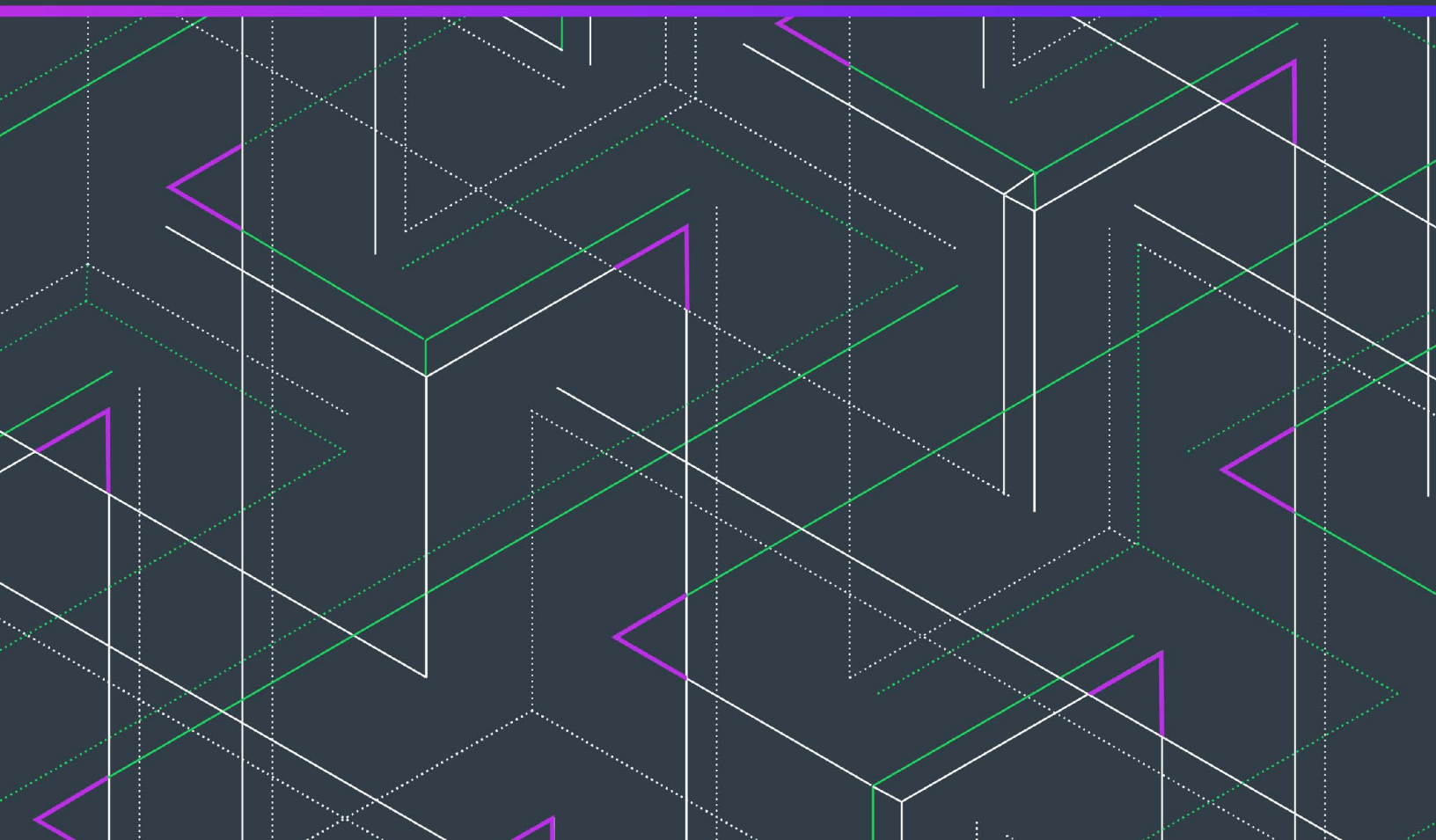
revenera

InstallShield 2021

InstallScript デバッガーユーザー ガイド



法的情報

文書名: InstallShield 2021 InstallScript デバッガー ユーザー ガイド

部品番号: ISP-2700-UG00

製品のリリース日: 2021 年 12 月

著作権情報

知的財産

(米国内向け) 制限付権利に関する表示

目次

InstallScript ベース インストールのデバッグ

デバッグは、コンピューター ソフトウェアのロジック エラー を発見して修正するプロセスです。それらのエラーは、スクリプトの異常動作や予期しない停止の原因となります。構文エラーとは異なり、ロジック エラーはコンパイラによって検出されません。コンパイラが保証することは、ステートメントが正しく*表現*されているかどうかだけです。ロジック エラーがあるステートメントでも、コンパイラにとっては完全に正しいものに見えます。コンパイラは、それらのステートメントによって表現されている命令に不備がなく正しいかどうかを判別できません。

ロジック エラーの検出

完成したスクリプト中のロジックをエラーを発見する最良の方法は、スクリプトを実行してその動作を観察することです。重大なロジック エラーの大部分はスクリプトの開発およびテスト期間中に現れますが、中には使用されるようになった後、長い期間が過ぎてもスクリプト中にひそんでいるものもあります。そのようなロジック エラーは、特定のコンピューター上でスクリプトが実行されたときや、ユーザーが特定の関数のシーケンスまたは特定の入力値を選択したときに現れるので、予想できません。

ロジック エラーは、スクリプトを突然停止させたり、ランタイムエラーメッセージを表示したりする重大なものである場合もあります。また、スクリプトは実行されるがその表示や動作に問題があるという、微妙なものである場合もあります。たとえば、選択していないフォルダーにファイルをインストールする、セットアップ時にオプションが選択されているにもかかわらずデスクトップ上または Windows の [スタート] メニューにショートカットを作成しない、などが考えられます。

ロジック エラーの解決

バグを発見したり、バグの報告を受け取ったら、以下の手順に従う必要があります。

1. セットアップを実行し、エラーを再現させます。作業を続ける前に、エラーの性質を必ず理解しておきます。
2. スクリプトを検討し、エラーの原因と考えられる場所を特定します。
3. **InstallScript デバッガー** を使用してエラーを解析します。
4. **InstallScript** ビューを使用して、スクリプト中のエラーを修正します。
5. スクリプトを再コンパイルし、エラーが修正されていることを確認します。

InstallScript デバッガー

InstallScript デバッガーはソース レベルのデバッガーです。同じウィンドウの別々のペインにデバッグ制御とインストール スクリプトを表示します。ウィンドウの [スクリプト] ペインには、次に実行されるステートメント（実行点）がマーカー付きで示されます。

InstallScript デバッガーからスクリプトをステートメントごとに実行できます。また、[InstallScript デバッガー] ウィンドウの スクリプトペイン中を移動する実行点をモニターすることによって制御の流れを追跡できます。また、スクリプト実行中どの時点でも、スクリプト中の変数の値を監視することもできます。これらの方法によって、スクリプトのエラーや非効率性の原因を特定することが容易になります。

スクリプト ウィンドウ

InstallScript デバッガーのスクリプト ウィンドウには、スクリプトが表示され、実行時しながら確認できます。次に実行するステートメントがウィンドウ内に配置され、黄色の矢印で示されます。ブレークポイントが含まれる行には、左に赤丸が表示されます。行をダブルクリックしてブレークポイントを設定または削除することはできません。

スクリプト ウィンドウのタイトル バーに、セットアップ ファイルのフル ネームが表示されます。[ステップイン] または [ステップオーバー] コマンドを使用して一群のステートメントをトレースする際に、必要な場合はスクリプト ウィンドウが自動的にスクロールします。同様に、スクリプトの実行がブレークポイントで停止した場合、このウィンドウは自動的にスクロールしてブレークポイントの位置を表示します。

ウォッチ ウィンドウ

InstallScript デバッガーの [ウォッチ] ウィンドウには、監視するように選択した各文字列変数と数値変数の現在の値が表示されます。[ウォッチ] ウィンドウは、InstallScript デバッガーの右下隅に表示されます。デバッグの間に、いつでもこのウィンドウに変数を追加または削除できます。

ボタン コントロール

InstallScript デバッガー ツールバー上にあるボタンを使って、スクリプトの実行を制御できます。

Table 1 · InstallScript デバッガーのボタンの説明

ボタン	説明
開く	スクリプト ファイルを開きます。
ブレークポイントの切り替え	ブレークポイントの切り替えボタンによって、ブレークポイントを配置したり削除したりできます。
Go	スクリプトの実行を開始します。このボタンを使用して次のブレークポイントまで実行します。ブレークポイントがない場合は、スクリプトの終わりまで実行します。
ブレーク	ブレークポイントをセットまたはクリアします。

Table 1 • InstallScript デバッガーのボタンの説明 (続き)

ボタン	説明
ステップイン	スクリプトの次のステートメントを実行します。そのステートメントがユーザー定義関数の呼び出しだった場合、デバッガーは、コード ウィンドウにその関数を表示し、実行点の位置がキーワード begin に続くステートメントに移ります。
ステップ オーバー	スクリプトの次のステートメントを実行します。そのステートメントがユーザー定義関数の呼び出しだった場合、その関数中のすべてのステートメントが実行され、実行点の位置は関数呼び出し後に実行される次のステートメントに移ります。
ステップ アウト	現在のルーチンをスキップします。
次のステートメントを表示	スクリプト ファイル中の次のステートメントが表示されます。

ユーザー変数

コード ウィンドウの [ユーザー] 変数セクションを構成するコントロールを使用して、ローカル変数の現在の値の調査し、ローカル変数を [ウォッチ] ウィンドウに追加することができます。

エラーの再現

デバッグ プロセスの最初の目標は、スクリプトおよびエラーを発生させると考えられるステップを実行してバグを再現することです。このステップが重要であるのは以下のような理由によります。

- バグがエンドユーザーから報告されたものである場合、報告の詳細は不正確で不完全である可能性があります。その場合、不適切な場所で不適切な問題を探すことに時間を費やすことになります。
- バグ自体が、特定のハードウェアおよびソフトウェア環境でしか起こらないものである可能性があります。ユーザーの報告したバグを再現できない場合、次の作業に取り組む前に、自分のシステムとユーザーのシステムとの違いをはっきりさせておくことをお勧めします。
- バグを見つけたとき、バグが発生した状況を正確にドキュメント化するために、予期されたタイミングでもう一度それが発生するのを見る必要が出てくる場合があります。
- 通常、バグを再現することで、バグ発生の場所と原因についての手ががりが得られます。
- バグを再現させる方法がよく分からない場合は、後で本当に問題が修正されたのかどうかを確かめることもできません。

エラーの原因の特定

デバッグ プロセスの 2 番目のステップは、エラーに関連する部分をセットアップ内で特定することです。

状況を自分で発見した場合、大抵の場合、原因についてどこを探せばいいか既に見当がついています。それでも、デバッグ作業に取り組む前に、エラーについて考え、そのエラーを起こす可能性のある明らかな設計やコーディングのミスを特定することをお勧めします。

最も単純なエラーでも、それによってセットアップのさまざま部分を調べる結果となることがよくあります。たとえば、標準ダイアログに製品名が表示されないとします。エラーはこのダイアログが呼び出される場所で見つかるかもしれません。または、初期化セクションが **SdProductName** の呼び出しによって製品名を確立することに失敗する可能性もあります。あるいは、**SdProductName** の呼び出しが未定義の文字列エントリを参照している可能性もあります。

これらの特定のバグの処理には、デバッガーは必要ないかもしれません。しかし、デバッガーが必要な場合には、手掛かりを見つけるためにスクリプトのどの部分をトレースし解析するかの計画を立てることによって、デバッガーをより効果的に使用することができます。

エラー原因の分析

効果的なデバッグ ツール無しでは、頭の中でスクリプトを「実行」してエラー発生の可能性がある部分のコードを分析するしかありません。まず疑わしい範囲の論理を追って一行一行注意深く読み、そこに現れるさまざまな変数を追跡することになります。スクリプトが間違った方向に進み始めた場所を見つけるまでこれを続けなくてはなりません。そして、異常な動作の理由を正確に解明できるまでこれらの作業を何度も繰り返すことになるでしょう。それがデバッグ ステップの目的、つまりエラーの正確な原因を見つけることだからです。

ソフトウェア デバッガーは、デバッグ プロセスのこの部分のために設計されたツールです。デバッガーは自動的にスクリプトを解析するものではありませんが、セットアップが異常に動作する理由の発見を容易にします。デバッガーを使用すると、1 つのステートメントごとにスクリプトを実行することによって、制御の流れを追うことができます。そして、スクリプト実行中の任意の時点におけるスクリプト中の変数の値を判別することができます。

エラーの修正

スクリプト中にエラーの原因となっている箇所を見つけたら、デバッガーを終了して問題の修正プロセスに取り組みます。修正を行うには、InstallShield で作業中のインストール プロジェクトに変更を加える必要があります。まず修正作業の計画を立て、InstallScript ビューを使用して変更を加えます。最後に、スクリプトを再コンパイルします。

実行ポイント

実行点は、スクリプト中の*次に*実行されるステートメントの位置です。この位置は、スクリプト ウィンドウ中にその属性とともに表示されます。[ステップイン] または [ステップ オーバー] ボタンを使用してスクリプトをトレースするときに、実行点の移動が制御の流れを表します。

ステップ コントロール

ステップ コントロールには、[ステップイン] ボタンと [ステップオーバー] ボタンがあります。これらのコントロールによって、スクリプトの全部または一部をステートメントごとに実行できます。これらのコントロールを使用してバグを見つけます。

- スクリプト中の疑わしい範囲をステップ実行し、ステートメントが実行される順番を調べます。このプロセスによって、意図通りに動作しない条件式によって発生するバグを発見できます。

- ステップ コントロールを使用してステートメントを実行した後、バグの原因と考えられる変数の値をチェックします。変数を追跡する最も簡単な方法は、その変数を [監視] ウィンドウに追加することです。追跡している変数の値が、解析の各時点で予想される値であることを確認します。
- ステップ コントロールをブレークポイントと共に使用します。まず、スクリプト中の解析したいセクションの最初のステートメントにブレークポイントをセットします。次に、[実行] ボタンをクリックしてステートメントを実行します。最後に、[ステップ イン] または [ステップ オーバー] ボタンをクリックして、ブレークポイントに続くステートメントをトレースします。

ユーザー定義関数へのステップ イン

実行点がユーザー定義関数呼び出しステートメントにあるときに、InstallScript デバッガーのツールバー上の [ステップ イン] ボタンをクリックします。スクリプト ウィンドウが自動的にスクロールしてユーザー定義関数中の最初の実行可能ステートメントが表示されます。そのステートメントは実行されておらず、そこが実行点になります。このステートメントは、次に [ステップ イン]、[ステップ オーバー]、または [実行] ボタンがクリックされると実行されます。

***メモ・**DLL のような別のファイルから呼び出される関数にステップ インした場合には、タイトル バーにそのファイル名が表示されます。*

ユーザー定義関数のステップ オーバー

実行点がユーザー定義関数呼び出しステートメントにあるときに、InstallScript デバッガーのツールバー上の [ステップオーバー] ボタンをクリックします。ユーザー定義関数中のすべてのステートメントが実行されます。実行点は現行ブロックの次のステートメントに移ります。

InstallScript ブレークポイント

ブレークポイントとは、InstallScript デバッガーの制御下でスクリプトを実行した際に、実行が停止されるスクリプト中の位置です。実行を停止させたいステートメント付近の左マージンをクリックして、ブレークポイントを設定できます。一度設定されたブレークポイントは、クリアされるかまたは InstallScript デバッガーが終了するまで有効です。

デバッグにおけるブレークポイントの役割は、スクリプトを詳細に解析したい場所で実行に割り込みをかけることです。スクリプトがブレークポイントまで実行されたら、次に続くステートメントをステップ実行し、スクリプトのそのセクションにとって重要な変数を監視できます。次のブレークポイントまで実行するには、ツールバーの [実行] ボタンをクリックします。

ブレークポイントが設定されたステートメントはその属性とともに表示されます。スクリプトがデバッガーの制御の下で実行される場合、その実行はブレークポイントとして設定されたステートメントの *手前*で停止します。

ソース行付近をクリックしてブレークポイントを設定する

タスク ***ソース行をクリックしてブレークポイントを設定するには、以下の手順に従います:***

1. スクリプト ウィンドウをスクロールして、ブレークポイントを設定する行を表示します。
2. その行付近の左マージンをクリックします。

InstallScript デバッガーは、その行にブレークポイント インジケーター (赤丸) を表示します。

メモ・ *設定済みのブレークポイント (赤丸) をクリックすると、InstallScript デバッガーはそのブレークポイントをクリアします。*

関数でブレークポイントを設定する

タスク ***関数にブレークポイントを設定するには、以下の手順に従います:***

1. スクリプト ウィンドウで、ブレークポイントを設定したい関数の最初の行の左横マージンをクリックします。
2. デバッガー中でスクリプトを実行すると、デバッガーはこのブレークポイントで停止します。

ブレークポイントが設定されると、ブレークポイント自体は関数中のコードの最初の実行可能行に設定されます。

メモ・ *ビルトインの InstallScript 関数にブレークポイントを設定することはできません。*

ブレークポイントまで実行する

次のブレークポイントまで実行するには、InstallScript デバッガーのツールバーにある [実行] ボタンをクリックします。ブレークポイントが設定されていない場合、または最後のブレークポイントに続くステートメントまでがすでに実行されている場合に [実行] ボタンをクリックすると、スクリプトは最後まで実行されます。

ブレークポイントのクリア

スクリプト中のブレークポイントをクリアするには、次の方法のうちの 1 つを使用します。

- 左マージンの赤丸をクリックしてブレークポイントをクリアします。
- **[デバッグ]** メニューから、**[すべてのブレークポイントを削除]** をクリックします。

スクリプトデータの調査、監視、および変更

最も一般的なソフトウェア バグの中には、変数がプログラム中のキーポイントで正しい値を持っていないために発生するものがあります。これは、以下のようなさまざま原因で発生します。

- 変数が初期化されていなかった。
- 変数が更新されなかった。
- 変数に不適切な値が割り当てられた。

この種のエラーは大きな影響を及ぼす可能性があり、その種類もさまざまです。たとえば、ある変数が特定の値を持つ場合にのみスクリプトの一部が実行される場合、この制御変数が正しく設定されていないとインストールは完全には動作しません。*while* ループの制御に変数が使用されていると、ループは実行されないか、または無限に実行されてインストールがハングアップします。

バグを完全に解析するには、スクリプトのトレース時にスクリプトデータの値を検査できなければなりません。そのために、InstallScript デバッガーは以下の 3 つの方法を備えています。

1. 変数を調査して、現在の値を判断します。任意のブレークポイント、またはスクリプトのステップ実行時に、スクリプト中のすべてのグローバル変数の値をチェックできます。実行点がユーザー定義関数内にある場合には、その関数にローカルなすべての変数をチェックすることもできます。
2. 変数を監視してスクリプト実行のトレースのために [ステップ イン] および [ステップ オーバー] コマンドを使用した際の、変数値の変化を観察します。コントロール ウィンドウには、[監視] ウィンドウが含まれ、ここにデバッグ セッションで監視する 1 つまたは複数の変数を挿入できます。
3. 変数の値を変更してプログラム中のキーポイントで解析中のバグにおけるその変数の役割についての論理をテストしたり、またはスクリプト中に発見したロジック エラーの効果をオーバーライドします。

変数の調査

変数を調査するには、変数ウィンドウに調査する変数を配置します。このウィンドウには、変数の現行ローカル値に続いて変数名が詳細に表示されます。

制限

構造化変数およびリスト変数は調査できません。

ローカル ユーザー変数コントロールがアクティブなのは、実行点がユーザー定義関数の begin ステートメントと end ステートメントの間にある場合のみです。実行点が関数の終端を越えて移動した場合には、[監視] ウィンドウに表示されるこの変数の値が < 変数が見つかりません> というメッセージに変更されます。

変数の監視

変数ウィンドウには、現在のユーザー定義関数中のすべてのローカル変数のリストが表示されます。

[監視] ウィンドウにグローバル変数の名前を入力して、その現在の値を見ることができます。

制限

- 構造化変数は、**[監視]** ウィンドウでは選択できません。構造体のメンバーの値を見るには、スクリプト エディターに戻り、スクリプト中のその値を知りたい場所にメンバーの値を単純なデータ型に割り当てるス

テートメントを追加します。その後、スクリプトを再コンパイルしてデバッガーを開始します。最後に、その単純な変数を **[監視]** ウィンドウに追加します。

- リスト変数は、**ウォッチ** ウィンドウで選択できますが、デバッガーにはその要素は表示されません。リストの要素を見るには、構造体のメンバーに対する上記の方法を使用します。
- ローカル ユーザー変数コントロールがアクティブなのは、実行点がユーザー定義関数の begin ステートメントと end ステートメントの間にある場合のみです。実行点が関数の終端を越えて移動した場合には、**[監視]** ウィンドウに表示されるこの変数の値が < 変数が見つかりません> というメッセージに変更されます。

変数の値の変更

InstallScript デバッガーでは、変数ウィンドウに変数のリストが表示されます。

リスト中の変更したい変数をクリックします。変数に変更したい値を入力します。

ビルトイン関数からの戻り値を監視する

ウォッチ ウィンドウで、LAST_RESULT と入力します。このシステム変数には InstallScript 関数への直近の呼び出しによる戻り値が保持されています。

[監視] ウィンドウから変数を削除する

タスク ***ウォッチ ウィンドウから変数を削除するには、以下の手順に従います:***

1. ウォッチ ウィンドウで、削除する変数をクリックします。
2. **Delete** キーを押す。

無限ループに入ったスクリプトの停止

無限ループに入ったスクリプトを停止するには、以下の手順に従います:

1. インストールのウィンドウから InstallScript デバッガーのウィンドウにフォーカスを変更するには、ALT+TAB キーを押します。
2. デバッガーのウィンドウで **[停止]** ボタンをクリックします。

構文エラー

構文エラーとは、キーワードのスペルミス、宣言されていない変数の参照、ステートメントの末尾にセミコロンがないなどといった言語の使用上のエラーです。構文エラーがあるとスクリプトのコンパイルができず、コンパイラによって通知されます。デバッガーには構文エラーを検出する機能はありません。プログラムは、構文エラーをなくして正常にコンパイルした後、デバッガーの制御下で実行できます。

ロジック エラー

ロジック エラーは、スクリプトのアルゴリズム (スクリプトの構造) 中に存在します。ロジック エラーはいわゆるランタイムエラーを生成します。ロジック エラーが現れるのはスクリプトの実行時だけだからです。ロジック エラーの中には、スクリプトを終了させるようなランタイムエラーを生成するものもあります。そうでない場合は、単にスクリプトの異常動作や、入力に対する応答の停止を発生させます。デバッガーはこの種のエラーの解析に役立ちます。

任意のコンピューターでのインストールのデバッグ

特定のハードウェアおよびソフトウェア構成でのみ発生するバグを見つけるために、開発マシン以外のシステム上でインストールをデバッグする必要がある場合があります。その場合、そのコンピューター上に InstallShield をインストールする必要はありません。

タスク

インストール開発マシンではなく、デバッグ マシン上でインストールをデバッグするには、以下の手順に従います:

1. 開発マシン上でインストールをコンパイルおよびビルドします。
2. 開発マシンからデバッグ マシンへ、以下のファイルをコピーします。
   - *InstallShield Program Files フォルダー*¥System¥ISDbg.exe

     これは、InstallScript デバッガーの実行可能ファイルです。
   - *InstallShield Program Files オルダー*¥System¥SciLexer.dll

     デバッグ マシン上の ISDbg.exe ファイルと同じフォルダ内に、このファイルが必要です。
   - *InstallShield Program Files オルダー*¥Program¥0409¥ISDbg.chm (InstallShield 英語版の場合) または *InstallShield Program Files オルダー*¥Program¥0411¥ISDbg.chm (InstallShield 日本語版の場合)

     これはヘルプ ファイルで、オプションとしてこれをデバッグ マシンにコピーすると、デバッガー内部から InstallScript デバッガー ヘルプにアクセスできます。
3. ISDbg.exe ファイルに /REGSERVER コマンドライン パラメーターに渡して、ISDbg.exe ファイルを登録します。
4. プロジェクト ファイルとプロジェクト フォルダーをデバッグ マシンにコピーします。これには、ビルド済みのセットアップのディスク イメージ (Setup.exe と Disk Images¥Disk*N* フォルダー中のメディア ファイルとサポート ファイル) が含まれます。

   ***Tip** - 開発マシン上のプロジェクト ファイルがネットワークを介してデバッグ マシンからアクセスできる場合は、ステップ 4 をスキップすることができます。*
5. デバッグ マシンに Visual Studio 2012 がインストールされていない場合、Visual C++ Redistributable for Visual Studio 2012 をインストールする必要があります。

Note *• Visual C++ 再配布可能パッケージは、Visual Studio 2012 を使って開発されたアプリケーション (InstallShield など) を実行するために必要な Visual C++ ライブラリのランタイム コンポーネントをインストールします。*

6. インストールをデバッグし、デバッグ シンボル (.dbg) ファイルの位置を提供するために、/d コマンドライン パラメーターを使用して Setup.exe を起動します。たとえば、Setup.dbg が C:¥Test にある場合、次のステートメントをコマンドラインで入力します:

   setup /d"C:¥Test"

   開発システムでインストール ファイル、スクリプト ファイル、およびデバッグ ファイルがすべて元の場所にある場合、デバッグ ファイルへのパスを指定する必要がありません。代わりに、次を使用します。

   setup /d

インストール スクリプトから呼び出された DLL 関数のデバッグ

タスク ***セットアップ スクリプトから呼び出された DDL 関数 のデバッグを行うには、以下の手順に従います:***

1. DLL を含む Microsoft Visual Studio プロジェクトを開きます。
2. 実行中のセットアップがあればすべてをシャットダウンします。また、Windows タスク マネージャーを使って実行中の Setup.exe がないことを確認します。Setup.exe が実行中の場合は終了させます。
3. DLL プロジェクトの [プロジェクトの設定] にある [デバッグ] タブで、"デバッグ セッションの実行可能ファイル" フィールドにセットアップの Setup.exe の場所 (<プロジェクト フォルダー>¥Media¥<メディア名>¥Disk Images¥Disk1¥Setup.exe) を指定します。
4. "プログラム引数" フィールドに以下を指定します。

   –deleter –d

5. **F5** キーを押して Setup.exe を起動し、デバッグを開始します。デバッガーが Setup.exe を起動します。セットアップが初期化して実行します。デバッガーは DLL 関数で設定された任意のブレークポイントで停止します。

識別されたエラーの訂正

InstallScript デバッガーはエラーの特定もコードの編集も行いません。デバッガーで実行できることはコンパイラが見落としたエラーを発見することのみであり、実際にコードを変更できるのはエディターだけです。

デバッガーの実行時にスクリプト エディターを開くことができます。スクリプト エディターにはデバッガーのコード ウィンドウと同じファイル (通常、Setup.rul) を表示できます。デバッガーを中止してコードを編集する場合には、それを IDE 中で行う必要があります。別のファイルが表示された場合、またはそのウィンドウが開いていない場合は、エラーを含むスクリプトを編集するために、それをセットアップする必要があります。

スクリプトに必要な変更を加えた後、以下の手順を実行します。

- スクリプトを再コンパイル します ([ビルド] メニューから [Setup.rul のコンパイル] を選択します)。

- セットアップを再ビルドします。
- スクリプトをデバッグします。
- 必要な場合は、ブレークポイントと変数を変更に対応するように調整します。

修正済みインストールのテスト

修正されたスクリプトをテストし、報告されたバグがなくなってセットアップが意図した通りに動作することを確認できたところでデバッグ作業が終了します。これを行なうには、バグを確認したときと全く同様にスクリプトを実行します。修正したことによって新しいバグが発生していないことを確認してください。

InstallScript デバッガーのトラブル シューティング

[ビルド] メニューで [InstallScript のデバッグ] を選択したときにデバッガーが表示されない場合は、次のファイルの存在を確認してください:

Table 2 - ファイルと説明

ファイル	場所
ISDbg.exe	*InstallShield Program Files フォルダー*¥System このファイルは登録の必要があります。まだ登録されていない場合は、/REGSERVER コマンドライン パラメーターを使用して登録してください。
SciLexer.dll	*InstallShield Program Files フォルダー*¥System
Setup.dbg	*InstallShield Project Folder*¥Project Name¥Script Files
Setup.rul	*InstallShield Project Folder*¥Project Name¥Script Files
Setup.inx	*InstallShield Project Folder*¥Project Name¥Script Files
Setup.exe	*InstallShield Project フォルダー*¥プロジェクト名¥*製品構成*¥*リリース名*¥DiskImages¥Disk1

ISDbg.exe と SciLexer.dll は、InstallShield と一緒にインストールされています。Setup.rul は、プロジェクト用にユーザーが作成したプライマリ スクリプトです。Setup.exe と Setup.inx は、ビルド プロセスの一部として作成されます。

カスタム ダイアログの表示

カスタム ダイアログが表示できないという問題は、テクニカル サポートによく寄せられます。カスタム ダイアログを作成してデバッグすることは複雑なプロセスになる可能性がありますが、以下のすべての条件を満たしているかどうかを確認することにより、しばしばカスタム ダイアログの表示を妨害しているバグを発見できます。

- 出荷しようとしているディスク上にダイアログを含む DLL が存在している。
- インストールは、DLL の場所としてスクリプト中に書かれたパスおよびサブフォルダー中にその DLL をコピーする。
- ロードするダイアログが DLL 中に存在する。
- ダイアログが DLL 中にある場合に、ダイアログの呼び出しに正しい ID を使用している。

プログラム フォルダーとショートカットの作成

プログラム フォルダーおよびショートカットの作成が困難な場合は、スクリプト中のそれらを作成するセクションを分離して別々に実行します。

SprintfBox 関数を使用して **AddFolderIcon** 関数のパラメーターを表示します。パラメーターに渡されるすべての値が有効であり正しい順番であることを確認してください。

ディスクの交換

ユーザーにディスクを抜いて次のディスクを挿入するように促すプロンプトを表示する際に、スクリプトに問題が発生する場合があります。たとえば、ユーザーに Disk 1 を取り外して Disk 2 を挿入してもらうところで「ファイルが見つかりません。」というエラーメッセージが表示される可能性があります。

通常、このような問題はディスク上のファイルを開いたままにしているために起こります。DOS でこのエラーメッセージが起こるのは、新しいディスク ボリューム ID、またはファイル アロケーション テーブルを見つけて、開いたファイルにアクセスできない場合です。

この問題を避けるため、セットアップ時にディスク上のファイルを直接*開かない*でください。情報ファイル、ビットマップ ファイル、DLL をディスクから直接使用しないでください。その代わり、ターゲット ハードディスクにファイルをコピーし、ハードディスクからそれらにアクセスするようにしてください。

デバッグ中にこの問題が発生した場合は、ターゲット ディスクにファイルを転送する前に開いたファイルがないことを確認してください。

お問い合わせの前に

ユーザーに何が異常なのかを正確に報告してもらい、問題を明確化してください。この問題はたくさんの顧客から報告されているものなのか、または数人の顧客またはこの顧客からのみ報告されている問題なのか。以下は、顧客の問題を診断する際に、確認できるいくつかの事項です。

- ユーザーはソフトウェアのインストールを何度か試みましたか？そうでない場合は、同じシステム上にソフトウェアを再度インストールしてもらってください。それでもまだ問題が起きる場合は、別のシステム、またはできれば複数のシステム上にインストールしてもらってください。1 つのシステムで使用しただけでは結論を下すのに十分とは言えません。
- 問題は配布メディアに関係していますか？複数の異なるメディア タイプで出荷する場合、ユーザーに別のメディア タイプを試してもらいます。これは、ディスクの問題かドライブの問題かを判別するのに役立つ場合があります。両方のメディアで同じ問題が起きるならば、問題をかなり減少したことになります。セットアップが正常であることが確実である場合は、ユーザーに別のディスクを送ってください。
- 可能場合は、有名ブランドの配布メディアを使用してください。以前にディスクの問題を経験したことがあれば、ディスクの購入費を切り詰めることが割りに合わないことをご存知でしょう。高品質のディスクは、

なんとか使用できるレベルのディスクよりもたった 10 円か 15 円高いだけですが、顧客を満足させることと苛立たせることとの間には大きな差があります。

- 顧客の問題がある特定のシステムでしか起こらないことがわかった場合は、そのシステムと他のシステムとの違いを見つける必要があります。
- 顧客にメモリとシステム リソースの空き容量を尋ねてください。システムリソースが極端に低い場合、他のプログラムがそれを使用している可能性があります。ユーザーに Windows を終了してシステム リソースを解放してもらってください。Windows の再起動後、十分なメモリとシステム リソースの空きがある場合は、ユーザーにセットアップ プログラムを再度実行してもらってください。
- ユーザーが実行している Windows のバージョンを判別します。通常 Setup.exe にはシステム上に最低 16 MB の RAM が必要です。ただし、大きなスクリプトやビットマップを使用している場合、それ以上の RAM が必要である場合もあります。顧客のコンピューターがこの条件を満たさない場合、より多くの RAM が必要です。
- Config.sys ファイルと Autoexec.bat ファイルを調べます。顧客に上記ファイルのコピーを送ってもらってください。Config.sys に多くの項目があり、多くのプログラムを特定の場所にロードしている場合、これが問題を引き起こす場合があります。特定のメモリ マネージャーや多くのパラメーターを持つドライバーが問題の原因である可能性もあります。最も基本的なドライバーだけを使用してください。Config.sys からできるだけ多くのドライバーを削除するように、ユーザーに指示してください。一般的に、Config.sys 中の行数が増えるほど、競合の可能性は大きくなります。すべての不要なステートメントを一時的にコメントにし (rem)、システムを再起動してもらってください。

ディスプレイ ドライバー

ディスプレイ ドライバーは、Windows がハードウェア ビデオ アダプタと通信できるようにするソフトウェア インタフェースです。Windows には任意のビデオ アダプタに対応した標準ディスプレイ ドライバーがいくつか装備されています。ただし、多くの Windows ユーザーはディスプレイ アダプタの製造元から提供されるカスタム ドライバーをコンピューターにインストールしています。これらのカスタム ドライバーは標準 Windows ディスプレイ ドライバーよりも多くの色と高い解像度を提供します。

顧客がセットアップを正しく*開始*できない場合、カスタム ディスプレイ ドライバーがその原因であることがよくあります。以下のような症状は、通常、問題がディスプレイ ドライバーにあることを示すものです。

- セットアップは開始するが、画面に表示されない。
- セットアップがビットマップ (.bmp) ファイルを表示しようとするとシステムがハングアップする。
- セットアップがビットマップをフェードインで表示しようとすると画面に異常が発生する。

顧客から上記のような現象が報告された場合は、顧客のシステムにインストールされているビデオ ドライバーを調べてください。標準の Microsoft Windows VGA ドライバーをインストールして、再度セットアップを実行してもらってください。必要な場合は、ソフトウェアのインストール後、標準でないドライバーに戻すことはいつでも可能です。

ウィルス対策プログラム

実行ファイルがハードディスクにコピーされないように設定されたウィルス対策プログラムは、インストールに支障をきたします。インストールは正常に実行されているように見えるのに、アプリケーションがハードディスクにコピーされないという報告をエンドユーザーから受けた場合は、ウィルス防止が問題である可能性があります。

大抵の場合、ウィルス対策プログラムのデフォルト設定は、ファイルのコピーをブロックしないようになっています。その場合は、ソフトウェアのインストールに支障はありません。ファイルのコピーをブロックするようにウィルス対策プログラムを設定しているユーザーのために、問題が起きる可能性があることを指摘した注意をユーザー ガイドや Readme ファイルに含めることをお勧めします。

この問題を解決するのは簡単です。ウィルス対策プログラムを無効にして、その後再度インストールを実行します。

ターゲット ドライブ

設計の品質やテストが完全であるにもかかわらず、エンド ユーザーのコンピューターのターゲット ドライブ上の障害のために、インストールの実行が失敗する場合もあります。磁気メディアでは、以下のようにさまざまな問題が発生する可能性があります。

- ディスク表面の傷や損傷によって、特定のディスク セクタに対してファイルの読み書きができない場合があります。
- クラスタの損失やクロスリンクのようなファイル アロケーション テーブルのエラーによって、インストールがファイルを開くまたは上書きできない場合があります。
- 極端なディスク フラグメンテーションによってディスクのパフォーマンスが低減し、インストールやインストールされたアプリケーションの実行が非常に遅くなる場合があります。

インストールやアプリケーションに関するエンド ユーザーの問題の原因がターゲット ディスクであると考えられる場合は、次の方法を推奨してください。

1. 問題の原因がディスク表面の傷やファイル アロケーション テーブルのエラーであると考えられる場合には、エンドユーザーにスキャン ディスクを実行してもらってください。このユーティリティは不良セクターをロックアウトし、プログラムがそのセクターに対して読み書きを行なわないようにします。また、ファイル アロケーション テーブルのエラーも修復します。
2. 問題の原因が極端なディスク フラグメンテーションであると考えられる場合は、エンドユーザーに Windows のデフラグを実行してもらってください。

Tip・デフラグの実行前にディスクのバックアップを行うように、エンドユーザーに助言することをお勧めします。

索引

た

て

ひ

ふ

へ

ほ

ゆ

ろ