MSI Tip: カスタムアクションによるログファイルへの出力

この文書は Acresso Software の次の文書を元に記載しています。

http://www.acresso.com/webdocuments/PDF/msi_writing_to_the_log_file.pdf

Tip: MSI Tip: Writing to the Log File from a Custom Action

検証したバージョン： InstallShield 2009 Premier Edition

要約

Windows Vista 上で実行される MSI インストーラの要件として、カスタムアクションが成功したか失敗かどうかを MSI Log ファイルに出力する要件が新しく追加されました。 通常の設定で Windows Installer は自動的にカスタムアクションの開始と終了をログに出力しますが、この記事ではより詳細な情報をログファイルに出力する場合の方法について紹介します。

MSI ログファイルの基本

MSI ログファイルは、特定のシステム上で実行されるインストールの多くの情報を提供するテキストファイルです。 MSI ログファイルには以下の情報が含まれます。

- プロパティの最終的な値
- 各アクションの開始情報
- 情報、警告、エラーのメッセージ

MSI ログファイルの作成には、以下のような多数の方法があります。

1. msiexec.exe のコマンドラインパラメータに以下のように「/L」スイッチを使用します。

msiexec /i product.msi /L*v everything.log

「/L」スイッチの後に指定されたパラメータ "*v" は Windows Installer により実行されるすべてのアクションの詳細ログとして出力することを表しています。

MSI ヘルプライブラリにはログに含める情報を制限するための他のスイッチに関する情報が記載されています。

Setup.exe ランチャファイルを使用する場合、**"MSI コマンドライン引数"** に以下のような値を指定することで msiexec.exe に /L スイッチを渡すことが可能です。

/L*v "%TEMP%¥everything.log"

この際、log ファイルのパスに MSI プロパティを使用できないことに注意してください。
MSI プロパティは インストールが初期化されるまで有効になりません。
（上記のコマンドラインは環境変数 %TEMP% を使用しています。）

2. Windows Vista 上の MSI 4.0 では、MsiLogging プロパティ に任意のログスイッチを指定することが可能です。InstallShield 2008 以降 では、[製品のプロパティ] ビューの**[MSI ログの作成]** 設定を使用することで行なうことも可能です。 ログファイルの保存先階層は MsiLogFileLocation プロパティに保存されます。（ このプロパティは読むことはできますが、修正することはできません。 ）

InstallShield の環境でこのスイッチを使用すると、 **"Windows Installer のログを表示"** チェックボックスが SetupCompleteSuccess ダイアログに表示されるようになります。

MSI のヘルプライブラリには、すべての MSI インストーラに対して テンポラリフォルダにランダムな名称のログファイルを作成させる ロギングポリシーの設定方法等の情報が記載されています。

InstallShieldの[ツール]メニューから使用できる **"MSI log File Analyzer"** は MSI ログファイルの作成や、ログファイルに基づいた 様々な カラーコード HTML レポートを作成することができます。

デフォルトで MSI ログファイルには それぞれの標準アクションおよびカスタムアクションの戻り値が記録されます。以下のセクションは、より詳細な情報をログファイルに記録する方法について記載します。

※上記の方法は /L スイッチを指定または MsiLogging プロパティを指定して、ログファイルの作成を指定しない限りは、MSI ログの作成を行ないませんし、ロギングも開始されないことに注意してください。

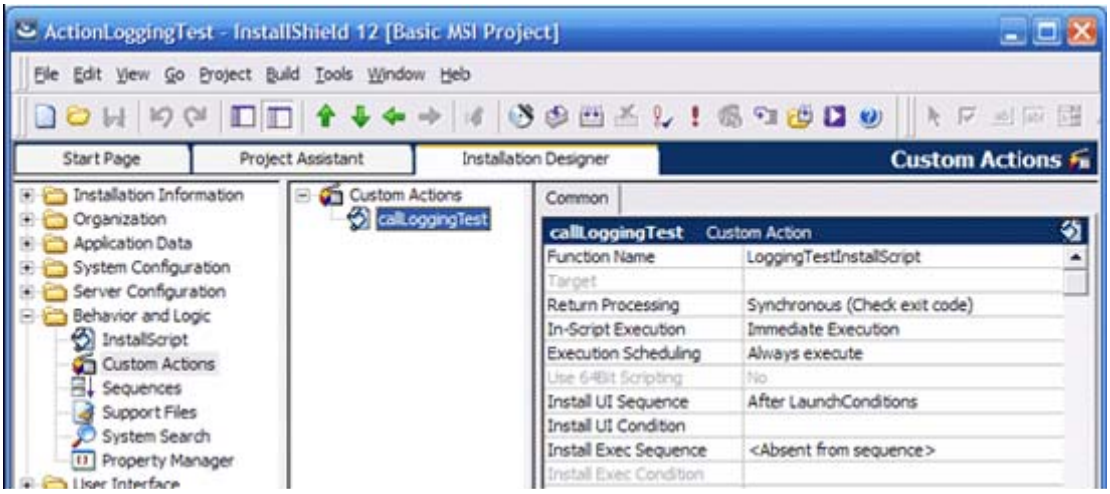
InstallScript カスタムアクションの場合

InstallScript カスタムアクションでは、MSI ログファイルに文字列を出力するために **SprintfMsiLog** 関数を使用します。例えば、以下の InstallScript コードはプロトタイプ宣言をして、LoggingTestInstallScriptと呼ばれる InstallScript のカスタムアクションを定義しています。

LoggingTestInstallScript.

```
#include "ifx.h"
// standard custom action prototype
export prototype LoggingTestInstallScript(HWND);
// custom action definition
function LoggingTestInstallScript(hInstall)
begin
SprintfMsiLog("Calling LoggingTestInstallScript...");
// return success to MSI
return ERROR_SUCCESS;
end;
```

この関数を呼び出すために、作成されたカスタムアクションはシーケンス内にスケジューリングされる必要があります。 この例では、カスタムアクションビューを表示して、LoggingTestInstallScript 関数を呼び出す callLoggingTest と呼ばれる即時実行のカスタムアクションを作成し、LaunchConditions の後に起動するようにスケジューリングします。



パッケージのビルドの後、/L*v スイッチを使用してインストーラを実行すると、ログファイルに以下のような行を確認できます。

InstallShield 25:00:00: Invoking script function LoggingTestInstallScript
1: Calling LoggingTestInstallScript...
InstallShield 25:00:00: CallScriptFunctionFromMsiCA() ends
Action ended 25:00:00: callLoggingTest. Return value 1.

SprintfMsiLog 関数は Sprintf や SprintfBox 機能と似ています。文字列 や 数値変数の値を結合したいとき、メッセージ文字列でプレースホルダ(「%s」か「%d」フィールド、書式指定子と呼ばれる)を入れることができます。

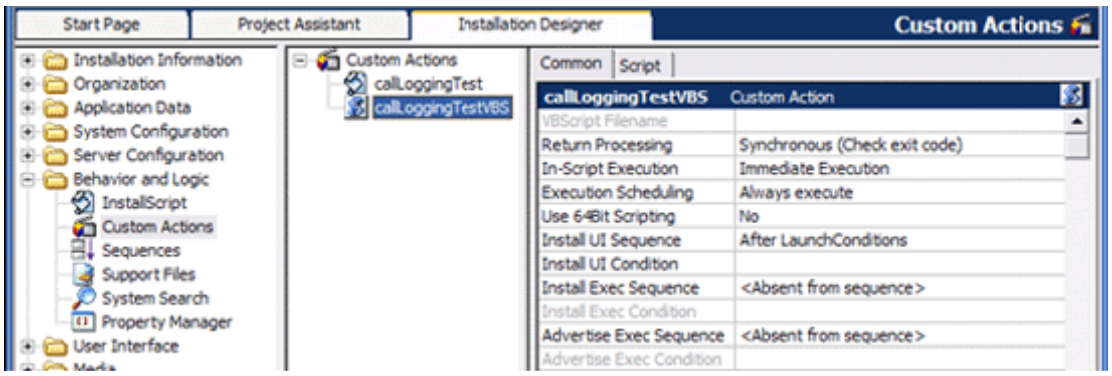
VBScript カスタムアクションの場合

VBScript アクション(後述のセクションで説明される MSI DLL アクションも同様に)で、ログを追加する場合は、一般的に メッセージ情報を含むレコードを作成し、作成したレコードを実行中のインストーラに送信する方法を使用します。
VBScript では、メッセージのレコードを作成するのに Installer.CreateRecord メソッドを使用し、実行中のインストールにレコードを送るのに Session.Message メソッドを使用します。

レコードはプレースホルダとなる文字列 [1] [2] 等を含む [0]番目のテンプレートフィールドから開始します。これらのプレースホルダは 1番目のレコード、2番目のレコード等の値の置き換えが行なわれます。 ログを記録する VBScript アクションを実際に確認するために、以下のコードを記述した callLoggingTestVBS と呼ばれる即時実行の VBScript カスタムアクションを作成して、LaunchConditions の後にスケジューリングしてください。
(この例では、スクリプトが短いので直接カスタムアクションにコードを保存しています。
一般的には 外部の .vbs ファイルを使用することをおすすめします。そうした場合、関数の戻り値を定義でき、カスタムアクションから任意にインストーラを終了させることが可能です。)

```
Const msiMessageTypeInfo = &H04000000
' create the message record
Set msgrec = Installer.CreateRecord(1)
InstallShield Tips & Tricks
' field 0 is the template
msgrec.StringData(0) = "Log: [1]"
' field 1, to be placed in [1] placeholder
msgrec.StringData(1) = "Calling LoggingTestVBS..."
' send message to running installer
Session.Message msiMessageTypeInfo, msgrec
```

カスタムアクションは、以下の図のようになります。



プロジェクトをビルドしてログファイルを作成すると、以下のような行がログに出力されます。

Action 25:00:00: callLoggingTestVBS.
Action start 25:00:00: callLoggingTestVBS.
[...省略...]
Log: Calling LoggingTestVBS...
Action ended 25:00:00: callLoggingTestVBS. Return value 0.

MSI DLL カスタムアクションの場合

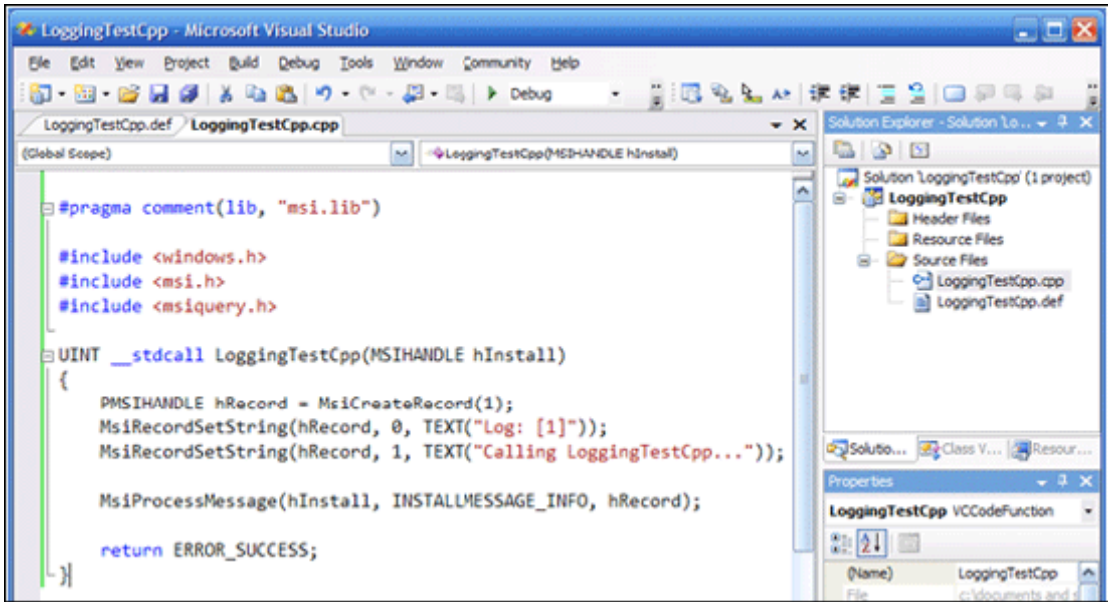
C や C++ で記述された MSI DLL カスタムアクションは、メッセージのレコードを作成する際に MsiCreateRecord 関数を使用する点、実行中のインストーラにレコードを渡す際にために MsiProcessMessage 関数を使用する点を除けば、VBScript コードでログファイルに記録するプロセスに似ています。 例えば、Visual Studio で C やC++ DLL のプロジェクトを作成した場合の C++コードは 以下のようになります。

```
#pragma comment(lib, "msi.lib")
#include <Windows.h>
#include <msi.h>
#include <msiquery.h>
// standard MSI DLL custom action signature
UINT __stdcall LoggingTestCpp(MSIHANDLE hInstall)
{
   PMSIHANDLE hRecord = MsiCreateRecord(1);
   // field 0 is the template
   MsiRecordSetString(hRecord, 0, "Log: [1]");
   // field 1, to be placed in [1] placeholder
   MsiRecordSetString(hRecord, 1, "Calling LoggingTestCpp...");
   // send message to running installer
   MsiProcessMessage(hInstall, INSTALLMESSAGE_INFO, hRecord);
   return ERROR_SUCCESS;
}
```

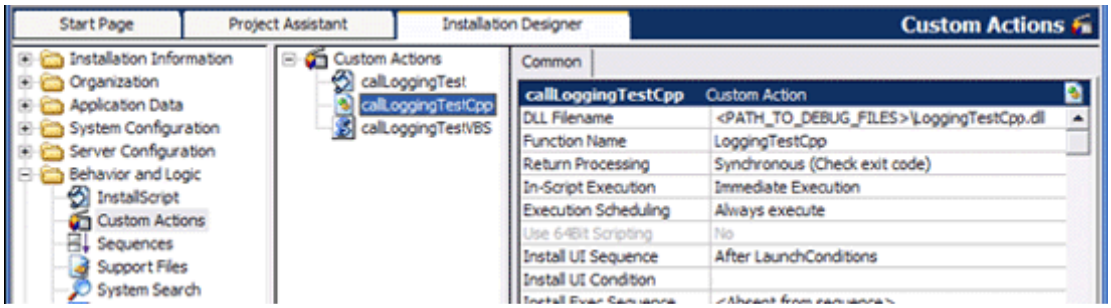
関数名がDLLから適切にエクスポートされるように、以下の.defファイルを作成します。

LIBRARY "LoggingTestCpp" ; *DLL name*
EXPORTS ; *exported function names*
LoggingTestCpp

Visual Studio でDLL プロジェクトは、以下のようになります。



DLL のビルド後、InstallShield で例えば callLoggingTestCpp という名前のMSI DLLのカスタムアクションを作成し、再び LaunchConditions の後に即時実行としてスケジューリングします。



プロジェクトを再ビルドして、ログオプションのスイッチをつけて実行すると、ログファイルには以下のような行が出力されます。

After rebuilding the project and running the MSI with the logging , lines similar to the following should appear in the log file.
Action 25:00:00: callLoggingTestCpp.
Action start 25:00:00: callLoggingTestCpp.
[...省略...]
Log: Calling LoggingTestCpp...
Action ended 25:00:00: callLoggingTestCpp. Return value 1.

詳しい情報は、MSI help library の ″Windows Installer Logging″ トピックを参照してください。