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

By Robert Dickau
Principal Technical Training Writer
Flexera Software

Abstract

A new requirement with Windows Vista is that custom actions write success or failure information to an MSI log file. Windows Installer automatically writes startup and shutdown information for custom actions into the log; this article describes different techniques for writing more detailed information into the log file.

MSI Log File Basics

An MSI log file is a text file that provides a great deal of information about an installation program during a given run on a particular system, including:

- Final property values
- Startup information for actions
- Status, warning, and error messages

There are several ways to create an MSI log file, including:

1. Use the /L switch to msiexec.exe, as in this command:

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

   The characters "*v" after the /L switch indicate to perform verbose logging of every action taken by Windows Installer. The MSI help library describes the other switches you can use to limit the information contained in a log file.

   If your release settings include creation of a setup.exe launcher, you can use the **MSI Command Line Arguments** setting in the release properties to pass in the /L switch to msiexec.exe, with a value such as this:

   /L*v "%TEMP%\everything.log"

   Note that you cannot use MSI properties in the log-file path, as properties will not be available until after the installation has initialized. The command above relies on the command processor expanding the %TEMP% environment variable.

2. With MSI 4.0 on Windows Vista, you can set the MsiLogging property to a string containing the logging flags you want to use. With InstallShield 2008, you can set this using the **Create MSI Logs** setting in the Product Properties view. The path to the log file will be stored in the MsiLogFileLocation property. (Note that you can read this property but not set it.) Using this switch in the InstallShield environment also

displays the Show the Windows Installer Log check box in the SetupCompleteSuccess dialog box.

The MSI help library describes other options, such as setting the Logging policy in the registry, which creates a randomly-named log file in the Temp directory for every MSI installation.

The MSI Log File Analyzer, available under InstallShield's **Tools** menu, can generate an MSI log file and create various color-coded HTML reports based on a log file.

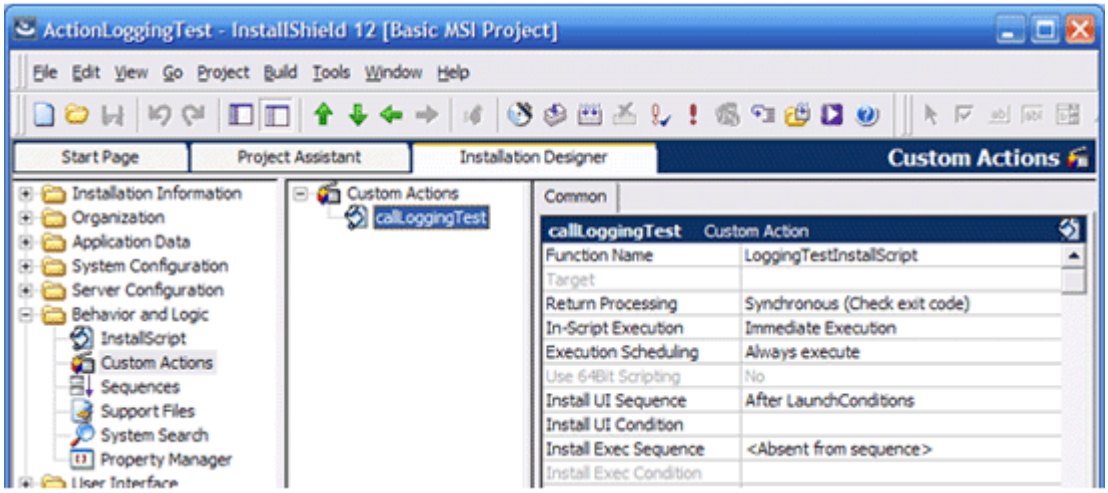
By default, an MSI log file contains a return value for each built-in and custom action. The following sections describe how to write more detailed information into the log file. Note that these techniques do not create an MSI log file or initiate logging, but instead write to a log file created by the /L switch, the MsiLogging property, or the like.

InstallScript Custom Actions

In an InstallScript custom action, you can call *SprintfMsiLog* to write a string to the MSI log file. For example, the following InstallScript code prototypes and defines an InstallScript custom action called *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;
```

In order to be called, the custom action must be scheduled in the sequences. For this example, open the Custom Actions view and create an immediate-mode InstallScript custom action called *callLoggingTest* that calls the *LoggingTestInstallScript* function, and schedule it to run after LaunchConditions.



After building the package and running it with the /L*v switch, you should see a line similar to the following in the log file:

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.

The SprintfMsiLog function is similar to the Sprintf and SprintfBox functions, in that you can include placeholders ("%s" or "%d" fields, called "format specifiers") in the message string if you want to splice in the values of string or numeric variables.

VBScript Custom Actions

With VBScript actions (as well as with MSI DLL actions, described in the following section), the general process is to assemble a *record* containing the message information, and then send the record to the running installation. With VBScript, you use Installer.CreateRecord to create the message record, and use Session.Message to send the record to the running installation.

The record begins with a "template" in field 0, which is a string containing placeholders of the form [1], [2], and so forth. These placeholders will then be filled in with the values in record fields 1, 2, and so on.

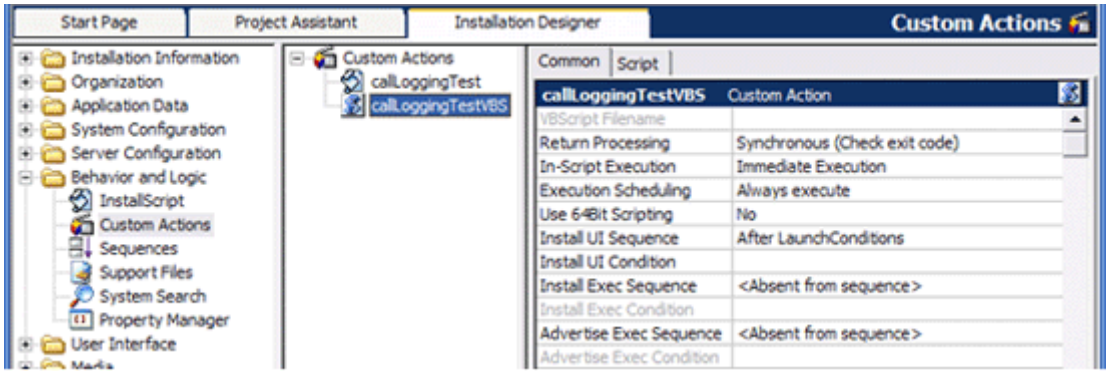
To demonstrate a VBScript action writing to the log, you can create an immediate-mode VBScript custom action called *callLoggingTestVBS*, scheduled immediately after LaunchConditions, with the following code.

(For this example, you can store the code directly in the custom action since the script is short. In practice, using an external .vbs file is generally recommended so you can specify a function return value and optionally exit the installer from the custom action.)

```
Const msiMessageTypeInfo = &H04000000

' create the message record
Set msgrec = Installer.CreateRecord(1)
```

```
' 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
```

The action might appear similar to the following figure.



After building the project and creating a log file, the following lines should appear in the log:

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

A possible refinement is to modify the template field (field 0) of the record to include a timestamp or information about whether the action is being called from immediate execution or deferred execution. Studying the log messages displayed by standard actions can provide a useful model.

MSI DLL Custom Actions

In an MSI DLL custom action written with C or C++, the process of writing to the log file is similar to the VBScript code, except that you use MsiCreateRecord to create the message record and MsiProcessMessage to pass the record to the running installer.

You begin by creating a C or C++ DLL project in Visual Studio, for example. The C++ code for this example might look like the following.

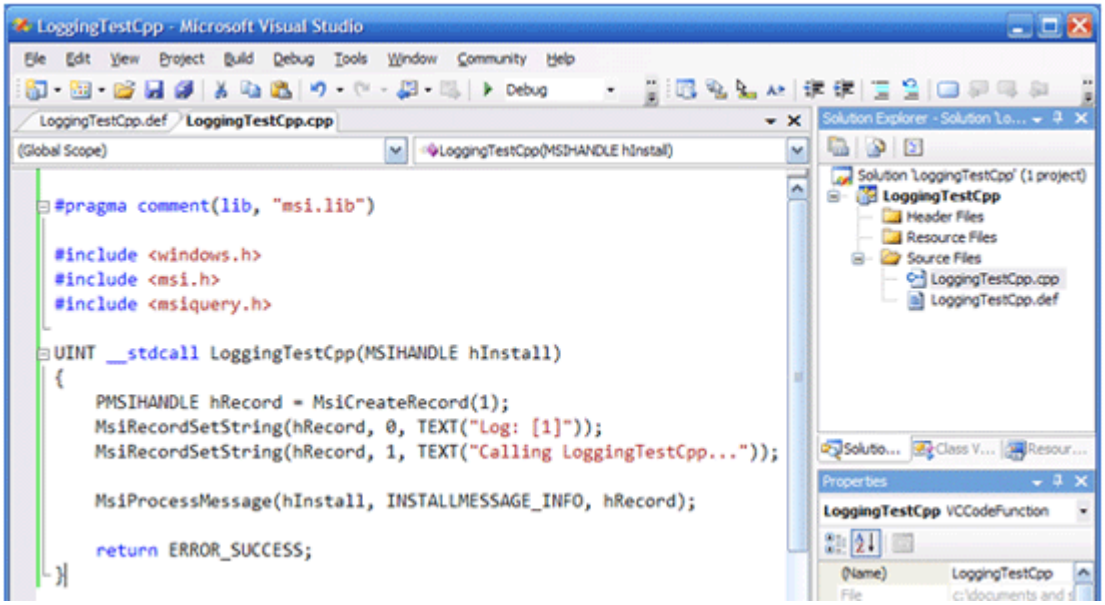
```
#pragma comment(lib, "msi.lib")

#include
#include
#include

// 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;
}
```

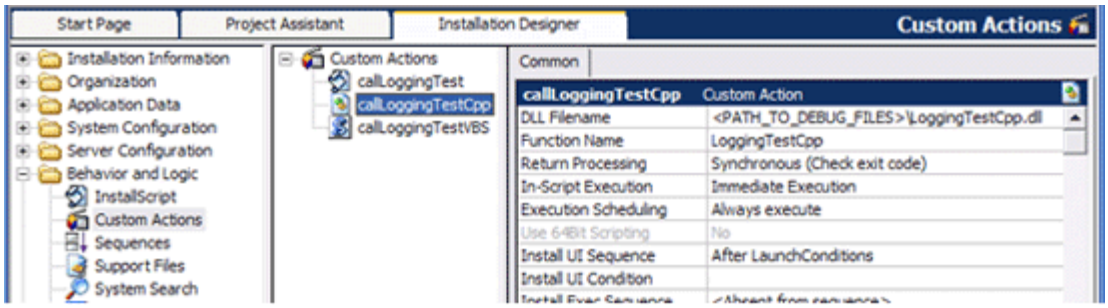
To ensure that the function name is properly exported from the DLL, you can create a .def file with contents similar to the following:

```
LIBRARY "LoggingTestCpp" ; DLL name

EXPORTS ; exported function names
   LoggingTestCpp
```

The DLL project in Visual Studio might look similar to the following.



After building the DLL, in InstallShield you can create an MSI DLL custom action, for this example calling it *callLoggingTestCpp*, and again scheduling it for immediate execution after LaunchConditions.



After rebuilding the project and running the MSI with the logging switch, lines similar to the following should appear in the log file.

Action 25:00:00: callLoggingTestCpp.
Action start 25:00:00: callLoggingTestCpp.
[...lines omitted...]
Log: Calling LoggingTestCpp...
Action ended 25:00:00: callLoggingTestCpp. Return value 1.

For more information, see the MSI help library topic "Windows Installer Logging".