altiris®

Preparing Windows Installer Installations for Deployment

by Corwin Oakman

ABSTRACT

This white paper discusses repackaging legacy .EXE installations into Windows Installer (.MSI) format and customizing .MSI installations using transforms. It includes useful techniques for determining if installations were authored with Windows Installer, and offers suggestions for working with non-standard .MSIs, as well as building transform (.MST) files to facilitate integration into a managed environment. The intended audience is IT administrators who have a good working knowledge of packaging processes, Windows Installer, and Wise Package Studio®.

THE CASE FOR WINDOWS INSTALLER TRANSFORM FILES

Packaging applications into .MSI format offers many advantages to organizations including self-healing, rollback and install-on-demand. But often it is difficult to know if an installation is already in .MSI format and how to proceed with it. For instance, you may have an .EXE-based (legacy) installation that needs to be converted into .MSI format. In this case, Wise Package Studio's SetupCapture can be used to repackage the application setup into a manageable and distributable .MSI package.

In other cases, you may have an installation that is already in .MSI format, but requires customization to satisfy your organization's requirements. In this case, the .MSI is not a candidate for repackaging with SetupCapture; rather it should be "transformed" into an .MST file. SetupCapture can customize the installation by creating a silent installation or eliminating applications for specific departments in your organization, for example.

Why can't .MSI files be repackaged? In some cases, repackaging a Windows Installer installation can lead to application failure. Some applications, such as Microsoft® Office 2000, have hard-coded product IDs that only work with the original Office setup routine. The repackaged .MSI does not have the same configuration information, such as Product Code and Component IDs, as the original Microsoft Office .MSI. If you attempt to run any of the Microsoft Office tools, you'll get an error message similar to "This application must be installed in order to run."

Another reason that .MSI files should not be repackaged is because the Windows Installer patch and upgrade technology relies on Package IDs and Product Codes. If you repackage an .MSI, the Package ID and Product Code will change. When the vendor releases a patch, or .MSP, for the product, the patch may not get applied in cases where the Package ID or Product Code of the vendor's original installation is not found.

A better approach to customizing .MSIs is to create a transform file. Wise Package Studio's InstallTailor creates transforms by allowing you to choose installation options and generate an .MST file based on your choices. The process is quick and uses the vendor's installation, which makes the installation process more reliable.

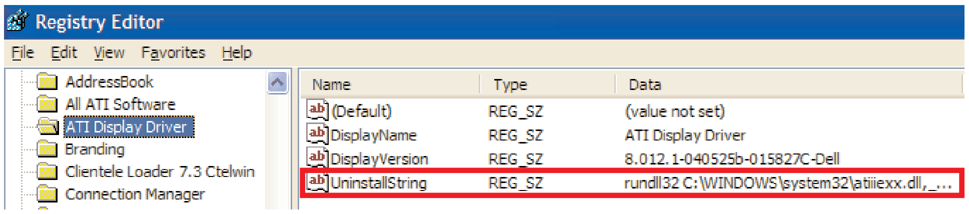
HOW TO DETERMINE IF YOUR APPLICATION IS IN .MSI FORMAT

Every application that will be prepared for deployment should be tested to ensure the quality of the software and the compatibility with your environment. This initial testing phase is an opportune time to determine if the installation is Windows Installer based. If so, you can create a transform of the .MSI using Wise Package Studio's InstallTailor. If the installation is EXE-based, you can use Wise Package Studio's SetupCapture to prepare the application.

If the vendor's installation media such as a CD or a download file includes an .MSI, you can use that file to generate your transform. You may need to include .CAB files from the media as well. Some vendor CDs that include .MSI files use external, uncompressed source files. You will notice directories on the CD like "Program Files" or "PFiles", and "Windows". You should copy those directories off the CD and keep them with your .MSI at all times, otherwise the MSI will not install without them. Another option is to perform an Administrative Installation which will extract the application's source files from the .MSI.

Is it Really an .MSI?

Often, the vendor media will not include an .MSI, but that doesn't mean the installation is not Windows Installer based. After you install the application, use the Registry Editor (regedit.exe) to view HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall. The Add/Remove Programs Control Panel is driven by these entries. Legacy, or non-.MSI applications, will have names listed as sub-keys of Uninstall. If your application is listed with a plain text name, it can be repackaged.

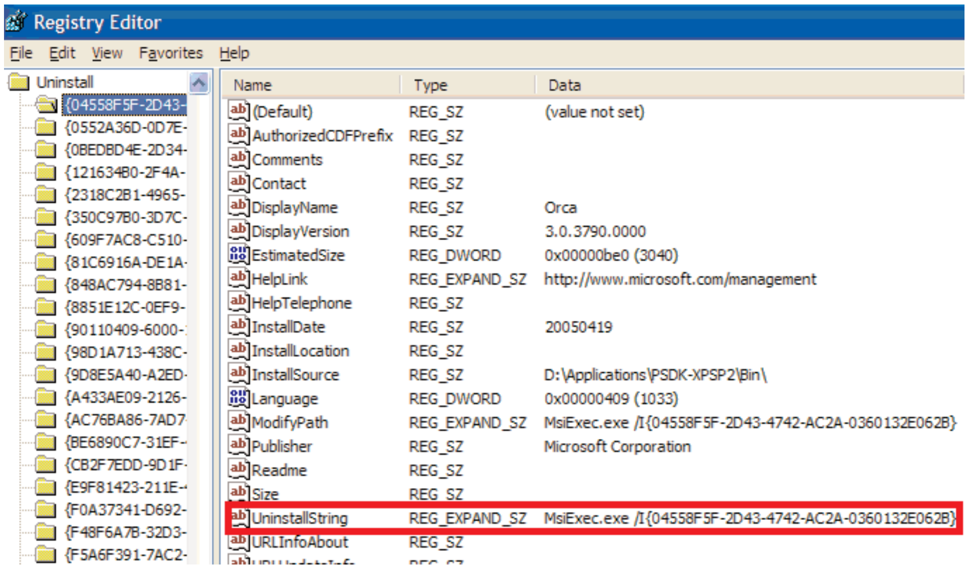


Name	Type	Data
(Default)	REG_SZ	(value not set)
DisplayName	REG_SZ	ATI Display Driver
DisplayVersion	REG_SZ	8.012.1-040525b-015827C-Dell
UninstallString	REG_SZ	rundll32 C:\WINDOWS\system32\atiiexx.dll,_...

In this example, the installation is a legacy or non-.MSI application because the value in the registry's UninstallString lists the name of the application

Windows Installer based installations will list their product code GUID as a sub-key of Uninstall. You can easily view the contents of these GUID entries to see if your application has placed an uninstall entry there. If you find your application's name listed as a GUID, check the "Uninstall String" value. If it references MSIEXEC.EXE or an .MSI file of some kind, you are dealing with an .MSI installation, which means you can create a transform.

FINDING THE MISSING .MSI



Uninstall
- {04558F5F-2D43-
- {0552A36D-0D7E-
- {0BEDBD4E-2D34-
- {121634B0-2F4A-
- {2318C2B1-4965-
- {350C97B0-3D7C-
- {609F7AC8-C510-
- {81C6916A-DE1A-
- {848AC794-8B81-
- {8851E12C-0EF9-
- {90110409-6000-
- {98D1A713-438C-
- {9D8E5A40-A2ED-
- {A433AE09-2126-
- {AC76BA86-7AD7
- {BE6890C7-31EF-
- {CB2F7EDD-9D1F-
- {E9F81423-211E-
- {F0A37341-D692-
- {F48F6A7B-32D3-
- {F5A6F391-7AC2-

Name	Type	Data
(Default)	REG_SZ	(value not set)
AuthorizedCDFPrefix	REG_SZ	
Comments	REG_SZ	
Contact	REG_SZ	
DisplayName	REG_SZ	Orca
DisplayVersion	REG_SZ	3.0.3790.0000
EstimatedSize	REG_DWORD	0x00000be0 (3040)
HelpLink	REG_EXPAND_SZ	http://www.microsoft.com/management
HelpTelephone	REG_SZ	
InstallDate	REG_SZ	20050419
InstallLocation	REG_SZ	
InstallSource	REG_SZ	D:\Applications\PSDK-XPSP2\Bin\
Language	REG_DWORD	0x00000409 (1033)
ModifyPath	REG_EXPAND_SZ	MsiExec.exe /I{04558F5F-2D43-4742-AC2A-0360132E062B}
Publisher	REG_SZ	Microsoft Corporation
Readme	REG_SZ	
Size	REG_SZ	
UninstallString	REG_EXPAND_SZ	MsiExec.exe /I{04558F5F-2D43-4742-AC2A-0360132E062B}
URLInfoAbout	REG_SZ	

In this example, the file is a true .MSI because the UninstallString references MSIEXEC.EXE.

As stated in the previous section, just because you don't see an .MSI on the vendor's media doesn't mean the installation is not an .MSI. Most installation-building software has the ability to "wrap" the .MSI within a setup executable. This is useful for pre-installing runtimes like Windows Installer or the Microsoft .NET Framework. Since administrators control the state of the desktops in their environment, we do not need to pre-install runtimes.

If you have already verified that you have a Windows Installer installation, you can easily locate the source installation file. First, restart the vendor's installation. When the first dialog comes up, do not proceed with the installation. Leave the installation open in the background and perform a search on the machine for "*.MSI". The .MSI that the vendor is using will have been extracted to a temporary location such as the %TEMP% folder or another location. Copy that .MSI and any associated .CAB files to a new folder. You now have an .MSI file with which you can build an .MST file.

INTERMEDIATE TRANSFORM CREATION TECHNIQUES AND TROUBLESHOOTING

If the .MSI you are customizing was extracted using the above techniques, and you experience problems running the .MSI or running InstallTailor, the vendor may have placed a check to verify the installation was launched from the setup executable. Sometimes these checks are superfluous as the installation is verifying the runtime pre-install has occurred.

It takes a bit of detective work to determine what is stopping the .MSI from installing without the executable. Look for custom actions or launch conditions that seem to have any tie-in with preventing the .MSI from installing. Launch conditions will be easier to find because the message displayed when the condition fails will match the error received when attempting to install the .MSI. Custom actions will be harder to trace. Look for custom actions with names that would hint toward setup executable checking. Trial and error will sometimes be necessary.

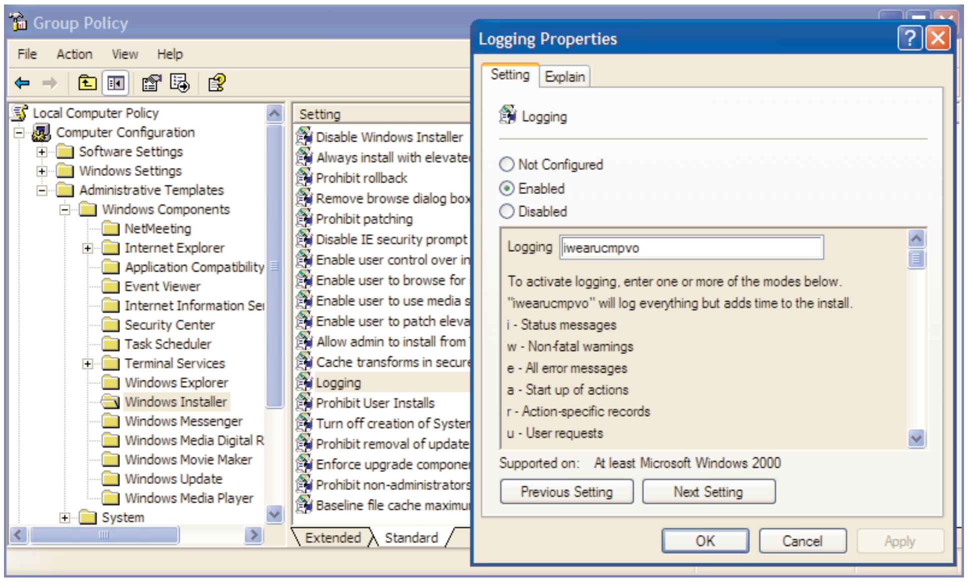
.MSI files from some vendors are reliant on a setup executable for other reasons. Some .MSI files contain no internal dialogs and update the executable based dialogs with custom actions. These custom actions will fail if the setup executable is not running. There is no simple solution to this. However, there are known custom actions to remove from .MSI files exhibiting this behavior. Removing those custom actions will give you a much greater chance of successfully freeing the .MSI from its setup executable requirements. Exercise caution in removing more than the actions listed below, as you could inadvertently remove a custom action that is required to install or configure parts of the application.

The custom actions are:

- ISVerifyScriptingRuntime
- ISStartup
- OnCheckSilentInstall
- OnGeneratingMSIScript
- OnMoving
- OnFeaturesInstalling
- OnInstallFilesActionBefore
- OnInstallFilesActionAfter
- OnFeaturesInstalled
- OnMoved
- OnGeneratedMSIScript
- ISRebootPatchHandler

Often the vendor's setup executable will install a scripting runtime before running these .MSI files. The runtime facilitates running the custom actions listed above, as well as other custom actions in the .MSI. This runtime must be present on the target machine before installing these .MSI files.

Sometimes the setup executable will pass properties and installation options to the .MSI using command line parameters. These properties can be discovered and used in your .MST file. To facilitate this, enable Windows Installer verbose logging to view the command line in a log file. Enable the Logging policy in the Group Policy editor under Computer Configuration > Administrative Templates > Windows Components > Windows Installer to activate Windows Installer logging. Set the Logging value to iwearucmpvo. The logging policy will create a log file for any Windows Installer process that runs on the computer. The log files are stored in the Temp folder, usually %TEMP%, and use the MSI?????.log naming convention.



Logging

Not Configured [✓ Enabled] Disabled

Logging: iwearucmpvo

To activate logging, enter one or more of the modes below.
"iwearucmpvo" will log everything but adds time to the install.
i - Status messages
w - Non-fatal warnings
e - All error messages
a - Start up of actions
r - Action-specific records
u - User requests

Supported on: At least Microsoft Windows 2000

Enable logging in the Group Policy Editor in order to discover properties and installation options that the setup executable is passing to the .MSI.

After running the vendor's installation, find the log file in the Temp folder. It should be the .LOG file with the most recent date. Open the log file, and search for "Command Line." The first result will be near the top of the log, and will contain any options passed on the command line. The command line will always contain the properties CURRENTDIRECTORY, CLIENTUILEVEL, and CLIENTPROCESSID.

An Example Command Line entry from a log file:

```
MSI (c) (08:B8) [14:58:45:027]: Command Line: ADDLOCAL=ALL
CURRENTDIRECTORY=C:\Testing CLIENTUILEVEL=0 CLIENTPROCESSID=2568
```

These properties should not be included in your transform. Any other property listed on the command line can be used.

APPLYING THE TRANSFORM

Once developed, the .MST file must be deployed with the .MSI file. The recommended method is to use the Windows Installer TRANSFORMS property. You could also use the command line switch, /T, but there have been documented problems using that switch in some instances.

Example command line:

```
Msiexec.exe /I “\\path\File.MSI” TRANSFORMS=”\\path\File.MST” /QB
```

/Q, /QB, or /QN will be necessary to deploy the .MSI if there are no dialogs to display in the User Interface sequence. For more information about command line switches, see the ‘Command Line Options’ help topic in the Windows Installer SDK.

SUMMARY

Almost any .MSI file can be transformed and run without its setup executable. Following the guidelines above and gaining experience with non-standard .MSI files will make the process of creating a transform much easier.

For further reading about transforms, and other topics introduced in this document, please see the following Windows Installer SDK topics:

- About Transforms
- Applying Transforms
- TRANSFORMS Property
- Command Line Options
- Administrative Installation