Using InstallAware 7

To Patch Software Products

August 2007

TABLE OF CONTENTS

Overview

This whitepaper describes how to use InstallAware for patching software products. While many third party patching tools are available, most are loosely integrated with installation tools. This makes the patch delivery and installation awkward, resulting in a less than satisfactory user experience. Certain setup authoring tools do provide an integrated mechanism for building patches, improving the user experience. However, in these cases the patch authoring process is very complex and requires special training. Patches are treated as a special kind of setup project, with many authoring pitfalls that the developer has to be conscious of both before and during actual patch development. Since patches are distributed after the actual product has shipped, this results in difficult situations where the original setups can no longer be modified, but were required to be authored differently for the patches to be successful.

InstallAware addresses these challenges in patch authoring using a unique technology: One-Click Patching. InstallAware literally builds patches for older versions of your products in a single click. You simply refer the IDE to the old, built versions of your setup to upgrade, and click the “Build Patch” button. No additional coding or development effort is required at any time. InstallAware takes care of the rest for you.

Despite this remarkable ease of use, in the InstallAware tradition, extensive customization of the patch process and patch user experience remains possible. This makes it possible for you to integrate your business logic into your installer at runtime, dramatically reducing development overhead that would otherwise be required to support that logic. For instance, before applying a patch, dynamic validation of the user’s existing product license may be performed, barring the software upgrade if the license has expired or is invalid. Similarly, a single patch may be applied onto multiple targets – making it possible to reduce integration workload when patching products in different target languages, editions, and other configurations.

InstallAware further aids patch deployment by reducing the payload size, improving compression ratios, and caching old version setup sources. Patches are built using binary incremental differencing technology, which includes only the delta between the old and new versions of your files in your patch packages. This data is then compressed using LZMA/BCJ2 compression technology, which pre-processes files to increase their compressibility for even more space savings. At runtime, InstallAware simply refers to cached setup sources, eliminating unnecessary CD/DVD and other kinds of original setup media prompts, which would otherwise derail patch deployment (other setup authoring mechanisms do not cache setup sources on the end-user system).

While offering these unique benefits, InstallAware is completely standards based — every InstallAware patch is a native Windows Installer patch package (MSP) that can execute directly, without requiring a third-party scripting runtime to be previously installed. Windows Installer is the only Microsoft approved installation technology and is a requirement of Microsoft logo certification programs. It offers several other advantages such as elevated-privilege installations in high-security contexts and automated corporate deployment using Active Directory Group Policy. These benefits make MSP format patches the only accepted software distribution format for large enterprises.

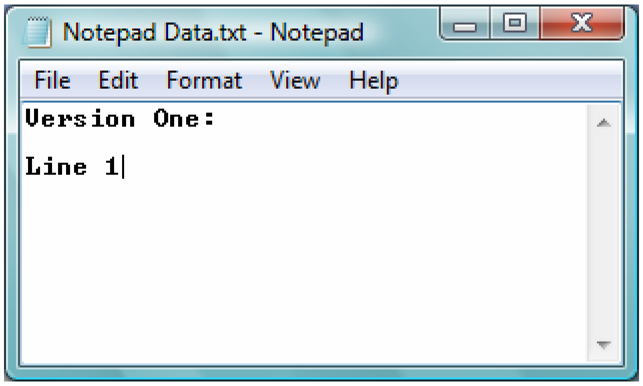
Building the First Version Installer

While it is possible to upgrade products built using any setup authoring technology using InstallAware, those upgrades cannot take advantage of InstallAware's One-Click Patching technology. Any product which you wish to patch using InstallAware must have originally been installed with InstallAware.

To build the installer for the first version of your product, perform the following steps.

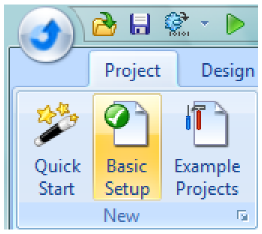
1. Our hypothetical product is comprised of two files, an application file and a data file. Windows's standard text editor Notepad, located in your Windows folder, will serve as our application file.

   We'll go ahead and create the data file. Right click your Desktop, and choose **New ▶ Text Document**. Name the file `Notepad Data`, double click it, and edit it so its contents look like the screenshot below:

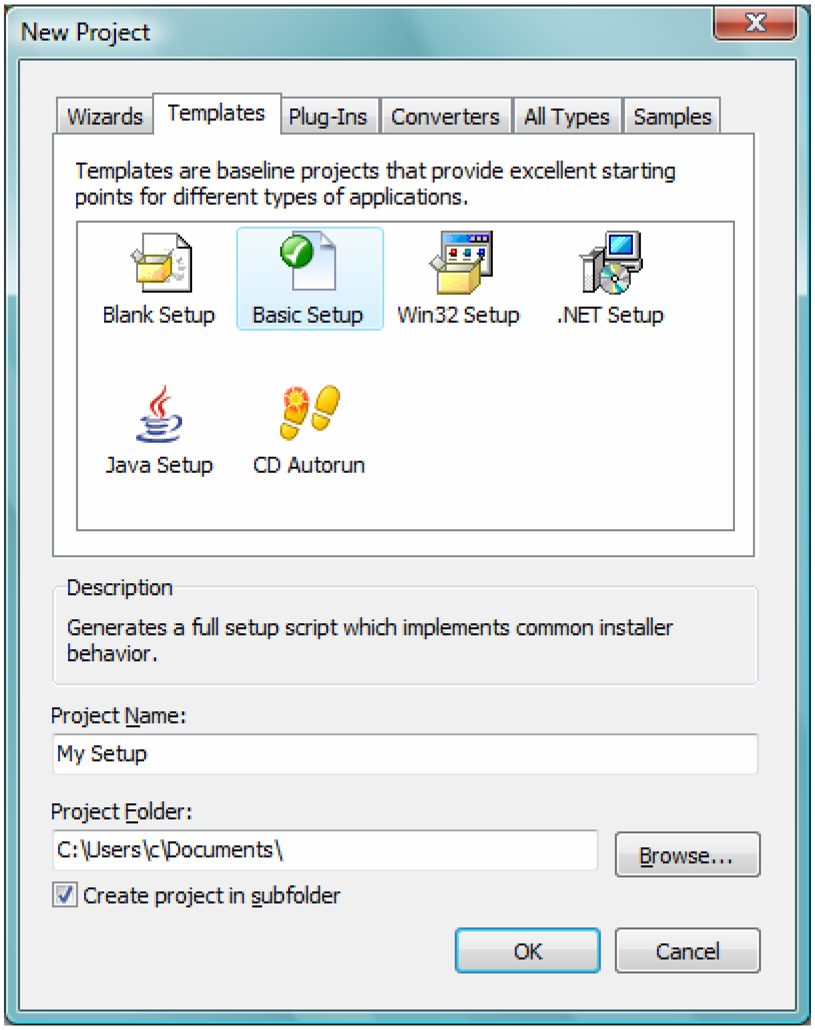
   ```
   Version One:
   Line 1
   ```

2. Now that our first version product files are ready, we can get started building the first version installer. Launch the InstallAware 7 IDE using the Windows **Start Menu**. Then, on the **Project** tab, in the **New** group, click the **Basic Setup** button. The **New Project** window is displayed.

3. The **Basic Setup** project type is pre-selected. Additionally, the **New Project** window lists other types of templates, wizards, sample projects, and project converters.

   The **Basic Setup** project type contains all the plumbing we need to build a fully working installation – with out of the box support for multiple features, application repair and maintenance, uninstall, and also patching, our area of focus in this whitepaper.



Templates are baseline projects that provide excellent starting points for different types of applications.

Description

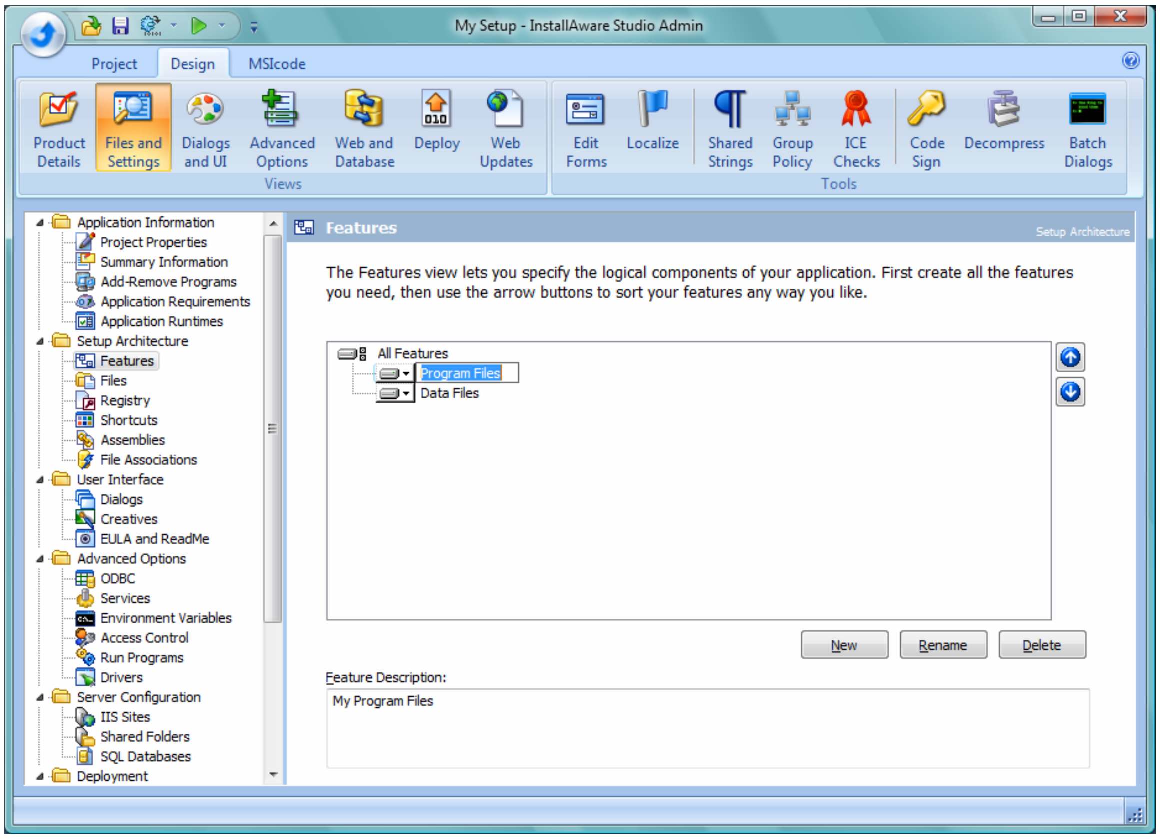
Generates a full setup script which implements common installer behavior.

Project Name: My Setup

Project Folder: C:\Users\c\Documents\

☑ Create project in subfolder

4. Under the **Project Name** field, accept the default value of `My Setup`, or provide your own project name. Projects are normally created under your **My Documents** folder, and reside inside their own subfolders. If you would like to use a different folder, enter that folder name here, or use the suggested value. Click **OK** when you're ready to create the project.



The Features view lets you specify the logical components of your application. First create all the features you need, then use the arrow buttons to sort your features any way you like.

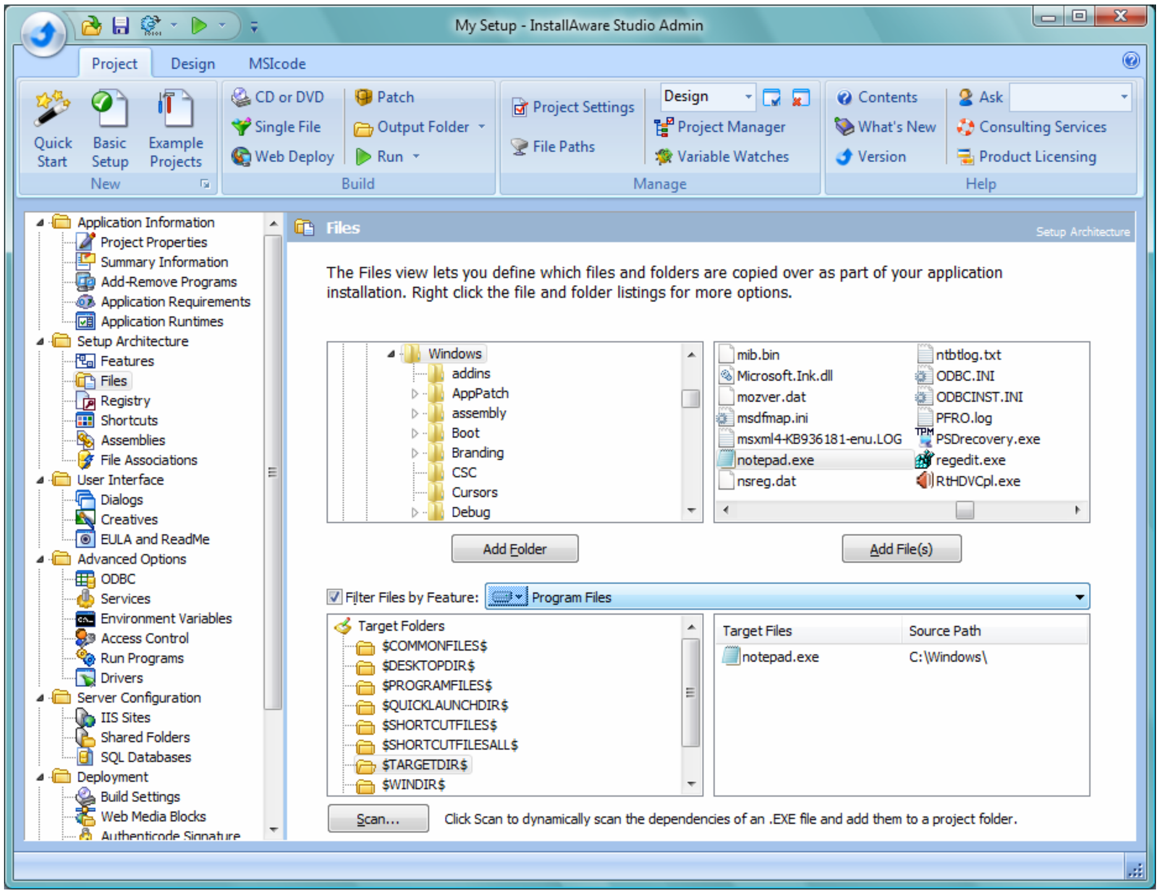
Feature Description: My Program Files

5. Now that our setup project is ready, we'll begin to flesh it out with our products's logical organization. On the **Design** tab, click the **Files and Settings** button. Select the **Features** designer under the **Setup Architecture** heading and define two features, so your setup project looks like the screenshot above. Use the **Rename** button to rename existing features, and the **New** button to insert new features. The **Up-Down** buttons on the top right corner of the design view let you sort your features. You may also customize feature descriptions by highlighting a feature, and then editing its description in the **Feature Description** field.

   When you're finished, select the **Files** designer under the **Setup Architecture** heading to add your actual product files to the setup project.

6. The top section of the **Files** designer lists the files available on your development system. The bottom section of the **Files** designer lists the files that are installed by your setup. Notice the list of variables that are displayed under the `Target Folders` list. `$TARGETDIR$` is a special variable which indicates the destination directory chosen by the user at install time. Select `$TARGETDIR$` before adding any files to make sure they will be installed into the end-user's chosen destination.

The **Filter Files by Feature** check-box lets you associate individual files with your product's logical organization (the features defined in the previous step). After checking this box, be sure to choose the product feature that you wish to associate files with.



The Files view lets you define which files and folders are copied over as part of your application installation. Right click the file and folder listings for more options.

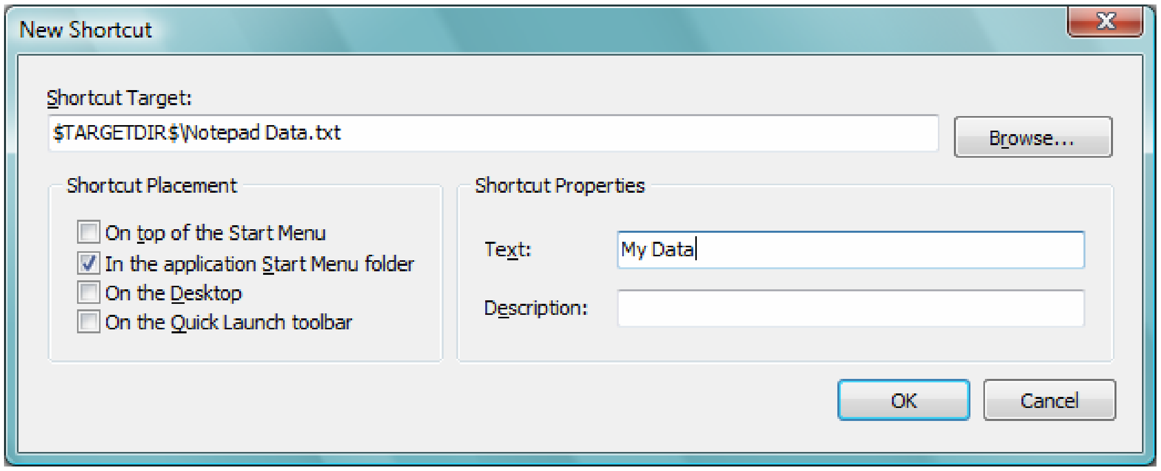
Filter Files by Feature: Program Files

Target Files	Source Path
notepad.exe	C:\Windows\

Click Scan to dynamically scan the dependencies of an .EXE file and add them to a project folder.

From your Windows folder, add the file `notepad.exe` under the `Program Files` feature. Then browse to your Desktop folder, and add the file `Notepad Data.txt` under the `Data Files` feature.

7. Shortcuts will let us easily open installed files and test that the patch is working. Select the **Shortcuts** designer under the **Setup Architecture** heading, and then click **New** to display the **New Shortcut** dialog.



Shortcut Target: $TARGETDIR$\Notepad Data.txt

[ ] On top of the Start Menu
[✓] In the application Start Menu folder
[ ] On the Desktop
[ ] On the Quick Launch toolbar

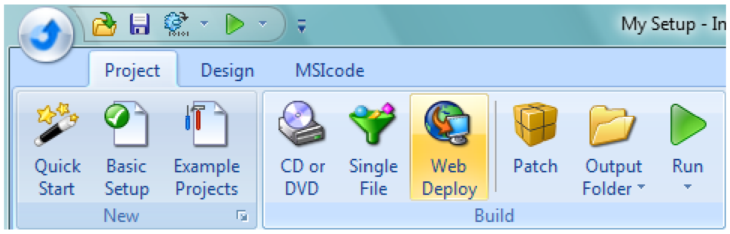
Text: My Data

Description:

It's easy to create shortcuts in a variety of locations using this dialog. Click the **Browse** button to visually locate your shortcut targets from among the files you're installing, or alternately type any file or folder location directly in the **Shortcut Target** field. Be sure to name your shortcut using the **Text** field, and pick at least one location to create the shortcut under from the **Shortcut Placement** group. Click **OK** to save your changes.

Create two shortcuts – one called `My Data` which points to our dummy text file, and another called `My Program` which points to the copy of Notepad that we're installing. We're now ready to build our first version setup!

8. On the **Project** tab, in the **Build** group, click **Web Deploy**. We choose the **Web Deploy** build mode, because when a setup is built this way, it locally caches its setup sources on the target system – eliminating the need for source media prompts that would be occurring with other build modes, such as the **CD or DVD** build mode. It's also possible to build a setup in **Web Deploy** mode without actually deploying anything on the web – we'll revisit this subject in the second half of this guide.



When the build is complete, click the downward arrow on the **Output Folder** button in the **Build** group, and choose the **Web Deploy** location. Explorer opens up with our built setup, with the following files located inside the folder:

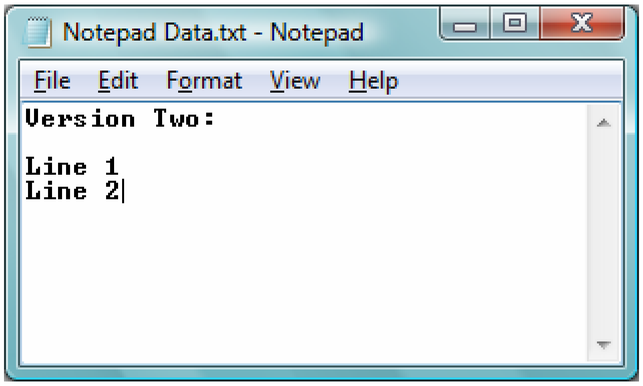
- `My Setup.exe`: This is the main setup executable, which contains the installer engine.
- `Data Files.7zip`: This setup data file contains all the files which were included in the `Data Files` feature.
- `Program Files.7zip`: This setup data file contains all the files which were included in the `Program Files` feature.

9. Create a new folder on your Desktop, calling it `Version One`. Copy all built setup files inside this folder. This folder serves as your backup location for your old version installer, which is all that is needed for building a patch targeting this first version. Old product files or setup sources are not required.

Building the Second (and further) Version Installers

Now that we've successfully built and saved a copy of our first version setup, we're ready to make the changes to the product and setup project for building the second version.

1. Edit the text file on your Desktop created in step 1 of the previous section, and edit its contents so it looks like the screenshot below:



```
Version Two:

Line 1
Line 2
```

2. Save your changes to the text file, and rebuild your setup following the procedure in step 8 of the previous section. Remember to create another folder on your Desktop called `Version Two`, and again copy all built setup files into this folder, backing up the second version installer for use with future patches.

3. If necessary, make further changes to your product files and setup project (such as adding more files), and build further version installers as necessary. Just be sure to backup each built setup, so you can patch them later on.

Building the Patch

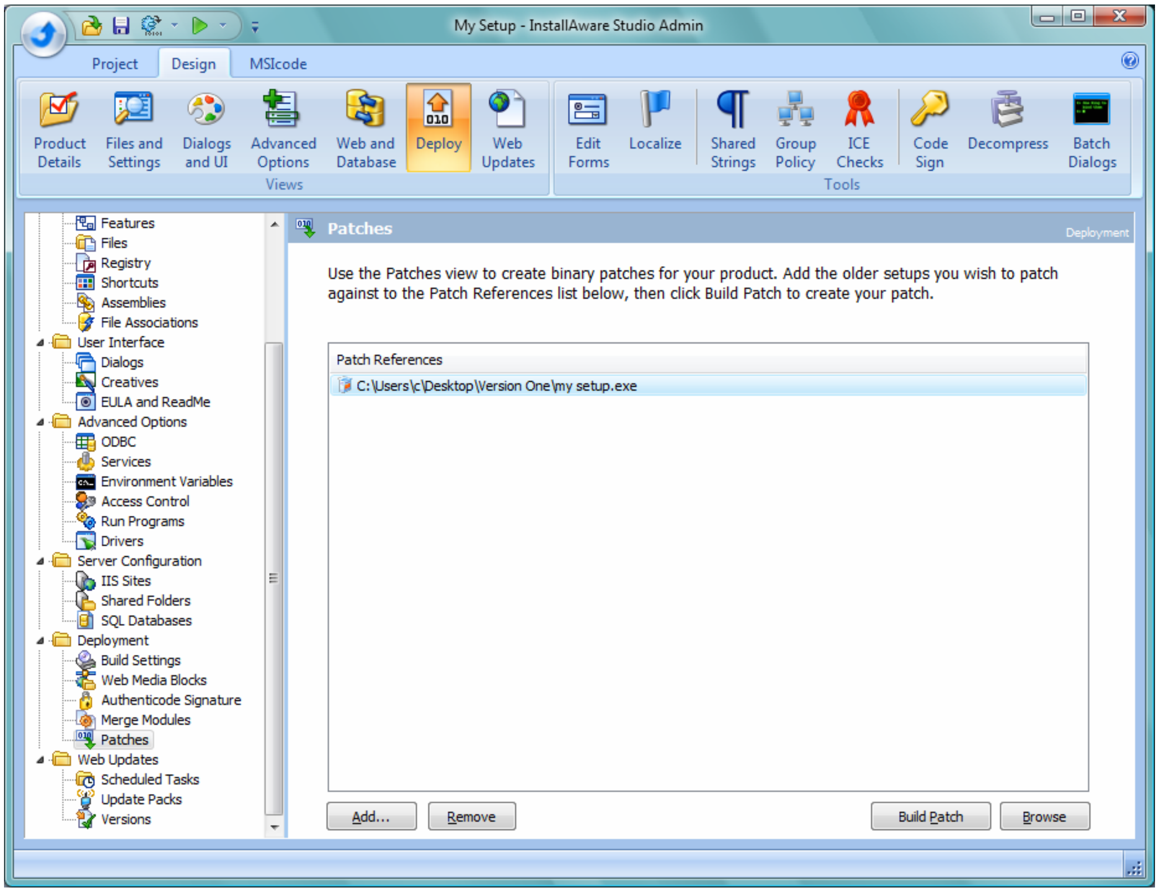
There are absolutely no additional steps required to build a patch, above and over the changes made to your product files and the new version installer. Follow the steps below to create your patch:

1. On the **Design** tab, in the **Views** group, click the **Deploy** button.

2. Under the **Deployment** heading, choose the **Patches** designer. Click the **Add** button in the Patches designer and add the file `My Setup.exe` from the `Version One` folder created earlier on your Desktop.

3. If there are other product versions to patch, add their installers from their backup folders as well. These saved old version installers are all InstallAware needs for creating a patch!

4. Click the **Build Patch** button. Once your patch has been built, click the **Browse** button to open Explorer inside the folder containing your patch file, `My Setup.exe`. This patch will successfully upgrade all referenced old versions to your latest version!



Use the Patches view to create binary patches for your product. Add the older setups you wish to patch against to the Patch References list below, then click Build Patch to create your patch.

Patch References

C:\Users\c\Desktop\Version One\my setup.exe

Congratulations! You have now built a patch for your software product using InstallAware. Install your first version install, open the text file to verify its contents; then install the patch and verify that your text file has been updated. Experiment with making more changes to your setup project, and let InstallAware take care of building more patches for you with a single click!

Deployment Scenarios

Patch Source Resolution

All Windows Installer patches require access to the old version product's setup files. The setup files for the old version must be successfully identified (resolved) before any patches may be applied. This section describes how InstallAware patches handle this source resolution requirement.

Patch Source Resolution for Web Deployed Setups

Since web deployed setups always locally cache their setup files, patch source resolution for web deployed setups is automatic. No manual steps are required during patch source resolution. It is also possible to author web deployed setups such that they do not require an Internet connection at any time during installation; these advantages make web deployed setups ideal for patching in InstallAware.

Patch Source Resolution for CD or DVD Setups

If it is undesirable to cache setup sources locally, CD or DVD setups may also be used. When patching a CD or DVD setup, a dialog box will ask for the original setup files. At this stage, the end-user simply needs to insert the original installation media, and select the drive containing the setup sources. If a CD or DVD setup was not stored on a removable medium but resides on a hard drive folder, simply selecting that folder will again suffice during patch source resolution.

Patch Source Resolution for Single File Setups

A single file setup packs all setup files into a single self extracting file. This self extracting archive, when run, first expands all setup files into a temporary folder, and then initiates setup. When patching against these setups, therefore, it is not sufficient to point to the folder containing the single self extracting file. Instead, the self extracting file should be run, allowing it to complete to the stage of extracting its setup files to a temporary folder, and then that temporary folder should be chosen. Identifying this folder can be a little daunting for novice end-users, therefore single file setups are not recommended as patch targets; however if necessary it is still possible to patch single file setups.

A practical step by step method to locate the temporary folder where setup files are extracted into is as follows:

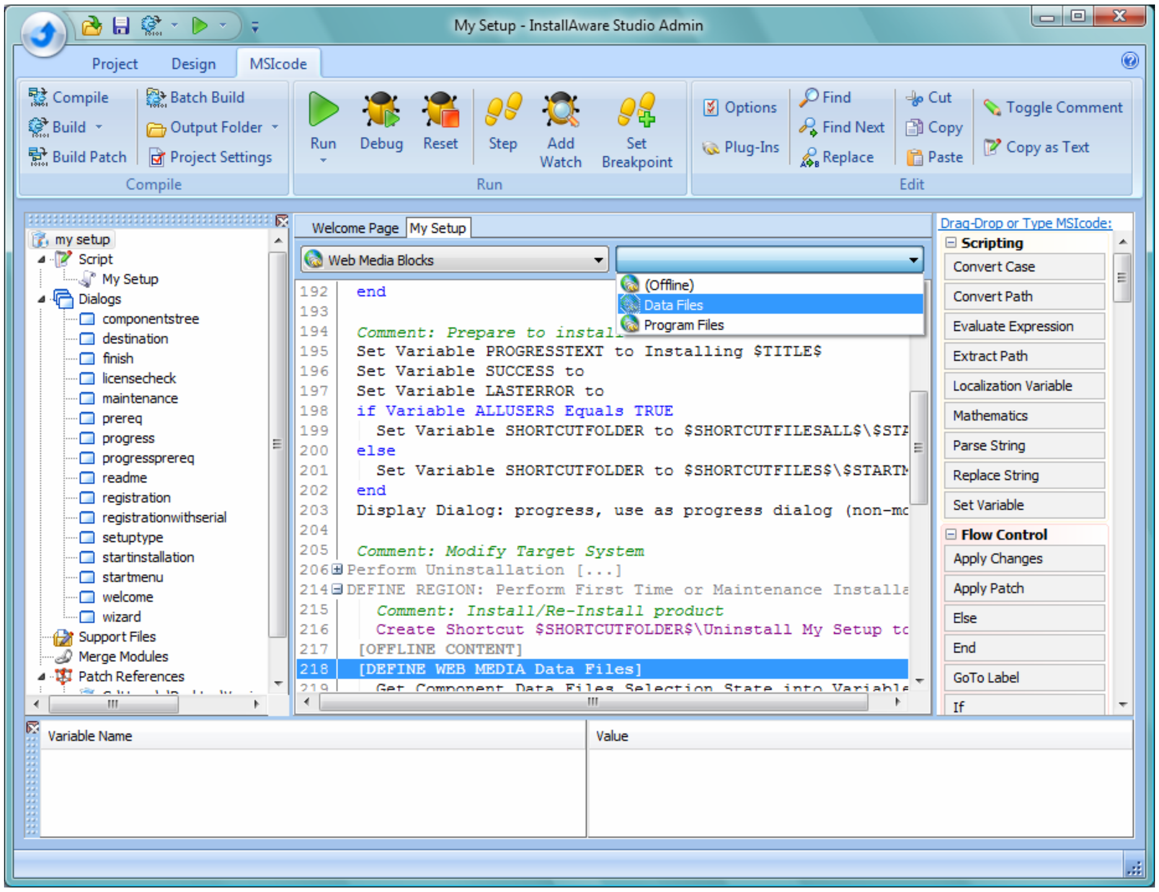
1. Click the Windows **Start** button, choose **Run**, and type in `%temp%` into the **Open** field. Then click **OK**.
2. Windows Explorer opens, displaying the contents of the user's temporary folder.
3. Look for a folder that starts with the characters `mia` and ends with the characters `.tmp`, with some random numbers and letters in between. In the rare event that there are multiple folders fitting this pattern, simply try each folder until the patch installer accepts one of your choices (the patch installer will check to make sure valid sources are specified).

The main advantage of using a single file setup is to create a monolithic package that contains everything needed to install an application without requiring an Internet connection or a CD/DVD (and other types of removable media). It is possible to build web deployed setups which are still monolithic and do not require access to the Internet at any time during installation, so in cases where a monolithic single file setup is desired, the Web Deploy build mode can be preferred over the Single File build mode. Patch source resolution is automatic with all web deployed setups, and the output of a setup built using the Web Deploy build mode can still be a single physical file, as described in the following section.

Building Monolithic Web Deployed Setups

Since web deployed setups are the easiest base setups to patch against, it is desirable to use them when deploying your full version installers. In certain situations you may want to build setups which do not require an Internet connection at any time. InstallAware makes it easy to build monolithic web deployed setups – that are fully self-contained – so they always cache setup files, contain all setup data in a single setup file, and are easy for your end-users to patch against. To convert any existing setup to a monolithic web deployed setup, follow these steps:

1. Open your setup project in the InstallAware IDE, and choose the **MSIcode** tab to access the full sources of your installation.
2. Inside the **MSIcode** editor, choose the tab immediately to the right of the **Welcome Page** tab. For our hypothetical setup, this tab is called `My Setup`.
3. Notice the two drop-down combo boxes immediately above the **MSIcode** script. Select Web Media Blocks in the left combo box. Then expand the right combo box to display a list of all your Web Media Blocks (parts of your setup that are downloaded from the Internet at runtime).

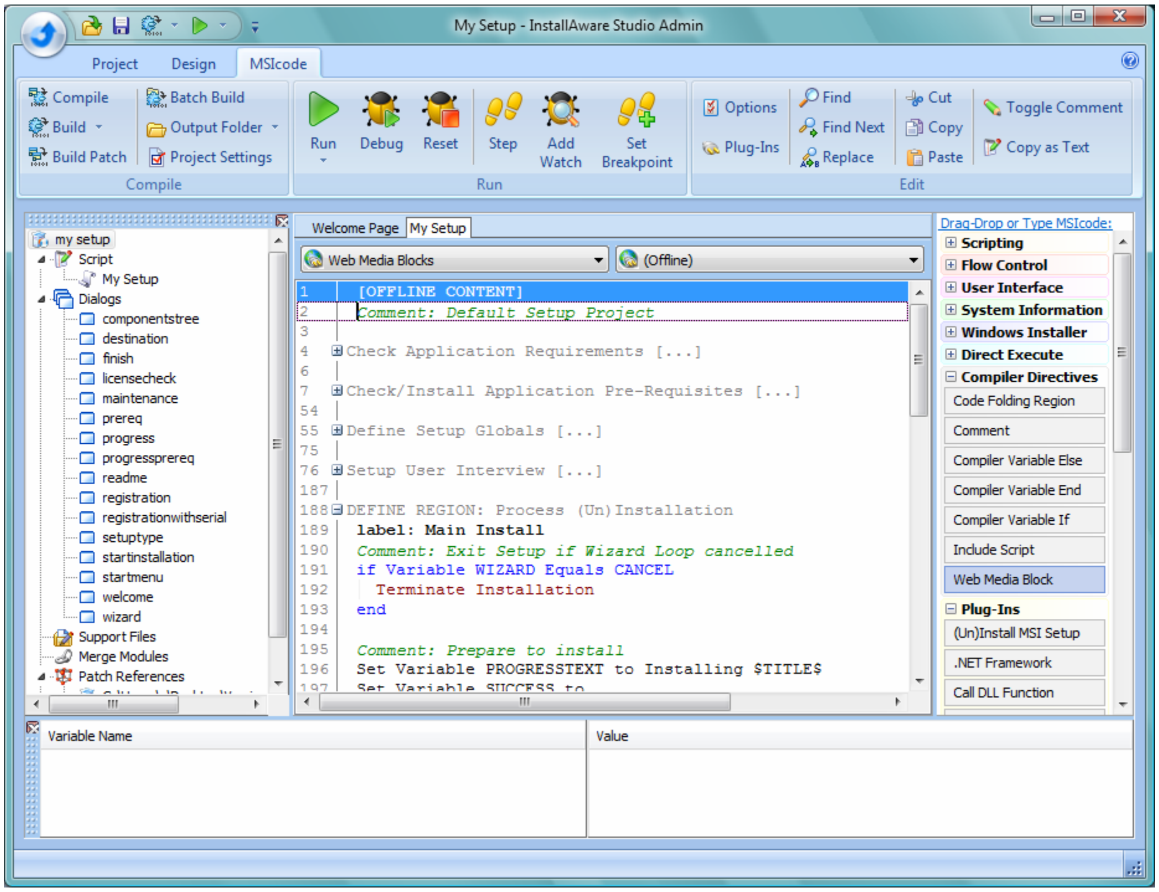


```
end

Comment: Prepare to instal
Set Variable PROGRESSTEXT to Installing $TITLE$
Set Variable SUCCESS to
Set Variable LASTERROR to
if Variable ALLUSERS Equals TRUE
  Set Variable SHORTCUTFOLDER to $SHORTCUTFILESALL$\$STA
else
  Set Variable SHORTCUTFOLDER to $SHORTCUTFILES$\$STARTM
end
Display Dialog: progress, use as progress dialog (non-mc

Comment: Modify Target System
⊞Perform Uninstallation [...]
⊟DEFINE REGION: Perform First Time or Maintenance Installa
  Comment: Install/Re-Install product
  Create Shortcut $SHORTCUTFOLDER$\Uninstall My Setup tc
[OFFLINE CONTENT]
[DEFINE WEB MEDIA Data Files]
  Get Component Data Files Selection State into Variable
```

4. Select a Web Media Block from the open combo box. The **MSIcode** editor positions the cursor at the exact location in the script where the Web Media Block is defined.
5. Right-click the highlighted Web Media Block line, and choose Comment Out/In. This comments out the Web Media Block definition, directing the InstallAware setup compiler to ignore this directive.
6. Repeat steps 4-5 until all Web Media Blocks in the **MSIcode** script have been commented out.
7. Press the **Page Up** key repeatedly until you are on top of the installation script.
8. Using the mouse, drag-drop the Web Media Block command from the MSIcode command palette (on the right of the MSIcode editor) to the top of your script. This inserts a new Web Media Block command into the MSIcode script.



```
[OFFLINE CONTENT]
Comment: Default Setup Project

Check Application Requirements [...]

Check/Install Application Pre-Requisites [...]

Define Setup Globals [...]

Setup User Interview [...]

DEFINE REGION: Process (Un)Installation
  label: Main Install
  Comment: Exit Setup if Wizard Loop cancelled
  if Variable WIZARD Equals CANCEL
    Terminate Installation
  end

  Comment: Prepare to install
  Set Variable PROGRESSTEXT to Installing $TITLE$
  Set Variable SUCCESS to
```

9. The **Define Web Media Block** dialog box appears. Without making any changes in this dialog, simply click the **OK** button. This inserts a new line into your MSIcode script called `[OFFLINE CONTENT]`. This instructs the InstallAware setup compiler to include each file of your setup package directly inside the main setup.exe file, instead of an Internet location where they will be downloaded from at runtime.

Building your setup in the **Web Deploy** build mode will now create a monolithic web deployed setup, which does not require access to the Internet at any time during installation, providing the easiest targets to patch against thanks to automated patch source resolution.

Customizing Patches

Compiler Variables

InstallAware features Compiler Variables, which offer an easy way to build multiple setup files from a single setup project. Using Compiler Variables, which are similar to conditional directives found in most programming languages, you may conditionally include/exclude parts of the MSIcode script, and build different flavors of an existing setup, all based on a single setup project.

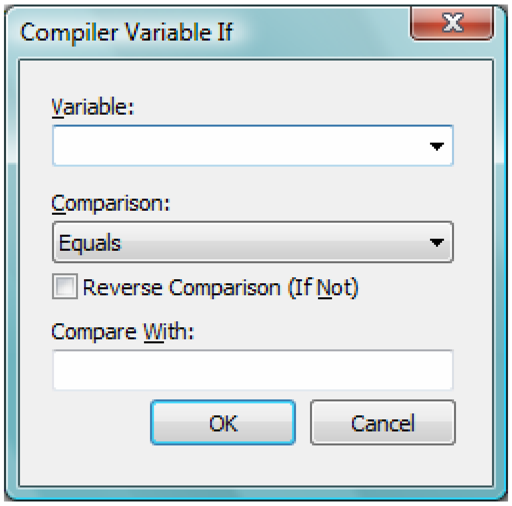
Built-In Compiler Variables

InstallAware provides various built-in Compiler Variables. For instance, the internal `BUILDMODE` Compiler Variable evaluates to `PATCH` when building patches in InstallAware. This makes it very easy to add additional logic to your MSIcode script, that is only compiled during a patch build (for instance, when clicking the **Patch** button in the **Build** group on the **Project** tab). Additional, custom Compiler Variables may be defined using the **Project Options** window (accessible by clicking the **Project Settings** button in the **Manage** group on the **Project** tab). Compiler Variable values can also be set from the command line when using the `miabuild.exe` command line build tool, for added convenience.

Conditional Compilation using Compiler Variables

Three MSIcode commands offer the possibility to test the values of Compiler Variables, and selectively include/exclude parts of the MSIcode script at build time. These commands are found on the **Compiler Directives** group of the MSIcode command palette (accessible on the **MSIcode** tab, to the right of the MSIcode editor):

- **Compiler Variable If**: Opens a new conditional compilation block. Simply drag-drop this command into your MSIcode script to create a new conditionally compiled block.



In the **Variable** field of this command, enter the Compiler Variable to test for. You may test for both built-in and custom Compiler Variables using this command. To start a new conditionally compiled block included only when building patches, enter `BUILDMODE` in this field.

In the **Comparison** field, choose a type of comparison to perform. Accept the default comparison of `Equals` for testing patch builds. If you need to reverse the type of comparison, check the **Reverse Comparison (If Not)** check-box.

In the **Compare With** field, enter the literal value to test the value of the Compiler Variable against. For patches, enter `PATCH` in this field.

- **Compiler Variable Else**: Branches a previously started conditional compilation block, such that if the immediately preceding **Compiler Variable If** test failed, the MSIcode commands following the **Compiler Variable Else** command are conditionally compiled.

- **Compiler Variable End**: Closes a conditional compilation block previously opened with a **Compiler Variable If** command. Each **Compiler Variable If** command must have a corresponding **Compiler Variable End** command.

You may freely nest multiple conditional compilation blocks as you see fit. You may also use any kind of MSIcode commands within your conditionally compiled blocks.

InstallAware internally uses Compiler Variables to make One-Click Patching possible. Most InstallAware templates (including the **Basic Setup** template used in this whitepaper) provide a properly constructed MSIcode script with all necessary **Compiler Variable If/Else/End** commands that provide effortless patch construction from the same setup project used for the main setup.

Using Compiler Variables in this way provides out-of-the-box One-Click Patching convenience for developers, while retaining developer flexibility and permitting additional customizations to patches being built based on a single setup project. You may freely customize any of the InstallAware templates to suit your requirements and embed any kind of business logic to your installations for patching, while avoiding the problems and overhead associated with having to create and maintain new, separate patch projects from scratch.

Additional Resources

Please visit the InstallAware website publications section at the following URL for more information on InstallAware technologies:

http://www.installaware.com/home/publications.asp

Whitepapers providing an in-depth analysis of InstallAware's scripting and web deployment technologies, as well as a Reviewer's Guide, are available at the above URL.

About InstallAware Software Corporation

Focusing solely on the Windows Installer (MSI) platform for Microsoft Windows operating systems, InstallAware is the premier provider of software installation tools for Internet deployment. Founded in 2003, InstallAware is supported by thousands of users worldwide, and is a Borland Integrated Partner. For more information, visit www.installaware.com.