InstallScript サンプル

注）このドキュメントは、*InstallShield 2015 Premier Edition* を基に作成しています。*InstallShield 2015* 以外のバージョンでは設定名などが異なる場合もあります。

概 要

InstallShield では、InstallShield 独自のプログラム言語 InstallScript を利用して、動作をカスタマイズすることができます。
InstallScript プロジェクトでは、インストールプログラムはプロジェクトの InstallScript コードによって制御されます。プロジェクトのスクリプトを変更して、インストールやアンインストール時に様々なタスクを実行することができます。

また、基本の MSI プロジェクトでは、インストールの制御は Windows Installer エンジンによって行われますが、InstallScript で定義した関数をカスタムアクションとして呼び出すことができます。

この記事では、「InstallScript プロジェクト」および「基本の MSI プロジェクト」での基本的な InstallScript の使用方法と、サンプルスクリプトを紹介します。

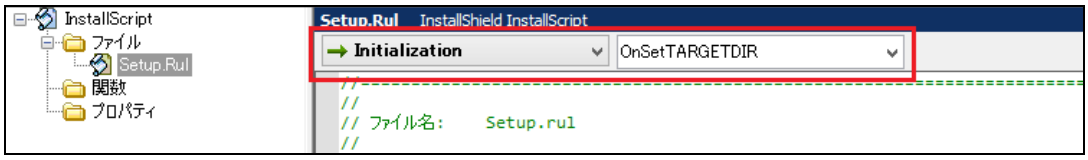
＜紹介するサンプル＞

A. InstallScript プロジェクト

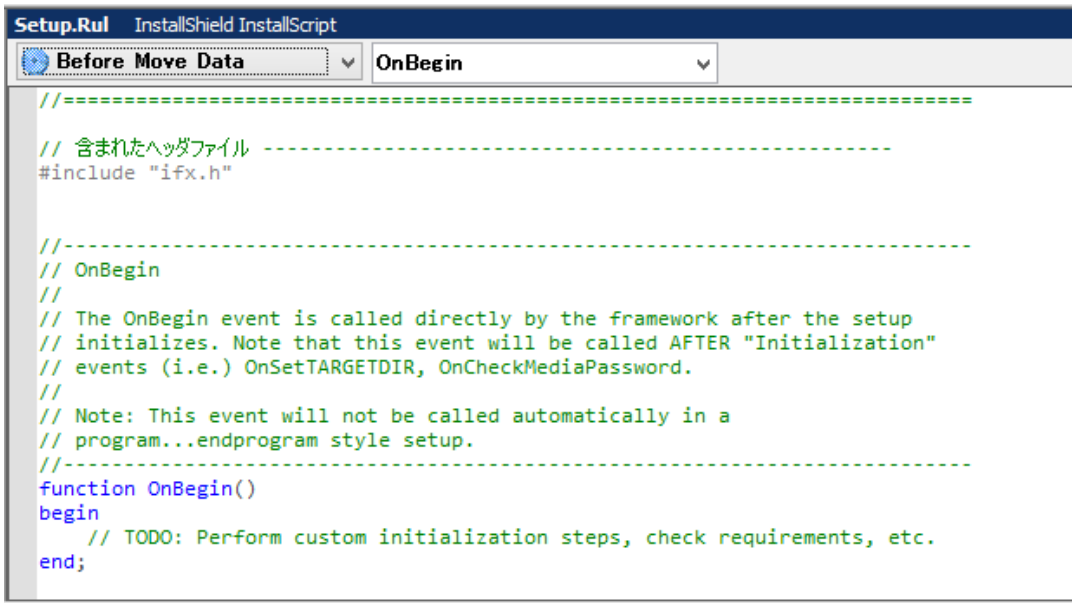
InstallScript インストールは InstallScript エンジンによって制御され、決められた順序で一連のイベントが生成されます。これらのイベントは、インストールを実行するソフトウェアハンドラーをトリガーします。たとえば、インストールがロードされた直後に Begin というイベントが生成されます。このイベントは、OnBegin というイベント ハンドラーの実行をトリガーします。ユーザーのマシンが製品のシステム要件と一致しているかどうかを判定する処理などを組み込むのに適しています。

イベントハンドラーは常に決められた順序で呼び出されます。イベントハンドラーはインストレーションの種類 (通常インストール、メンテナンスインストール、管理インストール、またはパッチインストール)に従って呼び出されます。
イベントハンドラーの詳細については、製品ヘルプをご参照ください。

InstallScript の編集は、［動作とロジック］ – ［InstallScript］ビューで行います。
［InstallScript］ビューの右ペインの上部にあるスクリプトツールバーを使って、イベントハンドラー関数ブロックを張り付けることができます。編集された内容は、インストールイベントに関連付けられているデフォルトのアクションを上書きするため、ここでアクションの内容を変更することができます。



イベントカテゴリ（左のリストボックス）でカテゴリを選択して、イベント（右のリストボックス）でイベントを選択します。
例えば、［Before Move Data］ –[OnBegin]と選択すると、OnBegin イベントが setup.rul ファイルに追加されます。



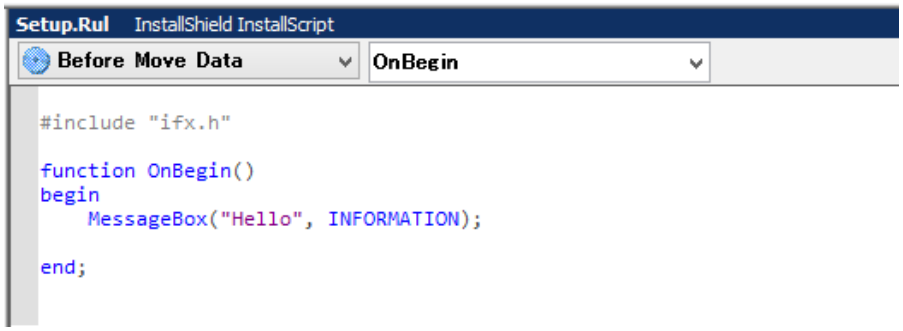
すべてのスクリプトは、以下の文で始まります。ヘッダファイル ifx.h は、すべての InstallScript プログラムで使用される関数を定義しています。

```
#include"ifx.h"
```

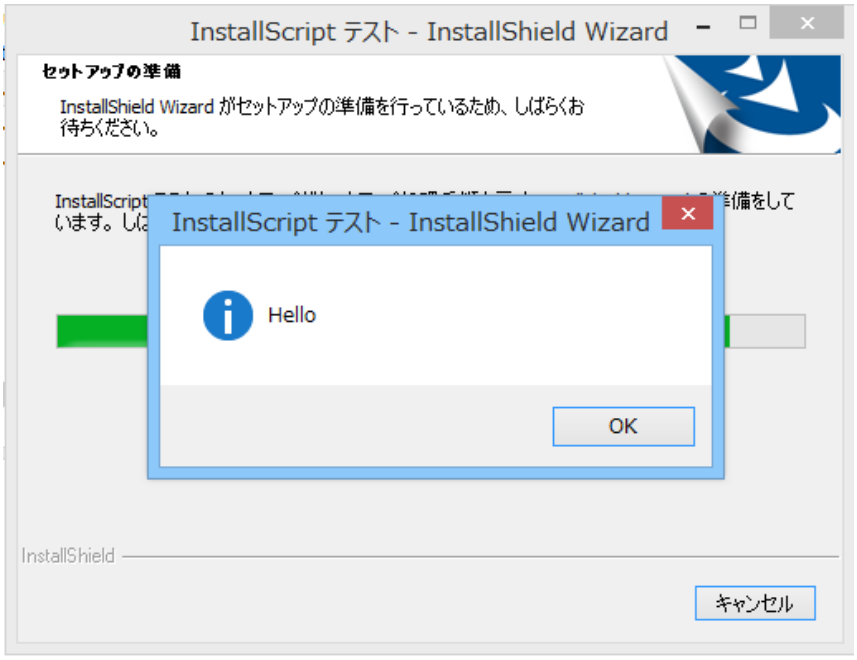
一般的なコードの記述方法は以下のとおりです。

```
#include"ifx.h"

function OnBegin()
  // ローカル変数
begin
  //コード
end;
```

たとえば、インストールを続行する前にメッセージを表示する場合、OnBegin は次のように記述できます。



ビルドしてインストーラーを実行すると、インストールの初めにメッセージボックスが表示されます。



また、ユーザー定義関数をイベントから呼び出すこともできます。ユーザー定義関数を作成するには、prototype で関数宣言します。

```
#include"ifx.h"

//prototype 宣言
export prototype MyFunction(STRING);

function OnBegin()
begin
  //MyFunction の呼び出し
  MyFunction("MyFunction を呼び出しました");
end;


//メッセージボックスを表示するユーザー定義関数を作成
function MyFunction(svMsg)
begin
  MessageBox(svMsg,INFORMATION);
end;
```

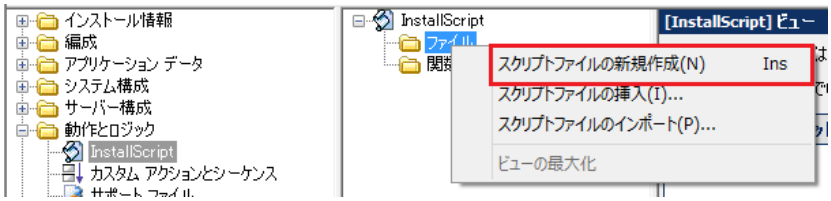
B. 基本の MSI プロジェクト（InstallScript カスタムアクション）

基本の MSI プロジェクトでは、カスタムアクションとして InstallScript 関数を呼び出して実行することができます。すべての InstallScript カスタム アクションは、スクリプトへのエントリポイントとして、エクスポートされたユーザー定義関数が必要です。

InstallScript カスタムアクションを実行するには、以下の 2 つの手順が必要となります。

- InstallScript コードでユーザー定義関数の作成
- InstallScript カスタムアクションの作成と実行スケジュールの設定

InstallScript のコードの作成は、［動作とロジック］ – ［InstallScript］ビューで行います。
［ファイル］アイコンを右クリックして、［スクリプトファイルの新規作成］を選択すると、Setup.rul ファイルが追加されます。



カスタムアクションで呼び出せる関数の引数は 1 つだけで、引数は .msi データベースへのハンドルでなければなりません。
一般的なコードの記述方法は以下のとおりです。

```
#include"ifx.h"
export prototype MyFunction(HWND);

function MyFunction(hMSI)
    // ローカル変数
begin
    //コード
end;
```

※InstallScript MSI プロジェクトでもカスタムアクションが使用できますが、その場合 MSI へのハンドルは「hMSI」の代わりに「ISMSI_HANDLE」を使用します。

たとえば、メッセージを表示する場合、MyFunction 関数は次のように記述できます。

Setup.rul InstallShield InstallScript

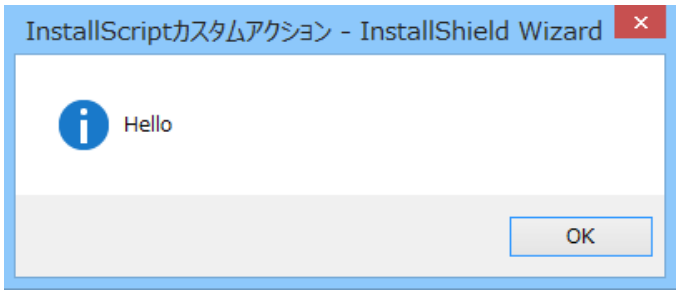
```
////////////////////////////////////////////////////////////////////////////

#include "ifx.h"

export prototype MyFunction(HWND);


function MyFunction(hMSI)

begin

    MessageBox("Hello",INFORMATION);

end;
```

次に、MyFunction 関数を呼び出すカスタムアクションを作成します。［動作とロジック］ – ［カスタム アクションとシーケンス］ビューで行います。

［カスタムアクション］アイコンを右クリックして、［新しい InstallScript］を選択し、カスタムアクション名（例えば、「caMyFunction」）を設定します。
作成されたカスタムアクションで、InstallScript で作成したユーザー定義関数名（MyFunction）を指定し、カスタムアクションを実行するタイミングを指定します。

以下の画面では、「インストール UI シーケンス」の「LaunchConditions の後」にスケジュールしています。実行条件としては、初回インストール時だけ実行させるように「not Installed」を指定しています。

インストーラーを実行すると、インストールの初めにメッセージボックスが表示されます。



C. サンプルスクリプト

ここではいくつかの処理を行うサンプルスクリプトを紹介いたします。
サンプルは、InstallScript プロジェクトもしくは基本のMSIプロジェクト（InstallScript カスタムアクション）用のいずれかで作成しています。内部のコードは共通で使用できるものもありますので、それぞれのプロジェクトタイプに合わせて修正してお使いください。

■システム要件を確認する（InstallScript プロジェクトのみ）

InstallScript プロジェクトでは、ターゲットシステムが製品のシステム要件を満たすかどうかを調べる関数および構造体が用意されています。
OnBeginイベントハンドラーは、1回目のインストールとメンテナンスインストールのいずれの場合にも呼び出されるため、初回インストール時のみシステム要件を確認する場合は、次のように、システム変数 MAINTENANCE を使用するコードを if 文の中に入れます。

```
#include"ifx.h"

function OnBegin( )
begin

  // Windows Vista 以降のクライアント OS 以外で実行された場合、インストールを中断

  if (!MAINTENANCE) then
    if (!(SYSINFO.nWinMajor >=6
            && SYSINFO.nOSProductType = VER_NT_WORKSTATION)) then

        MessageBox(
        "このプログラムには Windows Vista 以降のクライアント OS が必要です。¥n¥n" +
        "セットアップを中止します。",
        SEVERE);
        abort;

    endif;

  endif;

end;
```

※セットアップの初期設定中に、インストールによって SYSINFO 構造体のメンバーが設定されます。オペレーティングシステム、オペレーティングシステムのメジャーおよびマイナーバージョン、システムが64ビットかどうかなど様々な判定に利用できます。
詳細については、製品ヘルプ「SYSINFO」をご参照ください。

■特定のディレクトリーがユーザーのシステムに存在するか確認する（InstallScript プロジェクト）

Is関数の定数「PATH_EXISTS」や「FILE_EXISTS」を使用することで、ディレクトリーやファイルの存在有無を確認することができます。

以下は、特定のディレクトリーが存在するかどうかを確認するサンプルです。

```
#include"ifx.h"

OnBegin()

  STRING szDirectoryPath;
  BOOL bResult;

begin

  if (!MAINTENANCE) then

    szDirectoryPath = "C:¥¥testDirectory¥¥testSubdirectory";        //検索するフォルダー
    bResult = FALSE;        //初期化

    //フォルダパスが存在するか確認します
    bResult = Is( PATH_EXISTS, szDirectoryPath );

    //結果にしたがって、適切なメッセージを表示します
    if ( bResult = TRUE ) then
      MessageBox("ディレクトリーが存在します", INFORMATION );
    else
      MessageBox("ディレクトリーは存在しません", INFORMATION );
    endif;

  endif;

end;
```

■ターゲットシステムからレジストリの情報を取得する（InstallScript プロジェクト）

RegDBGetKeyValueEx 関数を使用すると、ターゲットシステムのレジストリ情報を取得することができます。また、RegDBSetKeyValueEx 関数ではレジストリに値をセットすることができます。

以下は、「HKEY_LOCAL_MACHINE¥SOFTWARE¥MyCompany¥TestApp」の「Test」の値を取得し、新たに”Test2”という値をセットするサンプルです。

```
#include"ifx.h"

OnBegin()

  STRING      szKey, szName, svValue;
  NUMBER      nType, nSize;
  NUMBER      nResult;

begin
  if (!MAINTENANCE) then

    //HKEY_LOCAL_MACHINE¥SOFTWARE¥MyCompany¥TestApp の Test 取得のため
    //レジストリのパラメータを設定
    szKey = "SOFTWARE¥¥MyCompany¥¥TestApp";
    szName = "Test";
    nType = REGDB_STRING;
    nSize=-1;

    //レジストリ関数が利用するルートキーを設定
    RegDBSetDefaultRoot(HKEY_LOCAL_MACHINE);

   //値の取得
    if(RegDBGetKeyValueEx ( szKey, szName, nType, svValue, nSize )< 0) then

      MessageBox ("RegDBGetKeyValueEx が失敗しました。”, SEVERE);

    else

      MessageBox ("Test の値は、” + svValue +”です。”, SEVERE);

    endif;

    //HKEY_LOCAL_MACHINE¥SOFTWARE¥MyCompany¥TestApp の Test に”Test2”をセット
    RegDBSetKeyValueEx(szKey, szName, nType,"Test2",nSize);

  endif;

end;
```

■インストール時にバッチファイルを実行する（InstallScript）

InstallScript プロジェクトでバッチファイルを呼び出す場合、LaunchAppAndWait 関数などが利用できます。実行するバッチファイルがターゲットシステムにインストールの必要がなく、インストール処理中のみ使用するファイルの場合、サポートファイルとしてインストーラーに含めます。
InstallScript からは、SUPPORTDIR システム変数にてサポートファイルへのパスを取得することができます。

以下の例は、サポートファイルに含めたバッチファイルをインストール完了時に実行する方法です。

```
#include"ifx.h"

function OnEnd()
  STRING szProgram,szCmdLine;
  NUMBER nOptions,nResult;

begin

  if (!MAINTENANCE) then

    // バッチファイル実行のパラメータを各種設定します
    szProgram = SUPPORTDIR^"sample.bat";
    szCmdLine = "";
    nOptions = LAAW_OPTION_WAIT;

    // LaunchAppAndWait 関数によって外部ファイルを実行
    nResult = LaunchAppAndWait ( szProgram, szCmdLine, nOptions );

    // 実行後に実行結果と外部ファイルの戻り値を戻り値表示
    SprintfBox( INFORMATION,"実行結果",
                "ファイルの戻り値 = %d",
                LAAW_PARAMETERS.nLaunchResult);

  endif;

end;
```

※ 上記サンプルは、サポートファイルに Sample.bat というファイルを含めている例です。

■テキストファイルの値を読み出す（InstallScript プロジェクト）

テキストファイルの内容を読み出すために、ListCreate 関数で文字列リストを作成して、ListReadFromFile 関数によりファイルを文字列リストへ読み込みます。

以下は、1行ずつリストから情報を取得して、メッセージボックスから表示するサンプルです。FileInsertLine 関数も使用してファイルの最後に行も追加しています。

```
#include"ifx.h"
#define FILEPATH "C:¥¥test.txt"

function OnBegin()
  STRING svString;
  NUMBER nResult;
  LIST listID;

begin

  //文字列リストの作成
  listID=ListCreate(STRINGLIST);

  //出力結果をリストに読み込み
  if(ListReadFromFile(listID, FILEPATH)<0) then

    MessageBox(FILEPATH + "の読み込みに失敗しました。",SEVERE);

  else

    //リストから最初の文字列を取得します。
    nResult=ListGetFirstString(listID,svString);

    //ループ処理で取得された情報を一行ずつ表示
    while(nResult!=END_OF_LIST)

      //取得情報の表示
      SprintfBox(INFORMATION,"取得情報の表示","%s",svString);

      //リストの次の文字列を取得します。
      nResult=ListGetNextString(listID,svString);

    endwhile;

  endif;

  // メモリからリストを削除します。
  ListDestroy(listID);

  //ファイルの 5 行目にデータを追加します。
  FileInsertLine(FILEPATH,"5 行目の情報です。",4,AFTER);

end;
```

■InstallScript からプロパティの値を取得/設定する（InstallScript カスタムアクションのみ）

InstallScript コードの中では直接 Windows Installer のプロパティにアクセスすることはできません。InstallScript でプロパティにアクセスするには、MSI API の MsiGetProperty と MsiSetProperty を使用します。

MsiSetProperty はプロパティに値をセットする関数で、MsiGetProperty はプロパティの値を取得する関数です。下記サンプルでは、MsiSetProperty で TESTPROP プロパティに Test Value という値をセットしたあと、セットされた値を MsiGetProperty で取り出し、メッセージとして表示しています。

```
#include"ifx.h"

export prototype GetAndSetProperty(HWND);

function GetAndSetProperty(hMSI)
  NUMBER nvSize, nReturnValue;
  STRING svValue;
begin
  nvSize = MAX_PATH;

  // TESTPROP プロパティに値 "Test Value" をセット
  if(MsiSetProperty(hMSI, "TESTPROP", "Test Value") == ERROR_SUCCESS ) then

    //TESTPROP プロパティの値を取得
    if(MsiGetProperty(hMSI, "TESTPROP", svValue, nvSize) == ERROR_SUCCESS ) then

      // TESTPROP プロパティに、"Test Value" がセットされているか確認
      SprintfBox(INFORMATION, "TESTPROP Value", "TESTPROP = %s", svValue);

    endif;

  endif;

end;
```

■カスタムアクションから強制的にエラーを返す（InstallScript カスタムアクション）

InstallScript の処理結果に応じて、カスタムアクションから強制的にエラーを返し、インストールを中断させたい場合があります。InstallScript コード内で、ERROR_INSTALL_FAILURE が返されると、Windows Installer のカスタムアクションはカスタムアクションが失敗したと判定し、インストールを中断することができます。

```
#include"ifx.h"

export prototype ForceFailure(HWND);

function ForceFailure(hMSI)
    STRING szKey, svValue;
    NUMBER nvType, nvSize, nReturn;

begin

  //App Paths キーから、InstallShield がインストールされたフォルダーを取得
  RegDBSetDefaultRoot (HKEY_LOCAL_MACHINE);
  szKey="SOFTWARE¥¥Microsoft¥¥Windows¥¥CurrentVersion¥¥App Paths¥¥isdev.exe";
  nReturn = RegDBGetKeyValueEx (szKey, "Path", nvType, svValue, nvSize);


  if nReturn = 0 then

    //InstallShield のインストール場所を表示
    MessageBox ("InstallShield のインストールされている場所 " + svValue, INFORMATION);
    return ERROR_SUCCESS;

  else

    //インストール場所が不明な場合、ERROR_INSTALL_FAILURE を返してインストールを中断
    MessageBox("InstallShield をインストールした場所が不明です。", SEVERE);
    return ERROR_INSTALL_FAILURE;

  endif;

end;
```

以上