Detecting Internet Explorer and Other Applications with Windows Installer

Abstract: *Often applications require that a certain version of Microsoft Internet Explorer or some other third party software be installed on the target system. For instance, HTML Help requires Internet Explorer 3.0. This article explains how to search for Internet Explorer 5.5 or above and abort setup with an error message if it is not found. The same technique can be used to search for other applications.*

Internet Explorer Versions

To find out which version of Internet Explorer is installed, we'll examine the file version of Shdocvw.dll. In knowledge base article Q164539 Microsoft has documented that the existance of this file indicates that IE is installed, and how its file version relates to the installed IE version. Shdocvw.dll can be found in the system folder, as stored in Windows Installer's built in property [SystemFolder]. On Windows NT and 2000 this property typically resolves to C:\Winnt\System32 and on Windows 95, 98 and Me to C:\Windows\System. The following table shows some selected entries for IE versions. For a complete list see the knowledge base article.

Shdocvw.dll File Version	Internet Explorer Version
4.70.1155	Internet Explorer 3.0
4.71.1712.5	Internet Explorer 4.0
4.72.3612.1707	Internet Explorer 4.01 SP2
5.0.2014.213	Internet Explorer 5.0
5.50.4134.600	Internet Explorer 5.5

Note that Shdocvw.dll has a major version of 4 even in Internet Explorer 3.0, only the minor version number is different between IE3 and IE4.

InstallShield's Power Editor

Windows Installer has some standard actions and tables that we can use to search for Shdocvw.dll and examine its version number. However in this case InstallShield has not provided a wizard or any other abstraction layer to populate these tables, so we have to use the Power Editor that can be found under Advanced Views in IPWI 2.0. It is also available from the menu bar: pull down the Project menu in IPWI 2 or the Tools menu in ISWI 1.x and select Power Editor.

AppSearch Table

The first table we'll populate is the AppSearch table. It requires two entries: a property and a signature. Windows Installer will set the property to the full path of Shdocvw.dll if a sufficient version is found. It has to be a public property, i.e. its name can only consist of upper case letters and numbers, no lower case letters are allowed. We don't have to add this property in the Property Manager. The signature is used to reference the other table entries we'll make to specify our search. Enter the following information in the AppSearch table:

Property	Signature_
IE55FOUND	sigShdocvw

Signature Table

Now we need to tell Windows Installer which file we are looking for, and what version we require. This is done by populating the Signature table as shown below. Since this is a table with many columns I've broken it into two lines for better readability.

Signature	FileName	MinVersion	MaxVersion
sigShdocvw	Shdocvw.dll	5.50.4134.599	

MinSize	MaxSize	MinDate	MaxDate	Languages

Signature is the same as above and used to connect the row in the Signature table to the corresponding row in the AppSearch table. The entry for FileName should be obvious. MinVersion specifies the minimal version of the file that must exist to meet our requirements (see below). We leave the other fields empty to indicate that we don't care about the file size, date or language of Shdocvw.dll.

File Versions
The MinVersion field is exclusive, i.e. the actual file version must be greater (>) than the version number we enter here, not "greater than or equal" (>=). Therefore we must enter a version number that is actually 1 less than the version we are looking for. We are looking for 5.50.4134.600, so we enter 5.50.4134.599.

Similarly the MaxVersion field performs a "less than" (<) comparison, not "less than or equal" (<=). Therefore you'd have to add 1 to the allowed maximum version number if you wanted to specify an upper border.

Remarks:

- The version info resource embedded in an executable file typically contains two file version numbers. Windows Installer uses the numerical info, not the string file version. In most cases these should be identical, but sometimes they differ. This is confusing, because most Windows versions display the string file version if you right click on the executable in Windows Explorer. Use a resource editor or a tool like Dependency Walker to see the numerical version info.
- The version info resource also includes a product version number. This is not relevant in Windows Installer at all.
- Windows Installer compares all four fields in a file version number (aaa.bbb.ccc.ddd). This behaviour is different from the comparison of the msi package's overall ProductVersion where only the first three fields (aaa.bbb.ccc) are relevant for update checks.
- The Windows Installer documentation in the Platform SDK says that the version number comparison is inclusive (>= and <= relations), but that's not true. I don't know whether this is an error in the documentation or a bug in the Windows Installer runtime engine.

File Dates
If you wanted to specify a file date you'd have to use the MS-DOS format in the MinDate and MaxDate columns, i.e. some arithmetic is required:

((Year - 1980) * 512 + Month * 32 + Day) * 65535 +
Hours * 2048 + Minutes * 32 + Seconds/2

Remarks:

- The Windows Installer documentation in the Platform SDK for the MinDate and MaxDate fields is incomplete. It only lists year, month and day information, but omits the fact that these bits must be placed in the high order word of the date value. The low order word specifies the file time or can be set to 0 for midnight.
- The Windows Installer documentation in the Platform SDK says that the value you enter in the MinDate and MaxDate columns is compared against the creation date of the file, but that's not true. In my tests I found that Windows Installer is using the modification date of the file, which makes more sense because the creation date would be the install date for a file.

DrLocator Table

Finally we must tell Windows Installer where to search for the file. We know that Shdocvw.dll will always reside in the system directory, so we fill in the DrLocator table as follows.

Signature_	Parent	Path	Depth
sigShdocvw		[SystemFolder]	0

The signature is again the same as above, to assocuiate the rows is the three tables with each other. We leave the Parent field empty because we specify the full path in the Path column. We use the built in property [SystemFolder] which will automatically resolve to the correct path of the system folder on the target machine. We set the search Depth to 0 because we know that Shodvw.dll is in the system folder, not a subdirectory thereof.

If you don't know where the file will be located on the user's hard disk, you can leave both fields, Parent and Path, empty. In this case Windows Installer will search all fixed drives for the file, scanning as many subdirectory levels as specified in the Depth column (0 will only scan the root directory). Note that this can take a long time, depending on the number and size of local drives/partitions and the specified search depth.

Custom Action to Display an Error Message

Now that we have a property that will only be set if Internet Explorer 5.5 is installed, we need to set up an error message that will abort the installation if the property is empty. This can be done with a custom action of type 19 "Display error message and terminate the installation". This type of custom action is not available in InstallShield's custom action wizard.

Right click the custom actions view and select "New" to create a new custom action. Name the action "IE55Error". Fill in the proprty sheet as shown in the following table:

Type	19
Source	
Target	This application requires Microsoft Internet Explorer 5.5 or above
Comments	Terminate setup if IE 5.5 isn't installed

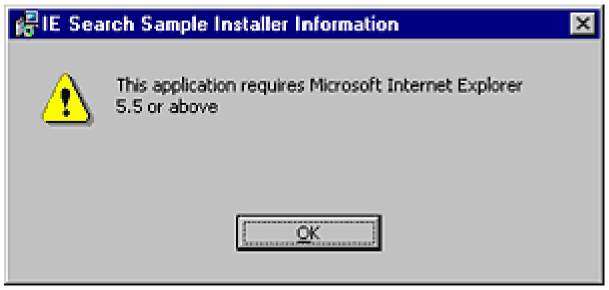
Instead of storing the text for the error message directly in the target field, we could have added an entry to the error table, which has advantages for multi-language setups. For simplicity I haven't used this method here.

Sequencing and Conditioning the Custom Action

Insert the IE55Error custom action in the Installation User Interface sequence after the AppSearch standard action and give it the following condition:

Not IE55FOUND And Not Installed

This will terminate setup with a message box like below if the specified minimum version of Shdocvw.dll isn't found. We want to do this only on first time install, else we would prevent uninstallation if the user deletes Shdocvw.dll after installing our application.


IE Search Sample Installer Information

This application requires Microsoft Internet Explorer 5.5 or above

We also need to insert the IE55Error custom action in the Installation Execute sequence (with the same condition), because in a silent install the User Interface sequence is not processed.

Finally make sure that the AppSearch standard action has no condition attached in either sequence. Early versions of ISWI used to put a condition of APPS_TEST on the AppSearch action, so that it wouldn't execute.

About the Author

Stefan Krueger is working as freelance setup consultant, offering support, custom programming and workshops independently from InstallShield Software Corporation. Besides his contract work, he answers questions in various newsgroups and maintains the InstallSite.org web site, a place where setup developers share resources and information among peers.

If you have any comments about this article, or want to suggest a topic that Stefan should discuss in a future article, please write to isnewsarticle@installsite.org. To read Stefan's articles from previous issues of the InstallShield Newsletter, please visit http://www.installsite.org/isnews.htm.