MSI Tip: Authoring a Custom ICE using InstallShield 2008

By Christopher Painter
Author: Hard Core Setup Engineering

Introduction

In a previous MSI Tip article, Robert Dickau wrote:

"An important step in the release process for an installation project is to validate the completed MSI database. Microsoft provides a collection of ICE (internal consistency evaluator) tests for detecting many common errors and inconsistencies that can occur when authoring a Windows Installer installation."

While Microsoft and InstallShield both provide extensive ICE suites, often additional unit tests would help improve the quality of an installation by automatically identifying problems earlier in the software development cycle. Windows Installer was designed to allow setup developers to author their own custom ICEs and store them in .CUB databases. ICEs may be written using either standard call DLL custom actions or script custom actions such as Jscript and VBScript. Since InstallShield 2008 compiles InstallScript custom actions into standard call DLLs , it is possible to also use InstallScript to develop custom ICEs.

HostICE Sample Project

Included with this tip is HostICE, an InstallShield 2008 Basic MSI project that demonstrates how to create a custom ICE and a .CUB database to contain it. HostICE contains a sample ICE (CPICE01) that checks for the existence of VBScript and Jscript custom actions. While script-based custom actions are relatively easy to develop, they are very difficult to debug and often fail in peculiar ways. These problems become more pronounced as the complexity of the custom action increases. For this reason CPICE01 has been written to demonstrate how a custom unit test can be written to keep a watchful eye out for these types of custom actions.

Demonstration

[Download and extract HostICE.zip](). Open the InstallShield project and build it. Finally install the package to generate the .CUB database. A dialog will ask you where to save the .CUB file. (Figure 1)
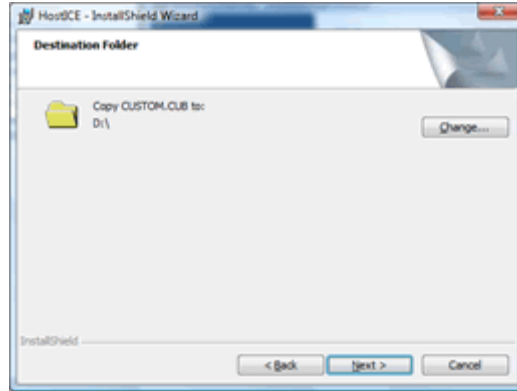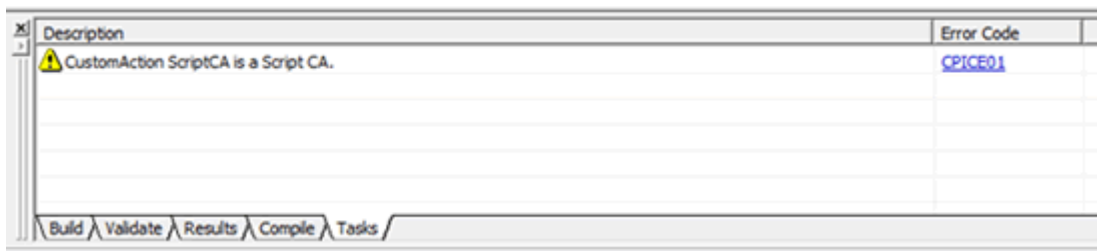
(Figure 1: Installing the .CUB)

Next, validate the package (Figure 2) by pulling down the Build Menu, select Validate, Browse for a New Validation Module and then select custom.cub in the directory that you saved the .CUB to earlier.
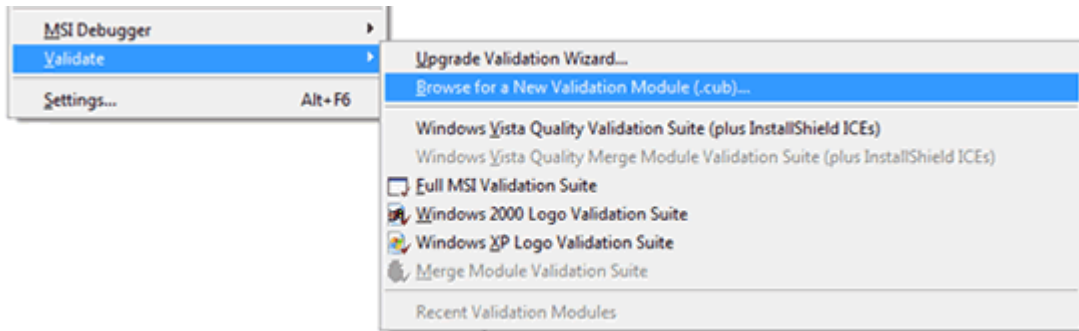
(Figure 2: Running Validation)

InstallShield then executes the ICE contained in the custom.cub and reports errors and warnings to the Tasks panel. (Figure 3)

(Figure 3: Viewing Validation Results)

Creating the .CUB Database

When you build HostICE, InstallShield compiles the InstallScript functions into ISSetup.dll and places it in the Binary table of the package. Additionally a generic .CUB database called custom.cub is stored in the ISSetupFile table.

When you run the HostICE package, a built in InstallShield custom action called ISSetupFilesExtract automatically saves custom.cub in a temporary directory. InstallShield also extracts a special configuration file called ISConfig.ini. This file describes the InstallScript custom actions contained within ISSetup.dll.

The CreateCUB custom action copies custom.cub from the temporary directory to the selected destination directory

```
CopyFile( svSUPPORTDIR ^ "custom.cub", svDir ^ "custom.cub" );
```

Then CreateCUB reads through ISConfig.ini and records the InstallScript ICE custom actions in the CustomAction and _ICESequence tables of custom.cub.

```
// Examine the IsConfig.ini file for the DLL function export names
// for all InstallScript CA's other then this one.
nvFunctionIndex = 1;
svSection="f1";
while( GetProfString( SUPPORTDIR ^ "IsConfig.ini", svSection, "Function", svResult ) = 0 )
    if( svResult != "CreateCub" ) then
      // Add the CA to the CUB.
      AddCustomAction( Database, svResult, svSection, nvFunctionIndex * 10 );
    endif;
    nvFunctionIndex++;
    Sprintf( svSection, "f%d", nvFunctionIndex );
endwhile;
```

These actions are performed in functions that invoke the Windows Installer API and Automation Interfaces to access MSI Databases at runtime. More detailed information on can be found in a previous InstallShield Tip written by Robert Dickau:
http://www.flexerasoftware.com/webdocuments/PDF/msiaccess.pdf

Validation

When InstallShield validates a database, it executes the ICE custom actions listed in the _ICESequence table of the .CUB package. Each ICE custom action is then provided an installation handle that can be used to get a handle to the active database being validated. The ICE gathers data, applies business rules and then uses the MsiProcessMesssage() function to report validation messages. InstallShield receives these messages, displays them on the screen and writes them to a log file.

CPICE01 then gets a list of all custom actions by querying the CustomAction table:

```
MsiDatabaseOpenView( hDatabase, "SELECT `Action`,`Type` FROM `CustomAction`",
hView );
MsiViewExecute( hView, hRecord );
```

Each custom action is then evaluated. The custom action type is then isolated from it's attributes by performing a bitwise and operation. The remaining number is then compared against a list of script custom action types.

```
Sprintf( svType, "%d", MsiRecordGetInteger( hRecord, 2 ) & 63 );
```

If a match is found an INSTALLMESSAGE_USER message is processed. The message consists of 3 tab delimited fields: ICE Name, Type (Failure | Error | Warning | Informational) and Message

```
// Create Message
Sprintf( svMessage, "%s\t%d\t%s", gsvICE, nvType, svDisplay );
hRecord = MsiCreateRecord( 1 );
MsiRecordSetString( hRecord, 0, svMessage );

// Send message
MsiProcessMessage( ghMSI, INSTALLMESSAGE_USER, hRecord );
```

Conclusion

By writing your own custom unit tests and automatically executing them as part of your build process, you can improve the quality of your install. Setup developers should consider implementing custom ICEs whenever a bug that could repeat itself is discovered, or when there is a packaging standard such as no script custom actions that the organization would like to enforce. When developing a custom ICE, InstallScript should be considered since it is a powerful, reliable and easy-to-use domain-specific language that fits very will with the requirements of authoring an ICE.