InstallShield の [ダイレクト エディター] ビューにある File テーブルの File 列で見ることができます。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合、コンポーネントのファイル リスト [キー] 列にも一覧表示されています。

ただし、ファイル キーは、値が分からない場合や、コンポーネントと動的にリンクされているファイルでは、再割り当てされている場合もあります。ファイル キーの代わりに、次の例のようなコードを記述して、特定のファイル名の存在を確認することができます。

```
Dim m_pFile As ISWiFile

For Each m_pFile In m_pComponent.ISWiFiles
    If UCase(m_pFile.DisplayName) = "MYFILE.EXE" Then
        Exit For
    End If
Next
```

ファイル名 MyFile.exe がコンポーネントのファイル中に見つかると For ループが終了しますが、この時 m_pFile には MyFile.exe のISWiFile オブジェクトがコピーされています。

メンバー

テーブル 77 • ISWiFile オブジェクトのメンバー

名前	プロジェクト	種類	説明
常に上書き	基本の MSI、InstallScript MSI、マージ モジュール	**読み書きプロパティ**	Windows Installer が選択されたファイルの実際のバージョン番号を無視して、その代わりにターゲット システムにそのファイルが既存する場合、常に上書きを試みるように設定するには、このチェック ボックスを選択します。実行時、このファイルがターゲット システム上にあるファイルと同じ名前および同じターゲット場所を持つ場合、Windows Installer がターゲット システムのファイルを現在のインストールに含まれているバージョンでアップデートするか、ファイルをそのままに残すかを決定するとき、インストールに含まれているファイルのバージョンを 65535.0.0.0 と判断します。 Windows Installer が既存ファイルを上書きするかどうかを判断する方法についての詳細は、「ターゲット システム上でファイルとコンポーネントを上書きする」を参照してください。 ファイルの最大バージョン番号は、65535.65535.65535.65535 です。
Attributes	基本の MSI、InstallScript MSI	プロパティ（使用不可）	このプロパティは、サポートされなくなりました。Hidden、ReadOnly、System および Vital プロパティを代わりに使用してください。
CompanionFile	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	後で使用するために予約されています。
DisplayName	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ファイルの標準の名前です。パッケージをインターネットで配布する場合、一部の Web サーバーは大文字小文字を区別するため、このエントリの大文字小文字がファイル名の大文字小文字と一致しているか確認してください。名前に、Windows のフォルダーやファイル名に有効でない文字を含めることはできません。 ファイルに長いファイル名と短いファイル名の 2 つがある場合（下記の ShortName を参照してください）、長いファイル名だけが表示名に使われます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。

テーブル 77 • ISWiFile オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DynamicFile	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み取り専用プロパティ	ファイルのソースがコンポーネントに動的にリンクされている場合はこのプロパティは True にし、静的にリンクされている場合は False とします。
FontTitle	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	フォントタイトルファイルの属性を文字列として読み込むか設定します。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
FullPath	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ファイルのソースの場所の完全修飾パス (ファイル名を含む) を保存します。あらゆるパス変数は、実際のパスへ解決されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
Hidden	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	True になるのは、ファイルが非表示の属性セットを持っている場合です。 OverrideSystemAttributes が False に設定されている場合、このプロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
Languages	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティは、ファイルの言語 ID のすべての文字列をカンマで区切った10進数で、含む必要があります。次の行は、ファイルを英語とドイツ語に識別します。 m_pFile.Languages = ″1031,1033″ OverrideSystemLanguage が True に設定されていると、このプロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
Modified	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ファイルの修正済みファイル属性を読み込みます。後で使用するために予約されています。

テーブル 77 • ISWiFile オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
名前	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ファイルキーの読み込みまたは設定を行います。ファイルキーの詳細については、上記の ISWiFile の説明を参照してください。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
OverrideSystemAttributes	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	Hidden、ReadOnly、Vital、System など、すべてのファイル属性で開発システムの設定をオーバーライドするには、このプロパティを True に設定します。 このプロパティを False に設定すると、Hidden プロパティ、ReadOnly プロパティ、System プロパティ、および Vital プロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
OverrideSystemLanguage	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	開発システムのファイル言語をオーバーライドするには、このプロパティを True に設定します。 このプロパティを False に設定すると、Languages プロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
OverrideSystemSize	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	開発システムのファイルのサイズをオーバーライドするには、このプロパティを True に設定します。 このプロパティを False に設定すると、Size プロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
OverrideSystemVersion	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	開発システムのファイル バージョンのバージョンをオーバーライドするには、このプロパティを True に設定します。 このプロパティを False に設定すると、Version プロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。

テーブル 77 • ISWiFile オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ReadOnly	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	読み取り専用ファイル属性を読み込むか設定します。ファイルを読み取り専用にマークするには、このプロパティを True に設定します。ファイルを読み取り – 書き込みにマークするには、未定義にしておくかFalseに設定します。 OverrideSystemAttributes が False に設定されている場合、このプロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
SelfRegister	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	自己登録 ファイル属性 を取得または設定します。このファイルを自動登録とマークするには、このプロパティをTrueに設定します。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
ShortName	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ファイルの短い名前を取得または設定します。配布メディアは長いファイル名を扱えないため、Windows Installer のプロパティ SHORTFILENAMES が設定されていると、ファイル名は対応する短い名前が必要になります。 リリース ウィザードの [詳細設定] パネルの中で [長いファイル名の使用] を選択解除した場合、短いファイル名だけが使用されます。 デフォルトでビルドプロセスによってファイルに短い名前が付けられるため、ファイルの ShortName プロパティは空の文字列を返します。ただしファイルの ShortName プロパティを明示的に設定した場合、ビルドプロセスでは指定された短い名前が使用され、ShortName の値を読み取ると指定された短い名前が返されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
Size	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ファイルのサイズがバイト単位で、長整数型で表示されます。 このプロパティを設定する場合、まず OverrideSystemSize を True に設定する必要があります。設定しない場合、InstallShield はファイルの実際のサイズを使用します。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。

テーブル 77 • ISWiFile オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
System	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	システムのファイル属性の読み込みまたは設定を行います。ファイルがシステムファイルの場合は、このプロパティをTrueに設定します。 OverrideSystemAttributes が False に設定されている場合、このプロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
UseSystemSettings	基本の MSI、InstallScript MSI	Property (旧形式)	このプロパティは、サポートされなくなりました。代わりに、OverrideSystemAttributes プロパティ、OverrideSystemSize プロパティ、OverrideSystemVersion プロパティ、および OverrideSystemLanguage プロパティを使用してください。
バージョン	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティを、セットアップに含めるファイルのバージョンに設定します。 コンパニオン ファイル を指定するために、このプロパティを別の ISWiFile オブジェクトの Name プロパティに設定することもできます。コンパニオン ファイルに関する詳しい情報は、「コンパニオン ファイル」をご覧ください。 OverrideSystemVersion が False に設定されている場合、このプロパティは無視されます。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。
Vital	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	必須ファイル属性の読み込みまたは設定を行います。このファイルを必須 (vital) とマークするには、このプロパティを True に設定します。 ファイルが動的にリンクされている場合、このプロパティを変更することはできません。

ISWiFolder オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiFolder は、InstallShield ユーザー インターフェイス内のコンポーネントのショートカット エクスプローラー内にあるプログラムフォルダーを表します。ISWiFolders コレクションの中の項目を指定して、フォルダーを呼び出すことができます。

フォルダーの属性の多くは、このオブジェクトのメソッドとプロパティを通してオートメーション インターフェイスで使用できます。使用中のプロジェクトの種類に使用できるメソッド、プロパティ、コレクションについての詳細は、次のテーブルを参照してください。

メンバー

テーブル 78 • ISWiFolder オブジェクトのメンバー

名前	種類	説明
AddShortcut	メソッド	ショートカットオブジェクトを現在のフォルダーオブジェクトに追加します。
AddSubFolder	メソッド	現在のプログラムフォルダーの下に、新たにプログラムフォルダーを作成します。
DeleteShortcut	メソッド	現在のフォルダーから指定されたショートカットを削除します。
DeleteSubFolder	メソッド	現在のフォルダーから指定されたサブフォルダーを削除します。
説明	読み書きプロパティ	フォルダーに対して入力した説明文を含みます。
DisplayName	読み書きプロパティ	フォルダーに表示される名前を読み出す、または設定します。
ISWiShortcuts	コレクション	現在のプログラムフォルダーの下にあるすべてのショートカットを含みます。
名前	読み書きプロパティ	フォルダーのキー名を取得します。これは、InstallShield ユーザー インターフェイスの [ショートカット] ビューに表示される名前です。DisplayName プロパティと同じではありません。
SharedFolder	読み書きプロパティ	このブール型プロパティは、フォルダーの [共有] プロパティを取得または設定します。このプロパティはフォルダーを常にシステムから削除するか (False)、セットアップで作成したサブフォルダーまたはファイルがない場合にのみ削除される (True) かを指定します。
SubFolders	コレクション	ここには、ショートカットエクスプローラー上で現在のフォルダーの直下にある個々のプログラムフォルダーを示す ISWiFolders コレクションが格納されます。このプロパティを使用したサブフォルダーの取り出し方については、ISWiFolders コレクション を参照してください。 ***メモ:*** *このプロパティは、InstallScript プロジェクトには適用されません。*

次に適用 :

- ISWiComponent

AddShortcut メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

AddShortcut を呼び出して、現在のプログラムフォルダー内にショートカットを作成します。

構文

AddShortcut (Name As String) As ISWiShortcut

パラメーター

テーブル 79 : AddShortcut メソッドのパラメーター

パラメーター	説明
名前	[ショートカット] エクスプローラーでショートカットに付ける名前を入力します。

次に適用 :

- ISWiFolder

AddSubFolder メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

AddSubFolder メソッドを呼び出して、現在の ISWiFolder オブジェクトの下位にプログラム フォルダーを作成します。

構文

AddSubFolder (Name As String) As ISWiFolder

パラメーター

テーブル 80 ・ AddSubFolder メソッドのパラメーター

パラメーター	説明
名前	[ショートカット] エクスプローラーでプログラムフォルダーに付ける名前を入力します。

次に適用：

- ISWiFolder

DeleteShortcut メソッド

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

DeleteShortcut を呼び出して、現在のフォルダーからショートカットを作成します。

構文

DeleteShortcut (pShortcut As ISWiShortcut) As Long

パラメーター

テーブル 81 ・ DeleteShortcut メソッド パラメーター

パラメーター	説明
pShortcut	ISWiShortcut オブジェクト を渡して、フォルダーから削除するショートカットを指定します。

例

```
Set pShortcut = pFolder.ISWiShortcuts("MyShortcut")
If Not pShortcut Is Nothing Then
  pFolder.DeleteShortcut pShortcut
Next
```

次に適用：

- ISWiFolder

DeleteSubFolder メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

DeleteSubFolder メソッドを呼び出して、現在の ISWiFolder オブジェクトから指定されたサブフォルダーを削除します。

構文

DeleteSubFolder (pSubFolder As ISWiFolder) As Long

パラメーター

テーブル 82 - DeleteSubFolder メソッド パラメーター

パラメーター	説明
名前	ISWiFolder オブジェクト を渡して、フォルダーから削除するサブフォルダーを指定します。

例

```
Set pSubFolder = pFolder.SubFolders("MySubFolder")
If Not pSubFolder Is Nothing Then
    pFolder.DeleteSubFolder pSubFolder
Next
```

次に適用：

- ISWiFolder

ISWiLanguage Object

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiLanguage は、現在のプロジェクトに含まれる文字列エントリを含む言語を表します。

言語を含む、または除外するには、それぞれの関連 ISWiLanguage オブジェクトに IsIncluded プロパティを使用します。

メンバー

テーブル 83 : ISWiLanguage オブジェクトのメンバー

名前	種類	説明
AddStringEntry	メソッド	新しい文字列エントリを言語に追加します。
DeleteStringEntry	メソッド	文字列エントリをプロジェクトから削除します。
Id	読み取り専用プロパティ	現在の言語の言語 ID を格納します。
IsIncluded	読み書きプロパティ	言語を含む、除外する、または現在の状態を確認します。
ISWiStringEntries	コレクション	現在の言語のすべての文字列エントリが含まれています。
Name	読み取り専用プロパティ	現在の言語の名前を取得します。

次に適用 :

- ISWiProject

AddStringEntry メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

AddStringEntry を呼び出して、新しい文字列エントリを言語に追加して、ISWiStringEntry を返します。

構文

AddStringEntry (sLangId) As ISWiStringEntry

パラメーター

テーブル 84 • AddStringEntry メソッドのパラメーター

パラメーター	説明
sLangId	文字列エントリを格納する言語の識別子を指定します。

次に適用：

- ISWiLanguage

DeleteStringEntry メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

DeleteStringEntry を呼び出して、言語から文字列エントリを削除します。

構文

DeleteStringEntry (pLangId As ISWiStringEntry)

パラメーター

テーブル 85 • DeleteStringEntry メソッドのパラメーター

パラメーター	説明
pLangId	ISWiStringEntry オブジェクトを指定して、プロジェクトから削除する文字列エントリを指定します。

次に適用：

- ISWiLanguage

ISWiLatestSetup オブジェクト

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

パッチ構成の [最新セットアップ] (または UpgradedImage) 項目で、インストールの最新バージョンの設定を指定します。

パッチを作成するためには、各パッチ構成に少なくとも最新セットアップが 1 つ、および以前のセットアップが 1 つ含まれている必要があります。最新のセットアップごとに以前のバージョンを複数指定することもできます。

最新のセットアップにパッチをあてる予定のアプリケーションまたはセットアップはすべて、以前のセットアップ ノードに追加する必要があります。

たとえば、アプリケーションのバージョン 1.0 および 1.1 をサポートする場合で、両方のアプリケーションをバージョン 2.0 にアップグレードすると想定します。両方のバージョンをアップグレードする単一のパッチを作成するには、最新のセットアップに以前のセットアップとして V1.0 と V1.1 を追加します。最新セットアップには V2.0 が入ります。

メンバー

テーブル 86 • ISWiLatestSetup オブジェクトのメンバー

名前	種類	説明
AddPatchExternalFile	メソッド	ISWiPatchExternalFile オブジェクトを現在の最新セットアップ構成に追加します。 追加外部ファイルをパッチすると、以前のセットアップの一部として出荷されなかったアプリケーションファイルのバージョンに対するバイナリファイルパッチを作成することが可能となります。たとえば、アプリケーション ファイルの 1 つが他のセットアップにより更新されている可能性がある場合、そのファイル バージョンを参照して、実行時にファイル レベルのパッチが確実に適用されるようにします。
AddPatchWholeFile	メソッド	ISWiPatchWholeFile オブジェクトを現在の最新セットアップ構成に追加します。 すべてのファイルを完全な形で含めるには、パッチ構成の IncludeWholeFilesOnly プロパティを [True] に設定します。
AddPreviousSetup	メソッド	ISWiPreviousSetup オブジェクトを現在の最新セットアップ構成に追加します。
DeletePatchExternalFile	メソッド	指定された外部ファイル オブジェクトを、現在の最新セットアップ構成から削除します。
DeletePatchWholeFile	メソッド	指定された完全なファイル オブジェクトを、現在の最新セットアップ構成から削除します。
DeletePreviousSetup	メソッド	指定された以前のセットアップ オブジェクトを、現在の最新セットアップ構成から削除します。
ISWiPatchExternalFiles	コレクション	この最新セットアップに所属する全ての外部ファイル オブジェクトを含みます。
ISWiPatchWholeFiles	コレクション	この最新セットアップに所属するすべての完全なファイル オブジェクトを含みます。
ISWiPreviousSetups	コレクション	この最新セットアップ (またはアップグレードされたイメージ) に所属するすべての以前のセットアップを含みます。
MsiPath	読み取り専用プロパティ	最新セットアップパスを取得または設定します。 最新セットアップは、パッチを作成するインストールの最新バージョンでなければなりません。デフォルト設定 <ISLatestRelease> は、ビルドする最新の完全リリースに解決します。 このプロパティで指定したリリース パスが、非圧縮状態であることを確認してください。前回の製品バージョンが圧縮の場合、できるかぎり管理インストールで作成します。

テーブル 86 • ISWiLatestSetup オブジェクトのメンバー (続き.)

名前	種類	説明
名前	読み取り専用プロパティ	最新セットアップ項目の名前を取得します。

次に適用 :

- ISWiPatchConfig

AddPreviousSetup メソッド

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

AddPreviousSetup メソッドを呼び出して、最新セットアップ項目に以前のセットアップ項目を追加します。このメソッドは、IDE の [パッチのデザイン] ビューで以前のセットアップ項目を追加する方法に似ています。ISWiPreviousSetup オブジェクトを使って、新しい以前のセットアップのプロパティを設定できます。

構文

AddPreviousSetup(PreviousSetupName As String) As ISWiPreviousSetup

パラメーター

テーブル 87 • AddPreviousSetup メソッドのパラメーター

パラメーター	説明
PreviousSetupName	以前のセットアップ項目の名前を保持します。

次に適用 :

- ISWiLatestSetup

DeletePreviousSetup メソッド

プロジェクト- この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

DeletePreviousSetup メソッドを呼び出して、最新セットアップ構成から指定された以前のセットアップ オブジェクトを削除します。

構文

DeletePreviousSetup (PreviousSetup As ISWiPreviousSetup) As Long

パラメーター

テーブル 88 • DeletePreviousSetup メソッドのパラメーター

パラメーター	説明
PreviousSetup	削除する以前のセットアップ オブジェクトを保持します。

次に適用：

- ISWiLatestSetup

ISWiMsiPatchSequence オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

他のパッチに比べて比較的小さいパッチのアプリケーション シーケンスを決定するためにインストーラーが必要とするすべての情報を含みます。

メンバー

テーブル 89 • ISWiMsiPatchSequence オブジェクトのメンバー

名前	種類	説明
FamilyName	読み書きプロパティ	パッチ ファミリー名を取得します。同じ製品バージョンをターゲットにする同じパッチ ファイミリーに含まれるパッチは、[シーケンス] 列の値の順番に表示されます。パッチ ファミリー内のパッチは、シーケンスの昇順にターゲット製品に適用されます。PatchFamily はまた、優先するパッチを判別するのにも使用されます。1 つのパッチが複数の製品に適用される、またはパッチに複数の修正が含まれている場合、パッチが複数行に表示され、複数のパッチ ファミリーに所属する場合があります。 Windows Installer は PatchFamily 値を他の PatchFamily 値に対して等しいかどうかを比較する以外の解釈を行いません。PatchFamily 値は、一連のパッチによってターゲットされる ProductCode 内において一意でなくてなりません。複雑なパッチ シナリオの場合、PatchFamily 識別子がグローバルに一意でなくてはならない場合があります。
Sequence	読み書きプロパティ	指定された PatchFamily 内でこのパッチのシーケンスを取得または設定します。シーケンスの値は、バージョン データのフォーマットで表示されます。値は 1 から 4 のフィールドを含み、各フィールドの値範囲は 0 から 65535 です。PatchFamily のメンバーは、シーケンス値の昇順で、ターゲット製品に順番に適用されます。たとえば、次の 6 つの値は降順です: 1、1.1、1.2、2.01、2.01.1、2.01.1.1。
Supersede	読み書きプロパティ	このブール値プロパティを Ture に設定して、同じ PatchFamily 内でより小さいシーケンス値を持つすべてのパッチが提供するアップデートを、小さいアップデート パッチが優先することを示します。
Target	読み書きプロパティ	ターゲット製品コードを取得または設定します。このプロパティの値はオプションです。このプロパティで製品コード GUID が設定されていて、パッチが特定の製品に適用された場合、パッチは指定された PatchFamily のメンバーとして順番に適用されます。製品コード GUID がこのフィールドに設定されていて、パッチが ProductCode で指定された製品に適用されない場合、この行は無視されます。ProductCode の値が NULL の場合、パッチは製品コードにかかわらず、パッチのすべてのターゲットの PatchFamily のメンバーとして順番に適用されます。 パッチは、同じ PatchFamily 内で複数の行を持ち、ターゲットされる各製品の異なる ProductCode を含む場合があります。PatchFamily のうちの 1 行で、ProductCode を NULL に指定できます。ターゲット製品が NULL 以外の ProductCode を持つ行と一致した場合、インストーラーはその一致する行を使用し、NULL ProductCode を含む行を無視します。指定された製品コードがターゲットと一致しなかった場合、パッチは製品コードにかかわらず、パッチのすべてのターゲットの PatchFamily のメンバーとして順番に適用されます。

次に適用：

- ISWiProject

ISWiObject オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

ISWiObject は、プロジェクトに属する InstallScript オブジェクトを表します。セットアップの種類は、InstallShield ユーザー インターフェイスの [オブジェクト] ビューにある既存の InstallScript オブジェクトでも、AddObject を呼び出して追加したものでもかまいません。

プロジェクトの ISWiObject コレクションを使用して、特定の InstallScript オブジェクトにアクセスします。ISWiObjects の Item プロパティには、インデックス番号かモニカのいずれかを指定します。モニカは、[ダイレクト エディター] ビューの ISFeatureExtended テーブルの [モニカ] としてのみ IDE で表示できます。

メンバー

なし。

例

次の Visual Basic コードは ISWiObject オブジェクトの使用方法を示しています:

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pObject As ISWiObject

Set pFeature = pProj.ISWiFeatures.Item(2)
Set pObject = pFeature.ISWiObjects.Item("@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9")
pFeature.RemoveObject pObject

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiFeature

ISWiPatchConfig オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiPatchConfig は、InstallShield ユーザー インターフェイスの [パッチのデザイン] ビューないのパッチ構成を表します。ISWiPatchConfigs コレクションの中の項目を指定して、構成を取得できます。

ビルドのステータスイベント

ISWiPatchConfig オブジェクトでは次のステータスイベントが発生します:

- Public Event ProgressIncrement(ByVal lIncrement As Long, pbCancel As Boolean)
- Public Event ProgressMax(ByVal lMax As Long, pbCancel As Boolean)
- Public Event StatusMessage(ByVal sMessage As String, pbCancel As Boolean)

これらのイベントは、パッチ のビルド プロセス中に発生してステータスのアップデートを提供します。ビルドを中止するには pbCancel を設定します。これらのイベントの使い方をデモンストレーションするサンプル コードは、「ビルドのステータス イベント」を参照してください。

機能プロパティと属性の多くは、このオブジェクトのメソッド、プロパティ、コレクションを通してオートメーション インターフェイスで使用できます。以下のテーブルでは、ISWiPatchConfig オブジェクトのメンバーが一覧表示されています。

名前	種類	説明
AddLatestSetup	メソッド	ISWiLatestSetup オブジェクトを現在のパッチ構成に追加します。
AddMsiPatchSequence	メソッド	ISWiMsiPatchSequences オブジェクトを現在のパッチ構成に追加します。
AllowProductCodeMismatches	読み書きプロパティ	製品コードが元と最新のインストールで異なる場合、パッチ作成実行可能ファイルにビルド時のプロンプトを抑制させるかどうかを指定するには、このプロパティを True に設定します。メジャー アップグレードでは、必ず True に設定してください。
AllowProductVersionMajorMismatches	読み書きプロパティ	製品バージョン規則が元と最新のインストールで異なる場合、パッチ作成実行可能ファイルにビルド時のプロンプトを抑制させるかどうかを指定するには、このプロパティを True に設定します。
AllowRemoval	読み書きプロパティ	アプリケーション全体と他のパッチをアンインストールして再度インストールし直す手間を掛けずにパッチをアンインストールできるようにしたいとき、このプロパティを True に設定します。 さらに詳しい情報は、Windows Installer ヘルプの「Removing Patches」を参照してください。
BuildPatch	メソッド	このメソッドを呼び出して、現在のパッチ構成をビルドします。このアクションは、[パッチのデザイン] ビューでパッチ構成を右クリックしてから 'パッチのビルド' を選択した場合と同じです。

名前	種類	説明
分類	読み書きプロパティ	アップグレード のカテゴリを取得または設定します。たとえば、クリティカルアップデート、ホットフィックス、および、サービスパック等。
CreateUpdateLauncher	読み書きプロパティ	現在のパッチに Update.exe アップデート ランチャーを作成する場合、このプロパティを True に設定します。
CryptographicProvider	読み書きプロパティ	暗号化サービス プロバイダー (CSP) を取得または設定します。 ***メモ:*** *暗号化サービス プロバイダー (CSP) 名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*
CustomSignArgs	読み書きプロパティ	署名ツール構成の引数を取得または設定して、コマンドライン引数を指定して、パッチ パッケージおよび Update.exe ファイルにデジタル署名を行います。 たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。 sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename] [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクトで定義されている任意のパス変数または環境変数を使用することができます。 ***メモ:*** ***SigningType*** *プロパティの値が* ***1*** *の場合に、このプロパティを使用します。*

名前	種類	説明
CustomSignPath	読み書きプロパティ	パッチ パッケージおよび Update.exe ファイルにデジタル署名を行うために使用する署名ツールの場所を取得または設定します。 ***メモ: SigningType プロパティの値が 1 の場合に、このプロパティを使用します。***
DeleteLatestSetup	メソッド	指定された最新のセットアップ オブジェクトを、現在のパッチ構成から削除します。
DeleteMsiPatchSequence	メソッド	指定されたパッチ シーケンス オブジェクトを、現在のパッチ構成から削除します。
説明	読み書きプロパティ	パッチに簡単な説明を取得または設定します。
DisableDefaultSequencing	読み書きプロパティ	製品のパッチにデフォルトのシーケンスを使用したいとき、このプロパティを False にを設定します。このデフォルトのシーケンスを参照して、もう使われていないパッチ、入れ替わったパッチ、または製品に既に適用済みのパッチを見ることができます。 このプロパティを True に設定すると、カスタム シークエンスを追加することができます。これを True してシーケンスを追加しなかった場合、シーケンス情報はパッチに組み込まれません。
DisplayName	読み書きプロパティ	パッチに名前を取得または設定します
EnablePatchCache	読み書きプロパティ	このプロパティを True に設定すると、InstallShield はパフォーマンス (速度) 向上のため、後にに続くビルドで使用される中間ファイルを作成します。これらの中間ファイルは無期限にシステム上に残ります。ディスク容量が限られている場合はこのチェック ボックスを選択しないで下さい。
GenerateMsiPatchOldAssembly	読み書きプロパティ	MsiPatchOldAssemblyFile および MsiPatchOldAssemblyName テーブルのエントリを自動的に生成する場合、このプロパティを True に設定します。これらのエントリを利用すると Windows Installer 3.0 以降で実行されるパッチ パッケージが元のインストール ソースを実行時に要求することなく、GAC (グローバル アセンブリ キャッシュ) のアセンブリをパッチすることができます。

名前	種類	説明
GeneratePatchGuid	読み書きプロパティ	パッチのビルド時、毎回、新しい GUID を生成するか場合、このプロパティを True に設定します。[いいえ] を選択すると、"パッチ GUID" プロパティにリストされた GUID が使用されます。デフォルト設定は True です。
IncludeWholeFilesOnly	読み書きプロパティ	パッチ パッケージに含まれている各ファイルに対してファイル全体を含めるには、このプロパティを True に設定します。
ISWiLatestSetups	コレクション	このパッチ構成に属するすべての最新セットアップを含みます
ISWiMsiPatchSequences	コレクション	このパッチ構成に属するすべてのパッチ シーケンスを含みます。
KeyContainerName	読み書きプロパティ	暗号化サービス プロバイダー (CSP) に関連付けられた秘密キー コンテナ名を取得または設定します。 ***メモ:*** *コンテナ名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*
LauncherCompanyName	読み書きプロパティ	会社名を取得または設定します。これによって、Update.exe のデフォルトの会社名が指定された会社名で上書きされます。 会社名が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。

名前	種類	説明
LauncherCopyright	読み書きプロパティ	ランチャーの著作権情報を取得または設定します。Update.exe のデフォルトの著作権情報を独自の製品著作権情報でオーバーライドするには、ここに製品の著作権情報を入力します。 著作権情報が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください/
LauncherDescription	読み書きプロパティ	ランチャーの説明を取得または設定します。Update.exe のデフォルトの説明を独自の説明でオーバーライドするには、ここに適切な説明を入力します。 説明が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックしたときに表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
LauncherIcon	読み書きプロパティ	Update.exe ファイルのアイコンを取得または設定します。Update.exe ファイルに独自のアイコンを使用する場合、そのアイコンを含むファイルの完全修飾名を設定します。ファイルを指定するには、完全修飾パスまたはパス変数の相対パスを入力してください。 デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、[アイコンの変更] ダイアログ ボックスでアイコンを選択するか、アイコンのインデックスまたはリソース ID (それぞれ前にマイナス (-) 記号がついている) をファイル名に付け足します。たとえば、C:¥Temp¥MyLibrary.dll,2 は 2 というインデックスを持つアイコンを、C:¥Temp¥MyLibrary.dll,-100 は 100 と言うリソース ID を持つアイコンを示します。 この設定を空白のままにした場合、Update.exe ファイルにデフォルトのアイコンが使用されます。

名前	種類	説明
LauncherPassword	読み書きプロパティ	アプリケーションを保護するパスワードを取得または設定します。PasswordProtected プロパティを True に設定してパスワード保護をアクティベートする必要があります。 パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。 この設定は Update.exe ファイルを使用するパッチのみに適用します。
LauncherProductName	読み書きプロパティ	製品名を取得または設定します。これによって、Update.exe のデフォルトの製品名が指定された製品名で上書きされます。 製品名が、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
LauncherProductVer	読み書きプロパティ	製品バージョンを取得または設定します。Update.exe のデフォルトの製品バージョンを独自の製品バージョンでオーバーライドするには、ここに製品バージョンを入力します。 製品バージョンが、アップデート起動ツールの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Update.exe ファイルを右クリックしてから、[プロパティ] をクリックすると表示されます。 詳細については、「アップデート起動ツールのファイルのプロパティをカスタマイズする」を参照してください。
LeaveIntermDecompressed	読み書きプロパティ	すべての .msp ファイル情報を一時フォルダーに残しておく場合、True に設定します。トラブルシューティングをするとき、この情報はとても役に立ちます。この情報を保存しない場合は、[いいえ] を選択します。

名前	種類	説明
ListOfPatchGuidsToReplace	読み書きプロパティ	現在のパッチで、1 つまたは複数の以前のインストール済みのパッチを置換するには、このプロパティにそれらのパッチのパッチ GUID をカンマで区切って設定します。例: {C86838C9-DEDC-4451-B96F-94AFB9460F15},{C8633E5B-AC44-45d8-B487-C68B3B1F60D6} [パッチのデザイン] ビューにパッチの構成が複数あっても、このプロパティの設定は通常、必要ありません。ただし パッチが以前の パッチに追加されたファイルを上書きしない場合、このプロパティの設定が必要な場合もあります。
ListOfTargetProductCodes	読み書きプロパティ	このプロパティは、このパッチでターゲットする製品コードを、コンマで区切って設定します。この設定にアスタリスク (*) を入力すると、InstallShield がビルド時に、パッチの構成で以前のイメージとしてリストされたインストールの製品コードで置き換えます。
ManufacturerName	読み書きプロパティ	アプリケーションの製品メーカーの名前を取得または設定します。
MinimumRequiredMsiVersion	読み書きプロパティ	パッチに必要な Windows Installer の最小バージョンを指定します。たとえば、Windows Installer 2.0 を最小バージョンとする場合、**200** と入力します。Windows Installer 3.0 を最小バージョンとする場合、**300** と入力します。Windows Installer 3.1 を最小バージョンとする場合、**301** と入力します。

名前	種類	説明
MinorUpdateTargetRTM	読み書きプロパティ	MSI 3.1 必須 現在のマイナーアップグレード パッチを適用する前に、マイナー アップグレード パッチで製品の RTM (Release to Manufacturing) バージョンまでのすべてのパッチ (製品の最新メジャー アップグレードがインストールされている場合は、そのメジャー アップグレード) を実質的に削除する場合、このプロパティを True に設定します。このオプションを選択すると、すべてのパッチが、シーケンシング データがあるないにかかわらず、削除されます。追加のベースライン バージョンをターゲットにする必要はないので、結果的にパッチのペイロードを増やすことになります。すべてのエンドユーザーは、中間のパッチを適用することなく、正常にこのパッチを適用することができます。 マイナーアップグレードによって以前のすべてのパッチを削除させない場合は、False を設定します。このオプションを選択すると、場合により、RTM (または、存在する場合、製品の最新メジャーアップグレード) の後に作成された以前の各マイナーアップグレードをターゲットするのに必要な情報をパッチに含める必要があります。 たとえば、SP 2 のマイナー アップグレード パッチを作成していて、このプロパティで [いいえ] を選択した場合、パッチは SP 1 のマイナー アップグレードをターゲットにする必要があります。またオプションで、他のベースライン (RTM など) をターゲットにすることもできます。これにより、ペイロードが増加します。RTM バージョンをターゲットにしない場合、RTM バージョンを持っているが SP 1 のマイナー アップパッチがないエンドユーザーは、SP 2 の前に SP 1 をインストールしなければなりません。 Windows Installer の 3.1 より以前の バージョンはこの設定を無視します。
MoreInfoURL	読み書きプロパティ	エンドユーザーにテクニカルサポートを提供するサポート URL を取得または設定します。

名前	種類	説明
OptimizedInstallMode	読み書きプロパティ	Windows Installer 3.1 を利用してパッチ処理を最適化し、現在のパッチを適用する時間を短縮する場合は、このプロパティを True に設定します。選択されたテーブルのリストのみをパッチする場合、Windows Installer 3.1 はその他のテーブルに関連付けられたアクションを無視します。テーブル一覧を含む詳細は、Windows Installer ヘルプライブラリの「Patch Optimization」を参照してください。 このプロパティに True を設定した場合、予定通りにパッチが動作することを慎重にテストすることがマイクロソフトによって推奨されています。 現在のパッチに対して Windows Installer 3.1 によるパッチ処理の最適化を行なわない場合、False を選択します。パッチがターゲットマシンに適用されたとき、Windows Installer はアプリケーションの完全修復を実行します。 Windows Installer の 3.1 より以前のバージョンはこのプロパティを無視します。パッチが選択したテーブルのリストのみを変更する場合、Windows Installer 3.0 はパッチを最適化します。Windows Installer 3.0 によるパッチの最適化を避けるためには、DisableFlyWeightPatching ポリシーを利用しなくてはなりません。 さらに詳しい情報は、Windows Installer ヘルプ ライブラリの「DisableFlyWeightPatching」を参照してください。 Windows Installer 3.0 より前のバージョンは、パッチが適用されたときにインストールの完全修復を実行します。
OptimizeForLargeFiles	読み書きプロパティ	このプロパティを True にを設定して、インストール プロジェクトで 4 MB よりも大きい全アプリケーションファイル用に小さな bit レベルのパッチを作成します。
PasswordProtected	読み書きプロパティ	パッチをパスワードで保護する場合、このプロパティを True に設定してから、LauncherPassword プロパティを使ってパスワードを設定してください。パッチをパスワードで保護すると、パッチを適用するエンド ユーザーは、アップデートを起動するために大文字小文字の区別があるパスワードを入力する必要があります。 この設定は Update.exe ファイルを使用するパッチのみに適用します。

名前	種類	説明
PatchCacheDirectory	読み書きプロパティ	パッチ キャッシングからのテンポラリ ファイルが保存されるディレクトリを指定します。これらの中間ファイルは無期限にシステム上に残ります。
PatchCommandLine	読み書きプロパティ	MSI コマンドライン引数を取得または設定します。 これらは、Setup.ini に書き込む Windows Installer 引数です。 [Startup] CmdLine=値 このプロパティのデフォルト値は、次のとおりです。 REINSTALLMODE=omus REINSTALL=ALL REINSTALL プロパティは、実行される再インストールのタイプを指定する文字を含む文字列です。詳細については、Windows Installer ライブラリの「REINSTALLMODE Property」を参照してください。
PatchGuid	読み書きプロパティ	パッチ パッケージを一意に識別するのに使用されるパッチ GUID を設定します。"パッチ GUID の生成" プロパティに [はい] が設定されていると、InstallShield は、パッチがビルドされるごとに、このプロパティに新しい GUID を自動生成します。
PatchOutputPath	読み書きプロパティ	パッチ ファイルをビルドする場所を指定、または既存フォルダーを検索します。InstallShield は、出力先に次のサブディレクトリを追加します。 ¥*PatchConfigName*¥Patch.
SignatureDescription	読み書きプロパティ	(必要な場合) パッチ パッケージおよび Update.exe ファイルに使用する署名の説明を取得または設定します。ここで指定した説明は、UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示されます。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。 この設定を空白のままに残すと、InstallShield は UAC (ユーザー アカウント制御) ボックスの [プログラム名:] ラベルの右側に表示される説明として、ファイル名を拡張子なしで使用します。

名前	種類	説明
SignaturePassword	読み書きプロパティ	使用する .pfx ファイルのパスワードを取得または設定します。InstallShield はパスワードを暗号化してプロジェクト ファイル (.ise) に保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。
SignatureSPC	読み書きプロパティ	パッチ パッケージの署名に使用するデジタル証明書情報を取得または設定します。.Pfx ファイルの場所または証明書を含む証明書ストアについての情報を使用します。
SignatureURL	読み書きプロパティ	完全修飾 URL (例、**http://www.mydomain.com**) を取得または設定します。この URL は、エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル署名の中で使用されます。
SigningType	読み書きプロパティ	次の値を入力してパッチ パッケージおよび Update.exe ファイルにデジタル署名を行う方法を取得または設定します。 • InstallShield のデフォルトの署名ツールを使用してパッチ パッケージおよび Update.exe ファイルにデジタル署名する標準の方法を選択するには、**0** を入力します。 • カスタマイズした署名ツールを使用してパッチ パッケージと Update.exe ファイルにデジタル署名するカスタム方法を選択するには、**1** を入力します。 ***メモ:** 既定で **SigningType** プロパティの値は **0** で、標準の方法を選択します。*
SignPackage	読み書きプロパティ	このプロパティを True に設定して、パッチ パッケージ にデジタル署名します。
SignUpdateExe	読み書きプロパティ	このプロパティを True に設定して、Update.exe ファイル にデジタル署名します。
TargetProductName	読み書きプロパティ	アプリケーションまたはターゲット アプリケーション スイートの名前を取得または設定しします。

名前	種類	説明
TokenPassword	読み書きプロパティ	暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを取得または設定します。 ***メモ:*** *EV トークンのパスワードを指定する際には、以下の情報に注意してください:* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*
UpdateLauncherName	読み書きプロパティ	InstallShield がビルド時に生成するアップデート起動ツールに使用するアップデート起動ツールのファイル名を取得または設定します(.exe ファイル拡張子は除きます)。この設定が空白の場合、InstallShield は「アップデート」のデフォルト値を使用し、アップデートランチャー ファイルは Update.exe と呼ばれます。
Use64bitSigning	読み書きプロパティ	64 ビット署名フレームワークを使用してパッケージにデジタル署名を行う場合は、このプロパティを [True] に設定します。 32 ビット署名フレームワークを使用してパッケージにデジタル署名を行う場合は、このプロパティを [False] に設定します。
ValidateAtBuild	読み書きプロパティ	パッチのビルド時、毎回、アップグレードおよびパッチの検証を利用するか場合、このプロパティを True に設定します。

AddLatestSetup メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddLatestSetup メソッドを呼び出して、現在のパッチ構成に最新のセットアップ項目を追加します。このメソッドは、IDE の [パッチのデザイン] ビューで最新のセットアップ項目を追加する方法に似ています。
ISWiLatestSetup オブジェクトを使って、新しい最新のセットアップのプロパティを設定できます。

構文

AddLatestSetup(LatestSetupName As String) As ISWiLatestSetup

パラメーター

テーブル 90 - AddLatestSetup メソッドのパラメーター

パラメーター	説明
LatestSetupName	最新のセットアップ項目の名前を保持します。

次に適用：

- ISWiPatchConfig

DeleteLatestSetup メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

DeleteLatestSetup メソッドを呼び出し、現在のパッチ構成から指定された最新のセットアップ項目を削除します。

構文

DeleteLatestSetup (LatestSetup As ISWiLatestSetup) As Long

パラメーター

テーブル 91 - DeleteLatestSetup メソッドのパラメーター

パラメーター	説明
LatestSetup	削除する最新のセットアップ項目を保持します。

次に適用：

- ISWiPatchConfig

ISWiPatchExternalFile オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*

以前のセットアップの一部として出荷されなかったアプリケーション ファイルのバージョンに対するバイナリ ファイル パッチを作成するファイブファイル オブジェクトを示します。

たとえば、アプリケーション ファイルの 1 つが他のセットアップにより更新されている可能性がある場合、そのファイル バージョンを参照して、実行時にファイル レベルのパッチが確実に適用されるようにします。

メンバー

テーブル 92 · ISWiPatchExternalFile オブジェクトのメンバー

名前	種類	説明
Component	読み取り専用プロパティ	FileKey プロパティで指定したファイルキーに関連付ける外部ファイルのファイルパスを取得または設定します。 これは、パッチが適用されるときに更新する必要があるかもしれないファイルです。
FileKey	読み取り専用プロパティ	外部ファイルに関連付けるファイル キーを取得または設定します。
名前	読み取り専用プロパティ	以前のセットアップ名を取得します。

次に適用 :

- ISWiProject

ISWiPatchWholeFile オブジェクト

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ビルドにファイルのビットレベルの差分を実行させるかわりに、パッチに完全な形で含めるファイル オブジェクトを示します。

***重要·** すべてのファイルを完全な形で含めるには、パッチ構成の ファイルを完全な形で含める プロパティを [はい] に設定します。*

メンバー

テーブル 93 • ISWiPatchWholeFile オブジェクトのメンバー

名前	種類	説明
Component	読み取り専用プロパティ	完全なファイルが属するコンポーネント名を取得または設定します。
FileKey	読み取り専用プロパティ	完全なファイルの FileKey プロパティを取得します。 この値は、AddPatchWholeFile メソッドに引数を渡して設定します。

次に適用：

- ISWiProject

ISWiPathVariable オブジェクト

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiPathVariable は、現在のプロジェクトに含まれるパス変数を表します。パス変数は、InstallShield の [パス変数] ビューから作成された既存のパス変数でも、AddPathVariable メソッドを呼び出して追加したパス変数でもかまいません。

重要• *定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

メンバー

テーブル 94 • ISWiPathVariable オブジェクトのメンバー

名前	種類	説明
名前	読み書きプロパティ	パス変数の名前を取得または設定します。
PathVarType	読み書きプロパティ	この列挙された整数プロパティは、パス変数の種類を指定します。次の値が可能です: • epvtPreDefined (1)—定義済みパス変数は特定の標準 Windows ディレクトリに設定された変数です。これらの変数は修正したり名前を変更することはできませんが、あらかじめ決められたディレクトリをポイントするためにインストール プロジェクトで使用することができます。 • epvtCustom (2)—カスタム パス変数は、*標準パス変数*または*ユーザー定義変数*と呼ばれることもあります。これらの変数タイプは InstallShield で定義されます。これらの変数は、レジストリ、システムパスなど、外部のソースには依存しません。 • epvtEnvironment (4)—環境パス変数は、システムの環境変数の値に基づきます。プロジェクトの環境変数パス変数に、マシン上に既存する環境変数を設定できます。 • epvtRegistry (8)—レジストリ パス変数を使用すると、指定したレジストリ キーのデフォルト値に基づいて独自の変数を定義することができます。パス変数をキーの値に設定する際、レジストリ キーが既に存在している必要があります。
TestValue	読み書きプロパティ	パス変数のテスト値を取得または設定します。
値	読み書きプロパティ	パス変数の値を取得または設定します。 **標準パス変数の場合** 標準変数の場合、変数がポイントするディレクトリを入力します。 ***メモ:*** *また、参照先となる他のパス変数の名前を山括弧で囲むことにより、定義済みの値の中でそのパス変数を参照することができます。たとえば、*C:¥*という値を持つ MyRoot というパス変数がある場合、そのパス変数を Games という別の変数のパス変数定義で参照することができます。Games 変数の実際のパスは* C:¥Programs¥GameFiles *のようになりますが、Games を* <MyRoot>¥Programs¥GameFiles *と定義することができます。ただし、パス変数を自己参照しようとした場合は、使用した文字列そのものが代わりに使用されます。たとえば、Games を* <MyRoot>¥Programs¥<Games> *と定義すると、実際には 、Games は* C:¥Programs¥<Games> *のように定義されます。*

テーブル 94 • ISWiPathVariable オブジェクトのメンバー (続き.)

名前	種類	説明
Value		**レジストリ パス変数の場合** 完全なレジストリ キーを入力します。最後の「サブキー」は、フォルダーが含まれる値名にします。たとえば、*MyRegVar* を次のように定義します。 HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey¥TestValue TestKey には次のサブキーと値があると想定します。 [HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey] @="C:¥¥MyPath1" "TestValue"="C:¥¥MyPath2" [HKEY_LOCAL_MACHINE¥Software¥TestKey¥TestValue] @="C:¥¥MyPath3" HKEY_LOCAL_MACHINE¥TestKey には TestValue というサブキーがありますが、*MyRegVar* は値 TestValue を指し、現在値は C:¥MyPath2 になります。(ただし、TestValue という値が存在しない場合、InstallShield はサブキー HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey¥TestValue のデフォルト値 (C:¥MyPath3) を読み取ります。) **環境パス変数の場合** 環境パス変数について、[環境] ダイアログ ボックスに表示されているとおりの変数名を入力します。

次に適用 :

- ISWiProject

ISWiPreviousSetup オブジェクト

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

最新セットアップにパッチする必要がある以前のセットアップ (または TargetImage) 項目を示します。

メンバー

テーブル 95 • ISWiPreviousSetup オブジェクトのメンバー

名前	種類	説明
IgnoreMissingFiles	読み取り専用プロパティ	このブール値プロパティを True に設定して、最新バージョンで変更されていない限り、より新しいバージョンでは見つからない 2 つの個別のバージョンに含まれるパッチ パッケージからファイルを除外します。ファイルを無視することの利点は、必要な変更されたファイルのみがパッケージに追加される点です。検出されないソースファイルを無視すると選択した場合、セットアップの完全イメージは不要です。
MsiPath	読み取り専用プロパティ	パッチする必要がある以前のバージョンのセットアップ パスを取得または設定します。
名前	読み取り専用プロパティ	以前のセットアップ名を取得します。
Order	読み取り専用プロパティ	指定された PatchFamily 内でこのパッチのシーケンスを取得または設定します。シーケンスの値は、バージョン データのフォーマットで表示されます。値は 1 から 4 のフィールドを含み、各フィールドの値範囲は 0 から 65535 です。PatchFamily のメンバーは、シーケンス値の昇順で、ターゲット製品に順番に適用されます。たとえば、次の 6 つの値は降順です: 1、1.1、1.2、2.01、2.01.1、2.01.1.1。
ValidateFlags	読み取り専用プロパティ	このブール値プロパティを Ture に設定して、同じ PatchFamily 内でより小さいシーケンス値を持つすべてのパッチが提供するアップデートを、小さいアップデート パッチが優先することを示します。

次に適用 :

- ISWiProject

ISWiProductConfig オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトには、ISWiProductConfigs に Media という要素 1 つだけが含まれています。

ISWiProductConfig の一部のメンバーのみが InstallScript プロジェクトと InstallScript オブジェクト プロジェクトに適用します。

ISWiProductConfig は、InstallShield ユーザー インターフェイスの [リリース] ビューで製品構成を表します。ISWiProductConfigs コレクションの中の項目を指定して、設定を取得できます。

メンバー

テーブル 96 • ISWiProductConfig オブジェクトのメンバー

名前	プロジェクト	種類	説明
AddRelease	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在の製品構成にリリースを追加します。
ArchitectureValidation	基本の MSI、マージ モジュール	読み書きプロパティ	このプロパティを次の値の 1 つに設定して、ビルド時に行うアーキテクチャ検証の種類を指定します: • epvtLenient (0)—デフォルトの検証の種類である、アーキテクチャの厳密な検証をしない場合、"テンプレートの概要" プロパティで指定されたアーキテクチャとリリースに含まれる 1 つ以上の製品ファイルまたはカスタム アクション ファイルのアーキテクチャが一致しないとき、ビルド エラーまたは警告がトリガされません。 • epvtStrict (1)—厳密な検証を行う場合、"テンプレートの概要" プロパティで指定されたアーキテクチャとリリースに含まれる 1 つ以上の製品ファイルまたはカスタム アクション ファイルのアーキテクチャが一致しないとき、ビルド エラーおよび警告がトリガされる可能性があります。 このプロパティの値は、InstallShield にビルトイン InstallShield カスタム アクション DLL の 32 ビット バージョンまたは 64 ビット バージョンのどちらを含むかにも影響します。 詳細については、「ビルドに適切なアーキテクチャ検証の種類を選択する」を参照してください。
ConfigFlags	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	機能のフィルターに使用するリリースフラグを持つ文字列を取得または設定します。複数のフラグは、コンマで区切ります。

テーブル 96 • ISWiProductConfig オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DeleteRelease	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	指定したリリースを現在の製品構成から削除します。
GeneratePackageCode	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このプロパティをTrueに設定すると、この製品構成の下にビルドした各リリースの新しいパッケージ コードを生成できます。 このプロパティを指定しないか、または偽に設定すると、その製品構成の下の既存のパッケージコードが使用されます。ただし、ここに指定しない場合は、概要情報ストリームのパッケージコードが使用されます。
ISWiReleases	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	コレクション	この製品構成に属するすべてのリリースを含みます。
MSIPackageFileName	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	現在の製品構成の "MSI パッケージ ファイル名" 設定の値を取得または設定します。
名前	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み取り専用プロパティ	現在の製品構成の名前を戻します。

テーブル 96 • ISWiProductConfig オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PackageCode	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	この製品構成のリリースにビルドされるパッケージ コードの文字列 GUID を取得または設定します。GeneratePackageCode プロパティを True に設定すると、このプロパティで入力したパッケージ コード、または概要情報ストリーム パッケージ コードのいずれも使用されません。
ProductCode	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	この製品構成のリリースにビルドされる製品コードの文字列 GUID を取得または設定します。 ***メモ:*** *GUID は、{12345678-1234-1234-1234-1234567890AB} のように中括弧で囲む必要があります。*
ProductName	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	この製品構成のリリースにビルドされる製品名を取得または設定します。
ProductVersion	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	製品のバージョン番号を含む文字列を取得または設定します。これは、この製品構成下でビルドするすべてのリリースの "製品バージョン" プロパティを上書きします。
SetupFileName	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	現在の製品構成の "セットアップ ファイル名" 設定の値を取得または設定します。
Setup64bitLauncher	基本の MSI	読み書きプロパティ	現在の製品構成の "64 ビット セットアップ ランチャー" 設定の値を取得または設定します。選択可能な値は、以下のとおりです: • TRUE – 64 ビット セットアップ ランチャー • FALSE – 32 ビット セットアップ ランチャー
件名	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	現在の製品構成の "件名" 設定の値を取得または設定します。

テーブル 96 • ISWiProductConfig オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
タイトル	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	現在の製品構成の "タイトル" 設定の値を取得または設定します。
UpgradeCode	基本の MSI、InstallScript MSI	読み書きプロパティ	作成されたセットアップの "アップグレード コード" プロパティを取得または設定します。

次に適用 :

- ISWiProject

AddRelease メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

AddRelease メソッドを呼び出して、現在の製品構成にリリースを追加します。このメソッドは、リリース ウィザード か、IDE の [リリース] ビューを使ってリリースを追加するのに似ています。

構文

AddRelease (ReleaseKey As String) As ISWiRelease

パラメーター

テーブル 97 • AddRelease メソッド パラメーター

パラメーター	説明
ReleaseKey	現在の製品構成に作成するリリースの名前を渡します。

次に適用 :

- ISWiProductConfigs

DeleteRelease メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

DeleteRelease メソッドを呼び出して、現在の製品構成からリリースを削除します。このメソッドは、IDE の [リリース] ビューでリリースを削除する方法に似ています。

構文

DeleteRelease (Release As ISWiRelease)

パラメーター

テーブル 98 • DeleteRelease メソッド パラメーター

パラメーター	説明
ReleaseKey	現在の製品構成から削除するリリースの名前を渡します。

次に適用 :

- ISWiProductConfigs

ISWiProperty オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiProperty は、InstallShield ユーザー インターフェイスのプロパティマネージャーにある Windows Installer のプロパティを表示します。ISWiProperties コレクションの中の項目を指定して、フォルダーを取得します。

メンバー

テーブル 99 • ISWiProperty オブジェクト

名前	種類	説明
Comments	読み書きプロパティ	Windows Installer プロパティと関連するコメントを取得または設定します。
名前	読み取り専用プロパティ	プロパティの名前を含みます。
値	読み書きプロパティ	プロパティの値を取得または設定します。

例

次の例では、ApplicationUsers プロパティが OnlyCurrentUser に設定されます。CustomerInformation ダイアログのデフォルトにより、オプション [自分だけ] が選択されるということです。

```
Dim pProject As ISWiProject

Set pProject = New ISWiProject
pProject.OpenProject "C:¥MySetups¥ISWiProperties.ism"

Dim pProperty As ISWiProperty
Set pProperty = pProject.ISWiProperties.Item("ApplicationUsers")

pProperty.Value = "OnlyCurrentUser"
pProperty.Comments = "CustomerInformation ダイアログで、""自分だけ"" を " & _
                     "デフォルトのラジオボタンとします。"

pProject.SaveProject
```

次に適用：

- ISWiProject

ISWiRelease オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiRelease は、InstallShield の [リリース] ビューにある既存リリースを表します。ISWiReleases コレクションの中の項目を指定して、リリースを取得します。リリースは、リリース ウィザード、[リリース] ビュー、またはコマンドライン ツール ISCmdBld.exe を使って作成したものでなくてはなりません。

メンバー

テーブル 100 • ISWiRelease オブジェクトのメンバー

名前	プロジェクト	種類	説明
AppxDistributionMethod	基本の MSI	読み書きプロパティ	Windows アプリ パッケージの配布方法を定義する文字列を取得または設定します。このプロパティは、Windows ストア ポリシー関連、またはサイドローディング関連の要件で問題の可能性がある場合に発生する一連の警告を指示します。次の文字列値が可能です: • **サイドロード**— Windows アプリ パッケージを Windows ストアからのダウンロードなしで配布します。サイドロード アプリは、Windows ストアによって認証されていません。サイドロード アプリが良く利用されるのは、エンタープライズ環境で、会社組織内部のみでアプリが配布される場合です。パッケージがサイドロードされる場合、カスタマが独自に署名することを希望しない限り、一般的に署名が必要です。既定では、AppxDistributionMethod プロパティがサイドロード文字列に設定されています。 • **ストア**— Windows アプリ パッケージを Windows ストアを通して配布します。この値が設定されているとき、サイドロードで有効なビルドであっても、Windows ストア ポリシーで拒絶されることが確認されているアイテムには、追加の警告が発生します。パッケージが Windows ストアを通して配布される場合、そのコンテンツを制限する追加ポリシーが適用されます。この種類のアプリは、パッケージが Microsoft によって署名されるため、Microsoft にアップロードする前に署名する必要がありません。 AppxDistributionMethod プロパティに無効な文字列値を設定しようとすると、デフォルトでサイドロード文字列になります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxFrameworkDependency	基本の MSI	読み書きプロパティ	製品で使用されているのと一致する C++ ランタイムのリテール ビルドまたはデバッグ ビルドを設定します。利用可能な値は、以下のとおりです。 • VC110 – Microsoft Visual C ++ 2012 ランタイム用。 • VC110D – Microsoft Visual C ++ 2012 デバッグ ランタイム用。 • VC120 – Microsoft Visual C ++ 2013 ランタイム用。 • VC120D – Microsoft Visual C ++ 2013 デバッグ ランタイム用。 • VC140 – Microsoft Visual C ++ 2015 ランタイム用。 • VC140D – Microsoft Visual C ++ 2015 デバッグ ランタイム用。 このプロパティは、有効な値以上にのみ設定できます。 ***メモ:*** *このプロパティを設定しても、サイドローディング シナリオで依存関係パッケージがインストールされることはありません。これらのシナリオの場合、依存関係は手動で配布されている可能性があります。また、C++ ランタイムのデバッグ バージョンは、内部テストで役立ちますが、Windows ストアにアップロードすることができません。ローカル テストまたはサイドローディングの場合、Windows 10 SDK に含まれる Framework パッケージも配布する必要があるかもしれません。*
AppxLogoSource	基本の MSI	読み書きプロパティ	パッケージを表すイメージのパスを取得または設定します。通常 50x50 のロゴを使用します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxMergeDependency	基本の MSI	読み書きプロパティ	結果となるパッケージに結合させる Windows アプリ パッケージの一覧を取得または設定します。 VFS からのレジストリ データおよびファイルがこのパッケージにコピーされますが、重複するデータはすべてパッケージから省略されます。
AppxPackageIdentityName	基本の MSI	読み書きプロパティ	Windows に対して Windows アプリ パッケージを識別する一意の英数字文字列を指定します。
AppxPackageIdentityPublisher	基本の MSI	読み書きプロパティ	パッケージの署名に使用する証明書のサブジェクト名 (所有者名) を取得または設定します。 Windows ストアを通してパッケージが配布される場合、Microsoft にアップロードする前に署名を行う必要はないので、証明書の署名をスキップしても構いません。パッケージがサイドロードされるとき、カスタマーによるパッケージの使用方法によって、これに署名が必要な場合と不要な場合があります: • カスタマーがパッケージの署名を自身で行いたい場合、証明書と一致する AppxPackageIdentityPublisher プロパティで設定された特定のパブリッシャーが必要です。 • カスタマーがパッケージの配布のみを行いたい場合、信頼された Authenticode 証明書で署名されている必要があります。その要件は、信頼された UAC プロンプトの要件と同様です。 • サイドロードを行うには、Windows の ″サイドロード″ 設定が有効でなくてはなりません。Windows 10 Anniversary Update では、既定でサイドロードが有効化されています。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxPackageIdentityVersion	基本の MSI	読み書きプロパティ	証明書が準拠する Windows アプリ パッケージのバージョン番号を取得または設定します。 バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を設定します: 14.0.25.1

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxPackageType	基本の MSI	読み書きプロパティ	リリース ビルド プロセスの一部としてビルドするパッケージの種類を設定します。このプロパティは、Windows デスクトップ アプリケーションを新しい Windows アプリ パッケージ形式に変換します。このプロパティで選択できる文字列値は、次のとおりです: • Appx—Windows アプリ パッケージのビルドこれは AppxPackageType プロパティのデフォルト値です。 • Msix—MSIX は、Windows プラットフォームの次世代ソフトウェア デプロイメント モデルで、MSI、AppX、および App-V の長所を単一パッケージに取り込みます。MSIX は、従来型の Win 32 アプリをコンテナーで実行する機能を持つため、アプリケーションの分離、デプロイメントの簡易化、完全にクリーンなインストール、およびシームレスなソフトウェア アップデートのすべてを同時に実現します。 • Appx;Msix— .appx および . msix パッケージの両方をビルドします。 AppxPackageType プロパティを使用して必要なパッケージの種類を設定するには、WindowsAppPackage プロパティを最初に True に設定する必要があります。 AppxDistributionMethod プロパティに無効な文字列値を設定しようとすると、デフォルトで Appx 文字列になります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxProductDescription	基本の MSI	読み書きプロパティ	エンドユーザーにパッケージを区別するのに役立つ説明を取得または設定します。 このプロパティに値 (説明) を設定すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。 ***メモ:*** *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージをビルドします。*
AppxProductDisplayName	基本の MSI	読み書きプロパティ	パッケージの名前として、ユーザーが認識できる文字列を取得または設定します。通常、アプリケーションの名前を使用します。このプロパティに値を設定すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。 ***メモ:*** *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージをビルドします。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxProductPublisherName	基本の MSI	読み書きプロパティ	パブリッシャーの名前として、ユーザーが認識できる文字列を取得または設定します。例 : Revenera このプロパティに値を設定すると、現在のプロジェクトに含まれているすべての言語に、文字列エントリがその初期値と共に作成されます。 ***メモ:*** *Windows アプリ パッケージのローカリゼーションを行うには、InstallShield が搭載されているマシン上に Windows 10 SDK もインストールする必要があります。これがインストールされていない場合、InstallShield はデフォルト言語のみを含む Windows アプリ パッケージをビルドします。*
AppxTargetDesktop	基本の MSI	読み書きプロパティ	コア Windows アプリ パッケージ (.appx/.msix Windows アプリ) フォーマットの一部として「フルトラスト」アプリや、その他の デスクトップ ブリッジ拡張を含めるかどうかを指定します。 このプロパティを True に設定して、含まれるアプリとデスクトップ拡張がサポートする最小 Windows バージョンとテスト済みの最大 Windows バージョンを指定するための、追加プロパティを構成することができます。 このプロパティを False を設定した場合、使用不可能なアイテムには警告が発生します。 デフォルトで、AppxTargetDesktop プロパティは True になっています。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxTargetDesktopMaxVersion	基本の MSI	読み書きプロパティ	アプリのテスト済み最大サポート対象 Windows バージョンを取得または設定します。バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1 AppxTargetDesktopMaxVersion プロパティを設定するには、AppxTargetDesktop プロパティを最初に True に設定する必要があります。
AppxTargetDesktopMinVersion	基本の MSI	読み書きプロパティ	アプリの済み最小サポート対象 Windows バージョンを取得または設定します。バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1 AppxTargetDesktopMaxVersion プロパティを設定するには、AppxTargetDesktop プロパティを最初に True に設定する必要があります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxTargetMSIXCore	基本の MSI	読み書きプロパティ	コア Windows アプリ パッケージ (.appx /.msix Windowsアプリ) の一部として MSIX コア拡張を含めるかどうかを指定します。これにより、古いバージョンの Windows に MSIX アプリをインストールできます。 このプロパティを True に設定して、MSIX アプリがサポートする最小 Windows バージョンとテスト済みの最大 Windows バージョンを指定するための、追加プロパティを構成することができます。さらに、指定の最小 Windows バージョンとテスト済みの最大 Windows バージョンと共に、マニフェスト ファイルに MSIX Core エントリを追加します。デフォルトで、AppxTargetMSIXCore プロパティは True になっています。 このプロパティを False に設定すると、マニフェストファイルの MSIX Core エントリが禁止されます。 ***ヒント:*** *MSIX Core パッケージには、古いバージョンのパッケージ上に MSIX 前提条件がインストールされている必要があります。*
AppxTargetMSIXCoreMaxVersion	基本の MSI	読み書きプロパティ	MSIX アプリのテスト済み最大サポート対象 Windows バージョンを取得または設定します。バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1 AppxTargetMSIXCoreMaxVersion プロパティを設定するには、AppxTargetMSIXCore プロパティを最初に True に設定する必要があります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxTargetMSIXCoreMinVersion	基本の MSI	読み書きプロパティ	MSIX アプリの最小サポート対象 Windows バージョンを取得または設定します。バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1 AppxTargetMSIXCoreMinVersion プロパティを設定するには、AppxTargetMSIXCore プロパティを最初に True に設定する必要があります。
AppxTargetServer	基本の MSI	読み書きプロパティ	コア Windows アプリ パッケージ (.appx/.msix Windows アプリ) フォーマットの一部として Windows Server 拡張を含めるかどうかを指定します。 このプロパティを True に設定して、このアプリがサポートする最小 Windows Server オペレーティング システムとテスト済みの最大 Windows Server オペレーティング システムを指定するための、追加プロパティを構成することができます。デフォルトで、AppxTargetSever プロパティは True になっています。 このプロパティを False を設定した場合、使用不可能なアイテムには警告が発生します。 ***ヒント•*** *InstallShield を使って Windows アプリ パッケージを作成するとき、デスクトップ拡張またはサーバー拡張のどちらかを含めることができますが、両方を選ぶことはできません。Windows アプリ パッケージにはデスクトップおよびサーバー拡張の両方を含めることが可能ですが、デスクトップまたはサーバーのいずれかをターゲットとすることが一般的です。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
AppxTargetServerMaxVersion	基本の MSI	読み書きプロパティ	アプリのテスト済み最大サポート対象 Windows Server オペレーティング システムを取得または設定します。バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1 AppxTargetServerMaxVersion プロパティを設定するには、AppxTargetServer プロパティを最初に True に設定する必要があります。
AppxTargetServerMinVersion	基本の MSI	読み書きプロパティ	アプリの最小サポート対象 Windows Server オペレーティング システム バージョンを取得または設定します。バージョン番号は数字のみを使用し、次のフォーマットのように 4 つのパートに分かれています (各パートは 0 から 65535 の値): *majorversion.minorversion.buildnumber.revisionnumber*。例、 1.0.0.60325 できる限り少ない数の文字を使用します。たとえば、最初の 0 が不要な場合は、14.00.25.1 と入力する代わりに、次を入力します: 14.0.25.1 AppxTargetServerMinVersion プロパティを設定するには、AppxTargetServer プロパティを最初に True に設定する必要があります。
BatchFileName	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの [バッチ ファイルの実行] プロパティを取得または設定します。このプロパティは、リリースのビルドが完了してから実行するバッチ ファイルまたは実行可能ファイルのファイル名を指定します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
BrowsersToSupport	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティは、One-Click Install タイプのインストールでサポートするブラウザーを示します。次の値を指定します: • **eocitbIE (0)** — Internet Explorer このプロパティは、One-Click Web リリースのみに適用されます。
ビルド	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	メソッド	現在のリリースをビルドします。これは完全ビルドです。
BuildErrorCount	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み取り専用プロパティ	ISWiRelease オブジェクトで Build メソッドを実行した後、BuildErrorCount はビルドプロセスが実行されている最中に発生したエラーの数を含みます。
BuildLocation	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	リリースがビルドされるトップレベルのディレクトリを識別する文字列を取得または設定します。
BuildTablesOnly	基本の MSI、InstallScript MSI、マージ モジュール	メソッド	現在のリリースの Windows Installer テーブルのみビルドします。
BuildTablesRefreshFiles	基本の MSI、InstallScript MSI	メソッド	リリースの新しいファイルと変更されたファイルを含んで .msi ファイル を再ビルドし、Files テーブルを更新します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
BuildUTF8Database	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	Windows Installer データベース、およびすべてのインスタンスまたは言語トランスフォームを UTF-8 エンコードを使ってビルドする場合、このブール型プロパティを True に設定します。UTF-8 エンコードは、すべての言語の文字を同時にサポートするため、エンド ユーザーに表示するテキストおよびファイル名とレジストリ キーの両方で、たとえば日本語とドイツ語、またはロシア語とポーランド語のように自由に言語を組み合わせて使用できます。組み合わされた言語は、ターゲット システムの現在の言語設定に関わらず正しく表示されます。ただし、一部の状況下では、ユーザー インターフェイスに問題が発生します。たとえば、エンド ユーザーが /qb コマンドライン オプションを指定するか、または [プログラムの追加と削除] から製品をアンインストールすると、Windows Installer がに非常に小さいフォントを使って UTF-8 データベースに含まれるユーザー インターフェイス テキストを表示します。 このプロパティを True に設定すると、リリースをビルドする際に ANSI データベースが作成されます。このオプションを使うと、異なるコード ページを持つ言語からの文字を同時に使用することができません。
BuildWarningCount	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み取り専用プロパティ	ISWiRelease オブジェクトで Build メソッドを実行した後、BuildWarningCount はビルドプロセスが実行されている最中に発生した警告の数を含みます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
CabCompressionType	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	Compressed プロパティが True に設定されている場合、この CabCompressionType プロパティを使って、リリースの .cab ファイルをビルドするときに InstallShield が使用する圧縮の種類を指定します。選択可能なオプションは以下のとおりです: • **ecctLZX (3)** — InstallShield は LZX 圧縮を使って、製品のデータ ファイルを .cab ファイルに圧縮します。このオプションを選択すると、最もサイズの小さい .cab ファイルが作成されますが、実行時に .cab ファイルからデータが抽出されるとき、最も時間がかかります。 • **ecctMSZIP (1)** — InstallShield は MSZIP 圧縮を使って、製品のデータ ファイルを .cab ファイルに圧縮します。 • **ecctNone (0)** — InstallShield は .cab ファイルを作成するときに圧縮を行いません。 ***重要:*** *圧縮を行うと、一般的に圧縮ファイルのサイズが減少します。ただし、ビルドプロセスの完了までの時間が長くなることがあります。圧縮されるファイルのサイズとその数によって、LZX 圧縮とビルドに数時間かかる場合もあります。したがって、LZX オプションを選択した場合、ビルドが完了するまでの所要時間をできるだけ短縮できるよう、ビルド マシンに最新のハードウェアを搭載することが推奨されます。*
CachePath	基本の MSI、InstallScript MSI	読み書きプロパティ	キャッシュされたインストールファイルがエンドユーザーのシステムに保存されるかどうかを指定します。**C:¥CachedFiles** のようにハードコード化された値を使用します。 このプロパティは、現在のビルドの CacheWebDownload プロパティが True に設定されている場合のみ使用されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
CacheWebDownload	基本の MSI、InstallScript MSI	読み書きプロパティ	現在のビルドのインストールファイルをターゲット システムにキャッシュするかどうか指定します。True に設定すると、インストールファイルがターゲット システム上にキャッシュされ、アプリケーションのメンテナンスや修復のときに使用されます。 CachePath プロパティを使用して、キャッシュされたファイルが入るがエンドユーザーのシステムに保存される場所を指定します。 このプロパティは、現在のビルドの WebType プロパティが ewtOneExe (2) (1 つの実行プログラム) に設定されている場合のみ使用されます。
CertificatePassword	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	書き込み専用のプロパティ	ファイルがパスワードで保護されている場合、.pfx ファイルのパスワードを設定します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ***メモ:*** *署名に使用する証明書が証明書ストアにある場合、このプロパティは適用しません。証明書がパスワード保護付きでストアにインポートされている場合、ビルド時にそのパスワードがプロンプトされます。*
CertificateURL	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル証明書の中で使用される URL を取得または設定します。完全修飾 URL を使用します (例、**http://www.mydomain.com**)。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
Compressed	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	すべてのパッケージファイルを .msi ファイル、.MSM ファイルまたは Setup.exe に圧縮する場合は、このプロパティを True に設定します。False に値を設定すると、アプリケーションファイルが圧縮されないままリリースの保存場所のサブフォルダーに配置されるます。 リリース ウィザードの [カスタム圧縮設定] パネルで利用できる柔軟性とは異なり、このプロパティはリリースにあるファイルのすべてを圧縮するか、どれも圧縮しないかのいずれかです。
CompressScript	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール型プロパティはリリースの [スクリプトの圧縮] プロパティを取得または設定します。このプロパティはコンパイルされたスクリプト ファイル (.inx ファイル) をキャビネット ファイルに配置する (True) か圧縮せずに Disk1 イメージフォルダーに配置するか (False) を指定します。
CopyToFolder	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの [フォルダーにコピー] プロパティを取得または設定します。このプロパティは、リリースをビルドした後でリリースのディスクイメージフォルダーをコピーするフォルダーの場所を指定します。コピーされたフォルダーと同じ名前の既存のフォルダーは上書きされますが、削除されるフォルダーはありません。 ***プロジェクト:*** *基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合、CopyToFolder の代わりに DistributeLoc プロパティを使って、フォルダーにリリースを配布します。*
CreateAutorunINF	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定すると、リリースの保存場所のルートで Autorun.inf ファイルを作成します。
CreatePDF	基本の MSI、InstallScript MSI	読み書きプロパティ	インストールのパッケージ定義ファイルを作成する場合、このプロパティを True に設定し、そうでない場合は False に設定します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
CryptographicProvider	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	暗号化サービス プロバイダー (CSP) を取得または設定します。 *メモ• 暗号化サービス プロバイダー (CSP) 名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*
CubFile	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	InstallShield がビルドされたリリースを検証するための検証モジュール (.cub ファイル) の名前を取得または設定します。
CustomSignArgs	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	署名ツール構成のコマンドライン引数を取得または設定して、ビルドが生成したファイルにデジタル署名を行います。 たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。 sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename] [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクトで定義されている任意のパス変数または環境変数を使用することができます。 *メモ• SigningType プロパティの値が eesCustom (1) の場合に、このプロパティを使用します。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
CustomSignPath	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ビルドが生成したファイルにデジタル署名行うために使用する署名ツールの場所を取得または設定します。 ***メモ: SigningType プロパティの値が eesCustom (1) の場合に、このプロパティを使用します。***
DefaultLang	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	読み書きプロパティ	指定のリリースのデフォルト言語の言語識別子を取得します。このプロパティ値は、[一般情報] ビューまたは [文字列エディタ] ビューで構成されたデフォルト プロジェクト言語をオーバーライドします。 詳細については、「デフォルトのプロジェクト言語の設定」を参照してください。
DelayMSIEngineReboot	基本の MSI、InstallScript MSI	読み書きプロパティ	アプリケーションのインストールが完了するまで、ターゲット システム上での Windows Installer エンジンのインストールまたはアップデートに関連した再起動を遅延するかどうかを指定します。 必要な場合、再起動を遅延するには True を選択します。False を選択すると、Windows Installer エンジンがインストールまたはアップデートされた直後 (アプリケーションのインストール前) に、必要に応じてシステムが再起動します。 このオプションには、Windows Installer バージョン 2.0 が必要です。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DigitalCertificateInfo	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	リリースの署名に使用するデジタル証明書を取得または設定します。InstallShield では、次のオプションから選択できます。 • 署名に使用する .pfx 証明書ファイルを指定するには、このプロパティをビルド マシン上の .pfx ファイルの完全修飾パスとファイル名に等しく設定します。 パスをハードコード化する代わりに、[パス変数] ビューまたは ISWiPathVariables コレクションの一部で定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。 • 署名に使用する証明書を格納する証明書ストアを参照するには、このプロパティを証明書ストア文字列に等しく設定します。 証明書ストア文字列を値として渡すには、次の規則を使用します: *Store*StoreLocation:Cert Subject **Store**に、次の値の 1 つを渡します: • MY • CA • Trust • Root **StoreLocation** に、次の値の 1 つを渡します: • ユーザー • マシン **Cert Store** に使用する証明書のサブジェクトを指定します。 詳細については、「デジタル署名とセキュリティ」を参照してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DisplayDotNetOptionDialog	基本の MSI、DIM、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	インストールで、実行時に .NET オプションのメッセージ ボックスを表示する場合、このプロパティを True に設定します。このメッセージ ボックスで、エンドユーザーは、.NET Framework をインストールするかどうかを指定することができます。 インストールで、エンドユーザーが .NET Framework をインストールするかどうかを選択できないようにする場合、このプロパティを False に設定します。 ***メモ:*** *このプロパティは、インストールに .NET Framework を含めるかどうかを決定するものではありません。エンドユーザーにインストールする選択肢を与えるかどうかを指定するだけです。*
DistributeAfterBuild	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	ビルドする度に、DistributeLoc プロパティまたは DistributeToURLLoc プロパティで指定した場所へビルド エンジンを使ってリリースを配布する場合、このプロパティを True に設定します。 ***メモ:*** *DistributeLoc プロパティと DistributeToURLLoc プロパティの両方が指定された場合、リリースは FTP ロケーションにのみコピーされます。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DistributeLoc	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	リリースを特定のフォルダーへ自動的に配布するようにするには、このプロパティでその場所を指定します。 DistributeAfterBuild プロパティは、True に設定されている必要があります。 コピーされたフォルダーと同じ名前の既存のフォルダーは上書きされますが、削除されるフォルダーはありません。 選択されたリリースのメディア形式がネットワーク イメージの場合 (ディスク イメージ フォルダーは 1 つのみ作成されます)、フォルダー自体ではなくディスク イメージ フォルダーの内容がコピーされます。自己展開型実行可能ファイルを作成することを選択した場合、ディスク イメージ フォルダーではなく実行可能ファイルがコピーされます。 ***メモ:*** *DistributeLoc プロパティと DistributeToURLLoc プロパティの両方が指定された場合、リリースは FTP ロケーションにのみコピーされます。*
DistributeToURLLoc	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	リリースを特定の FTP サーバーに自動的に配布できるようにするには、このプロパティでその場所の FTP URL を指定します。 DistributeAfterBuild プロパティは、True に設定されている必要があります。 ***メモ:*** *FTP デフォルト フォルダー以外のパスにリリースを配布する場合は、ダブル スラッシュ (//) を使います。たとえば、リリースを、FTP の URL が* **ftp://ftp.mydomain.com** *である* **myproduct** *という名前のルートレベル フォルダーへ配布する場合、FTP の場所に* **ftp://ftp.mydomain.com//myproduct** *と入力します。* *DistributeLoc プロパティと DistributeToURLLoc プロパティの両方が指定された場合、リリースは FTP ロケーションにのみコピーされます。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DistributeToURLPassword	基本の MSI、InstallScript MSI、マージ モジュール	書き込み専用のプロパティ	指定する FTP の場所へのアップロードにパスワードが必要な場合、このプロパティでパスワードを指定します。
DistributeToURLUserName	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	指定する FTP の場所へのアップロードにユーザー名が必要な場合、このプロパティでユーザー名を指定します。
DotNetBaseLanguage	基本の MSI、InstallScript MSI	読み書きプロパティ	.NET Framework インストールのユーザー インターフェイス言語を設定します。DotNetVersion を「1」に設定した場合、このプロパティを設定する必要があります。
DotNetBuildConfiguration	基本の MSI、InstallScript MSI	読み書きプロパティ	これを対応する .NET プロジェクトファイルのアクティブなリリース名に設定します。通常の値は、*[リリース]* と *[デバッグ]* です。
DotNetDelayReboot	基本の MSI、InstallScript MSI	読み書きプロパティ	アプリケーションのインストールが完了するまで、ターゲット システム上での .NET Framework のインストールまたは更新に関連付けられた再起動を遅延するかどうかを指定します。 インストールが完了するまで再起動のプロンプトを遅延するには、このプロパティを True に設定します。 ***メモ:*** *.NET Framework は、ターゲット システムが再起動するまで機能しない場合があります。したがって .NET Framework がインストール中に使用される場合、このプロパティを False に設定することを強く推奨します。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DotNetFrameworkLocation	基本の MSI、InstallScript MSI	読み書きプロパティ	これを次の値の 1 つに設定して、.NET Framework 再配布可能ファイルをインストールにパッケージする方法を指定します。 • **eelSourceMedia (0)** — .NET Framework ランタイムをソース メディアのルートに残します。 • **eelSetupExe (1)** — 実行時に、Setup.exe から .NET Framework を抽出します。 • **eelWeb (2)** — .NET Framework ランタイムをデフォルトの URL または DotNetFrameworkURL プロパティで指定した URL (下に表示) からダウンロードします。 • **eelDotNetNone (3)** — セットアップに .NET Framework を含めません。
DotNetFrameworkURL	基本の MSI、InstallScript MSI	読み書きプロパティ	DotNetFrameworkLocation を eelWeb (2) に設定した場合、プロパティによってインターネット上の dotnetfx.exe のデフォルト場所の上書きが可能になります。
DotNetFxCmdLine	基本の MSI、InstallScript MSI	読み書きプロパティ	.NET 1.1 再配布可能ファイル (dotnetfx.exe) 内のセットアップに渡すコマンドラインを設定します。
DotNetLanguagePackCmdLine	基本の MSI、InstallScript MSI	読み書きプロパティ	.NET 1.1 言語パック再配布可能ファイル (langpack.exe) 内のセットアップに渡すコマンドラインを設定します。
DotNetLanguagePacks	基本の MSI、InstallScript MSI	読み書きプロパティ	インストールする追加の .NET 1.1 言語パックを表す言語 ID を指定します。
DotNetUI	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定すると、実行時に .NET Framework インストールからユーザー インターフェイス (UI) を表示できます。
DotNetVersion	基本の MSI、InstallScript MSI	読み書きプロパティ	.NET 1.1 を出荷する場合はこのプロパティを 1 に設定します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
EnableDifference	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このBoolean プロパティは、リリースの "差分メディア" プロパティを取得または設定します。このプロパティは、現在のリリース ([リリース] ビューで選択済み) が差分リリース (つまり、指定した既存リリース (複数可) に存在しないファイルだけを含めるリリース) か、または製品のいずれのバージョンもインストールされていないシステム上にインストールできる、製品のすべてのファイルを含む完全リリースかを指定します。
EnableLangDlg	All	読み書きプロパティ	このプロパティを True に設定すると、Setup.exe から言語ダイアログを表示できます。
ExpirationDate	基本の MSI、InstallScript MSI	読み書きプロパティ	Setup.exe の有効期限日 (ISWiDate) を取得または設定します。このプロパティを設定すると、指定された日付以降にエンド ユーザーが Setup.exe を起動することはできません。 2012 年 12 月 30 日に有効期限を設定するサンプル Visual Basic コードは以下の通りです。 ISWiDate date date.wYear = 2012 date.wMonth = 12 date.wDay = 30 pISWiRelease.ExpirationDate = date このプロパティで有効期限日を設定した場合、ExpirationMessage プロパティを使って、エンド ユーザーが有効期限日またはその日付以降に Setup.exe を起動しようとしたとき、実行時に表示されるメッセージを取得または設定できます。
ExpirationMessage	基本の MSI、InstallScript MSI	読み書きプロパティ	ExpirationDate プロパティで有効期限日を設定した場合、ExpirationMessage プロパティを使って、エンド ユーザーが有効期限日またはその日付以降に Setup.exe を起動しようとしたとき、実行時に表示されるメッセージを取得または設定できます。 有効期限切れメッセージを設定するサンプル Visual Basic コードは以下の通りです。 pISWiRelease.ExpirationMessage = "このセットアップの有効期限は %s です。セットアップを終了します。"

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
FilterLangs	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定すると、以下の SupportedLangsData プロパティの設定に従って、言語固有のコンポーネントをフィルターすることができます。
FTPFolder	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの [FTP サイト フォルダー] プロパティを取得または設定します。このプロパティはディスク イメージ フォルダーを FTP サーバー上のどのフォルダーにコピーするかを指定します。FTPHostAddress プロパティが空白の場合のみ、このプロパティが適用されます。
FTPHostAddress	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "FTP ホスト アドレス" プロパティを取得または設定します。このプロパティはビルド プロセスでディスク イメージ フォルダーをアップロードする FTP の URL を指定します。リリースを FTP サーバーにアップロードしない場合は、このプロパティを空白のままにしてください。
FTPPassword	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "FTP サイト パスワード" プロパティを取得または設定します。このプロパティは FTPUserName プロパティに入力するユーザー名のパスワードを指定します。FTPHostAddress プロパティが空白の場合のみ、このプロパティが適用されます。
FTPUserName	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの [FTP サイト ユーザー名] プロパティを取得または設定します。このプロパティは FTP サーバーへのアクセスを取得するユーザー名を指定します。FTPHostAddress プロパティが空白の場合のみ、このプロパティが適用されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
GenerateFileHashValues	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	バージョン作成されていないビルドの各ファイルに対して MsiFileHash テーブルを作成するかどうかを示します。 これを True に設定すると、ビルドでバージョンが付けられていない各ファイルについて MsiFileHash テーブルが作成されます。False にすると、MsiFileHash テーブルが作成されません。このプロパティが False に設定すると、InstallShield はプロジェクトの MsiFileHash テーブル (ダイレクトエディターを使用して作成) で見つかったエントリをビルドします。 特定のファイルについて MsiFileHashテーブルが既に作成されている場合、ビルドは、ビルド時に情報を生成する代わりに、その情報を使用します。
GenerateOneClickInstall	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定してワンクリック インストールを作成します。このプロパティを True に設定した場合、このリリース用に OneClickHTMLBaseName と OneClickCabJarBaseName のファイル名を指定する必要があります。 リリースの "メディア フォーマット" プロパティがリリース ウィザードの [メディア タイプ] パネルを使って Webに設定されていない限り、このプロパティは無視されます。
IFTWCabSizeInKb	基本の MSI、InstallScript MSI	読み書きプロパティ	Web メディアタイプ用にビルドされたキャビネット (.cab) ファイルのサイズをキロバイトで指定します。コンポーネントごとに別々のキャビネット ファイルをビルドするには、0 (ゼロ) を指定します。 このプロパティは、現在のリリースの WebType プロパティが ewtIFTW (1) (Web からインストール) に設定されている場合にのみ使用されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
InitDlgProductName	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの初期化ダイアログ製品名のプロパティを取得または設定します。このプロパティは、セットアップ初期化ダイアログ ボックスで表示する製品名を指定します。このプロパティを空白のままにすると、[一般情報] ビューで指定された製品名が使用されます。
JSharpCmdLine	基本の MSI、InstallScript MSI	読み書きプロパティ	vjredist.exe に渡すコマンド ライン パラメーターを指定します。有効なコマンド ライン パラメーターについては Microsoft サポートに問い合わせてください。
JSharpOptionalIfSilent	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定すると、[J# オプション] ダイアログを表示できない場合(インストールをサイレントで実行している場合など)も J# 再配布可能ファイルがインストールされます。
JSharpOptionDlg	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定すると、J# 再配布可能ファイルをインストールするかどうかエンドユーザーにたずねる [J# オプション] ダイアログが表示されます。.NET 1.1 がセットアップに含まれている場合は、.NET 1.1 もインストールすることを通知します。
JSharpRedistLocation	基本の MSI、InstallScript MSI	読み書きプロパティ	この値を次の値の 1 つに設定して、J# 再配布可能ファイルをどのようにセットアップにパッケージするかを示します。 • **eelSourceMedia (0)** – J# 再配布可能ファイルをソース メディアのルートに残します。 • **eelSetupExe (1)** – 実行時に Setup.exe から J# 再配布可能ファイルを抽出します。 • **eelWeb (2)** – デフォルト URL か、以下で説明する DotNetFrameworkURL プロパティで指定された URL から J# 再配布可能ファイルをでダウンロードします。 • **eelDotNetNone (3)** – セットアップに J# 再配布可能ファイルを含めません。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
KeepUnusedDirectories	基本の MSI、InstallScript MSI、マージ モジュール	読み書きプロパティ	このリリースをビルドしたとき、.msi ファイルの Directory テーブルから未使用のディレクトリを削除するかどうかを指定します。選択可能なオプションは以下のとおりです: • **False** — Directory テーブルの Directory 列に一覧されているディレクトリが .msi ファイル内のすべての既知の場所で参照されていない場合、InstallShield はそれをビルド時に作成した .msi ファイルの Directory テーブルから削除します。基本の MSI プロジェクトおよび InstallScript MSI プロジェクトで、これはマージ モジュールがマージされてから削除されますが、.msi ファイルに存在するディレクトリのみが削除の対象となります。したがって、マージ モジュールの Directory テーブルに新しい未使用のディレクトリが含まれている場合、そのディレクトリはインストールに追加されます。 • **True** — InstallShield がビルド時に作成する .msi ファイルの Directory テーブルにあるディレクトリは、そのまま保持されます。 デフォルト値は False です。 ***メモ:*** *特定の条件下では、定義済みディレクトリが解決されないためにインストールが失敗する場合があります。*Directory *テーブルから未使用ディレクトリを削除することで、無駄なエラーを回避することができます。したがって、未使用のディレクトリを保持しないことが推奨されます。*
KeyContainerName	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	暗号化サービス プロバイダー (CSP) に関連付けられた秘密キー コンテナ名を取得または設定します。 ***メモ:*** *コンテナ名は、EV ベンダー認証クライアント ツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
LauncherCopyright	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	Setup.exe ファイルの [プロパティ] ダイアログ ボックスに表示される著作権情報を指定します。UseMyVersionInfo プロパティを True に設定して デフォルト著作権情報をオーバーライドする必要があります。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
LauncherFileDescription	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	Setup.exe ファイルの [プロパティ] ダイアログ ボックスに表示されるファイルの説明を指定します。UseMyVersionInfo プロパティを True に設定して デフォルトのファイルの説明をオーバーライドする必要があります。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
LauncherFileVersion	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	Setup.exe ファイルの [プロパティ] ダイアログ ボックスに表示されるファイルのバージョンを指定します。UseMyVersionInfo プロパティを [True] に設定して デフォルトのファイル バージョンをオーバーライドする必要があります。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
LauncherPassword	基本の MSI、InstallScript MSI	読み書きプロパティ	アプリケーションを保護するパスワードを指定します。PasswordProtectLauncher プロパティを True に設定してパスワード保護をアクティベートする必要があります。 このプロパティは、次の条件を満たすリリースにのみ適用できます。 • 単一ファイル Setup.exe • 圧縮ファイル • ネットワーク イメージ メディア タイプ
LogFileName	基本の MSI、InstallScript MSI、InstallScript プロジェクト	読み取り専用プロパティ	最後に行われたビルドのログファイルの場所を保存します。このファイルの場所を使って、最後に行ったビルドのログファイルを参照することができます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
MediaType	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	読み取り専用プロパティ	リリースの配布メディアの種類を取得します。使用できる値は次のとおりです。 • 0—CD-ROM • 1—ネットワーク イメージ • 2—カスタム • 3—DVD • 4—Web • 5—144 MB フロッピー ディスク • 6—InstallScript オブジェクト
MSI30EngineURL	基本の MSI、InstallScript MSI	読み書きプロパティ	エンジン場所の URL を取得または設定します。この場所は、実行時にエンジンをダウンロードする場所として使われます。
MsiEngineLocation	基本の MSI、InstallScript MSI	読み書きプロパティ	MSI エンジンのインストーラーの場所を指定します。次のうちの 1 つに設定します。 • **eelSourceMedia (0)** — ソース メディアからコピーします。選択した MSI エンジン インストーラーをソース メディアのルートに残します。このオプションは、すべてのファイルを Setup.exe に圧縮している Web メディア タイプまたはネットワーク イメージ メディア タイプでは使用できません。 • **eelSetupExe (1)** — エンジンを Setup.exe から抽出します。 • **eelWeb (2)** — エンジンを Web からダウンロードします。この設定を使用する場合、必ずMSI30EngineURL プロパティを適切な Web の場所に設定してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
MsixAppInstallerFileUrl	基本の MSI	読み書きプロパティ	アプリ インストーラー ファイルの場所を取得または設定します。 アプリ インストーラー ファイルは、HTTP エンドポイント、UNC 共有 フォルダー、ローカル ファイル共有、などの共有場所でホストすることが可能で、インストールされる MSIX パッケージを検索するためのパスが含まれています。ユーザーはこのアプリを共有場所または Web からインストールして、定期的な新しいアップデートのチェックを有効化します。
MsixAppInstallerVersion	基本の MSI	読み書きプロパティ	アプリ インストーラー ファイルのバージョンを取得または設定します。4 つの部分から成るバージョン文字列 *Major.Minor.Build.Revision*。例: 1.0.0.60325
MsixAutomaticBackgroundTask	基本の MSI	読み書きプロパティ	アプリが起動されているかどうかに関わらず、バックグラウンドでアップデートをチェックするかどうかを指定します。 このプロパティを True に設定して、アプリが起動されているかどうかに関わらず、バックグラウンドでのアップデートのチェックを有効化します。ユーザーがアプリを起動するかどうかにかかわらず、8 時間おきにチェックが行われます。 このプロパティを False に設定して、アプリが起動されているかどうかに関わらず、バックグラウンドでのアップデートのチェックを無効化します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
MsixForceUpdateFromAnyVersion	基本の MSI	読み書きプロパティ	アプリのバージョンの変更を許可するかどうかを指定します。 このプロパティを Ture に設定して、アプリのバージョン番号をインクリメントまたはデクリメントすることができます。このプロパティを設定しない場合、アプリは上位バージョンにのみアップデートが可能で、Windows デプロイメント サービスは古いバージョンへのダウングレードを許可しません。このプロパティを True に設定すると、古いバージョンのアプリ パッケージ バージョンにロールバック処理が必要な時に便利です。 このプロパティを False 設定して、アプリのバージョン番号が変更されるのを防ぐことができます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
MsixHoursBetweenUpdateCheck	基本の MSI	読み書きプロパティ	デプロイメント サービスがアプリ インストーラー ファイルへのアップデートをチェックする頻度を設定します。このプロパティに有効な値は、0、2、4、8、16、および 24 です。 このプロパティは、有効な値以上にのみ設定できます。 MsixHoursBetweenUpdateCheck が 0 に設定された場合、デプロイメント サービスはアプリケーションが起動される度にアップデートをチェックします。 その他の有効な値の場合、MsixHoursBetweenUpdateCheck で指定されている最後の時間数内にチェックされていない場合のみ、アプリケーションが起動されたときにデプロイメント サービスがアップデートのチェックを行います。たとえば、MsixHoursBetweenUpdateCheck が 16 に設定されている場合、過去 16 時間以内にアップデートがチェックされていない場合のみ、アプリケーションが起動されたときにデプロイメント サービスがアップデートをチェックします。アップデートが利用可能な場合、Windows デプロイメント サービスはそれを自動的にダウンロードします。アップデートは、アプリケーションが閉じた後にインストールされます。 MsixHoursBetweenUpdateCheck プロパティを設定するには、UpdateSettingsOnAppLaunch プロパティを最初に True に設定する必要があります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
MsixMainPackageUrl	基本の MSI	読み書きプロパティ	アプリ パッケージの場所の URL を取得または設定します。このプロパティに URL が設定されていない場合、MsixAppInstallerFileUrl プロパティに指定された URL が考慮されます。 MainPackage 要素のその他の情報 (名前、発行元およびバージョンを含む) は、それぞれ AppxPackageIdentityName、AppxPackageIdentityPublisher、および AppxPackageIdentityVersion を使って変更することができます。 AppxPackageIdentityName、AppxPackageIdentityPublisher、およびAppxPackageIdentityVersion プロパティで設定された値は、MSIX アプリケーション パッケージのマニュフェストに含まれている情報と一致する必要があるため、これらは MSIX パッケージとアプリ インストーラー ファイルの両方でビルド中に考慮されます。
MsixShowPrompt	基本の MSI	読み書きプロパティ	このプロパティを True に設定すると、デプロイメントでユーザーにアップデートについて通知するプロンプトが表示されます。アップデートは Windows 10 バージョン 1903 以降で使用可能です。最新アップデートについては、Microsoft ドキュメントを参照してください。 このプロパティを False に設定すると、デプロイメントでユーザーにアップデートについて通知するプロンプトが表示されません。 MsixShowPrompt プロパティを設定するには、UpdateSettingsOnAppLaunch プロパティを最初に True に設定する必要があります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
MsixUpdateBlocksActivation	基本の MSI	読み書きプロパティ	ユーザーがアプリケーションを起動するのを防ぐかどうかを指定します。 True を設定すると、アップデートが適用されるまでユーザーがアプリケーションを起動しないように阻止します。また、ユーザーがアップデートのみを行ってアプリを閉じることもできます。このプロパティを設定して、いくつかの重要な修正と共にアプリを起動する必要があるクリティカル アップデートを配布するのに役立ちます。 このプロパティを False に設定すると、ユーザーはアプリのアップデートを適用するかどうかにかかわらず、アプリケーションを起動することができます。 MsixUpdateBlocksActivation プロパティを設定するには、UpdateSettingsOnAppLaunch プロパティを最初に True に設定する必要があります。
名前	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み取り専用プロパティ	現在のリリースの名前。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ObjDiffOptions	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの [オブジェクトの差異] プロパティを取得または設定します。このプロパティは差分リリースに InstallShield オブジェクトを含める条件を指定します。次の定数を使用してこのプロパティを設定します。 • eodoAll(0)—差分リリースは、相当する完全アップデート リリースに含まれるすべてのオブジェクトを含みます。 • eodoExcludeAll(1)—差分リリースには、どのオブジェクトも含まれません。 • eodoIncludeIfChanged (2)—差分リリースは相当する完全アップデート リリースに含まれるすべてのオブジェクトを含みます。もしくは、最低 1 つの差分リリースにある対応オブジェクトと異なるオブジェクトを含みます。 このプロパティは、EnableDifference プロパティが True に設定されている場合にのみ適用されます。
OneClickCabJarBaseName	基本の MSI、InstallScript MSI	読み書きプロパティ	ビルド プロセスによって One-Click インストールに生成されるキャビネット ファイル (.cab) のベース ファイル名を指定します。ここで指定された名前に .cab 拡張子が追加されます。生成された 1 つまたは複数のファイルは、現在のリリースの Disk1 フォルダーに作成されます。 このプロパティは、現在のビルドの GenerateOneClickInstall プロパティが True に設定されている場合のみ使用されます。
OneClickHTMLBaseName	基本の MSI、InstallScript MSI	読み書きプロパティ	ビルドプロセスで生成された HTML ファイルのベースファイル名を指定します。指定したベース名の末尾には .htm 拡張子が付けられ、生成されたファイルは現在のリリース保存場所の Disk1 フォルダーに作成されます。 このプロパティは、現在のビルドの GenerateOneClickInstall プロパティが True に設定されている場合のみ使用されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
OptimizeSize	基本の MSI、InstallScript MSI	読み書きプロパティ	リリースのキャビネット ファイルで使用する圧縮のレベルを指定します。このプロパティを True に設定すると、一般的に圧縮ファイルのサイズが減少します。ただし、ビルドプロセスの完了までの時間が長くなることがあります。 このプロパティは、リリースのファイルの一部または全部を圧縮する場合にのみ使用できます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
OSFilter	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "プラットフォーム" 設定を取得または設定します。この設定はこのリリースでサポートするオペレーティング システムを指定します。コンポーネントに指定されたプラットフォームが、この設定で選択されたプラットフォームのどれにも一致しない場合、そのコンポーネントはリリースに含まれません。 次の定数を使用してこのプロパティを設定します。 • eosOSIndepedent = 0 • eosWin95 = &H10 (16) • eosWin98 = &H40 (64) • eosWinMe = &H80 (128) • eosWinNT4 = &H10000 (65536) • eosWin2000 = &H100000 (1048576) • eosWinXP = &H400000 (4194304) • eosWinServer2003 = &H800000 (8388608) • eosWinVista = &H1000000 (16777216)—これらの定数は Windows Vista および Windows Server 2008 用です。 • eosWin7 = &H2000000 (33554432)—これらの定数は Windows 7 および Windows Server 2008 R2 用です。 • eosWin8 = &H4000000 (67108864)—これらの定数は Windows 8 および Windows Server 2012 用です。 • eosWin81 = &H8000000 (134217728)—これらの定数は Windows 8.1 および Windows Server 2012 R2 用です。 • eosAll = &HFD100D0 (265355472) 複数のプラットフォームを指定できます (例、**eosWin7 Or eosWinServer2008 Or eosWinVista Or eosWinServer2003**)。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PasswordProtectLauncher	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定してセットアップをパスワード保護します。パスワードを LauncherPassword プロパティで指定します。 このプロパティは、次の条件を満たすリリースにのみ適用できます。 • 単一ファイル Setup.exe • 圧縮ファイル • ネットワーク イメージ メディア タイプ
PathVariableOverrides	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	指定のリリースについて、プロジェクト内のパス変数の値を取得または設定します。このプロパティを使ってパス変数の値を設定すると、[パス変数] ビューで設定されたすべての値がオーバーライドされます。 この変数を使って [パス変数] ビューで構成されたユーザー定義のパス変数、環境変数、およびレジストリ変数を取得またはオーバーライドすることができますが、*<WindowsFolder>* などの定義済みパス変数をオーバーライドすることはできません。 以下は、PathVar1 というパス変数の値をオーバーライドするサンプル Visual Basic コードです: pISWiRelease.PathVariableOverrides = "PathVar1=C:¥Test1" + vbCrLf + "PathVar2=C:¥test2" パス変数に関する詳細については、「パス変数を使用する」を参照してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PostbuildEvent	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール、	読み書きプロパティ	InstallShield がリリースをビルドおよび署名した後に実行するコマンドを取得または設定します。 コマンドを指定するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。 2 つのメッセージを設定するサンプル Visual Basic コードは以下の通りです: pISWiRelease.PostbuildEvent = ″Event1″ + vbCrLf + ″Event2″ ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PrebuildEvent	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	InstallShield がリリースをビルドする前に実行するコマンドを取得または設定します。このイベントは InstallShield がリリース フォルダとログ ファイルを作成した後、リリースのビルドを開始する前に実行します。 コマンドを指定するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。 2 つのメッセージを設定するサンプル Visual Basic コードは以下の通りです: pISWiRelease.PrebuildEvent = ″Event1″ + vbCrLf + ″Event2″ ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PrepareFilesEvent	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	InstallShield のビルド中に実行するコマンドを取得または設定します。このイベントは、InstallShield がリリース フォルダーおよびログ ファイルを作成した後、InstallShield ビルドによって生成された DLL が .msi パッケージまたは Setup.exe ファイルにストリームされる前に実行されます。 InstallScript および InstallScript オブジェクトの場合、バッチ ファイル (.bat、.cmd) または実行可能ファイル (.exe) のパスを指定します。 コマンドを指定するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PrecompressionEvent	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	製品のデータ ファイルを .cab ファイルに格納する場合、InstallShield が .msi パッケージと .cab ファイルをビルドした後に実行するコマンドを取得または設定します。このイベントは .cab ファイルが .msi パッケージにストリームされた後、.msi パッケージにデジタル署名が行われて Setup.exe ファイルにストリームされる前に発生します。 コマンドを指定するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。 2 つのメッセージを設定するサンプル Visual Basic コードは以下の通りです: pISWiRelease.PrecompressionEvent = "Event1" + vbCrLf + "Event2" ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。*
PreProcessorDefines	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "コンパイラ プリプロセッサ定義" 設定を取得または設定します。このプロパティはプリプロセッサ変数定義を指定します。
PreviousPackage	基本の MSI、InstallScript MSI	読み書きプロパティ	将来のパッチパッケージのサイズを最小化するには、以前のパッケージ (.msi ファイル) への完全なパスを指定します。
ReleaseFlags	基本の MSI、InstallScript MSI	読み書きプロパティ	機能のフィルターに使用するリリースフラグを持つ文字列を取得または設定します。複数のフラグは、コンマで区切ります。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
RequiredExecutionLevel	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	Windows Vista 以降のプラットフォーム上でインストール (セットアップランチャー、InstallShield 前提条件および .msi ファイル) を実行するためのインストールの Setup.exe ファイルによって必要とされる最低特権レベルを取得し設定します。選択可能なオプションは、以下のとおりです。 • **erelAsInvoker (0)** — Setup.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Setup.exe を実行することができます。Setup.exe は、資格情報または同意 (コンセント) を求める UAC メッセージを表示しません。これは、基本の MSI プロジェクトのデフォルト オプションです。 • **erelAsHighestAvail (1)** — Setup.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を承認する必要があります。非管理者は、管理者権限を持たずに Setup.exe を実行します。これは、InstallScript プロジェクトと InstallScript MSI プロジェクトのデフォルト オプションです。 • **erelAsAdmin (2)** — Setup.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。非管理者は、管理者としての認証が必要になります。 詳細については、「インストール中におけるユーザー アカウント制御のプロンプトの数を最小化する」を参照してください。
SetupCmdLine	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "コマンドラインのセットアップ" プロパティを取得または設定します。このプロパティはセットアップ起動時に Setup.exe に渡すコマンドライン パラメーターを指定します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SetupEXE	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定して、現在のリリースに Setup.exe インストール ランチャを作成します。 SetupEXE プロパティと TargetOS プロパティは、[リリース] ビューの Setup.exe タブにある "セットアップランチャー" 設定に対応しています。 セットアップランチャーが必要になるシナリオについては、「セットアップランチャーの作成」をご覧ください。
ShallowFolderStructure	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを設定して .msi ファイルや関連するファイルをサブフォルダー無しにリリース場所に直接作成します。
ShowPasswordDlg	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール型プロパティは、リリースの [パスワード ダイアログの表示] プロパティを取得または設定します。このプロパティは OnCheckMediaPassword イベント ハンドラー関数のデフォルトのコードでパスワードをチェックするコードを実行するかどうかを指定します。このプロパティは "メディア パスワード" プロパティでヌル値以外のパスワードを指定した場合のみ適用されます。
SignFiles	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	リリース内に署名するファイルがある場合、まずこの Boolean プロパティを True に設定してから、SignFilesInclude プロパティと SignFilesExclude プロパティを使用して、署名するファイルを指定します。 ***Windows ロゴ •*** *インストールのすべての実行可能ファイル (.exe、.dll、.ocx、.sys、.cpl、.drv、および .scr ファイル) は、Windows ロゴ プログラムに準拠するためにデジタル署名が必要です。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SignFilesExclude	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ビルド時にデジタル署名を行わないファイルおよびファイル パターンを指定します。次の事項に注意してください。 • ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、.drv ファイルはどれも署名しない場合 ***.drv** のように指定します。 ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイル含め、特定のパターンに一致するファイルに署名を行なうのを避ける場合、特に便利です。 • 各ファイルおよび各ファイル パターンはそれぞれの行に,改行コードで区切って入力します。 • 署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、SignFilesInclude プロパティと SignFilesExclude プロパティで ***.exe** を指定すると、.exe ファイルはすべて署名されません。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SignFilesInclude	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ビルド時にデジタル署名するファイルおよびファイル パターンを指定します。次の事項に注意してください。 • ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、すべての .exe ファイルに署名する場合、*.exe のように指定します。**.exe** ワイルドカード文字の使用は、プロジェクトに動的にリンクされたファイル含め、特定のパターンに一致するすべてのファイルに署名を行う場合、特に便利です。 • 各ファイルおよび各ファイル パターンはそれぞれの行に,改行コードで区切って入力します。 • 署名しないファイルとファイル パターンは、すべての署名するファイルとファイル パターンをオーバーライドしますので注意してください。たとえば、SignFilesInclude プロパティと SignFilesExclude プロパティで ***.exe** を指定すると、.exe ファイルはすべて署名されません。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SignFilesInPlace	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	InstallShield で署名をするとき、元のファイルにも署名をするか、またはリリースに組み込まれるファイルのみ署名するかを判別します。 • **False** — InstallShield は、各ファイルの一時コピーに署名して、その署名された一時コピーを使って、リリースをビルドします。この動作の場合、元のファイルの変更および署名は行われませんので注意してください。 • **True** — InstallShield で元のファイルに署名する場合、このプロパティを True に設定します。 このプロパティのデフォルト値は、False に設定されています。 基本の MSI プロジェクトまたは InstallScript MSI プロジェクトで、True を指定すると、もともと署名されていなかったファイルを含むリリースの圧縮バージョンと非圧縮バージョンの両方を更新する単一のパッチを作成する場合に便利です。
SigningType	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	ビルドが生成したファイルにデジタル署名を行う方法を取得または設定します。次のオプションを使用してこのプロパティを設定します: • **eesBuiltIn (0)** — この値を使用して、デフォルトの InstallShield 署名ツールを使用してビルドで生成されたファイルにデジタル署名標準の方法を選択します。 • **eesCustom (1)** — この値を使用して、カスタマイズされた署名ツールを使用してビルドで生成されたファイルにデジタル署名するカスタム方法を選択します。 ***メモ:*** *既定で* ***SigningType*** *プロパティの値は* ***eesBuiltIn (0)*** *で、標準の方法を選択します。*
SignLauncher	基本の MSI、InstallScript MSI	読み書きプロパティ	このプロパティを True に設定して Setup.exe ファイルにデジタル署名します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SignMedia	状況によって、異なります。(詳しくは「SignMedia」を参照してください。)	読み書きプロパティ	このプロパティを設定して、メディアに署名するかどうかを示します。また、署名する場合、署名するファイルも指定します。
SignSignedFiles	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	プロジェクトにデジタル署名されているファイルが既に存在する場合、InstallShield で、既存のデジタル署名を DigitalCertificateInfo プロパティで指定したデジタル署名で置き換えるかどうかを指定します。 • True — 既にファイルに含められている既存のデジタル署名情報の代わりに、DigitalCertificateInfo プロパティで指定するデジタル署名情報を使用します。 • False — 既に署名されているファイルについて、既存のデジタル署名情報をそのまま残します。これがデフォルトの値です。 これは SignFilesInclude プロパティと SignFilesExclude プロパティで指定された要件を満たすファイルにのみ適用されますので注意してください。
SingleEXEFileName	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "単一実行可能ファイル名" 設定を取得または設定します。このプロパティは、インストールを実行する自己展開型実行可能ファイルのファイル名を拡張子も含めて指定します。プロパティの値がヌルの場合は自己展開型実行可能ファイルは作成されません。
SingleEXEIconName	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "単一実行ファイルアイコンファイル" 設定を取得または設定します。このプロパティは実行可能ファイルのアイコンがビルド時に使う完全修飾名を指定します。プロパティの値がヌル値の場合、デフォルトのアイコンが使用されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SkinName	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "スキンの指定" 設定を取得または設定します。このプロパティは、このリリースに適用されるダイアログ ボックスのスキンを指定します。このプロパティは、UseProjectSkin プロパティが True に設定されている場合にのみ適用されます。
SmallInitializationDialog	基本の MSI、InstallScript、InstallScript MSI	読み書きプロパティ	このプロパティを設定して、ユーザーがこのリリースを実行した時に簡単な開始ダイアログを表示します。
SupportedLangsData	基本の MSI、InstallScript MSI	読み書きプロパティ	リリースに含むコンポーネントを判別する言語 ID のリストを取得または設定します。FilterLangs プロパティを True に設定すると、言語依存性がないコンポーネントおよび特定の言語のコンポーネントみがリリースにビルドされます。 言語 ID の 10 進数の値を使用し、複数言語をカンマで区切ります。
SupportedLangsUI	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	リリース言語を判別する 10 進数の言語 ID のカンマで区切られた文字列を取得または設定します。
SupportedVersions	InstallScript、InstallScript オブジェクト	読み書きプロパティ	リリースの "サポートされているバージョン" プロパティを取得または設定します。このプロパティは、セミコロン区切りのバージョン数 (例 1.2.3;1.2.4) を指定します。このリリースは、旧バージョンの [アップデート] として適用されます。フィールドを空白にする場合、全バージョン、バージョンなしの製品がインストールされているシステムでセットアップは実行できます。このプロパティは、EnableDifference プロパティが False に設定されている場合にのみ適用されます。

テーブル 100・ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
SuppressLauncherWarning	基本の MSI、InstallScript MSI	読み書きプロパティ	Windows Installer サービスをターゲット システムでインストールまたはアップデートできなかった時に、このプロパティが False に設定されている場合、警告が表示されます。警告を抑止する場合は True に設定します。 このプロパティは、Setup.exe を作成してパッケージを起動する場合のみ適用できます。
TargetOS	基本の MSI、InstallScript MSI	読み書きプロパティ	リリースに含める Windows Installer のプラットフォーム バージョンを判別する値を取得または設定します。次の値のうち 1 つを指定します。 ・ **osNone (0)** — Windows Installer を含みません。 ・ **osWin9x (1)** — InstMsi.exe の Windows 9x バージョンのみ含みます。 ・ **osWinNT (2)** — InstMsi.exe の Windows NT バージョンのみ含みます。 ・ **os9xNT (3)** — InstMsi.exe の Windows NT および Windows 9x バージョン両方を含みます。 SetupEXE プロパティ と TargetOS プロパティは、[リリース] ビューの Setup.exe タブ にある "セットアップランチャー" 設定に対応しています。 ***メモ:*** *この設定は、Windows Installer 3.1 以前の再配布可能ファイルに適用します。Windows Installer 4.5 再配布可能ファイルについては、「Windows Installer 再配布可能ファイルをプロジェクトに追加する」をご覧ください。Windows Installer 4.0 は、再配布可能ファイルとしては提供されていません。*

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
TokenPassword	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	暗号化されてプロジェクト ファイルに保存される EV トークン パスワードを取得または設定します。 ***メモ:*** *EV トークンのパスワードを指定する際には、以下の情報に注意してください:* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*
UpdateSettingsOnAppLaunch	基本の MSI	読み書きプロパティ	このプロパティを True に設定し、MsixHoursBetweenUpdateCheck、MsixShowPrompt、および MsixUpdateBlocksActivation プロパティを設定して、アプリの起動中に MSIX パッケージの更新プロパティを構成することができます。 アプリの起動中に MSIX パッケージのアップデート プロパティを構成しない場合は、このプロパティを False に設定します。
URLForYourFiles	基本の MSI、InstallScript MSI	読み書きプロパティ	データキャビネット ファイルをダウンロードするディレクトリを指定します。このプロパティでは、**http://www.yourcompany.com/download** の URL 形式が許可されています。 このプロパティは、現在のリリースの WebType プロパティが ewtDownloader (0) (ダウンロード) または ewtIFTW (1) (Web からインストール) に設定されている場合にのみ使用されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
Use64bitSigning	基本の MSI、InstallScript MSI、InstallScript、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	64 ビット署名フレームワークを使用してパッケージにデジタル署名を行う場合は、このプロパティを [True] に設定します。 32 ビット署名フレームワークを使用してパッケージにデジタル署名を行う場合は、このプロパティを [False] に設定します。
UseMyVersionInfo	基本の MSI、InstallScript MSI	読み書きプロパティ	Setup.exe のデフォルト InstallShield 著作権情報、ファイルの説明およびファイル バージョンをバーライドするには、このプロパティを [True] に設定します。著作権情報、ファイルの説明およびファイル バージョンを、それぞれ LauncherCopyright、LauncherFileDescription および LauncherFileVersion プロパティで指定します。 このプロパティは、次の条件を満たすリリースにのみ適用できます。 • 単一ファイル Setup.exe • 圧縮ファイル • ネットワーク イメージ メディア タイプ
UsePathVariableTestValues	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	読み書きプロパティ	パス変数 にテスト値を使用した場合は、このプロパティを True に設定して現時点での実際の値に変数を設定します。
UseProjectSkin	InstallScript、InstallScript オブジェクト	読み書きプロパティ	このブール型プロパティは、リリースの "スキンの指定" プロパティを [プロジェクト設定を使う] に設定するかどうかを取得または設定します。このプロパティは、[ダイアログ] ビューの [スキン] フォルダーで選択したスキンを使用します。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
VMConfig	基本の MSI、InstallScript、InstallScript MSI	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* この設定には、選択されたリリースを VM に配布するときに使用される VM 構成設定グループの名前を取得または設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
VMMachinePassword	基本の MSI、InstallScript、InstallScript MSI	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* VMConfig プロパティで選択されている VM 構成にパスワードが指定されている場合、そのパスワードが VMMachinePassword プロパティのデフォルト値として取り扱われます。選択されたリリースのパスワードをオーバーライドするにはVMMachinePassword プロパティを設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
VMMachineUserName	基本の MSI、InstallScript、InstallScript MSI	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* VMConfig プロパティで選択されている VM 構成にユーザー名が指定されている場合、そのユーザー名が VMMachinePassword プロパティのデフォルト値として取り扱われます。選択されたリリースのユーザー名をオーバーライドするには VMMachineUserName プロパティを設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
VMSnapShot	基本の MSI、InstallScript、InstallScript MSI	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* オプションで、リリースの配布に使用するスナップショットの名前を取得または設定します。 VMConfig プロパティで選択されている VM 構成にスナップショットが指定されている場合、そのスナップショットが VMSnapShotプロパティのデフォルト値として取り扱われます。 VM 構成にスナップショットが指定されていない場合、InstallShield は特定のスナップショットには戻りません。InstallShield は特定のスナップショットに戻さずに VM の電源をオンにして、VM にインストールをコピーします。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
VMStageMachineCopyPath	基本の MSI、InstallScript、InstallScript MSI	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* コピー パスは、リリースの配布先となる VM 上の場所を識別します。パスの最後のサブフォルダーとして、InstallShield が VM 上に作成するパスを使用できます。その他のパスは既存していなくてはなりません。 VMConfig プロパティで選択されている VM 構成にインストール先パスが指定されている場合、そのインストール先パスが VMStageMachineCopyPath プロパティのデフォルト値として取り扱われます。選択されたリリースのインストール先パスをオーバーライドするには VMStageMachineCopyPath プロパティを設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
VMStagePostBuild	基本の MSI、InstallScript、InstallScript MSI	読み書きブール値プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* ビルドが成功するたびに選択されたリリースを、InstallShield で自動的に配布するかどうかを指定します。VM にリリースを配布するには、以下のタスクが必要です: 1. (指定されている場合) VM を指定されたスナップショットに戻す 2. VM の電源をオンにする。 3. テスト用にビルドされたリリースを VM にコピーする。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
WebCreateDefaultPage	InstallScript	読み書きプロパティ	このブール型プロパティは、リリースの "Web デフォルトのページの作成" 設定を取得または設定します。このプロパティは、InstallShield がデフォルトの Web ページ (.htm ファイル) を作成して Disk1 フォルダーに配置するかどうかを指定します。
WebPageUrl	InstallScript	読み書きプロパティ	リリースの Web Page URL 設定を取得または設定します。このプロパティは、InstallShield の [ビルド] メニューにある [Web から実行] コマンドをクリックしたときに起動される Web ページの場所を指定します。 このプロパティで何も入力されなかった場合、ビルド時に Setup.htm が作成され、Disk1 フォルダーに配置されます。 このプロパティに URL を入力する場合、ページ名を含めないでください。また、末尾にスラッシュ (/) を使用することも避けてください。たとえば、Setup.htm ファイルの完全 URL が **http://www.mypages.com/setup/Setup.htm** の場合、ページ URL を **http://www.mypages.com/setup** のように入力します。 [ビルド] メニューで [Web から実行] コマンドをクリックしたとき、適切な URL (上記例の **http://www.mypages.com/setup/Setup.htm**) が起動されます。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
WebType	基本の MSI、InstallScript MSI	読み書きプロパティ	Web インストールセットアップパッケージの設定を示します。次のうちの 1 つに設定します。 • ewtDownloader (0) — ダウンローダー。リリースを Setup.exe と MSI パッケージの組み合わせとしてビルドします。エンドユーザーが Setup.exe をダウンロードして起動すると、すべてのファイルを含んだ MSI データベースが順にダウンロードされ実行されます。 • ewtIFTW (1) — Web からインストール。Setup.exe、MSI データベースおよび外部キャビネット (.cab) ファイルを組み合わせてこのリリースをビルドします。エンドユーザーは Setup.exe をダウンロードして実行し、それによって Setup.exe が MSI データベースをダウンロードして実行します。エンドユーザーのセットアッププタイプと機能の選択によって、要求された CAB ファイルだけがダウンロードされてインストールされるため、ダウンロード時間の縮小につながります。 アプリケーションのメンテナンスと修復では、インストール ソースとして URLForYourFiles プロパティで指定した URL が使用されます。 • ewtOneExe (2) — 1 つの実行プログラム。このリリースを単一の自己展開型 Setup.exe ファイルとしてビルドします。インストール パッケージは自己完結型のため、この設定は、多くの Web または FTP サイトからダウンロードされるパッケージに理想的です。インストールの全パッケージがダウンロードされ、ewtIFTW (1) (Web からインストール) の設定よりもダウンロード時間がことを確認してください。

テーブル 100 • ISWiRelease オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
WindowsAppPackage	基本の MSI	読み書きプロパティ	Windows アプリ パッケージ (.appx/.msix) をこのリリースのビルドの一部としてビルドする場合、このプロパティを True に設定します。次のシナリオをビルドするには、Windows アプリ パッケージが必要です。 • アプリを Windows ストア経由で配布する。 • Windows Server Nano をターゲットにする。 • UWP (ユニバーサル Windows プラットフォーム) を活用する Appx/MSIX をサイドロードします。 Windows アプリ パッケージ (.appx/.msix) をこのリリースのビルドの一部としてビルドしない場合、このプロパティを False に設定します。このプロパティ値を False に設定すると、.msi パッケージのみをビルドすることができます。
WrapMSIIntoCab	基本の MSI、InstallScript MSI	読み書きプロパティ	リリースの MSI データベースをキャビネット (.cab) ファイル内に配置するかどうかを指定します。 このプロパティは、現在の Web リリースの WebType プロパティが WebType (ダウンロード) に設定されている場合にのみ適用されます。

例

以下の Visual Basic の行は、プロジェクトを開き、リリースのプロパティ内の複数のメッセージボックスを表示します:

```
Dim pProject As ISWiProject

Set pProject = New ISWiProject
pProject.OpenProject "C:¥MySetups¥Commercial.ism", True

Dim pProdConfig As ISWiProductConfig
Set pProdConfig = pProject.ISWiProductConfigs.Item("Version 1")
' InstallScript プロジェクトでは、
' ISWiProductConfigs の要素は "Media" の 1 つだけです。

Dim pRelease As ISWiRelease
Set pRelease = pProdConfig.ISWiReleases.Item("NewRelease1")

Dim sProps As String
```

```
sProps = pRelease.Name & " はこれらのプロパティを含みます:" & vbNewLine
sProps = sProps & "ビルド場所: " & pRelease.BuildLocation & vbNewLine
sProps = sProps & "圧縮: " & pRelease.BuildLocation & vbNewLine
sProps = sProps & "デフォルト言語: " & pRelease.DefaultLang & vbNewLine
sProps = sProps & "Setup.exe: " & pRelease.SetupEXE


MsgBox sProps

pProject.CloseProject
```

ビルドのステータスイベント

ISWiRelease オブジェクトでは次のステータスイベントが発生します。

- Public Event ProgressIncrement(ByVal lIncrement As Long, pbCancel As Boolean)
- Public Event ProgressMax(ByVal lMax As Long, pbCancel As Boolean)
- Public Event StatusMessage(ByVal sMessage As String, pbCancel As Boolean)

これらのイベントは、ビルドプロセス中に発生してステータスのアップデートを提供します。ビルドを中止するには pbCancel を設定します。これらのイベントの使い方をデモンストレーションするサンプル コードは、「ビルドのステータス イベント」を参照してください。

ビルド メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

Build メソッドを呼び出して現在のリリースをビルドします。これは、リリースの完全な再ビルドです。

構文

Build ()

次に適用：

- ISWiRelease

BuildTablesOnly メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *マージ モジュール*

BuildTablesOnly メソッドを呼び出して、現在のリリースの Windows Installer テーブルのみをビルドします。

メモ: *このインストールをまだビルドしていない場合、新しい .msi ファイルが作成されますが、インストールにはファイルが追加されません。インストールを既にビルドしている場合、.msi ファイルは、すべてのテーブルがビルドされたとき更新されますが、ファイルは転送されません。*

構文

BuildTablesOnly ()

次に適用：

- ISWiRelease

BuildTablesRefreshFiles メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

BuildTablesOnly メソッドを呼び出すと、インストールの新しいファイルや変更されたファイルをはじめ、MSI ファイルが再ビルドされ Files テーブルが更新されます。

メモ: *変更されたファイルは、サイズまたはタイムスタンプが、ビルドにすでに含まれるコピーと異なる場合にのみ更新されます。削除されたファイルへの参照は、セットアップから削除されますが、ファイルは、ビルドの保存場所に残ります。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワークイメージである場合だけ作動します。*

構文

BuildTablesRefreshFiles ()

次に適用：

- ISWiRelease

CubFile プロパティ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*
- *マージ モジュール*

CubFile は、ISWiRelease.Build の前に指定する必要がある読み書き可能なプロパティです。このビルド オプションは、ISCmdBld.exe を使ってコマンドラインでリリースをビルドするのと同じです。

このプロパティに関連付けられている文字列パラメーターが、.cub ファイル名です。

次に適用：

- ISWiRelease

SignMedia プロパティ

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiRelease.SignMedia プロパティを設定して、メディアに署名するかどうかを示すことができます。また、署名する場合、署名するファイルも指定できます。

次のテーブルに、ISWiRelease.SignMedia プロパティの割り当てまたは取得を行うことのできる 4 つの状態を示します。

テーブル 101 ・ ISWiRelease.SignMedia プロパティの値

戻り値	種類	説明
esBoth (3)	読み書きプロパティ	基本の MSI プロジェクトと InstallScript MSI プロジェクトの場合: Windows Installer パッケージ (.msi ファイル) と Setup.exe に署名します。 InstallScript プロジェクトの場合: メディア ヘッダー ファイル (Data1.hdr) と Setup.exe の両方に署名します。
esMsi (2)	読み書きプロパティ	基本の MSI、InstallScript MSI およびマージ モジュール プロジェクトの場合: Windows Installer パッケージ (.msi ファイルまたは .msm ファイル) にのみデジタル署名します。 InstallScript プロジェクトの場合: メディア ヘッダー ファイル (Data1.hdr) に署名します。
esNone (4)	読み書きプロパティ	基本の MSI、InstallScript、および InstallScript MSI プロジェクトの場合: ビルド時にメディア ファイル (Setup.exe、Windows Installer パッケージまたはヘッダー ファイル) は署名されません。

テーブル 101 • ISWiRelease.SignMedia プロパティの値 (続き.)

戻り値	種類	説明
esSetupExe (1)	読み書きプロパティ	基本の MSI、InstallScript、および InstallScript MSI プロジェクトの場合: Setup.exe にのみデジタル署名します。

次に適用 :

- ISWiRelease オブジェクト

ISWiRemoveFile オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiRemoveFile は、InstallShield ユーザー インターフェイスのファイルまたはフォルダーの削除エントリを示します。ファイルまたはフォルダーの削除エントリを取得するには、ISWiRemoveFiles コレクションでアイテムを指定します。

メンバー

テーブル 102 • ISWiRemoveFile オブジェクトのメンバー

名前	種類	説明
FileKey	読み書きプロパティ	このファイルまたはフォルダーの削除エントリ用の FileKey プロパティを取得または設定します。
FileName	読み書きプロパティ	このファイルまたはフォルダーの削除エントリ用の FileName プロパティを取得または設定します。
DirProperty	読み書きプロパティ	このファイルまたはフォルダーの削除エントリ用の DirProperty プロパティを取得または設定します。
InstallMode	読み書きプロパティ	このファイルまたはフォルダーの削除エントリ用の InstallMode プロパティを取得または設定します。次の種類のうち 1 つを指定します。 • **erfimOnInstall (1)** – 関連コンポーネントがインストールされるときのみファイルまたはフォルダーを削除します。 • **erfimOnUninstall (2)** – 関連コンポーネントが削除されるときのみファイルまたはフォルダーを削除します。 • **erfimOnBoth (3)** – 関連コンポーネントがインストールまたは削除されるときにファイルまたはフォルダーを削除します。

ISWiSequenceRecord オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiSequenceRecord は、InstallShield ユーザー インターフェイスの [カスタム アクションとシーケンス] ビューのアクションを表します。

メンバー

テーブル 103 • ISWiSequenceRecord オブジェクトのメンバー

名前	種類	説明
名前	読み書きプロパティ	指定されたアクションの文字列名を保持します。
コメント	読み書きプロパティ	指定されたアクションのコメントを取得または設定します。
条件	読み書きプロパティ	指定されたアクションの条件を取得または設定します。
Sequence	読み書きプロパティ	指定されたアクションのシーケンス番号を取得または設定します。

例

次の VBScript コードは、プロジェクトの [インストール] – [ユーザー インターフェイス] シーケンスに、LaunchConditions アクションのシーケンス番号を表示します。

```
Set oISM = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")

oISM.OpenProject "C:¥MySetups¥MyProject.ism", True ' open read-only

Set oSequenceRecord = oISM.InstallUISequence("LaunchConditions")

MsgBox "LaunchConditions はシーケンス番号位置にあります " & oSequenceRecord.Sequence

oISM.CloseProject
```

次に適用 :

- ISWiSequence

ISWiSetupFile オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ISWiSetupFile は、プロジェクトに含まれるサポート ファイルを表します。サポート ファイルは、InstallShield の [サポート ファイル] ビュー (または [サポート ファイル/ビルボード] ビュー) の既存ファイルでも、AddAdvancedFile を呼び出して追加したファイルでもかまいません。

プロジェクトの ISWiSetupFiles コレクションを使用して特定のサポート ファイルにアクセスします。
ISWiSetupFiles の Item プロパティには、インデックス番号かファイルキーのいずれかを指定します。

メンバー

テーブル 104 • ISWiSetupFile オブジェクトのメンバー

名前	種類	説明
FileName	読み取り専用プロパティ	サポート ファイルのファイル名を取得します (パスなし)。
言語	読み書きプロパティ	サポートファイルの言語識別子を識別する文字列を取得します。 例: m_pFile.Language = "1033"
パス	読み取り専用プロパティ	サポート ファイルのソースの場所の完全修飾パス (ファイル名を含む) を取得します。あらゆるパス変数は、実際のパスへ解決されます。
Splash	読み書きプロパティ	このサポート ファイルがスプラッシュ画面 (True) かそうでないか (False) を指定します。
ターゲット	読み書きプロパティ	SUPPORTDIR 内のサポート ファイルの ターゲット ディレクトリを取得または設定します。次の行は、SUPPORTDIR の下にある MyFolder1¥MyFolder2 フォルダーにサポートファイルを保存します。 m_pFile.Target = "MyFolder1¥MyFolder2"

例

ファイル キーは、[ダイレクト エディター] ビューの ISSetupFile テーブルの最初のエントリとしてのみ IDE で表示されます。ファイル キーの代わりに、次の例のようなコードを記述して、特定のファイル名の存在を確認することができます。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pSetupFile As ISWiSetupFile

For Each pSetupFile In pProj.ISWiSetupFiles
    If UCase(pSetupFile.FileName) = "MYSUPPORTFILE.EXT" Then
        Exit For
    End If
Next

pProj.SaveProject
pProj.CloseProject
```

ファイル名 MySupportFile.ext がプロジェクトのセットアップ ファイル中に見つかると For ループが終了しますが、この時 pSetupFile には MySupportFile.ext の ISWiSetupFile オブジェクトがコピーされています。

ISWiSetupType オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ISWiSetupType は、プロジェクトに属するセットアップの種類を表します。セットアップの種類は、InstallShield ユーザー インターフェイスの [サポート ファイル] ビューにある既存のセットアップの種類でも、AddSetupType を呼び出して追加したものでもかまいません。

プロジェクトの ISWiSetupTypes コレクションを使用して特定のセットアップの種類にアクセスします。ISWiSetupFiles の Item プロパティには、インデックス番号かセットアップの種類名のいずれかを指定します。セットアップの種類名は、[セットアップの種類] ビューのツリーコントロールの IDE で表示できます。

メンバー

テーブル 105 ・ ISWiSetupType オブジェクトのメンバー

名前	種類	説明
名前	読み取り専用プロパティ	セットアップの種類の名前を格納します。
説明	読み書きプロパティ	編集ボックス名によって示された情報を、テキストを入力するかコンボボックスのリストから既存のパス変数(アプリケーションの種類の場合は既存のアプリケーションの種類)を選択することによって指定します。(この説明は、スクリプトで SdSetupTypeEx を呼び出した場合のみ 、インストール プロセス中にエンドユーザーに表示されます。)
DisplayName	読み書きプロパティ	この文字列プロパティでは、ユーザーに表示するセットアップの種類の名前を指定できます。

例

次の Visual Basic コードは、特定のセットアップの種類のプロパティの設定方法を示したものです。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pSetupType As ISWiSetupType

Set pSetupType = pProj.ISWiSetupTypes("CSSaveSpace")
pSetupType.DisplayName = "クライアント セットアップ – 容量の節約"

pProj.SaveProject
pProj.CloseProject
```

ISWiShellProperty オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiShellProperty オブジェクトは、Windows シェルが設定するショートカット プロパティを示します。シェルが設定できるプロパティは、Windows SDK に含まれている propkey.h で定義されています。

メンバー

テーブル 106 : ISWiShellProperty オブジェクト メンバー

名前	種類	説明
名前	読み書きプロパティ	MsiShortcutProperty テーブルで指定されたシェル プロパティのプライマリ キー名を取得、または設定します。
PropertyName	読み書きプロパティ	Shell プロパティを取得または設定します。プロパティは、Windows SDK に含まれている propkey.h で定義されています。 PropertyName は通常、GUID とプロパティ ID で構成されます。例: **{9F4C2855-9F79-4B39-A8D0-E1D42DE1D5F3}, 9**
PropertyValue	読み書きプロパティ	プロパティの文字列を取得または設定します。

ISWiShortcut オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiShortcut オブジェクトは [ショートカット] エクスプローラーのショートカットを表します。ショートカットには、ISWiShortcuts コレクションの既存の項目、または AddShortcut を呼び出して作成したものを使用できます。

このオブジェクトのプロパティを使用して、IDE とほぼ同じ方法でショートカットの属性を設定します。

メンバー

テーブル 107 • ISWiShortcut オブジェクトのメンバー

名前	プロジェクト	種類	説明
AddShellProperty	基本の MSI、InstallScript MSI、マージ モジュール	メソッド	現在のショートカット オブジェクトに Shell プロパティを追加します。
引数	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	対象のファイルに渡されるコマンドライン引数を取得または設定します。"%1" をファイル名として代用できます。次のステートメントを使用すると、対象ファイルをサイレントモードで起動できます。 m_pShortcut.Arguments = "–s"
DeleteShellProperty	基本の MSI、InstallScript MSI、マージ モジュール	メソッド	現在のショートカット オブジェクトから、指定された Shell プロパティを削除します。
Description	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	このショートカットに対して入力した説明文が含まれます。説明ツール ヒントとして表示されます。このプロパティが文字列エントリを使用する場合、Description には、ISWiProject の ActiveLanguage プロパティで指定される文字列エントリの値が入ります。
DisplayName	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	このショートカットに入力した表示名を含みます。この名前は、ローカライズ可能な値で、[ショートカット] エクスプローラでショートカットを識別する名前の代わりにこのショートカットの名前がインストールされた場合、これがショートカットの名前として使用されます。このプロパティが文字列エントリを使用する場合、DisplayName には、ISWiProject の ActiveLanguage プロパティで指定される文字列の値が入ります。

テーブル 107 • ISWiShortcut オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
DoNotHighlightAsNew	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	エンド ユーザーが製品をインストールした後に、ショートカットが [スタート] メニューで新規アイテムとして強調表示されるように構成されているかどうかを示すブール型プロパティを取得または設定します。これは、ターゲット システム上で個別のアイテムに対して [[スタート] メニューのカスタマイズ] ダイアログ ボックスで **[新しくインストールされたプログラムを強調表示する]** チェック ボックスをクリアした場合と同じ効果を持ちます。 [スタート] メニューでショートカットが新しくインストールされたアイテムとして強調表示されないように防ぐには、このブール型プロパティを True に設定します。ショートカットの強調表示を有効化するには、このブール型プロパティを False に設定します。 この強調表示に関する詳細は、「[スタート] メニューにあるショートカットを、新しくインストールされた製品として強調表示しないように防ぐ」を参照してください。
EnableWin8StartPin	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ショートカットを Windows 8 ターゲット システム上で、デフォルトでスタ—ト画面にピン留めするように構成されているかどうかを示すブール型プロパティを取得または設定します。 デフォルトでショートカットをピン留めするには、このブール型プロパティを True に設定します。ピン留めを禁止するには、このブール型プロパティを False に設定します。 このピン留め機能についての詳細は、「Windows 8 スタート画面への初回のショートカットのピン留めを抑制する」を参照してください。
HotKey	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ショートカットのアクセラレータキーを長整数型で格納します。

テーブル 107 • ISWiShortcut オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
IconFile	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ショートカットのアイコンが含まれるファイルの完全なパスを入力します。アイコンリソースが含まれる実行可能ファイル、または .ico ファイルを指定する必要があります。Windows Installer ベースのプロジェクトのみ:) このプロパティを空白のままにすると、このコンポーネントのキーファイルはショートカットのアイコンとして自動設定されます。 Windows Installer では、コンポーネントがアドバタイズされるときに別のアイコンが必要なため、InstallShield は指定した実行ファイルからアイコンを取り出します。実行形式ファイルに複数のアイコンリソースがある場合は、必ず IconIndex プロパティを設定してください。
IconIndex	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	IconFile に複数のアイコンリソースがある場合、長整数型として指定されているアイコンのインデックスが含まれます。 負の数以外の整数を指定すると、実行可能ファイルのアイコンリソースの順番が参照されます。たとえば 0 はファイルの最初のアイコン、1 は 2 番目、2 は 3 番目というようになります負の数字は特定のリソース ID を参照するために使用します。たとえばアイコン インデックス −12 は、リソース ID が 12 のアイコンを示します。
InternetShortcut	InstallScript	読み書きプロパティ	このショートカットのインターネット ショートカットのプロパティを取得または設定します。ショートカットの [ターゲット] プロパティで指定される URL をポイントするインターネット ショートカットかどうかを指定します。このプロパティはブール値です。
ISWiShellProperties	基本の MSI、DIM、InstallScript、InstallScript MSI、マージ モジュール	コレクション	現在のショートカット オブジェクトのすべてのシェルのプロパティを含みます。
Name	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み取り専用プロパティ	ショートカット名を取得します。これは [ショートカット] エクスプローラーに表示される名前と同じです。ショートカットの [表示名] プロパティと混同しないようにしてください。

テーブル 107 • ISWiShortcut オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
PreventPinning	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	エンド ユーザーが製品をインストールした後、ショートカットをタスクバーおよび [スタート] メニューにピン留めするためのコンテキスト メニュー コマンドを表示するかどうかを示すブール型プロパティを取得または設定します。 ショートカットをタスクバーおよび [スタート] メニューにピン留めするためのコンテキスト メニュー コマンドを隠すには、このブール型プロパティを True に設定します。コンテキスト メニュー コマンドを表示してピン留めを許可するには、このブール型プロパティを False に設定します。 このピン留め機能についての詳細は、「エンド ユーザーがタスクバーまたは [スタート] メニューにショートカットをピン留めできるかどうかを指定する」を参照してください。
ReplaceExistingIfFound	InstallScript	読み書きプロパティ	このショートカットの "既存のショートカットを置換" 設定を取得または設定します。ここで、ショートカットがターゲット システムの同じ場所にある同じ表示名を持つ既存のショートカットを置き換えるかどうかを指定します。このプロパティはブール値です。
ShowCmd	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	以下のいずれかのオプションを指定して、ファイルを開くモードを決定します。 • **scNormal (1)** – プログラムは通常のウィンドウ サイズで起動します。 • **scMaximized (3)** – プログラムは全画面表示で起動します。 • **scMinNoActive (7)** – プログラムは最小ウィンドウで起動し、タスクバーだけに表示されます。

テーブル 107 • ISWiShortcut オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
ターゲット	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ショートカットの対象ファイルのパスとファイル名を指定します。パスをハードコード化せずに、Windows Installer ディレクトリ プロパティを角カッコで囲んで使用できます (例、[INSTALLDIR]MyApp.exe)。または InstallScript プロジェクトでは、システム変数を山かっこで囲んで使用できます(例、<TARGETDIR>¥MyApp.exe)。 Windows Installer ベースのプロジェクトのみ:) このプロパティを空白のままにすると、コンポーネントのキーファイルはショートカットのターゲットとして自動設定されます。
Uninstall	InstallScript	読み書きプロパティ	アプリケーションをアンインストールしたときにショートカットを削除するかどうか指定するアンインストールのプロパティを取得または設定します。このプロパティはブール値です。
UserType	InstallScript	読み書きプロパティ	このショートカットのタイプのプロパティを取得または設定します。全ユーザーに対する共通領域で、アプリケーションをインストールするユーザーに対して、ショートカットを個人ショートカットのリストに表示するかどうか、または、ALLUSERS システム関数に基づいて自動的に判断されるようにするかを選択できます。以下のいずれかのオプションを指定して、ファイルを開くモードを決定します。 • estAutomatic (0) – ALLUSERS がゼロ以外の場合は共通のショートカットのリストに、FALSE の場合は個人のショートカットのリストに表示されます。 • estCommon (1) – ショートカットは、すべてのユーザーに共通のショートカットのリストに表示されます。 • estPersonal (2) – ショートカットは、アプリケーションをインストールするユーザー用個人用ショートカットのリストに表示されます。

テーブル 107 • ISWiShortcut オブジェクトのメンバー (続き.)

名前	プロジェクト	種類	説明
WorkingDirectory	基本の MSI、InstallScript、InstallScript MSI、マージ モジュール	読み書きプロパティ	ショートカットの [開始] フォルダーとなる文字列を指定します。パスをハードコード化せずに、Windows Installer ディレクトリ プロパティを角カッコで囲んで使用できます (例、[INSTALLDIR]MyApp.exe)。または InstallScript プロジェクトでは、システム変数を山かっこで囲んで使用できます(例、<TARGETDIR>¥MyApp.exe)。

次に適用 :

- ISWiFolder

AddShellProperty メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

AddShellProperty メソッドは、現在のショートカット オブジェクトに シェル プロパティを追加します。シェルが設定できるプロパティは、Windows SDK に含まれている propkey.h で定義されています。

ショートカット用のシェルのプロパティの設定についての情報は、「カスタム アクション シェルのプロパティを設定する」を参照してください。

構文

Function AddShellProperty (sName As String) As ISWiShellProperty

パラメーター

テーブル 108 • AddShellProperty メソッドのパラメーター

パラメーター	説明
sName	ショートカットに追加するシェル プロパティの名前を指定する文字列を渡します。

戻り値

AddShellProperty メソッドは、ISWiShellProperty オブジェクトを戻します。

例

次のサンプル VBScript は、シェル プロパティをショートカットに追加します:

```
Dim pShellProperty As ISWiShellProperty
Set pShellProperty = pShortcut.AddShellProperty("MyShellProperty")
```

次に適用：

- ISWiShortcut

DeleteShellProperty メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

DeleteShellProperty メソッドは、現在のショートカット オブジェクトから、指定されたシェル プロパティを削除します。

ショートカット用のシェルのプロパティの設定についての情報は、「カスタム アクション シェルのプロパティを設定する」を参照してください。

構文

```
DeleteShellProperty (pShellProp As ISWiShellProperty) As Long
```

パラメーター

テーブル 109 • DeleteShellProperty メソッドのパラメーター

パラメーター	説明
pShellProp	ISWiShellProperty オブジェクトを渡して、ショートカットから削除するシェル プロパティを指定します。

例

次のサンプル VBScript は、シェル プロパティをショートカットから削除します:

```
Dim pShellProperty As ISWiShellProperty
   For Each pShellProperty In pShortcut.ISWiShellProperties
      pShortcut.DeleteShellProperty pShellProperty
   Next
```

次に適用：

- ISWiShortcut

ISWiSISProperty オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*
- *MSI データベース*
- *MSM データベース*
- *トランスフォーム*

ISWiSISProperty は、概要情報ストリーム にあるプロパティを表示します。ISWiSISProperties コレクションの中のアイテムを指定して、フォルダーを取得できます。次の概要情報ストリーム プロパティ名がサポートされています。

- タイトル
- 件名
- 作成者
- キーワード
- PackageCode
- TemplateSummary
- Comments
- スキーマ
- RequireAdministrativePrivileges

InstallShield では、これらのプロパティの一部に文字列エントリを使用しなくてはなりません。その場合、これらのプロパティには、ISWiProject の ActiveLanguage プロパティによって識別される文字列の値が含まれます。

メンバー

テーブル 110 · ISWiSISProperty オブジェクトのメンバー

名前	種類	説明
名前	読み取り専用プロパティ	プロパティ名を含みます。上に表示されている既存の "概要情報ストリーム" プロパティの 1 つであることが必要です。
値	読み書きプロパティ	プロパティの値を取得または設定します。 このプロパティが文字列エントリを使用する場合、Value には、ISWiProject の ActiveLanguage プロパティで識別される文字列の値が入ります。

例

以下の VBScript の例は、Title を **[Windows Installer セットアップ パッケージ]** に、RequireAdministrativePrivileges を **[いいえ]** に設定します。

```
Option Explicit

Dim pProject
Set pProject = CreateObject ("IswiAutoAutomation Interface Version.ISWiProject")

pProject.OpenProject "C:¥MySetups¥SampleProject.ism"

Dim pTitle, sTitle
sTitle = "Windows Installer セットアップ パッケージ"

Set pTitle = pProject.ISWiSISProperties.Item("Title")
pTitle.Value = sTitle

Dim pReqAdminPriv, sReqAdminPriv
sReqAdminPriv = "いいえ"

set pReqAdminPriv = pProject.ISWiSISProperties.Item("RequireAdministrativePrivileges")
pReqAdminPriv.Value = sReqAdminPriv

pProject.SaveProject
pProject.CloseProject
```

次に適用 :

- ISWiProject

ISWiSQLConnection オブジェクト

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLConnection は、InstallShield インターフェイスの [SQL スクリプト] ビュー内の SQL Server 接続を意味します。ISWiSQLConnections コレクションを使用して、ISWiSQLConnection オブジェクトを読み出すことができます。

メンバー

テーブル 111 • ISWiSQLConnection オブジェクトのメンバー

名前	種類	説明
AddSQLRequirement	メソッド	AddSQLRequirement メソッドを呼んで特定のサーバーバージョンを指定して、SQL Server がアプリケーションの最小要件を満たすかどうかを判別します。
AddSQLScript	メソッド	AddSQLScript メソッドを呼んで、SQL 接続に新しいスクリプト ファイルを追加します。 ***メモ*** *AddSQLScript メソッドは、SQLScript1、SQLScript2、または SQLScript3 などの名前を持つ ISSQLScriptFile テーブル キーを生成します。プロジェクトに複数の SQI スクリプト ファイルが含まれる場合、この命名規則を使ってファイルを区別するのが困難なこともあります。したがって、キーを指定できるようにするには、AddSQLScriptEx メソッドを使用します。*
AddSQLScriptEx	メソッド	AddSQLScriptEx メソッドを呼び出して、新しいスクリプト ファイルを SQL 接続に追加し、渡された文字列から有効な名前を生成します。
Authentication	読み書きプロパティ	この文字列は SQL Server に接続するとき、認証モードを取得または設定します。
Comments	読み書きプロパティ	この文字列プロパティは、インストール プロジェクトに含める内部コメントを取得または設定します。
Database	読み書きプロパティ	この文字列はデータベース名を取得または設定します。実行時にインストールは、この接続内に含まれる SQL スクリプト ファイルをインストールするためのデフォルトのデータベースとしてこれを使用します。この値にプロパティを使用することもできます。プロパティを使用すると、実行時にこのデータベースの名前を動的に設定することができます。
DeleteSQLScript	メソッド	DeleteSQLScript メソッド を呼び出して、InstallShield インターフェイスで、SQL 接続から SQL スクリプト ファイルを削除します。
GetDatabaseServer	メソッド	GetDatabaseServer メソッドは ISWiSQLDatabaseServer オブジェクトを戻します。
InsertSQLScript	メソッド	InsertSQLScript メソッド を呼び出して、InstallShield インターフェイスで、SQL 接続に既存のスクリプト ファイルを挿入します。

テーブル 111 • ISWiSQLConnection オブジェクトのメンバー (続き.)

名前	種類	説明
名前	読み書きプロパティ	この文字列プロパティは、ServicePackLevel 必要条件を取得または設定します。セットアッププロジェクトでこの接続を一意に識別するのに使う内部名を入力します。
Order	読み書きプロパティ	この整数のプロパティは、接続の順番を取得または設定します。
Password	読み書きプロパティ	この文字列は SQL Server 認証を使用してサーバーに接続するように選択するとパスワードを取得または設定します。
RemoveSQLRequirement	メソッド	指定された SQL 要件をターゲット製品から削除します。
SelectDatabaseServer(Name As String) As Boolean	読み書きプロパティ	この文字列は選択したデータベースサーバー名を取得または設定します。実行時にインストールは接続を確立し、一覧されている通りにサーバーのシーケンスに基づいて、設定されたデータベース サーバーの指定された要件確認を行います。
サーバー	読み書きプロパティ	この文字列はサーバー名を取得または設定します。実行時に、インストールはこのサーバー名を使用して SQL Server に接続します。このサーバーは、この接続内に含まれる SQL スクリプトをインストールするのに使用されます。この値にプロパティを使用することもできます。プロパティを使用すると、実行時に、このサーバーの名前を動的に設定することができます。[プロパティ] の注釈は必ず使用します。
UserName	読み書きプロパティ	この文字列は SQL Server 認証を使用してサーバーに接続するように選択するとユーザー名を取得または設定します。

次に適用：

- ISWiProject

AddSQLRequirement メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

AddSQLRequirement は SQL 要件をターゲット製品に追加し、ISWiSQLRequirement オブジェクトを戻します。

構文

Function AddSQLRequirement() As ISWiSQLRequirement

戻り値

このメソッドは ISWiSQLRequirement オブジェクトを戻します。

次に適用：

- ISWiSQLConnections

AddSQLScript メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

AddSQLScript メソッドを呼んで、SQL 接続に新しいスクリプト ファイルを追加します。

メモ: *AddSQLScript メソッドは、SQLScript1、SQLScript2、または SQLScript3 などの名前を持つ ISSQLScriptFile テーブル キーを生成します。プロジェクトに複数の SQI スクリプト ファイルが含まれる場合、この命名規則を使ってファイルを区別するのが困難なこともあります。したがって、キーを指定できるようにするには、AddSQLScriptEx メソッドを使用します。*

構文

Function AddSQLScript() As ISWiSQLScript

戻り値

このメソッドは ISWiSQLScript オブジェクトを戻します。

次に適用：

- ISWiSQLConnections

AddSQLScriptEx メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*

AddSQLScriptEx メソッドを呼んで、SQL 接続に ISSQLScriptFile エントリを追加します。このメソッドは、渡された文字列から有効な名前を生成します。このメソッドは、ISSQLScriptFile テーブルに追加されるエントリ名が一意であること、またその文字数が 47 文字未満であることを保証します。

構文

Function AddSQLScriptEx (ByVal sCandidateName As String) As ISWiSQLScript

パラメーター

テーブル 112 · AddSQLScriptEx メソッドのパラメーター

パラメーター	説明
sCandidateName	SQL 接続に追加するスクリプトの名前を指定します。 ファイル名が無効な文字を含んでいたり、キーに使用するには長すぎる場合があります。しかし、このパラメーターでファイル名を渡す場合、AddSQLScriptEx メソッドがファイル名に基づいた有効なキーを使ってエントリを作成するので、SQL スクリプトを [SQL スクリプト] ビューの接続に追加した場合と同じ結果となります。 空白文字列 (“”) を指定すると、AddSQLScriptEx メソッドは SQLScript1、SQLScript2、または SQLScript3 などの名前で ISSQLScriptFile テーブル キーを生成します。これは、AddSQLScript メソッド を使って ISSQLScriptFile エントリを追加した場合と同じです。

戻り値

このメソッドは ISWiSQLScript オブジェクトを戻します。

次に適用：

- ISWiSQLConnections

DeleteSQLScript メソッド

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

DeleteSQLScript メソッド を呼び出して、InstallShield インターフェイスで、SQL 接続から SQL スクリプト ファイルを削除します。

構文

Function DeleteSQLScript(Script As ISWiSQLScript) As Long

パラメーター

テーブル 113 • DeleteSQLScript メソッド パラメーター

パラメーター	説明
Script	InstallShield インターフェイスで、SQL 接続から削除したいスクリプトの名前を指定します。

次に適用：

- ISWiSQLConnections

GetDatabaseServer メソッド

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

GetDatabaseServer メソッドを呼び出して ISWiSQLDatabaseServer オブジェクトを戻します。

構文

Sub SetPredefinedRequirement(RHS As ISWiSQLServerVersion)

戻り値

このメソッドは ISWiSQLDatabaseServer オブジェクトを戻します。

次に適用：

- ISWiSQLConnections

InsertSQLScript メソッド

***プロジェクト•** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*

- *InstallScript MSI*

InsertSQLScript メソッド を呼び出して、InstallShield インターフェイスで、SQL 接続に既存のスクリプト ファイルを挿入します。

構文

Function InsertSQLScript(strScriptPath As String) As ISWiSQLScript

パラメーター

テーブル 114 • InsertSQLScript メソッド パラメーター

パラメーター	説明
strScriptPath	InstallShield インターフェイスで、SQL 接続の下に挿入するスクリプト ファイルへのパスを指定します。

戻り値

このメソッドは ISWiSQLScript オブジェクトを戻します。

次に適用：

- ISWiSQLConnections
- ISWiSQLDatabaseServer

RemoveSQLRequirement メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

RemoveSQLRequirement メソッドは、指定された SQL 要件をターゲット製品から削除します。

次に適用：

- ISWiSQLConnections

ISWiSQLDatabaseServer オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLDatabaseServer は、実行時にバージョン要件によってターゲットとして選択可能なデータベースサーバーを意味します。ISWiSQLDatabaseServers コレクションを使用して、ISWiSQLDatabaseServer オブジェクトを読み出すことができます。

メンバー

テーブル 115 • ISWiSQLDatabaseServer オブジェクトのメンバー

名前	種類	説明
InsertSQLScript	メソッド	InsertSQLScript メソッド を呼び出して、InstallShield インターフェイスで、 SQL 接続に既存のスクリプト ファイルを挿入します。
ISWiSQLRequirements	コレクション	この接続に関連付けられている ISWiSQLRequirements のコレクションを戻します。
ISWiSQLScripts	コレクション	この接続に関連付けられている ISWiSQLScripts のコレクションを戻します。
名前	読み書きプロパティ	この文字列プロパティは、データベース サーバーの接続名を取得または設定します。
Order	読み書きプロパティ	この整数のプロパティは、接続の順番を取得または設定します。
Password	読み書きプロパティ	この文字列は SQL Server 認証を使用してサーバーに接続するように選択するとパスワードを取得または設定します。
SelectDatabaseServer(Name As String) As Boolean	読み書きプロパティ	この文字列は選択したデータベースサーバー名を取得または設定します。実行時にインストールは接続を確立し、一覧されている通りにサーバーのシーケンスに基づいて、設定されたデータベース サーバーの指定された要件確認を行います。
サーバー	読み書きプロパティ	この文字列はサーバー名を取得または設定します。実行時に、インストールはこのサーバー名を使用して SQL Server に接続します。このサーバーは、この接続内に含まれる SQL スクリプトをインストールするのに使用されます。この値にプロパティを使用することもできます。プロパティを使用すると、実行時に、このサーバーの名前を動的に設定することができます。[プロパティ] の注釈は必ず使用します。
UserName	読み書きプロパティ	この文字列は SQL Server 認証を使用してサーバーに接続するように選択するとユーザー名を取得または設定します。

次に適用 :

- ISWiSQLConnections

ISWiSQLReplace オブジェクト

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLReplace を利用すると、InstallShield インターフェイスで、スクリプト ファイルに関連付けられている [テキスト置換] タブ内のプロパティを取得し設定することができます。ISWiSQLReplaces コレクションを使用して、ISWiSQLReplace オブジェクトを読み出すことができます。

メンバー

テーブル 116 • ISWiSQLReplace オブジェクトのメンバー

名前	種類	説明
名前	読み書きプロパティ	この文字列プロパティは [SQL スクリプト置換] オブジェクトの名前を取得または設定します。
Search	読み書きプロパティ	この文字列プロパティは、探しているテキスト文字列を取得または設定します。
Replace	読み書きプロパティ	この文字列プロパティは、テキストの置換実行中、既存の文字列を置換したいテキスト文字列を取得または設定します。
MatchWholeWord	読み書きプロパティ	これは、ブール型プロパティです。インストール中、このプロパティは単語単位の一致のみを識別するのに使われます。
MatchCase	読み書きプロパティ	これは、ブール型プロパティです。インストール実行中、このプロパティはテキスト置換中に検索された単語の大文字小文字の識別が確実に一致することを示します。
ReplaceOnce	読み書きプロパティ	これは、ブール型プロパティです。インストール中、このプロパティは、異なる文字列で検索して置換する文字列が、複数回にわたって置換されることがないことを示します。
PreserveCase	読み書きプロパティ	これは、ブール型プロパティです。インストール中、このプロパティはテキスト置換の最中に大文字と小文字が保持されることを示します。

次に適用：

- ISWiSQLScript

ISWiSQLRequirement オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLRequirement は、InstallShield インターフェイスで、新しい SQL 接続用の必要条件プロパティタブを意味します。ISWiSQLRequirements コレクションを使用して、ISWiSQLRequirement オブジェクトを読み出すことができます。

メンバー

テーブル 117 • ISWiSQLRequirement オブジェクトのメンバー

名前	種類	説明
AnyGreaterServicePack	読み書きプロパティ	これは、ブール型プロパティです。ServicePackLevel 必要条件の制限を取得または設定します。
AnyGreaterVersion	読み書きプロパティ	これは、ブール型プロパティです。MajorVersion 必要条件の制限を取得または設定します。
Disable	読み書きプロパティ	このブール型プロパティは SQL 必要条件を無効にします。
MajorVersion	読み書きプロパティ	この文字列プロパティは、MajorVersion 必要条件を取得または設定します。
名前	読み書きプロパティ	この文字列プロパティは、接続用にサポートされている SQL Server バージョンを判断するのに必要な [SQL 必要条件] オブジェクトの名前を取得または設定します。
ServicePackLevel	読み書きプロパティ	この文字列プロパティは、ServicePackLevel 必要条件を取得または設定します。
SetPredefinedRequirement	メソッド	インストール作成者が決めた一組の定義済みの値から、必要条件を構成します。

次に適用：

- ISWiSQLConnections

SetPredefinedRequirement メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*

- *InstallScript*
- *InstallScript MSI*

SetPredefinedRequirement メソッドを呼んでインストール プロジェクトに SQL 接続を追加するときの必要条件を指定します。

構文

Sub SetPredefinedRequirement(RHS As ISWiSQLServerVersion)

定数

以下は、ISWiSQLServerVersion オブジェクトタイプに関連付けられている定数のリストです。

- essv2K
- essv2Ksp1
- essv2Ksp2
- essv2Ksp3
- essv65
- essv65sp1
- essv65sp2
- essv65sp3
- essv65sp4
- essv65sp5
- essv65sp5a
- essv65sp5aUpdate
- essv7
- essv7sp1
- essv7sp2
- essv7sp3
- essv7sp4

戻り値

このメソッドは ISWiSQLRequirement オブジェクトを戻します。

次に適用：

- ISWiSQLRequirement

ISWiSQLScript オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiSQLScript は、InstallShield の [SQL スクリプト] ビュー内で [SQL スクリプト] エクスプローラーにある SQL スクリプト ファイルを意味します。ISWiSQLScripts コレクション を利用して ISWiSQLScript オブジェクトを読み出すことができます。

メンバー

テーブル 118 • ISWiSQLScript オブジェクトのメンバー

名前	種類	説明
AddSQLScriptError	メソッド	ISWiSQLScriptError オブジェクトを作成し戻します。
AddSQLScriptReplacement	メソッド	ISWiSQLReplace オブジェクトを作成し戻します。
Comments	読み書きプロパティ	この文字列プロパティは、インストール プロジェクトに含める内部コメントを取得または設定します。スクリプトに関する注釈を入力することもできます。入力された注釈は保存されインストール プロジェクト内でのみ使用されます。
コンポーネント	読み書きプロパティ	この文字列プロパティは、SQL スクリプト オブジェクトに関連付けられている機能に関連付けられているコンポーネントの名前を取得または設定します。
条件	読み書きプロパティ	 ***プロジェクト:*** *このプロパティは、次のプロジェクト タイプで使用できます:* • *基本の MSI* • *DIM* • *InstallScript MSI* このプロパティは、SQL スクリプトをインストールまたはアンインストール中に実行するかどうかを判断するために実行時に評価される条件を指定します。この条件が True と評価された場合に、スクリプトが実行します。 このプロパティは、[SQL スクリプト] ビュー内の SQL スクリプトの [ランタイム] タブにある [スクリプト条件] 領域で入力した条件に対応します。
DeleteSQLReplace	メソッド	ISWiSQLReplace オブジェクトを削除します。
DeleteSQLScriptError	メソッド	ISWiSQLScriptError オブジェクトを削除します。
ErrorHandling	読み書きプロパティ	次の値から 1 つを取得または設定します。 • esehAbort (2) — エラー時に、インストールを中止する。 • esehGotoNextScript (0) — エラー時に、次のスクリプトへ移動する。 • esehGotoNextStatement (1) — エラー時に、次のステートメントへ移動する。

テーブル 118 • ISWiSQLScript オブジェクトのメンバー (続き.)

名前	種類	説明
FullPath	読み書きプロパティ	この文字列プロパティは、インストールで実行したい SQL スクリプト オブジェクトに関連付けられている .sql ファイルのパスを取得または設定します。
InstallText	読み書きプロパティ	この文字列プロパティは、インストール中、スクリプトが実行されるときのステータステキストを取得または設定します。このステータステキストは、ステータスメッセージの中に表示され、エンドユーザーには、スクリプトが実行中に表示されます。
ISWiSQLScriptErrors	読み取り専用プロパティ	この SQL スクリプトに関連付けられている ISWiSQLScriptErrors のコレクションを戻します。
名前	読み書きプロパティ	この文字列プロパティは、インストールで実行したい SQL スクリプト オブジェクトの名前を取得または設定します。
Order	読み書きプロパティ	この整数プロパティは、スクリプト ファイルの実行順序を取得または設定します。
RunOnInstall	読み書きプロパティ	このブール型プロパティは、インストール中の SQL スクリプトの実行スケジュールを組むかどうかを指定します。
RunOnLogon	読み書きプロパティ	このブール型プロパティは、ログイン中のに選択された SQL スクリプトを実行するかどうかを指定します。このプロパティは、[SQL スクリプト] ビューにある SQL スクリプトの [ランタイム] タブにある [ログイン中にスクリプトを実行] チェック ボックスに対応します。
RunOnRollback	読み書きプロパティ	このブール型プロパティは、ロールバック中の SQL スクリプトの実行スケジュールを組むかどうかを指定します。このプロパティは、[SQL スクリプト] ビューにある SQL スクリプトの [ランタイム] タブにある [ロールバック中にスクリプトを実行] チェック ボックスに対応します。
RunOnUninstall	読み書きプロパティ	このブール型プロパティは、アンインストール中の SQL スクリプトの実行スケジュールを組むかどうかを指定します。
UninstallText	読み書きプロパティ	この文字列プロパティは、アンインストール中、スクリプトが実行されるときのステータステキストを取得または設定します。このステータステキストは、ステータスメッセージの中に表示され、エンドユーザーには、スクリプトが実行中に表示されます。

テーブル 118 • ISWiSQLScript オブジェクトのメンバー (続き.)

名前	種類	説明
バージョン	読み書きプロパティ	このプロパティは、SQL スクリプト ファイルのバージョン番号を取得または設定します。詳細については、「SQL スクリプト ファイルのバージョン番号を指定する」を参照してください。

AddSQLScriptError メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

AddSQLScriptError メソッドを呼び出して、カスタム エラー処理エントリを SQL スクリプトに追加します。

構文

Function AddSQLScriptError()

戻り値

このメソッドは ISWiSQLScriptError オブジェクトを戻します。

次に適用 :

- ISWiSQLScript

AddSQLScriptReplacement メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

AddSQLScriptReplacement メソッドを呼んで、SQL スクリプト ファイルの中の異なる文字列で置き換えたい文字列を指定し置換します。

構文

Function AddSQLScriptReplacement() As ISWiSQLReplace

戻り値

このメソッドは ISWiSQLReplace オブジェクトを戻します。

次に適用：

- ISWiSQLScript

DeleteSQLReplace メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

DeleteSQLReplace メソッドを呼んで、追加したテキスト置換エントリを削除します。

構文

Function DeleteSQLReplace(Replace As ISWiSQLReplace) As Long

次に適用：

- ISWiSQLScript

DeleteSQLScriptError メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

DeleteSQLScriptError メソッドを呼んで、追加済みのカスタム エラー処理エントリを削除します。

構文

Function DeleteSQLScriptError(ByVal pScriptError As ISWiSQLScriptError)

次に適用：

- ISWiSQLScript

ISWiSQLScriptError オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLScriptError は、InstallShield の [SQL スクリプト] ビューにある SQL スクリプト ファイルの カスタム エラー処理エントリを示します。ISWiSQLScriptErrors コレクション を利用して ISWiSQLScriptError オブジェクトを読み出すことができます。

メンバー

テーブル 119 ・ ISWiSQLScriptError オブジェクトのメンバー

名前	種類	説明
ErrorHandling	読み書きプロパティ	次の値から 1 つを取得または設定します。 ・ esehAbort (2) — エラー時に、インストールを中止する。 ・ esehGotoNextScript (0) — エラー時に、次のスクリプトへ移動する。 ・ esehGotoNextStatement (1) — エラー時に、次のステートメントへ移動する。
ErrorNumber	読み書きプロパティ	エラー処理をカスタマイズする SQL エラー番号を示す数値を取得または設定します。

ISWiStringEntry オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiStringEntry オブジェクトあ、特定言語の文字列エントリを表します。

メンバー

テーブル 120 • ISWiStringEntry オブジェクトのメンバー

名前	種類	説明
Id	読み書きプロパティ	文字列の言語非依存 ID を取得または設定します。プロジェクトに含まれる各文字列 ID は、1 つまたは複数の値にリンクされています。
Value	読み書きプロパティ	実行時文字列を取得または設定します。
Comment	読み書きプロパティ	文字列エントリについての内部メモを取得または設定します。

次に適用：

- ISWiLanguage

ISWiUpgradeTableEntry オブジェクト

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiUpgradeTableEntry は、InstallShield の [アップグレード] ビューにあるアップグレード アイテムを表します。ISWiUpgradeTableEntries collection でアイテムを指定して、アップグレード アイテムを読み出すことができます。

メンバー

テーブル 121 • ISWiUpgradeTableEntry オブジェクトのメンバー

名前	種類	説明
Delete	メソッド	プロジェクトファイルから StaticUpgradeEntry を削除します。
DisplayName	読み書きプロパティ	アップグレード エントリの名前を取得または設定します。これは [アップグレード] ビューのアップグレード アイテムに表示される内部名です。
UpgradeCode	読み書きプロパティ	アップグレード テーブル エントリの Upgrade Code を取得または設定します。アップグレード コードは関連のある一連の製品を象徴する GUID です。Upgrade テーブルで既にインストール済みの製品に関連するバージョンの検索に利用します。
VersionMin	読み書きプロパティ	FindRelatedProducts が検出する、製品の最小バージョンを指定します。
VersionMax	読み書きプロパティ	FindRelatedProducts が検出する、製品の最大バージョンを指定します。
言語	読み書きプロパティ	FindRelatedProducts が検出する言語セットを指定します。コンマで区切られた数値言語識別子 (LANGID) のリストを入力します。
Attributes	読み書きプロパティ	Upgrade テーブルの属性を指定するビットフラグを含みます。
削除	読み書きプロパティ	このプロパティに入力するフォーマット済み文字列は、コンマ区切りの機能名リストとして評価される必要があります。
ActionProperty	読み書きプロパティ	FindRelatedProducts アクションがシステム上に関連製品を検出すると、このフィールドで指定されたプロパティにプロパティコードを追加します。

次に適用：

- ISWiProject

Delete メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*

- *InstallScript MSI*

Delete メソッドを呼び出すと、プロジェクトファイルから StaticUpgradeEntry が削除されます。

構文

Delete()

次に適用：

- ISWiUpgradeTableEntry

オートメーション コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

各 InstallShield オートメーション コレクションは、1 セットのオブジェクトです。たとえば、ISWiFeature コレクションは、現在のプロジェクトの 1 セットの ISWiFeature オブジェクトです。このセクションでは、各 InstallShield オートメーション コレクションが説明されています。

ISWiAdvancedFiles コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ISWiSAdvancedFiles は、プロジェクトのあらゆるアドバンス ファイルを (ISWiAdvancedFile オブジェクトとして) 含むコレクションです。たとえば次の Visual Basic コードは、特定のアドバンスファイルのプロパティの設定方法を示したものです。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"

pProj.ISWiAdvancedFiles(1).Disk = edtOther

pProj.SaveProject
pProj.CloseProject
```

メンバー

テーブル 122 · ISWiAdvancedFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiAdvancedFiles コレクション内のアドバンスファイルの合計数を返す場合に使用します。
項目	読み取り専用プロパティ	アドバンス ファイルのインデックス番号または名前を指定して、ISWiAdvancedFile オブジェクトを読み出します。たとえば次のステートメントはコレクションに最初のアドバンスファイルへの参照と、CSSaveSpaceというアドバンスファイルを作成します。 Set pSetupFile1 = pProj.ISWiAdvancedFiles.Item(1) Set pSetupFile2 = pProj.ISWiAdvancedFiles.Item("CSSaveSpace") Item は ISWiAdvancedFiles のデフォルトプロパティです。つまり pProj.ISWiAdvancedFiles.Item("CSSaveSpace") は pProj.ISWiAdvancedFiles("CSSaveSpace") と同じです。

次に適用：

- ISWiProject

ISWiAutomaticUpgradeEntries コレクション

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiAutomaticUpgradeEntries は、IDE で定義された通りに ISWiAutomaticUpgradeEntry 項目のコレクションを戻します。

メンバー

テーブル 123 • ISWiAutomaticUpgradeEntries コレクションのメンバー

プロパティ	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiAutomaticUpgradeEntries コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	ISWiAutomaticUpgradeEntries のアイテムを読み出すために、インデックス番号またはアップグレードコードを提供します。 Item は ISWiAutomaticUpgradeEntries のデフォルトプロパティです。つまり、m_ISWiProject.ISWiAutomaticUpgradeEntries.Item("12345678-1234-1234-1234-123456789012") は m_ISWiProject.ISWiAutomaticUpgradeEntries("12345678-1234-1234-1234-123456789012") と同じです。

例

```
Dim m_ISWiProj As ISWiProject
Dim m_AutomaticUpgradeEntries As ISWiAutomaticUpgradeEntries

Set m_ISWiProj = New ISWiProject
m_ISWiProj.OpenProject "C:¥mysetups¥build 34.ism"

For Each m_AutomaticUpgradeEntries In m_ISWiProject.ISWiAutomaticUpgradeEntries
    m_AutomaticUpgradeEntries.Name = "Msi Item1 のアップグレード"
Next

m_ISWiProj.SaveProject
m_ISWiProj.CloseProject
```

次の行は、特定の自動アップグレードエントリへのアクセス方法を示します。

```
Set m_AutomaticUpgradeEntries = ISWiProject.ISWiAutomaticUpgradeEntries ("Upgrade Msi Item1")
```

次に適用 :

- ISWiProject

ISWiChainPackageDatas コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

現在のプロジェクト内のすべての連鎖 .msi パッケージに関連するデータのコレクションを含みます。

メンバー

テーブル 124 • ISWiChainPackageDatas コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiChainPackageDatas コレクションの要素の合計数を戻します。
項目	読み取り専用プロパティ	連鎖 .msi パッケージのインデックス番号または名前を指定して、ISWiChainPackageData オブジェクトを読み出します。 連鎖 .msi パッケージの名前は大文字と小文字を区別します。つまり、“Chain Package One” と “Chain Package one” は、2 つの異なる構成になります。 Item は ISWiChainPackageDatas コレクションのデフォルトプロパティです。つまり、pProject.ISWiChainPackageDatas.Item(″PackageData1″) は equivalent to pProject.ISWiChainPackageDatas (″PackageData1″) に相当します。

例

下の VBScript 例は、プロジェクト内の特定の連鎖 .msi パッケージに関連するデータを読み出す方法を示します:

```
Set pProject = CreateObject(″IswiAutoAutomation Interface Version.ISWiProject″)
    pProject.OpenProject ″C:¥MySetups¥MyProject.ism″, True ' open read-only

    sNames = ″このプロジェクトは次を含みます:″ & vbNewLine
    iCount = pProject.ISWiChainPackages.Count
    For i = 1 to iCount
        Set pChainPackage = pProject.ISWiChainPackages.Item(i)
        sNames = sNames & vbTab & pChainPackage.DisplayName & vbNewLine
        sNames = sNames & vbTab & ″この連鎖パッケージ プロジェクトは次のファイルとフォルダーを含みます:″ & vbNewLine
        iCount2 = pChainPackage.ISWiChainPackageDatas.Count
        For j = 1 to iCount2
            Set pPackageData = pChainPackage.ISWiChainPackageDatas.Item(j)
            sNames = sNames & vbTab & pPackageData.ISBuildSourcePath & vbNewLine
        次へ
    次へ

     MsgBox sNames
```

次に適用 :

- ISWiProject

ISWiChainPackages コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

現在のプロジェクト内の連鎖 .msi パッケージのコレクションを含みます。

メンバー

テーブル 125 • ISWiChainPackages コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiChainPackages コレクションの要素の合計数を戻します。
項目	読み取り専用プロパティ	連鎖 .msi パッケージのインデックス番号またはファミリー名を指定して、ISWiChainPackage オブジェクトを読み出します。

例

下の VBScript 例は、プロジェクト内の連鎖 .msi パッケージ一覧を読み出す方法を示します:

```
Set pProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
Project.OpenProject "C:¥MySetups¥MyProject.ism", True ' open read-only

sNames = "このプロジェクトは次を含みます:" & vbNewLine
iCount = pProject.ISWiChainPackages.Count
For i = 1 to iCount
    Set sMy = pProject.ISWiChainPackages.Item(i)
    sNames = sNames & vbTab & sMy.DisplayName & vbNewLine
次へ

MsgBox sNames

Set pChainPackage = pProject.ISWiChainPackages.Item("Package 1")
```

次に適用 :

- ISWiProject

ISWiComponents コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiComponents コレクションには、指定された機能に関連付けられているすべてのコンポーネントが含まれます。ISWiComponents は、機能に含まれるすべてのコンポーネントを列挙するインターフェイスを備えています。ISWiComponents が ISWiProject のメンバーである場合、指定した機能に属するコンポーネントだけでなく、プロジェクト内のすべてのコンポーネントが含まれます。

メンバー

テーブル 126 • ISWiComponents コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiComponents コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	コンポーネントのインデックス番号または名前を指定して、ISWiComponent オブジェクトを読み出します。 コンポーネントの名前は大文字と小文字を区別します。たとえば、“Help_Files” と “Help_files” は異なるコンポーネントとして認識されます。 Item は ISWiComponents のデフォルトプロパティです。つまり、m_ISWiFeature.ISWiComponents.Item(″Help_Files″) は m_ISWiFeature.ISWiComponents(″Help_Files″) と同じです。

例

次の Visual Basic コードは、プロジェクトを開き、NewFeature1 という機能の各コンポーネントをループして、各コンポーネントの RemoteInstallation プロパティをソースから実行するように設定してから、最後にプロジェクトを保存して閉じます。

```
Dim m_ISWiProj As ISWiProject
Dim m_Feature As ISWiFeature
Dim m_Comp As ISWiComponent

Set m_ISWiProj = CreateObject(″IswiAutoAutomation Interface Version.ISWiProject″)
m_ISWiProj.OpenProject ″C:¥mysetups¥build 34.ism″

Set m_Feature = m_ISWiProj.ISWiFeatures(″NewFeature1″)

If Not m_Feature Is Nothing Then
  For Each m_Comp In m_Feature.ISWiComponents
      m_Comp.RemoteInstallation = rfsSource
  Next
EndIf

m_ISWiProj.SaveProject
m_ISWiProj.CloseProject
```

次の行は、特定のコンポーネントへのアクセス方法を示します。

```
Set m_Comp = m_ISWiProj.ISWiFeatures("MyFeature").ISWiComponents("MyComponent")
```

次に適用：

- ISWiFeature
- ISWiProject

ISWiComponentSubFolders コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

ISWiComponentSubFolders は、現在のコンポーネントまたはコンポーネント サブフォルダーにすべてのコンポーネント サブフォルダーを (ISWiComponentSubFolder オブジェクトとして) 含むコレクションです。たとえば次の Visual Basic コードは、特定のコンポーネントサブフォルダーにメソッドを適用する方法を示したものです。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pComponent As ISWiComponent

Set pFeature = pProj.ISWiFeatures.Item("MyFeature")
Set pComponent = pFeature.ISWiComponents.Item("MyComp")
pComponent.ISWiComponentSubFolders.Item("MyCompSubFolder").AddFile("C:¥My Files¥MyFile.ext")

pProj.SaveProject
pProj.CloseProject
```

メンバー

テーブル 127 • ISWiComponentSubFolders コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiComponentSubFolders コレクション内のコンポーネント サブフォルダーの合計数を返す場合に使用します。
項目	読み取り専用プロパティ	コンポーネント サブフォルダーのインデックス番号または名前を指定して、ISWiComponentSubFolder オブジェクトを読み出します。たとえば次のステートメントはコレクションに最初のコンポーネント サブフォルダーへの参照と、MyCompSubFolderというコンポーネント サブフォルダーを作成します。 Set pComponentSubFolder1 = pProj.ISWiComponentSubFolders.Item(1) Set pComponentSubFolder2 = pFeature.ISWiComponentSubFolders.Item("MyCompSubFolder") Item は ISWiComponentSubFolders のデフォルトプロパティです。つまり、pFeature.ISWiComponentSubFolders.Item("MyCompSubFolder") は pFeature.ISWiComponentSubFolders("MyCompSubFolder") と同じです。

次に適用：

- ISWiComponent
- ISWiComponentSubFolder

ISWiConditions コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiConditions コレクションには、指定された機能に関連付けられているすべての条件が含まれます。ISWiConditions は、機能に含まれるすべての条件を列挙するインターフェイスを備えています。

メンバー

テーブル 128 • ISWiConditions コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiConditions コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	条件のインデックス番号を指定して、ISWiCondition オブジェクトを取得します。 Item は ISWiConditions のデフォルトプロパティです。つまり m_ISWiFeature.ISWiConditions.Item(2) は m_ISWiFeature.ISWiConditions(2) と同じです。

次に適用：

- ISWiFeature

ISWiCustomActions コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiCustomActions コレクションには、現在のプロジェクトのすべてのカスタム アクションが含まれています。

以下の VBScript の例では、プロジェクトにおける各カスタム アクションの文字列を表示します。

```
Set pProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProject.OpenProject "C:¥MySetups¥MyProject.ism", True ' open read-only

sNames = "このプロジェクトは次のカスタム アクションを含みます:" & vbNewLine

iCount = pProject.ISWiCustomActions.Count
For i = 1 to iCount
    sNames = sNames & vbTab & pProject.ISWiCustomActions(i).Name & vbNewLine
Next

pProject.CloseProject

MsgBox sNames
```

メンバー

テーブル 129 • ISWiCustomActions コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	ISWiCustomActions コレクション中の要素数を返します。
項目	読み取り専用プロパティ	カスタム アクションのインデックス番号または名前を指定して、ISWiCustomAction オブジェクトを読み出します。 カスタム アクション名は大文字と小文字を区別します。したがって、“MyAction” と “myAction” は 2 つの異なるアクションと認識されます。 Item は ISWiCustomActions のデフォルトプロパティです。つまり、pProject.ISWiCustomActions.Item(″Action1″) は pProject.ISWiCustomActions(″Action1″) と同じです。

次に適用 :

- ISWiProject

ISWiDynamicFileLinkings コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

プロジェクト固有の違いについては、必要に応じて記述されています。

ISWiDynamicFileLinkings コレクションは、指定されたコンポーネントに関連付けられた、すべてのダイナミック ファイル リンクを含みます。

メンバー

テーブル 130 • ISWiDynamicFileLinkings コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiDynamicFileLinkings コレクション内のダイナミック ファイル リンクの総数を返すのに使用します。
項目	読み取り専用プロパティ	動的にリンクされたフォルダーのインデックス番号または名前を指定して、ISWiDynamicFileLinking オブジェクトを読み出します。

次に適用 :

- ISWiComponent

ISWiEnvironmentVars コレクション

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiEnvironmentVars コレクションは、指定されたコンポーネントに関連付けられたすべての環境変数を含みます。

メンバー

テーブル 131 • ISWiEnvironmentVars コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiEnvironmentVars コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	環境変数のインデックス番号または名前を指定して、ISWiEnvironmentVar オブジェクトを取得します。

次に適用 :

- ISWiComponent

ISWiFeatures コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*

ISWiFeatures は、プロジェクトのあらゆる機能を (ISWiFeature オブジェクトとして) 含むコレクションです。

ISWiFeatures が ISWiProject のメンバーの場合、これにはプロジェクトのトップレベルの機能がすべて含まれます。サブ機能のコレクションを読み出すには、ISWiFeature オブジェクトの [機能] プロパティにアクセスします。その結果、現在のトップレベル機能のすぐ下にあるサブ機能を含む ISWiFeatures コレクションが表示されます。これを繰り返すことで、さらに下のレベルのサブ機能が表示されます。

メンバー

テーブル 132 • ISWiFeatures コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWIFeatures コレクション内の要素の合計数を返す場合に使用します。

項目	読み取り専用プロパティ	機能のインデックス番号または名前を指定して、ISWiFeature オブジェクトを読み出します。たとえば、次のステートメントでは、コレクションの最初の項目のコピーおよび Help_Files という名前の機能のコピーが作成されます。
		`Set pFeat1 = m_ISWiProject.ISWiFeatures.Item(1)`
		`Set pFeat2 = m_ISWiProject.ISWiFeatures.Item("Help_Files")`
		機能の名前は大文字と小文字を区別します。たとえば、“Help_Files” と “Help_files” は異なる機能として認識されます。
		Item は ISWiFeatures のデフォルトプロパティです。つまり、m_ISWiProject.ISWiFeatures.Item("Help_Files") は m_ISWiProject.ISWiFeatures("Help_Files") と同じです。

例

次の Visual Basic コードでは、プロジェクト内のすべての機能およびコンポーネントを使用して文字列をビルドします:

```
Dim pFeature As ISWiFeature
Dim pComponent As ISWiComponent

For Each pFeature In m_ISWiProject.ISWiFeatures
    ' 文字列に機能を追加する
    strProject = strProject & "機能: " & pFeature.Name & vbNewLine
    strProject = strProject & "コンポーネント: "
    For Each pComponent In pFeature.ISWiComponents
        ' 文字列を機能のコンポーネントに追加する
        strProject = strProject & pComponent.Name & " "
    Next
```

```
        strProject = strProject & vbNewLine
    Next
```

次に適用：

- ISWiProject

ISWiFiles コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiFiles コレクションには、指定のコンポーネントに関連したすべてのファイル、または InstallScript プロジェクトの場合はコンポーネントのサブフォルダーが含まれます。ISWiFiles には複数の読み取り – 書き込みのプロパティが含まれており、これらを使用して、IDE と同様の方法でコンポーネントのプロパティとファイルの属性を設定できます。

次の Visual Basic コードは、コンポーネントの特定のファイルにプロパティを設定する方法の例です。この例では、変数 m_oMyComp によって既にコンポーネントが参照されているものとします。

```
m_oMYComp.ISWiFiles("F1028_MyFile.dll").Hidden = True
```

メンバー

テーブル 133 ・ ISWiFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiFiles コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	ファイルのインデックス番号またはファイル キーを指定して、ISWiFile オブジェクトを取得します。ファイル名は使用できません。ファイルキーを指定する必要があります。この値は、[ダイレクト エディタ] ビュー (File テーブルの File 列) または、Windows Installer ベースのプロジェクトでは、コンポーネントのファイル リストの Key 列に表示されます。ダイナミック リンク ファイルの場合、ファイル キーが変更の対象である点にご注意ください。 Item は ISWiComponents のデフォルトプロパティです。つまり、m_ISWiComponent.ISWiFiles.Item("F1036_MyFile.exe") は m_ISWiComponent.ISWiFiles("F1036_MyFile.exe") と同じです。

次に適用 :

- ISWiComponent

ISWiFolders コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiFolders には、指定したコンポーネントに属するプログラムフォルダーが格納されます。

各コンポーネントは、最低でも下記に一覧したデフォルトフォルダーを ISWiFolders コレクション内に持ちます。これらに加えて、[ショートカット] を右クリックして [フォルダーの表示] を選択することで、定義済みフォルダーを [ショートカット] エクスプローラーに追加している場合もあります。

- [TaskBarFolder]
- [SendToFolder]
- [DesktopFolder]

次に、タスクバーを表す ISWiFolder オブジェクトにアクセスするコードを示します。

```
Dim m_pFolder As ISWiFolder
Set m_pFolder = m_pComponent.ISWiFolders("[TaskBarFolder]")
```

タスクバーには独自のサブフォルダーがあります。これらのサブフォルダーは SubFolders プロパティに格納されますが、このプロパティ自体が ISWiFolders コレクションです。先の例に続けて次の行を記述すると、[スタート] メニューのオブジェクトのコピーを作成できます。これは、[ショートカット] エクスプローラーではタスクバーのサブフォルダーになります。

```
Set m_pFolder = m_pFolder.SubFolders("[StartMenuFolder]")
```

次の 1 行を追加してこの例をさらに発展させると、[プログラム] メニューの下のフォルダー [MyProgFolder] のコピーを取得できます。

```
Set m_pFolder = m_pFolder.SubFolders("[ProgramMenuFolder]").SubFolders("MyProgFolder")
```

メンバー

テーブル 134 ・ ISWiFolders コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiFolders コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	フォルダーのインデックス番号または名前を指定して、ISWiFolder オブジェクトを読み出します。 Item は ISWiFolders のデフォルトプロパティです。つまり、m_ISWiComponent.ISWiFolders.Item("[DesktopFolder]") は m_ISWiComponent.ISWiFolders("[DesktopFolder]") と同じです。

次に適用：

- ISWiComponent

ISWiLanguages コレクション

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiLanguages コレクションには、現在のプロジェクトにおけるすべての言語が含まれています。

メンバー

テーブル 135 • ISWiLanguages コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiLanguages コレクションの言語の合計数を返します。
項目	読み取り専用プロパティ	言語のインデックス番号または名前を指定して、ISWiLanguage オブジェクト を読み出します。

次に適用：

- ISWiProject

ISWiLatestSetups コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

現在のパッチ構成に属するすべての最新セットアップ (アップグレードされたイメージ) を含みます

テーブル 136 • ISWiLatestSetups コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiLatestSetups コレクション の要素の合計数を戻します。
項目	読み取り専用プロパティ	ISWiLatestSetup オブジェクトを読み出すため、パッチ シーケンスのインデックス番号またはファミリー名を提供します。 アップグレードされたイメージの名前は大文字と小文字を区別します。したがって、“Latest1” と “latest1” は 2 つの異なるアップグレードされたイメージ構成と認識されます。 Item は ISWiLatestSetups のデフォルト プロパティで、pPatchConfig を意味します。ISWiLatestSetups.Item(″Latest1″) is equivalent to pPatchConfig.ISWiLatestSetups (″Latest1″).

次に適用：

- ISWiPatchConfig

AddMsiPatchSequence メソッド

プロジェクト：この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

AddMsiPatchSequence メソッドを呼び出して、現在のパッチ構成に ISWiMsiPatchSequence オブジェクトを追加します。

構文

AddMsiPatchSequence (FamilyName As String) As ISWiMsiPatchSequence

パラメーター

テーブル 137 • AddMsiPatchSequence メソッドのパラメーター

パラメーター	説明
FamilyName	パッチ シーケンスのファミリー名を保持します。

次に適用：

- ISWiPatchConfig

DeleteMsiPatchSequence メソッド

プロジェクト：この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

DeleteMsiPatchSequence メソッドを呼び出し、現在のパッチ構成から指定されたパッチ シーケンスを削除します。

構文

DeleteMsiPatchSequence (MsiPatchSequence As ISWiMsiPatchSequence) As Long

パラメーター

テーブル 138 • DeleteMsiPatchSequence メソッドのパラメーター

パラメーター	説明
MsiPatchSequence	削除するパッチ シーケンス オブジェクトを保持します。

次に適用：

- ISWiPatchConfig

ISWiMsiPatchSequences コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

現在のパッチ構成に属するすべてのパッチ シーケンスを含みます。

テーブル 139 • ISWiMsiPatchSequences コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiMsiPatchSequences コレクションの要素の合計数を戻します。
項目	読み取り専用プロパティ	ISWiMsiPatchSequence オブジェクトを読み出すため、パッチ シーケンスのインデックス番号またはファミリー名を提供します。 パッチ シーケンスのファミリー名は大文字と小文字を区別します。したがって、“FamilyName” と “familyName” は 2 つの異なる構成と認識されます。 Item は SWiMsiPatchSequences のデフォルト プロパティで、pPatchConfig を意味します。 ISWiMsiPatchSequences.Item(″FamilyName ″) is equivalent to pPatchConfig. ISWiMsiPatchSequences (″FamilyName″).

次に適用：

- ISWiProject

BuildPatch メソッド

プロジェクト: IswiPatchConfig *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

BuildPatch を呼び出して、現在のパッチ オブジェクトによって識別されたパッチ設定をビルドします。このアクションは、[パッチのデザイン] ビューでパッチ構成を右クリックしてから 'パッチのビルド' を選択した場合と同じです。

構文

BuildPatch()

パラメーター

テーブル 140 • BuildPatchConfiguration メソッドに関連づけられたプロパティ

プロパティ	説明
PatchConfigErrorCount	特定のパッチビルドに関連するエラー カウントを含みます。
PatchConfigWarningCount	特定のパッチビルドに関連する警告カウントを含みます。

例

次の Visual Basic 行では、このメソッドを示します。

```
'プロジェクトオブジェクトの作成
dim pProject
set pProject = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
'OpenProject メソッドの呼び出し
'########################################################################
'# このサンプルは有効なパスなしには適切に実行しません
'########################################################################
'pProject.OpenProject("¥Your project.ism")
pProject.OpenProject("C:¥Documents and Settings¥Testlab¥My Documents¥MySetups¥your project name-21.ism")
'OpenProject メソッドの呼び出し
' Get existing configuration
Set oMyConfig = pProject.ISWiProductConfigs.Item("製品構成 1")
' プロジェクトに、既にリリース構成が含まれることを前提とします…
Set oNewRelease = oMyConfig.ISWiReleases.Item("リリース 1")
' パッチをビルドする前に、リリース設定が圧縮解除されていることを確認します…
oNewRelease.Compressed = False
oNewRelease.SetupEXE = True
' パラメーターの署名作成および設定...
Set oMyPatchConfig = pProject.AddPatchConfig("PatchDemo1")
oMyPatchConfig.SignPackage = True
oMyPatchConfig.SignUpdateExe = True
oMyPatchConfig.SignatureSPC = "C:¥Users¥TestLab¥Desktop¥Test.pfx"
oMyPatchConfig.SignaturePassword = "XX"
Set oMyLatestImage = oMyPatchConfig.AddUpgradedImage("LatestImg1")
Set oMyPreviousImage = oMyLatestImage.AddTargetImage("PrevImage1")
' ベース イメージを設定…
oMyPreviousImage.MsiPath = "<ISProjectFolder>BaseImage¥PatchDesignInterfaceDemo.msi"
' 既存のパッチ構成に、ベース MSI パスを設定します。
'MyPatchConfig.ISWiUpgradedImages.Item("LatestImg1").ISWiTargetImages.Item("PrevImage1").MsiPath =
"<ISProjectFolder>BaseImage¥PatchDesignInterfaceDemo.msi"
pProject.SaveProject
oNewRelease.Build( )
oMyPatchConfig.BuildPatch
MsgBox "Patch Congifuration Config1 build with " & _
???? oMyPatchConfig.PatchConfigErrorCount & " エラーおよび " & _
???? oMyPatchConfig.PatchConfigWarningCount & " 警告"
```

```
pProject.CloseProject
```

ISWiObjects コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

ISWiObjects は、プロジェクトのあらゆる InstallScript オブジェクトを (ISWiObject オブジェクトとして) 含むコレクションです。次の Visual Basic コードはこのコレクションの使用方法を示しています。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"
Dim pFeature As ISWiFeature
Dim pObject As ISWiObject

Set pFeature = pProj.ISWiFeatures.Item(2)
Set pObject = pFeature.ISWiObjects.Item("@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9")
pFeature.RemoveObject pObject

pProj.SaveProject
pProj.CloseProject
```

メンバー

テーブル 141 • ISWiObjects コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWIObjects コレクション内の InstallScript オブジェクトの合計数を返す場合に使用します。
項目	読み取り専用プロパティ	InstallScript オブジェクトのインデックス番号またはモニカを指定して、ISWiObject オブジェクトを読み出します。(InstallScript オブジェクトのモニカを取得するには、IDE で新しい Professional プロジェクトを作成し、InstallScript オブジェクトを機能に追加して、[ダイレクトエディター] ビューの ISFeatureExtended テーブルに移動し、Moniker 列を参照します。)たとえば次のステートメントでは最初の InstallScript オブジェクトへの参照をコレクションに作成します。InstallScript オブジェクトは、@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9: というモニカを持っています。 Set pObject1 = pProj.ISWiObjects.Item(1) Set pObject2 = pFeature.ISWiObjects.Item("@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9") Item はISWiObjects のデフォルトプロパティです。これはpFeature.ISWiObjects.Item("@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9") が pFeature.ISWiObjects("@ismk2:755142B0-1EB4-11D3-8B09-00105A9846E9")と同じだということです。

次に適用：

- ISWiFeature

ISWiPatchConfigs コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

メンバー

テーブル 142 • ISWiPatchConfigs コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiPatchConfigs コレクション の要素の合計数を戻します。
項目	読み取り専用プロパティ	パッチ構成のインデックス番号または名前を指定して、ISWiPatchConfig オブジェクトを読み出します。 パッチ構成の名前は大文字と小文字を区別します。つまり、“Patch One” と “Patch one” は、2 つの異なる構成になります。 Item は ISWiPatchConfigs のデフォルトプロパティです。つまり、pProject.ISWiPatchConfigs.Item(″Patch1″) は pProject.ISWiPatchConfigs (″Patch1″) に相当します。

例

以下の VBScript の例では、プロジェクトにおける各パッチ構成の文字列を表示します:

```
Dim pProject
Set pProject = CreateObject (″IswiAutoAutomation Interface Version.ISWiProject″)
pProject.OpenProject ″C:¥MySetups¥Vegetarian.ism″, True

Dim sNames
sNames = ″この製品は、次のパッチ構成を含みます: ″ & vbNewLine & vbTab

Dim pPatchConfigs
For Each pPatchConfigs In pProject.ISWiPatchConfigs
        sNames = sNames & pPatchConfigs.Name & vbNewLine & vbTab
Next

pProject.CloseProject

WScript.Echo sNames
オブジェクト
```

次の行に、特定のパッチ構成へのアクセス方法を示します。

```
Set pPatchConfig = pISWiProj.ISWiPatchConfigs.Item(″Patch 1.1″)
```

次に適用 :

- ISWiProject

ISWiPatchExternalFiles コレクション

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

最新セットアップに所属する外部ファイル オブジェクトのコレクションを含みます。

メンバー

テーブル 143 • ISWiPatchWholeFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiPatchExternalFiles コレクションの要素の合計数を戻します。
項目	読み取り専用プロパティ	インデックス番号または FileKey を指定して、ISWiPatchExternalFiles オブジェクトを取得します。

次に適用：

- ISWiLatestSetup

AddPatchWholeFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddPatchWholeFile メソッドを呼び出して、最新セットアップ項目に ISWiPatchWholeFile オブジェクトを追加します。

構文

AddPatchWholeFile (FileKey As String) As ISWiPatchWholeFile

パラメーター

テーブル 144 • AddPatchWholeFile メソッド パラメーター

パラメーター	説明
FileKey	完全なファイル オブジェクトのファイル キーを保持します。

次に適用：

- ISWiLatestSetup

DeletePatchWholeFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

eletePatchWholeFile メソッドを呼び出して、最新セットアップ項目から ISWiPatchWholeFiles オブジェクトを削除します。

構文

DeletePatchWholeFile (PatchWholeFile As ISWiPatchWholeFile) As Long

パラメーター

テーブル 145 : DeletePatchWholeFile メソッド パラメーター

パラメーター	説明
PatchWholeFile	削除する完全なファイル オブジェクトを保持します。

次に適用 :

- ISWiLatestSetup

ISWiPatchWholeFiles コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

最新セットアップに所属する完全なファイル オブジェクトのコレクションを含みます。

メンバー

テーブル 146 • ISWiPatchWholeFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiPatchWholeFiles コレクション の要素の合計数を戻します。
項目	読み取り専用プロパティ	インデックス番号または FileKey を指定して、ISWiPatchWholeFile オブジェクトを取得します。

次に適用：

- ISWiProject

ISWiPathVariables コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiPathVariables は、現在のプロジェクトに関連付けられているすべてのパス変数を含みます。AddPathVariable メソッドを使ってパス変数を追加し、DeletePathVariable メソッドで削除することができます。

以下の VBScript コードにより、プロジェクトにある各パス変数のリストを取得し、メッセージボックスに表示します。

```
Option Explicit
  Dim pProject
  Set pProject = CreateObject ("ISWiAutoAutomation Interface Version.ISWiProject")

  pProject.OpenProject "C:¥MySetups¥ISWiPathVariables.ism", True

  Dim sPathVars
  Dim pPathVar

  For Each pPathVar In pProject.ISWiPathVariables
     sPathVars = sPathVars & pPathVar.Name & ": " & pPathVar.Value & vbNewLine
  Next

  WScript.Echo sPathVars
  pProject.CloseProject
```

次の行に、特定のパス変数へのアクセス方法を示します。

```
Set pPathVar = pProject.ISWiPathVariables.Item("MyPathVar")
```

重要: *定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

メンバー

テーブル 147 : ISWiPathVariables コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiPathVariables コレクションのパス変数の合計数を返します。
項目	読み取り専用プロパティ	パス変数のインデックス番号または名前を指定して、ISWiPathVariable オブジェクトを取得します。

次に適用 :

- ISWiProject

ISWiPreviousSetups コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

最新セットアップに所属するすべての以前のセットアップのコレクションを含みます。

パッチを作成するためには、各パッチ構成に少なくとも最新セットアップが 1 つ、および以前のセットアップが 1 つ含まれている必要があります。最新のセットアップごとに以前のバージョンを複数指定することもできます。

最新のセットアップにパッチをあてる予定のアプリケーションまたはセットアップはすべて、以前のセットアップ ノードに追加する必要があります。

たとえば、アプリケーションのバージョン 1.0 および 1.1 をサポートする場合で、両方のアプリケーションをバージョン 2.0 にアップグレードすると想定します。両方のバージョンをアップグレードする単一のパッチを作成するには、最新のセットアップに以前のセットアップとして V1.0 と V1.1 を追加します。最新セットアップには V2.0 が入ります。

メンバー

テーブル 148 • ISWiPreviousSetups コレクション メンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiPreviousSetups コレクション の要素の合計数を戻します。
項目	読み取り専用プロパティ	ISWiPreviousSetup オブジェクトを読み出すため、以前のセットアップのインデックス番号または名前を提供します。 以前のセットアップ名は大文字と小文字を区別します。したがって、“Prev1” と “prev1” は 2 つの異なる構成と認識されます。 Item は ISWiPreviousSetups のデフォルト プロパティで、pLatestSetup を意味します。ISWiPreviousSetups.Item(″Prev1″) is equivalent to pLatestSetup.ISWiPreviousSetups(″Prev1″).

次に適用：

- ISWiProject

AddPatchExternalFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

AddPatchExternalFile メソッドを呼び出して、最新セットアップ項目に ISWiPatchExternalFile を追加します。

構文

AddPatchExternalFile (Name As String) As ISWiPatchExternalFile

パラメーター

テーブル 149 • AddPatchExternalFile メソッドのパラメーター

パラメーター	説明
名前	外部ファイルの名前を保持します。

次に適用：

- ISWiLatestSetup

DeletePatchExternalFile メソッド

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

DeletePatchExternalFile メソッドを呼び出して、最新セットアップ項目から ISWiPatchExternalFile オブジェクトを削除ます。

構文

DeletePatchExternalFile (PatchExternalFile As ISWiPatchExternalFile) As Long

パラメーター

テーブル 150 · DeletePatchExternalFile メソッド パラメーター

パラメーター	説明
PatchExternalFile	削除する外部 ファイルオブジェクトを保持します。

次に適用 :

- ISWiLatestSetup

ISWiProductConfigs コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

InstallScript プロジェクトおよび InstallScript オブジェクト プロジェクトには、ISWiProductConfigs に Media という要素 1 つだけが含まれています。

ISWiProductConfigs コレクションには、現在のプロジェクトにおけるすべての製品構成が含まれています。

メンバー

テーブル 151 • ISWiProductConfigs コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiProductConfigs コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiProductConfig オブジェクトを読み出します。 製品構成の名前は大文字と小文字を区別します。つまり、“Product One” と “Product one” は、2 つの異なる設定になります。 Item は ISWiProductConfigs のデフォルトプロパティです。つまり pProject.ISWiProductConfigs.Item(″Version 1″) は pProject.ISWiProductConfigs(″Version 1″) と同じです。

例

以下の VBScript の例では、プロジェクトにおける各製品構成の文字列を表示します。

```
Dim pProject
Set pProject = CreateObject (″IswiAutoAutomation Interface Version.ISWiProject″)
pProject.OpenProject ″C:¥MySetups¥Vegetarian.ism″, True

Dim sNames
sNames = ″この製品は、次の製品構成を含みます: ″ & vbNewLine & vbTab

Dim pProductConfigs
For Each pProductConfigs In pProject.ISWiProductConfigs
    sNames = sNames & pProductConfigs.Name & vbNewLine & vbTab
Next

pProject.CloseProject

WScript.Echo sNames
```

次の行に、特定の製品構成へのアクセス方法を示します。

```
Set pProdConfig = pISWiProj.ISWiProductConfigs.Item(″Version 1.1″)
```

次に適用：

- ISWiProject

ISWiProperties コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiProperties コレクションには、現在のプロジェクトにおける Windows Installer のすべてのプロパティが含まれています。このインターフェイスはプロパティマネージャーに似ています。AddProperty メソッドを使ってプロパティを追加し、DeleteProperty メソッドで削除することができます。

以下の VBScript コードにより、プロジェクトにある各プロパティのリストを取得し、メッセージボックスに表示します。

```
Option Explicit

Dim pProject
Set pProject = CreateObject ("ISWiAutoAutomation Interface Version.ISWiProject")

pProject.OpenProject "C:¥MySetups¥ISWiProperties.ism", True

Dim sProps
Dim pProp

For Each pProp In pProject.ISWiProperties
    sProps = sProps & pProp.Name & ": " & pProp.Value & "  // " & _
            pProp.Comments & vbNewLine
Next

WScript.Echo sProps

pProject.CloseProject
```

メンバー

テーブル 152 ・ ISWiProperties コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiProperties コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSProperty オブジェクトを読み出します。 プロパティの名前は大文字と小文字を区別します。したがって、“UpgradeCode” と “Upgradecode” は 2 つの異なるプロパティと認識されます。 Item は ISWiProperties のデフォルトプロパティです。つまり、pISWiProject.ISWiProperties.Item(″UpgradeCode″) は pISWiProject.ISWiProperties(″UpgradeCode″) と同じです。

次に適用 :

- ISWiProject

ISWiReleases コレクション

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiRereleases コレクションには、現在の製品構成に属するすべてのリリースが含まれています。

次の行で、特定の ISWiRelease オブジェクトへのアクセス方法を示します。

```
Set pRelease = pProject.ISWiProductConfigs(″Version 1″).ISWiReleases(″NewRelease1″)
' InstallScript プロジェクトでは、
' ISWiProductConfigs の要素は ″Media″ の 1 つだけです。
```

メンバー

テーブル 153 • ISWiReleases コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiReleases コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	インデックス番号またはリリースの名前を指定して、ISWiRelease オブジェクトを読み出します。 リリースの名前は大文字と小文字を区別します。したがって、“NewRelease1” と “Newrelease1” は 2 つの異なるリリースと認識されます。 Item は ISWiReleases のデフォルトプロパティです。つまり、m_ISWiBldLbl.ISWiReleases.Item(″NewRelease1″) は m_ISWiBldLbl.ISWiReleases(″NewRelease1″) と同じです。

次に適用 :

- ISWiProductConfigs

ISWiRemoveFiles コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiRemoveFiles コレクションは、InstallShield ユーザー インターフェイスの ISWiRemoveFile オブジェクトのコレクションを示します。

メンバー

テーブル 154 • ISWiRemoveFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiRemoveFiles コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	ファイルまたはフォルダー削除エントリのインデックス番号またはキーを指定して、その ISWiRemoveFile オブジェクトを読み出します。キーを指定しなくてはなりません。[ダイレクト エディター] ビューでこの値を確認できます (RemoveFile テーブルの FileKey 列。Item はISWiRemoveFiles のデフォルトプロパティです。つまり、m_ISWiComponent.ISWiRemoveFiles.Item("RemoveFile1") は m_ISWiComponent.ISWiRemoveFiles("RemoveFile1") と同じです)。

次に適用 :

- ISWiComponent

ISWiSequence コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiSequence コレクションは、現在のプロジェクトのすべてのアクションがシーケンス番号別に並べられて含まれている仮想コレクションです。

メンバー

テーブル 155 • ISWiSequence コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiSequence コレクション内の要素の合計数を返す場合に使用します。
InsertCustomAction	メソッド	このメソッドを使用して、カスタム アクションを選択したシーケンスに挿入します。このメソッドは ISWiSequenceRecord オブジェクトを戻します。
項目	読み取り専用プロパティ	アクションのインデックス番号または名前を指定して、ISWiSequenceRecord オブジェクトを読み出します。 アクションの名前は、大文字と小文字を区別します。したがって、“ThisAction” と “thisAction” は異なるアクションとして認識されます。 Item は ISWiSequence のデフォルトプロパティです。つまり、pProject.InstallUISequence.Item(″ThisAction″) は pProject.InstallUISequence(″ThisAction″) と同じです。
RemoveSequenceRecord	メソッド	このメソッドを呼び出して、プロジェクトのシーケンスから項目を削除します。

例

以下の VBScript の例では、プロジェクトのインストール ユーザー インターフェイス シーケンスに、各アクション名とシーケンス番号を持つ文字列を表示します。

```
Dim pProject
Set pProject = CreateObject(″IswiAutoAutomation Interface Version.ISWiProject″)
pProject.OpenProject ″C:¥MySetups¥YourProject.ism″, True ' open read-only

Dim sNames
sNames = ″インストール UI シーケンスは、次のアクションを含みます:″ & vbNewLine & vbTab

For i = 1 to pProject.InstallUISequence.Count
    sNames = sNames & pProject.InstallUISequence.Item(i).Name & ″ ″ & _
            pProject.InstallExecuteSequence.Item(i).Sequence & vbNewLine & vbTab
Next

pProject.CloseProject

MsgBox sNames
```

次に適用：

- ISWiProject

InsertCustomAction メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

InsertCustomAction メソッドを呼び出して、カスタム アクションをプロジェクトのシーケンスに挿入します。

構文

InsertCustomAction (CustomAction As ISWiCustomAction, Comment As String, Condition as String, SequenceNumber as Long) as ISWiSequenceRecord

パラメーター

テーブル 156 · InsertCustomAction メソッド パラメーター

パラメーター	説明
CustomAction	作成するカスタム アクション。
コメント	カスタム アクションに関するオプションの文字列コメント。
条件	カスタム アクションに関するオプションの文字列条件。
SequenceNumber	カスタム アクションの数値シーケンス番号。

戻り値

InsertCustomAction メソッドは ISWiSequenceRecord オブジェクトを返します。

次に適用：

- ISWiSequence

RemoveSequenceRecord メソッド

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *基本の MSI*
- *InstallScript MSI*

RemoveSequenceRecord メソッドを呼び出して、現在のプロジェクトのシーケンスからアイテムを削除します。

構文

(sName As String) As Long

パラメーター

テーブル 157 • RemoveSequenceRecord メソッドのパラメーター

パラメーター	説明
sName	ISWiSequenceRecord オブジェクトを渡して、プロジェクトから削除するシーケンス レコードを指定します。

戻り値

テーブル 158 • RemoveSequenceRecord メソッドの値

戻り値	説明
0	アイテムが、シーケンスより正常に削除されました。
1	アイテムが、シーケンスより正常に削除されました。

次に適用 :

- ISWiSequence

ISWiSetupFiles コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*

ISWiSetupFiles は、プロジェクトのあらゆるサポート ファイルを (ISWiSetupFile オブジェクトとして) 含むコレクションです。

メンバー

テーブル 159 • ISWiSetupFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWISetupFiles コレクション内のサポートファイルの合計数を戻す場合に使用します。
項目	読み取り専用プロパティ	サポート ファイルのインデックス番号または名前を指定して、ISWiSetupFile オブジェクトを読み出します。たとえば次のステートメントはコレクションに最初のサポート ファイルへの参照と、MySupportFile.bmp というサポート ファイルを作成します。 Set pSetupFile1 = pProj.ISWiSetupFiles.Item(1) Set pSetupFile2 = pProj.ISWiSetupFiles.Item("MySupportFile.bmp") Item は ISWiSetupFiles のデフォルトプロパティです。つまり pProj.ISWiSetupFiles.Item("MySupportFile.bmp") は pProj.ISWiSetupFiles("MySupportFile.bmp") と同じです。

例

次の Visual Basic コードは、特定のサポート ファイルのプロパティの設定方法を示したものです。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"

pProj.ISWiSetupFiles(1).Splash = True

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

ISWiSetupTypes コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript MSI*

ISWiSetupTypes は、プロジェクトのあらゆるセットアップの種類を (ISWiSetupType オブジェクトとして) 含むコレクションです。

メンバー

テーブル 160 • ISWiSetupTypes コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWISetupTypes コレクション内のセットアップの種類の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	セットアップの種類のインデックス番号または名前を指定して、ISWiSetupType オブジェクトを読み出します。たとえば次のステートメントはコレクションに最初のセットアップタイプへの参照と、CSSaveSpace というセットアップタイプを作成します。 Set pSetupFile1 = pProj.ISWiSetupTypes.Item(1) Set pSetupFile2 = pProj.ISWiSetupTypes.Item("CSSaveSpace") Item は ISWiSetupTypes のデフォルトプロパティです。つまり pProj.ISWiSetupTypes.Item("CSSaveSpace") は pProj.ISWiSetupTypes("CSSaveSpace") と同じです。

例

次の Visual Basic コードは、特定のセットアップの種類のプロパティの設定方法を示したものです。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("IswiAutoAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.ism"

pProj.ISWiSetupTypes("CSSaveSpace").DisplayName = "クライアント セットアップ – 容量の節約"

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

ISWiShellProperties コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

ISWiShellProperties は、指定された ISWiShortcut オブジェクトのシェルのプロパティすべてのコレクションです。

メンバー

テーブル 161 • ISWiShellProperties コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、DeleteShellProperty コレクションに含まれる要素の合計数を戻します。
項目	読み取り専用プロパティ	ISWiShellProperty オブジェクトを取得する順番で、シェル プロパティのインデックス番号または名前を提供します。

例

次のサンプル VBScript は、ISWiShellProperties コレクションからプライマリ キー MyShellProperty の ISWiShellProperty オブジェクトを取得します。

```
Dim pShellProperty As ISWiShellProperty
Set pShellProperty = pShortcut.ISWiShellProperties("MyShellProperty")
```

次に適用 :

- ISWiShortcut

ISWiShortcuts コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *マージ モジュール*

ISWiShortcuts には、指定した ISWiFolder オブジェクトに属するすべてのショートカットが格納されています。

次の例では、m_pFolder に格納されているプログラムフォルダーの中の MyShortcut という名前のショートカットのコピーを作成します。

```
Dim m_pShortcut As ISWiShortcut
Set m_pShortcut = m_pFolder.ISWiShortcuts("MyShortcut")
```

メンバー

テーブル 162 • ISWiShortcuts コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiShortcuts コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	ショートカットのインデックス番号または名前を指定して、ISWiShortcut オブジェクトを読み出します。 Item は ISWiShortcuts のデフォルトプロパティです。つまり、m_pFolder.ISWiShortcuts.Item("MyShortcut") は m_pFolder.ISWiShortcuts("MyShortcut") と同じです。

次に適用 :

- ISWiFolder

ISWiSISProperties コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript MSI*
- *マージ モジュール*

このコレクションには、現在のプロジェクトにおけるすべての "概要情報ストリーム" 設定の値が含まれています。このコレクションにある要素にアクセスすると、[一般情報] ビューのすべての "概要情報ストリーム" 設定を設定できます。

以下の Visual Basic コードでは、プロジェクトにある各 "概要情報ストリーム" 設定を読み取り、各設定とその値を表示します。

```
Dim pProject As ISWiProject

Set pProject = New ISWiProject
pProject.OpenProject "C:¥MySetups¥SampleApp.ism", True

Dim sSISProps As String
sSISProps = "このパッケージの概要情報ストリームには、次の " & _
        pProject.ISWiSISProperties.Count & " プロパティ:"

Dim pSISProp As ISWiSISProperty
For Each pSISProp In pProject.ISWiSISProperties
    sSISProps = sSISProps & vbNewLine & vbTab & _
            pSISProp.Name & ": " & pSISProp.Value
Next
```

```
MsgBox sSISProps

pProject.CloseProject
```

メンバー

テーブル 163 ・ ISWiSISProperties コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSISProperties コレクション内の要素の合計数を戻します。概要情報ストリームには現在 7 項目あるため、Count の値は常に 7 です。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSISProperty オブジェクトを読み出します。 プロパティの名前は大文字と小文字を区別します。したがって、“TemplateSummary” と “Templatesummary” は 2 つの異なるプロパティと認識されます。 Item は ISWiSISProperties のデフォルトプロパティです。つまり、pISWiProject.ISWiSISProperties.Item(″Subject″) は pISWiProject.ISWiSISProperties(″Subject″) と同じです。

次に適用 :

- ISWiProject

ISWiSQLConnections コレクション

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLConnections コレクションには、現在のプロジェクトのすべての SQL Server 接続が含まれています。

メンバー

テーブル 164 • ISWiSQLConnections コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSQLConnections コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSQLConnectionオブジェクトを読み出します。 接続の名前は大文字と小文字を区別します。したがって、“SQLConnection” と “SQLconnection” は 2 つの異なる接続と認識されます。 Item は ISWiSQLConnections のデフォルトプロパティです。つまり、pISWiProject.ISWiSQLConnections.Item(″SQLConnection″) は pISWiProject.ISWiSQLConnections(″SQLConnection″)と同じです。

次に適用 :

- ISWiProject

ISWiSQLDatabaseServers コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLDatabaseServers コレクションには、実行時にバージョン要件によってターゲットとして選択可能なすべてのデータベースサーバーが含まれます。

メンバー

テーブル 165 • ISWiSQLDatabaseServers コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiSQLDatabaseServers コレクション内の要素の合計数を戻す場合に使用します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSQLDatabaseServer オブジェクトを読み出します。 データベース サーバー名は大文字と小文字を区別します。したがって、“SQLDatabaseServer” と “SQLDatabaseserver” は 2 つの異なる項目と認識されます。 Item は ISWiSQLDatabaseServers のデフォルトプロパティです。つまり、pISWiSQLConnection.ISWiSQLDatabaseServers.Item(″SQLDatabaseServer″) は pISWiSQLConnection.ISWiSQLDatabaseServers(″SQLDatabaseServer″) と同じです。

次に適用：

- ISWiSQLConnections

ISWiSQLReplaces コレクション

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLReplaces コレクションには、現在のプロジェクトの中のすべての ISWiSQLReplace オブジェクトが含まれています。

メンバー

テーブル 166 • ISWiSQLReplaces コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSQLReplaces コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSQLReplace オブジェクトを読み出します。 コレクションの名前は大文字と小文字を区別します。したがって、“SQLReplaces” と “SQLreplaces” は 2 つの異なるコレクションと認識されます。 Item は ISWiSQLReplaces のデフォルトプロパティです。つまり、pISWiSQLScript.ISWiSQLReplaces.Item(″SQLReplace″) は pISWiSQLScript.ISWiSQLReplaces(″SQLReplace″) と同じです。

次に適用：

- ISWiSQLScript

ISWiSQLRequirements コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLRequirements は、データベースサーバーと関連付けられたすべての ISWiSQLRequirement オブジェクトのコレクションを戻します。

メンバー

テーブル 167 • ISWiSQLRequirements コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSQLRequirements コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSQLRequirement オブジェクトを読み出します。 プロパティの名前は大文字と小文字を区別します。したがって、“SQLRequirements” と “SQLrequirements” は 2 つの異なるプロパティと認識されます。 Item は ISWiSQLRequirements のデフォルトプロパティです。つまり、pISWiSQLConnection.ISWiSQLRequirements.Item(″SQLRequirement″) は ppISWiSQLConnection.ISWiSQLRequirements(″SQLRequirement″) と同じです。

次に適用：

- ISWiSQLConnections

ISWiSQLScriptErrors コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLScriptErrors コレクションには、ISWiSQLScriptError オブジェクトのカスタム SQL スクリプト エラー処理のエントリすべてが含まれます。

メンバー

テーブル 168 - ISWiSQLScriptErrors コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSQLScriptErrors コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSQLScriptError オブジェクトを読み出します。

次に適用：

- ISWiSQLScript

ISWiSQLScripts コレクション

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*

ISWiSQLScripts コレクションには、SQL 接続の中のすべての SQL スクリプトが含まれています。

メンバー

テーブル 169 · ISWiSQLScripts コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSQLScripts コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiSQLScript オブジェクトを読み出します。 スクリプトの名前は大文字と小文字を区別します。したがって、“SQLScript” と “SQLscript” は 2 つの異なる項目と認識されます。 Item は ISWiSQLScripts のデフォルトプロパティです。つまり、pISWiSQLConnection.ISWiSQLScripts.Item(″SQLScript″) は pISWiSQLConnection.ISWiSQLScripts(″SQLScript″) と同じです。

次に適用 :

- ISWiSQLConnections

ISWiStringEntries コレクション

***プロジェクト·** この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *DIM*
- *InstallScript*
- *InstallScript MSI*
- *InstallScript オブジェクト*
- *マージ モジュール*

ISWiStringEntries コレクションには、現在の言語のすべての文字列エントリが含まれています。

メンバー

テーブル 170 - ISWiStringEntries コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiStringEntries コレクションの文字列エントリの合計数を戻します。
項目	読み取り専用プロパティ	文字列エントリのインデックス番号または名前を指定して、ISWiStringEntry オブジェクトを読み出します。

次に適用：

- ISWiLanguage

ISWiUpgradeTableEntries コレクション

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*
- *InstallScript MSI*

ISWiUpgradeTableEntries コレクションは、ダイレクト エディターの Upgrade テーブルだけでなく、[アップグレード] ビューで定義されたスタティック メジャー アップグレード アイテムのすべてを含みます。

メンバー

テーブル 171 • ISWiUpgradeTableEntries コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティは、ISWiUpgradeTableEntries コレクション内の要素の合計数を返す場合に使用します。
項目	読み取り専用プロパティ	ISWiUpgradeTableEntries アイテムを読み出すために、インデックス番号またはアップグレードコードを提供します。 Item は IISWiUpgradeTableEntries のデフォルトプロパティです。つまり、m_ISWiProject.ISWiUpgradeTableEntries.Item("12345678-1234-1234-1234-123456789012") は m_ISWiProject.ISWiUpgradeTableEntries("12345678-1234-1234-1234-123456789012") と同じです。

例

```
Dim pProj As ISWiProject
Dim oUpgradeItem As ISWiUpgradeTableEntry

Set oProj = New ISWiProject
oProj.OpenProject "C:¥MySetup.ism"

For Each oUpgradeItem In oProj.ISWiUpgradeTableEntries
    oUpgradeItem.Language = "1033"
Next

oProj.SaveProject
oProj.CloseProject
```

特定のアイテムをターゲットとするには、上のコードの For Each ループを次のコードに置き換えます:

```
For Each oUpgradeItem In oProj.ISWiUpgradeTableEntries
    If oUpgradeItem.UpgradeCode = "{12345678-1234-1234-1234-123456789012}" Then
        oUpgradeItem.Language = "1033"
    End If
Next
```

次に適用 :

- ISWiProject

≫ アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

各 InstallShield オートメーション オブジェクトは、機能、コンポーネント、ファイルなどのインストール要素を表します。このセクションでは、各 InstallShield オートメーション オブジェクトが説明されています。

オートメーションインターフェイスで最高レベルのオブジェクトは ISWiProject オブジェクト です。このインターフェイスにアクセスするには、まずこのオブジェクトを作成する必要があります。次いで、そのオブジェクト、メソッド、プロパティを使用して、プロジェクトを変更できます。

ISWiProject オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiProject には、アドバンスト UI またはスイート/アドバンスト UI プロジェクト (.issuite) ファイルを開いたり閉じたり、保存するためのインターフェイスが含まれます。InstallShield をシステムにインストールすると、次の VB や VBScript 構文を使って ISWiProject のコピーのインスタンスを作成できます。

```
Set m_ISWiProject = CreateObject("ISWiAutoSuiteAutomation Interface Version.ISWiProject")
```

インストール プロジェクトを開くには、以下の手順で OpenProject メソッドを呼び出します。

```
' .ism ファイルへのパスのビルド
strFile = "C:¥InstallShield 2021 Projects¥Test.issuite"
m_ISWiProject.OpenProject strFile
```

重要: *オートメーション インターフェイスを 64 ビット マシンで使用する場合、これを 32 ビット実行可能ファイルを使ってロードする必要があります。これを怠った場合、ISWiProject またはその他のオブジェクトのインスタンスを作成したときに、エラーが発生する可能性があります。詳細については、「64 ビット システム上でオートメーション インターフェイスを使用する」を参照してください。*

メンバー

オートメーション インターフェイスで、ISWIProject オブジェクトの次のメンバーが使用できます。一部のメソッド、プロパティ、およびコレクションは、特定のプロジェクトの種類でのみ使用できます。

テーブル 1 ・ ISWiProject オブジェクトのメンバー

名前	種類	説明
ActiveLanguage	読み書き文字列プロパティ	プロジェクトのデフォルト言語を取得またはは設定します。単一の言語を言語識別子として指定します。例: project.ActiveLanguage = "1033";
AddExitCondition	メソッド	アドバンスト UI またはスイート/アドバンスト UI インストールがインストール終了の前に、様々な条件に応じて表示する終了エラー メッセージを追加します。
AddLanguage	メソッド	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトに言語を追加します。
AddPathVariable	メソッド	プロジェクトにパス変数を追加します。
AddProperty	メソッド	指定された名前を持つアドバンスト UI またはスイート/アドバンスト UI プロパティをプロジェクトに追加します。
AddSetupFile	メソッド	指定された名前でプロジェクトにサポート ファイルを追加します。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
AddSuiteAction	メソッド	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトにアクションを追加します。
AddSuiteFeature	メソッド	機能をプロジェクトに追加、または既存する機能のサブ機能として追加します。
AddSuitePackage	メソッド	指定されたターゲット ファイル、およびファイルの追加オプションと共にパッケージをプロジェクトに追加します。
AddSuiteRelease	メソッド	プロジェクトにリリースを追加します。
AddSuiteTransaction	メソッド	Windows Installer トランザクションをプロジェクトに追加します。 プロジェクトにトランザクションを追加した後、トランザクションに .msi および .msp パッケージを追加することができます。
ArpComments	読み書き文字列プロパティ	製品に関するコメントを取得または設定します。この情報は、[プログラムと機能] で製品のエントリに表示されます。
ArpDisableChange	読み書きブール値プロパティ	[一般情報] ビューの "変更ボタンを無効にする" 設定の値を取得または設定します。この設定を使って、[プログラムと機能] で、製品のエントリの [変更] ボタンを無効にするかどうかを指定します。[変更] ボタンは、エンドユーザーが製品のインストール後、インストール オプションを変更できるボタンです。エン ドユーザーは必要に応じて機能を追加または削除できます。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
ArpDisableRegistration	読み書きブール値プロパティ	[プログラムと機能] で製品の登録を抑制または許可します。登録が無効化された場合、その他の Arp プロパティは無視されます。次の値が可能です: • False—アドバンスト UI またはスイート/アドバンスト UI 製品のエントリが、ターゲット システムの [プログラムと機能] で表示されます。これがデフォルトの値です。 • True—アドバンスト UI またはスイート/アドバンスト UI 製品のエントリが、ターゲット システムの [プログラムと機能] で表示されません。エンドユーザーが [プログラムと機能] を使って、製品の削除やメンテナンスを行ったり、サポート情報を表示できなくなります。このオプションを選択すると、その他の Arp プロパティは無関係となります。
ArpDisableRemove	読み書きブール値プロパティ	[一般情報] ビューの "削除ボタンを無効にする" 設定の値を取得または設定します。この設定を使って、[プログラムと機能] で、製品のエントリの [削除] ボタンを無効にするかどうかを指定します。[削除] ボタンを使用すると、エンドユーザーは 1 つのボタンをクリックするだけで製品を削除することができます。この場合、アンインストーラーはコンパクトなユーザー インターフェイスで実行します。 エンド ユーザーが [削除] ボタンをクリックして製品を削除すると、プロジェクトの [ユーザー インターフェイス] シーケンス内のアクションが実行されます。
ArpHelpContact	読み書き文字列プロパティ	エンドユーザーのテクニカルサポート窓口となる担当者または部門の名前を取得または設定します。 Windows の一部のバージョンでは、この情報は、[プログラムと機能] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。
ArpHelpTelPhone	読み書き文字列プロパティ	製品のテクニカル サポートの電話番号を取得または設定します。 Windows の一部のバージョンでは、この情報は、[プログラムと機能] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。
ArpHelpUrl	読み書き文字列プロパティ	エンド ユーザーが製品のテクニカル サポート情報を参照できる URL を取得または設定します。この URL は、[プログラムと機能] で製品のエントリに表示されます。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
ArpIcon	読み書き文字列プロパティ	[プログラと機能] で製品のエントリに使用するアイコン リソースを含む .ico、.exe、または .dll ファイルへの、開発システム上でのパスを取得または設定します。 ターゲット システム上に存在するファイルに含まれるアイコンを使用する場合 (たとえば、アドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージの 1 つによってインストールされるファイルに含まれるアイコンを使用する場合)、ディレクトリ プロパティをアイコン ファイルのパスの一部として指定できます。このプロパティを空白のままにすると、[リリース] ビューの Setup.exe タブにある "Setup.exe アイコン ファイル" 設定で指定されたアイコンが使用されます。
ArpIconIndex	読み書き整数プロパティ	指定したアイコン ファイルに 1 つ以上のアイコン リソースがある場合、ArpIconIndex プロパティにインデックスを指定します。 • 負の数以外の整数を指定すると、実行可能ファイルのアイコン リソースの順番が参照されます。たとえば、0 はファイル内の最初のアイコン、1 は 2 番目のアイコン、2 は 3 番目のアイコンを参照します。 • 負の数字は特定のリソース ID を参照するために使用します。たとえばアイコン インデックス -12 は、リソース ID が 12 のアイコンを示します。
ArpProductUrl	読み書き文字列プロパティ	会社または製品の汎用 URL を取得および設定します。たとえば、http://www.mycompany.com。 Windows の一部のバージョンでは、[サポート情報] ダイアログ ボックスに表示される発行元の名前は、この URL へのハイパーリンクです。[サポート情報] ダイアログ ボックスは、エンド ユーザーが [プログラムと機能] で製品のエントリ用のサポート情報ハイパーリンクをクリックすると表示されます。
ArpPublisher	読み書き文字列プロパティ	製品を作成した会社の名前を取得および設定します。この情報は、[プログラムと機能] で製品のエントリに表示されます。
ArpReadMe	読み書き文字列プロパティ	製品の Readme ファイルまたは [カスタマー サポート] ページへの有効な URL を取得または設定します。 Windows の一部のバージョンでは、この情報は、[プログラムと機能] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
ArpUpdateUrl	読み書き文字列プロパティ	エンド ユーザーが製品のアップデート情報を参照、または最新バージョンをダウンロードできる URL を取得または設定します。 Windows の一部のバージョンでは、この情報は、[プログラムと機能] で製品のエントリ用の [サポート情報] ダイアログ ボックスで表示されます。
CloseProject	メソッド	アドバンスト UI またはスイート/アドバンスト UI プロジェクトを閉じます。
CreateProject	メソッド	アドバンスト UI またはスイート/アドバンスト UI プロジェクトの新しい InstallShield プロジェクト ファイル (.issuite) を作成します。
DeleteExitCondition	メソッド	プロジェクトから終了エラー メッセージを削除します。
DeleteLanguage	メソッド	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトから言語を削除します。
DeletePathVariable	メソッド	プロジェクトからパス変数を削除します。
DeleteProperty	メソッド	プロジェクトからアドバンスト UI またはスイート/アドバンスト UI プロパティを削除します。
DeleteSetupFile	メソッド	プロジェクトからサポートファイルを削除します。
DeleteSuiteAction	メソッド	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトからアクションを削除します。
DeleteSuiteFeature	メソッド	プロジェクトから機能を削除します。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
DeleteSuitePackage	メソッド	プロジェクトからパッケージまたは Windows Installer トランザクションを削除します。
DeleteSuiteRelease	メソッド	プロジェクトからリリースを削除します。
ForceUpgrade	メソッド	プロジェクトを InstallShield の以前のバージョンから現在のバージョンにアップグレードします。
GenerateGUID	メソッド	文字列として返される新しい一意の GUID を生成します。
OpenProject	メソッド	オートメーション インターフェイスを使って変更できるように、アドバンスト UI またはスイート/アドバンスト UI プロジェクトを開きます。
ISWiLanguages	コレクション	現在のプロジェクトに含まれるすべての言語を含みます。
ISWiPathVariables	コレクション	現在のプロジェクトに含まれるすべてのアドバンスト UI またはスイート/アドバンスト UI パス変数を含みます。
ISWiProperties	コレクション	現在のプロジェクトに含まれるすべてのアドバンスト UI またはスイート/アドバンスト UI プロパティを含みます。
ISWiSetupFiles	コレクション	現在のプロジェクトに含まれるすべてのサポートファイルを含みます。
ISWiSuiteActions	コレクション	 ***プロジェクト*** *- これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション*** *- スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトに含まれるすべてのスイート/アドバンスト UI アクションを含みます。
ISWiSuiteEvent	メソッド	 ***プロジェクト*** *- これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション*** *- スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクト内の特定のイベントにアクセスします。

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
ISWiSuiteEvents	コレクション	 ***プロジェクト*** *- これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション*** *- スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトに含まれるすべてのイベントを含みます。
ISWiSuiteFeatures	コレクション	プロジェクトのすべての機能を (ISWiSuiteFeature オブジェクトとして) 含みます。 ISWiSuiteFeatures が ISWiProject のメンバーの場合、これにはプロジェクトのルートレベルの機能がすべて含まれます。サブ機能のコレクションを読み出すには、ISWiSuiteFeature オブジェクトの "機能" プロパティにアクセスします。その結果は、現在のルートレベル機能のすぐ下にあるサブ機能を含む ISWiSuiteFeatures コレクションです。これを繰り返すことで、さらに下のレベルのサブ機能が表示されます。
ISWiSuiteExitConditions	コレクション	プロジェクトのすべての終了条件が含まれます。
ISWiSuitePackages	コレクション	プロジェクトのすべてのパッケージ (ISWiSuitePackage オブジェクトとして)、およびすべての Windows Installer トランザクション (ISWiSuitePackage オブジェクトとして) を含みます。
ISWiSuiteReleases	コレクション	アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるすべてのリリースを含みます。
ProductName	読み書き文字列プロパティ	製品名を取得または設定します。製品名がどのように使用されるかについては、「製品名の指定」を参照してください。
ProjectType	読みとり専用整数プロパティ	プロジェクトの種類を取得します。有効な値は次のとおりです: • eptSuite (16)—アドバンスト UI またはスイート/アドバンスト UI プロジェクト

テーブル 1 • ISWiProject オブジェクトのメンバー (続き.)

名前	種類	説明
ProductVersion	読み書き文字列プロパティ	製品のバージョン番号を取得または設定します。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャー バージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。詳細については、「製品バージョンを指定する」を参照してください。
SaveProject	メソッド	アドバンスト UI またはスイート/アドバンスト UI プロジェクトを保存します。
SuiteGuid	読み書き文字列プロパティ	アドバンスト UI またはスイート/アドバンスト UI インストールを一意に識別する GUID を取得または設定します。異なる GUID を生成するには、GenerateGUID を呼び出します。 このコードはアドバンスト UI またはスイート/アドバンスト UI インストールを一意的に識別するため、リリースを既に配布している場合はスイート GUID の変更はお薦めできません。

AddExitCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddExitCondition メソッドは、プロジェクトに終了条件を追加します。このメソッドの使用方法は、InstallShield の [一般情報] ビューで終了条件を定義する方法に似ています。ISWiSuiteExitCondition オブジェクトを使って、終了条件のプロパティを設定できます。

構文

AddExitCondition (Message As String)

パラメーター

テーブル 2 ・ AddExitCondition メソッドのパラメーター

パラメーター	説明
Message	構成中の条件が実行時に True 評価された場合にアドバンスト UI またはスイート/アドバンスト UI インストールで表示するエラー メッセージを指定します。 別の方法として、値がインストールで使用するエラー メッセージである文字列 ID を入力することもできます。

次に適用：

- ISWiProject

AddLanguage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

***エディション・** 複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

AddLanguage メソッドは、プロジェクトに言語を追加します。このメソッドの使用方法は、InstallShield の [一般情報] ビューで言語を追加する方法に似ています。ISWiLanguage オブジェクトを使って、新しい言語のプロパティを設定できます。

構文

AddLanguage (LangId As String)

パラメーター

テーブル 3 ・ AddLanguage メソッド パラメーター

パラメーター	説明
LangId	現在のプロジェクトに追加する言語 ID を文字列で指定します。例: oProject.AddLanguage "1035"

次に適用：

- ISWiProject

AddPathVariable メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddPathVariable メソッドは、プロジェクトにパス変数を追加します。このメソッドの使用は、InstallShield の [パス変数] ビューでパス変数を追加するのに類似しています。新しいパス変数のプロパティは、ISWiPathVariable オブジェクトを使って設定することができます。

重要: *定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

構文

AddPathVariable (sName As String) As ISWiPathVariable

パラメーター

テーブル 4 : AddPathVariable メソッド パラメーター

パラメーター	説明
sName	追加するパス変数の名前を指定します。

次に適用 :

- ISWiProject

AddProperty メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddProperty メソッドは、指定された名前を持つアドバンスト UI またはスイート/アドバンスト UI プロパティをプロジェクトに追加します。

構文

AddProperty (Name As String) As ISWiProperty

パラメーター

テーブル 5 ・ AddProperty メソッド パラメーター

パラメーター	説明
名前	追加するアドバンスト UI またはスイート/アドバンスト UI プロパティの名前を指定します。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("ISWiAutoSuiteAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.issuite"

Dim pProperty As ISWiProperty
Set pProperty = pProj.AddProperty ("MYPROP")
pProperty.Value = "MyPropValue"

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

AddSetupFile メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSetupFile メソッドは、現在のプロジェクトに指定された名前でサポート ファイルを追加します。このメソッドの使用は、InstallShield の [サポート ファイル] ビューでパス変数を追加するのに類似しています。新しいパス変数のプロパティは、ISWiSetupFile オブジェクトを使って設定できます。

構文

AddSetupFile (FileName As String) As ISWiSetupFile

パラメーター

テーブル 6 : AddSetupFile メソッドのパラメーター

パラメーター	説明
FileName	追加するサポート ファイルの完全修飾ファイル名を指定します。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("ISWiAutoSuiteAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.issuite"
Dim pSetupFile As ISWiSetupFile

Set pSetupFile = pProj.AddSetupFile ("C:¥My Files¥MySupportFile.ext")

pProj.SaveProject
pProj.CloseProject
```

次に適用 :

- ISWiProject

AddSuiteAction メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuiteAction メソッドは、プロジェクトにアクションを追加します。

構文

AddSuiteAction (ActionName As String, ActionType As ISWiSuiteActionType) as ISWiSuiteAction

パラメーター

テーブル 7 • AddSuiteAction メソッドのパラメーター

パラメーター	説明
ActionName	追加するアクションの名前を指定します。
ActionType	アクションの種類を指定します。選択可能なオプションは以下のとおりです: • **esatDll (1)**—DLL ファイルにある関数を呼び出します。 • **esatPowerShell (2)**—PowerShell スクリプトを実行して、システムの構成タスクを行います。 • **esatExe (4)**—実行可能ファイルを起動します。スイート/アドバンスト UI エンジンは、アクションが完了するまで待機してから、インストールの次の処理に進みます。 • **esatStartExe (8)**—実行可能ファイルを起動します。スイート/アドバンスト UI エンジンは、次のインストール処理と同時にアクションを実行します。 • **esatSetProperty (16)**—スイート/アドバンスト UI プロパティを設定します。 • **esatIScript (32)**—InstallScript コードを実行します。 • **esatClr (64)**—マネージ アセンブリのパブリック メソッドを呼び出します。 • **esatClr (128)**— .NET Core マネージ アセンブリのパブリック メソッドを呼び出します。

次に適用 :

- ISWiProject

AddSuiteFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

AddSuiteFeature メソッドを使うと、指定した名前で機能を作成できます。AddSuiteFeature は、ISWiProject のメンバーです。新しい機能が、現在のプロジェクトに最上位機能として追加されましたが、このメソッドが ISWiSuiteFeature オブジェクトのメンバーのとき、新しい機能は現在の機能のサブ機能として追加されます。

機能には、新機能を InstallShield に追加すると、その新機能に与えられるデフォルトと同じプロパティが作成されます。

機能を追加した後で、その機能の他のプロパティを設定し、パッケージと関連付けできます。

構文

AddSuiteFeature (SuiteFeatureKey As String) As ISWiSuiteFeature

パラメーター

テーブル 8 ・ AddSuiteFeatureメソッドのパラメーター

パラメーター	説明
SuiteFeatureKey	機能キーは、[機能] ビューに表示される機能の名前と同一です。その名前は、新しい ISWiSuiteFeature オブジェクトのインデックスになります。この文字列には、アルファベット、ピリオド (.) およびアンダースコア (_) を使用できますが、名前の先頭は文字またはアンダースコアにする必要があります。プロジェクトにある機能名に、FeatureKey と同じ名前は付けられません。 機能名では、大文字と小文字が区別されます。

例

次の Visual Basic の行は、AddSuiteFeature を呼び出して機能およびサブ機能を追加します:

```
Dim pProject As ISWiProject

Set pProject = New ISWiProject
pProject.OpenProject "C:¥MySetups¥SampleApp.issuite"

Dim pFeat1, pFeat2 As ISWiSuiteFeature
Dim sFeat1Name, sFeat2Name As String

sFeat1Name = "ParentFeature"
sFeat2Name = "SubFeature"

' 機能 ParentFeature を追加し、次に SubFeature を ParentFeature の下に追加
Set pFeat1 = pProject.AddSuiteFeature(sFeat1Name)
Set pFeat2 = pFeat1.AddSuiteSubFeature(sFeat2Name)

pProject.SaveProject
pProject.CloseProject
```

次に適用 :

- ISWiProject

AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

プロジェクト固有の違いについては、必要に応じて記述されています。

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuitePackage メソッドは、パッケージ (.msi パッケージ、InstallScript パッケージ、.exe パッケージ、その他のサポートされている種類のパッケージ) を指定されたターゲット ファイルおよび追加ファイルのオプションと共に追加します。このメソッドの使用方法は、InstallShield の [パッケージ] ビューでパッケージを追加する方法に似ています。ISWiSuitePackage オブジェクトを使って、新しいパッケージのプロパティを設定できます。

構文

AddSuitePackage (PackageType As ISWiSuitePackageType, TargetFileName As String, AdditionalFiles As ISWiAdditionalFiles) As ISWiSuitePackage

パラメーター

テーブル 9 • AddSuitePackage メソッドのパラメーター

パラメーター	説明
PackageType	プロジェクトに追加するパッケージの種類を指定します。 次のプロジェクトの種類が、アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用できます: • esptMsi (1)—Windows Installer パッケージ (.msi) • esptMsp (2)—Windows Installer パッチ (.msp) • esptIsp (4)—InstallScript パッケージ 次のプロジェクトの種類はスイート/アドバンスト UI プロジェクトでのみ使用できます: • esptTransaction (-1)—Windows Installer トランザクションこのパッケージの種類は、スイート/アドバンスト UI プロジェクトで使用できます。 • esptExe (3)—実行可能パッケージ (.exe)このパッケージの種類は、スイート/アドバンスト UI プロジェクトで使用できます。 • esptIsmMsi (5)—基本の MSI プロジェクト (.ism) • esptIsmIsp (6)—InstallScript プロジェクト (.ism) • esptAppx (7)—サイドローディング Windows アプリ パッケージ (.appx) • esptWebDeploy (8)—Web 配置パッケージ (.zip) • esptAppxBundle (9) — サイドローディング Windows アプリ パッケージ (.appxbundle) • esptWpm (10) — Windows パッケージ マネージャー パッケージ
TargetFileName	アドバンスト UI またはスイート/アドバンスト UI インストールが呼び出すパッケージ内のファイル (.msi ファイルなど) を指定します。
AdditionalFiles	パッケージに追加するファイルの種類を指定します。選択可能なオプションは以下のとおりです: • eaftNothing (0)—何も追加しません。このパッケージには追加ファイルが不要です。 • eaftAdjacentFiles (1)—隣接するファイル、つまりこのパッケージの隣にあるファイルを追加します。 • eaftSubFolders (2)—サブフォルダー内のファイル、つまりこのパッケージの下にあるファイルを追加します。 • eaftAdjacentFilesAndSubFolders (3)—隣接するファイルおよびサブフォルダー内のファイル、つまりこのパッケージの隣および下にあるファイルを追加します。

メモ: *この方法を使って Windows パッケージ マネージャー (WPM) パッケージをプロジェクトに追加する場合、ターゲット ファイルまたは追加ファイルは指定しません。*

次に適用：

- ISWiProject

AddSuiteRelease メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuiteRelease メソッドは、プロジェクトに指定された名前でリリースを追加します。このメソッドの使用方法は、InstallShield の [リリース] ビューでリリースを追加する方法に似ています。ISWiSuiteRelease オブジェクトを使って、新しいリリースのプロパティを設定できます。

構文

AddSuiteRelease (ReleaseName As String) as ISWiSuiteRelease

パラメーター

テーブル 10 : AddSuiteRelease メソッドのパラメーター

パラメーター	説明
ReleaseName	追加するリリースの名前を指定します。

次に適用：

- ISWiProject

AddSuiteTransaction メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuiteTransaction メソッドは、Windows Installer トランザクションをスイート/アドバンスト UI プロジェクトに追加して、パッケージを返します。プロジェクトにトランザクションを追加した後、トランザクションに .msi および .msp パッケージを追加することができます。

構文

AddSuiteTransaction () As ISWiSuitePackage

パラメーター

AddSuiteTransaction メソッドにはパラメーターがありません。

次に適用：

- ISWiProject

CloseProject メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

OpenProject で開いた .issuite ファイルを閉じるには、CloseProject を呼び出します。

構文

CloseProject ()

戻り値

CloseProject は常に 0 を返します。

次に適用：

- ISWiProject

CreateProject メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

CreateProject メソッドは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトの新しい InstallShield プロジェクト ファイル (.issuite) を作成します。

構文

CreateProject (strISWiProjectFile As String, ByVal ProjectType As ISWiProjectType)

パラメーター

テーブル 11 ・ CreateProject メソッド パラメーター

パラメーター	説明
strISWiProjectFile	作成する .issuite ファイルへの完全修飾パスを入力します。完全なファイル名を含みます。
ProjectType	作成するプロジェクトの種類を指定します。有効な値は次のとおりです: • **eptSuite (16)**—アドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成します。

例

次のサンプル コードは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトを作成して、その製品バージョンを 2.00.0000 に設定します。

```
dim proj
set proj = CreateObject("ISWiAutoSuite22.ISWiProject")

proj.CreateProject "d:¥scratch¥packagetest.issuite", 16   '16: ISWiProjectType.eptSuite
proj.OpenProject "d:¥scratch¥packagetest.issuite", false

proj.ProductVersion = "2.00.0000"

proj.SaveProject
proj.CloseProject
```

次に適用：

- ISWiProject

DeleteExitCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteExitCondition メソッドは、プロジェクトから終了条件を削除します。このメソッドの使用方法は、InstallShield の [一般情報] ビューで終了条件を削除する方法に似ています。

構文

DeleteExitCondition (ExitCondition As ISWiSuiteExitCondition)

パラメーター

テーブル 12 - DeleteExitCondition メソッドのパラメーター

パラメーター	説明
ExitCondition	メソッドがプロジェクトから削除する ISWiSuiteExitCondition オブジェクトを指定します。

次に適用：

- ISWiProject

DeleteLanguage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

DeleteLanguage メソッドは、現在のプロジェクトから言語を削除します。このメソッドの使用方法は、InstallShield の [一般情報] ビューで言語を削除する方法に似ています。

構文

DeleteLanguage (Language As ISWiLanguage)

パラメーター

テーブル 13 • DeleteLanguage メソッドのパラメーター

パラメーター	説明
言語	メソッドがプロジェクトから削除する ISWiLanguage オブジェクトを指定します。

次に適用：

- ISWiProject

DeletePathVariable メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeletePathVariable メソッドは、プロジェクトから指定された名前のパス変数を削除します。

重要• *定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

構文

DeletePathVariable (pPathVar ISWiPathVariable) As Long

パラメーター

テーブル 14 • AddPathVariable メソッド パラメーター

パラメーター	説明
pPathVar	削除するパス変数の ISWiPathVariable オブジェクトを渡します。

次に適用：

- ISWiProject

DeleteProperty メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteProperty メソッドは、指定された名前を持つアドバンスト UI またはスイート/アドバンスト UI プロパティをプロジェクトをプロジェクトから削除します。

構文

DeleteProperty(ByVal pProperty As ISWiProperty) As Long

パラメーター

テーブル 15 • DeleteProperty メソッド パラメーター

パラメーター	説明
プロパティ	削除する ISWiProperty オブジェクトを渡します。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim proj As ISWiAutoSuiteAutomation Interface Version.ISWiProject
Set proj = New ISWiAutoSuiteAutomation Interface Version.ISWiProject
proj.OpenProject "c:¥mysetups¥your project name-3.issuite"
Dim pProp As ISWiAutoSuiteAutomation Interface Version.ISWiProperty
Set pProp = proj.ISWiProperties.Item("MyProperty")
proj.DeleteProperty pProp
```

次に適用：

- ISWiProject

DeleteSetupFile メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSetupFile メソッドは、プロジェクトから指定のサポート ファイルの選択を削除します。

構文

DeleteSetupFile (pSetupFile As ISWiSetupFile) As Long

パラメーター

テーブル 16 ・ DeleteSetupFile メソッド パラメーター

パラメーター	説明
pSetupFile	削除する ISWiSetupFile オブジェクトを渡します。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("ISWiAutoSuiteAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.issuite"
Dim pSetupFile As ISWiSetupFile

Set pSetupFile = pProj.ISWiSetupFiles.Item("MySupportFile.abc")
pProj.DeleteSetupFile pSetupFile

pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiProject

DeleteSuiteAction メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSuiteAction メソッドは、プロジェクトからアクションを削除します。

構文

DeleteAction (SuiteAction As ISWiSuiteAction)

パラメーター

テーブル 17 : AddSuiteAction メソッドのパラメーター

パラメーター	説明
SuiteAction	削除する ISWiSuiteAction オブジェクトを指定します。

次に適用 :

- ISWiProject

DeleteSuiteFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSuiteFeature メソッドは、現在のプロジェクトから完全に機能を削除します。

構文

DeleteSuiteFeature (Feature As ISWiSuiteFeature) As Long

パラメーター

テーブル 18 • DeleteSuiteFeature メソッドのパラメーター

パラメーター	説明
機能	削除する ISWiSuiteFeature オブジェクトを指定します。

次に適用：

- ISWiProject

DeleteSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSuitePackage メソッドは、現在のプロジェクトからパッケージまたは Windows Installer トランザクションを削除します。

構文

DeleteSuitePackage (Package As ISWiSuitePackage) As Long

パラメーター

テーブル 19 • DeleteSuitePackage メソッドのパラメーター

パラメーター	説明
パッケージ	削除する ISWiSuitePackage オブジェクトを指定します。

次に適用：

- ISWiProject

DeleteSuiteRelease メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSuiteRelease メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトからリリースを削除します。

構文

DeleteSuiteRelease (SuiteRelease As ISWiSuiteRelease)

パラメーター

テーブル 20 : DeleteSuiteRelease メソッドのパラメーター

パラメーター	説明
SuiteRelease	削除する ISWiSuiteRelease オブジェクトを渡します。

次に適用 :

- ISWiProject

ForceUpgrade メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ForceUpgrade メソッドは、プロジェクトを InstallShield の以前のバージョンから現在のバージョンにアップグレードします。

構文

ForceUpgrade (ISWiProjectFile As String)

パラメーター

テーブル 21・ForceUpgrade メソッドのパラメーター

パラメーター	説明
ISWiProjectFile	アップグレードするアドバンスト UI またはスイート/アドバンスト UI プロジェクト ファイル (.issuite) の完全修飾パスを入力します。完全なファイル名を含みます。

次に適用：

- ISWiProject

GenerateGUID メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

GenerateGUID メソッドは、文字列として返される新しい一意の GUID を生成します。

構文

GenerateGUID ()

次に適用：

- ISWiProject

ISWiSuiteEvent メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteEvent は、プロジェクト内の特定のイベントにアクセスします。

構文

ISWiSuiteEvent (EventType as ISWiSuiteEventType) as ISWiSuiteEvent

パラメーター

テーブル 22 • ISWiSuiteEvent メソッドのパラメーター

パラメーター	説明
種類	イベントの種類を指定します。選択可能なオプションは以下のとおりです: • **esetOnBegin (0)**—OnBegin イベントはスイート/アドバンスト UI インストールの最初のイベントです。このイベントにスケジュールされているアクションは、そのアクションに定義されている条件ステートメントが False 評価されない限り、すべてのモード (インストール、メンテナンス、修復など) で実行されます。 • **esetOnResuming (1)**—OnResuming イベントは、ターゲット マシンが再起動された後、スイート/アドバンスト UI インストールが再開するときに、そのアクションを起動します。 • **esetOnStaging (2)**—OnStaging イベントは、スイート/アドバンスト UI インストールがステージングされる直前に、そのアクションを起動します。つまり、(適切な場合) スイート/アドバンスト UI インストールがパッケージをダウンロードして、(適切な場合) Setup.exe ファイルからパッケージを抽出し、BrowseStageFolder ウィザード ページでエンド ユーザー が指定するディレクトリ、またはコマンドラインを使って **ISRootStagePath** プロパティで設定されたディレクトリにパッケージをコピーする前にアクションを起動します。 • **esetOnStaged (3)**—OnStaged イベントは、(適切な場合) スイート/アドバンスト UI インストールがパッケージをダウンロードし、Seup.exe ファイルから抽出して、ステージング ディレクトリにコピーした後に、そのアクションを起動します。 • **esetOnPackagesConfiguring (4)**—PackagesConfiguring イベントは、スイート/アドバンスト UI インストールがインストール、変更、修正、または削除される直前に、そのアクションを起動します。 • **esetOnPackagesConfigured (5)**—OnPackagesConfigured イベントは、スイート/アドバンスト UI インストールがインストール、変更、修正、または削除された時に、そのアクションを起動します。

テーブル 22・ISWiSuiteEvent メソッドのパラメーター (続き.)

パラメーター	説明
Type (続き)	• esetOnRebooting (6)—OnRebooting イベントは、スイート/アドバンスト UI インストールがマシンの再起動をトリガする直前に、そのアクションを起動します。 • esetOnEnd (7)—OnEnd イベントは、スイート/アドバンスト UI インストールが正常に終了する直前に、そのアクションを起動します。エンド ユーザーがスイート/アドバンスト UI インストールをキャンセルした場合、またはインストールでエラーが発生した場合、OnEnd イベントは実行しません。 • esetOnPackageConfiguring (8)—このイベントは、スイート/アドバンスト UI インストールがパッケージをインストール、削除、修正、または変更する前に起動するアクションを起動します。 • esetOnPackageConfigured (9)—このイベントは、スイート/アドバンスト UI インストールがパッケージをインストール、削除、修正、または変更した後に起動するアクションを起動します。 詳細については、「スイート/アドバンスト UI インストールにおけるイベントの種類」を参照してください。

次に適用：

- ISWiProject

OpenProject メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクトを開いてオートメーション インターフェイスを使って変更するには、.issuite ファイルへの完全修飾パスを使って OpenProject を呼び出します。このメソッドは、InstallShield にプロジェクトをロードするのと同じです。

メモ: *OpenProject メソッドは、InstallShield プロジェクトが開かれていると失敗します。*

構文

OpenProject (strISWiProjectFile As String, Optional ByVal bReadOnly As Boolean) As Long

パラメーター

テーブル 23 • OpenProject メソッドのパラメーター

パラメーター	説明
strISWiProjectFile	プロジェクト ファイル (.issuite) への完全修飾パスを入力します。
bReadOnly	書き込み権限を使ってプロジェクトを開くかどうかを指定します。True を指定すると、プロジェクトは読み取り専用ファイルとして開かれます。このオプションのパラメーター値は、デフォルトで False に設定されています。

戻り値とエラー

テーブル 24 • OpenProject メソッドの戻り値とエラー

戻り値	説明
0	プロジェクトが正しく開きました。
1	プロジェクトは開きましたが、他のプロセスによりロックされています。
2	プロジェクトを開いて書き込みアクセス権を得ようとしましたが、プロジェクトは読み取り専用モードで開かれました。
エラーコード 1100	ファイルを開けません: %1.ファイルが有効な InstallShield プロジェクトであることを確認してください。

例

次の Visual Basic 行では、このメソッドを示します。

```
Dim pProj As ISWiProject
Set pProj = CreateObject("ISWiAutoSuiteAutomation Interface Version.ISWiProject")
pProj.OpenProject "C:¥MySetups¥Project1.issuite"
' プロジェクトをここで変更します。
pProj.SaveProject
pProj.CloseProject
```

次に適用：

- ISWiProject

SaveProject メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI またはスイート/アドバンスト UI プロジェクト ファイルに加えた任意の変更をオートメーションインターフェイスで保存するには、SaveProject メソッドを呼び出します。

構文

SaveProject ()

戻り値とエラー

テーブル 25 - SaveProject メソッドの値

戻り値	説明
ERROR_SUCCESS (0)	プロジェクトが保存されました。
エラーコード 1104	ファイルを保存できません: %1プロジェクトが読取専用で開かれているか、もしくは別の処理で使用中です。

次に適用 :

- ISWiProject

ISWiLanguage オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiLanguage オブジェクトは、プロジェクトに文字列エントリが含まれている言語を表します。これを使って翻訳された文字列へのアクセスが可能となります。

言語を含む、または除外するには、それぞれの関連 ISWiLanguage オブジェクトに IsIncluded プロパティを使用します。

メンバー

テーブル 26 ・ ISWiLanguage オブジェクトのメンバー

名前	種類	説明
AddStringEntry	メソッド	新しい文字列エントリを言語に追加します。
DeleteStringEntry	メソッド	文字列エントリをプロジェクトから削除します。
Id	読み取り専用文字列プロパティ	言語の ID を取得します。
IsIncluded	読み書きブール値プロパティ	言語を含む、除外する、または現在の状態を確認します。
Name	読み取り専用文字列プロパティ	現在の言語の名前を取得します。

次に適用：

- ISWiProject

AddStringEntry メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddStringEntry メソッドは、新しい文字列エントリを言語に追加します。

構文

AddStringEntry (StringId) As ISWiStringEntry

パラメーター

テーブル 27 • AddStringEntry メソッドのパラメーター

パラメーター	説明
StringId	文字列エントリの言語識別子を指定します。

次に適用：

- ISWiLanguage

DeleteStringEntry メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteStringEntry メソッドは、プロジェクトから文字列エントリを削除します。

構文

DeleteStringEntry (StringEntry As ISWiStringEntry)

パラメーター

テーブル 28 • DeleteStringEntry メソッドのパラメーター

パラメーター	説明
StringEntry	プロジェクトから削除したい ISWiStringEntry オブジェクトを指定します。

次に適用：

- ISWiLanguage

ISWiPathVariable オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

SWiPathVariable は、プロジェクトに含まれるパス変数を表します。パス変数によって、プロジェクトのビルド パスをマシン間で移動し易くなります。

パス変数を追加または削除するには、ISWiProject.AddPathVariable または ISWiProjectDeletePathVariable を呼び出します。

重要: *定義済みパス変数を変更または削除することはできません。これを試みると、例外エラー 3142 が発生します。詳細については、「定義済みパス変数」を参照してください。*

メンバー

テーブル 29 • ISWiPathVariable オブジェクトのメンバー

名前	種類	説明
Name	読み書き文字列プロパティ	パス変数の名前を取得または設定します。
PathVarType	読み書き整数プロパティ	この列挙された整数プロパティは、パス変数の種類を取得または設定します。次の値が可能です: • epvtPreDefined (1)—定義済みパス変数は特定の標準 Windows ディレクトリに設定された変数です。これらの変数は修正したり名前を変更することはできませんが、あらかじめ決められたディレクトリをポイントするためにインストールプロジェクトで使用することができます。 • epvtCustom (2)—カスタム パス変数は、*標準パス変数*または*ユーザー定義変数*と呼ばれることもあります。これらの変数タイプは InstallShield で定義されます。これらの変数は、レジストリ、システムパスなど、外部のソースには依存しません。 • epvtEnvironment (4)—環境パス変数は、システムの環境変数の値に基づきます。プロジェクトの環境変数パス変数に、マシン上に既存する環境変数を設定できます。 • epvtRegistry (8)—レジストリ パス変数を使用すると、指定したレジストリ キーのデフォルト値に基づいて独自の変数を定義することができます。パス変数をキーの値に設定する際、レジストリ キーが既に存在している必要があります。
TestValue	読み書き文字列プロパティ	パス変数のテスト値を取得または設定します。

テーブル 29 • ISWiPathVariable オブジェクトのメンバー (続き.)

名前	種類	説明
Value	読み書き文字列プロパティ	パス変数の値を取得または設定します。 **標準パス変数の場合** 標準変数の場合、変数がポイントするディレクトリを入力します。 ***メモ:*** *また、参照先となる他のパス変数の名前を山括弧で囲むことにより、定義済みの値の中でそのパス変数を参照することができます。たとえば、*C:¥ *という値を持つ MyRoot というパス変数がある場合、そのパス変数を Games という別の変数のパス変数定義で参照することができます。Games 変数の実際のパスは* C:¥Programs¥GameFiles *のようになりますが、Games を* <MyRoot>¥Programs¥GameFiles *と定義することができます。ただし、パス変数を自己参照しようとした場合は、使用した文字列そのものが代わりに使用されます。たとえば、Games を* <MyRoot>¥Programs¥<Games> *と定義すると、実際には 、Games は* C:¥Programs¥<Games> *のように定義されます。*
Value		**レジストリ パス変数の場合** 完全なレジストリ キーを入力します。 最後の「サブキー」は、フォルダーが含まれる値名にします。たとえば、*MyRegVar* を次のように定義します。 HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey¥TestValue TestKey には次のサブキーと値があると想定します。 [HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey] @="C:¥¥MyPath1" "TestValue"="C:¥¥MyPath2" [HKEY_LOCAL_MACHINE¥Software¥TestKey¥TestValue] @="C:¥¥MyPath3" HKEY_LOCAL_MACHINE¥TestKey には TestValue というサブキーがありますが、*MyRegVar* は値 TestValue を指し、現在値は C:¥MyPath2 になります。(ただし、TestValue という値が存在しない場合、InstallShield はサブキー HKEY_LOCAL_MACHINE¥SOFTWARE¥TestKey¥TestValue のデフォルト値 (C:¥MyPath3) を読み取ります。) **環境パス変数の場合** 環境パス変数について、[環境] ダイアログ ボックスに表示されているとおりの変数名を入力します。

次に適用 :

- ISWiProject

ISWiProperty オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiProperty オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロパティを表します。ISWiProperties コレクションの中の項目を指定して、プロパティを取得します。

メンバー

テーブル 30 • ISWiProperty オブジェクトのメンバー

名前	種類	説明
Comments	読み書き文字列プロパティ	文字列エントリについての内部メモを取得または設定します。コメントは実行時には表示されません。
FormatPropertyValue	読み書きブール値プロパティ	[プロパティ マネージャー] ビューで [フォーマット済み] 列のチェックボックスを取得または設定します。値が実行時に解決されるプロパティ名、環境変数リファレンス、その他の特殊文字列を含む形式化された式であるかどうかを指定できるチェック ボックスです。実行時の解決処理は、インストール中に OnBegin イベントのアクションが実行される前、早い段階で発生します。 実行時にプロパティ値をそのまま残す場合、このプロパティを False に等しく設定します。各括弧その他の文字は、何の意味も持ちません。 形式化された式を実行時に置換するには、このプロパティを True と等しく設定します。 形式化された式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
Name	読み書き文字列プロパティ	アドバンスト UI またはスイート/アドバンスト UI プロパティの名前を取得または設定します。
Value	読み書き文字列プロパティ	アドバンスト UI またはスイート/アドバンスト UI プロパティの値を取得または設定します。

次に適用 :

- ISWiProject

ISWiSetupFile オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

ISWiSetupFile は、プロジェクトに含まれるサポート ファイルを表します。

サポート ファイルを追加または削除するには、ISWiProject.AddSetupFile または ISWiProject.DeleteSetupFile を呼び出します。

メンバー

テーブル 31 ・ ISWiSetupFile オブジェクトのメンバー

名前	種類	説明
FileName	読み取り専用文字列プロパティ	サポート ファイルのファイル名を取得します (パスなし)。
言語	読み書き文字列プロパティ	サポートファイルの言語識別子を識別する文字列を取得します。 例: m_pFile.Language = "1033"
パス	読み取り専用文字列プロパティ	サポート ファイルのソースの場所の完全修飾パス (ファイル名を含む) を取得します。 パスをハードコード化する代わりに、[パス変数] ビューまたは ISWiPathVariables コレクションの一部で定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。
ターゲット	読み書き文字列プロパティ	SETUPSUPPORTDIR 内のサポート ファイルの ターゲット ディレクトリを取得または設定します。次の行は、SETUPSUPPORTDIR の下にある MyFolder1¥MyFolder2 フォルダーにサポートファイルを保管します。 m_pFile.Target = "MyFolder1¥MyFolder2"

次に適用 :

- ISWiProject

ISWiSuiteAction オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteAction オブジェクトは、スイート/アドバンスト UI プロジェクトのアクションを表します。

アクションを追加または削除するには、ISWiProject.AddSuiteAction または ISWiProject.DeleteAction を呼び出します。ISWiSuiteEvent.AddSuiteAction Ref または ISWiSuiteEvent.DeleteSuiteActionRef を呼び出してアクションをスケジュールします。

一部の ISWiSuiteAction プロパティは特定の種類のアクションにのみ使用できます。

メンバー

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー

名前	種類	説明
AbortCode	読み書き文字列プロパティ	 ***メモ•*** *このプロパティは .exe アクションに適用します。* インストールが中止するときにアクションが戻す可能性のあるコードのリストをコンマ区切りで取得または設定します。アクションが起動またはロードできない場合、インストールは中止します。 このプロパティは、アクションの WaitForExit プロパティの値が True の場合に使用できます。
ActionName	読み取り専用文字列プロパティ	アクション名を取得または設定します。
引数	読み書き文字列プロパティ	 ***メモ•*** *このプロパティは .exe アクションに適用します。* 適切な場合、アクションと共に使用するコマンドライン パラメーターを取得または設定します。複数のパラメーターはスペースで区切ります。
CancelCode	読み書き文字列プロパティ	 ***メモ•*** *このプロパティは .exe アクションに適用します。* インストールがキャンセルするときにアクションが戻す可能性のあるコードのリストをコンマ区切りで取得または設定します。 このプロパティは、アクションの WaitForExit プロパティの値が True の場合に使用できます。
Class	読み書き文字列プロパティ	 ***メモ•*** *このプロパティはマネージ コードおよび . NET Core アクションに適用します。* 選択されたアクションが呼び出すマネージ メソッドのクラス名を取得または設定します。 詳しくは、次を参照してください: • スイート/アドバンスト UI インストールでマネージ コード アクションを使用する. • スイート/アドバンスト UI インストールで .NET Core アクションを使用する.

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー (続き.)

名前	種類	説明
DefaultResponse	読み書き整数プロパティ	 *メモ• このプロパティは .exe アクションに適用します。* アクションが戻すコードが IgnoreCode、AbortCode、RebootCode、または CancelCode プロパティに指定されていない場合にインストールが応答する方法を説明する文字列を取得または設定します。次の値が可能です: • esadcIgnore (0)—インストールは次のインストール次の処理を続行します。デフォルトでは、これが設定されています。 • esadcAbort (1)—インストールが中止します。 • esadcReboot (2)—ターゲット システムを再起動します。 • esadcCancel (3)—インストールがキャンセルします。 このプロパティは、アクションの WaitForExit プロパティの値が True の場合に使用できます。
FormatPropertyValue	読み書きブール値プロパティ	 *メモ• このプロパティは、プロパティを設定するアクションに適用します。* [イベント] ビューの "プロパティ値の形式化" 設定の値を取得または設定します。この設定を使って、"プロパティ値" 設定 (またはオートメーション インターフェイスの PropValue プロパティを使って) 入力する値が、実行時に解決させるプロパティ名、環境変数リファレンス、その他の特殊文字列を含む形式化された式であるかどうかを示します。選択可能なオプションは以下のとおりです: • True—実行時に形式化された式を置換します。 • False—実行時にプロパティ値をそのまま残します。入力する値は、形式化された式を含みません。 これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー (続き.)

名前	種類	説明
FunctionName	読み書き文字列プロパティ	 *メモ• このプロパティは DLL および InstallScript アクションに適用します。* DLL アクションの場合: 呼び出す DLL 内の関数名を取得または設定します。 InstallScript アクションの場合: 呼び出す InstallScript 関数を取得または設定します。
IgnoreCode	読み書き文字列プロパティ	 *メモ• このプロパティは .exe アクションに適用します。* インストールが次のインストール処理を続行するときにアクションが戻す可能性のあるコードのリストをコンマ区切りで取得または設定します。 このプロパティは、アクションの WaitForExit プロパティの値が True の場合に使用できます。
メソッド	読み書き文字列プロパティ	 *メモ• このプロパティはマネージ コードおよび . NET Core アクションに適用します。* インストールで呼び出すパブリック メソッドを設定または取得します。 詳しくは、次を参照してください: • スイート/アドバンスト UI インストールでマネージ コード アクションを使用する. • スイート/アドバンスト UI インストールで .NET Core アクションを使用する.
PropName	読み書き文字列プロパティ	 *メモ• このプロパティは、プロパティを設定するアクションに適用します。* プロパティ名を取得または設定します。

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー (続き.)

名前	種類	説明
PropValue	読み書き文字列プロパティ	*メモ• このプロパティは、プロパティを設定するアクションに適用します。* プロパティの値を取得または設定します。 FormatPropertyValue プロパティに True を選択した場合、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。
RebootCode	読み書き文字列プロパティ	*メモ• このプロパティは .exe アクションに適用します。* ターゲット システムが再開始するときに、アクションが戻す可能性のあるコードのリストをコンマ区切りで取得または設定します。 このプロパティは、アクションの WaitForExit プロパティの値が True の場合に使用できます。
RequireAdminPrivileges	読み書きブール値プロパティ	[イベント] ビューの "管理者権限が必要" 設定の値を取得または設定します。この設定は、このアクションが Windows Vista 以降または Windows Server 2008 以降のシステム上で管理者権限を必要とするかどうかを示します。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える」を参照してください。

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー (続き.)

名前	種類	説明
SourcePath	読み書き文字列プロパティ	 *メモ• このプロパティは PowerShell アクションに適用します。* アクションに使用する PowerShell スクリプト (.ps1) へのソースパスを取得または設定します。 実行時にそのファイルがターゲット システムに存在しない場合で、スイート/アドバンスト UI インストール内のいずれのパッケージもそれをインストールしないとき、プロジェクトに含まれるファイルをサポート ファイルとして参照できます。実行時、このファイルは製品のインストール処理中の時だけターゲット システムで使用できます。 場所をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。
TargetFile	読み書き文字列プロパティ	 *メモ• このプロパティは .exe、DLL、マネージ コード、および .NET Core アクションに適用します。* アクションに使用するファイルへのランタイム パスを取得または設定します。 実行時にそのファイルがターゲット システムに存在しない場合で、スイート/アドバンスト UI インストール内のいずれのパッケージもそれをインストールしないとき、プロジェクトに含まれるファイルをサポート ファイルとして参照できます。実行時、このファイルは製品のインストール処理中の時だけターゲット システムで使用できます。
Type	読みとり専用整数プロパティ	アクションの種類を取得します。次の値が可能です: • esatDll (1)—DLL ファイルにある関数を呼び出します。 • esatPowerShell (2)—PowerShell スクリプトを実行して、システムの構成タスクを行います。 • esatExe (4)—実行可能ファイルを起動します。 • esatSetProperty (16)—スイート/アドバンスト UI プロパティを設定します。 • esatIScript (32)—InstallScript コードを実行します。 • esatClr (64)—マネージ アセンブリのパブリック メソッドを呼び出します。 • esatClr (128)— .NET Core マネージ アセンブリのパブリック メソッドを呼び出します。

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー (続き.)

名前	種類	説明
Verb	読み書き文字列プロパティ	 ***メモ:*** *このプロパティは .exe アクションに適用します。* 選択されたアクションのファイルと共に使用する動詞を取得または設定します。 このプロパティを空白にすると、ターゲット システム上でこのファイルのデフォルトの動詞が使用されます。デフォルトの動詞が指定されていない場合は、*open* 動詞が使用されます。いずれの動詞も使用できない場合、レジストリで最初にリストされている動詞が使用されます。 runas 動詞の使用は推奨されません。アクションに管理者権限が必要な場合、アクションの RequireAdminPrivileges プロパティに True を選択します。詳細については、「アドバンスト UI およびスイート/アドバンスト UI インストール中に、ユーザー アカウント制御のプロンプト回数を最小限に抑える」を参照してください。
WaitForExit	読み書きブール値プロパティ	 ***メモ:*** *このプロパティは .exe アクションに適用します。* [イベント] ビューの "終了を待機" 設定の値を取得または設定します。この設定を使って、アクションが完了するまでインストールを待機してから、次のインストール処理を続行するどうかを示します。選択可能なオプションは以下のとおりです: • **True**—インストールは、アクションが完了するまで待機してから再開します。デフォルトでは、これが設定されています。 • **False**—インストールは、次のインストール処理と同時にアクションを実行します。 この設定で True を選択するとき、IgnoreCode、AbortCode、RebootCode、および CancelCode プロパティを使って、アクションが戻す可能性のある様々なコードを指定できます。

テーブル 32 • ISWiSuiteAction オブジェクトのメンバー (続き.)

名前	種類	説明
Window	読み書き整数プロパティ	 ***メモ:*** *このプロパティは .exe アクションに適用します。* .exe アクションが起動されたときの最初のウィンドウの状態を取得または設定します。選択可能なオプションは以下のとおりです: • esawShow (0)—ウィンドウを現在のサイズと位置で表示します。 • esawHide (1)—ウィンドウを非表示にします。 • esawMin (2)—ウィンドウを最小化して表示します。 • esawMax (3)—ウィンドウを最小化して表示します。 • esawNormal (4)—ウィンドウを元のサイズと位置で表示します。

次に適用 :

- ISWiProject

ISWiSuiteActionRef オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteActionRef オブジェクトは、スイート/アドバンスト UI イベント中にアクションのスケジュールへの参照を表します。単一イベント内を含み、アクションを複数回スケジュールすることができます。

アクションをスケジュールするには、ISWiSuiteEvent.AddSuiteActionRef を呼び出します。アクションのスケジュールへの参照を削除するには、ISWiSuiteEvent.DeleteSuiteActionRef を呼び出します。

メンバー

テーブル 33 • ISWiSuiteActionRef オブジェクトのメンバー

名前	種類	説明
DisplayText	読み書き文字列プロパティ	 ***メモ:*** *このプロパティは .exe、DLL、PowerShell、InstallScript、およびマネージコード アクションに適用します。* 選択されたアクションについて説明するテキストを取得または設定します。たとえば、アクションがターゲット システムを構成するスクリプトを実行している場合、次の文字列を入力できます: システムの構成中 InstallationProgress ウィザード ページが表示されるときに、インストールがアクションを起動する場合、入力されたテキストがウィザード ページに表示されます。
MoveTo	メソッド	スイート/アドバンスト UI アクションを指定された別のアクションの直後に発生するように再スケジュールします。
Name	読み取り専用文字列プロパティ	スケジュールされているアクションの名前を取得します。

次に適用 :

- ISWiProject

MoveTo メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

MoveTo メソッドは、スイート/アドバンスト UI アクションを指定された別のアクションの直後に発生するように再スケジュールします。

構文

MoveTo (PrecedingActionRef As ISWiSuiteActionRef)

パラメーター

テーブル 34 • MoveTo メソッドのパラメーター

パラメーター	説明
PrecedingActionRef	再スケジュールするアクションの前にあるアクションを指定します。デフォルト値 Nothing は、一番上を示します。

次に適用：

- ISWiSuiteActionRef

ISWiSuiteCondition オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteCondition オブジェクトは、次の種類の条件を表します:

- Exit 条件 (ISWiSuiteExitCondition オブジェクト の Condition プロパティに使用)
- Feature 条件 (ISWiSuiteFeature オブジェクト の Condition プロパティに使用)
- Detection 条件 (ISWiSuitePackage オブジェクト の DetectionCondition プロパティに使用)
- 対象条件 (ISWiSuitePackage オブジェクト の EligibleCondition プロパティに使用)

ISWiSuiteCondition オブジェクトは、All、Any、または None 条件グループの Conditions コレクションにも存在することが可能です。

条件グループに条件を追加または削除するには、AddCondition、AddGroup、AddExtensionCondition、または DeleteCondition を呼び出します。

メンバー

テーブル 35 • ISWiSuiteCondition オブジェクトのメンバー

名前	種類	説明
AddCondition	メソッド	条件グループに条件を追加します。
AddExtensionCondition	メソッド	条件グループに拡張条件を追加します。
AddGroup	メソッド	条件グループの設定に All、Any、または None 演算子を追加します。
AddParameter	メソッド	拡張条件のパラメーターを追加します。
ChangeGroupType	メソッド	条件グループの All、Any、または None 演算子を変更します。
DeleteCondition	メソッド	条件オブジェクトを削除します。
DeleteParameter	メソッド	拡張条件のパラメーターを削除します。
ISWiSuiteConditionParameters	コレクション	条件ステートメントのすべての設定を含みます。
ISWiSuiteConditions	コレクション	条件グループの下にあるすべての条件ステートメントを含みます。
Move	メソッド	条件ツリー内の条件または条件グループを移動します。
Name	文字列プロパティ: 拡張条件の読み書き、その他すべての条件の読み取り専用	拡張条件の場合: DLL エントリ ポイントの名前の一部を取得または設定します。 他の条件の種類: 条件の名前を取得します。
リソース	読み書き文字列プロパティ	条件のソースを取得または設定します。拡張条件の場合、このプロパティの値は DLL ファイルの名前に設定されます。その他のビルトイン条件の場合、このプロパティの値は読み取り専用です。

テーブル 35 • ISWiSuiteCondition オブジェクトのメンバー (続き.)

名前	種類	説明
Type	読みとり専用整数プロパティ	条件の種類、条件グループ、または条件チェックを取得します。選択可能なオプションは以下のとおりです: • esctAll (0)—All 条件グループは、論理演算子 AND と同じ要領で動作します。All グループが True 評価されるためには、All グループに含まれる条件のすべてが True 評価される必要があります。 • esctAny (1)—Any 条件グループは、論理演算子 OR と同じ要領で動作します。Any グループに含まれる任意の条件が True 評価された場合、条件グループ全体が True 評価されます。Any グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が False 評価されます。 • esctNone (2)—None 条件グループは、論理演算子 NOR と同じ要領で動作します。None グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が True 評価されます。None 条件グループに含まれる任意の条件が True 評価された場合、条件グループ全体が False 評価されます。None 条件グループが 1 つの条件ステートメントで構成される場合、これは論理演算子 NOT と同じ要領で動作します。 • esctBuiltin (11)—ビルトイン条件チェックには、1 つ以上のサブ設定があります。ビルトイン条件の確認の例は、[プラットフォーム]、[ファイルの存在]、[有効なパッケージ]、および [機能の操作] があります。 • esctExtension (21)—ターゲット システム上で、他の種類の条件チェックによって評価されない要因をチェックする自作のカスタム条件を実装するために作成した C/C++ DLL を使う拡張条件。 グループ条件を識別するには、次を使用します: pCondition.Type < 10

次に適用 :

- ISWiSuiteExitCondition
- ISWiSuiteFeature
- ISWiSuitePackage

AddCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• この情報は、次のプロジェクト タイプに適用します:

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddCondition メソッドは、条件グループに条件を追加します。

条件グループに拡張条件を追加するには、AddExtensionCondition メソッド を使用します。

構文

AddCondition (ConditionCheckName As String) As ISWiSuiteCondition

パラメーター

テーブル 36 • AddCondition メソッド パラメーター

パラメーター	説明
ConditionCheckName	条件チェックの種類の名前を指定します。選択可能なオプションは以下のとおりです: • **Platform**—ターゲット システムをチェックして、オペレーティング システムのバージョンやアーキテクチャ、その他のプラットフォーム情報など、1 つまたは複数の特性を確認します。この種類の条件チェックに表示される各設定の説明は、「"プラットフォーム" 条件の設定」を参照してください。 • **FileExists**—ターゲット システムをチェックして、特定のファイルの有無を確認します。この種類の条件チェックに表示される各設定の説明は、「"ファイルの存在" 条件の設定」を参照してください。 • **FileValue**—ターゲット システムをチェックして、特定のファイルの特定の情報 (日付、バージョン番号、またはコンテンツ) を確認します。この種類の条件チェックに表示される各設定の説明は、「"ファイルの比較" 条件の設定」を参照してください。 • **RegistryExists**—ターゲット システムをチェックして、特定のレジストリ キーの存在、およびオプションで特定の値名の存在を確認します。この種類の条件チェックに表示される各設定の説明は、「"レジストリの存在" 条件の設定」を参照してください。 • **RegistryValue**—ターゲット システムをチェックして、特定のレジストリ エントリの特定の情報 (DWORD、QWORD、文字列、複数文字列、ファイル バージョン、または製品バージョン) を確認します。値名を指定しなかった場合、指定したキーのデフォルト値が比較時に使用されます。この種類の条件チェックに表示される各設定の説明は、「"レジストリの比較" 条件の設定」を参照してください。 • **PropertyValue**—特定のビルトイン アドバンスト UI またはスイート/アドバンスト UI プロパティ、あるいは [プロパティ マネージャー] ビューで定義されたプロパティの値をチェックできます。この種類の条件チェックに表示される各設定の説明は、「"プロパティの比較" 条件の設定」を参照してください。 • **MsiInstalled**—ターゲット システムをチェックして、特定の .msi パッケージによってインストールされた製品の存在を確認します。この種類の条件チェックに表示される各設定の説明は、「"MSI パッケージ" 条件の設定」を参照してください。 • **MsiRelated**—ターゲット システムをチェックして、アップデートする製品のバージョンが存在するかどうか、またはダウングレードしてはならない製品のより新しいバージョンが存在するかどうかを確認します。いずれの場合も、その存在を確認する製品は .msi パッケージによってインストールされます。この種類の条件チェックに表示される各設定の説明は、「"MSI アップグレード" 条件の設定」を参照してください。

テーブル 36・AddCondition メソッド パラメーター（続き）

パラメーター	説明
ConditionCheckName（続き）	• ParcelRef—アドバンスト UI またはスイート/アドバンスト UI インストール内の別のパッケージまたは機能がターゲット システム上でインストールの対象になるかどうかを確認します。たとえば、ベース パッケージのインストールが可能である場合のみパッチ パッケージのインストールが可能である場合、[有効なパッケージ] 条件を作成して、パッチ パッケージのインストールをベース パッケージのインストールが可能であるかどうかに関連付けます。この種類の条件チェックに表示される各設定の説明は、「"対象パッケージ" 条件の設定」を参照してください。 • IspInstalled—ターゲット システムをチェックして、特定の InstallScript パッケージによってインストールされた製品の存在を確認します。この種類の条件チェックに表示される各設定の説明は、「"InstallScript パッケージ" 条件の設定」を参照してください。 • Locale—ターゲット システムで、1 つまたは複数のロケール関連の設定に一致するものを検索します。この種類の条件チェックに表示される各設定の説明は、「"ロケール" 条件の設定」を参照してください。 • SuiteInstalled—アドバンスト UI またはスイート/アドバンスト UI インストールの特定のバージョンがターゲット システム上で既にインストールされているかどうかを確認します。この種類の条件チェックに表示される各設定の説明は、「"インストール済み" 条件の設定」を参照してください。 • AppxInstalled—ターゲット システムをチェックして、特定の Windows アプリの存在を確認します。この種類の条件チェックに表示される各設定の説明は、「「Windows アプリ パッケージの条件" の設定」を参照してください。 • ParcelAction—スイート/アドバンスト UI インストール内の特定のパッケージのターゲット状態をチェックします。この種類の条件は、[イベント] ビューまたは [パッケージ] ビューのイベントに関連付けられたアクションでのみ使用できます。この種類の条件チェックに表示される各設定の説明は、「"パッケージの操作" 条件の設定」を参照してください。 • FeatureAction—スイート/アドバンスト UI インストール内の特定の機能のターゲット状態をチェックします。この種類の条件は、[イベント] ビューまたは [パッケージ] ビューのイベントに関連付けられたアクションでのみ使用できます。この種類の条件チェックに表示される各設定の説明は、「"機能の操作" 条件の設定」を参照してください。 ***プロジェクト-*** *ParcelAction および FeatureAction タイプの条件は、スイート/アドバンスト UI プロジェクトで使用できます。*

次に適用：

- ISWiSuiteCondition

AddExtensionCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddExtensionCondition メソッドは、条件グループに拡張条件を追加します。

構文

AddExtensionCondition (ConditionCheckName As String, Resource as String) As ISWiSuiteCondition

パラメーター

テーブル 37 : AddExtensionCondition メソッドのパラメーター

パラメーター	説明
ConditionCheckName	条件の名前を指定します。これは、DLL エントリ ポイントの名前の一部です。
リソース	DLL ファイルの名前を指定します。別の種類のビルトイン条件用です。

次に適用 :

- ISWiSuiteCondition

AddGroup メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddGroup メソッドは、条件グループの設定に All、Any、または None 演算子を追加します。

構文

AddGroup (GroupType As ISWiSuiteConditionType) As ISWiSuiteCondition

パラメーター

テーブル 38 • AddGroup メソッドのパラメーター

パラメーター	説明
GroupType	条件グループの設定に追加する演算子を指定します。選択可能なオプションは以下のとおりです: • **esctAll (0)**—All 条件グループは、論理演算子 AND と同じ要領で動作します。All グループが True 評価されるためには、All グループに含まれる条件のすべてが True 評価される必要があります。 • **esctAny (1)**—Any 条件グループは、論理演算子 OR と同じ要領で動作します。Any グループに含まれる任意の条件が True 評価された場合、条件グループ全体が True 評価されます。Any グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が False 評価されます。 • **esctNone (2)**—None 条件グループは、論理演算子 NOR と同じ要領で動作します。None グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が True 評価されます。None 条件グループに含まれる任意の条件が True 評価された場合、条件グループ全体が False 評価されます。None 条件グループが 1 つの条件ステートメントで構成される場合、これは論理演算子 NOT と同じ要領で動作します。

次に適用 :

- ISWiSuiteCondition

AddParameter メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddParameter メソッドは、条件グループに拡張条件を追加します。

構文

AddParameter (Name As String) As ISWiSuiteConditionParameter

パラメーター

テーブル 39 - AddParameter メソッドのパラメーター

パラメーター	説明
Name	拡張条件に追加するパラメーターの名前を指定します。

次に適用：

- ISWiSuiteCondition

ChangeGroupType メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション- *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ChangeGroupType メソッドは、条件グループの All、Any、または None 演算子を変更します。

構文

ChangeGroupType (GroupType As ISWiSuiteConditionType)

パラメーター

テーブル 40 • ChangeGroupType メソッドのパラメーター

パラメーター	説明
GroupType	条件グループに使用する演算子を指定します。選択可能なオプションは以下のとおりです: • **esctAll (0)**—All 条件グループは、論理演算子 AND と同じ要領で動作します。All グループが True 評価されるためには、All グループに含まれる条件のすべてが True 評価される必要があります。 • **esctAny (1)**—Any 条件グループは、論理演算子 OR と同じ要領で動作します。Any グループに含まれる任意の条件が True 評価された場合、条件グループ全体が True 評価されます。Any グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が False 評価されます。 • **esctNone (2)**—None 条件グループは、論理演算子 NOR と同じ要領で動作します。None グループに含まれる条件のいずれも True 評価されなかった場合、条件グループ全体が True 評価されます。None 条件グループに含まれる任意の条件が True 評価された場合、条件グループ全体が False 評価されます。None 条件グループが 1 つの条件ステートメントで構成される場合、これは論理演算子 NOT と同じ要領で動作します。

次に適用：

- ISWiSuiteCondition

DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteCondition メソッドは条件オブジェクト (ビルトイン条件、拡張条件、または条件グループ) を削除します。

構文

DeleteCondition (pCondition As ISWiSuiteCondition)

パラメーター

テーブル 41 • DeleteCondition メソッド パラメーター

パラメーター	説明
pCondition	削除する条件の ISWiSuiteCondition オブジェクトを指定します。

次に適用：

- ISWiSuiteCondition

DeleteParameter メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteParameter メソッドは、条件ステートメントから拡張条件のパラメーターを削除します。

構文

DeleteParameter (pParam As ISWiSuiteConditionParameter)

パラメーター

テーブル 42 • DeleteParameter メソッドのパラメーター

パラメーター	説明
pParam	拡張条件の条件ステートメントから削除する ISWiSuiteConditionParameter オブジェクトを指定します。

次に適用：

- ISWiSuiteCondition

Move メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

Move メソッドは、条件ツリー内の条件または条件グループを移動します。

構文

Move (Direction As ISWiDirection)

パラメーター

テーブル 43 : Move メソッドのパラメーター

パラメーター	説明
Direction	条件ツリー内の条件または条件グループを移動する方向を指定します。選択可能なオプションは以下のとおりです: • **edUp (0)**—条件ツリー内の項目を上に移動します。 • **edDown (1)**—条件ツリー内の項目を下に移動します。 • **edLeft (2)**—条件ツーリー内の項目を左に移動します。 • **edRight (3)**—条件ツーリー内の項目を右に移動します。

次に適用 :

- ISWiSuiteCondition

ISWiSuiteConditionParameter オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

ISWiSuiteConditionParameter オブジェクトは、条件チェックのパラメーターを表します。これは、条件ステートメントの条件サブ設定で構成された値に相当します。ISWiSuiteConditionParameters コレクションの中の項目を指定して、パラメーター値を取得できます。

ISWiSuiteConditionParameters コレクションには、ビルトイン条件のすべての有効なパラメーターが含まれています。拡張条件にパラメーターを追加または削除するには、ISWiSuiteCondition.AddParameter または ISWiSuiteCondition.DeleteParameter を呼び出してください。

メンバー

テーブル 44 • ISWiSuiteConditionParameter オブジェクトのメンバー

名前	種類	説明
コンテンツ	読み書き文字列プロパティ	パラメーターの値を取得または設定します。
Name	読み書き文字列プロパティ	パラメーターの名前を取得または設定します。 Name プロパティのカスタマイズが必要なのは、プロジェクトで拡張条件を構成する場合のみです。

次に適用：

- ISWiSuiteCondition

ISWiSuiteDFLFilter オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

ISWiSuiteDFLFilter オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージが必要とする追加ファイルを含むダイナミック リンクのフィルターを表します。

メンバー

テーブル 45 • ISWiSuiteDFLFilter オブジェクトのメンバー

名前	種類	説明
CanMove	メソッド	フィルターがフィルター基準の中を上下に移動できるかどうかを示します。結果はブール値で返されます。
ISWiSuiteDFLFilters	コレクション	パッケージが必要とする追加ファイルのダイナミック リンクと関連付けられているすべてのフィルターを含みます。
MoveDown	メソッド	フィルターを次のフィルターの下に移動します。
MoveUp	メソッド	フィルターを前のフィルターの上に移動します。
パターン	読み書き文字列プロパティ	ダイナミック リンクに含める、または除外するファイルを含むダイナミック フォルダーの名前を指定するか、ダイナミック リンクに含めるダイナミック ファイルの名前を指定します。 ワイルドカード文字の指定には、アスタリスク (*) を使用します。たとえば、すべての .htm ファイルを含める場合、**"含めるファイル"** 設定で ***.htm** と入力します。すべてのファイルを含める場合、***.*** と入力できます。 パスでルート フォルダー (パッケージ ファイルを含むフォルダー) を参照する場合、ドットと円記号を使用します: **.¥** たとえば、**MyDirectory** という名のサブフォルダー内 (**MyDirectory** は、パッケージを含むフォルダーのサブフォルダーです) にある **ReadMe.txt** ファイルを参照する場合、次のようなフィルターを使うことができます: **.¥MyDirectory¥ReadMe.txt** InstallShield で、リリースに含める、または、リリースから除外するファイルおよびフォルダーがどう判別されるかなどを含む、さらに詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトで、ダイナミック リンク フォルダーのフィルターを定義する」を参照してください。

次に適用 :

- ISWiSuiteDynamicFileLink

CanMove メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

CanMove メソッドは、フィルターがフィルター基準の中を上下に移動できるかどうかを示します。結果はブール値で返されます。

構文

CanMove (Direction As ISWiDirection)

パラメーター

テーブル 46 ・ CanMove メソッドのパラメーター

パラメーター	説明
Direction	次の ISWiDirection 値のひとつを指定して、フィルター基準のリスト内でこの方向にフィルターを移動可能かどうかを確認します。 • **edUp (0)**—フィルター基準のリスト内でフィルターを上に移動可能かどうかを検証します。 • **edDown (1)**—フィルター基準のリスト内でフィルターを下に移動可能かどうかを検証します。

次に適用：

- ISWiSuiteDFLFilter

MoveDown Method メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

MoveDown メソッドは、フィルターを次のフィルターの下に移動します。

構文

MoveDown ()

次に適用：

- ISWiSuiteDFLFilter

MoveUp Method メソッド（アドバンスト UI およびスイート/アドバンスト UI）

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

MoveUp メソッドは、フィルターを前のフィルターの上に移動します。

構文

MoveUp ()

次に適用：

- ISWiSuiteDFLFilter

ISWiSuiteDynamicFileLink オブジェクト（アドバンスト UI およびスイート/アドバンスト UI）

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteDynamicFileLink オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージが必要とする追加ファイルのダイナミック リンクを表します。

ダイナミック リンクを追加または削除するには、AddDynamicFileLink または DeleteDynamicFileLink を呼び出します。

メンバー

テーブル 47 • ISWiSuiteDynamicFileLink オブジェクトのメンバー

名前	種類	説明
AddFilter	メソッド	アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージが必要とする追加ファイルのダイナミック リンクにフィルターを追加します。
DeleteFilter	メソッド	ダイナミック リンクからフィルターを削除します。
ISWiSuiteDFLFilters	コレクション	パッケージが必要とする追加ファイルのダイナミック リンクと関連付けられているすべてのフィルターを含みます。
ソース	読み書き文字列プロパティ	アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含めるパッケージを含むソース フォルダーのパスを取得または設定します。これはまた、パッケージに動的に含まれているファイルのソースでもあります。 場所をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。

次に適用：

- ISWiSuiteFolder

AddFilter メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddFilter メソッドは、パッケージが必要とする追加ファイルのダイナミック リンクにフィルターを追加します。

構文

AddFilter (FilterType As ISWiSuiteDFLFilterType) As ISWiSuiteDFLFilter

パラメーター

テーブル 48 • AddFilter メソッドのパラメーター

パラメーター	説明
FilterType	ダイナミック リンクに追加するフィルターの種類を指定します。選択可能なオプションは以下のとおりです: • **esdfIncFolder (301)**—指定されたフィルターと一致するフォルダーを含みます。 • **esdfExcFolder (302)**—指定されたフィルターと一致するフォルダーを除外します。 • **esdfIncFile (303)**—指定されたフィルターと一致するファイルを含みます。 • **esdfExcFile (304)**—指定されたフィルターと一致するファイルを除外します。

次に適用：

- ISWiSuiteDynamicFileLink

DeleteFilter メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteFilter メソッドは、パッケージが必要とする追加ファイルのダイナミック リンクからフィルターを削除します。

構文

DeleteFilter (Filter As ISWiSuiteDFLFilter)

パラメーター

テーブル 49 • DeleteFilter メソッドのパラメーター

パラメーター	説明
フィルター	削除する ISWiSuiteDFLFilter オブジェクトを指定します。

次に適用：

- ISWiSuiteDynamicFileLink

ISWiSuiteEvent オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteEvent オブジェクトは、スイート/アドバンスト UI 実行時イベントを表します。指定のイベントにスケジュールされているアクションが、実行時に特定の条件下で実行されます。

メンバー

テーブル 50 • ISWiSuiteEvent オブジェクトのメンバー

名前	種類	説明
AddSuiteActionRef	メソッド	実行時イベント中にスイート/アドバンスト UI アクションをスケジュールします。
DeleteSuiteActionRef	メソッド	実行時イベントからアクションを削除します。
ISWiSuiteActionRefs	コレクション	 ***プロジェクト•*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション•*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* パッケージ内で特定のイベントにスケジュールされているすべてのスイート/アドバンスト UI アクションを含みます。

次に適用：

- ISWiProject

AddSuiteActionRef メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuiteActionRef メソッドは、実行時イベント中にスイート/アドバンスト UI アクションをスケジュールします。必要であれば、イベント中にアクションを複数回実行するようにスケジュールできます。

構文

AddSuiteActionRef (Action As ISWiSuiteAction)

パラメーター

テーブル 51 • AddSuiteActionRef メソッドのパラメーター

パラメーター	説明
アクション	このイベントでスケジュールするアクションを指定します。

次に適用：

- ISWiSuiteEvent

DeleteSuiteActionRef メソッド（アドバンスト UI およびスイート/アドバンスト UI）

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSuiteActionRef メソッドは、実行時イベント中からスイート/アドバンスト UI アクションを削除します。

構文

DeleteSuiteActionRef (ActionRef As ISWiSuiteActionRef)

パラメーター

テーブル 52 • DeleteSuiteActionRef メソッドのパラメーター

パラメーター	説明
ActionRef	イベントから削除する ISWiSuiteActionRef オブジェクトとしてアクションを指定します。

次に適用：

- ISWiSuiteEvent

ISWiSuiteExitCondition オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteExitCondition オブジェクトは、スイート/アドバンスト UI プロジェクトの終了条件を表します。

終了条件を追加または削除するには、ISWiProject.AddExitCondition または ISWiProject.DeleteExitCondition を呼び出します。

メンバー

テーブル 53 : ISWiSuiteExitCondition オブジェクトのメンバー

名前	種類	説明
Message	読み書き文字列プロパティ	このメッセージに定義された条件が True となった時、アドバンスト UI またはスイート/アドバンスト UI インストールが実行時に表示するエラー メッセージを取得または設定します。
条件	読みとり専用オブジェクト プロパティ	このプロパティは、ISWiSuiteCondition オブジェクトにアクセスします。 アドバンスト UI またはスイート/アドバンスト UI インストールが実行時に終了条件で評価するルートレベルの条件グループ (Any、All、または Not) を取得します。 終了条件には常に条件があるため、条件を追加する必要はありません。

次に適用 :

- ISWiProject

ISWiSuiteFeatureオブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteFeature オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトでルートレベルの機能またはサブ機能を表します。ISWiSuiteFeatures コレクション内の項目を指定して、機能を取得できます。

メンバー

テーブル 54 • ISWiSuiteFeature オブジェクトのメンバー

名前	種類	説明
AddCondition	メソッド	条件を持たない機能に条件を追加します。
AddSuiteSubFeature	メソッド	既存の機能にサブ機能を追加します。
AllowSelectionChange	読み書きブール値プロパティ	機能の "UI 選択の変更を許可する" 設定の値を取得または設定します。この設定は、エンド ユーザーが InstallationFeatures ページに表示されている機能を選択または選択解除できるようにするかどうかを指定します。
AttachPackage	メソッド	プロジェクトのパッケージを機能に関連付けます。
条件	読みとり専用オブジェクトプロパティ	このプロパティは、ISWiSuiteCondition オブジェクトにアクセスします。 条件が機能に関連付けられている場合、機能の条件を取得します。
Cost	読み書きプロパティ	機能がターゲット システム上で必要とする空き容量をバイト単位で取得または設定します。
DeleteCondition	メソッド	機能から条件を削除します。
DeleteSubSuiteFeature	メソッド	機能からサブ機能を削除します。
説明	読み書き文字列プロパティ	この機能の説明を、取得または設定します。
DisplayName	読み書き文字列プロパティ	機能の名前を取得または設定します。

テーブル 54 • ISWiSuiteFeature オブジェクトのメンバー (続き.)

名前	種類	説明
FollowParentState	読み書きブール値プロパティ	このプロパティはサブ機能で使用できますが、ルート レベルの機能では使用できません。 サブ機能の "親の状態に従う" 設定の値を取得または設定します。この設定は、親機能の状態によって サブ機能の状態を決定するかどうかを示します。次の値が可能です: • True—選択されたサブ機能は、親機能がインストールされているかどうかに従ってインストールされます。 • False—選択されたサブ機能の親機能がインストールされている場合、選択されたサブ機能はインストールまたはインストールされていない状態のどちらも可能です。デフォルトでは、これが設定されています。選択されたサブ機能の親機能がインストールされていない場合、選択されたサブ機能をインストールすることはできません。 この設定は、InstallationFeatures ウィザード ページ上でサブ機能を非表示の状態にするときに使用できますが、親機能は表示された状態です。
ISWiSuiteFeatures	コレクション	機能に割り当てられているすべてのサブ機能を含みます。
ISWiSuitePackages	コレクション	機能に関連付けられているすべてのパッケージを含みます。
Name	読み書き文字列プロパティ	コレクション内の機能の名前を取得または設定します。
ReleaseFlags	読み書き文字列プロパティ	この機能に関連付けられたリリース フラグを取得または設定します。複数のフラグは、コンマで区切ります。 ISWiRelease の ReleaseFlags プロパティで、プロジェクトの機能をフィルターする際に使用するリリース フラグを指定できます。
RemovePackage	メソッド	機能からパッケージを削除します。
Visible	読み書きブール値プロパティ	機能の "可視" 設定を取得または設定します。この設定はインストール中に InstallationFeatures ウィザード ページで機能を可視にするかどうかを指定します。

例

次の VBScript 行は、機能についての情報を表示します。機能にはサブ機能が含まれているため、スクリプトが再帰的に実装されます。

```
Sub WriteFeature(indent, f)
    out.WriteLine indent & f.DisplayName & " (" & f.Name & ")"
    out.WriteLine indent & "  リリース フラグ:" & f.ReleaseFlags
    If f.Visible Then out.WriteLine indent & "  表示"
    If Not f.AllowSelectionChange Then out.WriteLine indent & "  無効"
```

```
    If f.FollowParentState Then out.WriteLine indent & "  親に従う"
    If Not f.Condition Is Nothing Then
        out.WriteLine indent & "  条件:"
        WriteCondition indent & "  ", f.Condition
    End If
    out.WriteLine indent & "  Packages:"
    For Each p in f.Packages
        out.WriteLine indent & "    " & IDS(p.DisplayName)
    次へ
    out.WriteLine indent & "  Child Features:"
    For Each sf in f.SubFeatures
        WriteFeature indent & "  ", (sf)
    次へ
End Sub
```

次に適用：

- ISWiProject
- ISWiSuiteFeature

AddCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddCondition メソッドは条件を持たない機能に条件を追加します。

構文

```
AddCondition () As ISWiSuiteCondition
```

パラメーター

AddCondition メソッドにはパラメーターがありません。

次に適用：

- ISWiSuiteFeature

AddSuiteSubFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuiteSubFeature メソッドはプロジェクト内の既存の機能にサブ機能を追加します。

構文

AddSuiteSubFeature (InternalName As String) As ISWiSuiteFeature

パラメーター

テーブル 55 : AddSuiteSubFeature メソッドのパラメーター

パラメーター	説明
InternalName	既存する機能に追加するサブ機能内部名 ([機能] ビューの [機能] ペインで表示される) を指定します。

次に適用 :

- ISWiSuiteFeature

AttachPackage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AttachPackage メソッドはパッケージをプロジェクト内の機能に関連付けます。

構文

AttachPackage (Package As ISWiSuitePackage)

パラメーター

テーブル 56 • AttachPackage メソッドのパラメーター

パラメーター	説明
パッケージ	機能と関連付ける ISWiSuitePackage オブジェクトを指定します。

次に適用：

- ISWiSuiteFeature

DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteCondition メソッドは、機能から条件を削除します。

構文

DeleteCondition ()

パラメーター

DeleteCondition メソッドにはパラメーターがありません。

次に適用：

- ISWiSuiteFeature

DeleteSuiteSubFeature メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*

- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteSuiteSubFeature メソッドはプロジェクト内の機能からサブ機能を削除します。

構文

DeleteSuiteSubFeature (SubFeature As ISWiSuiteFeature)

パラメーター

テーブル 57 : DeleteSuiteSubFeature メソッドのパラメーター

パラメーター	説明
SubFeature	機能から削除したい ISWiSuiteFeature オブジェクトを指定します。

次に適用 :

- ISWiSuiteFeature

RemovePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

RemovePackage メソッドはプロジェクト内の機能からパッケージを削除します。

構文

RemovePackage (Package As ISWiSuitePackage)

パラメーター

テーブル 58 • RemovePackage メソッドのパラメーター

パラメーター	説明
パッケージ	機能から削除したい ISWiSuitePackage オブジェクトを指定します。

次に適用：

- ISWiSuiteFeature

ISWiSuiteFile オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteFile オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに含まれるファイルを表します。ファイルまたはそのソース パスを変更できます。ファイルの別のプロパティは読み取り専用です。

メンバー

テーブル 59 • ISWiSuiteFile オブジェクトのメンバー

名前	種類	説明
FullPath	読み取り専用文字列プロパティ	ファイルの完全パスを取得します。
LinkPath	読み書き文字列プロパティ	ファイルへのソース パスを取得または設定します。 パスをハードコード化する代わりに、[パス変数] ビューまたは ISWiPathVariables コレクションの一部で定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。
Modified	読み取り専用文字列プロパティ	ファイルの最終更新日を取得します。
Name	読み書き文字列プロパティ	パッケージ内の機能の名前を取得または設定します。
Size	読みとり専用整数プロパティ	ファイルのサイズを取得します。
Version	読み取り専用文字列プロパティ	ファイルのバージョン番号を取得します。

次に適用：

- ISWiSuiteFolder

ISWiSuiteFolder オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロ*

ジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。

ISWiSuiteFolder オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに含まれるファイルのフォルダーを表します。

メンバー

テーブル 60 • ISWiSuiteFolder オブジェクトのメンバー

名前	種類	説明
AddDynamicFileLink	メソッド	アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージが必要とする追加ファイルのダイナミック リンクを追加します。
AddFile	メソッド	パッケージ内のフォルダーにファイルを追加します。
AddFolder	メソッド	パッケージ内のフォルダーにサブフォルダーを追加します。
DeleteDynamicFileLink	メソッド	パッケージが必要とする追加ファイルのダイナミック リンクを削除します。
DeleteFile	メソッド	パッケージ内のフォルダーからファイルを削除します。
DeleteFolder	メソッド	パッケージ内のフォルダーからサブフォルダーを削除します。
ISWiSuiteDynamicFileLinks	コレクション	アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージのフォルダーに含まれるすべてのダイナミック ファイル リンクを含みます。
ISWiSuiteFiles	コレクション	アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージのフォルダーに含まれるすべてのファイルを含みます。
ISWiSuiteFiles	コレクション	フォルダー内のすべてのファイルを含みます。
ISWiSuiteFolders	コレクション	現在のフォルダー内のすべてのサブフォルダーを含みます。
Name	読み書き文字列プロパティ	パッケージ内のフォルダーの名前を取得または設定します。

次に適用：

- ISWiSuitePackage

AddDynamicFileLink メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddDynamicFileLink メソッドは、パッケージが必要とする追加ファイルのダイナミック リンクを追加します。

構文

AddDynamicFileLink (SourceFolder As String)

パラメーター

テーブル 61 ・ Method メソッドのパラメーター

パラメーター	説明
SourceFolder	パッケージに動的に含まれているソース フォルダーへのパスを指定します。 場所をハードコード化する代わりに、[パス変数] ビューで定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。

次に適用 :

- ISWiSuiteFolder

AddFile メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddFile メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに含まれるフォルダーにファイルを追加します。

構文

AddFile (Name As String) As ISWiSuiteFile

パラメーター

テーブル 62 ・ AddFile メソッド パラメーター

パラメーター	説明
Name	パッケージ内のフォルダーに追加するファイルの名前を指定します。

次に適用：

- ISWiSuiteFolder

AddFolder メソッド (アドバンスト UI およびスイート/アドバンスト UI)

***プロジェクト・** この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

***エディション・** アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddFolder メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに含まれるフォルダーにサブフォルダーを追加します。

構文

AddFolder (Name As String) As ISWiSuiteFolder

パラメーター

テーブル 63 ・ AddFile メソッド パラメーター

パラメーター	説明
Name	パッケージ内のフォルダーに追加するサブフォルダーの名前を指定します。

次に適用：

- ISWiSuiteFolder

DeleteDynamicFileLink メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteDynamicFileLink メソッドは、パッケージが必要とする追加ファイルのダイナミック リンクを削除します。

構文

DeleteDynamicFolderLink (DynamicFileLink As ISWiSuiteDynamicFileLink)

パラメーター

テーブル 64 : DeleteDynamicFolderLink メソッドのパラメーター

パラメーター	説明
DynamicFileLink	削除する ISWiSuiteDynamicFileLink オブジェクトを指定します。

次に適用 :

- ISWiSuiteFolder

DeleteFile メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteFile メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに含まれるフォルダーからファイルを削除します。

構文

DeleteFile (File as ISWiSuiteFile)

パラメーター

テーブル 65 • DeleteFile メソッド パラメーター

パラメーター	説明
File	削除する ISWiSuiteFile オブジェクトを指定します。

次に適用：

- ISWiSuiteFolder

DeleteFolder メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteFolder メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに含まれるフォルダーからサブフォルダーを削除します。

構文

DeleteFolder (Name As ISWiSuiteFolder)

パラメーター

テーブル 66 • DeleteFile メソッド パラメーター

パラメーター	説明
Name	削除する ISWiSuiteFolder オブジェクトを指定します。

次に適用：

- ISWiSuiteFolder

ISWiSuiteOperation オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteOperation オブジェクトは、パッケージ上の操作を表します。

一部の ISWiSuiteOperation 項目は、すべての種類のアドバンスト UI またはスイート/アドバンスト UI パッケージ ファイルに適用できませんので注意してください。

メンバー

テーブル 67 • ISWiSuiteOperation オブジェクトのメンバー

名前	種類	パッケージ ファイルの種類	説明
CmdLine	読み書き文字列プロパティ	.msi、.msp、.exe、InstallScript、Basic MSI project、InstallScript プロジェクト	適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがユーザー インターフェイスで実行するときにターゲット ファイルを起動するのに使用するコマンドラインを取得または設定します。このプロパティには、ファイル名を含めないでください。 このプロパティの値には、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 このプロパティで使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。
Enable	読み書きブール値プロパティ	サポートは様々	パッケージの操作が有効かどうかを示す値を取得または設定します。

テーブル 67 • ISWiSuiteOperation オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
ExitBehavior	読み書き整数プロパティ	.exe	パッケージが実行された後でもターゲット システム上で検出条件が満たされない場合、以下のいずれかが True である可能性があります。 • 実行可能ファイル パッケージの実行が失敗した。 • パッケージに指定された検出条件が不正確である。 たとえば、ターゲット システムに特定のファイルが存在しない場合に、検出条件が実行可能ファイルを起動する必要があることを示す可能性があります。アドバンスト UI またはスイート/アドバンスト UI インストールによって実行可能アフィル パッケージが実行された後でもファイルが不足している場合、条件に誤りがある可能性があります。 ExitBehavio は、プロパティ1 つ以上の検出条件が、パッケージを起動する必要があることを示し続けている場合に発生する動作を示します。選択可能なオプションは以下のとおりです: • epseDetectPrompt (0)—セットアップを続行するかどうかを確認します。 アドバンスト UI またはスイート/アドバンスト UI インストールは、エンド ユーザーに対してアドバンスト UI またはスイート/アドバンスト UI インストールを続行するかどうかを指定するようにプロンプトするメッセージ ボックスを表示します。 • epseDetectAbort (1)—セットアップを中止します。 アドバンスト UI またはスイート/アドバンスト UI インストールは、後に続く処理を行わずに終了します。 • epseDetectIgnore (2)—セットアップを続行します。 アドバンスト UI またはスイート/アドバンスト UI インストールが、基本的に実行可能ファイルで満たされなかった条件を無視して (必要な場合に) インストール内の次のパッケージに進みます。

テーブル 67 • ISWiSuiteOperation オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
ForceApplicationShutdown	読み書きブール値プロパティ	.appx	インストール実行時に選択されたサイドロード アプリケーション パッケージ (またはこのパッケージに依存する任意のパッケージ) が使用中の場合、登録処理を行うためにパッケージに関連付けられたプロセスを強制的にシャットダウンすることを要求できます。このプロパティでは、この状況下でアプリケーションをシャットダウンするかどうかを指定します。
RebootCode	読み書き文字列プロパティ	.exe	選択されたパッケージが製品のインストール後にターゲットマシンの再起動を要求する場合、このプロパティは 1 つ以上のリターン コードを取得または設定します。複数のリターンコードが存在する場合、各コードはコンマで区切ります。 再起動が必要なアドバンスト UI またはスイート/アドバンスト UI パッケージの詳細は、「アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動」を参照してください。
RebootRequest	読み書き文字列プロパティ	.msi、.msp、.exe、InstallScript、基本の MSI プロジェクト InstallScript プロジェクト	「アドバンスト UI またはスイート/アドバンスト UI パッケージにおけるターゲット システムの再起動」でも説明されているように、アドバンスト UI またはスイート/アドバンスト UI パッケージをインストールするためにターゲット マシンを再起動する必要がある場合があります。 RebootRequest プロパティは、パッケージが、ターゲット システムの再起動を必要とする場合の動作を指定します。選択可能なオプションは以下のとおりです: • **epsrDetectExit (1)**—マシンの再起動を許可します。 • **epsrDetectReboot (2)**—プロンプトしてから、終了またはマシンを再起動します。 • **epsrDetectIgnore (4)**—再起動要求を無視します。 • **epsrAlwaysRebootPrompt (8)**—常にプロンプトしてから、終了またはマシンを再起動します。 • **epsrAlwaysRebootSilent (16)**—常にマシンを再起動します。 • **epsrDetectDelay (32)**—プロンプトを遅延してから、終了またはマシンを再起動します。

テーブル 67 • ISWiSuiteOperation オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
サイレント	読み書き文字列プロパティ	.msi、.msp、.exe、InstallScript、Basic MSI project、InstallScript プロジェクト	適切な場合、アドバンスト UI またはスイート/アドバンスト UI インストールがサイレント (ユーザー インターフェイスなし) で実行するとき、ターゲット ファイルを起動するのに使用するコマンドラインを取得または設定します。このプロパティには、ファイル名を含めないでください。 このプロパティの値には、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。 このプロパティで使用できる種類を含む詳細については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」を参照してください。
Target	読み書き文字列プロパティ	サポートは様々	アドバンスト UI またはスイート/アドバンスト UI インストールが呼び出すパッケージに含まれるファイルを取得または設定します。 このプロパティは、次の状況下で使用できます: • Type プロパティが esotInstall (1) で、パッケージ ファイルの種類が .msi、.msp、.exe、および InstallScript パッケージの場合 • Type プロパティが esotRemove (2)、esotRepair (3)、または esotModify (4) で、パッケージ ファイルの種類が .exe パッケージの場合

テーブル 67 • ISWiSuiteOperation オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
Type	読み取り専用文字列プロパティ	サポートは様々	操作の種類を取得します。選択可能なオプションは以下のとおりです: • esotInstall (1)—パッケージは初回インストールとして起動されます。 この処理は、.msi、.msp、.exe、.appx、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクトのパッケージ ファイルで使用できます。 • esotRemove (2)—パッケージがインストールした製品が削除されます。 この操作はほとんどの種類のパッケージで使用できます (.msi、.exe、.appx、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。 • esotRepair (3)—パッケージがインストールした製品を修正します。 この操作はほとんどの種類のパッケージで使用できます (.msi、.exe、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。 • esotModify (4)—パッケージがメンテナンス モードで起動され、エンド ユーザーが製品をインストール、削除、または変更することができます。 この操作はほとんどの種類のパッケージで使用できます (.msi、.msp、.exe、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。

次に適用 :

- ISWiSuitePackage

ISWiSuitePackage オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: この情報は、次のプロジェクト タイプに適用します:

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuitePackage オブジェクトは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるパッケージまたは Windows Installer トランザクションを表します。

パッケージおよびトランザクションを追加または削除するには、ISWiProject.AddSuitePackage または ISWiProject.DeleteSuitePackage を呼び出します。

重要: *一部の ISWiSuitePackage 項目は、すべての種類のアドバンスト UI またはスイート/アドバンスト UI パッケージ ファイルに適用できません。PackageType プロパティを使って、パッケージの種類を判別します。もうひとつの ISWiSuitePackage プロパティが空白値を戻した場合、そのプロパティは指定されたパッケージの種類には使用できません。*

InstallShield Professional Edition で使用できるアドバンスト UI プロジェクト タイプでは、ほとんどの種類のパッケージ ファイルがサポートされていません。.msi、.msp、および InstallScript パッケージのみがサポートされています。

メンバー

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー

名前	種類	パッケージ ファイルの 種類	説明
AddDetectCondition	メソッド	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに検出条件を追加します。
AddEligibleCondition	メソッド	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに対象条件を追加します。
AddSuitePackage	メソッド	.msi、 .msp	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* Windows Installer トランザクションに .msi または .msp パッケージを追加します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
CachePath	読み書き文字列プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	キャッシュされたパッケージおよびその他のパッケージ ファイルを保存するエンド ユーザーのシステム上の場所を指定します。ハード コード化された値、たとえば **C:¥CachedFiles** を入力できますが、パスにはインストール先プロパティーを使用することが推奨されます。次は、デフォルトの値です: **[LocalAppDataFolder]Downloaded Installations** [リリース] ビューの Setup.exe タブでは、ターゲット システム上で実行されるパッケージで、キャッシュ パスが定義されている場合に、ターゲット システム上にパッケージをキャッシュするかどうかを指定できます。非圧縮のリリースをビルドする場合は、ターゲット システム上でパッケージをキャッシュしないことをお勧めします。 このプロパティの値には、プロパティ名、環境変数リファレンス、その他の特殊文字列を含む 1 つ以上の形式化された式を使用することができます。実行時、インストールはこれらの式の値を拡張します。これらの式で使用できる構文については、「アドバンスト UI およびスイート/アドバンスト UI インストールが実行時に解決する形式化された式を使用する」を参照してください。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
CerFile	読み書き文字列プロパティ	.appx	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* サイドロード パッケージで使用する .cer ファイルを取得または設定します。 Windows アプリ パッケージをサポートする Windows のバージョンは、パッケージのソースを信頼しない限り、サイドロード Windows パッケージをインストールしません。たとえば Visual Studio で生成されたカスタム証明書を使用する場合、スイート/アドバンスト UI インストールではターゲット システムの証明書ストアにカスタム証明書を追加することで、Windows がパッケージのソースを信頼するように設定できます。
DeleteCondition	メソッド	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	パッケージから検出または対象条件を削除します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
DetectCondition	読みとり専用オブジェクト プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	このプロパティは、ISWiSuiteCondtion オブジェクトにアクセスします。 パッケージがターゲット システム上に既にインストール済みであるかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する条件を取得または設定します。
DisplayName	読み書き文字列プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	実行時、パッケージに表示する名前を取得または設定します。
DynamicParcelProductConfig	読み書き文字列プロパティ	基本の MSI プロジェクト	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* 選択されたパッケージに関連付ける InstallShield プロジェクトのリリースを含む製品構成へのパスを取得または選択します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージファイルの種類	説明
DynamicParcelRelease	読み書き文字列プロパティ	基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* 選択されたパッケージと関連付ける InstallShield プロジェクトのリリースを取得または設定します。
DynamicParcelSourcePath	読み書き文字列プロパティ	基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* スイート/アドバンスト UI プロジェクトに含める基本の MSI または InstallScript プロジェクト (.ism) のソースパスを取得または設定します。
Elevate	読み書きブール値プロパティ	.msi、.msp、.exe、InstallScript、Web 配置、基本の MSI プロジェクト、InstallScript プロジェクト	[パッケージ] ビューの "昇格された権限が必要" 設定の値を取得または設定します。この設定は、パッケージが Windows Vista 以降および Windows Server 2008 以降のシステム上で昇格された権限を必要とするかどうかを示します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
EligibleCondition	読みとり専用オブジェクト プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	このプロパティは、ISWiSuiteCondtion オブジェクトにアクセスします。 ターゲット システムで実行するパッケージに必要な要件を満たしているかどうかを判断するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する条件を取得または指定します。
IsPrimary	読み書きブール値プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *スイート/アドバンスト UI プロジェクトでは、このプロパティを編集することができます。アドバンスト UI プロジェクトでは、プライマリ パッケージの数が 1 つのみサポートされているため、この設定は読み取り専用になっています。* [パッケージ] ビューでパッケージの "パッケージの種類" 設定を取得または設定します。この設定は、このパッケージがターゲット システム上に存在するかどうかによって、スイート/アドバンスト UI インストールを初回インストール モードまたはメンテナンス モードで実行するかどうかに影響するかどうかを示します。 このプロパティで選択できるオプションは、次のとおりです: • **True**—選択されたパッケージはプライマリ パッケージで、アドバンスト UI またはスイート/アドバンスト UI インストールのメインの部分です。 実行時、ターゲット システム上からインストールのプライマリ パッケージのすべてが不足している場合、インストールは初回インストールとして実行します。ターゲット システム上にスイート インストールのいずれかのプライマリ パッケージが存在する場合、インストールはメンテナンス モードで実行します。 • **False**—選択されたパッケージは依存関係パッケージで、アドバンスト UI またはスイート/アドバンスト UI インストールをどのモードで実行するかを決定する要因として見なすことはできません。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
ISWiSuiteActionRefs	コレクション	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* 特定のパッケージ イベントにスケジュールされているすべてのスイート/アドバンスト UI アクションを含みます。
ISWiSuiteEvents	コレクション	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* プロジェクトに含まれるパッケージに関連付けられているすべてのイベントを含みます。
ISWiSuiteFolders	コレクション	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	プロジェクトのパッケージに含まれるすべてのフォルダーが含まれます。
ISWiSuitePackages	コレクション	transaction	Windows Installer トランザクション内のすべてのパッケージ (ISWiSuitePackage オブジェクトとして) が含まれます。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
LoggingCmdLine	読み書き 文字列プ ロパティ	.exe、 InstallScript、 InstallScript プロジェク ト	このプロパティは、LoggingEnable プロパティの値が True の場合に使用できます。 [パッケージ] ビューの "ログのコマンドライン" 設定 の値を取得または設定します。この設定は、アドバン スト UI またはスイート/アドバンスト UI インストール で、ログ機能を有効にするために .exe パッケージに渡 すコマンドラインを指定します。サポートされている 適切なフラグを含めます。構成中の .exe パッケージで ログ機能がサポートされている場合、ファイルが作成 されるディレクトリのパスの代わりに、アドバンスト UI またはスイート/アドバンスト UI のプロパティ ISLogDir を参照するパスを角かっこで囲んで含めます。 たとえば、InstallShield でビルドされた Setup.exe ファイ ルによって実行される .msi パッケージについて、すべ てを詳細に記録するログ ファイルを生成する場合、次 のコマンドラインを入力します: /v"/l*v ¥"[ISLogDir]FileName.log¥"" アドバンスト UI またはスイート/アドバンスト UI イン ストールによって [ISLogDir] が、ログ ファイルを含む フォルダーへのパスで置き換えられます。アドバンス ト UI またはスイート/アドバンスト UI の /log コマンド ライン パラメーターを使用すると、エンドユーザーは、 パッケージのログ ファイル用のディレクトリを指定す ることができます。 詳細については、「アドバンスト UI またはスイート/ア ドバンスト UI インストールをコマンドラインから起動 したときに作成できるパッケージ ログ ファイルのサ ポート」を参照してください。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
LoggingEnable	読み書き ブール値 プロパ ティ	.msi、 .msp、 .exe、 InstallScript、 基本の MSI プロジェク ト InstallScript プロジェク ト	[パッケージ] ビューの "ログの有効化" 設定の値を取得または設定します。この設定は、アドバンスト UI またはスイート/アドバンスト UI インストールが、/log コマンドライン パラメーターが使われて、コマンドラインから起動される場合、パッケージでログ ファイルを生成するかどうかを示します。パッケージにログ機能がサポートされていない場合、そのパッケージに対して、ログ機能を有効にしないことをお勧めします。 [はい] を選択した場合、その他のログ関連プロパティを必要に応じて構成してください。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。
LoggingMsiLogFile	読み書き 文字列プ ロパティ	.msi、 .msp、 基本の MSI プロジェク ト	このプロパティは、LoggingEnable プロパティの値が True の場合に使用できます。 [パッケージ] ビューの "ログのコマンドライン" 設定の値を取得または設定します。この設定は、ログファイルの名前を示します。ファイルのパスを使用しないでください。アドバンスト UI またはスイート/アドバンスト UI の /log コマンドライン パラメーターによって、エンドユーザーは、パッケージのログ ファイル用のディレクトリを指定することができます。 この設定を空白のままにしておいた場合、インストールで作成されたログ ファイルに、*PackageGUID*.log という名前が使用されます。*PackageGUID* は、[パッケージ] ビューの "パッケージ GUID" 設定でパッケージに割り当てられた GUID です。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
LoggingMsiLogOptions	読み書き 文字列プ ロパティ	.msi、 .msp、 基本の MSI プロジェク ト	このプロパティは、LoggingEnable プロパティの値が True の場合に使用できます。 [パッケージ] ビューの "ログのオプション" 設定の値を取得または設定します。この設定は、ログ ファイルを生成するとき、パッケージで使用する Windows Installer ログ /L フラグを指定します。たとえば、すべてを詳細に記録するログ ファイルを作成する場合は、この設定で、次のように入力します: *v その他の使用可能なフラグについては、/L の説明を参照してください。 この設定を空白のままにしておくと、アスタリスク (*) と v フラグがログ ファイルの生成時に使用されます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。
MoveDown	メソッド	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェク ト InstallScript プロジェク ト、トラン ザクション	プロジェクトに 1 つ以上のパッケージまたはトランザクションが含まれている場合、[パッケージ] ビューにはアドバンスト UI または スイート/アドバンスト UI インストールが実行時にそれらを起動するのと同じ順番でリストされています。MoveDown メソッドは、順番リストの中でパッケージまたはトランザクションを下向きに移動し、次の項目の下に配置します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
MoveUp	メソッド	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト、トランザクション	プロジェクトに 1 つ以上のパッケージまたはトランザクションが含まれている場合、[パッケージ] ビューにはアドバンスト UI または スイート/アドバンスト UI インストールが実行時にそれらを起動するのと同じ順番でリストされています。MoveUp メソッドは、順番リストの中でパッケージまたはトランザクションを上向きに移動し、前の項目の上に配置します。
Name	読み書き 文字列プロパティ	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェクト InstallScript プロジェクト	パッケージの内部名を取得または設定します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
演算子	読みとり 専用オブ ジェクト プロパ ティ	サポートは 様々	このプロパティは、ISWiSuiteOperation オブジェクトに アクセスします。 指定された種類の操作を取得します。選択可能なオプ ションは以下のとおりです: • esotInstall (1)—パッケージは初回インストールとし て起動されます。 この処理は、.msi、.msp、.exe、.appx、InstallScript、 基本の MSI プロジェクト、および InstallScript プロ ジェクトのパッケージ ファイルで使用できます。 • esotRemove (2)—パッケージがインストールした製 品が削除されます。 この操作はほとんどの種類のパッケージで使用で きます (.msi、.exe、.appx、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。 • esotRepair (3)—パッケージがインストールした製品 を修正します。 この操作はほとんどの種類のパッケージで使用で きます (.msi、.exe、InstallScript、基本の MSI プロ ジェクト、および InstallScript プロジェクト)。 • esotModify (4)—パッケージがメンテナンス モード で起動され、エンド ユーザーが製品をインストー ル、削除、または変更することができます。 この操作はほとんどの種類のパッケージで使用で きます (.msi、.msp、.exe、InstallScript、基本の MSI プロジェクト、および InstallScript プロジェクト)。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
PackageLocation	読み書き 文字列プ ロパティ	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェク ト InstallScript プロジェク ト	このプロパティは、ISWiPackageLoc オブジェクトにア クセスします。 構成されているパッケージの配置場所を取得または設 定します。選択可能なオプションは以下のとおりです: • eplSourceMedia (0)—ソース メディアからコピーし ます。つまり、パッケージとそのファイルを、 ソース メディアに格納します。 アドバンスト UI またはスイート/アドバンスト UI インストールを CD、DVD またはローカル ネット ワークなどの固定メディアから非圧縮で実行する 場合は、このオプションを使用します。 • eplSetupExe (1)—エンジンを Setup.exe から抽出しま す。つまり、パッケージとそのファイルを Setup.exe に圧縮し、実行時に必要に応じて抽出さ れるようにします。 アドバンスト UI またはスイート/アドバンスト UI インストール全体を Setup.exe に完全に含める場合、 このオプションを使います。Web オプションを選 択すると、インストールが小さくなりダウンロー ド時間も短くなりますが、Setup.exe オプションは 完全に独立したインストールを提供します。 • eplWeb (2)—Web からダウンロードします。つまり、 パッケージとそのファイルファイルを、必要な場 合に、パッケージに指定された URL からダウン ロードします。 このオプションは、インストールがインターネッ トからダウンロードされ、かつ、アドバンスト UI またはスイート/アドバンスト UI パッケージのサ イズとダウンロード時間を最小化したい場合に推 奨されます。アドバンスト UI またはスイート/ア ドバンスト UI パッケージは、適切なバージョンが 既にターゲット システム上にある場合はダウン ロードされません。 リリースの構成中にすべてのパッケージでこのプロパ ティをオーバーライドすることができます。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
PackageType	読み取り専用文字列プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト、WPM パッケージ	プロジェクト内のパッケージの種類を取得します: • esptTransaction (-1)—Windows Installer トランザクション • esptMsi (1)—Windows Installer パッケージ (.msi) • esptMsp (2)—Windows Installer パッチ (.msp) • esptExe (3)—実行可能パッケージ (.exe) • esptIsp (4)—InstallScript パッケージ • esptIsmMsi (5)—基本の MSI プロジェクト (.ism) • esptIsmIsp (6)—InstallScript プロジェクト (.ism) • esptAppx (7)—サイドローディング Windows アプリ パッケージ (.appx) • esptWebDeploy (8)—Web 配置パッケージ (.zip) • esptAppxBundle (9) — サイドローディング Windows アプリ パッケージ (.appxbundle) • esptWpm (10) — Windows パッケージ マネージャー パッケージ
PackageURL	読み書き文字列プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	選択されたパッケージおよびそのフォルダーとファイルが含まれるルート フォルダーの URL を取得または設定します。パッケージを起動する必要がある場合、実行時にパッケージおよび関連ファイルが、この場所からターゲット システムにダウンロードされます。 このプロパティは、PackageLocation プロパティの値が eplWeb (2) のときに適用します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
ParcelType	読み書き文字列プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *スイート/アドバンスト UI プロジェクトでは、このプロパティを編集することができます。アドバンスト UI プロジェクトでは、プライマリ パッケージの数が 1 つのみサポートされているため、このプロパティは読み取り専用になっています。* このパッケージがターゲット システム上に存在するかどうかによって、スイート/アドバンスト UI インストールを初回インストール モードまたはメンテナンス モードで実行するかどうかを示す、パッケージの種類を取得します。 この設定で選択できるオプションは、次のとおりです: • **espItPrimary (0)**—選択されたパッケージは、アドバンスト UI またはスイート/アドバンスト UI インストールのメインの部分です。 実行時、ターゲット システム上からインストールのプライマリ パッケージのすべてが不足している場合、インストールは初回インストールとして実行します。ターゲット システム上にスイート インストールのいずれかのプライマリ パッケージが存在する場合、インストールはメンテナンス モードで実行します。 • **espItDependency (1)**—選択されたパッケージは、アドバンスト UI またはスイート/アドバンスト UI インストールをどのモードで実行するかを決定する要因ではありません。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI プロジェクトにおけるプライマリ パッケージと依存パッケージの違い」を参照してください。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
ProgressCapture	読み書き 文字列プ ロパティ	.msi、 .msp、 基本の MSI プロジェク ト	[パッケージ] ビューの "レポートされるステータス メッセージ" 設定の値を取得または設定します。この設定は、アドバンスト UI またはスイート/アドバンスト UI インストールのユーザー インターフェイスで表示可能なステータス メッセージの種類を示します。選択可能なオプションは以下のとおりです: • epsptCaptureFull (3)—アクション テキスト。アドバンスト UI またはスイート/アドバンスト UI インストールのステータス メッセージは、パッケージに含まれる標準アクションおよびカスタム アクションからのアクション テキストを含みます。 • epsptCaptureFullVerbose (4)—アクション テキストおよびアクション データ。アドバンスト UI またはスイート/アドバンスト UI インストールのステータス メッセージは、パッケージに含まれる標準アクションおよびカスタム アクションからのアクション テキストおよびアクション データを含みます。 アドバンスト UI またはスイート/アドバンスト UI インストールの実行時、ISParcelStatus プロパティは、ISParcelStatus プロパティのアクション テキストおよびアクション データ (適切な場合) によって更新されます。アクション データは、デフォルトで、アドバンスト UI またはスイート/アドバンスト UI インストールの InstallationProgress ウィザード ページで表示されます。 たとえば、.msi パッケージの InstallFiles アクションを実行中に、ISParcelStatus プロパティはアクション テキスト "新しいファイルをコピーしています" で更新されて、エンド ユーザーに対してインストールの現在の進行状況を通知します。ステータス メッセージにアクション データを含める場合は、さらにターゲット システム上にインストール中の各ファイルの名前、ディレクトリ、およびサイズといったアクション データで ISParcelStatus プロパティが更新されます。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの 種類	説明
ReleaseFlags	読み書き 文字列プ ロパティ	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェク ト InstallScript プロジェク ト	パッケージに関連付けられたリリース フラグを取得ま たは設定します。複数のフラグは、コンマで区切りま す。 リリース フラグを使って、アドバンスト UI またはス イート/アドバンスト UI インストールの異なるビルド にこのパッケージを含めるか、除外するかを選択でき ます。
RootFolder	読みとり 専用オブ ジェクト プロパ ティ	.msi、 .msp、 .exe、 .appx、 InstallScript、 Web 配置、 基本の MSI プロジェク ト InstallScript プロジェク ト	このプロパティは、ISWiSuiteFolder オブジェクトにアク セスします。 選択されたパッケージおよびそのフォルダーとファイ ルが含まれるルート フォルダーを取得します。
SharedParckage	読み書き ブール値 プロパ ティ	.msi、 .msp、 .exe、 .appx、 InstallScript、 基本の MSI プロジェク ト InstallScript プロジェク ト	[パッケージ] ビューの "共有" 設定の値を取得または 設定します。この設定は、パッケージが別の製品と共 有されていて、あとに残された他の製品でこのパッ ケージが使用されていない場合にのみ削除するかどう かを示します。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
UpgradeType	読み書き文字列プロパティ	.msi、基本の MSI プロジェクト	[パッケージ] ビューの "マイナー アップグレードの処理" 設定の値を取得または設定します。この設定は、ターゲット システム上にパッケージの以前のバージョンが存在する場合に起こる動作を示します。選択可能なオプションは以下のとおりです: • epsutNone (0)—なし。アドバンスト UI またはスイート/アドバンスト UI インストールはパッケージをマイナー アップグレード モードで実行せずに起動します。つまり、アドバンスト UI またはスイート/アドバンスト UI インストールは、このパッケージの REINSTALL または REINSTALLMODE プロパティを設定しません。 • epsutAuto (1)—自動。アドバンスト UI またはスイート/アドバンスト UI インストールは REINSTALL プロパティを *ALL* に、および REINSTALLMODE プロパティを *vomus* に設定して、パッケージをアップグレード モードで起動します。エンド ユーザーには、製品がアップグレードされる事を通知しません。 • epsutPrompt (2)—ユーザーに確認する。アドバンスト UI またはスイート/アドバンスト UI インストールは 2 番目のウィンドウを表示して、エンド ユーザーが処理の続行を希望するかどうかを確認します。エンド ユーザーが続行を希望した場合、アドバンスト UI またはスイート/アドバンスト UI インストールはアップグレード モードでパッケージを起動します。エンド ユーザーが続行しないことを選択した場合、パッケージは実行しません。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
WindowsFeatures	読み書き文字列プロパティ	.msi、.msp、.exe、.appx、InstallScript、Web 配置、基本の MSI プロジェクト InstallScript プロジェクト	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* Windows Vista 以降または Windows Server 2008 以降が搭載されているターゲット システム上で有効にする必要がある、選択されたパッケージが必要とする 1 つ以上の Windows の役割と機能のリストをセミコロン区切りで取得または設定します。 ロールと機能の名前は、Deployment Image Servicing and Management (DISM.exe) および Package Manager (Pkgmgr.exe) などのツールが Windows の特定のバージョンでロールまたは機能を識別するのに使用する文字列です。各 Windows のバージョンで使用可能な Windows の役割および機能の一覧は、Microsoft TechNet を参照してください。 実行時、このパッケージがインストールの対象であり、無効化されている 1 つ以上の Windows の役割または機能を必要とする場合、スイート/アドバンスト UI インストールは、パッケージを起動する前にこれらを有効化します。 詳細については、「スイート/アドバンスト UI インストール中に Windows 役割と機能を有効化する」を参照してください。
WPMAdditionalParameter	読み書き文字列プロパティ	WPM パッケージ	Windows パッケージ マネージャー リポジトリからパッケージをインストール中に渡したい任意の追加パラメーターを取得または設定します。 たとえば --accept-source-agreements パラメーターを使って、ソース ライセンスの使用許諾契約に同意することで問題を回避し、パッケージが正しくインストールされるようにします。

テーブル 68 • ISWiSuitePackage オブジェクトのメンバー (続き.)

名前	種類	パッケージ ファイルの種類	説明
WPMPackageId	読み書き文字列プロパティ	WPM パッケージ	Windows パッケージ マネージャー (WPM) パッケージを識別する Windows パッケージ マネージャー (WPM) パッケージ ID を取得または設定します。 Windows パッケージ マネージャー (WPM) パッケージ ID を取得するには、Microsoft リポジトリのパッケージ マニュフェストを検索するか、Windows パッケージ マネージャー クライアント winget コマンドライン ツールを **search** または **show** オプションで実行してください
WPMSpecificVersion	読み書き文字列プロパティ	WPM パッケージ	Windows パッケージ マネージャー リポジトリからインストールするパッケージの特定バージョンを取得または設定します。 **WPMVersionType** プロパティの値が **eswvSpecificVersion (1)** の場合に、このプロパティを使用します。
WPMVersionType	読み書き文字列プロパティ	WPM パッケージ	このプロパティには、**ISWiSuiteOperation** オブジェクトがアクセスします。 Windows パッケージ マネージャー リポジトリからインストールするパッケージのバージョンを取得または設定します。このプロパティで選択できるオプションは、次のとおりです: • eswvLatestVersion (0) — Windows パッケージ マネージャー リポジトリで使用可能な最新のパッケージをインストールするには、このオプションを選択します。 • eswvSpecificVersion (1) —Windows パッケージ マネージャー リポジトリで使用可能な任意のパッケージをインストールするには、このオプションを選択します。 ***メモ:*** *既定で **WPMVersionType** プロパティの値は **eswvLatestVersion (0)** で、最新パッケージをインストールします。*

例

次の BVScript 行は、プロジェクト内のパッケージについての情報を表示する方法をデモンストレーションします。Windows Installer トランザクションに複数のパッケージを含めて、スクリプトを再帰的に実装することができます。

```
Sub WritePackage(indent, p)
    If p.PackageType = -1 Then ' Transaction
```

```
        out.WriteLine indent & "トランザクション:"
        For Each sp In p.ISWiSuitePackages
            WritePackage indent & "  ", sp
        次へ
    Else
        out.WriteLine indent & IDS(p.DisplayName) & " (" & p.Name & ")"
        out.WriteLine indent & "種類: " & p.PackageType
        out.WriteLine indent & "説明: " & p.Description
        out.WriteLine indent & "共有: " & p.SharedPackage
        out.WriteLine indent & "場所n: " & p.PackageLocation & " " & p.PackageUrl
        out.WriteLine indent & "リリース フラグ: " & p.ReleaseFlags
        out.WriteLine indent & "ファイル:"
        WriteFolder indent & "  ", p.RootFolder
        out.WriteLine indent & "検出条件:"
        WriteCondition indent & "  ", p.DetectCondition
        out.WriteLine indent & "対象条件:"
        WriteCondition indent & "  ", p.EligibleCondition
    End If
End Sub
```

次に適用：

- ISWiProject

AddDetectCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddDetectCondition メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに検出条件を追加します。

検出条件は、パッケージがターゲット システム上に既にインストール済みであるかどうかを評価するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する条件です。実行時にこれが false から true (または true から false) に変更されない場合、アドバンスト UI またはスイート/アドバンスト UI インストールはパッケージがペイロードをインストール (または削除) に失敗したものとみなします。また、その後に実行されるメンテナンスと削除操作 (またはその後のメンテナンスおよびインストール操作) も、期待通りに動作しない可能性もあります。

構文

```
AddDetectCondition () As ISWiSuiteCondition
```

パラメーター

AddDetectCondition メソッドにはパラメーターがありません。

次に適用：

- ISWiSuitePackage

AddEligibleCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddEligibleConditionPackage メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージに対象条件を追加します。

対象条件は、ターゲット システムで実行するパッケージに必要な要件を満たしているかどうかを判断するためにアドバンスト UI またはスイート/アドバンスト UI インストールが使用する条件です。たとえば、パッケージが 64 ビット システム上でのみ実行する場合、条件に x64 プラットフォーム要件を設定できます。これによって、アドバンスト UI またはスイート/アドバンスト UI インストールは 64 ビット システム上でのみパッケージを起動します。また、エンド ユーザーが現在のパッケージ バージョンを使って、将来リリースされる新しいバージョンを上書きインストールすることを防ぐために、対象条件を設定することもできます。

構文

AddEligibleCondition () As ISWiSuiteCondition

パラメーター

AddEligibleConditionPackage メソッドにはパラメーターがありません。

次に適用：

- ISWiSuitePackage

AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

AddSuitePackage メソッドは、パッケージ (.msi パッケージまたは .msp パッチ) をプロジェクト内の Windows Installer トランザクションに追加します。これはまた、指定されたターゲット ファイル、および追加ファイルのためのオプションを追加します。このメソッドの使用方法は、InstallShield の [パッケージ] ビューでパッケージをトランザクションに追加する方法に似ています。ISWiSuitePackage オブジェクトを使って、新しいパッケージのプロパティを設定できます。

別の種類のパッケージをプロジェクトに追加するには、「AddSuitePackage メソッド (アドバンスト UI およびスイート/アドバンスト UI)」を参照します。

構文

AddSuitePackage (PackageType As ISWiSuitePackageType, TargetFileName As String, AdditionalFiles As ISWiAdditionalFiles) As ISWiSuitePackage

パラメーター

テーブル 69 • AddSuitePackage メソッドのパラメーター

パラメーター	説明
PackageType	トランザクションに追加するパッケージの種類を指定します。選択可能なオプションは以下のとおりです: • **esptMsi (1)**—Windows Installer パッケージ (.msi) • **esptMsp (2)**—Windows Installer パッチ (.msp)
TargetFileName	アドバンスト UI またはスイート/アドバンスト UI インストールが呼び出すパッケージ内のファイル (.msi ファイルなど) を指定します。
AdditionalFiles	パッケージに追加するファイルの種類を指定します。選択可能なオプションは以下のとおりです: • **eaftNothing (0)**—何も追加しません。このパッケージには追加ファイルが不要です。 • **eaftAdjacentFiles (1)**—隣接するファイル、つまりこのパッケージの隣にあるファイルを追加します。 • **eaftSubFolders (2)**—サブフォルダー内のファイル、つまりこのパッケージの下にあるファイルを追加します。 • **eaftAdjacentFilesAndSubFolders (3)**—隣接するファイルおよびサブフォルダー内のファイル、つまりこのパッケージの隣および下にあるファイルを追加します。

次に適用 :

- ISWiSuitePackage

DeleteCondition メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

DeleteCondition メソッドは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージから検出条件または対象条件を削除します。

構文

DeleteCondition (pCondition As ISWiSuiteCondition)

パラメーター

テーブル 70 • DeleteCondition メソッド パラメーター

パラメーター	説明
pCondition	削除する条件の ISWiSuiteCondition オブジェクトを指定します。指定する ISWiSuiteCondition オブジェクトは、パッケージの EligibleCondition プロパティまたはその DetectionCondition プロパティから返される条件オブジェクトでなくてはなりません。

次に適用：

- ISWiSuitePackage

MoveDown Method メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

プロジェクトに 1 つ以上のパッケージまたはトランザクションが含まれている場合、[パッケージ] ビューにはアドバンスト UI または スイート/アドバンスト UI インストールが実行時にそれらを起動するのと同じ順番でリストされています。MoveDown メソッドは、順番リストの中でパッケージまたはトランザクションを下向きに移動し、次の項目の下に配置します。

構文

MoveDown ()

パラメーター

MoveDown メソッドにはパラメーターがありません。

次に適用：

- ISWiSuitePackage

MoveUp Method メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

プロジェクトに 1 つ以上のパッケージまたはトランザクションが含まれている場合、[パッケージ] ビューにはアドバンスト UI または スイート/アドバンスト UI インストールが実行時にそれらを起動するのと同じ順番でリストされています。MoveUp メソッドは、順番リストの中でパッケージまたはトランザクションを上向きに移動し、前の項目の上に配置します。

構文

MoveUp ()

パラメーター

MoveUp メソッドにはパラメーターがありません。

次に適用 :

- ISWiSuitePackage

ISWiSuiteRelease オブジェクト (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

一部のエディションおよびプロジェクトによって異なる情報がある場合は、その内容が記載されています。

ISWiSuiteRelease オブジェクトは、InstallShield の [リリース] ビューのリリースを表します。

リリースを追加または削除するには、ISWiProject.AddSuiteRelease または ISWiProject.DeleteSuiteRelease を呼び出します。

メンバー

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー

名前	種類	説明
Build	メソッド	リリースをビルドします。ビルド プロセスに関するフィードバックを得るには、リリース上のイベントをサブスクライブする必要があります。詳細については、「ビルドのステータス イベント」を参照してください。
BuildLocation	読み書き文字列プロパティ	リリースがビルドされるトップレベルのディレクトリを取得または設定します。
CachePackagesLocally	読み書きブール値プロパティ	[リリース] ビューの [Setup.exe] タブにある "パッケージをローカルにキャッシュする" 設定を取得または設定します。この設定を使って、ターゲット システム上で実行されるアドバンスト UI またはスイート/アドバンスト UI パッケージを、[パッケージ] ビューで各パッケージに対して定義した場所にキャッシュするかどうかを指定します。デフォルト値は True です。 非圧縮リリースをビルドする場合、この設定の値に [いいえ] を指定することが推奨されます。
CertPassword	書き込み専用文字列プロパティ	ファイルがパスワードで保護されている場合、.pfx ファイルのパスワードを設定します。InstallShield はパスワードを暗号化して、プロジェクト (.ism) ファイルに保存します。 ビルド時に InstallShield はパスワードを使って .pfx でファイルの署名を行います。証明書がパスワードで保護されているのにもかかわらず、この設定に何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ***メモ:*** *署名に使用する証明書が証明書ストアにある場合、このプロパティは適用しません。証明書がパスワード保護付きでストアにインポートされている場合、ビルド時にそのパスワードがプロンプトされます。*

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
CheckPeriodicUpdates	読み書きブール値プロパティ	[メディア] ビューの [リリース] エクスプローラー内で、[アップデート] タブの [アプリのアップデートを確認する] 領域の下にある "アプリのアップデートを定期的に確認する" 設定を取得または設定します。この設定を使って、アプリケーションの更新を定期的に確認する追加の設定を構成するかどうかを指定できます。このプロパティで選択できるオプションは、次のとおりです: • **True** — このプロパティ値は、UpdateFrequencyType プロパティの使用を有効化します。 • **False** — このプロパティ値は、UpdateFrequencyType プロパティの使用を無効化します。
CryptographicProvider	読み書き文字列プロパティ	暗号化サービス プロバイダー (CSP) を取得または設定します。 ***メモ:*** *暗号化サービス プロバイダー (CSP) 名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*
CustomSignArgs	読み書き文字列プロパティ	署名ツール構成のコマンドライン引数を取得または設定して、ビルドが生成したファイルにデジタル署名行います。 たとえば、次のコマンドライン引数を使って、Microsoft ビルトイン署名ツールがバイナリに署名を行うカスタム オプションとして構成します。 sign /fd SHA256 /f "<ProgramFilesFolder>¥testCA.pfx" /t http://timestamp.digicert.com /p MyPassword [filename] [filename] 変数は、署名を行う完全修飾パスのプレースホルダーです。ビルド時に、これはバイナリ ファイルの完全修飾パスを解決して署名を行います。デフォルトで、引数の終わりにファイル パスが追加され、カスタム署名ツールに渡されます。ハードコード化されたパスを使用する代わりに、プロジェクトで定義されている任意のパス変数または環境変数を使用することができます。 ***メモ:*** *SigningType プロパティの値が* ***esesCustom (1).*** *の場合に、このプロパティを使用します。*

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
CustomSignPath	読み書き文字列プロパティ	ビルドが生成したファイルにデジタル署名行うために使用する署名ツールの場所を取得または設定します。 ***メモ:** SigningType プロパティの値が esesCustom (1) の場合に、このプロパティを使用します。*
DefaultLanguage	読み書き文字列プロパティ	指定のリリースのデフォルト言語の言語識別子を取得します。このプロパティ値は、[一般情報] ビューまたは [文字列エディタ] ビューで構成されたデフォルト プロジェクト言語をオーバーライドします。 詳細については、「デフォルトのプロジェクト言語の設定」を参照してください。
DigitalCertificateInfo	読み書き文字列プロパティ	リリースの署名に使用するデジタル証明書を取得または設定します。 マシン上で .pfx 証明書を指定するには、パスとファイル名を指定します。 証明書を含む証明書ストアを参照するには、次の形式の文字列を使用します: *Store*StoreLocation:Cert Subject **Store** は、証明書を含む証明書ストアの名前を表します。選択可能なオプションは次のとおりです: • My • Root • Trust • CA **StoreLocation** は、証明書を含む証明書ストアの場所を表します。選択可能なオプションは以下のとおりです: • ユーザー • マシン **Cert Subject** は、証明書のサブジェクトを表します。
DigitalURL	読み書き文字列プロパティ	エンドユーザーが製品や組織、会社に関しての情報を得るためのリンク先として、デジタル証明書の中で使用される URL を取得または設定します。完全修飾 URL を使用します (例、**http://www.mydomain.com**)。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
EnableLangSelectionDialog	読み書きブール値プロパティ	[リリース] ビューの [ビルド] タブにある "言語ダイアログ" 設定を取得または設定します。この設定を使って、インストールが完全なユーザー インターフェイスで実行されたときに、インストールで言語の選択ウィザード ページを表示するかどうかを指定します。
EnableSuiteUpdates	読み書きブール値プロパティ	[メディア] ビューの [リリース] エクスプローラー内で、[アップデート] タブの [全般] 領域の下にある "アップデートの有効化" 設定を取得または設定します。このプロパティを使って、アドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーについてダウンロード可能なアップデート サポートを行うかどうかを指定します。このプロパティで選択できるオプションは、次のとおりです: • **True** — このプロパティ値は、ダウンロード可能なアップデートを有効かします。 • **False** — このプロパティ値は、ダウンロード可能なアップデートを無効化します。
HourlyUpdateFrequency	読み書き文字列プロパティ	アプリケーションの更新を確認する時間間隔を取得または設定します。このプロパティは、UpdateFrequencyTyp プロパティの値が Enumvalues(1) の場合のみ使用されます。次の文字列フォーマットを使用します: n 時間
KeyContainerName	読み書き文字列プロパティ	暗号化サービス プロバイダー (CSP) に関連付けられた秘密キー コンテナ名を取得または設定します。 ***メモ:*** *コンテナ名は、EVベンダー認証クライアントツールのユーザー証明書の秘密キー証明書プロパティから取得できます。*

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
LauncherPrivileges	読み書き文字列プロパティ	[リリース] ビューの [Setup.exe] タブにある "必要実行レベル" 設定を取得または設定します。この 設定を使って、Windows Vista 以降のプラットフォーム上でプロジェクトの Setup.exe ファイルを実行するのに必要な最低実行レベルを指定します。選択可能なオプションは、以下のとおりです。 • **erelAsAdmin (2)** — Setup.exe の実行には、管理者権限が必要です。管理者は、Setup.exe の実行を承認する必要があります。 非管理者は、管理者としての認証が必要になります。 • **erelAsHighestAvail (1)** — Setup.exe の実行には、管理者権限が推奨されます。管理者は、Setup.exe の実行を承認する必要があります。 非管理者は、管理者権限を持たずに Setup.exe を実行します。 • **erelAsInvoker (0)** — Setup.exe の実行に、管理者権限は必要ありません。したがって、管理者権限を持たないユーザーも Setup.exe を実行することができます。Setup.exe は、資格情報または同意 (コンセント) を求める UAPrewC メッセージを表示しません。これは、アドバンスト UI およびスイート/アドバンスト UI プロジェクトのデフォルト オプションです。 InstallShield は、アプリケーション マニフェストを Setup.exe 起動ツールに埋め込みます。このマニフェストは選択された実行レベルを指定します。Windows Vista よりも古いバージョンのオペレーティング システムでは、必要実行レベルは適用されません。
Name	読み書き文字列プロパティ	リリースの内部名を取得または設定します。
PackageLocation	読み書きブール値プロパティ	プロジェクトの [パッケージ] ビューで構成したパッケージの実行時の場所を取得または設定します。選択可能なオプションは以下のとおりです: • **eplSourceMedia (0)**—ソース メディアからコピーします。 • **eplSetupExe (1)**—Setup.exe から抽出します。 • **eplWeb (2)**—Web からダウンロードします。 • **eplIndividual (3)**— 個別の選択に従います。 詳細については、「リリース レベルで、アドバンスト UI またはスイート/アドバンスト UI パッケージのランタイムの場所を指定する」を参照してください。

テーブル 71・ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
Password	読み書き文字列プロパティ	インストールのパスワードを取得または設定します。パスワードは大文字と小文字を区別します。 このプロパティは、アクションの PasswordProtectプロパティの値が True の場合に使用できます。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する」を参照してください。
PasswordProtect	読み書きブール値プロパティ	[リリース] ビューの [Setup.exe] タブにある "起動ツールをパスワードで保護" 設定を取得または設定します。この設定を使って、インストールをパスワードで保護するかどうかを示します。 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをパスワードで保護する」を参照してください。
PathVariableOverrides	読み書き文字列プロパティ	指定のリリースについて、プロジェクト内のパス変数の値を取得または設定します。このプロパティを使ってパス変数の値を設定すると、[パス変数] ビューで設定されたすべての値がオーバーライドされます。 この変数を使って [パス変数] ビューで構成されたユーザー定義のパス変数、環境変数、およびレジストリ変数を取得またはオーバーライドすることができますが、*<WindowsFolder>* などの定義済みパス変数をオーバーライドすることはできません。 以下は、PathVar1 というパス変数の値をオーバーライドするサンプル Visual Basic コードです: pISWiSuiteRelease.PathVariableOverrides = "PathVar1=C:¥Test1" + vbCrLf + "PathVar2=C:¥test2"

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
PostbuildEvent	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* InstallShield がリリースをビルドおよび署名した後に実行するコマンドを取得または設定します。 1 つ以上のコマンドを指定するには、この設定で 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。
PrebuildEvent	読み書き文字列プロパティ	 ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* InstallShield がリリースをビルドする前に実行するコマンドを取得または設定します。このイベントは InstallShield がリリース フォルダとログ ファイルを作成した後、リリースのビルドを開始する前に実行します。 1 つ以上のコマンドを指定するには、この設定で 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを入力するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳細については、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
PrepareFilesEvent	読み書き文字列プロパティ	 ***プロジェクト:*** *この情報は、スイート/アドバンスト UI プロジェクトに適用します。* ***エディション:*** *このプロパティは、InstallShield の Premier Edition で使用できます。* InstallShield のビルド中に実行するコマンドを取得または設定します。このイベントは、InstallShield がリリース フォルダーおよびログ ファイルを作成した後、InstallShield ビルドによって生成された DLL が .msi パッケージまたは Setup.exe ファイルにストリームされる前に実行されます。 1 つ以上のコマンドを指定するには、この設定で 1 行につき 1 つのコマンドを入力します。ビルド時に、InstallShield はリストされた順番で各コマンドを実行します。ビルドは、1 つのコマンドが完了するまで、次のコマンドの処理を待ちます。 コマンドを指定するとき、ハードコード化されたパスの代わりに、プロジェクトで定義された任意のパス変数と環境変数を使用できます。ビルド イベントのコマンドとして定義された特定の変数を使用することもできます。詳しくは、「ビルド前、ビルド中、およびビルド後に実行するコマンドを指定する」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
PreProcessorDefines	読み書き文字列プロパティ	 ***プロジェクト-*** *この情報は、スイート/アドバンスト UI プロジェクトに適用します。* プリプロセッサ変数の定義を取得または設定します。ここに指定されたプリプロセッサ変数定義は現在のリリースにのみ適用されます。他のリリースのスクリプトをコンパイルしているときには使用されません。等号およびコンマの前後に空白を置かないようにして、次の形式を使用します。 MYVARIABLE1=123,MYVARIABLE2 このような変数は、スクリプトのフローを制御する #if と #ifdef ステートメントにより、スクリプト内でテストできます。 この設定にプリプロセッサ変数の名前を入力すると、その変数が定義されます。たとえば、この設定に **MYVARIABLE** と入力すると、次の #ifdef ループ中のスクリプトコマンドが実行されます。 #ifdef MYVARIABLE //コマンド #endif この設定でプリプロセッサ変数定義を追加または変更した後、その追加または変更を有効にするためにインストールをコンパイルする必要があります。
ProductNameOverride	読み書き文字列プロパティ	[一般情報] ビューで指定された製品名を選択されたリリースの新しい値で取得またはオーバーライドします。製品名がどのように使用されるかについては、「製品名の指定」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
ProductVersionOverride	読み書き文字列プロパティ	[一般情報] ビューで指定された製品バージョンを選択されたリリースの新しい値で取得またはオーバーライドします。バージョンには、数値のみを使用できます。一般的なフォーマットは *aaa.bbb.ccccc* または *aaa.bbb.ccccc.ddddd* で、*aaa* はメジャーバージョン番号、*bbb* はマイナー バージョン番号、*ccccc* はビルド番号、および *ddddd* はバージョン番号を示します。*aaa* と *bbb* の最大値は 255 です。*ccccc* と *ddddd* の最大値は、65,535 です。 4 番目のフィールド (*ddddd*) を含めることもできますが、インストールは異なる製品バージョンを区別するときに製品バージョンのこの部分を無視します。詳細については、「製品バージョンを指定する」を参照してください。 指定した製品バージョンは、Setup.exe の [プロパティ] ダイアログ ボックスに表示されます。詳細については、「セットアップ ランチャーのファイルのプロパティをカスタマイズする」を参照してください。
ReleaseFlags	読み書き文字列プロパティ	機能のフィルターに使用するリリースフラグを持つ文字列を取得または設定します。複数のフラグは、コンマで区切ります。 リリース フラグを使用すると、各リリースで特定のアイテムを追加または削除することによって、インストールをスタマイズすることができます。 リリース フラグをアドバンスト UI またはスイート/アドバンスト UI プロジェクトの機能およびパッケージに割り当てると、割り当てたフラグに従って機能およびパッケージを含めたリリースを作成することができます。デフォルトでは、すべての機能およびパッケージがリリースに含まれます。ReleaseFlags プロパティ (または [リリース] ビューの [ビルド] タブにある "リリース フラグ" 設定) のフラグを指定すると、フラグが付いていないアイテムおよび指定されたリリース フラグを持つアイテムだけがインストールに含まれます。 ***メモ:*** *リリースがリリース フラグを持たない場合、リリース フラグを持つ該当するすべてのアイテムが含められます。フラグが付いていないアイテムのみを含める場合、存在しないフラグを指定します。たとえば、***NoFlags*** *の様に指定します。こうすると、フラグが付いていないアイテムだけがリリースに組み込まれます。*

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
SetupExeCompany	読み書き文字列プロパティ	Setup.exe のデフォルト会社名をオーバーライドする値を取得または設定します。なお、UseMyVersionInfo プロパティに、必ず True 値を選択してください。 会社名が、セットアップ起動プログラムの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
SetupExeCopyright	読み書き文字列プロパティ	Setup.exe のカスタム著作権情報を取得または設定します。なお、UseMyVersionInfo プロパティに、必ず True 値を選択してください。 著作権情報が、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
SetupExeDescription	読み書き文字列プロパティ	Setup.exe のカスタム ファイルの説明を取得または設定します。なお、UseMyVersionInfo プロパティに、必ず True 値を選択してください。 説明が、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
SetupExeVersion	読み書き文字列プロパティ	Setup.exe のデフォルトの製品バージョンおよびファイル バージョンを独自のバージョン番号でオーバーライドする値を取得または設定します。なお、UseMyVersionInfo プロパティに、必ず True 値を選択してください。 製品バージョンおよびファイル バージョンが、セットアップ ランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンド ユーザーが Setup.exe ファイルを右クリックして、[プロパティ] をクリックしたときに表示されます。 ファイル バージョンは、常に 4 つのフィールドで構成されます。製品バージョンに 4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、製品バージョンとして **1.1** を指定すると、Setup.exe のバージョン リソースで使用されるファイル バージョンは **1.1.0.0** となります。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
SetupFileName	読み書き文字列プロパティ	InstallShield がビルド時に生成するセットアップランチャー ファイルに使用するファイル名 (.exe ファイル拡張子は除く) を取得または設定します。この設定が空白の場合、InstallShield はデフォルト値 *Setup* を使用し、セットアップランチャーの名前は Setup.exe となります。
Build64bitSetupExe	読み書き文字列プロパティ	現在のスイート リリースの "64 ビット セットアップ ランチャー" 設定の値を取得または設定します。選択可能な値は、以下のとおりです: • TRUE – 64 ビット セットアップ ランチャー • FALSE – 32 ビット セットアップ ランチャー

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
SetupIcon	読み書き文字列プロパティ	リリースの "Setup.exe アイコンファイル" 設定を取得または設定します。このプロパティは実行可能ファイルのアイコンがビルド時に使う完全修飾名を指定します。プロパティの値がヌル値の場合、デフォルトのアイコンが使用されます。 デフォルトではインデックス 0 のアイコンが使われます。別のアイコンを指定するには、コンマ (,) と、アイコンのインデックスまたはリソース ID (前にマイナス (-) 記号がついている) をファイル名の後に指定します。たとえば、**C:¥Temp¥MyLibrary.dll,2** は 2 というインデックスを持つアイコンを、**C:¥Temp¥MyLibrary.dll,-100** は 100 と言うリソース ID を持つアイコンを示します。 パスをハードコード化する代わりに、[パス変数] ビューまたは ISWiPathVariables コレクションの一部で定義されたパス変数を使用できます。ビルド時、InstallShield によって、パス変数が適切な値に置換されます。
SignatureDescription	読み書き文字列プロパティ	Setup.exe ファイルの署名の説明を取得または設定します。ここで指定した説明は、UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示されます。UAC ダイアログ ボックスは、エンド ユーザーが署名されたファイルを起動したとき、昇格された権限が必要な場合に開きます。 この設定を空白のままに残すと、InstallShield は UAC (ユーザー アカウント制御) ボックスの「プログラム名:」ラベルの右側に表示される説明として、ファイル名を拡張子なしで使用します。
SignEnabled	読み書きブール値プロパティ	[リリース] ビューの [署名] タブにある "Setup.exe の署名" 設定を取得または設定します。この設定は、アドバンスト UI またはスイート/アドバンスト UI インストールに署名するかどうかを示します。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
SigningType	読み書き文字列プロパティ	ビルドが生成したファイルにデジタル署名行う方法を取得または設定します。次のオプションを使用してこのプロパティを設定します: • esesBuiltIn (0) — この値を使用して、デフォルトの InstallShield 署名ツールを使用してビルドで生成されたファイルにデジタル署名標準の方法を選択します。 • esesCustom (1) — この値を使用して、カスタマイズされた署名ツールを使用してビルドで生成されたファイルにデジタル署名するカスタム方法を選択します。 *メモ• 既定で SigningType プロパティの値は esesBuiltIn (0) で、標準の方法を選択します。*
SuiteGuidOverride	読み書き文字列プロパティ	アドバンスト UI またはスイート/アドバンスト UI インストールを一意に識別する GUID を取得またはオーバーライドします。
TokenPassword	読み書き文字列プロパティ	暗号化されてプロジェクト ファイルに保存される EV トークンパスワードを取得または設定します。 *メモ• EV トークンのパスワードを指定する際には、以下の情報に注意してください:* • *EV 証明書ベンダーは、EV トークンのパスワードの有効期限と、ロックされるまでの無効なパスワードの試行回数を決定します。したがって、この設定を選択すると、特定の間隔でパスワードを変更する必要があります。* • *EV トークンのパスワードがロックされている場合、トークンのパスワードのロック解除/リセットには管理者パスワードが必要です。*
UILanguages	読み書き文字列プロパティ	リリースに含まれるユーザー インターフェイス言語の 10 進数言語 IDをコンマ区切りの文字列で取得または設定します。 プロジェクトの [一般情報] ビューにある "セットアップ言語" 設定で言語が選択されていない場合、その言語は UILanguages プロパティの使用可能な言語ではありません。
UpdateDialogHeader	読み書き文字列プロパティ	アプリケーションのアップデート ダイアログ ボックスのタイトルを取得または設定します。
UpdateDialogText	読み書き文字列プロパティ	アプリケーションのアップデート ダイアログ ボックスのテキストを取得または設定します。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
UpdateFrequencyType	読み書きブール値プロパティ	アプリケーション更新プログラムの確認頻度を決定付ける、定期的な間隔を取得または設定します。このプロパティは、CheckPeriodicUpdates プロパティの値が True の場合に使用できます。このプロパティで選択できるオプションは、次のとおりです: • Enumvalues(0) — このプロパティ値は、毎日アップデートを確認します。 • Enumvalues(1) — このプロパティ値は、毎時アップデートを確認し、HourlyUpdateFrequency プロパティを有効化します。 • Enumvalues(2) — このプロパティ値は、毎週アップデートを確認し、WeeklyUpdateFrequencyWeeks と WeeklyUpdateFrequencyDaysプロパティの両方を有効化します。
UpdateUrl	読み書き文字列プロパティ	リリースされたときにダウンロードされるアドバンスト UI またはスイート/アドバンスト UI のセットアップ ランチャーのアップデートを作成する可能性がある場合、このプロパティを使って更新されたアドバンスト UI またはスイート/アドバンスト UI セットアップ ランチャーの絶対パス (ファイル名も含めます) を指定します。このサポートにより、エンド ユーザーが製品の以前のバージョンをインストールする前に、より新しいバージョンを簡単に取得することができます。 アップデートの要件を含む、アドバンスト UI またはスイート/アドバンスト UI インストールのアップデートに関する詳しい情報は、「アドバンスト UI またはスイート/アドバンスト UI インストールでダウンロード可能なアップデートをサポート」をご覧ください。
Use64bitSigning	読み書き文字列プロパティ	64 ビット署名フレームワークを使用してパッケージにデジタル署名を行う場合は、このプロパティを [True] に設定します。 32 ビット署名フレームワークを使用してパッケージにデジタル署名を行う場合は、このプロパティを [False] に設定します。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
UseMyVersionInfo	読み書きブール値プロパティ	**[リリース]** ビューの **[Setup.exe]** タブにある "カスタム バージョンのプロパティを使用する" 設定を取得または設定します。この設定を使って、Setup.exe のデフォルトの著作権情報とファイルの説明を独自の著作権情報とファイルの説明でオーバーライドするかどうかを指定します。True の値を指定する場合、SetupExeCopyright および SetupExeDescription プロパティに独自のカスタム値を入力します。 著作権情報とファイルの説明説明が、セットアップランチャーの [プロパティ] ダイアログ ボックスに表示されます。この [プロパティ] ダイアログ ボックスは、エンドユーザーが Setup.exe ファイルが右クリックして、[プロパティ] をクリックしたときに表示されます。 詳細については、「セットアップランチャーのファイルのプロパティをカスタマイズする」を参照してください。
UsePathVariableTestValues	読み書きブール値プロパティ	パス変数 にテスト値を使用した場合は、このプロパティを True に設定して現時点での実際の値に変数を設定します。
VMConfig	読み書き文字列プロパティ	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* この設定には、選択されたリリースを VM に配布するときに使用される VM 構成設定グループの名前を取得または設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
VMMachinePassword	読み書き文字列プロパティ	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* VMConfig プロパティで選択されている VM 構成にパスワードが指定されている場合、そのパスワードが VMMachinePassword プロパティのデフォルト値として取り扱われます。選択されたリリースのパスワードをオーバーライドするには VMMachinePassword プロパティを設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
VMMachineUserName	読み書き文字列プロパティ	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* VMConfig プロパティで選択されている VM 構成にユーザー名が指定されている場合、そのユーザー名が VMMachinePassword プロパティのデフォルト値として取り扱われます。選択されたリリースのユーザー名をオーバーライドするには VMMachineUserName プロパティを設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
VMSnapShot	読み書き文字列プロパティ	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* オプションで、リリースの配布に使用するスナップショットの名前を取得または設定します。 VMConfig プロパティで選択されている VM 構成にスナップショットが指定されている場合、そのスナップショットが VMSnapShotプロパティのデフォルト値として取り扱われます。 VM 構成にスナップショットが指定されていない場合、InstallShield は特定のスナップショットには戻りません。InstallShield は特定のスナップショットに戻さずに VM の電源をオンにして、VM にインストールをコピーします。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
VMStageMachineCopyPath	読み書き文字列プロパティ	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* コピー パスは、リリースの配布先となる VM 上の場所を識別します。パスの最後のサブフォルダーとして、InstallShield が VM 上に作成するパスを使用できます。その他のパスは既存していなくてはなりません。 VMConfig プロパティで選択されている VM 構成にインストール先パスが指定されている場合、そのインストール先パスが VMStageMachineCopyPath プロパティのデフォルト値として取り扱われます。選択されたリリースのインストール先パスをオーバーライドするには VMStageMachineCopyPath プロパティを設定します。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。

テーブル 71 • ISWiSuiteRelease オブジェクトのメンバー (続き.)

名前	種類	説明
VMStagePostBuild	読み書きブール値プロパティ	 ***プロジェクト:*** *これは、スイート/アドバンスト UI プロジェクトで使用できます。* ***エディション:*** *スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。* ビルドが成功するたびに選択されたリリースを、InstallShield で自動的に配布するかどうかを指定します。VM にリリースを配布するには、以下のタスクが必要です: 1. (指定されている場合) VM を指定されたスナップショットに戻す 2. VM の電源をオンにする。 3. テスト用にビルドされたリリースを VM にコピーする。 VM 配布の詳しい情報については、「ビルド時またはオンデマンドで InstallShield が初期化する仮想マシンにリリースを配布する」を参照してください。
WeeklyUpdateFrequencyWeeks	読み書き文字列プロパティ	アプリケーションの更新を確認する週数を取得または設定します。このプロパティは、UpdateFrequencyTyp プロパティの値が Enumvalues(2) の場合のみ使用されます。サポートされている値の範囲は 1 から 52 です。
WeeklyUpdateFrequencyDays	読み書き文字列プロパティ	アプリケーションの更新を確認する週数 (WeeklyUpdateFrequencyWeeks プロパティの値) を取得または設定します。このプロパティは、UpdateFrequencyTyp プロパティの値が Enumvalues(2) の場合のみ使用されます。 詳細については、「WeeklyTrigger::put_DaysOfWeek method (taskschd.h)」を参照しください。

次に適用：

- ISWiProject

Build メソッド (アドバンスト UI およびスイート/アドバンスト UI)

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI または スイート/アドバンスト UI プロジェクトのリリースをビルドするには、Build メソッドを呼び出します。

構文

Build ()

ビルドのステータスイベント

ISWiSuiteRelease オブジェクトは次のビルド ステータスイベントを使用します:

- StatusMessage (string Message, ref bool Cancel)
- ProgressMax (int Max, ref bool Cancel)
- ProgressIncrement (int Increment, ref bool Cancel)
- TextCancel (ref bool Cancel)
- GetDotNetLibrary (out object ???)
- WarningMessage (string Message)
- ErrorMessage (string Message)
- IsCancelledByUserAnError (ref bool IsError)
- StatusMessageEx (string Message, object Tag, ref bool Cancel)

これらのイベントは、ビルドプロセス中に発生してステータスのアップデートを提供します。ビルドを中止するには pbCancel を設定します。

次に適用：

- ISWiRelease

» アドバンスト UI およびスイート/アドバンスト UI プロジェクトのオートメーション コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

各アドバンスト UI またはスイート/アドバンスト UI オートメーション コレクションは、1 セットのオブジェクトです。たとえば、ISWiSuiteFeatures コレクションは、現在のプロジェクトの 1 セットの ISWiSuiteFeature オブジェクトです。このセクションでは、各アドバンスト UI またはスイート/アドバンスト UI オートメーション コレクションが説明されています。

ISWiLanguages コレクション

エディション: *複数言語インストールのサポートは、InstallShield Premier Edition のみで提供されています。*

プロジェクト: *この情報は、次のプロジェクトの種類に適用します :*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiLanguages コレクションには、現在の言語のすべての言語が含まれています。

メンバー

テーブル 72 · ISWiLanguages コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiLanguages コレクションの文字列エントリの合計数を戻します。
項目	読み取り専用プロパティ	言語のインデックス番号または名前を指定して、ISWiLanguage オブジェクト を読み出します。

次に適用 :

- ISWiProject

ISWiPathVariables コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiPathVariables コレクションには、現在のプロジェクトに含まれるすべてのアドバンスト UI またはスイート/アドバンスト UI プロパティが含まれています。

メンバー

テーブル 73 : ISWiPathVariables コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiPathVariables コレクションのパス変数の合計数を返します。
項目	読み取り専用プロパティ	パス変数のインデックス番号または名前を指定して、 ISWiPathVariable オブジェクトを取得します。

次に適用 :

- ISWiProject

ISWiProperties コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiPathVariable コレクションには、現在のプロジェクトに含まれるすべてのアドバンスト UI またはスイート/アドバンスト UI プロパティが含まれています。

メンバー

テーブル 74 • ISWiProperties コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiPathVariable コレクションの文字列エントリの合計数を戻します。
項目	読み取り専用プロパティ	プロパティのインデックス番号または名前を指定して、ISWiProperty オブジェクトを取得します。

次に適用：

- ISWiProject

ISWiSetupFiles コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSetupFiles コレクションには、現在のプロジェクトのすべてのサポート ファイルが含まれています。

メンバー

テーブル 75 • ISWiSetupFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使用して、ISWiSetupFiles コレクションの文字列エントリの合計数を戻します。
項目	読み取り専用プロパティ	セットアップ ファイルのインデックス番号または名前を指定して、ISWiSetupFiles オブジェクトを取得します。

次に適用：

- ISWiProject

ISWiSuiteActionRefs コレクション

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteActionRefs コレクションは、パッケージ内で特定のイベントにスケジュールされているすべてのスイート/アドバンスト UI アクションを含みます。

メンバー

テーブル 76 : ISWiSuiteActionRefs コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteActionRefs コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	イベントのインデックス番号または名前を指定して、ISWiSuiteActionRef オブジェクトを取得します。

次に適用 :

- ISWiSuitePackage

ISWiSuiteActions コレクション

プロジェクト: *この情報は、スイート/アドバンスト UI プロジェクトに適用します。*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteActions コレクションは、プロジェクトに含まれるすべてのスイート/アドバンスト UI アクションを含みます。

メンバー

テーブル 77 • ISWiSuiteActions コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteActions コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	アクションのインデックス番号または名前を指定して、ISWiSuiteAction オブジェクトを取得します。

次に適用：

- ISWiProject

ISWiSuiteConditionParameters コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteConditionParameters コレクションは、条件ステートメントのすべての設定を含みます。

メンバー

テーブル 78 • ISWiSuiteConditionParameters コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteConditionParameters コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	条件パラメーターのインデックス番号または名前を指定して、ISWiSuiteConditionParameters オブジェクトを取得します。

次に適用：

- ISWiSuiteCondition

ISWiSuiteConditions コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteConditions コレクションは、条件グループの下にあるすべての条件ステートメントを含みます。

メンバー

テーブル 79 : ISWiSuiteConditions コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteConditions コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	条件ステートメントのインデックス番号を指定して、ISWiSuiteConditions オブジェクトを取得します。

次に適用 :

- ISWiSuiteCondition

ISWiSuiteDFLFilters コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteDFLFilters コレクションは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトのパッケージが必要とする追加ファイルのダイナミック リンクと関連付けられたすべてのフィルターが含まれています。

メンバー

テーブル 80 • ISWiSuiteDFLFilters コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteDFLFilters コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	ダイナミック ファイル リンクのインデックス番号または名前を指定して、ISWiSuiteDFLFilters オブジェクトを取得します。

次に適用：

- ISWiSuitePackage

ISWiSuiteDynamicFileLinks コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteDynamicFileLinks コレクションは、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージのフォルダーに含まれるすべてのダイナミック ファイル リンクを含みます。

メンバー

テーブル 81 • ISWiSuiteDynamicFileLinks コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteDynamicFileLinks コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	ダイナミック ファイル リンクのインデックス番号または名前を指定して、ISWiSuiteDynamicFileLinks オブジェクトを取得します。

次に適用：

- ISWiSuiteFolder

ISWiSuiteEvents コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteEventsは、プロジェクトまたはプロジェクト内のパッケージに関連付けられているイベントのイベントを含むコレクションです。

メンバー

テーブル 82 - ISWISuiteEvents コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWISuiteEvents コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	ISWISuiteEvent オブジェクトを取得するイベントのインデックス番号を提供します。

次に適用：

- ISWiProject
- ISWiSuitePackage

ISWiSuiteExitConditions コレクション

プロジェクト- *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション- *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWISuiteExitConditions コレクションには、プロジェクトのすべての終了条件が含まれます。

メンバー

テーブル 83 - ISWISuiteExitConditions コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWISuiteExitConditions コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	終了条件のインデックス番号を指定して、ISWISuiteExitCondition オブジェクトを取得します。

次に適用：

- ISWiProject

ISWiSuiteFeatures コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteFeatures コレクションは、プロジェクトのあらゆる機能を (ISWiSuiteFeature オブジェクトとして) 含みます。

ISWiSuiteFeatures が ISWiProject のメンバーの場合、これにはプロジェクトのルートレベルの機能がすべて含まれます。サブ機能のコレクションを読み出すには、ISWiSuiteFeature オブジェクトの "機能" プロパティにアクセスします。その結果は、現在のルートレベル機能のすぐ下にあるサブ機能を含む ISWiSuiteFeatures コレクションです。これを繰り返すことで、さらに下のレベルのサブ機能が表示されます。

メンバー

テーブル 84 • ISWiSuiteFeatures コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteFeatures コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	機能のインデックス番号または名前を指定して、ISWiSuiteFeatures オブジェクトを読み出します。たとえば、次のステートメントでは、コレクションの最初の項目のコピーおよび Help_Files という名前の機能のコピーが作成されます。 Set pFeat1 = m_ISWiProject.ISWiSuiteFeatures.Item(1) Set pFeat2 = m_ISWiProject.ISWiSuiteFeatures.Item("Help_Files") 機能の名前は大文字と小文字を区別します。たとえば、"Help_Files" と "Help_files" は異なる機能として認識されます。 Item は ISWiSuiteFeatures のデフォルトのプロパティで、次の 2 行のコードに相当します: m_ISWiProject.ISWiSuiteFeatures.Item("Help_Files") m_ISWiProject.ISWiSuiteFeatures("Help_Files").

次に適用：

- ISWiProject

ISWiSuiteFiles コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteFiles コレクションは、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージのフォルダーに含まれるすべてのファイルを含みます。

メンバー

テーブル 85 • ISWiSuiteFiles コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteFiles コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	ファイルのインデックス番号またはファイル名を指定して、ISWiSuiteFiles オブジェクトを読み出します。

次に適用：

- ISWiSuiteFolder

ISWiSuiteFolders コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteFolders コレクションは、アドバンスト UI またはスイート/アドバンスト UI プロジェクト内のパッケージのフォルダーに含まれるすべてのサブフォルダーを含みます。

メンバー

テーブル 86 • ISWiSuiteFolders コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteFolders コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	サブフォルダーのインデックス番号または名前を指定して、ISWiSuiteFolder オブジェクトを読み出します。

次に適用：

- ISWiSuiteFolder

ISWiSuitePackages コレクション

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション• *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuitePackages コレクションは、プロジェクトに含まれる Windows Installer トランザクション内のすべてのパッケージ (ISWiSuitePackage オブジェクトとして) が含まれます。

メンバー

テーブル 87 • ISWiSuitePackages コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuitePackages コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	トランザクションのインデックス番号または名前を指定して、ISWiSuitePackages オブジェクトを読み出します。

次に適用：

- ISWiProject

ISWiSuiteReleases コレクション

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

ISWiSuiteReleases コレクションは、アドバンスト UI またはスイート/アドバンスト UI プロジェクトに含まれるすべてのリリースを含みます。

メンバー

テーブル 88 • ISWiSuiteReleases コレクションのメンバー

名前	種類	説明
Count	読み取り専用プロパティ	このプロパティを使って、ISWiSuiteReleases コレクション内の要素の合計数を戻します。
項目	読み取り専用プロパティ	リリースのインデックス番号または名前を指定して、ISWiSuiteRelease オブジェクトを読み出します。

次に適用：

- ISWiProject

アドバンスト UI およびスイート/アドバンスト UI プロジェクトで使用可能な式のオブジェクト リファレンス

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI プロジェクトの様々な設定では、ファイル、レジストリ エントリ、その他のアイテムについての情報をターゲット システムでクエリするオブジェクト式を埋め込むことができます。オブジェクト式には、この規則を使用します:

[@Object(Parameters, ...).Property(Parameters, ...)]

各オブジェクト式には、オブジェクト固有プロパティの集まりである、オブジェクトへの参照が含まれています。オブジェクトとプロパティにはパラメーターを含めることができます。

ドキュメントの本章では、オブジェクト式で使用可能な各オブジェクトについて説明します。

テーブル 1 • オブジェクト式で使用可能なオブジェクト

オブジェクト	説明
Feature オブジェクト	ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストールの機能についての情報を確認します。
File オブジェクト	ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストール中に、ファイルについての情報を確認します。
Parcel オブジェクト	ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストールのパッケージについての情報を確認します。
Platform オブジェクト	ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストール中に、オペレーティング システムの詳細を確認します。
Registry オブジェクト	ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストール中に、レジストリを確認します。

Feature オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストールの機能についての情報を確認するオブジェクト式を作成するには、Feature オブジェクトを使用します。Feature オブジェクトは、DisplayName、ActionState、および InstalledState といったプロパティをサポートします。

構文

[@Feature(*FeatureName*).ActionState]

[@Feature(*FeatureName*).AllowSelectionChange]

[@Feature(*FeatureName*).Cost]

[@Feature(*FeatureName*).Description]

[@Feature(*FeatureName*).DisplayName]

[@Feature(*FeatureName*).InstalledState]

[@Feature(*FeatureName*).Visible]

パラメーター

Feature オブジェクトには、1 つのパラメーターを使用できます。

テーブル 2 ・ Feature オブジェクトのパラメーター

パラメーター	説明
FeatureName	機能の名前を指定します。

プロパティ

Feature オブジェクトには、次のプロパティを使用できます。

テーブル 3 ・ Feature オブジェクトのプロパティ

プロパティ	説明
ActionState	現在設定されている機能のアクションの状態を取得します。利用可能な値は、以下のとおりです。 ・ 0—機能に対して発生する操作はありません。 ・ 1—機能がインストールされます。 ・ 2—機能が削除されます。

テーブル 3 • Feature オブジェクトのプロパティ (続き.)

プロパティ	説明
AllowSelectionChange	エンド ユーザーが InstallationFeatures ウィザード ページに表示されている機能を選択または選択解除できるようにするかどうかを指定するブール値を取得します。利用可能な値は、以下のとおりです。 • **true** — エンド ユーザーがウィザード インターフェイスを使って機能を選択または選択解除できます。デフォルトでは、これが設定されています。 • **false** — エンド ユーザーがウィザード インターフェイスを使って機能を選択または選択解除できません。InstallationFeatures ウィザード ページでこの機能が表示されるときは、読み取り専用の状態です (機能の条件などのその他の要因によって選択/非選択となります)。
コスト	この値を [機能] ビューで指定して、ターゲット システム上の機能に必要な空き容量をバイト単位で取得します。
説明	機能の翻訳済みの説明を取得します。
DisplayName	機能の翻訳済みの表示名を取得します。
InstalledState	機能に含まれるパッケージのパッケージ検出状態によって判別された機能のインストール状態を取得します。利用可能な値は、以下のとおりです。 • **0**—機能は現在インストールされていません。 • **3**—機能はインストールされています。
Visible	インストール中、InstallationFeatures ページで機能を表示するかどうかを指定するブール値を取得します。

例

次の例では、オブジェクト式における Feature オブジェクトの使用方法をデモンストレーションします。

テーブル 4 • Feature オブジェクト式のサンプル

式	説明
[@Feature(FeatureABC).ActionState]	アドバンスト UI またはスイート/アドバンスト UI が FeatureABC 機能をインストールするかどうかを判別します。
[@Feature(FeatureABC).Cost]	機能が必要とする空き容量をバイト単位で判別します。

File オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストールのファイルについての情報を確認するオブジェクト式を作成するには、File オブジェクトを使用します。File オブジェクトは、Date や Version といったプロパティをサポートします。

構文

[@File(*FilePath*,*64Bit*).Content]

[@File(*FilePath*,*64Bit*).Date(Parameter)]

[@File(*FilePath*,*64Bit*).Version]

パラメーター

File オブジェクトには、次のパラメーターを使用できます。

テーブル 5 · File オブジェクトのパラメーター

パラメーター	説明
FilePath	ファイルへのパスを指定します。[ProgramFilesFolder] などのプロパティ ディレクトリ識別子を使用できます。
64Bit	このオプション ブール パラメーターを使って、ファイルが 64 ビットの場所 (64 ビット システム上の System32 フォルダーなど) にあるかどうかを指定します。有効なオプションは次のとおりです: • **true**—64 ビットの場所を確認します。 • **false**—32 ビットの場所を確認します。 このプロパティを省略したとき、またはターゲット システムが 32 ビット版 Windows を実行中の場合、インストールは 32 ビットの場所を確認します。

プロパティ

File オブジェクトには、次のプロパティを使用できます。

テーブル 6 · File オブジェクトのプロパティ

プロパティ	説明
コンテンツ	ファイルがテキストの場合 (たとえば .txt、.htm、.xml、.config、.ini、または .sql)、ファイルのコンテンツを取得します。

テーブル 6 • File オブジェクトのプロパティ (続き.)

プロパティ	説明
Date	そのファイルの日時を取得します。このプロパティには、オプション パラメーターが使用できます: • **modified**—ファイルの最終変更時の日付けと時刻を取得します。これがデフォルトのパラメーターです。 • **access**—ファイルの最終アクセス時の日付けと時刻を取得します。 • **create**—ファイル作成時の日付けと時刻を取得します。 Date プロパティのプロパティを省略すると、File オブジェクトは最終変更時の日付けを返します。
Version	ファイルのバージョンを取得します。バージョン付きでないファイルの場合、このプロパティは空白を返します。

例

次の例では、オブジェクト式における File オブジェクトの使用方法をデモンストレーションします。

テーブル 7 • File オブジェクト式のサンプル

式	説明
[@File([ProgramFilesFolder]MyCompany¥MyProduct.exe,false).Date(modified)]	MyProduct.exe という名前のファイルの最終変更日時を取得します。 32 ビット ターゲット システム上でのパス例: C:¥Program Files¥MyCompany¥MyProduct.exe 64 ビット ターゲット システム上でのパス例: C:¥Program Files (x86)¥MyCompany¥MyProduct.exe
[@File([SystemFolder]notepad.exe).Version]	32 ビットターゲット システム上では System32 フォルダー、64 ビット ターゲット システム上では SysWOW64 フォルダーにある notepad.exe のバージョン番号を取得します。
[@File([SystemFolder]notepad.exe,true).Version]	32 ビットおよび64 ビット ターゲットシステム上の System32 フォルダーにある notepad.exe のバージョン番号を取得します。

Parcel オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストール パッケージについての情報を確認するオブジェクト式を作成するには、Parcel オブジェクトを使用します。Parcel オブジェクトは、Eligible、ActionState、および DetectedState といったプロパティをサポートします。

構文

[@Parcel(*PackageGUID*).ActionState]

[@Parcel(*PackageGUID*).DetectedState]

[@Parcel(*PackageGUID*).Eligible]

[@Parcel(*PackageGUID*).ExitCode]

[@Parcel(*PackageGUID*).Type]

パラメーター

Parcel オブジェクトには、1 つのパラメーターを使用できます。

テーブル 8 • Parcel オブジェクトのパラメーター

パラメーター	説明
PackageGUID	パッケージの GUID を指定します。

プロパティ

Parcel オブジェクトには、次のプロパティを使用できます。

テーブル 9 • Parcel オブジェクトのプロパティ

プロパティ	説明
ActionState	現在設定されているパッケージのアクションの状態を取得します。この値は、プロパティが照会されるタイミングによって異なります。利用可能な値は、以下のとおりです。 • 1—パッケージがインストール操作を実行します。 • 2—パッケージが変更操作を実行します。 • 3—パッケージが修正操作を実行します。 • 4—パッケージが削除操作を実行します。 • 5—パッケージに対して発生する操作はありません。
DetectedState	パッケージのインストール状態を取得します。利用可能な値は、以下のとおりです。 • 0—パッケージはインストールされていません。 • 1—パッケージは現在インストールされています。

テーブル 9 • Parcel オブジェクトのプロパティ (続き.)

プロパティ	説明
Eligible	ターゲット システムが、パッケージを実行するのに必要な要件を満たしているかどうかを示すブール値を取得します。
ExitCode	スケジュールされた操作が完了した時点で、パッケージの終了コードを取得します。パッケージの操作が実行される前に、このプロパティの値は不確定な状態です。
Name	パッケージの表示名を取得します。
Type	パッケージの種類を取得します。利用可能な値は、以下のとおりです。 • 0—実行可能パッケージ (.exe) • 1—Windows Installer パッケージ (.msi) • 2—パッチ パッケージ (.msp) • 3—InstallScript パッケージ (.hdr) • 5—サイドローディング Windows アプリ パッケージ (.appx) • 6—Web 配置パッケージ (.zip)

例

次の例では、オブジェクト式における Parcel オブジェクトの使用方法をデモンストレーションします。

テーブル 10 • Parcel オブジェクト式のサンプル

式	説明
[@Parcel(0F283EC0-E9D1-4D18-801D-0014F6CA96DF).DetectedState]	参照されるパッケージがインストール済みかどうかを判別します。
[@Parcel(0F283EC0-E9D1-4D18-801D-0014F6CA96DF).ExitCode]	スケジュールされた操作が完了した時点で、パッケージの終了コードを取得します。

Platform オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストール中にオペレーティング システムの詳細を確認するオブジェクト式を作成するには、Platform オブジェクトを使用します。Platform オブジェクトは、Architecture、FullVersion、および MajorVersion といったプロパティをサポートします。

構文

[@Platform.Architecture]

[@Platform.BuildNumber]

[@Platform.CSDVersion]

[@Platform.FullVersion]

[@Platform.MajorVersion]

[@Platform.MinorVersion]

[@Platform.ProductType]

[@Platform.ServicePackVersion]

[@Platform.Value]

パラメーター

Platform オブジェクトにはパラメーターがありません。

プロパティ

Platform オブジェクトには、次のプロパティを使用できます。

テーブル 11 • Parcel オブジェクトのプロパティ

プロパティ	説明
アーキテクチャ	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンのアーキテクチャ (x86、x64、IA64、ARM、または不明) を取得します。
BuildNumber	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンのビルド番号バージョンを取得します。 有効なビルド番号のリストは、MSDN ライブラリのドキュメントに記載されている OSVERSIONINFOEX 構造 の dwBuildNumber メンバーを参照してください。
CSDVersion	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンの Windows の CSD バージョンを取得します。Service Pack 2 は CSD バージョンの一例です。
FullVersion	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンの Windows の完全バージョン番号を取得します。値は MajorVersion X 100 + MinorVersion で計算されます。 たとえば、Windows 8.1 のバージョン番号は 6.3 です。Windows 8.1 が投資されたマシンの FullVersion 値は 603 です。

テーブル 11 • Parcel オブジェクトのプロパティ (続き.)

プロパティ	説明
MajorVersion	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンの Windows のメジャー バージョン番号を取得します。
MinorVersion	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンの Windows のマイナー バージョン番号を取得します。
ProductType	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンの製品の種類を取得します。利用可能な値は、以下のとおりです。 • 1—ワークステーション • 2—ドメイン コントローラー • 3—サーバー
ServicePackVersion	アドバンスト UI またはスイート/アドバンスト UI インストールが実行中のマシンの Windows のサービス パック バージョンを取得します。
Value	FullVersion プロパティと同じ値を取得します。

例

次の例では、オブジェクト式における Platform オブジェクトの使用方法をデモンストレーションします。

テーブル 12 • Platform オブジェクト式のサンプル

式	説明
[@Platform.FullVersion]	ターゲット システムのバージョン番号を取得します。
[@Platform.Architecture]	ターゲット システムのアーキテクチャを取得します。

Registry オブジェクト

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

ターゲット システムで、アドバンスト UI またはスイート/アドバンスト UI インストール中にターゲット システム上のレジストリを確認するオブジェクト式を作成するには、Registry オブジェクトを使用します。

構文

[@Registry(*RegistryKeyPath*,*64Bit*).KeyValue(*DataValueName*)]

[@Registry(*RegistryKeyPath*,*64Bit*).Value]

パラメーター

Registry オブジェクトには、次のパラメーターを使用できます。

テーブル 13 • Registry オブジェクトのパラメーター

パラメーター	説明
RegistryKeyPath	ターゲット システムで確認するレジストリ キーの完全パスを指定します。ルートには、次の省略形を使用します。 • HKCR • HKCU • HKLM
64Bit	このオプション ブール パラメーターを使って、レジストリの [64 ビット] ビューを確認するかどうかを示します。有効なオプションは次のとおりです: • **true**—レジストリの [64 ビット] ビューを確認します。 • **false**—レジストリの [32 ビット] ビューを確認します。 このプロパティを省略したとき、またはターゲット システムが 32 ビット版 Windows を実行中の場合、インストールはレジストリの [32 ビット] ビューを確認します。

プロパティ

Registry オブジェクトには、次のプロパティを使用できます。

テーブル 14 • Registry オブジェクトのプロパティ

プロパティ	説明
KeyValue	KeyValue プロパティのパラメーターとして渡された値の値データを取得します。
値	レジストリ キーのデフォルト値の値データを取得します。

例

次の例では、オブジェクト式における Registry オブジェクトの使用方法をデモンストレーションします。

テーブル 15 • Registry オブジェクト式のサンプル

式	説明
[@Registry(HKLM¥Software¥Microsoft¥ Windows NT¥CurrentVersion, false). KeyValue(RegisteredOwner)]	32 ビット ターゲット システム上で、次のレジストリ キーの RegisteredOwner 値のデータを取得します: HKLM¥Software¥Microsoft¥Windows NT¥CurrentVersion 64 ビット ターゲット システム上で、次のレジストリ キーの RegisteredOwner 値のデータを取得します: HKLM¥Software¥Wow6432Node¥Microsoft¥Windows NT¥CurrentVersion
[@Registry(HKLM¥Software¥[COMPANY]¥ [PRODUCT], true).Value]	32 ビットおよび 64 ビット ターゲット システム上で、次のレジストリ キーの デフォルト値のデータを取得します: HKLM¥Software¥My Company Name¥My Product Name
[@Registry(HKCR¥Software¥My Company Name¥My Product Name, false).KeyValue(MyFile[¥[]1[¥]].exe)]	32 ビット ターゲット システム上で、次のレジストリ キーの MyFile[1].exe 値のデータを取得します: HKCU¥Software¥My Company Name¥My Product Name 64 ビット ターゲット システム上で、次のレジストリ キーの MyFile[1].exe 値のデータを取得します: HKLM¥Software¥Wow6432Node¥My Company Name¥My Product Name データ値名の各括弧がパラメータの解析インジケーターとして処理されないよう、サンプル式では各括弧がエスケープされています。

InstallShield カスタム アクション リファレンス

このセクションでは、様々な機能をサポートするために InstallShield プロジェクトに自動的に追加されるビルトイン InstallShield カスタム アクションそれぞれについて説明します。

Windows ロゴ・ *Windows ロゴ プログラムを適用する場合、インストールに含まれるカスタム アクションそれぞれについて、その正常時の動作を文書化しなくてはなりません。これは、カスタム アクション タイプ 19 (エラー、失敗、およびインストールの終了を表示する)、タイプ 35 (インストール ディレクトリを設定する)、または タイプ 51 (プロパティを設定する) には適用されません。ISICE10 は、各カスタム アクションが文書化されていることを検証します。*

_serial_verifyCA_isx

SERIALNUMVALRETRYLIMIT プロパティの値を 1 減らします。

_serial_verifyCA_isx_helper

SERIALNUMVALRETRYLIMIT プロパティの値を 1 減らします。

CheckForProductUpdates

FlexNet Connect を使用して、製品のアップデートを確認します。

カスタム アクションは Agent.exe という名前の実行可能ファイルを起動して、次の内容を渡します。

/au[ProductCode] /EndOfInstall

CheckForProductUpdatesOnReboot

FlexNet Connect を使用して、起動時に製品のアップデートを確認します。

カスタム アクションは Agent.exe という名前の実行可能ファイルを起動して、次の内容を渡します。

/au[ProductCode] /EndOfInstall /Reboot

DLLWrapCleanup

抽出されたデータをクリーンアップする標準 DLL ラッパー。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は dllwrap.dll で、そのエントリ ポイントは DLLWrapCleanup です。

DLLWrapStartup

呼び出しを説明するデータを抽出する標準 DLL ラッパー。

これは、DLL カスタム アクションです。DLL ファイルの名前は dllwrap.dll で、そのエントリ ポイントは DLLWrapStartup です。

ISChainPackageCommit

複数パッケージのトランザクションから抽出されたファイルを削除します。

ISChainPackagePrepare

複数パッケージのトランザクション用にファイルを準備して抽出します。

ISChainPackageRollback

複数パッケージのトランザクションから抽出されたファイルを削除します。

ISComponentServiceCosting

ISComPlusApplication テーブルから情報を抽出し、COM+ アプリケーションの一時ファイルに保存します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は iscomsrv.dll で、そのエントリ ポイントは ISComponentServiceCosting です。

ISComponentServiceFinalize

インストールおよびアンインストール時に、COM+ アプリケーションをコミットします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は iscomsrv.dll で、そのエントリ ポイントは ISComponentServiceFinalize です。

ISComponentServiceInstall

インストール時に、COM+ アプリケーションをインストールします。

これは、DLL カスタム アクションです。DLL ファイルの名前は iscomsrv.dll で、そのエントリ ポイントは ISComponentServiceInstall です。

ISComponentServiceRollback

インストールまたはアンインストールが失敗したとき、COM+ アプリケーションをロールバックします。

これは、DLL カスタム アクションです。DLL ファイルの名前は iscomsrv.dll で、そのエントリ ポイントは ISComponentServiceRollback です。

ISComponentServiceUninstall

アンインストール時に、COM+ アプリケーションを削除します。

これは、DLL カスタム アクションです。DLL ファイルの名前は iscomsrv.dll で、そのエントリ ポイントは ISComponentServiceUninstall です。

ISIISCleanup

IIS インストールの一時ファイルとレジストリ エントリを削除します。

これは、DLL カスタム アクションです。DLL ファイルの名前は IISHelper.dll で、そのエントリ ポイントは ISIISCleanup です。

ISIISCosting

IIS インストールの一時ファイルにアクション一覧を作成します。他の IIS アクションの CustomActionData プロパティを設定します。

これは、DLL カスタム アクションです。DLL ファイルの名前は IISHelper.dll で、そのエントリ ポイントは ISIISCosting です。

ISIISInstall

IIS インストールに Web サイト、アプリケーション、仮想ディレクトリ、および他のアイテムを作成します。

これは、DLL カスタム アクションです。DLL ファイルの名前は IISHelper.dll で、そのエントリ ポイントは ISIISInstall です。

ISIISRollback

IIS のロールバック中に Web サイト、アプリケーション、仮想ディレクトリ、および他のアイテムを削除します。

これは、DLL カスタム アクションです。DLL ファイルの名前は IISHelper.dll で、そのエントリ ポイントは ISIISRollback です。

ISIISUninstall

IIS のアンインストール中に Web サイト、アプリケーション、仮想ディレクトリ、および他のアイテムを削除します。

これは、DLL カスタム アクションです。DLL ファイルの名前は IISHelper.dll で、そのエントリ ポイントは ISIISUninstall です。

ISInstallPrerequisites

機能と関連付けられた前提条件のインストールを起動します。

これは、DLL カスタム アクションです。DLL ファイルの名前は PrqLaunch.dll で、そのエントリ ポイントは InstallPrerequisites です。

ISJITCompileActionAtInstall

インストール時に、.NET アセンブリをプリコンパイルします。

このカスタム アクションは、ngen.exe という名前の Microsoft 実行可能ファイルを起動します。

ISJITCompileActionAtUnInstall

アンインストール時にプリコンパイル済み .NET アセンブリを削除します。

このカスタム アクションは、ngen.exe という名前の Microsoft 実行可能ファイルを起動します。

ISLockPermissionsCost

ISLockPermissionsInstall アクションの CustomActionData プロパティを設定します。

これは、DLL カスタム アクションです。DLL ファイルの名前は ISLockPermissions.dll で、そのエントリ ポイントは ISLockPermissionsCostAction です。

ISLockPermissionsInstall

製品がインストールされるときのアクセス許可を設定します。

これは、DLL カスタム アクションです。DLL ファイルの名前は ISLockPermissions.dll で、そのエントリ ポイントは ISLockPermissionsInstallAction です。

ISNetApiInstall

.ini ファイルからユーザーおよびグループを作成します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetApiInstall です。

ISNetApiRollback

ユーザーとグループの変更をロールバックします。

これは、DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetApiRollback です。

ISNetCreateIniForOneUser

ユーザーとグループ用の一時ファイルに操作を抽出します。

これは、ISNetAPI.dll という名前の DLL カスタム アクションです。

ISNetDeleteIniFile

ユーザーとグループ用の一時ファイルをクリーンアップします。

これは、ISNetAPI.dll という名前の DLL カスタム アクションです。

ISNetGetGroups

グループをコンボ ボックスに加えます。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetGetGroups です。

ISNetGetServers

サーバーの一覧をコンボ ボックスに加えます。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetGetServers です。

ISNetGetUsers

ユーザーの一覧をコンボ ボックスに加えます。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetGetUsers です。

ISNetSetLogonName

LogonInformation ダイアログで入力された "ユーザー"、"グループ"、および "サーバー" プロパティを格納します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetSetLogonName です。

ISNetValidateLogonName

LogonInformation ダイアログで、有効な組み合わせのユーザー名、サーバー、パスワードが入力されたことを検証してください。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetValidateLogonName です。

ISNetValidateNewUserInformation

LogonInformation ダイアログで入力された "ユーザー"、"グループ"、および "サーバー" プロパティを格納します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetAPI.dll で、そのエントリ ポイントは ISNetValidateNewUserInformation です。

ISNetworkSharesCosting

ISNetworkShares テーブルから情報を抽出し、ネットワーク共有の一時ファイルに保存します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetworkShares.dll で、そのエントリ ポイントは ISNetworkSharesCosting です。

ISNetworkSharesFinalize

ネットワーク共有の構成用の一時ファイルをクリーンアップします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetworkShares.dll で、そのエントリ ポイントは ISNetworkSharesFinalize です。

ISNetworkSharesInstall

1 つ以上のフォルダー用のネットワーク共有を構成します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetworkShares.dll で、そのエントリ ポイントは ISNetworkSharesInstall です。

ISNetworkSharesRollback

1 つ以上のフォルダー用のネットワーク共有をロールバックします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetworkShares.dll で、そのエントリ ポイントは ISNetworkSharesRollback です。

ISNetworkSharesUninstall

1 つ以上のフォルダー用のネットワーク共有を削除します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISNetworkShares.dll で、そのエントリ ポイントは ISNetworkSharesUninstall です。

ISPrint

ダイアログ上の ScrollableText コントロールの内容を印刷します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は SetAllUsers.dll で、そのエントリ ポイントは PrintScrollableText です。

ISQuickPatchFinalize

QuickPatch の共有参照カウントをクリーンアップします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は QuickPatchHelper.dll で、そのエントリ ポイントは ISQuickPatchFinalize です。

ISQuickPatchFixShortcut

QuickPatch のショートカットを再インストールします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は QuickPatchHelper.dll で、そのエントリ ポイントは ISQuickPatchFixShortcut です。

ISQuickPatchHelper

QuickPatch のクリーンな機能の状態を適用します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は QuickPatchHelper.dll で、そのエントリ ポイントは ISQuickPatchHelper です。

ISQuickPatchInit

QuickPatch のコンポーネントと機能の状態をクリーンアップします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は QuickPatchHelper.dll で、そのエントリ ポイントは ISQuickPatchInit です。

ISQuickPatchInit9X

QuickPatch のコンポーネントと機能の状態をクリーンアップします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は QuickPatchHelper.dll で、そのエントリ ポイントは ISQuickPatchInit9X です。

ISQuickPatchInit9X2

QuickPatch のコンポーネントと機能の状態をクリーンアップします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は QuickPatchHelper.dll で、そのエントリ ポイントは ISQuickPatchInit9X2 です。

ISRunSetupTypeAddLocalEvent

SetupType ダイアログの [次へ] ボタンに関連付けられている AddLocal イベントを実行します。インストールが SetupType ダイアログを表示しない場合、このアクションを呼び出す必要があります。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は ISXExpHlp.dll で、そのエントリ ポイントは RunSetupTypeAddLocalEvent です。

ISSearchReplaceCosting

テキスト ファイルに追加する必要がある変更の一覧を含む一時ファイルを作成します。

これは、DLL カスタム アクションです。DLL ファイルの名前は isschrpl.dll で、そのエントリ ポイントは ISSearchReplaceCosting です。

ISSearchReplaceFinalize

テキスト ファイルの変更用の一時ファイルをクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は isschrpl.dll で、そのエントリ ポイントは ISSearchReplaceFinalize です。

ISSearchReplaceInstall

製品がインストールされる時に、テキスト ファイルの変更を行います。

これは、DLL カスタム アクションです。DLL ファイルの名前は isschrpl.dll で、そのエントリ ポイントは ISSearchReplaceInstall です。

ISSearchReplaceRollback

テキスト ファイルの変更をロールバックして、テキストファイルの変更用の一時ファイルをクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は isschrpl.dll で、そのエントリ ポイントは ISSearchReplaceRollback です。

ISSearchReplaceUninstall

製品がアンインストールされる時に、テキスト ファイルの変更を行います。

これは、DLL カスタム アクションです。DLL ファイルの名前は isschrpl.dll で、そのエントリ ポイントは ISSearchReplaceUninstall です。

ISSelfRegisterCosting

自己登録用の一時ファイルに操作を抽出します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は isregsvr.dll で、そのエントリ ポイントは ISSelfRegisterCosting です。

ISSelfRegisterFiles

自己登録ファイルを登録します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は isregsvr.dll で、そのエントリ ポイントは ISSelfRegisterFiles です。

ISSelfRegisterFinalize

自己登録用の一時ファイルをクリーンアップします。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は isregsvr.dll で、そのエントリ ポイントは ISSelfRegisterFinalize です。

ISSetAllUsers

アップグレードまたは初回インストール要件の ALLUSERS を設定します

これは、DLL カスタム アクションです。DLL ファイルの名前は SetAllUsers.dll で、そのエントリ ポイントは SetAllUsers です。

ISSetTARGETDIR

TARGETDIR を [INSTALLDIR] に設定します。

これは、DLL カスタム アクションです。DLL ファイルの名前は SetAllUsers.dll で、そのエントリ ポイントは SetTARGETDIR です。

ISSetupFilesCleanup

サポート ファイルの一時ディレクトリをクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は SFHelper.dll で、そのエントリ ポイントは SFCleanupEx です。

ISSetupFilesExtract

一時ディレクトリに、サポート ファイルを抽出します。

これは、DLL カスタム アクションです。DLL ファイルの名前は SFHelper.dll で、そのエントリ ポイントは SFStartupEx です。

ISSQLQueryDatabases

指定されたデータベース サーバーで使用できるデータベース カタログを検索する。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLQueryDatabases です。

ISSQLServerCosting

SQL スクリプト用の一時ファイルに操作を抽出します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerCosting です。

ISSQLServerFilteredList

SQL スクリプトに使用可能なサーバーを検索します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerFilteredList です。

ISSQLServerFinalize

一時ファイルと SQL スクリプトの状態をクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerFinalize です。

ISSQLServerInitialize

MsiHiddenProperties に SQL ログイン パスワードのプロパティ名を追加する。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerInitialize です。

ISSQLServerInstall

インストール中に SQL スクリプトを実行します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerInstall です。

ISSQLServerList

使用可能なサーバーを検索します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerList です。

ISSQLServerRemoveLoginInfo

アンインストール中に SQL データベース ログイン認証情報を Windows レジストリから削除します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerRemoveLoginInfo です。

ISSQLServerRollback

インストールが失敗したとき、またはアンインストール中に SQL スクリプトを実行します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerRollback です。

ISSQLServerRollbackLoginInfo

インストールが失敗したとき、またはアンインストール中に Windows レジストリに格納されている SQL データベース ログイン認証情報をロールバックします。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerRollbackLoginInfo です。

ISSQLServerUninstall

アンインストール中に SQL スクリプトを実行します。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerUninstall です。

ISSQLServerValidate

サーバーの接続をテストします。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerValidate です。

ISSQLServerWriteLoginInfo

インストール中に SQL データベース ログイン認証情報を Windows レジストリに書き込みます。

これは、DLL カスタム アクションです。DLL ファイルの名前は issqlsrv.dll で、そのエントリ ポイントは ISSQLServerWriteLoginInfo です。

ISUnSelfRegisterFiles

自己登録ファイルの登録を解除します。

これは、DLL カスタム アクションです。DLL ファイルの名前は isregsvr.dll で、そのエントリ ポイントは ISUnSelfRegisterFiles です。

ISVerifyScriptingRuntime

Setup.exe を使用して InstallScript MSI インストールが起動されたかどうかを確認し、起動されなかった場合、STANDARD_USE_SETUPEXE プロパティにあるメッセージを表示します。このカスタム アクションはすべての InstallScript MSI プロジェクトに含まれており、ユーザー インターフェイス シーケンスの先頭にスケジュールされています。

メモ: *ISVerifyScriptingRuntime カスタム アクションは、InstallScript UI スタイルが従来型のスタイル (InstallScript エンジンを外部 UI ハンドラーとして使用するスタイル) である InstallScript MSI インストールで呼び出されます。新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) では呼び出されません。*

詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。

ISXmlAppSearch

XML のシステム検索を実行します。つまり、ファイルを検索してから、XPath クエリを実行し、その結果をプロパティに格納します。ISXmlLocator テーブルに基づいて動作します。

これは、DLL カスタム アクションです。DLL ファイルの名前は isxmlcfg.dll で、そのエントリ ポイントは ISXmlAppSearch です。

ISXmlCosting

XML ファイルの変更用の一時ファイルに操作を抽出します。

これは、DLL カスタム アクションです。DLL ファイルの名前は isxmlcfg.dll で、そのエントリ ポイントは ISXmlCosting です。

ISXmlFinalize

XML ファイルの変更用の一時ファイルをクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は isxmlcfg.dll で、そのエントリ ポイントは ISXmlFinalize です。

ISXmlInstall

.xml ファイルの変更を適用します。

これは、DLL カスタム アクションです。DLL ファイルの名前は isxmlcfg.dll で、そのエントリ ポイントは ISXmlInstall です。

ISXmlRollback

.xml ファイルの変更をロールバックします。

これは、DLL カスタム アクションです。.DLL ファイルの名前は isxmlcfg.dll で、そのエントリ ポイントは ISXmlRollback です。

ISXmlUnInstall

.xml ファイルの変更をアンインストールします。

これは、DLL カスタム アクションです。DLL ファイルの名前は isxmlcfg.dll で、そのエントリ ポイントは ISXmlUnInstall です。

LaunchProgramFileFromSetupCompleteSuccess

インストールの終わりで実行可能ファイルを起動します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は SerialNumCAHelper.dll で、そのエントリ ポイントは LaunchProgram です。

LaunchReadmeFileFromSetupCompleteSuccess

インストールの終わりで Readme ファイルを起動します。

これは、Windows Installer DLL カスタム アクションです。DLL ファイルの名前は SerialNumCAHelper.dll で、そのエントリ ポイントは LaunchReadMe です。

setAllUsersProfile2K

Windows 2000 以降の ALLUSERSPROFILE ディレクトリ識別子を初期化します。

SetAllUsersProfileNT

Windows NT 4 以降の ALLUSERSPROFILE ディレクトリ識別子を初期化します。

SetARPINSTALLLOCATION

ARPINSTALLLOCATION プロパティの値を、アプリケーションのプライマリ フォルダーの完全修飾パスに設定します。

setUserProfileNT

USERPROFILE ディレクトリ識別子を初期化します。

ShowMsiLog

エンドユーザーが SetupCompleteSuccess、SetupCompleteError、または SetupInterrupted ダイアログで **"Windows Installer のログを表示"** チェック ボックスを選択して [完了] をクリックしたとき、メモ帳で Windows Installer ログ ファイルを表示します。これは、Windows Installer 4.0 以降でのみ使用できます。

WiseScriptFinalize

WiseScript カスタム アクションの一時ファイルをクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は WiseScriptBridge.dll で、そのエントリ ポイントは WiseScriptFinalize です。

WiseScriptInitialize

WiseScript カスタム アクションに必要なファイルを抽出します。

これは、DLL カスタム アクションです。DLL ファイルの名前は WiseScriptBridge.dll で、そのエントリ ポイントは WiseScriptInitialize です。

WiseScriptRollback

WiseScript カスタム アクションの一時ファイルをクリーンアップします。

これは、DLL カスタム アクションです。DLL ファイルの名前は WiseScriptBridge.dll で、そのエントリ ポイントは WiseScriptFinalize です。

Summary List of All Custom Action Types (Windows Installer ヘルプ ライブラリ)

コマンドライン ツール

グラフィック ユーザー インターフェイスの他に、InstallShield はビルド時 (たとえば、バッチ処理の一部、またはインストールの動作をカスタマイズするための実行時) に利用できるいくつかのコマンドライン プログラムも提供します。

オーサリング時のプログラム

テーブル 1 • オーサリング時のプログラムの説明

タスク	プログラム
IIS Web サイトをスキャンしてその設定を記録し、InstallShield プロジェクトにインポートできるようにします。	iisscan.exe

ビルドタイム プログラム

テーブル 2 • ビルドタイム プログラムの説明

タスク	プログラム
InstallScript ファイル (.rul) のコンパイル	Compile.exe
リリースのビルド。適切な場合は、InstallScript もコンパイルします。	ISCmdBld.exe
リリースのビルド (レガシー InstallScript プロジェクトで利用可能)	ISBuild.exe
ビルド済みの InstallScript リリースに署名を行う	iSign.exe
リリースのビルド (InstallScript 自己抽出型実行可能ファイル)	ReleasePackager.exe
基本の MSI または InstallScript MSI.exe ファイルに埋め込まれている Setup.ini を変更する	SetupIni.exe

ランタイム プログラム

テーブル 3 • ランタイム プログラムの説明

タスク	プログラム
Setup.exe インストールを実行して、コマンドライン パラメーターを渡す (基本の MSI、InstallScript、および InstallScript MSI プロジェクト)	Setup.exe
Setup.exe インストールを実行して、コマンドライン パラメーターを渡す (アドバンスト UI およびスイート/アドバンスト UI プロジェクト)	Setup.exe
インストールの実行 (InstallScript プロジェクト)	Setup.exe (InstallScript プロジェクト)
.msi パッケージを実行する	MsiExec.exe
パッチ パッケージの実行し (Windows Installer ベースのプロジェクト)、コマンドライン パラメーターを渡す	Update.exe

Compile.exe

InstallShield は、コマンドライン プロンプト ウィンドウまたは、DOS バッチ ファイルから起動できるコマンドライン コンパイラを搭載しています。このプログラムは Compile.exe と呼ばれ、次のフォルダーに保存されています。

InstallShield Program Files フォルダー/System

メモ: ISCmdBld.exe *を使用して、コマンドラインからリリースをビルドすると、ビルド エンジンは自動的にスクリプトをコンパイルします。したがって、InstallShield のプロジェクトに指定されているコンパイラ オプション以外のオプションを使用する必要がない場合、*Compile.exe *を直接使用する必要はありません。詳細については、「ISCmdBld.exe」を参照してください。*

構文

Compile script_file [iswi_obl_file] [isrt_obl_file] [ifx_obl_file] switches

いくつかまたはすべてのコマンドライン パラメーターをコマンド ファイルの形で Compile.exe に渡すことができます。コマンド ファイルを Compile.exe に指定するには、次の構文を使います:

Compile @command_file

この例は、*command_file* がテキストファイルの名前で、拡張子を含んでいる場合です。ファイル名を使った絶対パスまたは相対パスを指定できます。

コマンドラインのすべて、または一部をコマンドファイルに指定できます。複数のコマンドファイルを[コンパイル] コマンドで使うこともできます。Compile.exe はコマンド ファイルの入力を、コマンドラインのその場所で指定されたかのように受け入れます。

コマンドファイルの中では、各パラメーターは同じ行で始まり同じ行で終わならなければならず、円記号 (¥) を使ってパラメーターの 2 つの行を組み合わせることはできません。コマンドファイルでは、引数をスペースやタブ (コマンドラインのように) 、および改行文字 (¥n) で区切ることができます。

パラメーター

次のパラメーターは、コンパイラと共に使用することができます。

テーブル 4 ・ Compile.exe のコマンドライン パラメーター

パラメーター	説明
script_file	セットアップスクリプトの名前を指定します。次の事項に注意してください。 ・ ファイル名には、ドライブのインストール先や、絶対パスまたは相対パスを含むことができます。ファイル名が完全でない場合、コンパイラはスクリプトを現在のドライブの現在のディレクトリの中から検索します。 ・ 指定したセットアップスクリプトが #include 指令を使ってセットアップの別のファイルを含む場合、また #include ステートメントが指定したファイル名がパスを含まない場合、これらのファイルは現在のディレクトリに常駐しない限り、場所を –i スイッチを使って指定しなければなりません。インクルードしたファイルは自分で #include 指令を指定することができるので注意してください。–i スイッチがすべてのインクルードしたソースファイルの検索に使用できる検索パスを指定していることを確認してください。 ・ 長いパス名とファイル名は二重引用符で囲まなければなりません。 ・ デフォルトでは、新しい InstallShield プロジェクトの一部として作成されたセットアップスクリプトには setup.rul の名前がつけられます。
isrt_obl_file	ライブラリ ファイル Isrt.obl の名前を指定します。パスが –libpath スイッチで明示されている限り、ライブラリ ファイルのフル パスを含める必要はありません。 Isrt.obl は、次のディレクトリに保存されています: *InstallShield Program Files フォルダー*¥Script¥Isrt¥Lib –c または –l スイッチを使用しない限り、このパラメーターは必須です。
ifx_obl_file	 ***プロジェクト:*** *このパラメーターは、InstallScript プロジェクトに適用します。* ライブラリ ファイル Ifx.obl の名前を指定します。パスが –libpath スイッチで明示されている限り、ライブラリ ファイルのフル パスを含める必要はありません。 Ifx.obl は、次のディレクトリに保存されています: *InstallShield Program Files フォルダー*¥Script¥Ifx¥Lib –c または –l スイッチを使用しない限り、このパラメーターは必須です。

テーブル 4 • Compile.exe のコマンドライン パラメーター (続き.)

パラメーター	説明
iswi_obl_file	 ***プロジェクト•*** *このパラメーターは、次のプロジェクト タイプに適用します:* • *基本の MSI* • *InstallScript MSI* ライブラリ ファイル Iswi.obl の名前を指定します。パスが -libpath スイッチで明示されている限り、ライブラリ ファイルのフル パスを含める必要はありません。 Iswi.obl は、次のディレクトリに保存されています: *InstallShield Program Files フォルダー*¥Script¥Iswi¥Lib -c または -I スイッチを使用しない限り、このパラメーターは必須です。
ifxobject_obl_file	 ***プロジェクト•*** *このパラメーターは、InstallScript オブジェクトプロジェクトに適用します。* ライブラリ ファイル Ifxobject.obl の名前を指定します。パスが -libpath スイッチで明示されている限り、ライブラリ ファイルのフル パスを含める必要はありません。 Ifxobject.obl は、次のディレクトリに保存されています: *InstallShield Program Files フォルダー*¥Script¥Ifx¥Lib -c または -I スイッチを使用しない限り、このパラメーターは必須です。

スイッチ

次のスイッチをコンパイラに渡すことができます。スクリプト ファイルおよびライブラリ ファイルへの完全パスを指定しない場合、-i および -libpath スイッチを使って、これらのファイルを含むフォルダーの場所を指定しなくてはなりません (下の -i および -libpath の説明を参照)。その他すべてのスイッチはオプションです。

テーブル 5 • Compile.exe 用のスイッチ

スイッチ	説明
/d または -d<変数名=プリプロセッサ定義>	使用する InstallScript に適切なプリプロセッサ定義を指定します。参考例: Compile.exe -dVARIABLENAME=Value 変数名は有効な InstallScript 識別子で、値は定数でなければなりません。スイッチと式の間、または式同士の間にスペースを置くことはできません。 InstallScript でプリプロセッサを参照する場合、次のいずれかの形式を使用します: • #ifdef VARIABLENAME • #if VARIABLENAME=Value #ifdef ステートメントの場合、値名のみを使用することができます。#if ステートメントの場合、name=value ペアを使用します。 このパラメーターはオプションです。 ***メモ:*** *このコマンドライン パラメーターは、InstallShield の次の領域でプリプロセッサが定義されていない場合のみ使用することをお勧めします。* • [設定] ダイアログ ボックスの [コンパイル/リンク] タブ ([ビルド] メニューで [設定] をクリックしたとき表示されます) • 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合: [リリース] ビューの製品構成における "プリプロセッサ定義" 設定 *InstallShield のこれらのいずれかの領域でプリプロセッサを定義してから、-d コマンドライン パラメーターを使って、同じプリプロセッサを定義した場合、Compile.exe によって、プリプロセッサが InstallShield プロジェクトまたは製品構成の設定を使って再定義できないことを通知する "定義不可能シンボル" メッセージが表示されます。*
/e または -e	エラーメッセージの最大数を指定します。デフォルト値は 50 です。エラーメッセージの最大数が生成されると、コンパイルは停止します。
/g または -g	デバッグ情報を生成するよう指定します。このファイルには、インストールの名前と拡張子 .dbg が付けられます。

テーブル 5 • Compile.exe 用のスイッチ (続き.)

スイッチ	説明
/gi または -gi	デバッグ情報がコンパイル済みスクリプト ファイルに含まれるよう指定します。この場合、デバッグ情報ファイルは必要ありません。 ただし、コンパイルされたスクリプトが大きくなりインストールに時間がかかる上、他人が容易にコードをリバースエンジニアリングできてしまうため、通常、エンドユーザーへの配布用の最終インストール作成では使用しないほうが適切です。
/i または -i<検索パス>	#include ステートメントによってメイン インストールの InstallScript コードに含まれたソースファイルを検索するディレクトリを識別する検索パス (つまり、パスのリスト、各々はセミコロンで区切られます) を指定します。長いパス名は二重引用符で囲まなければなりません。 ***プロジェクト:*** *InstallScript および InstallScript オブジェクト プロジェクトでは、次のフォルダーの完全パスを指定しなくてはなりません:* • *InstallShield Program Files フォルダー¥Script¥Ifx¥Include* • *InstallShield Program Files フォルダー*¥Script¥Isrt¥Include *基本の MSI および InstallScript MSI プロジェクトでは、次のフォルダーの完全パスを指定しなくてはなりません:* • *InstallShield Program Files フォルダー*¥Script¥Iswi¥Include • *InstallShield Program Files フォルダー*¥Script¥Isrt¥Include *上述のディレクトリには、ビルトイン InstallScript 関数用のスクリプト ヘッダーが含まれます。これらのフォルダーを指定しなかった場合、コンパイラ エラーが発生します。*

テーブル 5 • Compile.exe 用のスイッチ (続き.)

スイッチ	説明
/libpath または –libpath<パス>	Ifx.obl、Isrt.obl、またはカスタム ライブラリ ファイルといった、ライブラリの検索対象ディレクトリを特定する単一のパスを指定します。複数ディレクトリを指定するには、–libpath スイッチを複数回にわたって使用します。長いパス名は二重引用符で囲まなければなりません。 ***プロジェクト:*** *InstallScript と InstallScript オブジェクト プロジェクトの場合、次のフォルダーに完全パスを指定しなくてはなりません (ライブラリ ファイル名を含む完全パスを使用した場合を除く):* • *InstallShield Program Files フォルダー*¥Script¥Ifx¥Lib • *InstallShield Program Files フォルダー*¥Script¥Isrt¥Lib *基本の MSI および InstallScript MSI プロジェクトの場合、次のフォルダーに完全パスを指定しなくてはなりません (ライブラリ ファイル名を含む完全パスを使用した場合を除く):* • *InstallShield Program Files フォルダー*¥Script¥Iswi¥Lib • *InstallShield Program Files フォルダー*¥Script¥Isrt¥Lib *上述のディレクトリには、ビルトイン InstallScript 関数を定義するライブラリが含まれています。これらのフォルダー、またはライブラリ名を含む完全パスを指定しなかった場合、コンパイラ エラーが 発生します。*
/o または –o<コンパイル済みスクリプト ファイル名>	コンパイル済みスクリプトに割り当てるファイル名を指定します。次の事項に注意してください。 • ファイル名には、ドライブのインストール先や、絶対パスまたは相対パスを含むことができます。パスが指定されない場合、コンパイラはコンパイルしたスクリプトを現在のディレクトリに格納します。 • このパラメーターを指定しない場合、コンパイルしたスクリプトはスクリプトファイルの名前と拡張子t .inx が付けられ、現在のディレクトリに格納されます。 • 長いパス名とファイル名は二重引用符で囲まなければなりません。
/q または –q	著作権メッセージとバージョン情報の出力を抑止します。
/v または –v	警告のレベルを次のいずれかの値に設定します。 • 0 – 警告メッセージは表示されません。 • 1 – InstallShield で処理できないシステム警告メッセージを表示します。 • 2 – レベル 1 メッセージに加えて、文字列長が制限を越えた場合のメッセージを表示します。 • 3 – すべての警告メッセージを表示します。これがデフォルトの設定になります。

テーブル 5 • Compile.exe 用のスイッチ (続き.)

スイッチ	説明
/w または -w	警告メッセージの最大数を指定します。デフォルト値は 50 です。警告メッセージの最大数が生成されると、コンパイルは停止します。

追加情報

- ファイル名を指定する場合、短いファイル名と長いファイル名のどちらを使うこともできますが、長いファイル名は二重引用符で囲まなければなりません。
- 相対パスをファイル名に指定する場合 (たとえば ..¥..¥My Functions)、指定したパスはコンパイラを呼び出すときの現在のドライブの現在のディレクトリの相対パスでなければなりません。

例

InstallScript プロジェクトのサンプル コマンドライン

InstallScript プロジェクトのための次のコマンドラインは、C:¥InstallShield 2021 Projects¥My InstallScript Project¥script files フォルダーにあるスクリプト ファイル Setup.rul をコンパイルします。

```
"C:¥Program Files¥InstallShield¥2021¥System¥Compile.exe"
"C:¥InstallShield 2021 Projects¥My InstallScript Project¥script files¥setup.rul"
ifx.obl
isrt.obl
-libpath"C:¥Program Files¥InstallShield¥2021¥Script¥Ifx¥Lib"
-libpath"C:¥Program Files¥InstallShield¥2021¥Script¥Isrt¥Lib"
-i"C:¥Program Files¥InstallShield¥2021¥Script¥Ifx¥Include"
-i"C:¥Program Files¥InstallShield¥2021¥Script¥Isrt¥Include"
-i"C:¥InstallShield 2021 Projects¥My InstallScript Project¥script files"
```

InstallScript MSI または基本の MSI プロジェクト用のサンプル コマンドライン

InstallScript MSI または基本の MSI プロジェクトのための次のコマンドラインは、C:¥InstallShield 2021 Projects¥My New Project¥script files フォルダーにあるスクリプト ファイル Setup.rul をコンパイルします。

```
"C:¥Program Files¥InstallShield¥2021¥System¥Compile.exe"
"C:¥InstallShield 2021 Projects¥My New Project¥script files¥setup.rul"
ISWI.obl
isrt.obl
-libpath"C:¥Program Files¥InstallShield¥2021¥Script¥Iswi¥Lib"
-libpath"C:¥Program Files¥InstallShield¥2021¥Script¥Isrt¥Lib"
-i"C:¥Program Files¥InstallShield¥2021¥Script¥Ifx¥Include"
-i"C:¥Program Files¥InstallShield¥2021¥Script¥Isrt¥Include"
-i"C:¥InstallShield 2021 Projects¥My New Project¥script files"
```

ISCmdBld.exe

Windows Installer ベースのプロジェクトと InstallScript プロジェクトでは 、ISCmdBld.exe を使って、コマンドラインからリリースをビルドできます。また、プロジェクトに InstallScript が含まれている場合、リリースがビルドされる前に、ISCmdBld.exe によってコンパイルされます。

構文

次に、ISCmdBld.exe を実行して**オセロベータ (Othello Beta)** のリリースをビルドする際のステートメントの例を示します。

```
ISCmdBld.exe -p "C:¥InstallShield 2021 Projects¥My Othello Project¥Othello.ism" -r "Othello Beta" -c COMP -a "Build 245"
```

上記の例の -pで始まる 最初のパラメーターは、ビルドする .ism ファイルへのパスです。次に、-r Othello Beta はリリース名です。パラメーター -c COMP は、パッケージを 1 つのファイルに圧縮するかどうかを指定します。最後に、-a "Build 245" は、特定の製品構成をポイントします。

コマンドライン ビルドがエラーを発生せずに完了した場合、InstallShield によって環境変数 ERRORLEVEL が 0 に設定されます。コマンドライン ビルドの最中にエラーが発生した場合、ERRORLEVEL は 1 に設定されます。ERRORLEVEL がその他の値に設定された場合、一般的に、無効なパラメターが ISCmdBld.exe に渡されたことを示し、そのエラー原因が ISCmdBld.exe を実行中のコマンド プロンプト ウィンドウに表示されます。

コマンドライン パラメーター

プロジェクト: *一部の* ISCmdBld.exe *コマンドライン パラメーターは特定のプロジェクト タイプにのみ適用します。*

テーブル 6 : ISCmdBld.exe コマンドラインのパラメーター

パラメーター	プロジェクトの種類	説明
-a <製品構成>	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、リリースの製品構成を指定します。パラメーターが存在しない場合は、作成されます。このパラメーターは必須ではありませんが、リリース名のパラメーターがある場合はこれも含んでください。
-b <ビルドの場所>	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	出力先フォルダーとファイルの保存先フォルダーへの完全なパスです。UNC パスを使用できます。ビルド済みインストールのファイルは、指定した場所の Disk Images¥Disk1 サブフォルダーに格納されます。 このパラメーターはオプションです。指定をしなかった場合、ビルドは、ビルド パッケージとファイルを [オプション] ボックスの [ファイルの場所] タブで指定されたディレクトリに配置します。 長いファイル名は引用符で囲んでください。
-c <リリース構成>	基本の MSI、InstallScript MSI	このパラメーターを利用して、リリースを単一ファイルに圧縮するのか、圧縮せずに複数ファイルとして残すのかを指定することができます。このパラメーターの有効な引数は COMP および UNCOMP です。リリースを 1 つのファイルに圧縮するよう指定するには、引数 COMP を使用します。リリースを圧縮しない場合は、引数 UNCOMP を使用します。 パラメーターが既存するリリースで省略された場合、構成は InstallShield インターフェイスで指定されたものに基づきます。新しいリリースでパラメーターが省略された場合、ファイルは非圧縮のままとなります。

テーブル 6 ・ ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–d <変数名=プリプロセッサ定義>	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト	このパラメーターを使って、InstallScript に適切なプリプロセッサの定義を指定します。複数のプリプロセッサの定義を指定する場合、次の例のように各定義をカンマで区切ります: ISCmdBld.exe –d VARIABLENAME1=Value1,VARIABLENAME2=Value2 InstallScript でプリプロセッサを参照する場合、次のいずれかの形式を使用します: • #ifdef VARIABLENAME • #if VARIABLENAME=Value #ifdef ステートメントの場合、値名のみを使用することができます。#if ステートメントの場合、name=value ペアを使用します。 このパラメーターはオプションです。 ***メモ-*** *このコマンドライン パラメーターは、InstallShield の次の領域でプリプロセッサが定義されていない場合のみ使用することをお勧めします。* • [設定] ダイアログ ボックスの [コンパイル/リンク] タブ ([ビルド] メニューで [設定] をクリックしたとき表示されます) • 基本の MSI プロジェクトおよび InstallScript MSI プロジェクトの場合: [リリース] ビューの製品構成における "プリプロセッサ定義" 設定 *InstallShield のこれらのいずれかの領域でプリプロセッサを定義してから、–d コマンドライン パラメーターを使って、同じプリプロセッサを定義した場合、ISCmdBld.exe によって、プリプロセッサが InstallShieldプロジェクトまたは製品構成の設定を使って再定義できないことを通知する "定義不可能シンボル" メッセージが表示されます。*
–e <Y/N>	基本の MSI、InstallScript MSI、マージ モジュール	インストール プロジェクトの場合、このパラメーターは、Setup.exe ファイルをインストールと同時に 作成するかどうかを指定します。Setup.exe をビルドする –e Y 、またはインストールのみを作成する –e N を指定します。 マージ モジュール プロジェクトの場合、–e の意味はインストール プロジェクトと異なります。–e n と指定すると、マージ モジュールがビルドされ、マージ モジュール フォルダーにコピーされます。–e Y と指定すると、マージ モジュールが作成されるだけで、マージ モジュール フォルダーにはコピーされません。

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–f <リリース フラグ>	アドバンスト UI、基本の MSI、InstallScript MSI、スイート/アドバンスト UI	このパラメーターを使用して、リリースに含めるリリース フラグを指定します。複数のフラグはカンマで区切ります。
–fc <ランチャーのコピー権限>	基本の MSI、InstallScript MSI	Setup.exe のデフォルトの著作権情報を製品の著作権情報でオーバーライドするには、このパラメーターを使って製品の著作権情報を指定します (例、–fc "Copyright © 2024 Flexera.All Rights Reserved")。
–fd <ファイルの説明>	基本の MSI、InstallScript MSI	Setup.exe のデフォルトのファイルの説明を独自の必要な説明でオーバーライドするには、このパラメーターを使って –fd "セットアップ ランチャー (Unicode)" などの説明を入力します。
–fv <バージョン番号>	基本の MSI、InstallScript MSI	Setup.exe のデフォルトファイル バージョンを独自のバージョン番号でオーバーライドするには、–fv 2.5.0.0. などの適切なバージョン番号を入力します。このパラメーターはオプションです。 ***メモ:*** *ファイル バージョンは、常に 4 つのフィールドが含まれます。ファイルバージョンに 4 フィールドよりも少ないフィールドを指定すると、残りのフィールドには 0 が挿入されます。たとえば、ファイルバージョンとして 1.1 を指定すると、Setup.exe のバージョン リソースで使用されるファイル バージョンは 1.1.0.0 となります。*
–g <最小ターゲット MSI バージョン>	基本の MSI、InstallScript MSI、マージ モジュール	このパラメーターは、ターゲット マシン上でインストールで必要な Windows Installer の最小バージョンを指定します (例、2.0.2600.0)。このパラメーターはオプションです。デフォルトでは、InstallShield インターフェイスがサポートする Windows Installer の最新バージョンに設定されています。
–h	基本の MSI、InstallScript MSI	ビルドの終わりでアップグレードの検証をスキップする場合、このパラメーターを使用します。
–i <.ini ファイル パス>	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	すべてのパラメーターをコマンドラインで渡す代わりに、すべてのパラメーターを初期化 (.ini) ファイルに含めておき、その .ini ファイルをコマンドラインから呼び出すことができます。詳細については、「.ini ファイルでコマンドライン ビルド パラメーターを渡す」を参照してください。 絶対および相対パスを使用できます。 長いファイル名は引用符で囲んでください。

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
-j <最小ターゲット Microsoft .NET Framework バージョン>	基本の MSI、InstallScript MSI	このパラメーターは、ターゲット マシン上でインストールが必要な .NET Framework の最小バージョンを指定します (例、1.0.3705.2)。このパラメーターはオプションです。デフォルトでは、InstallShield インターフェイスがサポートする .NET Framework の最新バージョンに設定されています。
-I <パス変数>="新しいパス"	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	このパラメーターを使用して、[パス変数] ビューで指定されたパス変数をオーバーライドします。このパラメーターを各パス変数のオーバーライドに 1 度づつ、複数回にわたって指定できます。例: ISCmdBld.exe -I VariableName="C:¥Path" -I VariableName2="C:¥Path2" 同じ Visual Studio ソリューション フォルダー内にある姉妹プロジェクトのファイルを参照するには、VSSolutionFolder と呼ばれる定義済みのパス変数を使用します。詳細については、「Visual Studio ソリューションで VSSolutionFolder パス 変数を使用する」を参照してください。
-licCheckTimeOut <秒>	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	このパラメーターを使って、ライセンス サーバーからライセンスを使用可能かどうかチェックする時間帯を設定することができます。 たとえば、次のコマンドラインで IsCmdBld は、構成されたライセンス サーバーでライセンスが使用可能かどうかの確認を 40 秒待機した後、15 秒間隔でチェックします。ライセンスが 40 秒以内に使用可能な場合、ビルドはその使用可能なライセンスでビルドを継続し、そうでない場合は既存の理由 (ライセンスの取得に失敗) と共にビルドが失敗します。 IsCmdBld.exe -p "C:¥InstallShield 2021 projects¥TestProj.ism" -licCheckTimeOut 40
-m <.cub ファイル名>	基本の MSI、InstallScript MSI、マージ モジュール	ビルドを実行したあと、インストールまたはマージ モジュール パッケージを検証する場合、このパラメーターを使用します。.cub ファイル名のパスをこのパラメーターと共に渡します。 絶対および相対パスを使用できます。複数の .cub ファイルを使用するには、各パスをセミコロン (;) で区切ります。長いファイル名は引用符で囲んでください。 たとえば、次のコマンドラインで -m パラメーターは、検証に InstallShield 検証スイート - Windows 7 (ISWin7.cub) および完全 MSI 検証スイート (darice.cub) が必要であることを示します: ISCmdBld.exe -m "..¥Support¥Validation¥ISWin7.cub;..¥Support¥Validation¥darice.cub"

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–n	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	Setup.rul がビルド処理の一部としてコンパイルしない場合、このパラメーターを指定します。 コマンド ラインからビルドして、ビルドの設定で .ini ファイルを指定する場合、–n コマンドライン ビルド フラグを追加する代わりに、.ini ファイルの [Project] セクションで次を指定することができます。 [Project] CompileScript=No コマンドライン ビルド フラグと同様に、この CompileScript キーワードは、Setup.rul がビルド処理の一部としてコンパイルされるべきかどうかを指定するのに役立ちます。 ***メモ:*** *–n パラメーターは –q3 パラメーターと共に使用することはできません。*
–o <マージ モジュール検索パス>	基本の MSI、InstallScript、InstallScript MSI	このパラメーターは、プロジェクトで参照させるマージ モジュール (.msm) ファイルを含むフォルダーをコンマ区切りで指定します(複数指定可)。 InstallShield では、マージ モジュールを含むフォルダーを指定するその他の方法も提供されています。詳細については、「マージ モジュールを含むディレクトリを指定する」を参照してください。
–p <プロジェクトの場所>	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	プロジェクト (.ism) ファイルへのパスを渡します。このパスは、完全修飾パス、相対パス、またはただのファイル名にできます。プロジェクトのファイル名のみを渡した場合、ファイルは現在の作業ディレクトリを基準にして検索されます。例: ISCmdBld.exe –p "C:¥InstallShield 2021 Projects¥MyProject1¥MyProject1.ism" UNC パスも使用できます。長いファイル名は引用符で囲んでください。 これは唯一の必須パラメーターです。
–patch_config <パッチ構成>	基本の MSI、InstallScript MSI	このパラメーターを使用して、標準パッチをビルドすることができます。[パッチのデザイン] ビューで、ビルドするパッチ構成の名前を渡します。たとえば、次のコマンドラインは、MyProject1.ism プロジェクトにある Version 1.2 という名前のパッチ構成をビルドします: ISCmdBld.exe –p "C:¥InstallShield 2021 Projects¥MyProject1¥MyProject1.ism" –patch_config "Version 1.2"

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–prqdownload	基本の MSI、InstallScript、InstallScript MSI	このパラメーターを使って、Standalone Build でビルド中に不足している InstallShield 前提条件バイナリ ファイルを適切なプロジェクトのディレクトリに自動的にダウンロードするように設定します。たとえば、次のコマンドラインはプロジェクトで不足している InstallShield 前提条件のバイナリ ファイルをダウンロードします: Iscmdbld.exe –p "projectfile" –prqdownload ***メモ•*** *Iscmdbld.exe を使ったスイート/アドバンスト UI プロジェクトがこのパラメーター無しでビルドされると、ISM パッケージに必要な InstallShield 前提条件のバイナリ ファイルが自動的にプロジェクト ファイルにダウンロードされます。そのため、このパラメーターはスイート/アドバンスト UI プロジェクトには適用しません。*
–prqpath <InstallShield 前提条件の検索パス>	基本の MSI、InstallScript、InstallScript MSI	このパラメーターは、プロジェクトで参照される InstallShield 前提条件ファイル (.prq) ファイルを含むフォルダーをコンマで区切って指定します(複数指定可)。 InstallShield では、InstallShield 前提条件ファイル ファイルを含むフォルダーを指定するその他の方法も提供されています。詳細については、「InstallShield 前提条件を含むディレクトリを指定する」を参照してください。
–q1	基本の MSI、InstallScript MSI、マージ モジュール	リリースの Windows Installer テーブルのみをビルドします。このセットアップをまだビルドしていない場合、新しい .msi ファイルが作成されますが、インストールにはファイルが追加されません。インストールをすでにビルドしている場合、すべてのテーブルがビルドされたときに .msi ファイルが更新されますが、ファイルは転送されません。理想的には、このオプションは、インストールのユーザー インタフェースをテストするときに使用します。 ***メモ•*** *–q1 パラメーターは、–q2 または –q3 パラメーターと共に使用することはできません。*

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
-q2	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトの場合: Windows Installer テーブルをビルドしてファイルを更新します。このオプションによって、インストール内の新しいファイルや変更されたファイルを含む .msi ファイルが再ビルドされ、Files テーブルが更新されます。変更されたファイルは、サイズまたはタイムスタンプが、ビルドにすでに含まれるコピーと異なる場合にのみ更新されます。削除されたファイルへの参照は、インストールから削除されますが、ファイルは、ビルドの場所に残ります。この種類のビルドは、ビルドが完全に実行された後でのみ実行されます。また、この種類のビルドは、メディアが非圧縮のネットワーク イメージである場合だけ作動します。 InstallScript および InstallScript オブジェクト プロジェクトの場合: リースの最後にビルドされてから変更になった部分のみを再ビルドします。このパラメーターが使用されない場合、ファイル メディア ライブラリ全体が再ビルドされます。 ***メモ:*** *-q2 パラメーターは、-q1 または -q3 パラメーターと共に使用することはできません。*
-q3	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	Setup.rul のみをコンパイルします。 また、基本の MSI プロジェクトの場合、このパラメーターは ISSetup.dll を .msi パッケージの Binary テーブルにストリームします (ビルド済みの .msi パッケージが存在する場合)。 ***メモ:*** *-q3 パラメーターは、-q2、-q3 、-n パラメーターと共に使用することはできません。* ***メモ:*** *このパラメーターはプロジェクトをアップグレードしません。InstallShield Developer 8 以前で作成されたプロジェクトの場合、-q3 を使用する前に最新バージョンの InstallShield にアップグレードしてください。たとえば、-u を使ってアップグレードします。*

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–r <リリース名>	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	[リリース] ビューで指定されているリリース名です。既存のリリース名を使用するか、または新しいリリース名を作成できます。 このパラメーターは必須ではありませんが、製品構成のパラメーター –a を含める場合、これを含めます。
–s	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このパラメーターを使用すると、サイレント モードでリリースをビルドできます。サイレント ビルドは、ビルドの実行時にエラーや警告メッセージが表示されないようにする場合に便利です。 このパラメーターはオプションです。
–se または –SE	基本の MSI、InstallScript、InstallScript MSI、、スイート/アドバンスト UI、WuickPatch、マージ モジュール	このパラメーターを使って、エラー メッセージ以外の情報を表示しないでビルドを実行することができます。このパラメーターを使って、ビルド エラー (エラー、致命的なエラー、および検証エラー) のみを表示してビルドを行います。 このパラメーターはオプションです。
–t <Microsoft .NET Framework パス>	基本の MSI、InstallScript MSI	Regasm.exe と InstallUtilLib.dll は .NET Framework の各バージョンに含まれるユーティリティです。このパラメーターは、.NET インストーラー クラスおよび COM Interop を含むリリースのビルド時に使用する、これらのファイルのインストール済みの 32 ビット バージョンを含むディレクトリのパスを指定します。これは、.NET Framework 再配布可能ファイルへのパスではありません。 ***メモ:*** *InstallShield を 64 ビット システムで使用する場合、*ISCmdBld.exe *はこのパラメーターに指定されたパスに基づいて対応する .NET Framework の 64 ビット バージョンの場所を判別し、適切な場合、*ISCmdBld.exe *は* Regasm.exe *および* InstallUtilLib.dll *の 64 ビットの場所を使用します。*
–u	基本の MSI	このパラメーターを使って、リリースをビルドではなく、アップグレードすることができます。このパラメーターを使って、InstallShield Windows Installer Edition バージョン 2.03 以前で作成したインストール プロジェクトをアップグレードできます。

テーブル 6 • ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–v	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	このパラメーターを使って、詳細ビルド ログ ファイルを作成できます。ログ ファイルは、Revenera サポートがインストールに関する問題をトラブルシュートする際に役立ちます。
–w	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	このパラメーターはビルドプロセス中に発生する警告をエラーとして処理します。警告が発生するたびに、エラーカウントが 1 ずつ増加します。
–x	基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール	エラーが発生したらビルドを中止する場合は、–x パラメータを使用します。警告の遭遇時にビルドを中止する場合は、このパラメーターを –w パラメータと組み合わせて使用します。
–y <製品バージョン>	アドバンスト UI、基本の MSI、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	このパラメーターを使用して、コマンドラインから製品のバージョンを指定することができます。これは、製品バージョンのビルド バージョン (3 番目のフィールド) を増加するときに、特に便利です。たとえば、製品バージョンを 1.0.5 に設定するには、次のように入力します。 ISCmdBld.exe –y "1.0.5" 有効な製品バージョン番号については、「製品バージョンを指定する」を参照してください。 このパラメーターはオプションです。

テーブル 6 ・ ISCmdBld.exe コマンドラインのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
–z <プロパティ名=プロパティ値>	アドバンスト UI、基本の MSI、InstallScript MSI、マージ モジュール、スイート/アドバンスト UI	基本の MSI、InstallScript MSI、およびマージ モジュール プロジェクトでは、このパラメーターを使用して、Windows Installer プロパティの値をオーバーライドすることができます。アドバンスト UI およびスイート/アドバンスト UI プロジェクトでは、このパラメーターを使って、アドバンスト UI またはスイート/アドバンスト UI プロパティの値をオーバーライドできます。 たとえば、PropertyName というプロパティの値を PropertyValue に設定するには、次のコマンドラインを使用します: IsCmdBld.exe –z PropertyName=PropertyValue この値を 1 つ以上のスペースを含む文字列に設定する場合、引用符を使用します。たとえば、PropertyName というプロパティの値を My Property Value に設定するには、次のコマンドラインを使用します: IsCmdBld.exe –z "PropertyName=My Property Value" Property テーブルに存在しないプロパティを指定すると、ビルド時に、このプロパティがインストール内で作成されます。 –y パラメーターを使って製品バージョンをオーバーライドし、同時に –z パラメーターを使って ProductVersion プロパティをオーバーライドすると、InstallShield は –y パラメーターで指定された値を使用します。 このパラメーターはオプションです。
cert_password	アドバンスト UI、基本の MSI、InstallScript、InstallScript MSI、InstallScript オブジェクト、マージ モジュール、スイート/アドバンスト UI	プロジェクト内で構成されいる署名オプションに基づいて、このパラメーターは PFX デジタル証明書パスワードまたは EV/OV トークンパスワードを指定します。指定されなかった場合、ファイルの署名にはプロジェクトで構成されたパスワードが使用されます。 このパラメーターはオプションです。

ISBuild.exe

***プロジェクト・** この情報は、InstallScript プロジェクトに適用します。*

***重要・** ISBuild.exe が使用できなくなったため、ISBuild.exe の代わりに ISCmdBld.exe を使ってビルドをリリースしてください。*

ISBuild.exe は、レガシー InstallScript プロジェクトで使用できるコマンドライン ツールです。次のパラメーターが ISBuild.exe と関連付けられています。

テーブル 7 • ISBuld.exe のコマンドライン パラメーター

パラメーター	説明
–p <プロジェクト ファイル>	プロジェクトファイルの完全修飾パスとファイル名を指定します。デフォルトで、プロジェクト ファイルは次の場所に格納されます: C:¥InstallShield 2021 Projects¥*プロジェクト名*¥*プロジェクト ファイル* このパラメーターは引用符で括ります。
–m <メディア名>	ビルドされるメディアのメディア名を指定します。このパラメーターは引用符で括ります。
–b <ビルド場所>	出力先フォルダーとファイルの保存先フォルダーへの完全なパスを指定します。このパラメーターはオプションです。
–s	ISBuild.exe ファイルをサイレント モードで実行します。このパラメーターはオプションです。
–r	メディアでビルドを更新するよう指定します。デフォルトは完全な再ビルドです。このパラメーターはオプションです。

IISscan.exe

エディション: *IIS データを InstallShield プロジェクトにインポートする機能は、InstallShield Premier Edition のみで使用できます。*

IIS スキャナー (IISscan.exe) は、IIS Web サイトをスキャンして、InstallShield の [IIS 構成] ビューで構成可能な設定の値を記録するツールです。IISscan.exe は、すべての値を含む XML ファイルを作成します。この XML ファイルを使って、[IIS 構成] ビューに値をインポートできます。次に、必要に応じて設定を変更できます。

IISscan.exe を使って Web サイトをスキャンするとき、IISscan.exe ファイルを IIS Web サイトが存在するマシン上に配置しなくてはなりません。

メモ: *IISscan.exe には、管理者権限が必要です。したがって、管理者特権で [コマンドライン プロンプト] ウィンドウからこれを起動する必要があります。*

IISscan.exe は、以下の場所にインストールされています:

InstallShield Program Files フォルダー¥System

構文

iisscan.exe –website "Site Name" –outfile "C:¥PathToFile¥FileName.xml"

パラメーター

テーブル 8 • IISscan.exe のコマンドライン パラメーター

パラメーター	説明
–website	スキャンする IIS サーバー上の Web サイトの名前を指定します。
–outfile	スキャナーが作成する XML ファイルの完全パスと名前を指定します。 このパラメーターはオプションです。このパラメーターを渡さなかった場合、IIS スキャナーは同じディレクトリにある iisscan.xml と名付けられたファイルに IIS 設定を記録します。

ヒント• *InstallShield では、特定の IIS データ (たとえば Web サイト、アプリケーション、仮想ディレクトリ、アプリケーション プール、またはそれらの設定) が [IIS 構成] ビューにインポートされないように防止するためのフィルターを構成できます。詳細については、「Web サイトおよびその設定を InstallShield プロジェクトにインポートするときに IIS データをフィルターする」を参照してください。*

iSign.exe

プロジェクト• *この情報は、次のプロジェクト タイプに適用します:*

- *InstallScript*
- *InstallScript オブジェクト*

コマンドライン ツール iSign.exe を使って、リリースのビルド後にコマンドラインを使って InstallScript プロジェクトのリリースにデジタル署名を行います。

iSign.exe は、次のディレクトリに保存されています:

InstallShield Program Files フォルダー¥System

iSign.exe では、次のオプションから選択できます:

- 使用中のマシンにある、署名に使用する .pfx 証明書ファイルを指定できます。
- 署名に使用する証明書を含む証明書ストアを参照できます。

構文

iSign.exe [options] *PathToData1.hdrFile*

たとえば、.pfx ファイルを使って署名する場合:

iSign.exe –pfx "C:¥Temp¥MyFile.pfx" –p "Password" "C:¥InstallShield 2025¥MyProject¥Media¥Release 1¥Disk Images¥Disk1¥data1.hdr"

証明書ストアにある証明書を使って書名する場合:

iSign.exe –store "Root" –sl "Machine" ss "My Certificate" "C:¥InstallShield 2025¥MyProject¥Media¥Release 1¥Disk Images¥Disk1¥data1.hdr"

コマンドライン パラメーター

以下は iSign.exe で使うことができるパラメーターのリストです: .pfx ファイル使用時にのみ有効なパラメーターと、証明書ストアにある証明書使用時にのみ有効なパラメーターがあります。

テーブル 9 • iSign.exe のコマンドライン パラメーター

パラメーター	署名メソッド	説明
–pfx <.pfx ファイルへのパス>	.pfx ファイル	リリースの署名に使用する証明書を含む .pfx ファイルへの完全修飾パスを指定します。
–p <パスワード>	.pfx ファイル	使用する .pfx にパスワードがある場合、それを指定します。 証明書がパスワードで保護されているのにもかかわらず、このパラメーターに何も入力しなかった場合、.pfx ファイルを使った署名が失敗します。 ***メモ:*** *署名に使用する証明書が証明書ストアにある場合、このパラメーターは適用しません。証明書がパスワード保護付きでストアにインポートされている場合、*iSign.exe *の署名時にそのパスワードがプロンプトされます。*
–store <ストア名>	証明書ストア内の証明書	使用する証明書を含む証明書ストアの名前を指定します。選択可能なオプションは以下のとおりです: • My • Root • Trust • CA
–sl <ストアの場所>	証明書ストア内の証明書	使用する証明書を含む証明書ストアの場所を指定します。選択可能なオプションは以下のとおりです: • ユーザー • マシン
–ss <サブジェクト>	証明書ストア内の証明書	使用する証明書のサブジェクトを指定します。このパラメーターは、証明書が証明書ストアにある場合に適用します。

ReleasePackager.exe

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

ReleasePackager.exe を使用して、コマンドラインから自己展開型実行可能ファイルをビルドすることができます。バッチ ファイルからビルドを行うときに便利です。

ReleasePackager.exe は、以下の場所にインストールされています:

InstallShield Program Files フォルダー¥System

構文

ReleasePackager.exe *"disk_images_folder" "package_file"* ["*icon_file*" [*icon_index*]]

パラメーター

テーブル 10 • ReleasePackager.exe のコマンドライン パラメーター

パラメーター	説明
disk_images_folder	このオプションのパラメーターは、自己展開型実行可能ファイルに組み込まれるディスクイメージを含むフォルダーを指定します。通常の値は、 C:¥InstallShield 2021 Projects¥ProjectName¥Media¥Releasename¥Disk Images
package_file	このパラメーターは、自己展開型実行可能ファイルの完全修飾名を指定します。通常の値は、 C:¥InstallShield 2021 Projects¥Projectname¥Media¥Releasename¥Package¥Packagename.exe
icon_file	このオプションのパラメーターは、自己展開型実行可能ファイル用に使用するアイコンを含むファイルの完全修飾名を指定します。
icon_index	このオプションのパラメーターは、自己展開型実行可能ファイル用に使用する *icon_file* からのアイコンの数値インデックスを指定します。インデックスを指定しない場合、インデックス 0 がそのアイコンに使用されます。
–s	このオプションのパラメーターは、ReleasePackager.exe をサイレント モードで実行するのに使用できます。この新しいパラメーターを ReleasePackager.exe に渡すには、次の構文を使用します: ReleasePackager.exe "disk_images_folder" "package_file" ["icon_file" [icon_index]] –s –s パラメーターは他のパラメーターの後に指定される必要がありますので注意してください。アイコン ファイルまたはアイコン インデックスを指定しない場合、アイコン ファイルのデフォルトにヌル文字列 (""")、およびアイコン インデックスに 0 を指定しなくてはなりません。

例

ReleasePackager.exe "C:¥InstallShield 2021 Projects¥My Project¥Media¥My Release¥Disk Images" "C:¥InstallShield 2021 Projects¥My Project¥Media¥My Release¥Package¥MyPackage.exe" "C:¥My Icon Files¥MyIcons.dll" 2 –s

SetupIni.exe

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *基本の MSI*

- *InstallScript MSI*

SetupIni.exe は、Setup.exe または Update.exe ファイルに埋め込まれている Setup.ini ファイルを変更できるコマンドライン ツールです。Setup.ini は、インストール プロジェクトのビルド プロセス中にインストールの一部の要素を制御するために作成される初期化ファイルです。

SetupIni.exe は、以下の場所にインストールされています:

InstallShield Program Files フォルダー¥System

構文

SetupIni.exe *PathToSetup.exe Section Key Value*

パラメーター

テーブル 11 ・ SetupIni.exe のコマンドライン パラメーター

パラメーター	説明
PathToSetup.exe	変更する Setup.ini ファイルを含む Setup.exe ファイルの名前とパスを指定します。 パスにスペースが含まれている場合は、引用符で囲みます。
Section	変更するセクションの名前を指定します。
Key	値を読み取るプロパティの名前を指定します。
Value	指定したキーに使用する値を指定します。 値にスペースが含まれている場合は、引用符で囲みます。

例

SetupIni.exe "C:¥InstallShield 2021 Projects¥My Project¥Media¥My Release¥Package¥MySetup.exe" Startup CmdLine MYPROPERTY="MyValue"

MsiExec.exe コマンドラインのパラメーター

MsiExec.exe は Windows Installer の実行可能プログラムで、インストール パッケージの解析とターゲット システムへの製品のインストールに使用されます。リリースのビルド完了後に、Windows Installer パッケージ (.msi) をコマンドラインからインストールできます。

作成した Windows Installer パッケージは、ビルドしたリリースを含むフォルダーからアクセスすることができます。次は、デフォルト保存先です。

C:¥InstallShield 2021 プロジェクト¥*プロジェクト名*¥*リリース名*¥DiskImages¥Disk1¥*製品名*.msi

製品のリリースをビルドした後で、コマンドラインからリリースをインストールすることができます。

msiexec /i "C:¥InstallShield 2021 プロジェクト¥*プロジェクト名*¥*リリース名*¥DiskImages¥Disk1¥*製品名*.msi"

次の表は、MsiExec.exe のコマンドライン パラメーターについて詳しく説明します。

テーブル 12 • MsiExec.exe コマンドライン パラメーター

パラメーター	説明
/i <パッケージ> または <製品コード>	たとえば、オセロ (Othello) という製品をインストールするには、次の形式を使用します。 msiexec /i "C:¥InstallShield 2021 Projects¥Othello¥Trial Version¥Release¥DiskImages¥Disk1¥Othello Beta.msi" 製品コードは、製品の [プロジェクト] ビューの "製品コード" プロパティで自動生成された GUID を参照します。
/f [p|o|e|d|c|a|u|m|s|v] <パッケージ> または <製品コード>	/fオプションを使用してインストールすると、存在しないファイルや破損したファイルは再インストールまたは修復されます。 たとえば、すべてのファイルを強制的に再インストールするには、次の構を利用します。 msiexec /fa "C:¥InstallShield 2021 Projects¥Othello¥Trial Version¥Release¥DiskImages¥Disk1¥Othello Beta.msi" 上記の構文を次のフラグと合わせて使用します。 • p ファイルが存在しない場合、再インストールします。 • o ユーザーのシステムにファイルが存在しない場合、または古いバージョンのファイルが存在する場合、ファイルを再インストールします。 • e ユーザーのシステムにファイルが存在しない場合、あるいは同等または古いバージョンのファイルが存在する場合、ファイルを再インストールします。 • c ファイルが存在しない場合、またはインストールされたファイルの格納済みチェックサムが新規ファイルの値と一致しない場合、ファイルを再インストールします。 • a – すべてのファイルを強制的に再インストールします。 • u または m – 必要なユーザー レジストリ エントリをすべて上書きします。 • s – 既存のショートカットをすべて上書きします。 • v – ソースからアプリケーションを実行し、ローカル インストール パッケージを再キャッシュします。
/a <パッケージ>	/a オプションを使用すると、管理者権限を持つユーザーはネットワークに製品をインストールできます。

テーブル 12・MsiExec.exe コマンドライン パラメーター (続き.)

パラメーター	説明
/x <パッケージ> または <製品コード>	/x オプションを使用して製品をアンインストールします。 ***プロジェクト:*** *この情報は、InstallScript UI に新しいスタイル (InstallScript エンジンを埋め込み UI ハンドラーとして使用するスタイル) が使用されている InstallScript MSI プロジェクトには適用しません。* Msiexec.exe /x {ProductCode} *ステートメントを使ってコマンドラインから製品のアンインストールを試みると、アンインストールが成功することがあります。ただし、アンインストールの終わりのほうで、Windows Installer によってエラー ダイアログが表示されます。エラー ダイアログでは、Windows Installer サービスにアクセスできなかったというメッセージが表示されます。* *このシナリオでコマンドラインからアンインストールを実行する場合、次のいずれかのメソッドを実行することが現在推奨されています:* msiexec.exe /i {ProductCode} REMOVE=ALL msiexec.exe /x {ProductCode} /qn *InstallScript MSI で、InstallScript UI に従来型のスタイル (InstallScript エンジンを 外部 UI ハンドラーとして使用するスタイル) が使用されている場合、これは必要ありません。* *詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。*
/j [u|m] <パッケージ> **/j [u|m] <パッケージ> /t <トランスフォーム リスト>** **/j [u|m] パッケージ /g** **/j <言語 ID>**	/j <パッケージ> オプションを使ってビルドすると、アプリケーションのコンポーネントがエンドユーザーのコンピューターにアドバタイズされます。 ・ u 現在のユーザーだけにコンポーネントをアドバタイズします。 ・ m コンピューターの全ユーザーにコンポーネントをアドバタイズします。 ・ g 言語 ID を指定します。 ・ t アドバタイズされた製品にトランスフォームを適用します。 トランスフォームを実行すると、異なる言語のアプリケーションの同期を取ることができます。たとえば、英語版の製品をアップグレードする場合、トランスフォームを適用してフランス語版の製品を自動的にアップグレードできます。

テーブル 12 • MsiExec.exe コマンドライン パラメーター (続き.)

パラメーター	説明
/L[i|w|e|a|r|u|c|m|o|p|v|x|+|!|*] <ログ ファイル>	/L オプションを使うと、ログ ファイルのパスを指定できます。オプションで、次のフラグを使用して、ログ ファイルに記録する情報を指定することもできます (複数可): • i – ステータス メッセージを記録します。 • w – 致命的な警告メッセージ以外の警告メッセージをログに記録します。 • e – すべてのエラー メッセージをログに記録します。 • a – アクション シーケンスの開始をログに記録します。 • r –アクション固有のレコードをログに記録します。 • u – ユーザー リクエストを記録します。 • c – 初期ユーザー インターフェイス パラメーターをログに記録します。 • m – メモリ不足を示すメッセージをログに記録します。 • o – ディスク容量不足のメッセージをログに記録します。 • p – インストールの終わりで、すべてのプロパティとその値をログに記録します。 • v – 詳細出力をログに記録します。 • x – 追加デバッグ情報をログに記録します。 • + – 既存のログ ファイルに情報を追加します。 • ! – 各行をログにフラッシュします。 • * は、すべての情報をログに記録できるようにするワイルドカード記号です (v で提供される詳細出力と、x で提供される追加デバッグ情報は除外されます)。 たとえば、すべてを詳細に記録するログ ファイルを作成する場合は、この設定で、次を使用します: /L*v "C:¥MyLogFiles¥package.log"
/p <パッチパッケージ>	パッチをインストール済みの製品に適用するとき、/p オプションを使用します。インストール済みの管理者イメージにパッチを適用するには、このオプションを次のように /a と組み合わせて使用します。 */p パッチ パッケージ /a パッケージ*

テーブル 12 • MsiExec.exe コマンドライン パラメーター (続き.)

パラメーター	説明
/q [n|b|r|f]	/q オプションは、以下のフラグと組み合わせてユーザー インターフェイス レベルを設定するために使用します。 • q または qn – ユーザー インターフェイスは作成されません。 • qb – 基本的なユーザー インターフェイスが作成されます。 以下のユーザー インターフェイス設定を行うと、インストールの最後にモーダル ダイアログが表示されます。 • qr – 簡易ユーザー インターフェイスが表示されます。 • qf – 完全なユーザー インターフェイスが表示されます。 • qn+ – ユーザー インターフェイスは表示されません。 • qb+ – 基本的なユーザー インターフェイスが作成されます。
/? または /h	どちらのコマンドでも、Windows Installer の著作権情報が表示されます。
/y <ファイル名>	このコマンドは、<ファイル名> で指定された DLL または OCX ファイルの DllRegisterServer エントリ ポイントの関数を呼び出します。
/z <ファイル名>	このコマンドは、<ファイル名> で指定された DLL または OCX ファイルの DllUnregisterServer エントリ ポイントの関数を呼び出します。
/n {製品コード}	/n <製品コード> オプションは、製品の複数インスタンスのインストールをサポートする基本の MSI プロジェクトの [パッチのデザイン] ビューで作成されたパッチについて使用できます。 パッチを適用するインスタンスの製品コードを指定する場合、/n オプションを /p オプションと一緒に使用します。例: msiexec /p mypatch.msp /n {00000001-0002-0000-0000-624474736554}
TRANSFORMS	TRANSFORMS コマンドライン パラメーターを使用して、基本パッケージに適用するトランスフォームを指定します。以下に TRANSFORM コマンドラインの呼び出し方法の例を示します。 msiexec /i "C:¥InstallShield 2021 Projects¥Othello¥Trial Version¥Release¥DiskImages¥Disk1¥Othello.msi" TRANSFORMS="New Transform 1.mst" 複数のキーワードはセミコロンで区切ります。トランスフォームの名前にセミコロンを使用すると、Windows Installer サービスで正しく解釈されない可能性があるため、トランスフォーム名にはセミコロンを使用しないことをお勧めします。

テーブル 12・MsiExec.exe コマンドライン パラメーター (続き.)

パラメーター	説明
Properties	すべての パブリック (public) プロパティはコマンドラインから設定または変更できます。パブリック プロパティは、すべて大文字で表記されている点がプライベート プロパティと異なります。たとえば、COMPANYNAME はパブリック プロパティです。 コマンドラインからプロパティを設定するには、構文 *PROPERTY=VALUE* を使用します。たとえば、COMPANYNAME の値を変更する場合、次のように入力します。 msiexec /i "C:¥InstallShield 2021 Projects¥Othello¥Trial Version¥Release¥DiskImages¥Disk1¥Othello.msi" COMPANYNAME="ソフトウェア会社名"

MsiExec.exe コマンドライン パラメーターを Setup.exe を通して渡す方法については、「Setup.exe および Update.exe コマンドライン パラメーター」を参照してください。

Command-Line Options (Windows Installer ヘルプ ライブラリ)
Standard Installer Command-Line Options (Windows Installer ヘルプ ライブラリ)

Setup.exe および Update.exe コマンドライン パラメーター

Setup.exe には、多くのコマンドライン パラメーターを使用できます。Update.exe (基本の MSI と InstallScript MSI プロジェクトでのみ使用可能) は、ほとんどすべての同じコマンドライン パラメーターを受け入れます。エンドユーザーは、これらのパラメーターを使用して、インストール実行時の言語や Setup.exe をサイレント モードで起動するかどうかなどのデータを指定することができます。基本の MSI および InstallScript MSI の場合、エンドユーザーは Setup.exe を使って、そこに含まれている .msi ファイルにパラメーターを渡すことも可能です。

メモ・ *パラメーターを必要とするコマンドライン オプションを指定する場合、オプションとパラメーターの間にスペースを入れることはできません。たとえば、Setup.exe /v"ALLUSERS=2" は有効ですが、Setup.exe /v "ALLUSERS=2"は無効です。パラメーターにスペースが含まれている場合にのみ、オプションのパラメーターを引用符で括る必要があります。パラメーター内のパスがスペースを含む場合は、次の例のように引用符内に引用符を使用する必要があります。*

Setup.exe /v"INSTALLDIR=¥"c:¥My Files¥""

プロジェクト・ *一部のコマンドライン オプションは、特定の種類のプロジェクトにのみ適用します。プロジェクト固有の情報は、オプションごとに表示されています。*

ビルトイン コマンドライン パラメーター

このセクションでは、Setup.exe の有効なコマンドライン パラメーターについて説明します。パラメーターは、次のカテゴリに分けられます:

- サイレント インストール
- 特殊インストール モード

- インストールにデータを渡す
- ダウンロードとキャッシュの場所 (基本の MSI および InstallScript MSI プロジェクト)
- デバッグ
- SMS データ
- 最小 UI
- その他

サイレント インストール

テーブル 13 • サイレント インストールのパラメーター

パラメーター	プロジェクトの種類	説明
/p"パスワード"	基本の MSI、InstallScript MSI	[リリース] ビューの Setup.exe タブで、リリースのパスワード関連設定を構成した場合、エンド ユーザーはインストールをサイレントで実行する時に /p オプションを使ってパスワードを指定しなくてはなりません。典型的なコマンドは、次のとおりです。 Setup.exe /s /p"パスワード"
/r	InstallScript、InstallScript MSI	このコマンドラインを使って、インストールをレコード モードで実行します。 InstallScript MSI または InstallScript インストールをサイレント モードで実行するには、/r オプションを使って最初に Setup.exe を実行して応答ファイルを作成する必要があります。このファイルには実行時にユーザーが入力したデータと選択したオプションについての情報が保存されます。 /r パラメーターを使って InstallScript MSI または InstallScript インストールを起動すると、すべての実行時ダイアログが表示され、そのデータはシステムの Windows フォルダー内に作成される Setup.iss というファイルに保存されます。代替応答ファイル名と場所を指定するには、以下に示す /f1 オプションを使用してください。 基本の MSI プロジェクトでは、応答ファイルはサイレント インストール用に作成または使用されません。

テーブル 13 • サイレント インストールのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/s	基本の MSI、InstallScript、InstallScript MSI	InstallScript MSI または InstallScript プロジェクトでは、Setup.exe /s コマンドは、同じディレクトリにある Setup.iss という応答ファイルに含まれる応答に基づいて、デフォルトでインストールをサイレント モードで実行します。(応答ファイルは /r オプションで Setup.exe を実行すると作成されます。)応答ファイルの代替ファイル名と場所を指定するには、/f1 オプションを使用してください。 また、Setup.exe /s のコマンドは、基本の MSI インストールで Setup.exe 初期化ダイアログの抑制もしますが、応答ファイルは読み込みません。基本の MSI インストールをサイレントで実行するには、次のコマンドラインを使用します: Setup.exe /s /v/qn サイレントの基本の MSI インストールでパブリック プロパティの値を指定するには、次のようなコマンドを使用します: Setup.exe /s /v"/qn INSTALLDIR=D:¥インストール先" ***メモ:*** *このコマンドライン パラメーターを使って InstallShield 前提条件を含むインストールを起動した場合、前提条件のインストールは自動的にサイレントで実行されません。必要に応じて、InstallShield 前提条件エディターにある [実行するアプリケーション] タブの* ***"セットアップがサイレント モードで実行中にアプリケーションのコマンドラインを指定する"*** *設定で、InstallShield 前提条件用の有効なサイレント コマンドライン パラメーターを指定してください。* *詳細については、「InstallShield 前提条件のコマンドライン パラメーターを指定する」を参照してください。*
/f1	InstallScript、InstallScript MSI	/f1 オプションを使うと、次の例のように、応答ファイルの場所 (または作成場所) と名前を指定することができます: Setup.exe /s /f1"C:¥Temp¥Setup.iss" 絶対パスを指定します。相対パスを使用すると予測できない結果が生じます。/f1 オプションは、応答ファイルの作成時 (/r オプションを使用) と応答ファイルの使用時 (/s オプションの使用) の両方で使用できます。

テーブル 13 • サイレント インストールのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/f2	InstallScript、InstallScript MSI	(/s オプションを使用して) サイレント モードで InstallScript MSI または InstallScript プロジェクト インストール プログラムを実行すると、デフォルトでは、応答ファイルと同じディレクトリに同じ名前(但し、拡張子は異なります)でSetup.log というログ ファイルが作成されます。/f2 オプションを使うと、次の例のように代替ログファイル場所とファイル名を指定することができます: Setup.exe /s /f2"C:¥Setup.log" 絶対パスを指定します。相対パスを使用すると予測できない結果が生じます。

特殊インストール モード

テーブル 14 • 特殊インストール モードのパラメーター

パラメーター	プロジェクトの種類	説明
–a	基本の MSI、InstallScript MSI	*メモ• /a パラメーターは Update.exe と一緒に使用できません。Update.exe は、システムにキャッシュされている既存の MSI にアクセスおよび修正するパッチを起動し、管理インストールは .msi ファイルをキャッシュしません。* /a オプションを使うと、Setup.exe で管理インストールが実行されます。管理インストールはデータファイルをユーザーが指定したディレクトリにコピー（および展開）しますが、ショートカットの作成や COM サーバーの登録またはアンインストールのログ記録は行いません。 *ヒント• インストールが InstallShield 前提条件を含んでいて、それを Setup.exe から抽出したい場合、パスを /a パラメーターのあとに追加すると、その場所へ前提条件が抽出されます。サンプル コマンドは、Setup.exe /a"C:¥temp" です。*
/j	基本の MSI、InstallScript MSI	/j オプションを使うと、Setup.exe でアドバタイズ インストールが実行されます。アドバタイズインストールはショートカットを作成し、COM サーバーとファイルの種類の登録を行いますが、ユーザーがこれらの「エントリポイント」の 1 つを起動するまで製品ファイルをインストールしません。
/x	基本の MSI、InstallScript MSI	/x オプションを使うと、Setup.exe は以前インストールされた製品をアンインストールします。

テーブル 14 • 特殊インストール モードのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/uninst	InstallScript MSI (InstallScript ユーザー インターフェイス (UI) のスタイルが、外部 UI ハンドラーとして InstallScript エンジンを使用する従来型のスタイルの場合)	 ***重要•*** *このパラメーターは、InstallScript UI スタイルが (埋め込み UI ハンドラーとして InstallScript エンジンを使用する) 新しいスタイルである InstallScript MSI プロジェクトには適用しません。詳細については、「InstallScript MSI インストールで InstallScript エンジンを外部エンジンとして使用する方法と、埋め込み UI ハンドラーとして使用する方法の違い」を参照してください。* /uninst オプションは、以前のバージョンの InstallShield との互換性を目的として提供されています。このオプションを使用すると、InstallScript エンジンはイベント ハンドラー関数 OnUninstall を実行します。この関数のデフォルト コードは以前にインストールされた製品をアンインストールします。 ***プロジェクト•*** *InstallScript MSI インストールでは、/uninst パラメーターがインストールをログを処理して、スクリプトを使って作成およびログ記録された任意のリソースを削除する InstallScript エンジンのアンインストール動作を呼び出すことはありません。その代わりに、/removeonly オプションがこの動作を呼び出します。*
/removeonly	InstallScript、InstallScript MSI	/removeonly オプションは REMOVEONLY システム変数をゼロ以外の値に設定します。OnMaintUIBefore イベント ハンドラー関数のデフォルト コードは、REMOVEONLY の値に従って条件付で **SdWelcomeMaint** ダイアログを表示します。

テーブル 14 • 特殊インストール モードのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/instance=<InstanceId>	基本の MSI	/instance=<InstanceId> オプション (製品の複数インスタンスのインストールをサポートする基本の MSI プロジェクトで提供されています) を使って、インストール、更新、またはアンインストールするインスタンスを指定することができます。*<InstanceId>* は、インスタンスを識別する InstanceId プロパティの値を示します。このオプションを使用して、有効な InstanceId 値を含めると、インストールでインスタンスの選択ダイアログが抑制されます。 たとえば、次のサンプル コマンドラインは InstanceId プロパティの値が 2 であるインスタンスをインストールします: Setup.exe /instance=2 次のように*デフォルト*を使用すると、ベースのインストール パッケージによってインストールされたインスタンスが識別されます: Setup.exe /instance=default アンインストールするインスタンスを指定する場合、/x オプションを /instance=<InstanceId> オプションと共に含めます。 詳細については、「インスタンスの名前付け」および「製品のインスタンスを複数インストールするときの実行時の動作」を参照してください。

インストールにデータを渡す

テーブル 15 • データをインストールに渡すためのパラメーター

パラメーター	プロジェクトの種類	説明
/v	基本の MSI、InstallScript MSI	/v オプションを使って、Msiexec.exe にコマンドライン オプションおよびパブリック プロパティの値を渡すことができます。 Msiexec.exe に複数の引数を渡す場合、コマンドラインで /v オプションを複数回 (1 引数につき 1 回) 使用することができます。例: Setup.exe /v"/l*v c:¥test.log" /v"MYPROPERTY1=value1" /v"/qb" また、次の例のように、/v オプションを使って、複数の引数を渡すこともできます: Setup.exe /v"/l*v c:¥test.log MYPROPERTY1=value1 /qb" ***メモ*** *Setup.exe を起動するときにコマンドライン プロンプトで /v パラメーターを使用する場合、Setup.ini にある CmdLine キー名に指定されたパラメーターはすべて無視されます。詳細については、「Setup.ini」を参照してください。*
/v"ISSCRIPTCMDLINE=¥"<Option1> <Option2>¥""	InstallScript カスタム アクションを含む基本の MSI プロジェクト	このオプションは、スクリプトへ渡すコマンドライン パラメーターを指定します。(適切な場所で) InstallScript MSI でサポートされているプロパティをすべて指定することができます。(もっとも一般的なのは /d と /z です。) 次の例は、スクリプトのデバッグを行う時に、*CMDLINE* 変数に TEST を含めることを示します。 Setup.exe /v"ISSCRIPTCMDLINE=¥"-d -zTEST¥"" 上記のように、二重引用符がコマンドラインの区切り記号ではなく、プロパティの区切り記号であることを示すには、¥"を使用してください。 また、Windows Installer のパブリック プロパティと同様に、このプロパティはすべて大文字で指定しなくてはなりません。

テーブル 15 • データをインストールに渡すためのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/z	InstallScript MSI	/z オプションは、次の例のように、InstallScript システム変数の CMDLINE に、データを渡すのに使用します: Setup.exe /z"カスタム データ" このコマンドラインを使用するとき、*CMDLINE* 変数には "マイ カスタム データ" の文字列が含まれます。

ダウンロードとキャッシュの場所 (基本の MSI および InstallScript MSI プロジェクト)

テーブル 16 ・ ダウンロードおよびキャッシュする場所のパラメーター

パラメーター	プロジェクトの種類	説明
/ua **/uw**	基本の MSI、InstallScript MSI	リリース ウィザードで、Windows Installer のインストーラー (InstMsiA.exe と InstMsiW.exe) をダウンロードする場所を指定できます。ユーザーは以下の例のように、/ua オプション (InstMsiA.exe の場合) または /uw オプション (InstMsiW.exe の場合) を使って実行時の代替 URL を指定することができます: Setup.exe /uw"http://www.otherlocation.com/engines" ファイル名は必要ありません。 ***メモ・** このパラメーターと共に完全 URL を指定しなくてはなりません。*
/um	基本の MSI、InstallScript MSI	リリース ウィザードを使うと、Web ダウンローダーのビルドで .msi データベースのダウンロード場所を指定することができます。ユーザーは次の例の要領で /um を使って、代替の URL を指定できます。 Setup.exe /um"http://www.otherlocation.com/packages/product.msi" ***メモ・** このパラメーターと共に完全 URL を指定しなくてはなりません。*
/b	基本の MSI、InstallScript MSI	リリース ウィザードを使うと、ダウンローダーのビルドでローカルシステム上に 圧縮パッケージの内容をキャッシュ するかどうかを指定することができます。/b オプションを使うと、ユーザーは次の例の要領で、インストール ファイルをキャッシュするディレクトリを指定することができます。 Setup.exe /b"C:¥CacheDirectory" ***メモ・** このパラメーターと共に完全 URL を指定しなくてはなりません。*

デバッグ

テーブル 17 • デバッグのためのパラメーター

パラメーター	プロジェクトの種類	説明
/d	InstallScript カスタム アクションがある基本の MSI プロジェクト、InstallScript、InstallScript MSI	InstallScript プロジェクトで、Setup.exe /d のコマンドを実行すると、InstallScript デバッガーを持つインストール プログラムが実行されます。 ***メモ•*** *InstallScript コードをデバッグするには、デバッグ情報ファイルの* Setup.dbg *が使用可能である必要があります。また、開発マシン以外のシステムでインストールをデバッグする場合、特定のファイルを開発マシンからデバッグ マシンにコピーする必要があります。詳細については、「任意のコンピューターでのインストールのデバッグ」を参照してください。* 基本の MSI プロジェクトでは、InstallScript デバッガーで次のコマンドが InstallScript カスタム アクションを実行します: Setup.exe /v"ISSCRIPTDEBUG=1 ISSCRIPTDEBUGPATH=¥"path-to-Setup.dbg¥"""

テーブル 17 • デバッグのためのパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/debuglog	基本の MSI、InstallScript MSI	/debuglog パラメーターを使って、Setup.exe のログ ファイルを生成できます。 長い名前を持つ InstallShield.log を Setup.exe ファイルと同じディレクトリに生成するには、コマンドライン パラメーターのみを渡します。Setup.exe ファイルが読み取り専用の場所にあるとき、この処理は実行できません。例: setup.exe /debuglog ログ ファイルの名前と場所を指定するには、パスと名前を次の要領で渡します。 Setup.exe /debuglog"C:¥PathToLog¥setupexe.log" インストールに含まれる機能前提条件にログ ファイルを生成するには、/v パラメーターを使って、次の例のように ISDEBUGLOG プロパティをログ ファイルの完全パスとファイル名を設定します: Setup.exe /debuglog"C:¥PathToSetupLogFile¥setup.log" /v"ISDEBUGLOG=prereq.log" 機能前提条件のログ ファイルのパスには、ディレクトリ プロパティおよび環境変数を使用できます。 さらに、/debuglog パラメーターと共に /newlog オプションを使用するか、IDE 設定の **[デバッグ ログを常に作成する]** オプションを [はい] に設定し、既存のデバッグログを上書きして新しいログを作成します。 Setup.exe ファイルと同じディレクトリにある InstallShield.log という名前の既存のログを上書きするには、コマンドライン パラメーターのみを渡します。たとえば、setup.exe /debuglog /newlog。 上書きする既存のログ ファイルの名前と場所を指定するには、パスと名前を次の要領で渡します: setup.exe /debuglog"C:¥PathToLog¥setupexe.log" /newlog ***メモ:*** *「セットアップ コマンドラインが無効です。セットアップを続行できません。」というメッセージが、/debuglog パラメーターと共に /newlog オプションが使用されなかった場合、または* ***[デバッグ ログを常に作成する]*** *オプションが [はい] に設定されなかった場合に表示されます。*

SMS データ

テーブル 18 • SMS データのパラメーター

パラメーター	プロジェクトの種類	説明
/m	InstallScript、InstallScript MSI	/m オプションを使うと、Setup.exe で SMS .mif (Management Information Format) ファイルが生成されます。典型的なコマンドは次のとおりです。 Setup.exe /m"SampleApp" ファイル拡張子 ".mif" を含める必要はありません。
/m1	InstallScript、InstallScript MSI	/m1 パラメーターを (/m と一緒に) 使うと、.mif ファイルに書き込むシリアル番号を指定できます。典型的なコマンドは、次のとおりです。 Setup.exe /m"SampleApp" /m1"1234-5678"
/m2	InstallScript、InstallScript MSI	/m2 パラメーターを (/m と一緒に) 使うと、.mif ファイルに書き込むロケール文字列を指定できます。典型的なコマンドは、次のとおりです。 Setup.exe /m"SampleApp" /m2"ENU"

最小 UI

テーブル 19 • 最小 UI のパラメーター

パラメーター	プロジェクトの種類	説明
/passive	アドバンスト UI、スイート/アドバンスト UI	/Passive パラメーターを使って、スイート インストーラー セットアップを最小 UI モード ([進行状況] パネルのみを表示) で実行します: Setup.exe /passive 最小 UI モードを使ってアンインストールするには、次のコマンドを使用します: Setup.exe /passive /remove

その他

テーブル 20 • その他のパラメーター

パラメーター	プロジェクトの種類	説明
/delayedstart:<秒数>	InstallScript	Setup.exe が起動した後にインストールの初期化を遅延させる時間 (秒単位) を指定します。 ***メモ:*** *(たとえば RunOnce キーを使って) 再起動の後、手動で追加インストールを起動する際は -delayedstart オプションの利用をお勧めします。遅延させることで OS が完全に初期化できるようにします。これによってシステムが完全に初期化される前にインストールが初期化された場合に発生する RPC (リモートプロシージャ呼び出し) エラーなどの問題を回避することができます。推奨される遅延時間は 30 秒です。* *このオプションはインストールが再起動の後に自動的に開始される場合には不要です (たとえば、再起動の前に* **SdFinishReboot** *を呼び出した場合)。*
/runfromtemp	基本の MSI、InstallScript、InstallScript MSI	このパラメーターを使うと、セットアップが一時ディレクトリから実行するための条件に適合しない場合でも、セットアップ作成者は常にセットアップを一時保管ディレクトリにクローンして、そこから実行することができます。セットアップが自己展開型の実行可能ファイル (.exe) の場合、このパラメーターは無視されます。
/Clone_wait	InstallScript、基本の MSI、InstallScript MSI、スイート/アドバンスト UI	このパラメーターは、クローンされたセットアップの処理が完了するまでオリジナル セットアップが待機してから終了することを意味します。 デフォルトでは、元のセットアップがインストール処理の完了を待機するように設定されています。 Clone_wait についての詳細は、「セットアップのデフォルト Clone_wait パラメーターを変更する」を参照してください。
/extract_all:<パス>	InstallScript	自己展開型パッケージファイルを実行せずに、<パス>で指定された場所に抽出するように指定します。典型的なコマンドは、次のとおりです。 Setup.exe /extract_all:"C:/ProductName セットアップファイル"

テーブル 20 ・ その他のパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/h	基本の MSI、InstallScript MSI	ビルドエンジンは、複製が必要な場合 Setup.exe の複製をサポートするインストールを自動的に作成します(たとえば、複数ディスクセットアップ)。これを手動で行う必要がある場合、/h を Setup.exe に渡します。セットアップは自動的に一時的保存場所にクローンされてそこから実行されます。

テーブル 20 • その他のパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/hide_progress	基本の MSI、InstallScript、InstallScript MSI	初期設定中に表示される小さい標準の進行状況ダイアログが表示されないように抑制します。 小さな進行状況ダイアログは通常、初期設定中にスプラッシュ画面を表示するインストールで使用されます。これは、標準サイズの進行状況ダイアログでは大きすぎてスプラッシュ画面を表示するスペースがないためです。/hide_progress オプションを利用すると、これらのインストールの小さな進行状況ダイアログを非表示にするため、エンドユーザーには、進行状況を含まないスプラッシュ画面のみが表示されます。 ***メモ:*** *InstallScript インストールのみ:* /hide_progress *を指定して InstallScript インストールにスプラッシュ画面を含めると、スプラッシュ画面が表示される時間は* Setup.ini *で* **SmallProgress=Y** *または* **SmallProgress=N** *のどちらが指定されているかどうかによって異なります。* • **SmallProgress=Y** *と指定した場合、*/hide_progress *が指定されていない限り、スプラッシュ画面は進行状況ダイアログが表示されている間、表示されます。* • **SmallProgress=N** *と指定した場合、SplashTime キーで指定された時間にしたがってスプラッシュ画面が表示されます。このため、*/hide_progress *と* **SmallProgress=Y** *を同時に使用することは避けるようにしてください。* *InstallScript MSI インストールの場合: スプラッシュ画面を含める場合、インストールは自動的に小さな進行状況ダイアログに切り替えられ、スプラッシュ画面は進行状況ダイアログが表示されている間のみ表示されます。これは* /hide_progress *が指定されている場合にも適用されます。したがって、InstallScript MSI インストールで* /hide_progress *とスプラッシュ画面の同時使用を避けることが推奨されます。*
/hide_splash	InstallScript、InstallScript MSI	スプラッシュ画面が含まれている場合、その表示を抑制します。

テーブル 20・その他のパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/hide_usd	InstallScript	更新する製品のインスタンスをエンドユーザが選択できるようにするため、アップデートが有効になっているインストールによって表示されるダイアログを抑制します。アップデート対応インストールが複数の以前のインスタンスを検出した場合に、デフォルトでこのダイアログが表示されます。このコマンドライン オプションが使用されているときに、アップデート可能なインストールが複数の以前のインスタンスを検出すると、インストールは新しいインスタンスを作成します。
/ig{InstanceGUID}	InstallScript	システム変数 INSTANCE_GUID の値を指定します。たとえば、-ig{722C7440-B317-4B3B-AECA-0199EA4E7CDB}。このオプションを利用しない場合、インストールは自動的に値を INSTANCE_GUID に割り当てます (複数インスタンスインストールの場合、この値は新しく生成された GUID、標準インストールの場合、この値は PRODUCT_GUID の値と同じ)このオプションは、インストールランチャ、つまりインストールに使用するインスタンス GUID の判別など、セットアップ前のタスクを実行する前に実行されるカスタムアプリケーションを作成している場合に便利です。このオプションを使って、有効な GUID 以外の値を指定しないでください。
/installfromweb"PathToDisk1Files"	InstallScript	このオプションは、メディア ファイルへのパスが URL 以外のときも、インストールを強制的に起動された ClickOnce Install インストールのように動作させるという点を除いて、media_path オプションに類似しています。インストールを Web ページから手動で起動する場合、このオプションを使用します。このオプションは、ビルトイン Setup.ocx ファイルがインストールの起動に使用されるとき、自動的に追加されます。

テーブル 20 • その他のパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/media_path"PathToDisk1Files"	InstallScript	このオプションは、インストールが指定された場所で Disk1 ファイルを検索することを示します。Setup.exe ファイルのみが、元の起動場所に存在する必要があります。インストールは他のすべての必須ファイルを指定された場所から取得します (これには、メディアの場所に存在する必要がある Setup.exe も含まれます)。URL をメディア ファイルへのパスとして指定することができます。この場合、インストールは、起動された ClickOnce Install インストールのように動作します。起動された ClickOnce Install インストールは常にセキュリティ ダイアログを表示します。ClickOnce Install インストールの動作については、「InstallScript プロジェクトの One-Click Install インストール」を参照してください。
/L*<LanguageID>*	基本の MSI、InstallScript、InstallScript MSI	このオプションは、インストールが指定された言語で指定されたように実行されることを示します。言語 ID は 16 進数または 10 進数の数値で指定できます。16 進数を指定した場合、その値の先頭に 0x を付加します。たとえば、次のコマンドはインストールをドイツ語で実行することを示します: Setup.exe -L0x0407 Setup.exe -L1031 インストールでサポートされていない言語 ID、または無効な言語 ID を指定した場合、このパラメーターは無視されます。このパラメーターが指定されて、かつパラメーターが有効な場合、言語ダイアログは自動的に抑制されます (有効になっている場合) ので注意してください。
/tempextractpath	基本の MSI、InstallScript、InstallScript MSI、スイート/アドバンスト UI	このパラメーターを使って %TEMP% ディレクトリを希望する場所にカスタマイズすることができます (デフォルトでは、すべての実行可能ファイルを含む場所)。典型的なコマンドは、次のとおりです。 Setup.exe /tempextractpath"C:¥IS_temp¥" *メモ: たとえばウィルス対策や、アプリケーションを %TEMP% ディレクトリから実行するなどの環境制限を行う場合、このパラメーターの使用が推奨されます。*

テーブル 20 • その他のパラメーター (続き.)

パラメーター	プロジェクトの種類	説明
/w	基本の MSI、InstallScript MSI	基本の MSI プロジェクトで /w オプションを利用すると、インストールが完了するまで Setup.exe が終了しないよう 強制的に待機させることができます。 ***メモ•*** *バッチ ファイルで /w オプションを使用している場合、すべての Setup.exe コマンドライン オプションを* start /WAIT *で優先させることをお勧めします。正しい形式の使用例は次の通りです。* start /WAIT setup.exe /w
/SMS	InstallScript MSI	InstallScript MSI プロジェクトでは、Setup.exe はインストールが終了するまで自動的に待機するため、(以前のバージョンの InstallShield Professional で使用された) このオプションは必要ありません。

ユーザー定義のビルトイン コマンドライン パラメーター

プロジェクト• *ユーザー定義のコマンドライン パラメーターについてのこの情報は、InstallScript プロジェクトに適用します。*

InstallScript MSI プロジェクトでのユーザー定義のコマンドライン パラメーターの場合は、前述の –z コマンドライン パラメーターを使用します。

上にリストされているコマンドライン パラメーターの他、–bd、–f、および –zi が InstallScript プロジェクト用に予約されているコマンドライン パラメーターです。大文字、小文字に関わらず、ユーザーによるこれらのコマンドライン パラメーターの再定義は、エラーの原因になります。

実行時にシステム変数 CMDLINE にコピーされる、ユーザー独自のコマンド ライン引数を定義することができます。予め定義されたコマンド ライン パラメーター同様、カスタム引数を直接 Setup.exe に渡し、Setup.ini に配置するか、(InstallShield IDE の使用中、テスト目的で) [設定] ダイアログ ボックス (の [ビルド] メニューで [設定] をクリックすると表示されます) に配置することもできます。

ユーザー定義のコマンドライン パラメーターを組み込みのコマンドライン パラメーターと組み合わせて使用する場合は、ユーザー定義のパラメーターがコマンド全体の終わりに指定されていることを確認します。組み込みのコマンドライン パラメーターの前または間にユーザー定義のコマンドライン パラメーターを配置すると、誤った実行や予期しない動作が発生する可能性があります。次は、組み込みコマンドライン パラメーターを使用したユーザー定義コマンドライン パラメーターの正しい使用方法を示します

Setup.exe/f 1 "C:¥ Temp¥ Setup.iss" "User" Name; CompanyName; "

この例では、ユーザー定義のパラメーター‘User "Name; CompanyName ;’がコマンドの最後に表示され、/f 1 組み込みのコマンドライン パラメーターの後に続きます。

メモ: *Setup.exe は、256 MB 以上のメモリを持つシステム上でも適切に初期化され、セットアップが完了するまで常にメモリ内に存在します。DOS の性質上、コマンド ラインから Setup.exe を起動すると、Setup.exe はまだメモリにあっても、DOS プロンプトが素早く戻されます。*

アドバンスト UI およびスイート/アドバンストの Setup.exe コマンドラインのパラメーター

プロジェクト: *この情報は、次のプロジェクト タイプに適用します:*

- *アドバンスト UI*
- *スイート/アドバンスト UI*

エディション: *アドバンスト UI プロジェクト タイプは、InstallShield の Professional Edition で使用できます。スイート/アドバンスト UI プロジェクト タイプは、InstallShield Premier Edition で使用できます。これら 2 つのプロジェクト タイプの違いについては、「アドバンスト UI プロジェクトとスイート/アドバンスト UI プロジェクトの違い」を参照してください。*

アドバンスト UI およびスイート/アドバンスト UI インストールの Setup.exe セットアップランチャーには、多くのコマンドライン パラメーターを使用できます。未定義のパラメーターまたは無効なプロパティの定義を渡すと、コマンドライン エラーの原因となります。

テーブル 21 • アドバンスト UI およびスイート/アドバンスト UI セットアップ ランチャーのパラメーター

パラメーター	説明
/d **/d"PathTo.dbgFile"**	InstallScript デバッガーと /d パラメーターを使って、スイート/アドバンスト UI インストールにおける InstallScript アクションをデバッグできます。 ***メモ:*** *InstallScript コードをデバッグするには、デバッグ情報ファイルの* Setup.dbg *が使用可能である必要があります。また、開発マシン以外のシステムでインストールをデバッグする場合、特定のファイルを開発マシンからデバッグ マシンにコピーする必要があります。詳細については、「任意のコンピューターでのインストールのデバッグ」を参照してください。* InstallScript デバッガーで次のコマンドを使って、InstallScript カスタム アクションを実行します: Setup.exe /d インストールが元のコンパイル場所であるパスにある場合、デバッグ シンボル ファイル (.dbg) へのパスも指定する必要があります: Setup.exe /d"Path-to-Setup.dbg" スイート/アドバンスト UI プロジェクトにおける InstallScript アクションについての詳細は、「スイート/アドバンスト UI インストールに含まれる InstallScript コードを実行するアクションでの作業について」を参照してください。 デバッグに関する問題については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。
/debuglog **/debuglog"PathToLog"**	/debuglog パラメーターを使って、Setup.exe のログ ファイルを生成できます。 長い名前を持つ InstallShield.log を Setup.exe ファイルと同じディレクトリに生成するには、コマンドライン パラメーターのみを渡します。Setup.exe ファイルが読み取り専用の場所にあるとき、この処理は実行できません。例: Setup.exe /debuglog ログ ファイルの名前と場所を指定するには、パスと名前を次の要領で渡します。 Setup.exe /debuglog"C:¥PathToLog¥setupexe.log" 詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールのトラブルシューティング」を参照してください。

テーブル 21 • アドバンスト UI およびスイート/アドバンスト UI セットアップ ランチャーのパラメーター (続き.)

パラメーター	説明
/language:LCID	/language パラメーターは、インストールが指定された言語で指定されたように実行されることを示します。10 進数で言語 ID を指定します。たとえば、インストールをドイツ語で実行することを示すには、次の言語 ID を渡します: Setup.exe /language:1031 インストールでサポートされていない言語 ID、または無効な言語 ID を指定した場合、このパラメーターは無視されます。このパラメーターが指定されて、かつパラメーターが有効な場合、言語ウィザード ページは自動的に抑制されます (有効になっている場合) ので注意してください。
/log **/log"PathToPackageLog"** **/log:"PathToPackageLog"**	/log パラメーターを使用すると、ログが有効されたアドバンスト UI またはスイート/アドバンスト UI インストール内の各パッケージに対して、ログ ファイルが作成されます。 特定のフォルダー内でパッケージ ログ ファイルを生成するには、フォルダーのパスを /log パラメーターと一緒に、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe ファイルに渡します。オプションで、/log パラメーターとパスをコロンで区切ることもできます。たとえば、次のどちらのコマンドラインを使用しても、ログ ファイルは PathToLogFiles フォルダーに作成されます: Setup.exe /log"C:¥PathToLogFiles" Setup.exe /log:"C:¥PathToLogFiles" ログ ファイルの場所へのパスは、事前に存在している必要があります。 パッケージのログ ファイルを %TEMP% ディレクトリに生成する場合、パスを空白のままにしておきます。例: Setup.exe /log ログは、パッケージに対して有効にする必要がありますが、パッケージのログ関連の設定 (設定は、パッケージが .msi パッケージか、.msp パッケージか、またはその他のパッケージかによって異なります) は [パッケージ] ビューで構成します。詳細については、「アドバンスト UI またはスイート/アドバンスト UI インストールをコマンドラインから起動したときに作成できるパッケージ ログ ファイルのサポート」を参照してください。
/password:*MyPassword*	[リリース] ビューの Setup.exe タブで、リリースのパスワード関連設定を構成した場合、エンド ユーザーはサイレント インストールの実行時に /password パラメーターを使ったパスワードを指定しなくてはなりません。典型的なコマンドは、次のとおりです。 Setup.exe /silent /password:*パスワード*

テーブル 21・アドバンスト UI およびスイート/アドバンスト UI セットアップ ランチャーのパラメーター (続き.)

パラメーター	説明
PropertyName=Value **/PropertyName:Value** **PropertyName=**	プロパティをアドバンスト UI またはスイート/アドバンスト UI インストールに渡すとき、その方法はいくつか存在します。例: Setup.exe MyPropertyName=MyPropertyValue Setup.exe /MyPropertyName:MyPropertyValue プロパティの値を、アドバンスト UI またはスイート/アドバンスト UI Setup.xml ファイルによって設定されたプロパティの値も含めて、クリアする場合は、以下の構文を使用します: Setup.exe MyPropertyName= 以下のプロパティをコマンドラインで渡した場合、アドバンスト UI またはスイート/アドバンスト UI インストールでは特別な意味を持ちます: • ISFeatureInstall - このプロパティは、現在のインストール処理でインストールされるコンマ区切りの機能名リストを提供します。サイレントの初回インストールで使用します。 • ISFeatureRemove - このプロパティは、現在のインストール処理で削除されるコンマ区切りの機能名リストを提供します。メンテナンス インストールで使用します。 詳細については、「アドバンスト UI およびスイート/アドバンスト UI のプロパティ リファレンス」を参照してください。
/remove	/remove パラメーターは、アドバンスト UI またはスイート/アドバンスト UI インストールがインストールした、現在ターゲット システムにインストールされているパッケージをアンインストールします。このオプションは、アドバンスト UI またはスイート/アドバンスト UI の Setup.xml ファイルのメンテナンス条件が False の場合、[プログラムの追加と削除] 内のアドバンスト UI またはスイート/アドバンスト UI インストールのエントリも削除します。
/repair	/repair パラメーターは、アドバンスト UI またはスイート/アドバンスト UI インストールがインストールした、現在ターゲット システムに存在するパッケージを修復します。
/silent	/silent パラメーターは、ユーザー インターフェイス無しでインストールを実行します。
/skipupdate	/skipupdate パラメーターを使ってスイート/アドバンスト UI サイレント インストールを構成し、自動的にユーザーによるダウンロード可能なアップデートの適用を強制する代わりに、インストーラーがアップデートのダウンロードをスキップしてサイレント インストールを続行するように指示します。 このコマンドライン パラメーターは、サイレント モードによるスイート/アドバンスト UI インストール機能を強化します。

テーブル 21・アドバンスト UI およびスイート/アドバンスト UI セットアップ ランチャーのパラメーター (続き.)

パラメーター	説明
/stage_only	/stage_only パラメーターは、アドバンスト UI またはスイート/アドバンスト UI インストールのステージング部分を実行します。ステージング中にアドバンスト UI またはスイート/アドバンスト UI インストールは、スイートのパッケージを BrowseStageFolder ウィザード ページでユーザーが指定したディレクトリにコピーします。 パッケージのいずれかが、アドバンスト UI またはスイート/アドバンスト UI の Setup.exe に圧縮されている場合、/stage_only オプションは Setup.exe ファイルのパッケージをステージング ディレクトリにコピーする前に、それらを抽出します。 また、任意のパッケージのが Web 上にある場合、/stage_only オプションは、パッケージをダウンロードして、それらをステージング ディレクトリに追加します。 このオプションによって、アドバンスト UI またはスイート/アドバンスト UI インストール内のパッケージは実行されませんので注意してください。さらに、いずれのパッケージも圧縮解除しません。そのため、アドバンスト UI またはスイート/アドバンスト UI インストールのパッチを作成するときに、たとえばアドバンスト UI またはスイート/アドバンスト UI インストールに含まれるパッケージに非圧縮 .msi パッケージを生成する必要がある場合は、[ステージのみ] オプションを使用してください。その後、.msi パッケージの管理インストールを行います。 ***ヒント:*** *ISRootStagePath プロパティは、コピー済みのファイルを含むフォルダーへのパスを格納します。このプロパティのデフォルト値を指定するには、次の例のように、/stage_only パラメーターを渡すときにこのプロパティをコマンドラインから設定します。* Setup.exe /stage_only ISRootStagePath="C:¥MyStagingArea"
/SuiteGUID	/SuiteGUID パラメーターは、Setup.exe からスイート GUID を取得し、GUID が Setup.exe の場所にある SuiteGUID.log ファイルに保存されます。
/Update	//Update パラメーターを使って、スイート/アドバンUI UI インストールのアップデートをサイレント モードでどのように実行するかを指定できます。 進行状況バーを表示せず、UI インタラクションなしでインストールをアップデートするには、次を使用します: setup.exe /silent /update

アドバンスト UI またはスイート/アドバンスト UI インストールで、パッケージの 1 つを起動するときに使用するコマンドコマンドラインの指定方法については、「コマンドライン パラメーターをアドバンスト UI またはスイート/アドバンスト UI インストールのパッケージに渡す」をご覧ください。

Setup.exe (InstallScript プロジェクト)

プロジェクト: *この情報は、InstallScript プロジェクトに適用します。*

Setup.exe は、メインのインストール実行可能ファイルで、インストール初期設定を行いインストール エンジンを起動して、ターゲット システム上でインストールを実行します。

リリースを作成およびビルドすると、Setup.exe およびその他のセットアップ エンジン再配布可能ファイルは Disk1 フォルダーに配置されます。これらのファイルを、配布メディアの Disk1 で出荷する必要があります。

メモ: *Setup.exe の名前を任意の有効なファイル名に変えることができます (例、Install.exe)。DoInstall 関数を使用して他のインストールによって起動されるインストールの場合、起動されたインストールの Setup.ini ファイルの [Startup] セクションにある LauncherName キーには新しいファイル名が必要です。*

コマンドライン オプションを、直接 Setup.exe に渡すことも、Setup.ini に配置することもできます。テストが目的の際、InstallShield を使用時に [ビルド] メニューにある [設定] をクリックしたときに表示される、[設定] ダイアログ ボックスの [実行/デバッグ] タブを使用してコマンドライン オプションを Setup.exe に渡すことができます。

メモ: *自己展開型実行可能ファイルでは、Setup.exe で使用できるすべてのコマンドライン パラメーターをすべて使用できます。*

サポートされているパラメーターの一覧は、「Setup.exe および Update.exe コマンドライン パラメーター」をご覧ください。

14

よく寄せられる質問 (FAQ)

このセクションには、トピックごとに一般的な質問、およびその回答へのリンクが掲載されています。プロジェクトの種類によって、特定のタスクを処理するために必要なテクニックまたはその手順が異なる場合は、各プロジェクト別に回答が掲載されています。

プロジェクトの種類

- 適切なプロジェプト タイプは?
- プロジェクトを別のプロジェクト タイプに変換する方法は?
  - 変換後、スクリプトのコンパイル中に “エラー C8025 ‘INSTALLDIR’ : 未定義識別子” が発生する原因について。
  - 「文字列 PRODUCT_NAME が文字列テーブルで見つかりません」エラーが発生する理由は?

ファイルとフォルダー

- INSTALLDIR と TARGETDIR の違いは ?
- 既存のファイルを上書きするかどうかをWindows Installer が判別する方法は ?
- COM サーバーの登録方法は ?
- Windows サービスをインストール、開始、停止、削除、アンインストール、または構成する方法は?
- 空のフォルダーの作成方法は ?
- ダイナミック ファイル リンクのキーファイルを設定する方法は ?
- 同じコンポーネントに複数の実行可能ファイルを入れると、どうなりますか?

機能およびコンポーネント

- 条件付きで機能を選択する方法は?
- 条件付きで機能を非表示にする方法は?

- コンポーネントのインストール先設定時に、大文字のディレクトリ識別子を使用しなければならないのはなぜですか?
- コンポーネントの [リンク] 列で値を変更する方法は?

再配布可能ファイル

- Windows Installer をプロジェクトに追加する方法は?
- .NET Framework をプロジェクトに追加する方法は?

ショートカット

- インターネットショートカットを作成する方法は?
- フォルダーのショートカットを作成する方法は?

レジストリ

- レジストリからデータを読み込む方法は?
- レジストリにプロパティの値を書き込む方法は?

InstallScript

- InstallScript 言語で使用できるビルトイン関数の種類を調べる方法は?
- コンパイルされたスクリプトファイルで使用できるステートメントの最大数は? InstallScript におけるその他の制限は?

INI ファイル

- .ini ファイルからデータを読み込む方法は?
- .ini ファイルからすべてのキー名を取得する方法は?

プロパティ

- InstallScript からの Windows Installer プロパティの取得や設定方法は?
- 作成した遅延カスタム アクションの Windows Installer プロパティが空白値を表示する理由は?

条件

- 管理者権限を検出する方法は?
- 初回インストールかどうかを見分ける方法は?
- インストールシーケンスの最中のみ関数を呼び出す方法は?
- ユーザーが特定の機能を選択したかを確認する方法は?
- ユーザーがどのオペレーティング システムを使用しているかを確認する方法は?

エンドユーザー インターフェイス

- 実行時にリストボックスに入力する方法は?

- ファイル参照ダイアログを表示する方法は?
- ネットワーク参照ダイアログを表示する方法は?

カスタム アクション

- エンド ユーザーのシステム上のファイルの検索方法は?
- インストールの終了後にアプリケーションを起動する方法は?
- ファイルを .msi データベースに配置して実行時にそれを抽出する方法は?
- カスタム アクションからインストールを終了する方法は?
- VBScript コードを実行するカスタム アクションを作成後に INSTALLDIR 変数を使用する方法は?
- スクリプトから ODBC プロパティ (DBQ や SystemDB など) を変更する方法は?

SQL スクリプト

- デフォルトの SQL ランタイム動作をオーバーライドする方法は?
- InstallScript プロジェクトで、ある条件下で SQL スクリプトの実行を制御する方法は?
- InstallScript プロジェクトでサーバーサイド インストールを強制する方法は?
- Windows Installer ベースのプロジェクトで SQL サーバー サイド インストールを強制する方法は?
- SQL ランタイム エラーの処理方法は?
- SQL スクリプトが完全 SQL Server に対して実行されるように設定する方法は?

リリースのビルド

- コマンドラインからプリリースをビルドする方法は?
- プロジェクトの設定をコマンドラインから変更する方法は?

配布

- ユーザーがサイレントモードでインストールを起動する方法は?

アンインストール

- インストールしたアプリケーションが作成したファイルを削除する方法は?
- インストールしたアプリケーションが作成したレジストリデータを削除する方法は?

その他の質問に対する答えの多くは、ナレッジベースに用意されています。

用語集

テーブル 1 ・ InstallShield Express 用語集

用語	定義
.bin ファイル	Unix プラットフォームで実行可能なバイナリ ファイル。アプリケーションファイルは通常 .bin 拡張子を持ちます。
.bmp ファイル	ビットの配列として表示されるイメージのビットマップファイル。ビットマップ グラフィックでは、イメージは集合体としてパターンを形成するピクセルと呼ばれる小さな四角の集まりでスクリーン上に表示されます。イメージ内の各ピクセルは、1 つまたは複数のビットに対応します。
.cab ファイル	「キャビネット ファイル」を参照してください。
.command ファイル	Mac OS X プラットフォーム上で実行可能なコマンドファイル。アプリケーション ファイルは通常 .command 拡張子を持ちます。
.cub ファイル	ファイル内部整合性評価プログラム (ICE) のカスタム アクションを格納し、アクセスを提供する検証モジュール。「カスタム アクション」および「内部整合性評価プログラム」も参照してください。
.exe ファイル	Windows プラットフォーム上で実行可能なバイナリファイル。アプリケーションファイルは通常 .exe 拡張子を持ちます。

テーブル 1・InstallShield Express 用語集

用語	定義
.gif ファイル	Web 上でサポートされているグラフィック交換形式ファイル。このファイルの種類は、ビットマップ イメージを表示するのに利用されるビットマップ グラフィック ファイル形式です。これらのファイルはデータが損なわれないロスレス圧縮メソッドを使います。このファイルタイプは 256 色までサポートします。そのほか透明色もサポートし、背景に透明を設定することで下にあるレイヤーの色が見えるようにすることが可能です。このファイル形式は、線画のような限られた色のみを使ったイメージの場合 .jpg 形式よりも適しています。
.idt ファイル	エクスポートされた Windows Installer データベース テーブル。
.ism ファイル	InstallShield がプロジェクト情報を保存するために利用する作業ファイル。リリースをビルドする際、InstallShield は .ism ファイルを使用して配布用 .msi ファイルを作成します。
.jar ファイル	インストールに必要なクラス、イメージ、およびサウンド ファイルを含み、単一ファイルに zip されている Java アーカイブ ファイルです。
.jpg ファイル	JPEG として知られる。標準化した委員会の元の名称をとった Joint Photographic Experts Group の略語。このファイルの種類は 256 色までサポートします。そのほか透明色もサポートし、背景に透明を設定することで下にあるレイヤーの色が見えるようにすることが可能です。ファイルは非可逆圧縮を利用します。この種類の圧縮はカラーおよびグレースケールの濃淡イメージを圧縮し、余分な情報または不必要な情報を省略するためにデータが失われます。これらのイメージは 1600 万色をサポートし、写真や複雑なグラフィックに適しています。
.mif ファイル	ハードウェアまたはソフトウェア コンポーネントについて記述するために使われる管理情報フォーマット ファイル。
.msi ファイル	完成した状態の Windows Installer インストール パッケージ。インストール リソース ファイルを含み、アプリケーションのデータファイルすべてを圧縮して格納することができます。.msi ファイルはエンドユーザーに配布されるファイルで、Windows Installer サービスとインタラクトしてアプリケーションをマシンにインストールします。
.msm ファイル	マージ モジュールのデータベースファイル。このファイル タイプはモジュールのインストール プロパティおよびインストール ロジックすべてを含みます。「マージ モジュール」も参照してください。
.msp ファイル	Windows Installer パッチ ファイル。このファイルの種類は概要情報ストリーム、トランスフォーム サブストレージ、およびキャビネット ファイルで構成されます。.msp ファイルは、ターゲットとするインストールパッケージのデータベースにパッチ情報を追加するデータベーストランスフォームが 1 つ以上含まれます。インストーラーはこのパッチ情報を利用してキャビネット ファイルに格納されているパッチファイルを適用します。

テーブル 1・InstallShield Express 用語集

用語	定義
.mst ファイル	トランスフォーム パッケージ。このファイルの種類は、2 つの .msi データベース間の差分を含む簡素化された Windows Installer データベースです。トランスフォームを使うと、管理者がインストールパッケージを配布する際、変更した設定をデータベースに適用することができます。「トランスフォーム」も参照してください。
.NET	情報、人間、システム、そしてデバイスをつなげるために Microsoft が作成したオペレーティング システムプラットフォーム。.NET 環境を利用すると、開発者は共通言語ランタイムを通して好みの言語を使ってアプリケーションおよび Web サービスをビルド、作成、そして配布することができます。
.NET Compact Framework	モバイル デバイスなど、リソースに制限のあるデバイス上で実行できるようデザインされた .NET Framework の縮小版。.NET Compact Framework は現在、Pocket PC と Windows CE .NET デバイスでのみ使用できます。他のデバイスのサポートは今後のリリースで実現していく予定です。
.NET Framework	アプリケーションおよび Web サービスのような .NET 技術を利用したサービスをビルド、配布、および実行するために Microsoft が作成したプログラムの基盤。.NET Framework は 3 つの主要な部分で構成されます。一般言語ランタイム、統一されたクラス ライブラリが階層に分かれたセット、および ASP.NET と呼ばれる Active Server Pages のコンポーネント化されたバージョン。
.ocx ファイル	OLE (Object Linking and Embedding) カスタム コントロール ファイル。このファイルタイプは、ウィンドウサイズの変更などを可能にする機能を実行するためにアプリケーションによって呼び出される、クロスプラットフォームの COM ファイルです。
.p12 ファイル	PKCS (Public-Key Cryptography Standard) 番号 12 に準拠する証明書ファイル。これはユーザーのプライベートキー、証明、機密情報、またはその他の情報を安全に保管したり移動したりするためのポータブルフォーマットを指定します。
.pcp ファイル	Windows Installer パッチ作成プロパティ ファイル。
.prq ファイル	メインのアプリケーションがインストールされる前にターゲットマシン上に既にインストール済みでなくてはならない、ベースアプリケーションまたはコンポーネントについての情報を含む前提条件ファイル。インストール プロジェクトにセットアップ前提条件を含めると、開発者は複数のインストーラー ファイルを単一の実行可能ファイルに連鎖させることができます。
.reg ファイル	「レジストリ ファイル」を参照してください。
.scm ファイル	ScreenCam ムービー ファイル。
.wmf ファイル	Windows メタファイル形式のファイル。この種類のグラフィック ファイルは Microsoft Windows ベースのアプリケーション間でグラフィック情報を交換するのに使用されます。

テーブル 1 • InstallShield Express 用語集

用語	定義
ActiveX	オブジェクト指向プログラミング技術およびそのツールのセット。ActiveX を利用すると、開発者は他のアプリケーションやオペレーティング システムが呼び出すようにアプリケーションを作成することができます。ActiveX 技術はスタティックページではなく、アプリケーションのような外観を持ち作動するインタラクティブ Web ページの作成を可能にします。
AlwaysInstallElevated	HKEY_LOCAL_MACHINE¥Software¥Polices¥Microsoft¥Windows¥Installer および HKEY_CURRENT_USER¥Software¥Polices¥Microsoft¥Windows¥Installer レジストリ キーの下で Windows プラットフォーム用に構成可能なユーザー ポリシー。Microsoft¥Windows¥Installer and HKEY_CURRENT_USER¥Software¥Polices¥Microsoft¥Windows¥Installer.ハイレベルなシステム権限でパッケージをインストールするには、これら両方のレジストリキーの下にある AlwaysInstallElevated の値を 1 に設定します。
American Standard Code for Information Interchange (ASCII)	各キャラクタに 0 から 127 が割り当てられている、文字、数字、句読点および他のキャラクタを表すコード。このコードによって、異なる種類のコンピュータとコンピュータ アプリケーション間でデータを交換できます。
API	「アプリケーション プログラミング インターフェイス」を参照してください。
ASCII	「American Standard Code for Information Interchange (ASCII)」を参照してください。
ATL	「アクティブ テンプレート ライブラリ」を参照してください。
Binary テーブル	ビットマップやアイコンなどのバイナリ データを収納するテーブル。カスタム アクションのデータも含みます。
CBT	「コンピューター ベース トレーニング」を参照してください。
CD	「コンパクト ディスク」を参照してください。
CD ブラウザー	CD がドライブに挿入された時に起動されるグラフィックユーザー インターフェイス。CD に含まれる 1 つまたは複数のアプリケーションを起動するのに利用されます。
CD-ROM	「コンパクト ディスク 読み取り専用メモリ」を参照してください。
COM	「コンポーネント オブジェクト モデル」を参照してください。
COM サーバー	コンポーネントオブジェクトモデル仕様に従って他のアプリケーションにオブジェクトを公開する実行可能ファイル。COM サーバーにより公開されるオブジェクトの種類には、ActiveX コントロール、ActiveX ドキュメント、Automation オブジェクト、MTS コンポーネントがあります。

テーブル 1・InstallShield Express 用語集

用語	定義
COM ファイル	製品間で共有できる再利用可能なコードが含まれるバイナリ ソフトウェア コンポーネントです。COM ファイルの代表的な例は、Word 文書に埋めこまれた Microsoft Excel スプレッドシートです。この場合 Excel が COM サーバーとして作動し、Word はそのクライアントとして作動します。
COM+ ファイル	拡張 COM ファイル。インストールに関して、COM+ は配布されたコンポーネントのビルド用にさらなる拡張サポートを追加します。たとえば、appID キーおよび値をレジストリに作成し、これによって様々な COM コンポーネントをリモートで実行または権限つきで実行するよう構成することが可能です。
DBMS	「データベース管理システム」を参照してください。
DCOM	「分散コンポーネント オブジェクト モデル」を参照してください。
DCOM ファイル	ネットワーク全体で通信できるソフトウェア コンポーネント。DCOM (以前はネットワーク OLE と呼ばれる) によって、1 つのアプリケーションのコンポーネントを複数のネットワーク コンピューターに渡って配布できるようになります。
DLL	「ダイナミック リンク ライブラリ」を参照してください。
EXE メディア タイプ	単一の自己展開型ファイルに含まれるすべてのインストールまたはアンインストール ファイル。
GUI	Graphical user interface (グラフィカル ユーザー インターフェイス)。
GUID	「Globally Unique Identifier (GUID)」を参照してください。
HTML	「hypertext markup language (ハイパーテキスト マークアップ言語)」を参照してください。
IAT	「インポート アドレス テーブル」を参照してください。
ICE	「Internal Consistency Evaluator (内部整合性評価プログラム)」を参照してください。
IDE	「インタラクティブ開発環境」 または 「統合開発環境」を参照してください。
IDriver.exe	InstallShield スクリプト ランタイム エンジン。IDriver.exe は、InstallShield で作成したインストールの一部を実行するためにコンピューター上で必要です。IDriver.exe は、デフォルトで、C:¥Program Files¥Common Files¥InstallShield¥Driver¥7¥Intel 32 と C:¥Program Files¥Common Files¥InstallShield¥Driver¥8¥Intel 32 の両方またはいずれかの共通ファイルの場所にあります。一部のインストール エラー メッセージは、IDriver.exe を参照する場合があります。
IE	「インターネット エクスプローラー」を参照してください。
ikernel.exe	InstallShield エンジン。「エンジン」も参照してください。

テーブル 1 • InstallShield Express 用語集

用語	定義
INSTALLDIR	インストールのルート インストール先ディレクトリを指定するシステム変数。
INSTALLLEVEL	デフォルトで機能のインストールは [ON] に選択されている初期レベル。
InstallShield スクリプト ランタイム	「IDriver.exe」を参照してください。
ISM	「.ism ファイル」を参照してください。
ISScript.msi	InstallScript エンジンインストーラー。ISScript.msi はインストールを実行するのに必要なファイルをインストールします。InstallScript エンジンは、InstallShield スクリプト ランタイムとしても知られています。InstallScript はインストールを作成するのに利用されるプログラム言語です。
JAR	Java アーカイブ ファイルの頭文字。「.jar ファイル」も参照してください。
Java アーカイブ	「.jar ファイル」を参照してください。
Java 仮想マシン (JVM)	バイトコードにコンパイルされた Java プログラムを解釈する擬似コンピューター。Java プログラムは通常 .class ファイルに保存されています。
Java ネイティブ インターフェイス (JNI)	共有ライブラリのネイティブコードを呼び出すための Java コードを有効にする Java の仕様。JNI ライブラリのネイティブコードは、JNI を通して Java コードを呼び出すこともできます。
JNI	「Java ネイティブ インターフェイス」を参照してください。
JPEG	「.jpg ファイル」を参照してください。
JVM	「Java 仮想マシン」を参照してください。
KB	「ナレッジ ベース」を参照してください。
LogDB	「ログ データベース」を参照してください。
MD	「メディア ディスクリプタ」を参照してください。
MFC	「Microsoft Foundation Classes」を参照してください。
Microsoft Developer Network (MSDN)	Microsoft の製品およびテクノロジを使用してアプリケーションを作成している開発者を支援するためにデザインされたオンラインおよびオフライン サービス。
Microsoft Foundation Classes (MFC)	開発者が Win32 アプリケーションを書く際に使用することができる大規模な C++ クラスライブラリ。MFC を使用するアプリケーションを実行するには、MFC エンジンがターゲットマシンにインストールされている必要があります。

テーブル 1・InstallShield Express 用語集

用語	定義
Microsoft ディスク オペレーティング システム (MS-DOS)	Microsoft が開発した一番最初のオペレーティング システム。MS-DOS は、 Windows 95、98 および ME の基礎となるオペレーティング システムです。Windows NT、2000 および XP システムは存在する DOS アプリケーションをサポートしています。
Microsoft ビルド エンジン (MSBuild)	SQL Server と完全互換性があるデスクトップおよび共有ソリューションのビルドのためのデータ エンジン。SQL Server 7.0 への簡単な移行パスを提供します。MSDE で構築されたソリューションは、コードを 1 行も変えずに SQL サーバ 7.0 に完全に移行することができます。
MIDI	「musical instrument digital interface」を参照してください。
MIME	「multipurpose internet mail extensions」を参照してください。
MM	「マージ モジュール」を参照してください。
MS-DOS	「Microsoft ディスク オペレーティング システム」を参照してください。
MS-DOS プロンプト	MS-DOS(Microsoft Disk Operating System)プロンプトは、MS-DOS が新しいコマンドを受け付けられる状態にあることを示す [コマンドプロンプト] ウィンドウ内のビジュアル インジケーターです。デフォルトの MS-DOS プロンプトは、通常、C:> プラス点滅カーソルです。
MSDE	「Microsoft Data Engine」を参照してください。
MSDE オブジェクト テンプレート	MSDE マージ モジュールのベース。MSDE マージ モジュールを Windows Installer プロジェクトまたは InstallScript プロジェクトに追加すると、InstallShield アプリケーションは、このマージ モジュールのベースファイル(または、スタブ)と共に起動します。アプリケーションが、ベースファイルに設定を適用できるように、このマージ モジュールを構成する必要があります。
MSDE 名前付きインスタンス	MSDE (Microsoft Data Engine) インストールの名前付きインスタンス。「名前付きインスタンス」も参照してください。
MSDN	「Microsoft Developer Network」を参照してください。
MSI	「.msi ファイル」を参照してください。
MSM	「.msm ファイル」を参照してください。
MSP	「.msp ファイル」を参照してください。
MST	「.mst ファイル」を参照してください。
multipurpose Internet mail extensions (MIME)	Web クライアントに送るファイルを識別する Web サーバーが使用する規格。MIME 規格は、送るときのファイルの種類およびそれをオリジナルの形式に戻す方法を指定する方法です。

テーブル 1 • InstallShield Express 用語集

用語	定義
musical instrument digital interface (MIDI)	音楽をデジタルシンセサイザーで録音/再生するために設計されたプロトコル。実際のサウンド（音声または音楽）をデジタル録音する .wav ファイルとは異なり、MIDI ファイルは単純に楽器、旋律、および、それをどのように再生するかを定義するだけです。エンドユーザーのシステム上にあるシンセサイザ（一般に MIDI 可能なサウンドボードの一部）は MIDI 指示を読み込んで、音楽を再生します。MIDI ファイルには実際の音楽ではなく命令のみが含まれているので、サイズは同じ長さのデジタル オーディオ ファイルに比較して何倍も小さくなっています。また、MIDI ファイルは再生においても、より少ないプロセッサパワーが必要です。
NT サービス	サービスとして実行するために Windows NT にインストールされたアプリケーション。つまり、インストールされた後は自動的に実行され、ユーザーには表示されず、サービスを開始するためにユーザーがログインする必要もありません。NT はレジストリで定義、および記述されているサービスを管理します。
One Really Cool Application (ORCA)	Windows Installer ファイルを編集するための外部ツール。
One-Click Install	One-Click Install インストールを使用すると、エンドユーザーは最小限の労力でアプリケーションをダウンロードし、すぐに使用することができます。ユーザーによるインプットは最小限で、インストールが自動的にダウンロードされ、圧縮解除され、そして実行されます。エンドユーザーにダウンロード場所を問い合わせることもなく、インストールを手動で起動する必要もありません。
ORCA	「One Really Cool Application」を参照してください。
OS	「オペレーティング システム」を参照してください。
PKCS	「public-key cryptography standards」を参照してください。
public-key cryptography standards (PKCS)	RSA Security Inc.によって開発され、ここで定義されている仕様は、パブリックキー暗号化に関する既存の規格団体が扱っていない部分を標準化するために開発されました。
QuickPatch	アプリケーションのファイルおよびインストールを特定バージョンにアップデートするのに必要なデータベースの一部のみを含む、特殊な種類のパッチ パッケージ。QuickPatch プロジェクトを使用して、InstallShield で QuickPatch を作成することができます。
Readme ファイル	重要な情報を含むアプリケーションの配布に含まれているテキストファイル。この情報は通常、インストール、アンインストール、または機能に関する、他のドキュメントに含まれていない可能性がある情報です。
red green blue (RGB)	混ぜて他の色を作成することができる光の三原色。色付きのイメージは通常、RGB 三重項のシーケンス、または、別々の赤、緑、青のオーバーレイとして格納されますが、これらに限りません。これらの三色は、カラー ブラウン管（CRT）の三本の ガン と人間の眼の色レセプターに相当します。

テーブル 1・InstallShield Express 用語集

用語	定義
Regedt32.exe、Regedit.exe	Windows Registry Editor の 2 つのバージョン。レジストリ エディターを使用して、レジストリ内のエントリの編集をすることができます。Regedt32 には、より多くのレジストリ編集機能が用意されています。インストール中にエラーが発生した場合、インストールを完了するためにレジストリの編集が必要な場合があります。Windows XP コンピューターでは、Regedt32.exe と Regedit.exe が既に融合されているので、新しいオペレーティング システム上では、これらの 2 つのアプリケーションは同じ機能を実行します。
RGB	「red green blue」を参照してください。
RPC スタブ	「リモート プロシージャ コール スタブ」を参照してください。
RTF	「リッチ テキスト フォーマット」を参照してください。
SCM	「サービス コントロール マネージャー」を参照してください。
SDK	「Software Development Kit (ソフトウエア開発キット)」を参照してください。
Section 508 (米国リハビリテーション法 508 条)	リハビリテーション法を改訂して、連邦政府各機関に対して障害を持つ人々もその電子情報技術にアクセスできるように要求した米国の法律。508 条は、情報技術におけるバリアの除去、障害を持つ人々に新たな機会の提供、およびこれらの目的を達成することを支援する技術の開発を推進することを目的に制定されました。この法律は、電子情報技術を使用、開発、購入、メンテナンスをするすべての連邦政府各機関に適用します。508 条の下で連邦政府各機関は、障害を持つ被雇用者および一般の人々に対して他の人々と同様の情報へのアクセシビリティを提供することが義務付けられています。
Setup.inx	コンパイルされたスクリプト ファイル。インストーラー エンジンが実行するオブジェクト コード。
SP	「サービス パック」を参照してください。
SQL	「Structured Query Language (構造化クエリ言語)」を参照してください。
SQL ステートメント	キーワードで始まり実行されるアクションを完全に記述する SQL の完全フレーズ。例、**SELECT * FROM Orders**。
TARGETDIR	Windows Installer ベースのインストールでは、TARGETDIR プロパティはインストールのルート インストール先ディレクトリを指定します。InstallScript インストールでは、TARGETDIR システム変数はデフォルトでインストールのルート インストール先ディレクトリを指定します。
UI	「ユーザー インターフェイス」を参照してください。
uniform/universal resource locator (URL)	Web サイト/ページの場所を示します。

テーブル 1 • InstallShield Express 用語集

用語	定義
unInstallShield	エンド ユーザーの PC から製品をアンインストールするセットアップ ウィザードに含まれているコンポーネント。
Update Manager (Windows または Java)	エンドユーザーがアップデートのスケジュールを設定する際、またはエージェントによって実行されるアップデートのクエリをサイレンスにするために使用することができるオプションの Win32 または Java ベースのクライアント アプリケーション。
URL	「uniform/universal resource locator (URL)」を参照してください。
VP	「バリューパック」を参照してください。
Web installation (Web インストール)	エンドユーザーが Web サイトにアクセスして、そこから実行するインストール。
Web サイト	World Wide Web 上の場所
WinDir	Win16 on Win32 (WOW) 用の実行可能ファイルへのパスを格納します。
Windows API	Windows アプリケーション プログラム インターフェイス (API) は、Windows プラットフォーム用に書かれたアプリケーションが利用するビルドブロックを提供します。各 API はコンピューター オペレーティング システム、またはアプリケーション プログラムが指定する特定のメソッドです。アプリケーションを作成するプログラマーは、Windows API を利用してオペレーティング システム、または別のアプリケーションを要求することができます。各 API を適切に実行するための要件は異なります。
Windows Installer	(1) インストールおよび構成サービス。Microsoft Windows Installer は、データ ドリブン モデルのエンジンで、すべてのインストールデータおよび命令が 1 つの完結したパッケージになっています。データ ドリブン インストール モデルでは、インストール テーブルのプライマリセットは、すべてのアプリケーション リソース(ファイル、レジストリキー等)がそれがサポートするコンポーネントまたは機能に明確に関連しているところに作成されます。ユーザーがインストールするオブジェクト、インストールする場所を選択すると、その手続き命令は Windows Installer によって処理されます。 (2) .msi パッケージのサービス、プロパティ、およびテーブルを意味します。
Windows Installer サービス	アプリケーションインストールの構成、およびアプリケーションのアンインストールを一元的に管理するオペレーティング システム コンポーネント。
Windows NT サービス	「Windows サービス」を参照してください。
Windows インストール CD	Windows オペレーティング システムを含むコンパクトディスク。コンピューター製作会社によってコンピューターにオペレーティング システムがインストールされた場合、通常マニュアルと共にインストール CD が付属されています。自分で Windows オペレーティング システムをインストールした場合、そのときに使った CD がインストール CD です。

テーブル 1・InstallShield Express 用語集

用語	定義
Windows サービス	それ自身の Windows セッションで長期にわたって作動する実行可能アプリケーション。例、イベント ログにメッセージを書き込むアプリケーション。
Windows システム フォルダ	Windows System フォルダーは、コンピューターが適切に稼動し続けるために必要なコア オペレーティング システムを含みます。インストール中に発生するエラーの原因としては、System フォルダーに含まれているファイルが破損しているか、または存在していないことが考えられます。
WYSIWYG	What You See Is What You Get (見たとおりのものが結果に反映されること) の略 。
XCopyFile	1 つ、または複数のファイルおよび/またはサブディレクトリを、必要に応じてターゲット マシンにサブディレクトリを作成して、ターゲット ディレクトリにコピーする InstallScript 関数。
XML	「拡張マークアップ言語」を参照してください。
XSL	「拡張スタイルシート言語」を参照してください。
アクション	インストールまたはアンインストール ウィザードの実行中に特定の時点で処理を行なうコマンド。インストール パッケージの開発者はビルトイン標準アクションを利用、および独自のカスタム アクションを作成することができます。アクションはエンド ユーザーに対して処理状況を表示したり、処理をキャンセルできるようにすることが可能です。
アクセシビリティ	障害者を含めてすべての人々に対する有用性、または利用の可能性。
アクティブ ディレクトリ	管理者がネットワーク上に存在するオブジェクトを検索することができる Microsoft Windows 2000 がサポートする構造。アクティブ ディレクトリは、分散コンピューター環境用 Windows 2000 Server で使用されるディレクトリです。
アクティブ テンプレート ライブラリ (ATL)	たとえば MFC などのバイナリよりも小さい COM コントロールを書くための一連のクラス。
圧縮	1 つまたは複数のファイルを基に、オリジナルファイルの合計サイズよりも小さいサイズの新しいファイルを作成します。この新しいファイルからオリジナルファイルを複製することが可能です。
圧縮解除	「伸張」を参照してください。
アップグレード	アプリケーションの古いバージョンからより最新のバージョン(通常、アプリケーションの改善版)への移動。
アップグレードコード	既存ソフトウェアの新しいバージョンをインストールするために必要なコード。
アップグレード	既存ソフトウェアの新しいバージョンをインストールするための処理。

テーブル 1 • InstallShield Express 用語集

用語	定義
アドバタイズ	アプリケーションのインストール処理中に機能がインストールされない「ジャスト イン タイム」インストール。その代わり、要求された時に直ちにインストールされます。機能のアドバタイズを有効にすると、アドバタイズを阻止するその他の要素がない限り、インストールが実行されているモードにかかわらず機能はアドバタイズされます。/jm オプションを使ってコマンドラインから MsiExec.exe を起動すると、機能はエンドユーザーのマシンにアドバタイズされます。/ju 関数を利用すると、機能は現在のエンドユーザーにアドバタイズされます。Custom Setup ダイアログで、エンドユーザーはすぐにインストールする機能とあとで使用する機能を制御できます。アドバタイズには assigning (割り当て) および publish (パブリッシュ) の 2 つの種類があります。「割り当て」、および「パブリッシュ」も参照してください。
[アドバタイズ] シーケンス	[アドバタイズ] シーケンスには、ユーザーが /jコマンドライン オプションを使ってインストールを起動したときに実行されるアクションのリストが含まれます。検証規則 ICE78 では、[アドバタイズ]-[ユーザー インターフェイス] シーケンスが空でなくてはなりません。
アドバタイズ ショートカット	エンドユーザーがショートカットを初めて起動するまでインストールされない、製品または機能へのショートカット。インストールの実行時、[エンドユーザーがショートカットを持つ製品または機能に対して 要求があった場合に、この機能をインストールします] オプションを選択した場合、ショートカットが作成されますが、コンポーネントのファイルはユーザーがショートカットを起動するまでインストールされません。ショートカットを初めて起動すると、Windows Installer サービスはコンポーネントのファイルと他のデータをインストールし、その後、ショートカットによってターゲット ファイルが起動されます。その後、ショートカットを使用すると常に、通常のショートカットと同じように動作します。
アドバタイズ	アプリケーションのインストール処理中に直ちにインストールされない機能をエンドユーザーに向けて表示する処理。その代わり、これらの機能は要求された時に直ちにインストールされます。「アドバタイズ」も参照してください。
アプリケーション パス	システムのパスにアプリケーションとその .dll ファイルの場所が指定されていない場合に、Windows がそれらの検索に使用するレジストリキー。app パスエントリは、コンポーネントの詳細設定で設定することができます。
アプリケーション プログラミング インターフェイス (API)	アプリケーションがコンピューターのオペレーティング システムとコミュニケーションをとるために利用する、またオペレーティング システムがアプリケーションに対してサービスを有効にするために利用する一連のルーチン。
アンインストール	インストールを取り消す処理です。アンインストールは、エンドユーザーが製品ファイルを削除し、インストール中にマシンに対して加えられた変更を元に戻すインストールのメンテナンス オプションです。
アンインストール ログ ファイル	インストール中に発生するすべてのアンインストール関連のイベントの記録です。ログファイルはインストールの始めに初期化されます。このログファイルが破損した場合、アンインストール中にエラーが発生する可能性があります。

テーブル 1・InstallShield Express 用語集

用語	定義
移行	1 つのオペレーティング環境から他の一般的により優れていると思われているオペレーティング環境に移るプロセス。たとえば、Windows NT Server から Windows 2000 Server へ変えることも、このプロセスによって新しい機能が生かされ、古い設定は今後変更が必要なくなり、現在のアプリケーションが継続して新しい環境で動作するようにいくつかの手順がとられるので、移行と考えることができます。また、Windows NT から UNIX ベースのオペレーティング システム (または、その逆)も移行と考えられます。また、1 つのデータベースを違う種類のデータベースに移行することも可能です。このためには、通常、データを古いデータベースからアウトプットして新しいデータベースにインプットできる共通のフォーマットにデータを変換する必要があります。また、データを 1 つの記憶装置から別の記憶装置に移すといった単純なプロセスも移行といいます。
依存関係	別のソフトウェア オブジェクトが必要とするソフトウェア オブジェクト。
一時ディレクトリ	オペレーティング システムまたはアプリケーションを使用中に、そのファイルを一時的に保管するハードドライブ上のフォルダー。アプリケーションが終了したとき、一時ファイルは削除されます。一時ディレクトリから手作業でファイルをクリーン、または削除する必要がある場合もあります。
インストーラー	コンピューター上でのインストールおよび構成サービス。
インストーラー データベース	アプリケーションのインストールに必要な情報すべてを含むデータベース。アプリケーションのグループをインストールするのに必要な情報を含む関係データベースを構成する、多くの相互に関係するテーブルで構成されます。
インストーラー プロパティ	インストール中に利用される変数。
インストーラー関数	アプリケーションがインストーラー サービスを取得するために呼び出すアプリケーション プログラム インターフェイス。
インストール	プログラムとその構成ファイル、機能、およびコンポーネントをソース メディアからターゲット システムへ転送すること。「インストール プロセス」も参照してください。
インストール オン デマンド	ファイルが存在しない機能でも、ユーザーやアプリケーションに対してその機能の提供を可能にする Windows Installer の機能性。
[インストール] シーケンス	実行時に、アプリケーション ファイルのインストール、アクションの実行、およびダイアログの表示が行われる順番(シーケンス)。[インストール] シーケンスは、たとえば インストールのランチャをダブルクリックするなどしてインストールを起動したときにデフォルトで実行されます。
インストール タイプ	エンドユーザーがインストールまたはアンインストールを選択することができる、予め定義された機能のグループ。

テーブル 1 • InstallShield Express 用語集

用語	定義
インストール パッケージ	アプリケーションのインストールまたはアンインストールのために合わせて使われる 1 つまたは複数のファイル。
インストール パッケージ オーサリング ツール	ユーザーがインストールパッケージを作成するのに利用できるサードパーティ ツール。
インストール プロジェクト	インストール作成中、インストールを構成するソースファイル、ダイアログ、アクションおよび条件の集合体。
インストール プロセス	パッケージまたはアプリケーションのコンテンツをターゲット コンピューターにインストールする全プロセス。インストール処理は、取得段階、実行段階、およびインストールに失敗した場合、ロールバックの 3 つの段階で構成されています。
インストール レベル	機能のインストールレベルは、インストールのデフォルトでそれが選択されているかどうかを部分的に判断します。その値は Windows Installer プロパティ の INSTALLLEVEL と比較され、どの機能のインストールが選択されたかを判断します。機能のインストール レベル プロパティが INSTALLLEVEL の値と同等またはそれ以下の場合、その機能がインストールされます。
インターネット エクスプローラー (IE)	Microsoft によって開発され、Windows オペレーティング システムと共に配布されるブラウザー。
インターフェイス	独立システムが相互にコミュニケーションをとるポイント。アプリケーションのユーザー インターフェイスは一般的にツールバー、メニュー、ボタン、ウィンドウ、およびその他のアイテムで構成されます。
インタラクティブ開発環境 (IDE)	ソフトウェア プログラムのインターフェイス。「統合開発環境」とも呼ばれます。
インポート アドレス テーブル (IAT)	.dll ファイルまたは実行可能ファイルをインポートするのに使用されるテーブル。
ウィザード	(1) ステップ バイ ステップ方式でユーザーが不慣れなタスクを完了することができる対話型ガイド プログラム ユーティリティ。 (2) InstallShield Wizard はインストールまたはアンインストールの実行時に表示されるウィザードです。インストールまたはアンインストールの手順をエンドユーザーにフレキシブルに案内する、予め定義されたひとまとまりのダイアログのことを指します。
ウィザード インターフェイス	実行時にエンドユーザーがウィザードとインタラクトするために使用されます。
ウィザード ダイアログ	インストール/アンインストールの最中にエンドユーザーに表示されるダイアログ。情報ウィンドウによって、エンドユーザーは情報を読み込んだり指定したりしながらオペレーションとインタラクトできます。

テーブル 1 • InstallShield Express 用語集

用語	定義
ウィンドウ	開かれたアプリケーションがスクリーン上に表示される領域。
エンジン	基本の関数を実行し、その他のプログラムの処理を全体的に調整するプログラム。エンジンは内部的に動作します。InstallShield は、インストールを駆動するために InstallScript エンジン (ISScript.msi) および Windows Installer エンジンを利用します。InstallShield はまた、ikernel と呼ばれる独自のエンジンも装備しています。
エンドユーザー	製品をインストールまたはアンインストールするユーザー。
エンドユーザー マシン	エンドが製品をインストールまたはアンインストールするマシン(コンピューター)。
オートノミック コンピューティング	人体の自律神経系にちなんで名づけられ、またそのパターンに従った自己管理コンピューティング モデル。オートノミック コンピューティング システムは、ユーザーがインプットすることなくコンピューター アプリケーションおよびシステムの機能をコントロールします。これは人体の自律神経系が身体を調節および保護するのと同じ原理です。
オブジェクト	(1) 個別の機能をインストールするために必要なロジックとファイルがすべて含まれたパッケージ。
オペレーティング システム (OS)	メモリの中のストレージ スペースを割り当てたり、入力または出力関数を制御したりすることで、コンピューターのオペレーションを制御し、プログラムの処理を管理するソフトウェア。
親/子インストール	「ネスト インストール」を参照してください。
外部ユーザー インターフェイス	インストールパッケージの作成者によって開発されたユーザー インターフェイス。インストーラーで利用可能な内部ユーザー インターフェイス機能は利用しません。
概要情報ストリーム	Windows Installer パッケージのために定義されアプリケーションのインストール時にインストーラーによって使用されるタイトル、作成者、パッケージ コード、テンプレート、概要およびスキーマ等のプロパティ。
拡張可能スタイルシート言語 (XSL)	XML 情報をどのように表示するかを記述するのに使われる言語。
拡張マークアップ言語 (XML)	(1) 実質的に Standard Generalized Markup Language (SGML) を簡素化したプログラム言語。開発者はこれを使用してカスタマイズされたタグを作成し、効率良くコンテンツを編成および配布することができます。 (2) MultiPlatform プロジェクトの出力フォーマットです。 (3) InstallShield で作成された Windows ベースの プロジェクトは、ソースコード管理ソフトウェアで保存できるように XML に変換することも可能です。

テーブル 1 • InstallShield Express 用語集

用語	定義
カスタム アクション	Windows Installer 組み込まれた標準アクションに対して、インストール作成者によって作成されたアクション。アクションは、アプリケーションのインストール、アンインストール、またはメンテナンス中に実行される関数をカプセル化します。
簡易ユーザー インターフェイス (UI)	簡易 UIは、インストール時に作成されたモードレス ダイアログ、ビルトイン モーダル エラー メッセージ、および ディスク プロンプト メッセージを表示します。この UI レベルは、作成されたモーダル ダイアログの表示はしません。この UI レベルでは、インストーラーの内部ユーザー インターフェイス 機能が使われます。「モーダル」、「モードレス」および「内部ユーザー インターフェイス」も参照してください。
環境変数	ターゲット システム上の複数のプログラムでアクセスできる変数。
間接ビルド	ペイロードファイルのみにリファレンスを作成するフルビルド。つまりインストール時、ペイロードはビルド時にあった場所と同じ場所になくてはなりません。たとえば、ペイロードがネットワーク サーバーにある場合、インストールは同じネットワーク サーバー上でペイロードを探します。
完全アップデート	製品のアップデート バージョンで以前のバージョンを上書きするインストール。
完全ユーザー インターフェイス (UI)	インストーラーの内部ユーザー インターフェイス (UI) 性能レベルのひとつ。完全 UI レベルでビルドされるインストール パッケージは、内部 UI に含まれるモーダル ダイアログおよびモードレス ダイアログの両方を表示することができます。
管理	物事を管理する処理またはその動作。
管理インストール	管理インストールはデータファイルをユーザーが指定したディレクトリにコピー (および展開) しますが、ショートカットの作成や COM サーバーの登録、またはアンインストールログは作成しません。
[管理] シーケンス	ユーザーが /a コマンドライン オプションを使ってインストールを起動した場合に実行されるアクションのリスト。これは、ネットワーク管理者がネットワーク上に共通のインストールポイントを提供し、複数のユーザーに対して異なるインストール基準を設定する場合に便利です。各 [管理] シーケンスは、[ユーザー インターフェイス] シーケンスと [実行] シーケンスから構成されます。
管理者権限	コンピューター ユーザーに与えられる最も高いレベルの権限。ネットワーク環境では、システムのセキュリティを確保し、コンピュータのハードウェアおよびソフトウェアに対するダメージを防ぐために、異なるレベルのアクセス許可が必要です。管理者権限を持つユーザーはソフトウェアのインストール及びアンインストール、並びにコンピューターの構成を変更することが可能です。管理者権限は通常 Windows NT 4.0、2000、または XP マシンに付属していますが、Windows 95、98、または ME マシンには付属しません。
キー パス	コンポーネントの有無を検出するために Windows Installer が使用する各コンポーネントの一意のレジストリ値。コンポーネントにはキーファイルまたはキーパスのいずれかを割り当てることができますが、両方を割り当てることはできません。

テーブル 1・InstallShield Express 用語集

用語	定義
キーファイル	コンポーネントの有無を検出するために Windows Installer が使用する各コンポーネント専用に 1 つだけ存在するファイル。コンポーネントの詳細設定やショートカットを作成するには、キーファイルを指定する必要があります。
機能	製品の機能についての論理表現。たとえば、インストールにデータベース、メイン アプリケーション ファイル、およびヘルプ システム ファイルを含む場合があります。データベース、アプリケーション、およびヘルプシステムはすべて製品の機能です。機能はエンドユーザーに表示されますが、機能を構成する各ファイルは機能のコンポーネント内部にあります。「コンポーネント」も参照してください。
基本の UI	インストーラーの内部ユーザー インターフェイス (UI) 性能レベルのひとつ。一般的に、基本の UI レベルを使ってビルドされるインストールパッケージは処理状況メッセージおよびディスクプロンプト メッセージを表示するモードレスのビルトインダイアログを表示します。作成されたダイアログは表示しません。
キャッシュ	*cash* と同じ読み方で、頻繁にアクセスされるデータの一時的な格納場所。キャッシュの目的は、頻繁にアクセスされる情報を簡単にアクセスできる場所へ保存することで、コンピューターの効率を高めることです。キャッシュには、メモリ キャッシュとディスク キャッシュの 2 種類があります。メモリキャッシュはデータおよびメインメモリ内でデータが保存されているアドレスを保存します。ほとんどのアプリケーションは同じデータに繰り返しアクセスするため、メモリキャッシュは便利です。ディスクキャッシュはメインメ モリを使用します。ハード ディスクが最近要求した情報、またはハード ディスクに最近書き込まれた情報を保存するのに使用されます。一般に、インストールは通常ディスク キャッシュを使用します。データがメイン メモリから読み出される、または書き込まれるとき、そのコピーがメモリキャッシュに保存されます。データが呼び出されるとき、コンピューターはまずメモリキャッシュをチェックしてからディスクキャッシュ、最後にメインメモリをチェックします。
キャビネット ファイル	いくつかの圧縮ファイルを含む単一ファイル。アプリケーションのインストール中、圧縮ファイルは圧縮が解除され、コンピューターにコピーされます。キャビネットファイルはソフトウェア配布中に容量と時間を節約できるという点で効率的です。キャビネット ファイルは通常ファイル拡張子 .cab を持ちます。ファイルが破損または不足している場合はインストールの完了を妨げることになります。オペレーティングシステム ファイル、または InstallShield ファイルの不足または破損ファイルは、キャビネット ファイルから抽出して置換する必要があります。
キャプション	ウィンドウの表題にあたるテキスト。ウィンドウキャプションは、スクリーン上でアクティブウィンドウの位置を変更する際にユーザーがクリック アンド ドラッグする部分です。
ギャラリー	(4) 共有可能な使用できるリソースのコレクション。
行	特定の要素を記述する関連する列のセット。*レコード*とも呼ばれます。

テーブル 1 • InstallShield Express 用語集

用語	定義
グローバルー意識別子 (GUID)	製品を識別するために InstallShield が生成する長い数値の文字列。InstallShield で文字列 GUID を入力するとき、常に {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX} という形式を使用します。
結果セット	SELECT ステートメン トの実行によって作成される行のセット。
検証	検証処理。データが適切な形式であることを確認すること。
構造化クエリ言語 (SQL)	データベースを操作するときに使用される言語。
コスト分析	ファイル コスト分析は、現在のインストールで必要なディスク容量の合計を判断するために InstallShield が利用する処理です。
コマンド	(1) コンピューターに与えられる指示。 (2) 特定のタスクを実行するために オペレーティング システムがコマンドラインから、または [コマンドプロンプト] ウィンドウから実行する指示。インストール中にエラーが発生した場合、たとえばユーティリティを実行するため、コンピューターのディレクトリを検索するため、またはファイルを削除するためなどの目的でコマンドを実行する必要がある場合があります。
コマンド プロンプト ウィンドウ	[コマンドプロンプト] ウィンドウは MS-DOS へのインターフェイスです。[コマンドプロンプト] ウィンドウへアクセスするには、[スタート] メニューで [ファイル名を指定して実行] をクリックし、Windows 95、98、および ME では **command.com** を入力、Windows NT、2000、または XP では **cmd.exe** を入力します。インストール中にエラーが発生した場合、たとえばコマンドを入力してユーティリティを実行する、またはコンピューターのディレクトリを検索するなどの目的で [コマンドプロンプト] ウィンドウを開く必要があります。
コマンドライン	コマンドが入力される [コマンドプロンプト] ウィンドウの領域。コマンドラインは、実行可能ファイル (例、.exe、.bin) にコマンドを渡すのに利用されます。
コマンドライン オプション	コマンドラインオプションは、コマンドの実行方法を変更するコマンドへの引数です。 「コマンド」も参照してください。
コミット実行	インストールがファイルの転送、COM サーバーの登録、およびショートカット並びにレジストリ エントリの作成を完了した時に発生する InstallFinalize アクションの完了と同時に Windows Installer アクションを実行します。
コンテキスト メニュー	コンテキストメニュー (右クリックメニューまたはポップアップメニューとも呼ばれる) は、デスクトップ、Windows エクスプローラー、またはアプリケーションにある項目を右クリックすると表示されるメニューです。
コンパクト ディスク (CD)	電子録音、保存、およびオーディオ、ビデオの再生、並びにその他のデジタル形式の情報に利用されるディスク。

テーブル 1・InstallShield Express 用語集

用語	定義
コンパクト ディスク 読み取り専用メモリ (CD-ROM)	読み取り専用のコンパクト ディスク。
コンピューター ベース トレーニング (CBT)	コンピューターを用いて行なわれる指導。人材を指導するために CBT ではよくグラフィック、サウンド、および動画が用いられます。
コンポーネント	インストール開発者の視点から見たアプリケーションの要素。コンポーネントはエンドユーザーには表示されません。エンドユーザーがインストール用の機能を選択すると、インストーラーによってその機能に関連したコンポーネントが判断され、それらのコンポーネントがインストールされます。アプリケーションのコンポーネントには実行可能バイナリファイル、データファイル、ショートカット、ヘルプ システムファイル、およびレジストリ エントリなどが含まれます。
コンポーネント オブジェクト モデル (COM)	コンポーネントベースのアプリケーションの作成を可能にする Microsoft が開発したソフトウェア アーキテクチャ。コンポーネントベースのアプリケーションは、その他のコンポーネントおよび他のアプリケーションが機能を利用できるようにし、プログラムに機能を追加します。
サービス	Windows Installer ベースまたは InstallScript ベースのプロジェクトでは、これはコンピューターがオンになっているとき常にバックグラウンドで実行されるプログラムです。サービスはソフトウェア インストール、プロセス モニターリング、ファイル転送、タスク スケジューリング、ネットワーク管理、その他多数のユーザー側の対話操作を必要としないタスクを実行します。Win32 では、サービスはサービス コントロール マネージャーによって管理されます。
サービス コントロール マネージャー (SCM)	サービスに使用するシステムのデータベースをメンテナンスし、これらのサービスを制御するインターフェイスを提供します。
サービス パック (SP)	既存の問題を修正し、製品強化を図るためのソフトウェア製品のアップデート。製品の次回のバージョンには前回までリリースされたすべてのサービスパックが取り込まれています。
再インストール	製品が既にマシンにインストールされている状態でインストールをもう一度実行すると、既存のファイル、ショートカット、およびレジストリ エントリが上書きされ、製品が再インストールされます。
再配布可能ファイル	インストールまたはアプリケーションと共に合法的に配布することができるマージ モジュール、オブジェクト、または他のファイル。
サイレント インストール	ユーザー インターフェイスまたはエンドユーザーの介入なしで実行されるインストール。

テーブル 1 • InstallShield Express 用語集

用語	定義
差分リリース	既存リリースの指定したセットに存在しないファイル（複数可）のみを含むリリース。差分リリースは、既存のリリースによってインストールした製品のバージョンをアップデートするのに使用します。
参照カウント	共有ファイルがインストールされるたびに増えるカウントの集計。HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥ に保存されます。Windows¥CurrentVersion¥SharedDLLs の下に保存されます。
シーケンス	インストール プロジェクトで連続して実行される 1 セットのアクションとダイアログです。
シーケンス テーブル	インストール プロセスの制御および実行順序を指定するアクションをリストするテーブル。
自己修復	「自動修復」を参照してください。
自己登録ファイル	Windows レジストリにそれ自身についての情報をエントリーして、アンインストール時にその情報を削除することができるファイル。他の種類のファイルも情報をレジストリにエントリすることなく使用することができます。自己登録ファイルをインストールするには、そのファイルを希望の場所にインストールし、それからコンピューターに登録します。
システム ポリシー	システムが準拠していなければならない規則。
システム権限	各ユーザーが利用できるシステム、プログラム、および関数。
実行可能（.bin）ファイル	UNIX プラットフォームで実行可能なプログラムを含むファイル。
実行可能（.exe）ファイル	Windows プラットフォームで実行可能なプログラムを含むファイル。
実行時	ターゲットマシンでアプリケーションをインストール/アンインストールするために、インストールが、インストーラーとインタラクトする時間。
実行スクリプト	Windows Installer インストール用の インストーラー アクション。実行スクリプトはインストールの取得段階で生成され、実行段階で実行されます。
実行段階	インストーラーのアクションが実行される段階。
自動修復	インストーラーがアプリケーションを元の状態に戻す自動復元。機能と関連付けられているファイルが不足、または破損している場合、アプリケーションの部分的、または完全な再インストールが必要な場合があります。
修飾コンポーネント	ポインターに類似する単一レベル インダイレクションのメソッド。修飾コンポーネントは、主として、並列機能を持つコンポーネントを複数のカテゴリーにグループ分けするのに使用されます。

テーブル 1・InstallShield Express 用語集

用語	定義
修復モード	修復モードでは、紛失または破損されたファイル、不正確なレジストリ エントリおよび自己登録ファイル等のすべての種類の不具合がインストールによってチェックされます。不具合が見つかった場合、インストールは修正を試みます。「メンテナンス モード」も参照してください。
取得段階	インストール処理で、インストーラーがデータベースを参照して指示を得る段階。取得段階の後に実行段階が続きます。
ショートカット	アプリケーションをポイントするファイル。ショートカットをクリックするだけでアプリケーションを素早く開くことができます。ショートカットは、通常 [スタート] メニューまたはデスクトップに配置されます。「アドバタイズ ショートカット」も参照してください。
昇格された権限	標準の権限よりも高いレベルの権限で、通常は一時的なもの。権限とは、システム上で特定の動作を行うことができる権限のことです。
条件	プロパティの値を固定値と比較する、またはプロパティが定義されているかどうかを判断する論理のステートメント。条件はオブジェクト、機能、コンポーネント、アクション、またはその他の条件と関連付けられている項目がインストール、または実行される場合 true 評価しなくてはなりません。
条件付きインストール	特定条件が満たされた場合のみ、その項目をインストールする。たとえば、オペレーティング システム条件がアプリケーションのインストールに関連付けられている場合、ターゲットマシンが特定のオペレーティング システムを実行している場合のみアプリケーションをインストールします。
消費/コンシューム	インストールがマージ モジュールまたは InstallShield オブジェクトなどの再配布可能ファイル パッケージに対して行なう処理。
進行状況バー	実行ファイルの進行状況を視覚的に表示するバー。
伸張	スペース文字を削除してファイルのサイズを最小化する圧縮解除プロセス。この処理はファイルを元の状態に戻します。
スキーマ	データ ソース内における現在のテーブルおよびビューの構造の記述。スキーマは各テーブルが持っている列、各列のデータ タイプおよびテーブル間の関係について説明します。
スパイウェア	個人または組織から知られることなしにその情報を集めることを手助けするテクノロジ。スパイウェアは、インターネット上で、個人のコンピューター上に置かれ、ユーザーについての情報をひそかに集め、それを広告主または他の関連団体に伝えるプログラムです。スパイウェアはソフトウェア ウィルスまたは新しいプログラムをインストールしたときの結果としてコンピューターにロードすることが可能です。正確には、ユーザーの知る知らないに限らずインストールされるデータ収集プログラムについて、ユーザーがどのデータが収集されていて、それが誰と共有されているかを理解している場合はスパイウェアとは呼びません。

テーブル 1・InstallShield Express 用語集

用語	定義
スプラッシュ画面	製品のインストールの起動中にエンドユーザーに表示されるイメージ。表示するイメージ、表示時間、ローカライズされたイメージの有無を指定することができます。
スモール アップデート	インストール済みパッケージと最新バージョンのパッケージのバージョン番号が同じ場合にパッケージをアップグレードするパッチ。
セーフ モード	セーフモードは、Windows 95、98、ME および 2000 で提供されているトラブルシューティング モードです。セーフモードでコンピュータを起動すると、オペレーティング システム、マウス、キーボードおよびディスプレイドライバのみがロードされます。コンピューターが何をしても起動しないときでも、セーフモードを使用すると起動できる場合があります。また、セーフ モードでは、オペレーティング システムをトラブルシュートして適切に機能していない箇所を見つけることができます。
制限付きパブリック プロパティ	インストール作成者が設定または変更可能範囲を制限することができるグローバル変数。制限付きパブリック プロパティの操作は、通常、システム管理者のみが行うことができます。制限は、安全な環境を維持するために使用されます。
製品	実際にインストールまたはアンインストールするアプリケーション。
製品コード	製品を一意に識別する文字列。
セキュリティ ツール	トロイの木馬、スパイウェア、アドウェアおよびハッカー ツールなど破壊的なペストをコンピューター上で検出し除去するプライバシー ツール。ウイルス対策およびファイヤー ウォール ソフトウェアを補完し、エンド ユーザーの既存のセキュリティを回避し個人情報を侵害することができる非ウィルス系の悪質なソフトウェアからのプロテクションを強化します。
絶対パス	指定されたドライブのルート ディレクトリを起点にしてファイル検索をするのに必要な情報をすべて含む絶対パス。たとえば、C:¥Program Files¥InstallShield は、C ドライブにインストールされたときの InstallShield フォルダへの絶対パスです。
セットアップ	「インストール」を参照してください。
セットアップ プロジェクト	「インストール プロジェクト」を参照してください。
ソース リスト	インストーラーがインストール ファイルを検索する場所を指定するリスト。ソース リストのエントリは、ネットワークの場所、URL およびコンパクト ディスクが可能です。
相対パス	現在のドライブ上の現在のフォルダーを起点にして、ファイル検索に必要な情報をすべて含むパス (例、InstallShield¥Support)。フォルダーは、現在のディレクトリに存在する場合のみ、相対パスで検索することができます。

テーブル 1・InstallShield Express 用語集

用語	定義
即時実行	インストール段階に使われる用語。一般的に InstallScript プロジェクトでインストーラーがスクリプトをビルドするときに直ぐに実行されるカスタム アクションを指します。
ソフトウェ ア開発キット(SDK)	開発者が特定のオペレーティング システムをターゲットしたアプリケーションおよびライブラリを作成することができるようにデザインされたドキュメント、サンプル、コマンドライン コンパイラ、デバッグ ツール、ユーティリティ、およびツール。SDK は通常、オペレーティング システムの製造元により提供されます。
ダイアログ	インストール/アンインストールの最中にエンドユーザーに表示される情報ウィンドウ。情報ウィンドウによって、エンドユーザーは情報を読み込んだり指定したりしながらオペレーションとインタラクトできます。「ウィザード」も参照してください。
ダイアログ ボックス	アプリケーション インターフェイス内で表示され、ユーザーが情報を入力したりコマンドを指定したりすることができるウィンドウ。
ダイナミック リンク ライブラリ (DLL)	他のアプリケーションから呼び出される関数を含むコードベースの共有ファイル。
タスク マネージャー	コンピューターのパフォーマンス、およびコンピューター上で実行中のアプリケーション並びに処理についての情報を提供する Windows プラットフォームで利用可能なツール。場合によっては、インストールする前に、競合を避けるために [タスクマネージャー] を使って実行中のアプリケーションを終了する必要があります。
チェックサム	破損がないかをテストするためにデータに適用される計算値。チェックサムは、ファイル内のデータのバイトを系統立てて順に結合することにより得られます。転送やストレージ圧縮の後、チェックサムの計算が再び行われ、結果が以前の結果と比較されます。数値が一致しない場合は、保存または転送されたデータにエラーがある可能性があります。
遅延実行	インストール スクリプトの実行と同時にインストール アクションを実行する。
抽出	キャビネット ファイルなどの圧縮ファイルからファイルを除外、または圧縮解除すること。圧縮されたファイルは、1 つまたは複数の他のファイルを含む単一ファイルです。圧縮ファイルはサイズが縮小されているので、容量を節約できます。圧縮済みファイルを使用するためには、まず圧縮ファイルからそれを抽出する必要があります。ファイルを抽出するには、Extract.exe と呼ばれるコマンドライン ユーティリティを利用することができます。インストール中に破損または不足しているオペレーティング システム ファイルまたは InstallShield ファイルは、インストールを完了するために圧縮ファイルから抽出する必要がある場合もあります。
ツリー	木のような構造を持つ追加データ。
ツリーノード	データ構造は、「子」ノードを含まない場合と、1 つ上含む場合があります。ノードはツリーの「ルート」です。「子」を含まないツリーノードは、一般的に「リーフ」と呼ばれます。

テーブル 1 • InstallShield Express 用語集

用語	定義
データベース	データベース管理システム (DBMS) の個別の部分からなるデータの集まり。
データベース ハンドル	データベース関数が呼び出されたときにデータベースを指定する構成要素。
データベース関数	データベースで作動する関数。
データベース管理システム (DBMS)	編成、保存、および多くのユーザー用のデータの読み出しをコントロールするアプリケーションのセット。
データベースのコミット	Windows Installer データベースについて蓄積された変更。変更はデータベースがコミットされるまで、つまり **MsiDatabaseCommit** が呼び出されるまで反映されません。
テーブル	データ行の集まり。
デジタル署名	アプリケーション内のコードがリリース後に改ざんまたは変更されていないことをエンドユーザーに対して保証するため、アプリケーションにデジタル署名を付けることができます。アプリケーションにデジタル署名を付けると、エンドユーザーが製品をダウンロードするときにデジタル証明書が表示されます。
展開	(1) 圧縮ファイルをオリジナルのサイズに復元すること。圧縮ファイルとは、たとえばキャビネット ファイルのような 1 つまたは複数の他のファイルを含む単一のファイル。圧縮ファイルはサイズが縮小されているので、容量を節約できます。 (2) 圧縮ファイルを復元するのに利用される DOS コマンド。インストール中に破損または不足しているオペレーティング システム ファイルまたは InstallShield ファイルは、インストールを完了するために圧縮ファイルから展開する必要がある場合もあります。
テンプレート	規定する、またはひな型として利用できるもの。
動画	動作を表現するために、1 秒間に特定数のフレームが交互に表示される複数のグラフィックイメージ。
同期実行	非同期実行の反対。非同期実行の場合、プロセス制御は全プロセスが完了するまでリリースされません。
統合開発環境 (IDE)	ソフトウェア プログラムのインターフェイス。「インタラクティブ開発環境」とも呼ばれます。
ドラッグ	アイテムを選択して別の場所へ移動させる動作。
トランスフォーム	トランスフォーム(.mst ファイル)は、2 つのインストール データベースの違いを表します。たとえば、ネットワーク管理者がある製品を会社の部署によって異なる構成で配布する場合があります。この場合、製品の各構成に対しトランスフォームを作成し、必要に応じて適切なトランスフォームを適用します。

テーブル 1 • InstallShield Express 用語集

用語	定義
トランスフォーム エラー条件フラグ	トランスフォームのエラー条件を設定する際に使用するプロパティのセット。
トランスフォーム検証フラグ	トランスフォームが Windows Installer パッケージに適用可能であることを検証するのに利用されるプロパティのセット。
内部整合性評価プログラム (ICE)	エラーの可能性 (またはエラーそのもの) を警告するために Windows Installer データベース上で実行できるテスト (Orca、Msival2、InstallShield Premier または InstallShield を使用)。
内部ソース ファイル	インストール アーカイブ ファイルに含まれるファイル。
内部ユーザー インターフェイス	インストール中にエンドユーザーに表示される、グラフィック ユーザー インターフェイスの作成に利用できるインストーラーのビルトイン機能。
名前付きインスタンス	インスタンスとは、規定のサーバー上にある SQL Server の完成された独立インストールです。ひとつのデフォルトのインスタンスおよび名前付きインスタンスの任意の番号 (16 まで) を単一の サーバーにインストールすることができます。インスタンス間で共有される管理ツールおよびクライアント コネクティビティ コンポーネント以外、各インスタンスは原則的にスタンドアロンです。各インスタンスは独自のセキュリティを持ち、個別に開始、および停止させることができます。さらに同じサーバー上にあるその他のインスタンスに対して個別にサービスパックを当てることができます。
ナレッジ ベース (KB)	ソフトウェアのヘルプ情報の範囲外にあるテクニカル情報、手順、記事を集めたもの。ナレッジベースでは、ヒント、こつ、およびテクニックに加え、FAQ(よく聞かれる質問)、技術およびデザイン上の問題に関する記事を参照することができます。Revenera の Web サイト (http://www.installshield.com) では、ナレッジベースの内容が定期的に更新されています。
ネスト インストール	実行中のインストール (「親製品」と呼ぶ) から別のインストールパッケージ (「子製品」と呼ぶこともある) をインストールまたは削除するカスタム アクションの一種。
ネットワーク	ハードウェア、ソフトウェア、および情報を共有するため相互に接続された 2 台以上のコンピューター。インストール中、ソフトウェアを適切にインストールするため、ネットワーク上にあるインストールファイルをコンピューターのハード ドライブへ移動する必要がある場合があります。
ハード ドライブ	コンピューターの主要な記憶デバイス。ハードドライブには、磁気的にデータが読み書きされるディスクが含まれます。ハード という用語はハードドライブで利用されるアルミニウムやガラス製のディスクを、プラスチックでできたフロッピーディスクから区別するのに使われます。
ハイパーテキスト マークアップ言語 (HTML)	ハイパーリンクやテキスト形式のマークアップを含む Web ページを作成するための言語。

テーブル 1 • InstallShield Express 用語集

用語	定義
配布メディア	ユーザーに配布される CD-ROM またはその他のメディア フォーマット。
パス	コンピューター オペレーティング システムで、パスは特定のファイルへのファイル システム内の経路。パス名はそのパスを詳述します。各オペレーティング システムは、パス名の指定に独自のフォーマットを持ちます。DOS、Windows、および OS/2 オペレーティング システムでは、driveletter:directorynamesubdirectorynamefilename.suffix フォーマットが使用されます。UNIX ベース システムは、/directory/subdirectory/filename フォーマットが使用されます。
パス変数	プロジェクトを移動したり、フォルダー構造を変更するたびに各ソース ファイルのパスを変更する必要がなくなるよう、中央ロケーションに 1 度だけ定義するだけで済むロケーションを示す変数。代わりにパス変数を使用して、l共通に使われるパスを1 度定義するだけで済みます。共通パスは、インストール プロジェクトの開発中に使用されます。これらのパスは、アプリケーションがインストールされるターゲットマシンには適用されません。代わりに、インストール プロジェクトに含める必要があるのソースファイルにリンクするために使用されます。プロジェクトをビルドする際、これらのリンクが評価され、ポイント先のファイルがインストール パッケージに組み込まれます。
パッケージ	インストールデータベースファイル、アプリケーションのファイル (インストール データベースファイルとは別、またはインストールデータベースファイルに圧縮されたもの)、および 実行可能ファイル (上記すべてのファイルを圧縮して含むことが可能) を含む、インストールの実行に必要なすべてのファイル。
パッケージ コード	インストール パッケージの Globally Unique Identifier (GUID)。「globally unique identifier (GUID)」も参照してください。
パッチ	アプリケーションのファイルおよびインストールを特定バージョンにアップデートする、または以前のバージョンのバグを修正するのに必要なアプリケーションの一部のみを含む、特殊なタイプのインストール パッケージ。
パッチ ファイル	パッチに利用されるパッチパッケージ。パッチパッケージ (.msp) ファイルには、インストールされた 1 つ以上の製品バージョンをアップグレードするのに必要なトランスフォームおよび指示が含まれています。Windows Installer はパッチパッケージを利用してローカルインストールおよび管理インストールをパッチします。パッチパッケージは通常のインストールパッケージのようなデータベースを含みません。その代わり、ターゲットとするインストールパッケージのデータベースにパッチ情報を追加するデータベーストランスフォームが 1 つ以上含まれます。インストーラーはこの情報を利用して、パッチパッケージのキャビネット ファイル ストリームに格納されているパッチファイルを適用します。
パッチの適用	アプリケーション全体ではなく、変更された部分のみを置換するインストールのアップデート方法。これによって、エンドユーザーは製品全体ではなく規模の小さい製品のパッチをダウンロードすることができます。

テーブル 1 • InstallShield Express 用語集

用語	定義
パネル	エンドユーザーが構成できる 1 つ以上の関連する設定を含むウィザード内にあるウィンドウ。「ウィザード」も参照してください。
パブリック プロパティ	エンドユーザーまたはシステム管理者によってその値が設定されるグローバル変数。パブリック プロパティは、インストール中、ユーザー インターフェイスとの対話、コマンドラインでのプロパティの設定、トランスフォームの適用、あるいは標準またはカスタム アクションの使用によって設定または変更することができます。プライベート プロパティの値とは異なり、パブリック プロパティの値は変更可能です。パブリック プロパティ名には、小文字は使えません。
パブリッシュ	アドバタイズ (「ジャストインタイム」インストール) の一種。インストール中にコンポーネントのユーザー インターフェイス要素は作成されませんが、コンポーネントは [コントロール パネル] の [プログラムの追加と削除] アプレットからインストールすることができます。また、インストールされたコンポーネントがインストーラーからパブリッシュされたコンポーネントを要求した際にもインストールできます。
バリューパック (VP)	インストールされたアプリケーションの機能をアップグレードするインストール。
左クリック	マウスの左ボタンを選択してクリックする動作。
ビットマップ	ビットの配列として表示されるイメージ。ビットマップ グラフィックでは、イメージは集合体としてパターンを形成するピクセルと呼ばれる小さな四角の集まりでスクリーン上に表示されます。イメージ内の各ピクセルは、1 つまたは複数のビットに対応します。
非同期実行	インストーラーがメインのインストールを実行中に独立してスレッドを実行し続けるカスタム アクションです。
非分割 CD/DVD メディア タイプ	ペイロードは CD/DVD アーカイブにレイアウトされますが、複数メディアへの分割はされません。
標準アクション	インストール開発ソフトウェア製品に組み込まれているアクション。InstallShield 製品は、カスタム アクションの作成および使用もサポートしています。「アクション」、「カスタム アクション」も参照してください。
ビルボード	インストール中に表示することが可能なマーケティング メッセージなどのイメージ。
ファイル	ファイルシステム内のデータ格納要素。
ファイル拡張子	ファイルに格納されているデータの種類を示す、最終ドットに続くファイル名の一部。
フォーマット済み	データが保存または表示のために分割、または配列された状態。

テーブル 1 • InstallShield Express 用語集

用語	定義
フォールス ポジティブ	テストで true 評価を出すが実際はそうではないもの。多くの場合、false positive はウィルス対策ソフトウェアに存在します。たとえば、ウィルス対策ソフトウェアは InstallShield ファイルの中にウィルスを発見したと報告するが、実際はそうではない場合があります。ソフトウェアのウィルス定義をアップデートする必要があります。
復元性	必要に応じてコンポーネントを再インストールするアプリケーションの機能。コンポーネントが誤って削除されたり壊れたりした場合でも、Windows Installer の技術によりアプリケーションの自己修復が可能です。
プラットフォーム	インストール プログラムがターゲットにするオペレーティング システム。あるインストール テクノロジは、特定のプラットフォームへのインストールの作成に限られていることもあります。
プレビュー モード	ユーザー インターフェイスのデザイン、またはダイアログおよびビルボードの現在の概観を表示するモード。*プレビューモード*は Windows Installer で使われている用語です。
プロパティ	インストールまたはアンインストール中にインストーラーまたはアンインストーラーによって使用されるインストール プロジェクト内のオブジェクトに配置された値。
分散コンポーネント オブジェクト モデル (DCOM)	拡張型コンポーネント オブジェクト モデル (COM)。コンポーネントベースのアプリケーションの作成を可能にする Microsoft が開発したソフトウェア アーキテクチャ。
分離アプリケーション	元のアプリケーションが開発およびテストされたときに使用されたすべてのバージョンのコンポーネント (.dll ファイルなど) が常にロードされるように変更されるアプリケーション。
変換プロジェクト	(1) 別のタイプに変換されたファイル。 (2) InstallShield の MSI/MSM オープン ウィザードは、インストール プロジェクト (.msi ファイル) やマージ モジュール (.msm ファイル) を、InstallShield で変更とビルドが可能な InstallShield インストール プロジェクト (.ism ファイル) に変換するためのツールです。
編集フィールド オブジェクト	ユーザーによるフィールドへのテキスト入力を可能にする対話型オブジェクト。
変数	コンピューターに格納されている値。値は、印刷可能な文字、数値、およびテキストを使って作成することができます。値が変化しない定数とは違い、変数はいつでも値を変更することが可能です。
ホットフィックス、または、修正プログラム	アプリケーションの次のリリースまで配布を延期することができない、重要かつ規模の小さいバグ修正。ほとんどのホットフィックスは小さいパッチ形式で、ソフトウェア ベンダーの Web サイトからダウンロードすることができます。
ボリューム	1 つのメディア上にある、ブロック単位の空きスペースの総容量。

テーブル 1・InstallShield Express 用語集

用語	定義
マージ モジュール (MM)	ランタイム .dll ファイルや仮想マシンなど特定の機能のインストールに必要なすべてのロジックおよびファイルを含むパッケージ。マージ モジュールは一度ビルドすると、他のセットアップ プロジェクトに追加できます。
マイナー アップグレード	インストールに加えられる変更の為に製品コードを変更する必要がない製品のアップデート。
マクロ	(1) ひとまとまりの命令群からなる命令。
マネージド アプリケーション	アプリケーションのインストールにシステム権限が使われた場合、アプリケーションは「マネージド アプリケーション」と呼ばれます。
マルチメディア	サウンド、ビデオ、グラフィック、および動画など、情報を伝える為にまとめて利用される様々なメディアの集合。
メジャー アップグレード	ProductCode プロパティで変更が必要な製品の完全アップデート。同じマシンに 同じ製品の 2 つのバージョンをインストールする場合、新しい製品コードが必要になります。
メタファイル	標準 Windows メタファイル形式で保存されたイメージメタファイルは 1 セットのドローイング命令群です。Windows メタファイルは .wmf というファイル拡張子を持っています。
メディア ディスクリプタ (MD)	インストール可能な単位 (IU) の意味。通常ファイルとして存在するデータの物理的なソースに対して、デプロイメント ディスクリプタ (DD) で要素のマッピングを行います。
メンテナンス モード	既に製品がインストールされているシステム上で、2 回目 (およびそれ以降) にインストール プログラムを実行をすると、インストールはメンテナンス モードで行われます。このメンテナンスモードでユーザーは、最初のインストールで選択した機能の変更、インストール済みの機能の修復、およびプログラム全体の削除が行えます。
モーダル	アプリケーションが続行する前に、ダイアログを表示してユーザーからの返答を待ちます。その間、ユーザーは他のウィンドウまたはダイアログ ボックスを使用できません。「モードレス」も参照してください。
モードレス	ユーザーが他のウィンドウ、ダイアログ ボックスと対話できるようにします。「モーダル」も参照してください。
文字列テーブル	すべてのサポート言語に対する文字列 ID、値、コメントを保存するデータベース。これらの文字列は、実行時にエンドユーザーに表示されるダイアログおよびメッセージ ボックスで使われます。
ユーザー インターフェイス (UI)	ソフトウェアのユーザーへのアクセス。

テーブル 1 • InstallShield Express 用語集

用語	定義
ユーザー プロファイル	ショートカット、お気に入り、アプリケーション、ディスプレイ、並びにハードウェアの設定など、各エンドユーザーの設定についての記録。ユーザープロファイルは、複数のユーザーがそれぞれの環境設定を保持しながら 1 台のコンピューターを共有することを可能にします。
ライブラリ	予備に保存されている類似オブジェクトのコレクション。たとえば、ソースコードまたはオブジェクトコード形式のプログラム、データファイル、スクリプトおよびテンプレート。プログラム ライブラリは、プログラマーがコードを書いているときに "呼び出す" ことができるプリコンパイル済み、再利用可能プログラム ルーチンのコレクションです。
リッチ テキスト フォーマット (RTF)	フォントやマージン等のフォーマット情報を表示する特殊コマンドを含む Microsoft のファイル フォーマット。RTF を使用すると、異なるワードプロセッサーおよびオペレーティング システム間でファイルを交換することができます。
リフレッシュ ビルド	このタイプのビルドはカスタムコードのみ再コンパイルします。また、全インストールを再ビルドすること無しに変更があった単一ファイルのみ再ビルドおよび追加/変更ができるので、単一ファイルの置換も可能です。このタイプのビルドは、InstallScript プロジェクトでのみ使用することができます。
リモート プロシージャ コール スタブ (RPC)	他のコンピューター上のサービスを要求するのに使用されるプログラムの中の小さいサイズのルーチン。RPC は、プロトコルまたは予め合意されたデータ送信のためのフォーマットで、これをプログラムで使用することにより、他のコンピューターにあるプログラムからサービスを要求することが可能になります。スタブはプログラムからのリクエスト受け入れて、それをリモートプロシージャへ転送します。プロシージャが完了すると、スタブは結果を受け取り、リクエストをしたプログラムへまた戻します。
リリース ノート	アプリケーションの配布と共に含まれるファイル。このファイルには、アプリケーションのインストールおよびアンインストールに関する重要な情報が含まれています。また、他の製品ドキュメントに含まれていない可能性がある情報も含まれることがあります。
リリース ビルド	リリースに適した完全再ビルド。
レジストリ	個人設定およびコンピューターにインストールされたソフトウェアおよびハードウェアを記録するために Windows オペレーティング システムによって使用される中央データベース。インストール時、インストールに関して選択された項目はレジストリに書き込まれます。
レジストリ ファイル	定義済み形式のテキストファイルで、レジストリにマージできるキーと値が含まれています。
列	縦に並んだ一連の情報。
ローカリゼーション	製品またはサービスを特定の言語および文化に翻訳適合するプロセス。

テーブル 1 • InstallShield Express 用語集

用語	定義
ロード順グループ	サービスの中には、起動前に他のサービスが既に実行されていなければならないものがあります。そのため、サービスをグループ化して指定した順序でロードするよう設定する必要があります。サービスのロード順グループは、HKEY_LOCAL_MACHINE¥System¥CurrentControlSet¥Control¥ServiceGroupOrder にリストされています。 Set¥Control¥ServiceGroupOrder にリストされています。サービスの ″起動タイプ″ プロパティにより、グループ内でいつロードされるかが決定されます。
ロールバック	インストールは、インストール プロセス中に行われた変更をすべて記録しており、エラーが発生してインストールが中止された場合でも、これらの変更は「ロールバック」され、マシンは元の状態に復元されます。
ログ データベース (LogDB)	インストール/アンインストール中にログを記録するように構成されたすべてのログを含むターゲットマシン上にあるデータベース。
ログ ファイル	ファイルには、コンピューター ワークステーションまたは Web サーバー上でソフトウェア プロセス中(インストール、ビルド、ダウンロード等) に発生したアクティビティの記録が含まれています。たとえば、Web サーバーは、すべてのサーバーへのリクエストをリストし、コピーされたファイル、転送されたバイト数、リクエストされたページ、イメージおよびファイルを要約して、ログファイルのメンテナンスを行います。ビルド ログ ファイルは、ビルドに含められたすべての機能、セットアップ タイプ、マージ モジュール、ダイナミック リンク、およびファイルをリストします。
ログの記録	インストール、ビルドまたはダウンロード等のソフトウェア プロセス のアクティビティをログファイルに記録するプロセス。
割り当て	アドバタイズの種類。管理者がアプリケーションをマシンに割り当てると、次回マシンを起動または再起動したときにインストール プログラムが自動的に実行します。管理者がアプリケーションをユーザーに割り当てると、インストール プログラムはユーザーのマシン上にある [スタート] メニューにショートカットを配置します。ユーザーがショートカットを選択する、または割り当てられたアプリケーションに関連付けられた文書を起動したとき、アプリケーションがインストールされます。

1

索引

記号

_ISCRIPT_ISDEV preprocessor constant
    サンプル コード 904
_ISCRIPT_ISDEV プリプロセッサ定数 903
_ISCRIPT_ISPRO preprocessor constant
    サンプル コード 904
_ISCRIPT_ISPRO プリプロセッサ定数 903
_ISCRIPT_VER プリプロセッサ定数 903
_serial_verifyCA_isx custom action 3761
_serial_verifyCA_isx_helper custom action 3761
.appx 1542
.cab ファイル
    InstallScript インストールから抽出 1901
    InstallScript インストールからの概要 1899
    InstallScript インストールから表示 1898
    InstallScript インストールから開く 1900
    InstallScript インストールについての .txt レポートを作成する 1901
    最大サイズの構成 382, 383
    制限 382, 383
.hdr ファイル 1899
    InstallScript インストールから表示 1898
    概要 1899
    ひらく 1900
    レポートを作成する 1901
.ilg 1902
    .txt ファイルに変換する 1904
    概要 1902
    参照 1902
    開く 1903
.isproj 1347
.isv ファイル 1957
.msi ファイル
    .ism ファイルへの変換 328
    IDE からの起動 1396
    ダイレクト モードで開く 2308
    複数を同時に実行 713
    変換エラーのラブルシューティング 3111
.msp ファイル 2309
    作成 2929
    編集 2309
.mst ファイル 1515
    2 つの .msi ファイルを比較して作成 1516
    削除 326
    単一の .msi パッケージを基に作成 1516
    適用する 1517
    デフォルトの応答のカスタマイズ 1520
    デフォルトの応答の変更 1521
.NET 1960
    32 ビット依存関係 624
    64 ビット依存関係 624
    カスタム アクション 926
        32 ビットと 64 ビットの違い 928
    再配布可能ファイル 733
    プロパティおよび依存関係 622
.NET Core カスタム アクション 928
.NET Framework
    バージョン 2.0、64 ビット 731, 734
    バージョン 3.0 731, 734
    バージョン 3.0、64 ビット 731, 734
.pfx 1366
.prq ファイル 2991
    ダウンロード用代替 URL を指定する 1635
    編集 2991
.swf 1209
.vdproj 477
    InstallShield プロジェクト (.ism ファイル) に変換またはインポート 477
.xml 554

.zip 1541
DO_NOT_BUILD 471

数字

-10000 - ユーザーによってプロセスがキャンセルされました 3280
-10001 - スイート ファイルが不足しています 3280
-10002 - スイート ファイルが重複 3281
-10003 - アプリケーション ファイルが不足しています 3281
-10004 - INI ファイルが不足しています 3281
11000 - Excluding TCPIP レジストリ エントリを除外中 3282
11001 - VMware が失敗しました 3282
11003 - コントロール パネル アプレット - Citrix 3283
11004 - コントロール パネル アプレット - ThinApp 3283
11005 - QuickTime 7.4.1 が原因で、致命的エラーが発生しました 3283
11006 - Adobe Distiller によって AdobePDFSettings が除外されました 3284
11007 - URL ショートカットの除外 3284
2 番目のウィンドウ 1160
  アンパサンドをテキストで表示する 1179
  キーボード ショートカット 1179
  空白の 2 番目のウィンドウを追加 1176
  コントロールを追加 1178
  タブの順番 1179
  背景 1167
27500 3115
27501 3116
27502 3116
27503 3116
27504 3116
27505 3116
27506 3117
27507 3117
27508 3117
27509 3117
27510 3118
27511 3118
27514 3118
27515 3118
27516 3119
27517 3119
27519 3119
27520 3120
27521 3120
27524 3120
27525 3120
27526 3120
27527 3121
27528 3121
27529 3121
27531 3121
27532 3121
27533 3121
27534 3121
27535 3122
27536 3122
27537 3122
27538 3122
27539 3122
27540 3122
27541 3123
27542 3123
27543 3123
27544 3123
27545 3123
27546 3123
27548 3124
27549 3124
27550 3124
27551 3124
27552 3124
27553 3125
27554 3125
27555 3125
32 ビット
  InstallShield 前提条件の要件 2163
  ビルド時の検証 1320
64 ビット
  InstallShield 前提条件の要件 2163
  ビルド時の検証 1320
  Oracle のインスタンスへの接続サポート 1250
  System32 フォルダー ソース ファイル 598
  オペレーティング システム、InstallScript インストールでターゲット 295
  オペレーティング システム、Windows Installer ベースのトールでターゲット 292
  オペレーティング システム、スイート/アドバンスト UI またはアドバンスト UI インストールでターゲット 296
  オペレーティング システム、ターゲットにする 291
  コンポーネント 292, 295
  自己登録 292, 295
  レジストリ リフレクション 662
-9000 - 不明な例外 3231
-9001 - 不明な COM 3231
-9002 - パッケージを開くときにエラーが発生しました 3231
-9003 - パッケージを保存するときにエラーが発生しました 3232
-9004 - ユーザーによってプロセスがキャンセルされました 3232
-9005 - 一時フォルダーの作成中にエラーが発生しました 3233
-9006 - パッケージの圧縮中にエラーが発生しました 3233
-9007 - 拡張子を持つファイルが見つかりませんでした 3234
-9008 - アイコンの抽出中にエラーが発生しました 3234
-9009 - 不明のプロバイダー 3234

-9010 – ターゲット ファイル名が正しくありません 3235
-9011 – MSI テーブルの読み込み中にエラーが発生しました 3235
-9012 – メソッドで予期しないエラーが発生しました 3236
-9013 – タイプ ライブラリが見つかりませんでした 3236
-9014 – ShellExecute が失敗しました 3236
-9015 – ドライバーの完全パスを判別できませんでした 3237
-9016
テーブルのコンテンツが無視されました 3238
-9017 – .NET 1.x アセンブリはサポートされていません 3238
-9018
カスタム アクションが無視されました 3238
-9019 – 条件付きコンポーネント 3239
-9020 – ディレクトリの親がヌル エラー 3240
-9021 – COM データを抽出できませんでした 3241
-9022 – Complus テーブル エラー 3241
-9024 – FileSFPCatalog 3242
-9026 – LaunchCondition テーブル警告 3242
-9027 – LockPermissions テーブル警告 3243
-9028 – MoveFile テーブル エラー 3244
-9029 – MsiDriverPackages テーブル エラー 3244
-9030 – ODBCTranslator テーブル警告 3245
-9031 – RemoveFile テーブル警告 3245
-9032 – RemoveIniFile テーブル警告 3246
-9033 – RemoveRegistry テーブル警告 3246
-9036 – ISCEInstall テーブル エラー 3247
-9037 – ISComPlusApplication テーブル エラー 3247
-9038 – ISPalmApp テーブル エラー 3248
-9039 – ISSQLScriptFile テーブル エラー 3248
-9040 – ISVRoot テーブル エラー 3249
-9041 – ISXmlFile テーブル エラー 3250
-9051
パッケージの圧縮中がキャンセルされました 3250
-9100 – パッケージ オブジェクトのインスタンスの作成が失敗しました 3250
-9101 – パッケージ オブジェクトの作成操作が失敗しました 3251
-9102 – ヘッダー情報の書き込みに失敗しました 3251
-9103 – Citrix の最終処理が失敗しました 3252
-9104 – Citrix の保存に失敗しました 3252
-9105 – Citrix ライターの初期化中にエラーが発生しました 3252
-9106 – Citrix パッケージの初期化中にエラーが発生しました 3253
-9107 – Citrix ファイル エントリの書き込み中にエラーが発生しました 3253
-9108 – ソース ファイル パスの判別中にエラーが発生しました 3254
-9109 – Citrix フォルダー エントリの書き込み中にエラーが発生しました 3254
-9110 – Citrix レジストリ エントリの書き込み中にエラーが発生しました 3254
-9113 – Citrix INI ファイル エントリの書き込み中にエラーが発生しました 3255
-9114 – Citrix ショートカットの書き込み中にエラーが発生しました 3255
-9115 – Citrix プロファイルを保存するときにエラーが発生しました 3256
-9116 – 空の Citrix プロファイルを作成するときにエラーが発生しました 3256
-9117 – 中間フォルダーの作成中にエラーが発生しました 3256
-9118 – Citrix プロファイルの初期化中にエラーが発生しました 3257
-9119 – Citrix プロファイルにデフォルト ターゲットを作成するときにエラーが発生しました 3257
-9120 – プロファイルからファイルを削除中にエラーが発生しました 3258
-9121 – ファイルを Citrix プロファイルにコピーできませんでした 3258
-9122 – ターゲットが Citrix プロファイルに存在しません 3258
-9124 – このプロファイルに作成されたショートカットがありません 3259
-9125 – Citrix ファイルの種類の関連付けの書き込み中にエラーが発生しました 3259
-9126 – 証明書を使ってプロファイルを署名できませんでした 3260
-9127 – スクリプトの実行を作成できませんでした 3260
-9128 – ショートカットが重複しています 3260
-9129 – ショートカットの名前が重複しています 3261
-9130 – ショートカットのターゲットが重複しています 3261
-9131 – インストーラーの変数を解決できませんでした 3262
-9132 – 16 色ショートカット アイコンが見つかりませんでした 3262
-9133 – ショートカット アイコンが見つかりませんでした 3262
-9134 – 実行可能ファイルからアイコンを抽出できませんでした 3263
-9135 – ショートカットのターゲットが 16 ビット 3263
-9136 – 一部のファイルが圧縮されていない可能性があります 3264
-9137 – 対象となるディレクトリが見つかりませんでした 3264
-9138 – DuplicateFile テーブルの警告 3265
-9150 警告 3265
-9151 エラー 3266
-9200 – ThinApp のインストールが必要です 3266
-9201 – ショートカット ファイルの拡張子は ".exe" である必要がある 3267
-9202
アプリケーションが作成されませんでした 3267
-9203 – ThinApp ツールがありません 3267
-9204 – ショートカットが重複しています 3268
-9205 – 同じ名前のショートカットが既に存在しますが、コマンドライン パラメーターが異なります 3268
-9206 – 同じ名前のショートカットが既に存在しますが、ターゲットが異なります 3269

-9207 - ビルド プロセス中にエラーが発生しました (vregtool.exe) 3269
-9208 - ビルド プロセス中にエラーが発生しました (vftool.exe) 3269
-9209 - ビルド プロセス中にエラーが発生しました (tlink.exe) 3270
-9300 - AdviseFile 操作で、未処理の例外が発生しました 3270
-9301 - AdviseRegistry 操作で、未処理の例外が発生しました 3271
-9302 - コマンド操作で、未処理の例外が発生しました 3271
-9303 - ファイルの変更操作で、未処理の例外が発生しました 3271
-9304 - レジストリの変更操作で、未処理の例外が発生しました 3272
-9305 - 作成操作で、未処理の例外が発生しました 3272
-9306 - 規則エンジンの実行時に、未処理の例外が発生しました 3272
-9401 - App-V ライターの初期化中に、エラー 3273
-9402 - App-V パッケージの初期化中に、エラー 3273
-9403 - App-V ファイル エントリを書き込み中にエラー 3273
-9404 - App-V フォルダー エントリを書き込み中にエラー 3274
-9405 - App-V レジストリ エントリを書き込み中にエラー 3274
-9406 - App-V INI ファイル エントリを書き込み中にエラー 3274
-9407 - App-V ショートカットを書き込み中にエラー 3275
-9408 - App-V ファイル型のデータを書き込み中にエラー 3275
-9409 - App-V データの保存中にエラーが発生しました 3275
-9410 - ソース ファイル パスの判別中にエラーが発生しました 3276
-9411 - OSD ファイルのテンプレートを抽出できませんでした 3276
-9412 - OSD ファイルを保存できませんでした 3276
-9413 - App-V OSD の本当の保存 3277
-9414 - ローカル App-V アプリケーションはプライマリ アプリケーションの依存関係として指定できない 3277
-9415 - 依存関係のアプリケーションが見つかりませんでした 3277
-9416 - プライマリ アプリケーション ディレクトリが無効です 3278
-9417 - 依存アプリケーションの OSD ファイルに、無効な HREF の値が含まれています。 3278
-9418 - サイドバイサイド アセンブリをプライベート化中にエラー 3279
-9419 - ウォーターマークを挿入中に、エラーが発生しました 3279
-9424 - App-V 5.x パッケージのビルド エラー 3279
-9500 - ショートカットが不足 3280

A

AddAdvancedFile メソッド 3305
AddComponent メソッド 3308
AddComponentSubFolder メソッド 3368
AddCondition メソッド 3396
AddCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3647, 3670
AddCustomAction メソッド 3309
AddDetectCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3708
AddDynamicFileLink メソッド (アドバンスト UI、スイート/アドバンスト UI) 3676
AddDynamicFileLinking メソッド 3369
AddEligibleCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3709
AddEnvironmentVar メソッド 3370
AddExitCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3603
AddExtensionCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3651
AddFeature メソッド 3310
AddFile メソッド 3371
AddFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3677
AddFilter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3661
AddFolder メソッド (アドバンスト UI、スイート/アドバンスト UI) 3678
AddGroup メソッド (アドバンスト UI、スイート/アドバンスト UI) 3651
AddLanguage メソッド 3312
AddLanguage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3604
AddMergeModule メソッド 3397
AddObject メソッド 3397
AddParameter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3652
AddPathVariable メソッド 3313
AddPathVariable メソッド (アドバンスト UI、スイート/アドバンスト UI) 3605
AddProductConfig メソッド 3314
AddProperty メソッド (アドバンスト UI、スイート/アドバンスト UI) 3605
AddRelease メソッド 3443
AddRemoveFile method 3372
AddSetupFile メソッド 3315
AddSetupFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3606
AddSetupType メソッド 3316
AddShellProperty メソッド 3520
AddShortcut メソッド 3410
AddStringEntry メソッド 3413
AddStringEntry メソッド (アドバンスト UI、スイート/アド

バンスト UI) 3628
AddSubFolder メソッド 3410
AddSuiteAction メソッド (アドバンスト UI、スイート/アドバンスト UI) 3607
AddSuiteActionRef メソッド (アドバンスト UI、スイート/アドバンスト UI) 3664
AddSuiteFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3608
AddSuitePackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3610, 3709
AddSuiteRelease メソッド (アドバンスト UI、スイート/アドバンスト UI) 3612
AddSuiteSubFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3671
AddSuiteTransaction メソッド (アドバンスト UI、スイート/アドバンスト UI) 3612
ALLUSERS 1915
ANSI 482
AppID 2214
Appx サポート 1542
ARPREADME 578
AttachComponent メソッド 3398
AttachPackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3671
AVI ファイル
InstallScript プロジェクトと InstallScript MSI プロジェクト
有効化 1220

B

BATCH_INSTALL
インストールまたはアンインストールの動作 1487
Bit flags 881
Build メソッド 3505
Build メソッド (アドバンスト UI、スイート/アドバンスト UI) 3733

C

C7501 警告 3228
C7502 警告 3228
C7503 警告 3229
C7505 警告 3229
C8001 エラー 3185
C8002 エラー 3186
C8003 エラー 3186
C8004 エラー 3186
C8005 エラー 3187
C8006 エラー 3187
C8007 エラー 3188
C8008 エラー 3188
C8009 エラー 3188
C8010 エラー 3189
C8011 エラー 3189
C8012 エラー 3189
C8013 エラー 3189
C8014 エラー 3190
C8015 エラー 3190
C8016 エラー 3191
C8017 エラー 3191
C8018 エラー 3191
C8019 エラー 3192
C8020 エラー 3192
C8021 エラー 3193
C8022 エラー 3193
C8023 エラー 3193
C8024 エラー 3193
C8025 error 3194
C8026 エラー 3195
C8027 エラー 3195
C8028 エラー 3195
C8031 エラー 3196
C8032 エラー 3196
C8033 エラー 3196
C8034 エラー 3197
C8035 エラー 3197
C8036 エラー 3197
C8037 エラー 3198
C8038 エラー 3198
C8039 エラー 3198
C8040 エラー 3199
C8042 エラー 3199
C8043 エラー 3199
C8044 エラー 3200
C8045 エラー 3200
C8046 エラー 3200
C8047 エラー 3201
C8048 エラー 3201
C8049 エラー 3201
C8050 エラー 3202
C8051 エラー 3202
C8052 エラー 3202
C8053 エラー 3202
C8054 エラー 3203
C8055 エラー 3203
C8057 エラー 3203
C8058 エラー 3204
C8059 エラー 3204
C8062 エラー 3204
C8063 エラー 3204
C8064 エラー 3205
C8065 エラー 3205
C8066 エラー 3205
C8067 エラー 3206
C8068 エラー 3206
C8069 エラー 3206
C8070 エラー 3207

C8071 エラー 3207
C8072 エラー 3207
C8073 エラー 3208
C8074 エラー 3208
C8075 エラー 3208
C8076 エラー 3209
C8077 エラー 3209
C8079 エラー 3210
C8080 エラー 3210
C8081 エラー 3210
C8082 エラー 3211
C8083 エラー 3211
C8084 エラー 3211
C8085 エラー 3212
C8086 エラー 3212
C8087 エラー 3212
C8088 エラー 3212
C8089 エラー 3213
C8090 エラー 3213
C8091 エラー 3213
C8092 エラー 3214
C8093 エラー 3214
C8097 エラー 3214
C8098 エラー 3215
C8099 エラー 3215
C8100 エラー 3215
C8101 エラー 3216
C8112 エラー 3216
C8113 エラー 3217
C8114 エラー 3217
C8127 3185, 3218
C8128 エラー 3218
C8522 3185
C8522 エラー 3218
C9001 エラー 3227
CanMove メソッド (アドバンスト UI、スイート/アドバンスト UI) 3658
ChangeGroupType メソッド (アドバンスト UI、スイート/アドバンスト UI) 3653
CheckForProductUpdates カスタム アクション 3761
CheckForProductUpdatesOnReboot カスタム アクション 3761
CloseProject メソッド 3322
CloseProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3613
COM オブジェクト 896
COM コンポーネント 633
  Reg-free 登録 752
  サンプル アプリケーション マニフェスト 754
  ビルド時に COM 登録を抽出する 657
  ファイル追加時に COM 登録を抽出する 621
COM サーバー 745
  登録 745
COM サーバーの登録 745
COM 抽出
  管理者権限あり/管理者権限なし 298
  レジストリ変更を除外 748
COM+ アプリケーション 2727
  アプリケーション プロキシ サポート 1266
  サーバー アプリケーション 1265
  サーバーおよびプロキシへの条件付きインストール 1268
  プロジェクトに追加する 2727
Compile.exe 3775
CreatePatch メソッド 3322
CreateProject メソッド 3323
CreateProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3614

D

DeleteAdvancedFile メソッド 3324
DeleteComponent メソッド 3325
DeleteCondition メソッド 3399
DeleteCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3654, 3672, 3711
DeleteCustomAction メソッド 3326
DeleteDynamicFileLink メソッド (アドバンスト UI、スイート/アドバンスト UI) 3679
DeleteExitCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3615
DeleteFile メソッド 3383
DeleteFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3679
DeleteFilterメソッド (アドバンスト UI、スイート/アドバンスト UI) 3662
DeleteFolder メソッド (アドバンスト UI、スイート/アドバンスト UI) 3680
DeleteLanguage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3615
DeleteMergeModule メソッド 3327
DeleteParameter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3655
DeletePathVariable メソッド 3328
DeletePathVariable メソッド (アドバンスト UI、スイート/アドバンスト UI) 3616
DeleteProductConfig メソッド 3329
DeleteProperty メソッド 3329
DeleteProperty メソッド (アドバンスト UI、スイート/アドバンスト UI) 3617
DeleteRelease メソッド 3444
DeleteSetupFile メソッド 3330
DeleteSetupFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3618
DeleteSetupType メソッド 3331
DeleteShellProperty メソッド 3521
DeleteShortcut メソッド 3411
DeleteStringEntry メソッド 3414
DeleteStringEntry メソッド (アドバンスト UI、スイート/ア

ドバンスト UI) 3629
DeleteSubFolder メソッド 3412
DeleteSuiteAction メソッド (アドバンスト UI、スイート/アドバンスト UI) 3619
DeleteSuiteActionRef メソッド (アドバンスト UI、スイート/アドバンスト UI) 3665
DeleteSuiteFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3619
DeleteSuitePackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3620
DeleteSuiteRelease メソッド (アドバンスト UI、スイート/アドバンスト UI) 3621
DeleteSuiteSubFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3672
Deployment Image Servicing and Management 1561
DeselectLanguage メソッド 3333
DetectIIS6AppPool32BitSupport 1291
DetectIIS6Interfaces 1295
DIFx 674, 903
DIM 1689, 2242
.ini ファイルを変更 1695
COM+ アプリケーションの管理 1696
DIM のインポート ウィザード 2294
IIS サービスの管理 1696
ODBC リソースの構成 1695
SQL Server を構成 1696
Windows Installer のプロパティを使用 1696
XML とバイナリ形式の違い 1690
XML ファイルを変更 1695
インストール プロジェクトに含めるための方法 755
インストール先をオーバーライド 759
インストール済みデータの検索 1696
カスタム アクションとダイアログをスケジュール 760
カスタム アクションを使用 1696
環境変数の設定 1695
基本の MSI プロジェクトで機能に関連付ける 757
基本の MSI プロジェクトで要素を識別 762
基本の MSI プロジェクトをインポート (復元不可能) 761
コンポーネントを作成 1695
ショートカット 1695
テキストファイルを変更 1695
デザイン時の競合を解決 761
デフォルトのインストール先 1690
パス変数をオーバーライド 762
パス変数を使用 1691
ビルド時の競合を解決 757
ビルドの手順を表示 759
ファイルとフォルダーを含める 1695
利点 1689
レジストリの変更 1695
基本の MSI プロジェクトで参照 756
DIM のインポート ウィザード 2294
2603
ビュー 2603
DirectX 9 オブジェクト 738
DirectX オブジェクト ウィザード 2283
DisableSharedComponent 663
Disk1 フォルダー、ファイルの追加 985
DISM.exe 1561
DLL 関数
.def ファイルを使う 923
配列を渡す 888
パラメーターを渡す 925
呼び出し 886
DllRegisterServer 750
DllUnregisterServer 750
DLLWrapCleanup カスタム アクション 3761
DLLWrapStartup カスタム アクション 3761

E

Ether オブジェクト 1379
EulaScrollWatcher.dll 1058
ExportProject メソッド 3334
ExportStrings メソッド 3334

F

F8501 エラー 3220
F8502 エラー 3221
F8503 エラー 3221
F8504 エラー 3222
F8505 エラー 3222
F8506 エラー 3223
F8508 エラー 3223
F8509 エラー 3223
F8510 エラー 3224
F8511 エラー 3224
F8512 エラー 3225
F8513 エラー 3225
F8514 エラー 3225
F8515 エラー 3225
F8516 エラー 3226
F8517 エラー 3226
F8518 エラー 3226
F8519 エラー 3227
FileBrowse カスタム アクション 1055
Filters.xml 743
COM 抽出のレジストリ変更の除外を指定 748
IIS データのインポートにおける IIS 除外を指定 1275
依存関係スキャナー除外を指定 743
Flash ファイル 1209
InstallScript プロジェクトと InstallScript MSI プロジェクト
有効化 1220
基本の MSI プロジェクトにビルボードとして追加する 1209
FlexNet Code Aware 統合 114

ForceUpgrade メソッド 3335
ForceUpgrade メソッド (アドバンスト UI、スイート/アドバンスト UI) 3621

G

GenerateGUID メソッド (アドバンスト UI、スイート/アドバンスト UI) 3622
GetLastError メソッド 1379
GUID 332

H

HKEY_USER_SELECTABLE 802
HTTP エラー、構成 1298
Hyper-V 1817

I

ICE 1421
IDE リファレンス
    IIS 仮想ディレクトリ 1268
IIS 1268
    ASP.NET 登録 1291
    ASP.NET バージョン 1290
    ASP.NET プラットフォーム 1291
    Enable32BitAppOnWin64 1291
    InstallScript テキスト置換の使用 1300
    INSTALLSHIELD_SSI_PROP 1271
    SSIEnableCmdDirective 1271
    SSL 証明書 1281
    TCP ポート番号 1278
    Web サービス拡張 1286
    Web サービス拡張のアンインストール 1289
    Web サイトのアンインストール 1288
    Web サイトの作成 1272
    Web サイトをスキャンして、その設定を InstallShield にインポートする 1273
    Windows Installer のプロパティを使用する 1298
    アプリケーション プール 1284
    アプリケーション プールのアンインストール 1289
    アプリケーションのマッピング 1294
    エラー -2147467259 1269
    エラーメッセージの構成 1298
    仮想ディレクトリのアンインストール 1288
    仮想ディレクトリの作成 1272
    機能とコンポーネントの関連付け 1287
    最初に利用可能な新しいサイト番号にインストールする 1278
    サイト番号 1278
    サポート対象バージョン 1269
    実行時の要件 1270
    タイムアウトのパラメーター 1295
    ネスト仮想ディレクトリ 1277
    ホスト ヘッダー 1281
    レガシー オブジェクト サポート 1295
XML
    実行時の要件 818
IIS 6 メタベースおよび IIS 6 構成の互換性機能
    存在を検出 1295
IIS_VERSION プロパティ 1270
IISscan.exe 3792
ImportINIFile メソッド 3375
ImportProject メソッド 3337
ImportRegFile メソッド 3376
ImportStrings メソッド 3337
ビュー 2662
 2662
InsertCustomAction メソッド 3579
INSTALLDIR 546, 560
    違い 546
InstallScript 859
    関数呼び出しのヒント 370
    コードのキーボード ショートカット 378
InstallScript MSI インストール プロジェクト
    外部または埋め込み UI ハンドラーとしての InstallScript エンジン 568
InstallScript MSI オブジェクトプロジェクト 454
InstallScript MSI プロジェクトで InstallScript エンジンを UI ハンドラーとして 568
InstallScript MSIインストールロジェクト 322
InstallScript UI における DPI 対応サポートとイメージ 1002
InstallScript インストール プロジェクト 321
InstallScript エンジン 860
    スクリプト ランタイムを検証する 3770
InstallScript で大きい数値を表現する 902
InstallScript の言語 ID 905
InstallScript と基本の MSI インストール プロジェクト 312
InstallScript の Microsoft .NET オブジェクト ウィザード 2300
InstallScript のデバッグ 864
InstallScript ビュー 2749
InstallScript または InstallScript MSI ダイアログの HTML 1013
InstallScript または InstallScript MSI ダイアログのハイパーリンク 1013
InstallScript ユーザー インターフェイス タイプの設定 574
InstallShield - Windows Installer Edition からの移行 470
InstallShield 2018 R2 の新しい機能 101
InstallShield 2019 の新しい機能 93, 95
InstallShield MSI Diff 1892
InstallShield MSI Grep 1897
InstallShield MSI Query 1895
InstallShield MSI Sleuth 1896
InstallShield MSIX/UWP アプリ適合性スイート 1460
InstallShield Professional からの移行 456
InstallShield キャビネット & ログ ファイル ビューアー 1898
InstallShield 前提条件 713
    64 ビット サポート 1636

InstallScript プロジェクトに追加 710
InstallShield 前提条件を使ってアプリケーションをアンインストール 726
InstallShield 前提条件を使ってインストールを実行 722
MySQL Connector ODBC 737
Oracle 11g Instant Client 738
Windows Installer ベースのプロジェクトに追加 708
依存関係 1649
インストール一覧に表示しない 1646
インストール順 717
インストール条件 1636
インストールのパラメーターを指定する 1641
エンド ユーザーによってスキップ可能 1646
起こりうる問題の対処方法について 1649
概要 713
管理者権限が必要 1646
機能から削除 716
機能との関連付け 715
個別の実行時の場所を設定 720
コマンドライン パラメーター 1642
コンピューターの再起動を指定する 1649
再配布可能ファイル ギャラリーに追加 704
作成および変更 1633
実行時の場所を指定 1362
進行状況バーを表示する 1648
前提条件をインストール後にターゲットマシンを再起動する 1645
前提条件を含むリリースを構成 718
代替ダウンロード URL を指定する 1635
ダウンロード URL 1640
動作 1645
必要実行レベルを指定 1358
ビルド時の場所を設定 719
ファイル 1639
プロジェクトから削除 711
プロパティ 1634
保存 1650
リリース フラグと場所の設定 721
InstallShield 前提条件エディター 2991
InstallShield 前提条件の "最小 CSD バージョン" 設定 2163
InstallShield 前提条件の "製品 (OS) タイプ" 設定 2163
InstallShield 前提条件の "プラットフォーム ID" 設定 2163
InstallShield 前提条件の "プロセッサ アーキテクチャ" 設定 2163
InstallShield テーブル 2958
ISAlias 2960
ISCOMCatalogCollection 2962
ISCOMCatalogCollectionObject 2962
ISCOMCatalogObject 2963
ISCustomActionReference 2964
ISIISItem 2965
ISIISProperty 2965
ISProductConfigurationInstance 2967, 2968
InstallShield ベスト プラクティス スイート 1470
InstallShield 仮想化整合性スイート 1444
INSTALLSHIELD_IIS_NEXT_NEW_SITE_NUMBER 1278
INSTALLSHIELD_SSI_PROP プロパティ 1271
InstanceId プロパティ 1797
Internet Explorer 前提条件 713
IS_BROWSE_FILEBROWSED 1055
IS_IIS_DO_NOT_USE_REG 1298
IS_PRINT_DIALOG プロパティ 1061
IS_PS_EXECUTIONPOLICY 933
IS_SCRIPT_TAG 470
IS_SQLSERVER_ALIAS_ONLY 1237
IS_SQLSERVER_AUTHENTICATION 1225
IS_SQLSERVER_CONNECTIONS_TO_VALIDATE 1236
IS_SQLSERVER_CXNS_ABSENT_FROM_INSTALL 1236
IS_SQLSERVER_DATABASE 1225
IS_SQLSERVER_DO_NOT_USE_REG 1237
IS_SQLSERVER_LOCAL_ONLY 1237
IS_SQLSERVER_PASSWORD 1225
IS_SQLSERVER_REMOTE_ONLY 1237
IS_SQLSERVER_SERVER 1225
IS_SQLSERVER_USERNAME 1225
IS_VM_DETECTED 1817
IS_VM_TYPE 1817
ISAlias テーブル 2960
ISBP 1470
ISBP01 1472
ISBP02 1473
ISBP03 1473
ISBP04 1474
ISBP05 1475
ISBP06 1476
ISBP07 1477
ISBP08 1477
ISBP09 1478
ISBP10 1479
ISBP11 1480
ISBP12 1480
ISBP13 1481
ISBP14 1482
ISBP15 1483
ISBP16 1483
ISBP17 1484
ISBP18 1485
ISBP19 1485
ISBP20 1486
ISBuild.exe 1333
ISChainPackageCommit カスタム アクション 3761
ISChainPackagePrepare カスタム アクション 3762
ISChainPackageRollback カスタム アクション 3762
ISClrWrap テーブル 928
ISCmdBld.exe 1326
構文 3781
コマンドラインのパラメーター 3781, 3791
ISCOMCatalogAttribute table 2961

ISCOMCatalogCollection テーブル 2962
ISCOMCatalogCollectionObject テーブル 2962
ISCOMCatalogObject テーブル 2963
ISCOMPlusApplication テーブル 2963
ISComponentServiceCosting カスタム アクション 3762
ISComponentServiceFinalize カスタム アクション 3762
ISComponentServiceInstall カスタム アクション 3762
ISComponentServiceRollback カスタム アクション 3762
ISComponentServiceUninstall カスタム アクション 3762
ISCustomActionReference テーブル 2964
ISDEBUGLOG 1911
ISDetectVM 1817
ISICE 1421
ISICE01 1424
ISICE02 1425
ISICE03 1426
ISICE04 1426
ISICE05 1427
ISICE06 1428
ISICE07 1429
ISICE08 1430
ISICE09 1431
ISICE10 1431
ISICE11 1432
ISICE12 1434
ISICE16 1434
ISICE17 1435
ISICE18 1436
ISICE19 1437
ISICE20 1439
ISICE21 1440
ISICE22 1441
ISICE23 1441
ISICE24 1442
ISICE25 1443
ISICE26 1443
iSign.exe 1369
　コマンドラインのパラメーター 3793
ISIIS6APPPOOLSUPPORTS32BIT 1291
ISIISCleanup カスタム アクション 3762
ISIISCosting カスタム アクション 3762
ISIISInstall カスタム アクション 3763
ISIISItem テーブル 2965
ISIISMETABASECOMPATPRESENT 1295
ISIISProperty テーブル 2965
ISIISRollback カスタム アクション 3763
ISIISUninstall カスタム アクション 3763
ISInstallPrerequisites 3763
ISJITCompileActionAtInstall カスタム アクション 3763
ISJITCompileActionAtUnInstall カスタム アクション 3763
ISLockPermissionsCost カスタム アクション 3763
ISLockPermissionsInstall カスタム アクション 3763
ISMSI 557, 2242
ISNetApiInstall カスタム アクション 3764
ISNetApiLogonGroup 853
ISNetApiLogonPassword 853
ISNetApiLogonUsername 853
ISNetApiRollback カスタム アクション 3764
ISNetCreateIniForOneUser カスタム アクション 3764
ISNetDeleteIniFile カスタム アクション 3764
ISNetGetGroups カスタム アクション 3764
ISNetGetServers カスタム アクション 3764
ISNetGetUsers カスタム アクション 3764
ISNetSetLogonName カスタム アクション 3764
ISNetValidateLogonName カスタム アクション 3765
ISNetValidateNewUserInformation カスタム アクション 3765
ISNetworkSharesCosting カスタム アクション 3765
ISNetworkSharesFinalize カスタム アクション 3765
ISNetworkSharesInstall カスタム アクション 3765
ISNetworkSharesRollback カスタム アクション 3765
ISNetworkSharesUninstall カスタム アクション 3765
ISO/IEC 19770-2 579
ISOBJ″ /> 557
IsPlaying メソッド 1381
ISPrint カスタム アクション 1061, 3766
ISProductConfigName 変数 1336
ISProductConfigurationInstance テーブル 2967, 2968
ISQuickPatchFinalize カスタム アクション 3766
ISQuickPatchFixShortcut カスタム アクション 3766
ISQuickPatchHelper カスタム アクション 3766
ISQuickPatchInit カスタム アクション 3766
ISQuickPatchInit9X カスタム アクション 3766
ISQuickPatchInit9X2 カスタム アクション 3766
ISReleaseFlags プロパティ 918
ISReleaseName 変数 1336
ISReleasePath 変数 1336
ISReleaseUsesShallowFolderPaths 変数 1336
ISRunSetupTypeAddLocalEvent カスタム アクション 3766
ISSCRIPT_ENGINE_VERSION プロパティ 1909
ISSCRIPT_VERSION_MISSING プロパティ 1909
ISSCRIPT_VERSION_OLD プロパティ 1909
ISSearchReplaceCosting カスタム アクション 3767
ISSearchReplaceFinalize カスタム アクション 3767
ISSearchReplaceInstall カスタム アクション 3767
ISSearchReplaceRollback カスタム アクション 3767
ISSearchReplaceUninstall カスタム アクション 3767
ISSelfReg 752
ISSelfReg テーブル 752
ISSelfRegisterCosting アクション 752
ISSelfRegisterCosting カスタム アクション 3767
ISSelfRegisterFiles アクション 752
ISSelfRegisterFiles カスタム アクション 3767
ISSelfRegisterFinalize アクション 752
ISSelfRegisterFinalize カスタム アクション 3767
ISSetAllUsers 982
ISSetAllUsers カスタム アクション 3768
ISSetTARGETDIR カスタム アクション 3768
ISSetup.dll 860

ISSetupFilesCleanup カスタム アクション 3768
ISSetupFilesExtract カスタム アクション 3768
ISSQLQueryDatabases カスタム アクション 3768
ISSQLServerCosting カスタム アクション 3768
ISSQLServerFilteredList カスタム アクション 3768
ISSQLServerFinalize カスタム アクション 3768
ISSQLServerInitialize カスタム アクション 3769
ISSQLServerInstall カスタム アクション 3769
ISSQLServerList カスタム アクション 3769
ISSQLServerRemoveLoginInfo カスタム アクション 3769
ISSQLServerRollback カスタム アクション 3769
ISSQLServerRollbackLoginInfo カスタム アクション 3769
ISSQLServerUninstall カスタム アクション 3769
ISSQLServerValidate カスタム アクション 3769
ISSQLServerWriteLoginInfo カスタム アクション 3770
ISSQLSRV.dll 1227
ISSupportPerUser 854
ISUnSelfRegisterFiles アクション 752
ISUnSelfRegisterFiles カスタム アクション 3770
ISUWP01 1462
ISUWP02 1462
ISUWP03 1463
ISUWP04 1463
ISUWP05 1463
ISUWP06 1464
ISUWP07 1464
ISUWP08 1465
ISUWP09 1465
ISUWP10 1466
ISUWP11 1466
ISUWP12 1467
ISUWP13 1467
ISUWP14 1468
ISUWP15 1468
ISUWP16 1468
ISUWP17 1469
ISUWP18 1469
ISUWP19 1470
ISUWP20 1470
ISVerifyScriptingRuntime カスタム アクション 3770
ISVICE 1444
ISVICE01 1447
ISVICE02 1447
ISVICE03 1448
ISVICE04 1449
ISVICE05 1450
ISVICE06 1450
ISVICE07 1451
ISVICE08 1452
ISVICE09 1452
ISVICE10 1453
ISVICE11 1454
ISVICE12 1454
ISVICE13 1455
ISVICE14 1456
ISVICE15 1457
ISVICE16 1457
ISVICE17 1458
ISVICE18 1459
ISVICE19 1459
ISWiAdvancedFile オブジェクト 3342
ISWiAdvancedFiles コレクション 3544
IswiAuto 3288
IsWiComponent オブジェクト 3350
ISWiComponents コレクション 3548
ISWiComponentSubFolder オブジェクト 3382
ISWiCustomAction オブジェクト 3349
ISWiCustomActions コレクション 3552
ISWiDynamicFileLinkings コレクション 3553
ISWiEnvironmentVars コレクション 3554
ISWiFeature オブジェクト 3389
ISWiFeatures コレクション 3554
ISWiFiles コレクション 3556
ISWiFolder オブジェクト 3407
ISWiFolders コレクション 3557
ISWiLanguage オブジェクト 3412, 3558
ISWiLanguage オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3627
ISWiLanguages コレクション (アドバンスト UI、スイート/アドバンスト UI) 3735
ISWiLatestSetup オブジェクト 3415
ISWiMsiPatchSequence オブジェクト 3418
ISWiObject オブジェクト 3420
ISWiObjects コレクション 3564
ISWiPatchConfig オブジェクト 3420
ISWiPatchExternalFile オブジェクト 3433
ISWiPatchWholeFile オブジェクト 3434
ISWiPathVariable オブジェクト 3435
ISWiPathVariable オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3630
ISWiPathVariables オブジェクト 3569
ISWiPathVariables コレクション (アドバンスト UI、スイート/アドバンスト UI) 3736
ISWiProductConfig オブジェクト 3438
ISWiProductConfigs コレクション 3572
ISWiProject object
AddProperty メソッド 3314
GenerateGUID メソッド 3336
ISWiProject オブジェクト 3288
ISWiProject オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3595
ISWiProperties コレクション 3574
ISWiProperties コレクション (アドバンスト UI、スイート/アドバンスト UI) 3736
ISWiProperty オブジェクト 3444
ISWiProperty オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3633
ISWiRelease オブジェクト 3445

ISWiReleases コレクション 3575
ISWiRemoveFileObject オブジェクト 3508
ISWiRemoveFiles コレクション 3576
ISWiRemoveFiles メソッド 3377
ISWiSequence コレクション 3577
ISWiSequenceRecord オブジェクト 3509
ISWiSetupFile オブジェクト 3510
ISWiSetupFile オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3634
ISWiSetupFiles コレクション 3580
ISWiSetupFiles コレクション (アドバンスト UI、スイート/アドバンスト UI) 3737
ISWiSetupType オブジェクト 3512
ISWiSetupTypes コレクション 3581
ISWiShellProperties コレクション 3582
ISWiShellProperty オブジェクト 3513
ISWiShortcut オブジェクト 3513
ISWiShortcuts コレクション 3583
ISWiSISProperties コレクション 3584
ISWiSISProperty オブジェクト 3522
ISWiStringEntries コレクション 3591
ISWiStringEntry オブジェクト 3541
ISWiSuiteAction オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3635
ISWiSuiteActionRef オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3643
ISWiSuiteActionRefs コレクション (アドバンスト UI、スイート/アドバンスト UI) 3738
ISWiSuiteActions コレクション (アドバンスト UI、スイート/アドバンスト UI) 3738
ISWiSuiteCondition オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3645
ISWiSuiteConditionParameter オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3656
ISWiSuiteConditionParameters コレクション (アドバンスト UI、スイート/アドバンスト UI) 3739
ISWiSuiteConditions コレクション (アドバンスト UI、スイート/アドバンスト UI) 3740
ISWiSuiteDFLFilter オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3657
ISWiSuiteDFLFilters コレクション (アドバンスト UI、スイート/アドバンスト UI) 3740
ISWiSuiteDynamicFileLink オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3660
ISWiSuiteDynamicFileLinks コレクション (アドバンスト UI、スイート/アドバンスト UI) 3741
ISWiSuiteEvent オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3663
ISWiSuiteEvent メソッド (アドバンスト UI、スイート/アドバンスト UI) 3622
ISWiSuiteEvents コレクション (アドバンスト UI、スイート/アドバンスト UI) 3742
ISWiSuiteExitCondition オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3666
ISWiSuiteExitConditions コレクション (アドバンスト UI、スイート/アドバンスト UI) 3743
ISWiSuiteFeature オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3666
ISWiSuiteFeatures コレクション (アドバンスト UI、スイート/アドバンスト UI) 3744
ISWiSuiteFile オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3674
ISWiSuiteFolder オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3675
ISWiSuiteFolders コレクション (アドバンスト UI、スイート/アドバンスト UI) 3746
ISWiSuiteOperation オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3681
ISWiSuitePackage オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3686
ISWiSuitePackages コレクション (アドバンスト UI、スイート/アドバンスト UI) 3747
ISWiSuiteRelease オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3713
ISWiSuiteReleases コレクション (アドバンスト UI、スイート/アドバンスト UI) 3748
ISXmlAppSearch カスタム アクション 3770
ISXmlCosting カスタム アクション 3770
ISXmlFinalize カスタム アクション 3770
ISXmlInstall カスタム アクション 3770
ISXmlRollback カスタム アクション 3771
ISXmlUnInstall カスタム アクション 3771
Itanium
および .NET Framework 731, 734

J

Java(TM) 2 Runtime Environment 前提条件 713
Jet 4.0 前提条件 713
JRE 前提条件 713

L

LaunchCondition テーブル 343
LaunchProgramFileFromSetupCompleteSuccess カスタム アクション 3771
LaunchReadmeFileFromSetupCompleteSuccesscustom アクション 3771
Line control 1109
LocalDB 1229

M

MDAC 2.8 前提条件 713
Microsoft .NET 8 Desktop Runtime 8.0.5, 32 ビット 736
Microsoft .NET 8 Desktop Runtime 8.0.5, 64 ビット 736
Microsoft .NET Core 736, 928

Microsoft .NET Framework 733
バージョン 2.0、64 ビット 731, 734
バージョン 3.0 734
バージョン 3.0、64 ビット 731, 734
Microsoft Visual Studio、InstallShield との統合 1960
Microsoft Windows Azure SQL サポート 1250
Microsoft XML Core Services 818
MM 2242
Move メソッド (アドバンスト UI、スイート/アドバンスト UI) 3656
MoveDown メソッド (アドバンスト UI、スイート/アドバンスト UI) 3659, 3712, 3713
MoveTo メソッド (アドバンスト UI、スイート/アドバンスト UI) 3644
MoveUp メソッド (アドバンスト UI、スイート/アドバンスト UI) 3660
MSBuild 1347
MSDE 2000 前提条件 713
MSI Diff 1887, 1892
MSI Grep 1897
MSI Query 1895
MSI Sleuth 1896
MSI 再配布可能ファイル 730
2951
ビュー 2951
MSIDB 557, 2242
msidbComponentAttributesShared 663
MsiExec.exe 3796
MsiPatchOldAssemblyFile 1768
MsiPatchOldAssemblyName 1768
MsiRMFilesInUse ダイアログ 1063
MSIX/UWP アプリ サポート
InstallShield MSIX/UWP アプリ適合性スイート 1460
MSMDB 2242
MSP オープン ウィザード 2309
MST ウィザード 2310
MSXML 818

O

2659
ビュー 2659
OnBegin
スイート/アドバンスト UI インストール 1585
One-Click Install インストール 1374
Ether オブジェクト 1379
InstallShield 12 以前からアップグレードする 433
InstallShield とInstallShield Professional の違い 1375
Player オブジェクト 1377
起動されたインストールにデータを渡す 1387
言語の設定 1387
サイレント 1387
作成 1386
システム要件 1385
デバッグ 1388
ユーザーを認証 1388
OnEnd
スイート/アドバンスト UI インストール 1585
OnPackagesConfigured 1585
OnPackagesConfiguring 1585
OnRebooting
スイート/アドバンスト UI インストール 1585
OnResuming
スイート/アドバンスト UI インストール 1585
OnStaged 1585
OnStaging 1585
Open メソッド 1378
OpenProject メソッド 3339
OpenProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3625
Oracle 11g Instant Client 前提条件 1250
Oracle サポート 1250
64 ビット ターゲット システム 1250

P

Pkgmgr.exe 1561
Play メソッド 1381
PowerShell スクリプト 933
POWERSHELLVERSION 933
ProgID 666
PVK2PFX 1366

Q

QP 2242
QuickPatch
InstallShield で作成 1744
QuickPatch 作成の簡素化 1744
簡素化の制限事項 1744
ビルトイン InstallShield カスタムアクション 1744
QuickPatch パッケージの簡素化 1744
QuickPatch プロジェクト 325
カスタム アクション 2985
新規 QuickPatch 作成ウィザード 2238
手順 1743
ファイル 2985
履歴 2985
レジストリ 2988
QuickPatch プロジェクトの履歴 2985

R

REG ファイル 797
複数行文字列サポート 2107
ReleasePackager.exe 3794
RemoveComponent メソッド 3399

RemoveComponentSubFolder メソッド 3378
RemoveDynamicFileLinking メソッド 3379
RemoveEnvironmentVar メソッド 3380
RemoveFeature メソッド 3341
RemoveFile メソッド 997
RemoveMergeModule メソッド 3400
RemoveObject メソッド 3401
RemovePackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3673
RemoveRemoveFile メソッド 3381
RemoveSequenceRecord メソッド 3579
Rollback 976
RTL 言語サポート 482

S

SaveProject メソッド 3342
SaveProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3626
Sd ダイアログ スタティック テキスト フィールド 1012
setAllUsersProfile2K カスタム アクション 3771
SetAllUsersProfileNT カスタム アクション 3771
SetARPINSTALLLOCATION カスタム アクション 3771
SetARPReadme (カスタム アクション) 578
SetProperty メソッド 1382
Settings.xml 379
Setup_UI.xml 1525
Setup.exe 1495
  InstallScript プロジェクト 3826
  InstallShield 前提条件を含める 721
  Windows Installer の再配布 730
  アイコンを指定する 1504
  コマンドライン パラメーター (スイート/アドバンスト UI およびアドバンスト UI) 3821
  コマンドライン パラメーター (基本の MSI、InstallScript、InstallScript MSI) 3801
  実行時のエラー (InstallScript) 3125
  実行時のエラー (基本の MSI および InstallScript MSI) 3128
  実行時のエラー (スイート/アドバンスト UI およびアドバンスト UI) 3133
  プロパティのカスタマイズ 1497
  戻り値 (InstallScript) 3125
  戻り値 (基本の MSI および InstallScript MSI) 3128
  戻り値 (スイート/アドバンスト UI およびアドバンスト UI) 3133
  ログ、基本の MSI 3813
  ログ、スイート/アドバンスト UI およびアドバンスト UI インストール 3822
  ログ、スイートのパッケージ 3823
Setup.exe の著作権情報 1497
Setup.ilg 1902
  .txt ファイルに変換する 1904
  概要 1902
  参照 1902
  開く 1903
Setup.ini 1389
Setup.log 1413
Setup.rul 861
Setup.xml 1525
SETUPEXEDIR 1934
SetupIni.exe 3795
SetupUI.dll 1525
setUserProfileNT カスタム アクション 3771
ShowMsiLog カスタム アクション 3771
SOURCELIST プロパティ
  値の順番を変更する 1523
  値を削除 1523
  値を追加 1522
  値を編集 1522
SQL 2731
  64 ビット ターゲット システムの Oracle サポート 1250
  InstallScript テキスト置換を使って、スクリプトを変更 1244
  InstallScript と InstallScript MSI プロジェクトの要件 1227
  LocalDB 1229
  SQL サーバー側インストールの要求 1246
  SQL スクリプトで、Oracle スキーマを作成 1261
  SQL スクリプトで、SQL Server データベースを作成 1258
  SQL スクリプトにデータベース サーバー タイプの条件を設定 1238
  SQL スクリプトの起動順序を指定 1235
  SQL スクリプトの追加 1230
  SQL スクリプトの編集 1233
  Unicode と Oracle データベースにおける問題の解決 1257
  Windows Azure SQL サポート 1250
  Windows Installer プロパティを使ってスクリプトを変更 1242
  オートコンプリート 368
  オートメーション
    AddSQLRequirement メソッド 3526
    AddSQLScript メソッド 3527
    AddSQLScriptEx メソッド 3527
    AddSQLScriptReplacement メソッド 3539
    AddSQLServerConnection メソッド 3317
    DeleteSQLConnection メソッド 3332
    DeleteSQLReplace メソッド 3540
    DeleteSQLScript メソッド 3528
    DeleteSQLScriptError メソッド 3540
    GetDatabaseServer メソッド 3529
    InsertSQLScript メソッド 3529
    ISWiSQLConnection オブジェクト 3523
    ISWiSQLConnections コレクション 3585
    ISWiSQLDatabaseServer オブジェクト 3530
    ISWiSQLDatabaseServers コレクション 3586
    ISWiSQLReplace オブジェクト 3532
    ISWiSQLReplaces コレクション 3587

ISWiSQLRequirement オブジェクト 3533
ISWiSQLRequirements コレクション 3588
ISWiSQLScript オブジェクト 3536
ISWiSQLScripts コレクション 3590
RemoveSQLRequirement メソッド 3530
SetPredefinedRequirement メソッド 3534
行番号の表示/非表示 374
構文ハイライト 372
自動字下げ 375
スクリプト エディタを使った作業 367
スクリプトのブックマーク 368
接続と Windows Installer のプロパティ 1226
接続を追加 1228
データベースのインポート 2268
デフォルト プロトコルを上書き 1228
デフォルトのランタイム動作をオーバーライド 1236
入力用のキーボード ショートカット 378
SQL Server Express Edition 前提条件 713
SQL Server Native Client の前提条件 1250
SQLCONV.obl 1227
SQLRT.dll 1227
SQLRT.ini 1227
SQLRT.obl 1227
SSL 証明書 1281
Standalone Build 1341
STANDARD_USE_SETUPEXE プロパティ 1909
subweb、エラー メッセージの構成 1298
Sysnative フォルダー 598
System.AppUserModel.PreventPinning 776

T

TARGETDIR 546
違い 546
Team Explorer 1968
Team Foundation Server の統合 1968
TFS の統合 1968
TR″ /> 557, 2242

U

UAC とインストール 540
Unicode 482
Update.exe 3801
アイコンを指定する 1728
プロパティのカスタマイズ 1727
Update.exe の著作権 1727
UWP アプリ サポート
UWP アプリ パッケージの作成 1393
UWP アプリ パッケージの配布 1512

V

ValidationFiles フォルダー 1420
VB .NET 1960
Virtual PC 1817
Visual Basic .NET および Visual C# .NET 用のVisual Studio .NET ウィザード 2379
Visual Studio Team Foundation Server
InstallShield と統合 1968
VMConfigurations.xml 394
VMware 1817
voicewarmup 2094
VSSolutionFolder 1962

W

WatchScroll カスタム アクション 1058
Web インストール 2324
Web サービス 1968
ターゲット マシン上で配置 1301
Web サービス拡張 1286
Web サイト
エラーメッセージの構成 1298
ファイルを追加 1283
フォーム認証 1302
Web サイト、アプリケーション、または仮想ディレクトリ のカスタム エラー メッセージ 1298
Web サイト、アプリケーション、または仮想ディレクトリ の 404 エラー メッセージ 1298
Web 配置サポート 1541
Windows API 関数 899
Windows Azure SQL サポート 1250
Windows Installer 再配布可能ファイル
バージョン 4.5 731
Windows Installer テーブル 2958
Windows Installer の再配布可能ファイル 730
Windows Installer のプロパティ 1907
遅延/コミット/ロールバック カスタム アクション 920
Windows Installer プロパティ
完全なリスト 1909
パブリック、プライベート、制限付き、および必須 1908
プロパティ マネージャを使用 2779
文字列エントリと関連付ける 1945, 1946
Windows Installer をインストール 730
Windows Management Instrumentation 前提条件 713
Windows Server Core 1320
Windows サービス
インストール、開始、停止、削除、アンインストール、および構成 851
Windows の役割と機能 1561
有効化 1561
Windows ロゴ プログラム 539
WiseScriptFinalize カスタム アクション 3772

WiseScriptInitialize カスタム アクション 3772
WiseScriptRollback カスタム アクション 3772
WMI 前提条件 713
WOW64 1320

X

XML 2668
    <、>、&、' 、および " を要素のコンテンツとして使用する 833
    InstallScript テキスト置換の使用 832
    Windows Installer のプロパティを使用 826, 830
    XML 設定のインポート ウィザード 2297
    XML データの検索 1876
    アンインストール時の変更のテスト 839
    インストール時の変更のテスト 838
    機能に関連付け 821
    名前空間の使用 834
XML (.ismを XML またはバイナリ形式で保存する) 554

あ

アーキテクチャの検証 1320
アイコン
    Setup.exe を指定する 1504
    Update.exe を指定する 1728
    ショートカット 769
アイコン コントロール 1106
アクション 2751
アクション テキスト 2751
    指定する 949
    設定 2773, 2783
アセンブリ 1768
    グローバルアセンブリキャッシュのパッチ 1768
 2816
アップグレード 1769
    InstallScript MSI オブジェクト プロジェクト 454
    ダウングレードの防止 547
    プロジェクト 2816
ビュー 2816
アップグレードの警告 3170
アップデート 481, 1769
    InstallShield 481
 967
アドバタイズ 688
    機能 688
    コンポーネントのパブリッシュ 675
    ショートカット 2627
シーケンス 967
アドバンスト UI インストール 317
    .appx パッケージを追加する 1542
    .exe パッケージを追加する 1539, 1540
    .msi、.msp、およびトランザクションをパッケージに追加する 1534
    .prq ファイルをインポート 1550
    InstallScript イベント、関数、および変数 1536, 1537
    InstallScript パッケージを追加する 1536, 1537
    InstallShield 前提条件をインポート 1550
    UAC プロンプト 1603
    Web 配置パッケージを追加する 1541
    依存関係パッケージ 1552
    オブジェクト式 1602, 3749
    カスタム パッケージ フォルダー名 1560
    機能とパッケージの関連付け 1554
    機能の状態 1598
    共有パッケージ 1555
    形式化された式 1600
    システム検索 1602
    自動アップデート 1609
    スイート/アドバンスト UI インストールとの違い 1527
    ダイナミック ファイル リンク 1546
    ダウンロード可能なアップデート 1609
    トラブル シューティング 1618
    パスワード保護 1617
    パッケージにコマンドライン パラメーターを渡す 1591
    パッケージのインストール順 1553
    パッケージの操作 1558
    パッケージのランタイムの場所 1557
    パッケージのログ記録 1623
    パッケージを追加するときのガイドライン 1532
    含まれているファイル 1525
    プライマリ パッケージ 1552
    プログラムの追加と削除エントリ 1590
    プロパティ 1935
    モード条件 1589
    予期しないメンテナンス モード 1618
    ログ記録 1618
アドバンスト UI およびスイート/アドバンスト UI インストールのオブジェクト式 3749
    Feature オブジェクト 3750
    File オブジェクト 3752
    Parcel オブジェクト 3753
    Platform オブジェクト 3755
    Registry オブジェクト 3757
アップグレード
    パッチとの違い 1712
 2606
ビュー 2606
アプリケーション パス キー 673
アプリケーション プール 1284
アプリケーション ライフサイクル 309
アプリケーションのライフサイクル 309
アンインストール 726
    アンインストール用にログされた InstallScript 関数 1308
    アンインストール時にのみスクリプト コードを実行する 1307
    前提条件を持つアプリケーション 726
    パッチ 1732

メンテナンスおよびアンインストールのためのインストールを作成する 1305
アンインストール ショートカット
InstallScript または InstallScript MSI プロジェクトに作成 782
基本の MSI プロジェクトに作成 781

い

移行コンポーネント 652
依存関係
64 ビット 742
スキャン結果の確認 742
スタティック スキャン 740
ダイナミックスキャン 741
依存関係スキャナー
ファイルのフィルター 743
ビュー 2390, 2435
2782
ビュー 2782
ボタン 1061
954
インストール 308, 537, 583
作成 537
定義 308
編成 583
シーケンス 954
インストール プロジェクト 313
1 つのプロジェクトから複数の製品設定のインストールを作成する 1374
オートメーション インターフェイスの使用 3288
作業を行う 312
作成 326
名前または場所の変更 329
開く 327
保存 329
ビュー 2389
インストールの実行 492, 1396
インストールを実行
InstallShield 前提条件を使う 722
複数回実行する 566
複数の .msi ファイルを同時に実行 713
インストールを連鎖させる 713
2389
インターネット インフォーメーション サービス
エラー -2147467259 1269
インターネット インフォメーション サービス 1268
ASP .NET 登録 1291
ASP .NET バージョン 1290
Enable32BitAppOnWin64 1291
IASP .NET プラットフォーム 1291
INSTALLSHIELD_SSI_PROP 1271
SSIEnableCmdDirective 1271
SSL 証明書 1281
TCP ポート番号 1278
Web サービス拡張のアンインストール 1289
Web サイトのアンインストール 1288
Web サイトの作成 1272
Web サイトをスキャンして、その設定を InstallShield にインポートする 1273
アプリケーション プール 1284
アプリケーション プールのアンインストール 1289
アプリケーションの作成 1272
アプリケーションのマッピング 1294
エラー メッセージの構成 1298
仮想ディレクトリのアンインストール 1288
仮想ディレクトリの作成 1272
機能とコンポーネントの関連付け 1287
最初に利用可能な新しいサイト番号にインストールする 1278
サイト番号 1278
サポート対象バージョン 1269
実行時の要件 1270
タイムアウトのパラメーター 1295
ネスト仮想ディレクトリ 1277
ホスト ヘッダー 1281
レガシー オブジェクト サポート 1295
インターネット配布 1508
プロキシ サーバー サポート 1508
インポート
Visual Studio プロジェクトを InstallShield プロジェクトにインポート 477

う

ウィザード
コントロールのアクション 1189
コントロールの検証 1186
ウィザード インターフェイスにおけるアンパサンド 1179
ウィザード ページ 1160
SQLLogin 定義済みウィザード ページ 1263
新しいブランク ウィザード ページを作成 1174
アンパサンドをテキストで表示する 1179
キーボード ショートカット 1179
コントロールに追加 1178
順序を構成する 1177
タブの順番 1179
定義済み ウィザード ページを追加 1175
ナビゲーション ボタン 1181
背景 1167
開く/閉じるときにアクションをスケジュール 1189
編集 1177
"上書きしない" コンポーネントプロパティ 655

え

エクスポート 911
コンポーネントのエクスポート 2290

エラー
DIFx 3178
Setup.exe 実行時 (基本の MSI および InstallScript MSI) 3128
Setup.exe 実行時 (InstallScript) 3125
Setup.exe ランタイム (スイート/アドバンスト UI およびアドバンスト UI) 3133
ビルド 3006
メディア ビルド 3103
エラー、Web サイト、アプリケーション、または仮想ディレクトリの構成 1298
エンド ユーアー ダイアログ
ダイアログ エディターのレイアウト編集 1021
エンド ユーザー ダイアログ
インポート 993
エクスポート 993
コンボ ボックスのプロパティ 1080
削除 997, 1020
実行時に表示 1019
ダイアログのプロパティ 1085
チェック ボックスのプロパティ 1075
テーマ 1036, 1054
テキスト領域のプロパティ 1149
デフォルトの応答のカスタマイズ 1520
動作 1024
プッシュ ボタンのプロパティ 1130
編集フィールドのプロパティ 1094
ローカライズ 1787
エンド ユーザー使用許諾契約
ユーザーが読むのを必須にする 1058
エンドユーザー ダイアログ 2794

お

オートメーション インターフェイス 3287
64 ビット マシン 1981
AddAdvancedFile メソッド 3305
AddComponent メソッド 3308
AddComponentSubFolder メソッド 3368
AddCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3647, 3670
AddCustomAction メソッド 3309
AddDetectConditionメソッド (アドバンスト UI、スイート/アドバンスト UI) 3708
AddDynamicFileLink メソッド (アドバンスト UI、スイート/アドバンスト UI) 3676
AddDynamicFileLinking メソッド 3369
AddEligibleCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3709
AddEnvironmentVar メソッド 3370
AddExitCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3603
AddExtensionCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3651
AddFeature メソッド 3310
AddFile メソッド 3371
AddFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3677
AddFilter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3661
AddFolder メソッド (アドバンスト UI、スイート/アドバンスト UI) 3678
AddGroup メソッド (アドバンスト UI、スイート/アドバンスト UI) 3651
AddLanguage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3604
AddMergeModule メソッド 3397
AddObject メソッド 3397
AddParameter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3652
AddPathVariable メソッド 3313
AddPathVariable メソッド (アドバンスト UI、スイート/アドバンスト UI) 3605
AddProductConfig メソッド 3314
AddProperty メソッド (アドバンスト UI、スイート/アドバンスト UI) 3605
AddRelease メソッド 3443
AddRemoveFile メソッド 3372
AddSetupFile メソッド 3315
AddSetupFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3606
AddSetupType メソッド 3316
AddShellProperty メソッド 3520
AddShortcutメソッド 3410
AddStringEntry メソッド (アドバンスト UI、スイート/アドバンスト UI) 3628
AddStringEntryメソッド 3413
AddSubFolder メソッド 3410
AddSuiteAction メソッド (アドバンスト UI、スイート/アドバンスト UI) 3607
AddSuiteActionRef メソッド (アドバンスト UI、スイート/アドバンスト UI) 3664
AddSuiteFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3608
AddSuitePackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3610, 3709
AddSuiteRelease メソッド (アドバンスト UI、スイート/アドバンスト UI) 3612
AddSuiteSubFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3671
AddSuiteTransaction メソッド (アドバンスト UI、スイート/アドバンスト UI) 3612
AddUpgradeTableEntry メソッド 3318
AttachComponent メソッド 3398
AttachPackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3671
Build メソッド 3505
Build メソッド (アドバンスト UI、スイート/アドバンス

ト UI) 3733
BuildPatchConfiguration メソッド 3319
BuildPCPFileメソッド 3321
CanMove メソッド (アドバンスト UI、スイート/アドバンスト UI) 3658
ChangeGroupType メソッド (アドバンスト UI、スイート/アドバンスト UI) 3653
CloseProject メソッド 3322
CloseProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3613
CreatePatch メソッド 3322
CreateProject メソッド 3323
CreateProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3614
Delete メソッド 3543
DeleteAdvancedFile メソッド 3324
DeleteCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3654, 3672, 3711
DeleteCustomAction メソッド 3326
DeleteDynamicFileLink メソッド (アドバンスト UI、スイート/アドバンスト UI) 3679
DeleteExitCondition メソッド (アドバンスト UI、スイート/アドバンスト UI) 3615
DeleteFile メソッド 3383
DeleteFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3679
DeleteFilter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3662
DeleteFolder メソッド (アドバンスト UI、スイート/アドバンスト UI) 3680
DeleteLanguage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3615
DeleteMergeModule メソッド 3327
DeleteParameter メソッド (アドバンスト UI、スイート/アドバンスト UI) 3655
DeletePathVariable メソッド (アドバンスト UI、スイート/アドバンスト UI) 3616
DeleteProperty メソッド (アドバンスト UI、スイート/アドバンスト UI) 3617
DeleteReleaseメソッド 3444
DeleteSetupFile メソッド 3330
DeleteSetupFile メソッド (アドバンスト UI、スイート/アドバンスト UI) 3618
DeleteSetupType メソッド 3331
DeleteShellProperty メソッド 3521
DeleteShortcut メソッド 3411
DeleteStringEntry メソッド 3414
DeleteStringEntry メソッド (アドバンスト UI、スイート/アドバンスト UI) 3629
DeleteSubFolder メソッド 3412
DeleteSuiteAction メソッド (アドバンスト UI、スイート/アドバンスト UI) 3619
DeleteSuiteActionRef メソッド (アドバンスト UI、スイート/アドバンスト UI) 3665
DeleteSuiteFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3619
DeleteSuitePackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3620
DeleteSuiteRelease メソッド (アドバンスト UI、スイート/アドバンスト UI) 3621
DeleteSuiteSubFeature メソッド (アドバンスト UI、スイート/アドバンスト UI) 3672
ExportProject メソッド 3334
ExportStrings メソッド 3334
ForceUpgrade メソッド 3335
ForceUpgrade メソッド (アドバンスト UI、スイート/アドバンスト UI) 3621
GenerateGUID メソッド (アドバンスト UI、スイート/アドバンスト UI) 3622
ImportProject メソッド 3337
ImportRegFile メソッド 3376
ImportStrings メソッド 3337
InsertCustomAction メソッド 3579
ISWiAdvancedFile オブジェクト 3342
ISWiAdvancedFiles コレクション 3544
ISWiComponent オブジェクト 3350
ISWiComponents コレクション 3548
ISWiComponentSubFolder オブジェクト 3382
ISWiComponentSubFolders コレクション 3550
ISWiCondition オブジェクト 3384
ISWiConditions コレクション 3551
ISWiCustomAction オブジェクト 3349
ISWiCustomActions コレクション 3552
ISWiDynamicFileLinking オブジェクト 3384
ISWiEnvironmentVar オブジェクト 3387
ISWiFeature オブジェクト 3389
ISWiFeatures コレクション 3554
ISWiFile オブジェクト 3401
ISWiFolder オブジェクト 3407
ISWiFolders コレクション 3557
ISWiLanguage オブジェクト 3412, 3558
ISWiLanguage オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3627
ISWiLanguages コレクション (アドバンスト UI、スイート/アドバンスト UI) 3735
ISWiObject オブジェクト 3420
ISWiObjects コレクション 3564
ISWiPathVariable オブジェクト 3435
ISWiPathVariable オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3630
ISWiPathVariables コレクション (アドバンスト UI、スイート/アドバンスト UI) 3736
ISWiProductConfig オブジェクト 3438
ISWiProductConfigs コレクション 3572
ISWiProject オブジェクト 3288
ISWiProject オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3595
ISWiProperties コレクション 3574

ISWiProperties コレクション (アドバンスト UI、スイート/アドバンスト UI) 3736
ISWiProperty オブジェクト 3444
ISWiProperty オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3633
ISWiRelease オブジェクト 3445
ISWiReleases コレクション 3575
ISWiRemoveFileObject オブジェクト 3508
ISWiRemoveFiles コレクション 3576
ISWiRemoveFiles メソッド 3377
ISWiSequence コレクション 3577
ISWiSequenceRecord オブジェクト 3509
ISWiSetupFile オブジェクト 3510
ISWiSetupFile オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3634
ISWiSetupFiles コレクション 3580
ISWiSetupFiles コレクション (アドバンスト UI、スイート/アドバンスト UI) 3737
ISWiSetupType オブジェクト 3512
ISWiSetupTypes コレクション 3581
ISWiShellProperties コレクション 3582
ISWiShellProperty オブジェクト 3513
ISWiShortcut オブジェクト 3513
ISWiShortcuts コレクション 3583
ISWiSISProperties コレクション 3584
ISWiSISProperty オブジェクト 3522
ISWiStringEntries コレクション 3591
ISWiStringEntry オブジェクト 3541
ISWiSuiteAction オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3635
ISWiSuiteActionRef オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3643
ISWiSuiteActionRefs コレクション (アドバンスト UI、スイート/アドバンスト UI) 3738
ISWiSuiteActions コレクション (アドバンスト UI、スイート/アドバンスト UI) 3738
ISWiSuiteCondition オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3645
ISWiSuiteConditionParameter オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3656
ISWiSuiteConditionParameters コレクション (アドバンスト UI、スイート/アドバンスト UI) 3739
ISWiSuiteConditions コレクション (アドバンスト UI、スイート/アドバンスト UI) 3740
ISWiSuiteDFLFilter オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3657
ISWiSuiteDFLFilters コレクション (アドバンスト UI、スイート/アドバンスト UI) 3740
ISWiSuiteDynamicFileLink オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3660
ISWiSuiteDynamicFileLinks コレクション (アドバンスト UI、スイート/アドバンスト UI) 3741
ISWiSuiteEvent オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3663
ISWiSuiteEvent メソッド (アドバンスト UI、スイート/アドバンスト UI) 3622
ISWiSuiteEvents コレクション (アドバンスト UI、スイート/アドバンスト UI) 3742
ISWiSuiteExitCondition オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3666
ISWiSuiteExitConditions コレクション (アドバンスト UI、スイート/アドバンスト UI) 3743
ISWiSuiteFeature オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3666
ISWiSuiteFeatures コレクション (アドバンスト UI、スイート/アドバンスト UI) 3744
ISWiSuiteFile オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3674
ISWiSuiteFolder オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3675
ISWiSuiteFolders コレクション (アドバンスト UI、スイート/アドバンスト UI) 3746
ISWiSuiteOperation オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3681
ISWiSuitePackage オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3686
ISWiSuitePackages コレクション (アドバンスト UI、スイート/アドバンスト UI) 3747
ISWiSuiteRelease オブジェクト (アドバンスト UI、スイート/アドバンスト UI) 3713
ISWiSuiteReleases コレクション (アドバンスト UI、スイート/アドバンスト UI) 3748
ISWiUpgradeTable オブジェクト 3542
ISWiUpgradeTableEntries コレクション 3592
Move メソッド (アドバンスト UI、スイート/アドバンスト UI) 3656
MoveDown メソッド (アドバンスト UI、スイート/アドバンスト UI) 3659, 3712, 3713
MoveTo メソッド (アドバンスト UI、スイート/アドバンスト UI) 3644
MoveUp メソッド (アドバンスト UI、スイート/アドバンスト UI) 3660
OpenProject メソッド 3339
OpenProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3625
RemoveComponent メソッド 3399
RemoveComponentSubFolder メソッド 3378
RemoveDynamicFileLinkingメソッド 3379
RemoveEnvironmentVar メソッド 3380
RemoveFeatureメソッド 3341
RemoveFile メソッド 3380
RemoveMergeModule メソッド 3400
RemoveObject メソッド 3401
RemovePackage メソッド (アドバンスト UI、スイート/アドバンスト UI) 3673
RemoveRemoveFile メソッド 3381
RemoveSequenceRecord メソッド 3579
SaveProject メソッド 3342

SaveProject メソッド (アドバンスト UI、スイート/アドバンスト UI) 3626
スイート/UI およびアドバンスト UI プロジェクト 3595
スタンドアロン オートメーション インターフェイス 1346
ビルドのステータス イベント 1980
オートメーションの ProgressIncrement イベント 1980
オートメーションの ProgressMaxイベント 1980
オートメーションの StatusMessageイベント 1980
応答トランスフォーム 1520
作成 1520
変更 1521
オブジェクト 2614
2620
InstallScript プロジェクトに追加 710
Windows Installer ベースのプロジェクトに追加 708
アップデートを取得する 481
ウィザード 1663
"オブジェクト ステータス" プロパティ 1675
オブジェクト固有の関数 1686
オブジェクトについて 2620
オブジェクトを含む InstallScript プロジェクトをアップグレードする 452
ギャラリーを管理 700
国際化 1666
作成 1659
サポートされていない関数 1685
サポートされていない機能 1680
サポートされていない定数 1685
システム変数 1688
使用中のコンピューターへのダウンロード 702
スクリプト定義のインストール先 1688
ステータスのプロパティ 1675
設計 1660
テスト 1682
登録 707
配布 1684
ビルド 1681
ファイル 1663
プロジェクトから削除 712
プロパティ 1668
メソッド 1674
ライブ再配布可能ファイルギャラリー 700
利用中のコンピューターに最新版をダウンロードする 702
ローカライズ 1665
ビュー 2620
オペレーティング システムの要件
インストールがチェックするタイミング 344
オペレーティング システム要件 343
親のないディレクトリ エントリ 1889

か

下位 31 ビット 902
会社名
Setup.exe のプロパティ ダイアログ ボックスに表示 1497
Update.exe のプロパティ ダイアログ ボックスに表示 1727
カスタム アクション 908
_serial_verifyCA_isx 3761
_serial_verifyCA_isx_helper 3761
32 ビットと 64 ビットの違い 1320
CheckForProductUpdates 3761
CheckForProductUpdatesOnReboot 3761
DLL へのパラメーターのパス 925
DLLWrapCleanup 3761
DLLWrapStartup 3761
IISSQLServerRollbackLoginInfo 3769
InstallScript カスタム アクション 921
InstallShield、説明 3761
ISComponentServiceCosting 3762
ISComponentServiceFinalize 3762
ISComponentServiceInstall 3762
ISComponentServiceRollback 3762
ISComponentServiceUninstall 3762
ISJITCompileActionAtInstall 3763
ISJITCompileActionAtUnInstall 3763
ISNetApiInstall 3764
ISNetApiRollback 3764
ISNetCreateIniForOneUser 3764
ISNetDeleteIniFile 3764
ISNetGetGroups 3764
ISNetGetServers 3764
ISNetGetUsers 3764
ISNetSetLogonName 3764
ISNetValidateLogonName 3765
ISNetValidateNewUserInformation 3765
ISNetworkSharesCosting 3765
ISNetworkSharesFinalize 3765
ISNetworkSharesInstall 3765
ISNetworkSharesRollback 3765
ISNetworkSharesUninstall 3765
ISPrint 3766
ISQuickPatchFinalize 3766
ISQuickPatchFixShortcut 3766
ISQuickPatchHelper 3766
ISQuickPatchInit 3766
ISQuickPatchInit9X 3766
ISQuickPatchInit9X2 3766
ISRunSetupTypeAddLocalEvent 3766
ISSelfRegisterCosting 3767
ISSelfRegisterFiles 3767
ISSelfRegisterFinalize 3767
ISSetAllUsers 3768
ISSetTARGETDIR 3768

ISSetupFilesCleanup 3768
ISSetupFilesExtract 3768
ISSQLQueryDatabases 3768
ISSQLServerCosting 3768
ISSQLServerFilteredList 3768
ISSQLServerFinalize 3768
ISSQLServerInitialize 3769
ISSQLServerInstall 3769
ISSQLServerList 3769
ISSQLServerRemoveLoginInfo 3769
ISSQLServerRollback 3769
ISSQLServerUninstall 3769
ISSQLServerValidate 3769
ISSQLServerWriteLoginInfo 3770
ISUnSelfRegisterFiles 3770
ISVerifyScriptingRuntime 3770
ISXmlAppSearch 3770
ISXmlCosting 3770
ISXmlFinalize 3770
ISXmlInstall 3770
ISXmlRollback 3771
ISXmlUnInstall 3771
LaunchProgramFileFromSetupCompleteSuccess 3771
LaunchReadmeFileFromSetupCompleteSuccess 3771
PowerShell スクリプトの実行 933
QuickPatch に含める 2985
setAllUsersProfile2K 3771
SetAllUsersProfileNT 3771
SetARPINSTALLLOCATION 3771
setUserProfileNT 3771
ShowMsiLog 3771
Windows ロゴ要件 917
WiseScriptFinalize 3772
WiseScriptInitialize 3772
WiseScriptRollback 3772
アイコン 2754
作成 909
シーケンスへの挿入 972
種類 2754
進行状況メッセージを指定する 949
遅延/コミット/ロールバック カスタム タイプとプロパティ 920
動作のドキュメント化 916
プロセスの終了 932
別のシーケンスにコピー 973
マネージ アセンブリ 926
カスタム セットアップ ダイアログ 1034
カスタム ファイル転送操作 900
カスタム ファイルの転送操作を埋め込む 900
仮想化
ビルド出力場所を指定する 389
仮想ディレクトリ 1272
エラーメッセージの構成 1298
ファイルを追加 1283
仮想マシン 1817
存在を検出する 1817
環境変数 2665
追加および変更 2666
パス変数内 590
関数ウィザード 2293
シーケンス 969
管理者権限 1909

き

キー パス 798
キーファイル 643
起動
管理者権限あり/管理者権限なしで InstallShield を起動 298
起動する
InstallShield を 32 ビットと 64 ビット システムで起動する違い 299
機能
Windows の役割と共に有効にする 1561
アドバタイズ 688
"インストールレベル" プロパティ 689
インストール先フォルダー 681
オートメーションレイヤーを使用 3554
関数呼び出しでの機能やサブ機能の指定 899
コンポーネントを追加 638
順序 692
条件付きインストール 682
定義 308
ディスク イメージにファイルを非圧縮のまま残す 1357
ビジビリティ ("表示" 設定) 684
必須とする 688
リモート インストール 690
リリースフラグ 1370
機能、作成 511
基本の MSI セットアップ プロジェクト 319
基本の MSI ダイアログにおけるハイパーリンク 1102
基本の MSI プロジェクトの作成 508
キャビネット & ログ ファイル ビューアー 1898
共有ファイル 653

く

グループ ボックス コントロール 1099
グローバリゼーション 529, 1775
およびダイアログ 1787
言語識別子 1790
コードページ 482
サポートされているセットアップ言語の選択 1781
デフォルト言語 1777
ヒント 1776
ユーザーに言語を選択させる 1789
リリースへの言語の組み込み 1784

グローバル アセンブリ キャッシュ
アセンブリのパッチ 1768
グローバルアセンブリキャッシュ 1768

け

言語サポート 481
言語識別子 1790
言語依存コンポーネント、インストール 1782
検証 1417
ICE 1421
InstallShield MSIX/UWP アプリ適合性スイート 1460
ISICE 1421
ISMSIXUWP 1460
ISVICE 1444
アップグレードとパッチ 1751
オン デマンドで実行する 1420
カスタム アクション 917
検証結果を表示する 1420
実行する ICE を指定する 1419
実行するタイミングを指定する 1419

こ

コード ページ
インストール 482
要件 482
コードページ 482
構成、1 つのプロジェクトから複数の製品設定のインストールを作成する 1374
構文
ハイライト 372
構文の折りたたみ 376
国際化 1775
コスト 900
コマンドライン ビルド
自己展開型実行可能ファイル 1333
コマンドラインのビルド 1326, 1333
コマンドラインを使ったサイレント アンインストール 1416
コンテキスト ヘルプ 302
コントロール 1064
HTML とハイパーリンク サポート 1013
コントロール イベント 1024
コントロールのテーマ 1163
ウィザードインターフェイス 1163
コンパイラ 3775
コンパイラー
InstallScript 3775
コンパニオンファイル 619
コンポーネント 2524
COM 633
Windows サービスのインストール、制御、および構成 851
アプリケーション プロキシ サポートに作成 1266
アプリケーションパスのエントリ 673
インストール先フォルダー 647
オートメーションレイヤーを使用 3548
機能との関連付け 638
共有ファイルのインストール 653
サーバー アプリケーションに作成 1265
作成 630
サブフォルダーの追加 643
自己登録 750
ショートカットの作成 643
条件 651
詳細設定 665
スクリプトからインストール先を指定する 649
データをパーマネントにする 650
定義 308
パブリッシュ 675
ファイルの種類 669
フォント 634
リモート インストール 656
レジストリ エントリの作成 642
オプション 634
コンポーネント ウィザード
ベスト プラクティス オプション 632
コンポーネント サービス 2727
アプリケーション プロキシ サポート 1266
サーバー アプリケーション 1265
サーバーおよびプロキシへの条件付きインストール 1268
プロジェクトに追加 2727
コンポーネントの詳細設定 665
コンポーネントのパブリッシュ 675
コンポーネント ウィザード
634

さ

サーバー 1268
アプリケーション プロキシ用に指定 1266
場所
削除 1523
追加 1522
複数の順番変更 1523
変更 1522
サーバー アプリケーションのプロキシ サポート
COM+ アプリケーションを条件付きでインストール 1268
サーバー アプリケーション用プロキシ サポート 1266
ビュー 2700
2700
サービス 851
インストール、開始、停止、削除、アンインストール、および構成 851
最後のディスク フォルダー、ファイルを追加する 986
サイドロード 1542
アプリケーション パッケージ (.appx) 1542

再配布可能ファイル
.NET Framework および言語パック 733
InstallScript プロジェクトに追加 710
InstallShield 698
Windows Installer ベースのプロジェクトに追加 708
アップデートを取得する 481
ギャラリーへマージ モジュールを追加 706
ギャラリーを管理 700
ライブギャラリー 700
ライブ再配布可能ファイル
最新版をマシンにダウンロードライブ 702
サイレント インストール 1396
削除
InstallShield 前提条件を再配布可能ファイル ギャラリーから削除 704
サーバーの場所 1523
再配布可能ファイル ギャラリーから削除 704
再配布可能ファイル ギャラリーからマージ モジュールを削除 707
プロジェクトから InstallShield 前提条件を削除 711
プロジェクトからマージ モジュールおよびオブジェクトを削除 712
作成 630
2 つの .msi ファイルを比較して作成されたトランスフォーム 1516
InstallShield 前提条件 1633
QuickPatch プロジェクト 1743
概要 1743
既存の QuickPatch 1744
新規 QuickPatch 作成ウィザード 2238
応答トランスフォーム 1520
カスタム アクション、ガイドライン 917
機能 680
セットアップ プロジェクト 326
単一の .msi パッケージを基に作成されたトランスフォーム 1516
サブ Web
ファイルを追加 1283
サブスクリプション 1028
差分 1887
差分リリースと完全リリース 1713
サポート 343
サポート ファイル 983
サポートされていない関数 890

し

シーケンス 2751, 969, 971
アクションを挿入 972
アドバタイズ シーケンス 967
インストール シーケンス 954
カスタム アクション 976
スモール アップデート パッチ 1729
前提条件をインストール 717
シールド アイコン 540
自己展開型実行可能ファイル 1333
ReleasePackager.exe を使ってビルド 3794
コマンドラインからビルド 1333
ビルド 1356
自己登録
COM サーバー サポート 746
自己登録ファイル 500, 524, 750
自己登録の順序指定 752
ビュー 2778
システム検索ウィザード 2357
レジストリの検索について 1873
ビュー
レジストリの検索について 1873
2621
システムの復元システムのふくげん 878
1873, 2778, 2621
シーケンス 971
実行可能ファイル
起動 935
単一の実行可能配布ファイルの作成 1356
配布する単一の実行可能ファイルを作成
InstallShield 前提条件を含める条件 721
実行可能ファイルの起動 935
ウィンドウ 2018
固定/取り外す 367
ショートカット 498, 521, 765
InstallScript と InstallScript MSI プロジェクトのアンインストール ショートカットの作成 782
アイコン 769
基本の MSI プロジェクトのアンインストール ショートカット作成 781
ショートカットの作成 643
ビュー 2625
ピン留め 774
プロパティの設定 769
ショートカットのピン留め 774
メニュー 776
ショートカットを抑制する
Windows スタート画面から 775
上位 31 ビット 902
上位/下位値 902
使用許諾契約
ユーザーが読むのを必須にする 1058
条件 1807
および InstallShield 前提条件 1636
および機能 682
およびコンポーネント 651
および製品 558
構文 1810
スイート/アドバンスト UI およびアドバンスト UI プロジェクトでのチェックの種類 1823
条件グループ 1820
条件付きインストール 558

条件ツリー 1820
条件の再評価 652
ダイアログ 2109
進行状況ダイアログ 949
アクションの起動メッセージを指定する 949
進行状況バー（基本の MSI プロジェクト） 1206
ビルボードを使って表示、または使わずに表示 1206
進行状況バー コントロール 1127

す

スイート/アドバンスト UI インストール 317
.appx パッケージを追加する 1542
.exe パッケージを追加する 1540
.msi、.msp、およびトランザクションをパッケージに追加する 1534
.prq ファイルをインポート 1550
InstallScript アクション内 1573
InstallScript イベント、関数、および変数 1536, 1537
InstallScript パッケージ内 1537
InstallScript パッケージを追加する 1536
InstallShield 前提条件をインポート 1550
InstallShield プロジェクトからパッケージを追加する 1539
Setup.exe コマンドライン パラメーター 3821
UAC プロンプト 1603
Web 配置パッケージを追加する 1541
アドバンスト UI インストールとの違い 1527
依存関係パッケージ 1552
オブジェクト式 1602, 3749
カスタム パッケージ フォルダー名 1560
機能とパッケージの関連付け 1554
機能の状態 1598
共有パッケージ 1555
形式化された式 1600
システム検索 1602
自動アップデート 1609
ダイナミック ファイル リンク 1546
ダウンロード可能なアップデート 1609
トラブル シューティング 1618
パスワード保護 1617
パッケージにコマンドライン パラメーターを渡す 1591
パッケージのインストール順 1553
パッケージの操作 1558
パッケージのランタイムの場所 1557
パッケージのログ記録 1623
パッケージを追加するときのガイドライン 1532
含まれているファイル 1525
プライマリ パッケージ 1552
プログラムの追加と削除エントリ 1590
プロパティ 1935
マネージ コード アクション 1575, 1578
モード条件 1589
予期しないメンテナンス モード 1618
ログ記録 1618
スイート/アドバンスト UI インストールのアクション 1564
.exe 1565
DLL 1566
PowerShell 1570
種類 1585
スケジュール 1583
スイート/アドバンスト UI およびアドバンスト UI インストールの DPI 対応サポートとイメージ 1195
スイート/アドバンスト UI またはアドバンスト UI プロジェクトの機能条件 1820
スイート/アドバンスト UI またはアドバンスト UI プロジェクトの検出条件 1820
スイート/アドバンスト UI またはアドバンスト UI プロジェクトの終了条件 1820
スイート/アドバンスト UI またはアドバンスト UI プロジェクトの対象条件 1820
スキン、InstallScript および InstallScript MSI ダイアログ 1017
制限 1017
スクリプト エディター 2750
スクリプト エディターのオートコンプリート 368
スクリプト エディターのブックマーク 368
スクリプト ファイル 862
スクロール可能テキスト コントロール 1143
ショートカットのピン留め
Windows タスクバーまたは 776
スタイル 1163
ウィザード インターフェイス 1163
ウィザード フォーマットを選択 1173
カスタム スタイルをウィザード インターフェイスに定義する 1164
テキスト スタイルをウィザード インターフェイスのテキストに適用する 1165
スモール アップデート 1729
パッチ シーケンス 1729

せ

製品
インストール先フォルダー 560
機能とコンポーネントの階層構造 308
条件付きインストール 558
テンプレートの概要 576
製品構成 1320
1 つのプロジェクトから複数の製品設定のインストールを作成する 1374
製品名
Setup.exe のプロパティ ダイアログ ボックスに表示 1497
Update.exe のプロパティ ダイアログ ボックスに表示 1727
セキュリティ 1366
セットアップ ベスト プラクティス 628
セットアップ デザインを監視 2112
セットアップ前提条件

アップデートを取得する 481
ユーザー アカウント制御のプロンプト 540
セットアップの種類 493, 2565
2508
ビュー 2508
セットアップのテスト実行 1395
選択ツリー コントロール 1145
前提条件 713
64 ビット サポート 1636
InstallScript プロジェクトに追加 710
InstallShield 前提条件 713
Oracle 11g Instant Client 738
Windows Installer ベースのプロジェクトに追加 708
アップデートを取得する 481
依存関係 1649
インストール一覧に表示しない 1646
インストール順 717
インストール条件 1636
インストールのパラメーターを指定する 1641
エンド ユーザーによってスキップ可能 1646
起こりうる問題の対処方法について 1649
概要 713
管理者権限が必要 1646
機能から削除 716
機能との関連付け 715
個別の前提条件の実行時の場所を設定 720
コマンドライン パラメーター 1642
コンピューターの再起動を指定する 1649
再配布可能ファイル ギャラリーから削除 704
再配布可能ファイル ギャラリーに追加 704
作成および変更 1633
実行時の場所を指定 1362
進行状況バーを表示する 1648
前提条件をインストール後にターゲットマシンを再起動する 1645
前提条件を使ってアプリケーションをアンインストール 726
前提条件を使ってインストールを実行 722
前提条件を含むリリースを構成 718
代替ダウンロード URL を指定する 1635
ダウンロード URL 1640
動作 1645
必要実行レベルを指定 1358
ビルド時の場所を設定 719
ファイル 1639
プロジェクトから削除 711
プロパティ 1634
保存 1650
ユーザー アカウント制御のプロンプト 540
リリース フラグと場所の設定 721

そ

ソース管理 1957
コマンドライン 1978
ソース管理からファイルをチェックアウトする 1959
ソース管理へファイルをチェックインする 1960
ソースの場所 655
その他のウィンドウ スタイル 2131
その他のディスク フォルダー、ファイルを追加する 987
ソフトウェア識別タグ
および SCCM アプリ モデル データ 580
579

た

ターゲット マシンの再開始
前提条件に指定する 1641
ターゲット マシンの再起動
Windows Vista 以降における UAC プロンプトへの影響 540
インストールまたはアンインストール中 (InstallScript または InstallScript MSI) 1487
前提条件に指定する 1641
ターゲット マシンの要件
インストールがチェックするタイミング 344
ターゲット マシンの再開始
インストールまたはアンインストール中 (InstallScript または InstallScript MSI) 1487
ターゲット マシンを再開する 1645
ターゲット マシンを再起動する 1645
ターゲット言語、選択 529
ターゲットマシンの要件 343
ダイレクト編集モード 325
ダイアログ 989
ダイアログ サンプラー 1016
ビュー 990
プロジェクトから削除 997
ランタイム言語サポート 481
ダイアログ ウィザード 2279
ダイアログ エディター 505, 528
ダイアログ エディター、InstallScript プロジェクト 1003
ダイアログ エディター、基本の MSI プロジェクト 1021
ダイアログ サンプラー 1016
ダイアログ スキン
制限 1017
ダイアログ テーマ 1036, 1054
ダイアログ ボックスのプロパティ 1085
ダイアログ ボックスのユーザーのデフォルトの選択 1520
ダイアログのインポート 993
ダイアログのコントロール 1064
ダイアログのスキン 1017
ダイアログをエクスポートする 993
ダイナミック ファイル リンク 610
アドバンスト UI およびスイート/アドバンスト UI プロジェクトで 1546
コンポーネントの作成のベスト プラクティス 612
制限事項 611

追加 615
ディレクトリごとのメソッド 612
タイル構成
設定 2648
タイルの構成
スタート画面上のデスクトップ アプリのタイルを構成する 783
ダイレクト MSI 325
ダイレクト エディター 2952
テーブル レコード参照の追跡 2957
列ヘッダー スキーマのツールヒント 2956
ダイレクト編集モードの MSI データベース 328
ダウングレード、防止 547
タグ 579
SCCM アプリ モデル データを含める 580
製品を識別 579
単一の実行可能ファイル、作成 1356

ち

チェック ボックス コントロール 1075, 1080
遅延カスタム アクションとプロパティ 920

つ

追加 1520
InstallShield 前提条件、マージ モジュール、およびオブジェクトを InstallScript プロジェクトに追加 710
Windows Installer ベースのプロジェクトに前提条件、マージ モジュールおよびオブジェクトを追加 708
サーバーの場所 1522
再配布可能ファイル ギャラリーに InstallShield 前提条件を追加 704
再配布可能ファイル ギャラリーにマージ モジュールを追加 706
ファイルを IIS 仮想ディレクトリに追加 1283
ファイルをソース管理に追加 1958
ビュー 2949
追加
プロジェクト アシスタント ダイアログサポートを既存の InstallShield Professional プロジェクトへ追加 470

て

データベース (SQL) 2268
テーマ
ウィザード インターフェイス内のコントロールのテーマを定義する 1169
テーマ、ダイアログ 1036, 1054
定義済み検索 2030
ディスクの分割 1492
ディスク分割
カスタム ディスク分割 1494
ディスク分割規則 1493
ディレクトリ コンボ コントロール 1088
ディレクトリ リスト コントロール 1091
テキスト領域コントロール 1149
デジタル証明書 1366
デジタル署名 1366
QuickPatch パッケージ 1748
アプリケーション 1366
タイムスタンプ サーバー 380
パッチ パッケージ 1739
ファイル 1366
デジタル署名のタイムスタンプ サーバー 380
デバイス ドライバー
InstallScript 903
パネル 2276
デバイス ドライバー (オートメーション) 3373, 3374
デバイス ドライバー ウィザード 2275
デバイス ドライバー
署名
デバイス ドライバー ウィザード – 2276
デバイスドライバー 674, 2276
デフォルト 1520
言語 1777
テンプレート 335
プロジェクト 335

と

動作エディター 1024
ビュー 2746
デバイス ドライバー
インストールのシーケンス
デバイス ドライバー詳細設定デバイス 674
トラブルシューティング 3111
トランザクション処理 1313
概要 1313
トランスフォーム 1515
2 つの .msi ファイルを比較して作成 1516
単一の .msi パッケージを基に作成 1516
適用する 1517
デフォルトの応答のカスタマイズ 1520
編集 326
トランスフォーム オープン ウィザード 2310
トランスフォームウィザード 2368
トランスフォームの自動エンド ユーザー応答 1520
トランスフォームの適用 1517

な

名前空間 834
XML ファイルで宣言 834, 837
プレフィックスを要素から削除する 837
プレフィックスを要素に追加する 835, 836
ダイアログ ボックス 2184

ね

ネスト インストール 940
ネットワーク リポジトリ 335
　概要 335
　設定 335

は

バージョン番号、InstallShield セットアップ 555
背景ウィンドウ
　InstallScript と InstallScript MSI インストール中に表示する 1220
パス 1124
　変数 2811
　　環境 590
　　定義済み 587
　　標準 588
　　標準パスの変換 593
　　レジストリ 589
ビュー 2811
パス変数と DIM 762
パスワード保護 1362
　QuickPatch 1748
 2811, 2566
ビュー 2566
パッケージ マネージャー 1561
パッチ 1729
　アップグレードとの違い 1712
　アンインストール 1732
　インストールが複数インスタンスをサポートする製品 1803
　共有コンポーネント 663
　グローバル アセンブリ キャッシュのアセンブリ 1768
　スモール アップデートのシーケンス 1729
バッチ ビルド 670
パッチの適用 2309
 1701
ビュー 1701
パッチの構成 1701
パッチ
　非管理者による 1733

ひ

非管理者によるパッチ 1733
　要件 1733
ビットマップ コントロール 1070
″必要実行レベル″ 設定 1358
ビュー フィルター 785
ビュー リスト 2389
開く
　.msi ファイル 328
　InstallShield 前提条件エディター 2991
　インストール プロジェクト 327
ビルド イベント 1336
ビルボード 1205
　Flash とイメージ ファイルの設定 2803
　InstallScript プロジェクトと InstallScript MSI プロジェクト 1213
　　実装するコード 1218
　　種類 1213
　　順番を設定 1218
　　スクリーン ショットのサンプル 1213
　　配置変更 1219
　　ファイルに名前を付ける 1215
　　有効化 1220
　基本の MSI プロジェクト
　　Flash ファイルを追加する 1209
　　イメージを追加する 1210
　　削除する 1213
　　サポートされているファイルの種類 1206
　　実行時の動作 1212
　　種類 1206
　　順番を設定 1211
　　使用する種類を指定する 1209
　　スクリーン ショットのサンプル 1206
　　設定を構成する 1210
　　リリースをビルドおよび実行せずにプレビューする 1211
　基本の MSI プロジェクトの場合 1205

ふ

ファイル 594
　圧縮せずにディスク イメージに残す 1357
　検索 603
　スクリプトからインストール先を指定する 649
　前提条件に指定する 1639
　ソースの場所へのダイナミック リンク 610
　属性 604
　ターゲット システムで上書き 617
ファイル、追加 511
 2607
ビュー 2607
ファイルの種類 669
ファイルの署名、デジタル 1366
ファイルのバージョン規則 617
ファイルのプロパティ 2070
　ファイルとディレクトリのアクセス許可のロック 620
ダイアログ 1055
 1055
フォーム認証 1302
フォント (InstallScript プロジェクトと共にインストールする) 626
フォントコンポーネント 634
複数インスタンスのサポート 1795

ProductCode の構成 1798
インスタンスに名前を付ける 1797
構成 1796, 1803
プロパティの構成 1799
ランタイムの動作 1804
ランタイム要件 1795
複数行のレジストリ値 801
複数の製品構成、1 つのプロジェクトで作成 1374
複数パッケージ インストール 1313
概要 1313
トランザクション プロセスを使用する 1313
"複数パッケージの共有コンポーネント" 設定 663
プッシュ ボタン コントロール 1130
ブラシ 1163
ウィザード インターフェイス 1163
カスタム スタイルをウィザード インターフェイスに定義する 1166
テーマをウィザード インターフェイスのコントロールに適用する 1169
ブラシをウィザード インターフェイスの背景に適用する 1170
プロキシ サーバーの設定 1508
プログラム互換性アシスタント 222
プロジェクト アシスタント 338, 488
プロジェクト テンプレート 335
作成 336
プロジェクトの種類として使用 337
プロジェクトの種類 314
Visual Studio プロジェクトを InstallShield プロジェクトに変換またはインポートする 477
別のプロジェクトへの変換 330
プロセスの終了 932
プロパティ マネージャー 2779
使用について 1942
制限付きパブリック プロパティ 1908
パブリック プロパティ 1908
必須プロパティ 1908
プライベート プロパティ 1908
プロパティのローカライズ 1945, 1946

へ

ベスト プラクティス 628
インストールの作成 628
ダイナミック ファイル リンク 612
別の InstallScript インストールからのインストール 904
ヘルプ
コンテキスト ヘルプ 302
使用 302
変換 330
Visual Studio プロジェクトを InstallShield プロジェクトに変換 477
プロジェクトを別の種類のプロジェクトに変換 330
変更
InstallShield 前提条件 1633
応答トランスフォーム 1521
トランスフォーム 1518
編集フィールド コントロール 1094
ビュー 2506

ほ

保存
InstallShield 前提条件 1650
新しい名前または場所にプロジェクトを保存 329
ボリューム コスト リスト コントロール 1154
ボリューム選択コンボ コントロール 1157

ま

マージ モジュール 2614
InstallScript プロジェクトに追加 710
Windows Installer ベースのプロジェクトに追加 708
アップデートを取得する 481
再配布可能ファイルギャラリーから削除 707
再配布可能ファイル ギャラリーに追加 706
参照 705
使用中のコンピューターへのダウンロード 702
ダイレクト編集モードで追加 1955
ビルド時の場所を設定 727
プロジェクトから削除 712
マージ モジュールについて 2614
ライブ再配布可能ファイルギャラリー 700
マージ モジュール プロジェクト 324
依存関係の追加 1655
言語の選択 1655
デフォルトのインストール先フォルダーを設定 1654
除外ファイルの追加 1656
マイナーアップグレード 1700
マスクされた編集コントロール 1122
マップされたドライブ
プロジェクトで参照 298
マニフェスト 752
サンプル 754
マネージ アセンブリ 926, 928
カスタム アクション 926
32 ビットと 64 ビットの違い 928

み

右から左方向へ読み書きされる言語のサポート 482

め

メジャー アップグレード 1699
2810
ビュー 2810

メンテナンス 1305
　メンテナンスおよびアンインストールのためのインストールを作成する 1305
″メンテナンスを有効にする″ プロパティ 567

も

モード条件 1820
文字列
　翻訳のための自動エクスポートとインポート 1974
文字列 エントリ
　翻訳のための自動エクスポートとインポート 1974
文字列エントリ
　InstallScript コードを使用 866
　InstallShield にアクセス 1198
　ビュー 2807
　複数言語 1197
　編集 1201
文字列エントリの翻訳 481
文字列テーブルのエントリ、作成 531
戻り値
　Setup.exe 実行時(InstallScript) 3125
　Setup.exe 実行時 (基本の MSI および InstallScript MSI) 3128
　Setup.exe の実行時 (スイート/アドバンスト UI およびアドバンスト UI) 3133

ゆ

ユーザー アカウント (追加または設定) 991
ユーザー アカウント制御 (UAC) とインストール 540
 971, 2791
シーケンス
　971
 971
ビュー 2791
ユーザー インターフェイスのローカライズ 1196
ユーザーごとのインストールとマシンごとのインストールの違い
　レジストリ エントリ 802

ら

ライブ再配布可能ファイル ギャラリー 700
　関連付けられたアイコン 700
ライブラリ ファイル、作成 867
ラジオ ボタン グループ コントロール 1138
ラジオ ボタン コントロール 1135

り

リスト ビュー コントロール 1117
リスト ボックス コントロール 1112
リスト ボックス コントロール、スイート/アドバンスト UI またはアドバンスト UI プロジェクト 1184
　構成 1184
リポジトリ 335
　概要 335
　ネットワークの設定 335
リモートインストール 690
利用中のコンピューターへの最新再配布可能ファイルのダウンロード 702
リリース
　ディスク イメージにファイルを非圧縮のまま残す 1357
リリース ウィザード 494, 516, 2317
リリース フラグ 1370
リリースのビルド 494, 516, 2829
　.ini ファイルでパラメーターを渡す 1327
　Windows Installer を含める 730
　クイック ビルド 1334
　コマンドラインからの自己展開型実行可能ファイルのビルド 1333
　コマンドラインを使う 1326
　前提条件のランタイムの場所を指定 1362
　バッチ ビルド 1335
　ビルド エラーと警告のトラブルシューティング 3006
　ビルドのキャンセル 1340
　ビルドの再ビルド 1333
　リリースをコピー 1393
リリースをビルド
　オートメーション インターフェイスを使う 3505

れ

レジストリ 499, 522
　Reg-Free COM 752
　リフレクションの有効化または無効化 662
レジストリ エントリ
　REG ファイルのインポート 797
　REG ファイルのエクスポート 798
　アンインストール動作を設定 795
　キー パス 798
　キーの作成 787
　コンポーネントのレジストリ データでのテキストの検索 794
　フラグ 793
　ユーザーごとのインストール 802
　レジストリ値 790
レジストリ リフレクション 662
レジストリのエントリ 784
連鎖 .msi パッケージ 1313
　概要 1313
　単一のトランザクションとして処理される 1313

ろ

ローカリゼーション 1196, 1775

ローカル リポジトリ 335
    概要 335
ロール 1561
    Windows の機能と共に有効にする 1561
ログ ファイル
    アクション情報を含める 949
    基本の MSI および InstallScript MSI Setup.exe に生成 3813
    スイート/アドバンスト UI インストールにおける
        InstallScript アクションのデバッグ 3822
    スイート/アドバンスト UI およびアドバンスト UI
        Setup.exe に生成 3822
    スイートのパッケージに生成 3823
ロックされた環境 1733
ロックダウン環境
    パッチ 1733