Genuine Scripting for Windows Installer

An InstallAware Whitepaper

March 2005

INSTALLAWARE STUDIO
INSTALLAWARE STUDIO
Copyright © InstallAware 2003-2006 (www.installaware.com) All Rights Reserved.

Contents

Introduction

InstallAware introduces branching, script-driven execution logic into Windows Installer. Yet, InstallAware does this without requiring any external scripting engine or language that needs to be installed as a scripting runtime before-hand. InstallAware generates 100% pure MSI installations while also allowing them to execute like the traditional setup scripts used before 1999. This provides developers with unprecedented levels of flexibility in coding their setup logic. Such flexibility was unavailable since the initial launch of Windows Installer in 1999.

Genuine Scripting for Windows Installer

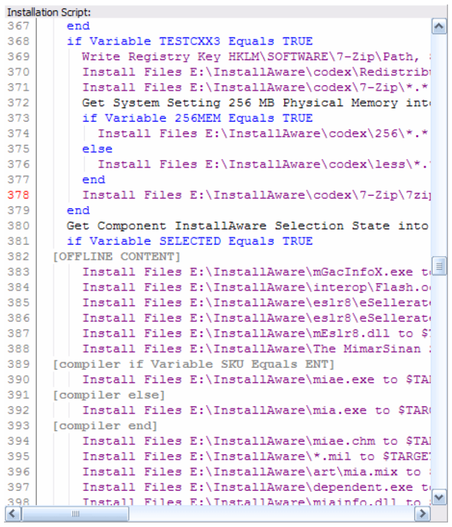
One of the major strengths of InstallAware v5 is the unique architecture it employs for creating Windows Installer packages.

While all other setup tools for Windows Installer provide high level visual editors for creating Windows Installer packages, none of them offer a native scripting capability for Windows Installer. Any scripting that is offered either requires an external scripting engine, which needs to be pre-installed and thus adds an extra layer of complexity to setups; or executes directly through another scripting technology which does not require to be pre-installed before use.

Even though this approach is suitable, it bifurcates installation projects into two disparate sets – one that goes through Windows Installer, and another that goes through the third party scripting engine (which may or may not need to be pre-installed). Separate editors and environments must be mastered for these two independent aspects of setup projects, and this increases the burden on the developer. It also complicates the development process as the two separate parts can interact with each other in unexpected ways and have unexpected dependencies. InstallAware has addressed this problem since its very first release in a conceptually clean, technically sound manner.

Types of Scripting Commands

InstallAware v5 presents a single, unified setup script which effectively deals with the bifurcation problem. The setup script is the heart of the installation and has complete control over what happens – and when.



Installation Script:

```
    end
    if Variable TESTCXX3 Equals TRUE
      Write Registry Key HKLM\SOFTWARE\7-Zip\Path,
      Install Files E:\InstallAware\codex\Redistrib
      Install Files E:\InstallAware\codex\7-Zip\*.*
      Get System Setting 256 MB Physical Memory int
      if Variable 256MEM Equals TRUE
        Install Files E:\InstallAware\codex\256\*.*
      else
        Install Files E:\InstallAware\codex\less\*.
      end
      Install Files E:\InstallAware\codex\7-Zip\7zi
    end
    Get Component InstallAware Selection State into
    if Variable SELECTED Equals TRUE
  [OFFLINE CONTENT]
      Install Files E:\InstallAware\mGacInfoX.exe t
      Install Files E:\InstallAware\interop\Flash.o
      Install Files E:\InstallAware\eslr8\eSellerat
      Install Files E:\InstallAware\eslr8\eSellerat
      Install Files E:\InstallAware\mEslr8.dll to $
      Install Files E:\InstallAware\The MimarSinan
  [compiler if Variable SKU Equals ENT]
      Install Files E:\InstallAware\miae.exe to $TA
  [compiler else]
      Install Files E:\InstallAware\mia.exe to $TAR
  [compiler end]
      Install Files E:\InstallAware\miae.chm to $TA
      Install Files E:\InstallAware\*.mil to $TARGE
      Install Files E:\InstallAware\art\mia.mix to
      Install Files E:\InstallAware\dependent.exe t
      Install Files E:\InstallAware\miainfo.dll to
```

The script contains several different classes of commands:

- Comments: As found in all programming environments, comments are hints for the developer.
- Flow Control: Commands that direct program flow into alternate branches based on the evaluation of conditions.
- Plug-In: Custom plug-in provided commands.
- Directive: Directives to the setup compiler, such as compiler variables, conditionally including/excluding certain types of code, and web media block declarations.
- Windows Installer: Commands that have direct correlates in the MSI database that is created at build time. Most commands that actually install the application (as opposed to, for instance, displaying dialogs and controlling program flow) fall in this category.
- Modify System: Commands that apply the pending changes to the system (either an installation or an uninstallation).
- Label: Labels for use in directing program flow control.
- Statement: Any scripting command that either obtains system information, or makes direct, immediate changes to the system without going through the Windows Installer engine.

If you are a developer familiar with other Windows Installer authoring tools, you may find it easy to think of every command type, except Windows Installer commands, as custom actions. In effect, they are custom actions that are executed by the InstallAware engine. And, InstallAware also achieves conditional flow for your Windows Installer commands. The two combined together in a single setup script effectively eliminates the bifurcation problem with zero technical/conceptual overhead.

Achieving Conditional Program Flow with Windows Installer

Windows Installer does not permit conditional program flow – that is, the developer may not create setups with execution logic like the following:

```
If Variable X is True then
  Install File Set A
Else
  Install File Set B
End
```

Windows Installer does permit attaching conditions to each component that installs, however this is a very inconvenient and counter-intuitive way of achieving conditional program flow, and in most cases is not a functional equivalent of conditional program flow either.

InstallAware, with its unique Genuine Scripting for Windows Installer technology, makes conditional program flow possible. Inside the installation script, you are free to use as many Windows Installer commands as you like, and you may enclose those commands inside as many nested conditional commands as you need, directing program flow according to your installation requirements. This flexibility lets you focus on the actual logic of your setup, and gets your application installed correctly on a diverse array of hardware and software platforms, without having to worry about the implementation details on the Windows Installer side.

When you build your setup, InstallAware does several things behind the scenes:

- It parses your script for Windows Installer commands, and populates the necessary MSI table structures for them, and
- It attaches a unique condition to each Windows Installer command.

At runtime, as the InstallAware setup driver executes your scripting code, when it comes across Windows Installer commands, it sets their unique condition to true. And when the InstallAware setup driver encounters a Modify System command, it applies all pending installation (or uninstallation) changes to the system. This mechanism constitutes the heart of Genuine Scripting for Windows Installer, and very effectively creates the effect of conditional program flow on the Windows Installer platform, embracing and extending MSI technology to do what could not be done before. Genuine Scripting for Windows Installer will save you countless hours of development and testing time when building setups that require complex installation logic.

A Single Script for Installs, Maintenance, and Uninstalls

Another time saving feature offered by the Genuine Scripting for Windows Installer technology is having a unified script for the entire install/maintain/uninstall cycle. If parts of an application are to be removed during a maintenance operation, and other parts added, absolutely no additional scripting code is required to install the new parts, or remove the old parts. And of course, no scripting code is required to remove all installed parts during an uninstallation.

A script with manual coding for the install/maintain/uninstall cycles could look like the following:

```
If Installing or Maintaining
  If Feature A is Selected
    Install File Set A
  If Feature B is Selected
    Install File Set B
End

If Maintaining
  If Feature A is Not Selected
    Remove File Set A
  If Feature B is Not Selected
    Remove File Set B
End

If Uninstalling
  Remove File Set A
  Remove File Set B
End
```

With InstallAware, you have a much simpler script that does everything in one place. If parts of your application are no longer needed, InstallAware automatically removes them (during either a maintenance operation, or an uninstallation) – no manual coding is necessary. If new parts of your application must be installed, InstallAware automatically adds them. All powered by the same block of code – including the first time install:

```
If Uninstalling
  Modify System: Apply Uninstall

If Installing or Maintaining
  If Feature A is Selected
    Install File Set A
  If Feature B is Selected
    Install File Set B
  Modify System: Apply Install
End
```

When you compare the two code listings above, you immediately see that Genuine Scripting for Windows Installer, through its unified scripting model, saves you many lines of manual coding for the maintenance and uninstallation cycles of your application, while still operating on a highly customizable scripting model.

Limitations of Genuine Scripting

Because each InstallAware setup is still based on Windows Installer technology, there are some unavoidable restrictions. Fortunately these are very few in number – only two – and will not impact most setups. Moreover, these restrictions would apply to any setup based on Windows Installer technology – regardless of whether InstallAware or another Windows Installer tool was used.

The first limitation is that loops are not supported. While the InstallAware script contains constructs that allow you to code loops, if you execute a Windows Installer statement multiple times within a loop, only the last iteration of that command will actually run on the target system. Because InstallAware cannot determine at compile time the number of times a command located inside a loop may execute, it can populate only a single set of Windows Installer table structures for that command. At runtime, even though the command may execute multiple times, only the last iteration will have any effect. This is because only the last execution's setting of internal Windows Installer conditions will survive – the previous internal conditions will all have overridden one another.

The second limitation is that no changes are actually made to the target system when a Windows Installer command executes. You may think of this as InstallAware having cached and queued the command for deferred execution. The deferred commands are all executed at once when one of the Modify System commands are called (Apply Install, Apply Advertised, Apply Uninstall). Apply Install installs the application, Apply Advertised installs the application in advertised mode, and Apply Uninstall removes a previously installed application. Because Windows Installer does not permit conditional program flow, the only alternative to this approach would be immediately applying changes to the system any time a Windows Installer command was called. While the InstallAware development team investigated this option early during development, it caused extremely severe drawbacks in execution time and was abandoned.

The impact of either limitation on installation development is very small.

For the first limitation, very rarely would commands that change the target system reside inside a loop. In fact, in the entire range of setup scripts developed by InstallAware Software Corporation, including internal development and external consulting projects, none of the setups had this requirement.

As for the second limitation, conditions that require setup commands to be applied immediately are also rare – they only occur under circumstances in which the setup flow branches based on the outcome of a preliminary setup command. If you do have a need for such branching, you will be relieved to learn that InstallAware allows you to call the Apply Install command more than once in your script, effectively sidestepping this limitation. The only drawback is that each time Apply Install is called, the Windows Installer engine is invoked – which performs a complete installation routine every time. Therefore calling Apply Install multiple times will noticeably degrade your installation speed, especially if the calls come after large blocks of file copy operations.

"Non-Windows Installer" Scripting Commands

As extensive a setup engine Windows Installer is, it does not provide the total set of installation commands – and conveniences – that one expects to find in a complete, self-contained installation authoring environment. InstallAware v5 provides an extended, user customizable set of scripting commands which save you both time and effort in developing application installations.

Get System Information

Built-in scripting commands, which fall under the generic Statement category, offer ways to obtain extensive system information, including hardware properties (physical memory, display resolution), software properties (the running operating system, third party applications), user properties (privilege level, logged on user/domain/computer name), and more.

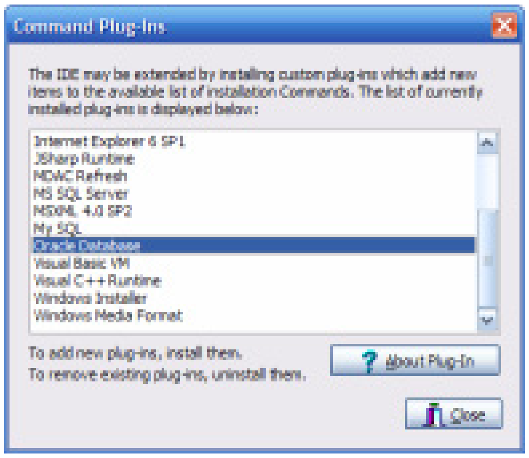
Advanced Install Actions

Statements are also available which perform advanced installation actions. For instance, you may create and update IIS websites and virtual folders. You can share folders or remove existing shares. You may set permissions on file system, registry, or other types of system objects. You may connect to a wide variety of database platforms and execute SQL scripts, including MS SQL, My SQL, and Oracle database servers (no client software is required on the target machine to connect to remote servers).

Run Programs, Call Dynamic Link Libraries

A very convenient way to extend the behavior of your setup scripts is to call outside code – especially if you've already encapsulated such code into stand alone executable files or dynamic link libraries. InstallAware lets you run any external program, or call any arbitrary dynamic link library function with a variable list of parameters, all directly as part of your setup script. The InstallAware v5 IDE also makes it even easier to run programs in your setup finish dialog, or before/after the main installation/uninstallation.

Plug-In Extensibility



The IDE may be extended by installing custom plug-ins which add new items to the available list of installation Commands. The list of currently installed plug-ins is displayed below:

Internet Explorer 6 SP1
JSharp Runtime
MDAC Refresh
MS SQL Server
MSXML 4.0 SP2
My SQL
Oracle Database
Visual Basic VM
Visual C++ Runtime
Windows Installer
Windows Media Format

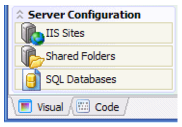
To add new plug-ins, install them.
To remove existing plug-ins, uninstall them.

Whenever you find that the InstallAware scripting language falls short of meeting your needs (and such cases ought to be rare), you can always tap the power of your favorite programming language and directly code what you want to do in the development environment that you are most comfortable and productive in. InstallAware is plug-in extensible, and what's even better, each plug-in command also looks and works just like a native part of your script: It is visible in the script editor, can be copied/cut/pasted like any other scripting command, and has full access to the state of the installation, including reading from/writing to script variables.

InstallAware ships with several pre-built plug-ins that perform various tasks, and also includes two plug-in templates for plug-ins implemented using the Visual C++ and Delphi programming languages. Of course, you may develop your own plug-ins in any environment capable of creating standard Win32 DLLs.

Two-Way Integrated Visual View

One of the most time-saving features when scripting in the InstallAware IDE, and one that has been greatly enhanced in the InstallAware v5 release, is the visual view that represents the installation actions taken in your setup script graphically.



Just click the visual tab at the bottom of the InstallAware IDE window to switch to the visual view. The visual view intelligently parses your setup script and also works with highly customized scripts. It contains 26 separate pages that represent different aspects of your installation, from files being installed to your setup dialog designs. Whenever you make changes in the visual view, it will automatically update the underlying scripting code for you. It will do so without damaging your existing setup logic and correctly insert/remove code where applicable.

The visual view is a great time saver and helps you make the best of your installation script – by letting you avoid custom scripting unless absolutely necessary.

Automation Interface

While scripting setups in the InstallAware v5 IDE is a fun and enjoyable process, there may be times in which you need to programmatically emit setup scripts without depending on the IDE. For instance, you may be building a large set of installers with very similar features but different branding/names for your network of resellers. Especially if you have a large number of resellers and installers to customize, this could very quickly escalate into a development nightmare.

With the automation interface in InstallAware v5, you get full programmatic access to InstallAware setup scripts. Directly from your own external programs, you may:

- Emit a complete setup script, line by line
- Emit a setup project, with its own settings, dialogs, support files, and such
- Build a setup, or build a patch

This provides unprecedented flexibility in developing installers. Continuing our reseller scenario, you could build a website where each reseller could log on to their account, and upload their custom graphics and text for their brand/version of the product. Then they would click a "Build Installer" button which would immediately invoke the InstallAware automation interface, emit and build their setup, and deliver the download link. Any time you updated your software to a newer version, you wouldn't have to worry about manually updating the many installers for your resellers either! You'd just let them know that you have a newer version available, and they could all log on to their accounts and generate their personalized installers – all delivered by the InstallAware automation interface.

This is just one example for a case where InstallAware automation takes a major burden off of your shoulders. There are countless other uses for this technology. Some InstallAware licenses even permit you to redistribute the automation libraries, so you can build installers directly on end-user systems! For instance, consider a screen saver creator application – each screen saver it generates would also need an installer. Developing a home made installer for the screen savers produced would be a major undertaking. With a redistributable automation interface, you can simply call the InstallAware library functions and let InstallAware do what its best at – building setups – while you get to focus on your core expertise.

The automation interface is cleanly encapsulated inside Win32 DLLs which can be called from any Windows programming language. The automation interface also provides an ASP object which can be called from an Active Server Pages website.

The InstallAware Customer Site, which vends licenses to developers who have purchased InstallAware, uses the automation interface to create watermarked installers that contain the name of the individual or organization that purchased the license.

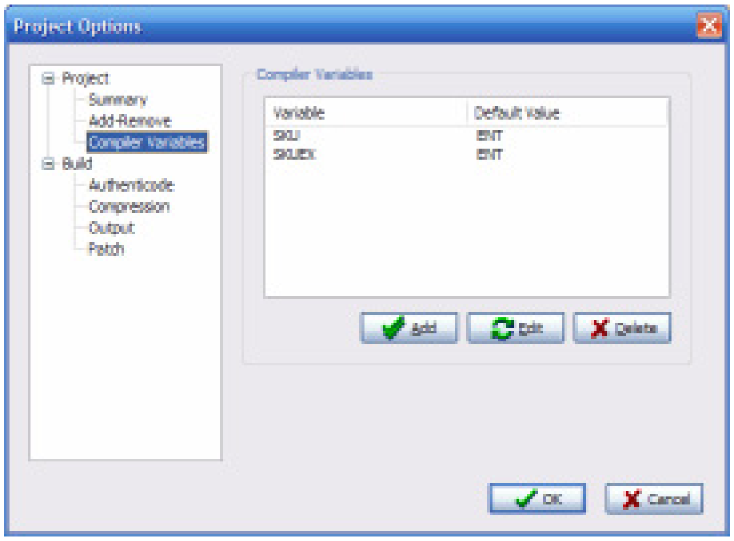
Compiler Variables

There may also be cases in which you need to customize your installers – but not as thoroughly as one might using the automation interface. For instance, you may have several editions for a product, each with progressively more features and higher prices. When building the installers for these various editions, instead of coding multiple scripts for each, you would rather code and maintain a single script. In this case, using the automation interface would be overkill – another way to quickly customize the script becomes necessary.

Compiler variables are resolved at build time, and let you achieve the following:

- Include or exclude various parts of code based on compiler variable values
- Substitute compiler variables used in script with their literal values

Continuing our example with the various editions, you could use Compiler Variable If statements to conditionally include/exclude parts of your setup script and application files, based on the edition. You could also use compiler variables directly in the script for build time substitution of their values into the setup script (such as, the edition name).



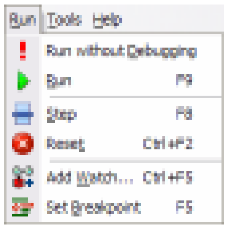
Variable	Default Value
SKU	ENT
SKUEX	ENT

Compiler variables, and their values, are defined in the Project Options window. The values specified in the Project Options window may also be overridden while building from the command line, or via the automation interface. Coupled with the redistributable command line build interface that is available in some InstallAware licenses, compiler variables may also be used to create customized installation packages directly on end-user systems.

The InstallAware product itself comes in various editions, and each edition has a different set of features and source files. The setup script for InstallAware makes use of compiler variables and delivers all the different editions from a single source script.

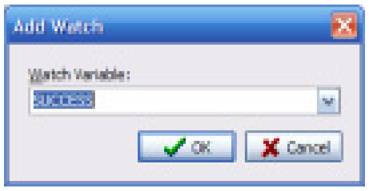
Integrated Debugging

Despite the beauty and power of Genuine Scripting for Windows Installer, or perhaps because of it, the need to debug the scripts you have developed in InstallAware will arise sooner or later. Along with the canonical MessageBox command that can be used to output values of variables to the screen, InstallAware v5 provides a visual, integrated debugger right within the main IDE.



The run menu contains all the debugging commands you need:

- Start debugging
- Step through code line by line
- End a debug session
- Add variable watches
- Set breakpoints in source code

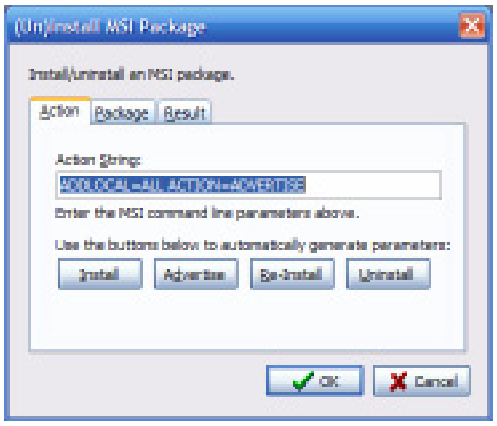


You may also override the values of variables while in a debugging session to test alternate routes of program flow.

Shelling to Other MSI Installations

Windows Installer offers the merge module mechanism for extending your setups. Merge modules are Windows Installer database files with the MSM extension, self contained blocks of installation logic and code, used for installing common parts of applications (such as application runtimes). Merge modules are "merged" into the main installation database, the MSI file, at build time, and with absolutely no effort on part of the developer, extend the MSI database with their additional logic and code, providing for the installation of the application runtimes, and so on.

InstallAware v5 supports merge modules, but again adds an extra bit of functionality. With InstallAware v5, you may (un)install or reconfigure complete third party products, given their MSI files, or product GUIDs, in addition to merging MSM databases with your setup.



Install/uninstall an MSI package.

Action String: ADDLOCAL=ALL ACTION=ADVERTISE

Enter the MSI command line parameters above.

Use the buttons below to automatically generate parameters:

The (Un)Install MSI Package command provides unprecedented flexibility in "shelling to" third party MSI packages. You may perform the full range of setup actions that are supported by Windows Installer on shelled packages – install or advertise, select all features or just a partial set, and even modify/remove an existing installation.

The command lets you customize the exact command line that will be used in the shelled-to installation, offers the ability to log setup execution, and reports whether the operation was successful, optionally capturing the error message if it failed. Setup packages are identified by their GUID or full path to the setup database. What's best, InstallAware even captures the progress of the shelled-to installation, displaying it as an integral part of your own application installation. Of course, you are able to completely hide the end-user interface of the MSI package you are shelling-to, including hiding any potentially confusing setup choices and dialogs. You get to install the packages with the exact settings you require, with no effort/confusion on part of the end-user running your master setup.

Shelling to setups provides a very viable alternative to application repackaging. Unlike application repackaging which is error-prone, and loses the intent of the original package author, shelling to setups preserves both the original setup logic of the installation, and adds the benefit of being a completely seamless and integral part of your own, larger installation.

Possible applications of this technology are endless. A very viable use for the shelling technology is resolving application conflicts automatically. For instance, if an application is detected on the end-user system which conflicts with your product, it can automatically be removed, all as a setup pre-requisite for your own installation. The Is MSI Setup Installed command reports whether a particular MSI package has been installed on the system, and complements the shelling facility in resolving application conflicts – all with zero end-user intervention/confusion.

Summary

Built from the ground up, InstallAware v5 achieves what no other installation development environment can, providing you with state-of-the-art tools that save you time and effort while building modern application installations. InstallAware v5's unique Genuine Scripting for Windows Installer technology makes InstallAware v5 the ultimate setup development solution.

About InstallAware Software Corporation

InstallAware Software Corporation was founded by InstallShield alumni in 2003. The company focuses on software installation technologies for the Windows Installer platform and brings a fresh approach to the setup development process. The privately held company has received numerous seed investments from venture capital firms and is a Borland Technology Partner.