

Using InstallAware 7

To Deploy Microsoft .NET Framework Applications

August 2007



The information contained in this document represents the current view of InstallAware Software Corporation on the issues discussed as of the date of publication. Because InstallAware must respond to changing market conditions, it should not be interpreted to be a commitment on the part of InstallAware, and InstallAware cannot guarantee the accuracy of any information presented after the date of publication.

This guide is for informational purposes only. INSTALLAWARE MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

InstallAware may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from InstallAware, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2007 InstallAware Software Corporation. All rights reserved.

InstallAware, MSIcode, Genuine Scripting for Windows Installer, WebAware Installations, Web Media Blocks, Partial Web Deploy, and One-Click Patching are either registered trademarks or trademarks of InstallAware Software Corporation in the United States and/or other countries.

Microsoft, Windows, and other Microsoft product names are trademarks or registered trademarks of Microsoft Corporation in the U.S. and other countries. All other marks are the property of their respective owners.

TABLE OF CONTENTS

OVERVIEW	1
CREATING A .NET PROJECT	2
DEPLOYMENT SCENARIOS	25
Independent Software Vendors (ISVs)	25
Isolate Application Runtimes	25
Secure Installers with Authenticode Code Signing	25
Protect Application Features	25
Deploy Unlimited Web Updates	25
Improve the End-User Experience	25
Leverage One-Click Incremental Patching	26
Build Logo Certifiable Installers	26
Rapidly Customize Installers for Product Editions	26
Communicate with End-Users	26
Enterprise Information Technology Scenarios	27
Allow Partners to Customize Their Own Installers	27
Lock Down Installers to Only Run Inside a Network	27
Use Silent Installs for Automated Software Distribution	27
Group Policy Deployment	27
Shell to and Repackage Installs	28
Allow Non-Administrative Installs	28
Use Advertising for Install-On-Demand and Self-Healing	28
ADDITIONAL RESOURCES	29
ABOUT INSTALLWARE SOFTWARE CORPORATION	29

Overview

This whitepaper describes how to use InstallAware for deploying applications running on the Microsoft .NET Framework. Although the .NET Framework makes deploying applications easier thanks to features like the Global Assembly Cache, which eliminates shared resource conflicts often encountered by Win32 applications, there are still unique challenges to the deployment of .NET applications. One significant challenge is the download and installation of the .NET Framework, which requires considerable bandwidth and increases the setup package size dramatically. InstallAware addresses this challenge using a combination of unique deployment technologies, including WebAware Installations, superior compression, and MSIcode scripting.

InstallAware's unique WebAware Installation technology is used to isolate application runtimes from the main setup program, while still providing a self contained installer that works without mandating an Internet connection. This technology works well with the .NET Framework runtime, greatly simplifying deployment by downloading and installing the runtime only when called by the setup.

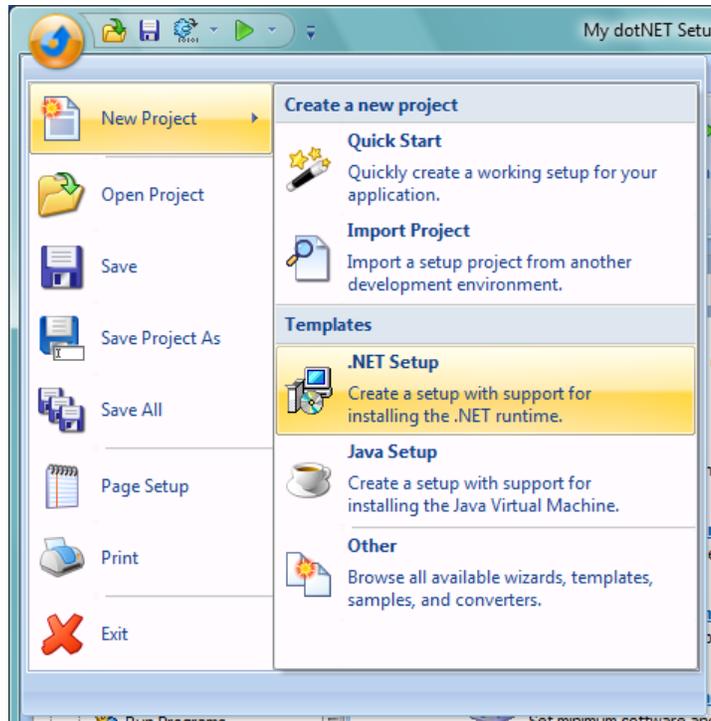
Another key InstallAware technology is the superior LZMA/BCJ2 compression, which pre-processes data using call jump converters to increase compressibility, and then compresses the optimized data streams using a very large 32 MB data buffer. The LZMA/BCJ2 algorithm is so efficient that it reduces the size of the .NET Framework to literally half its already compressed size.

While offering new and unique technologies, InstallAware is completely standards based — every InstallAware setup is a pure Windows Installer (MSI) package 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 MSI packages the only accepted software distribution format for large enterprises.

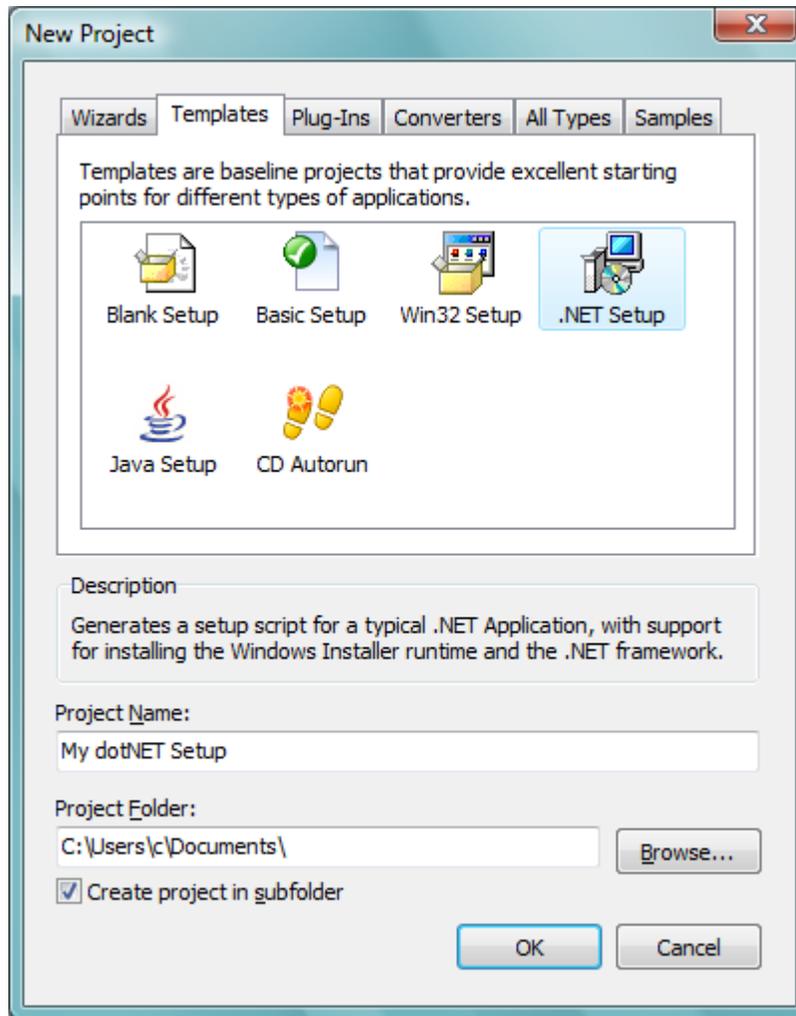
Creating a .NET Project

To create a simple .NET Project using InstallAware, perform the following steps.

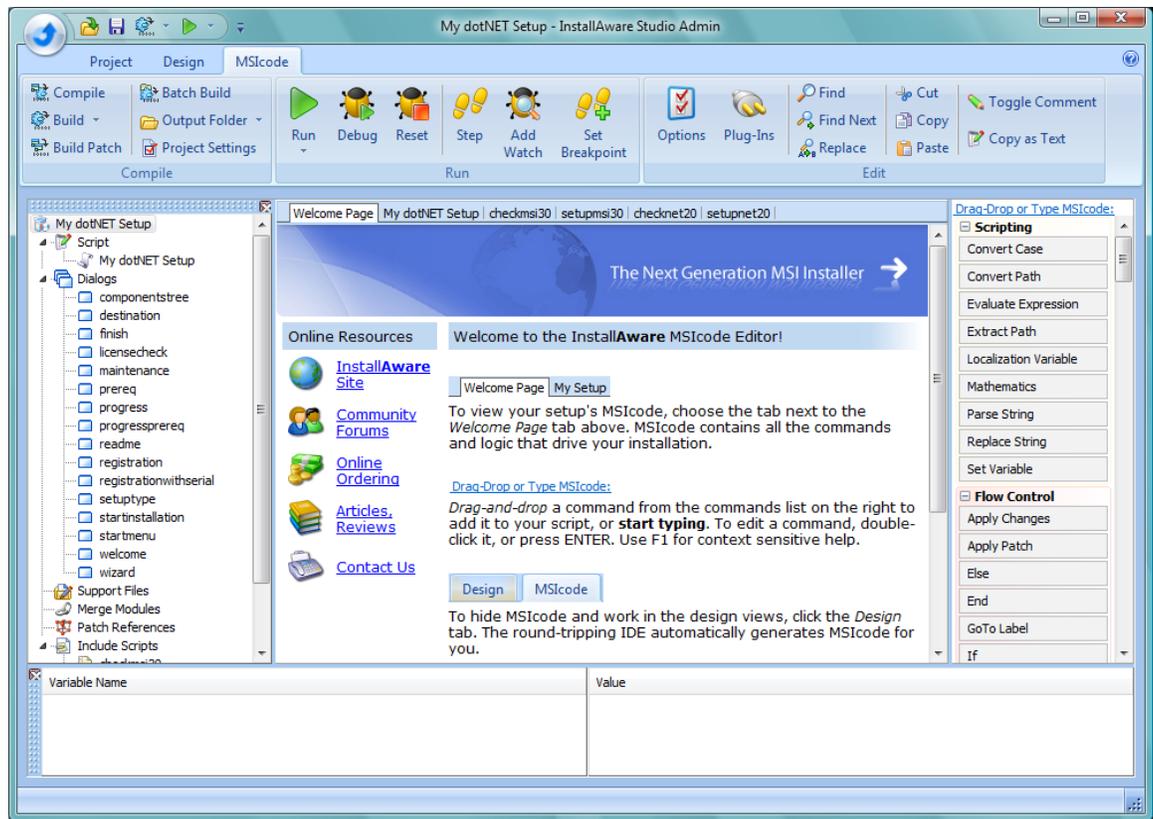
1. Click the InstallAware button on the top left corner of the IDE window, then choose **New Project** ► **.NET Setup** on the application menu. The **New Project** window is displayed.



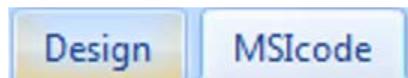
2. The **.NET Setup** project type is pre-selected. Additionally, the **New Project** window lists other types of templates, wizards, sample projects, and project converters.



3. Under the **Project Name** field, accept the default value of **My dotNET 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.



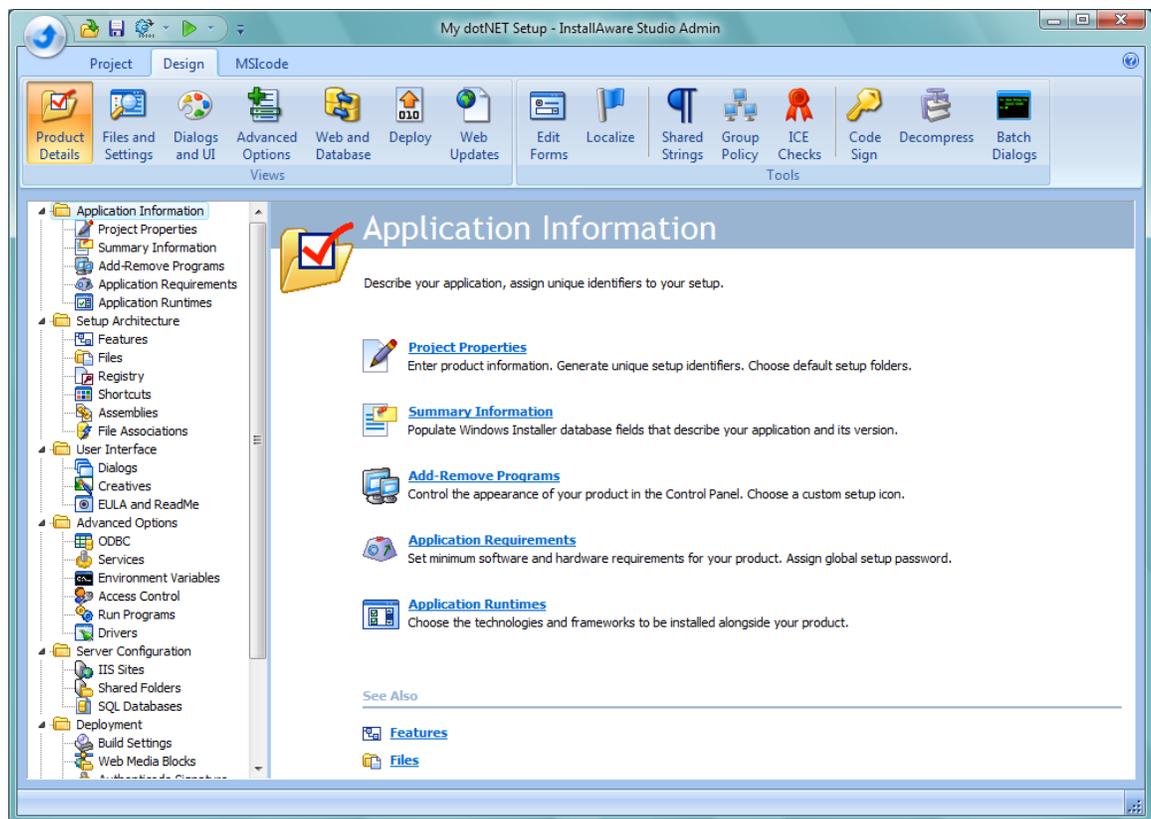
4. The IDE has now created a new setup project from the chosen template, and displays the **Welcome Page** in the **MSICode** tab. Various other pages are available allowing you to view the complete source code of our setup project. You can take a look at these source codes later, once the basic setup has been built. To get started, choose the **Design** tab switch to the **Design Views** which you'll be using to create your setup.



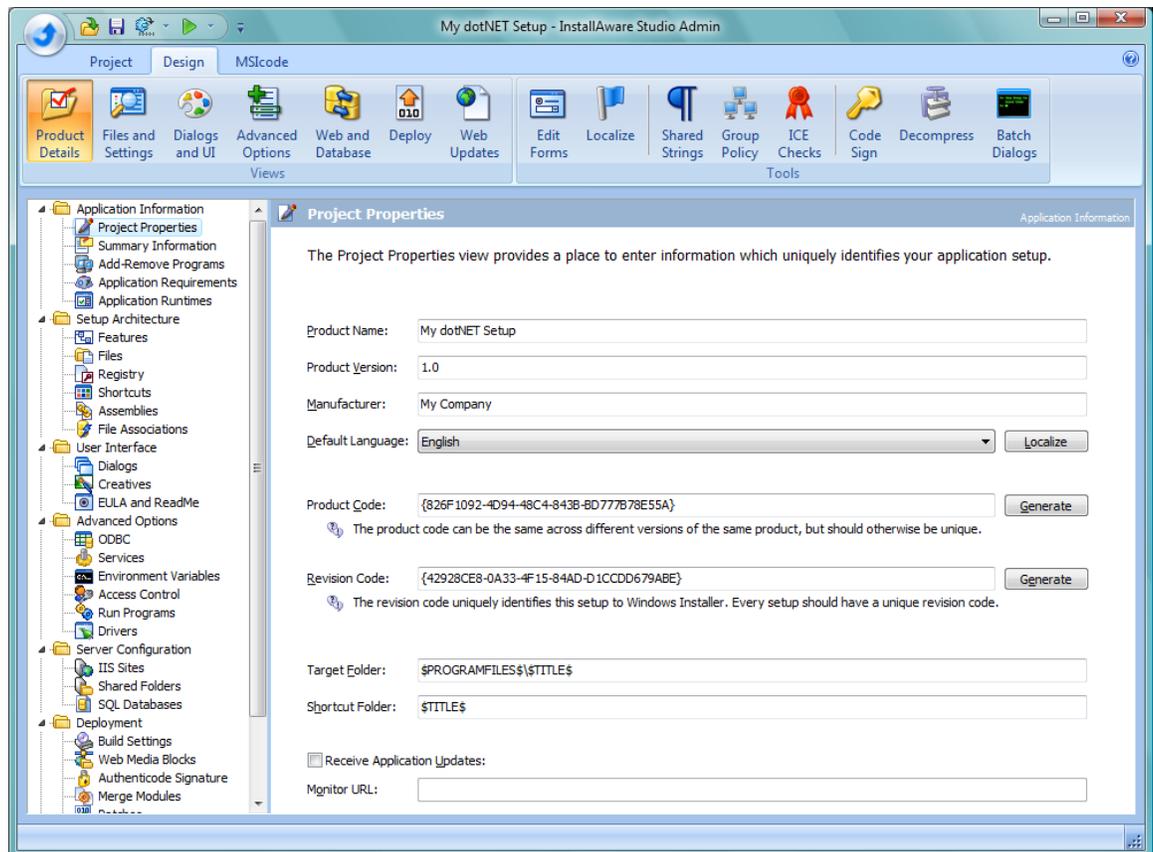
5. The **Application Information** designer appears, providing a summary of the various designers that have been categorized under this heading. In the InstallAware IDE, each designer is categorized under a particular heading for easy access. All available headings and designers are listed on the tree view at the left of the IDE, and also in the **View** group of the on the **Design** tab. Clicking a heading or a designer switches to that design view. Designers perform a variety of installation tasks – from basic steps like copying files and creating registry keys, to complex operations like installing application runtimes, hardware drivers, running SQL scripts, and more. As you work with the designers, they seamlessly add to and update the underlying **MSIcode** sources of your setup. Remember – the designers provide a rapid way to prototype the basic behavior of your setup, but you are not restricted to what they offer. The full power and richness of the **MSIcode** scripting language is always just one click away.

Take a moment to review the **Application Information** heading. It provides links to the designers it contains, along with their brief descriptions. The **See Also** section provides links to related designers under other headings.

When you're ready, click the **Project Properties** link to access the first design view we'll be working with.



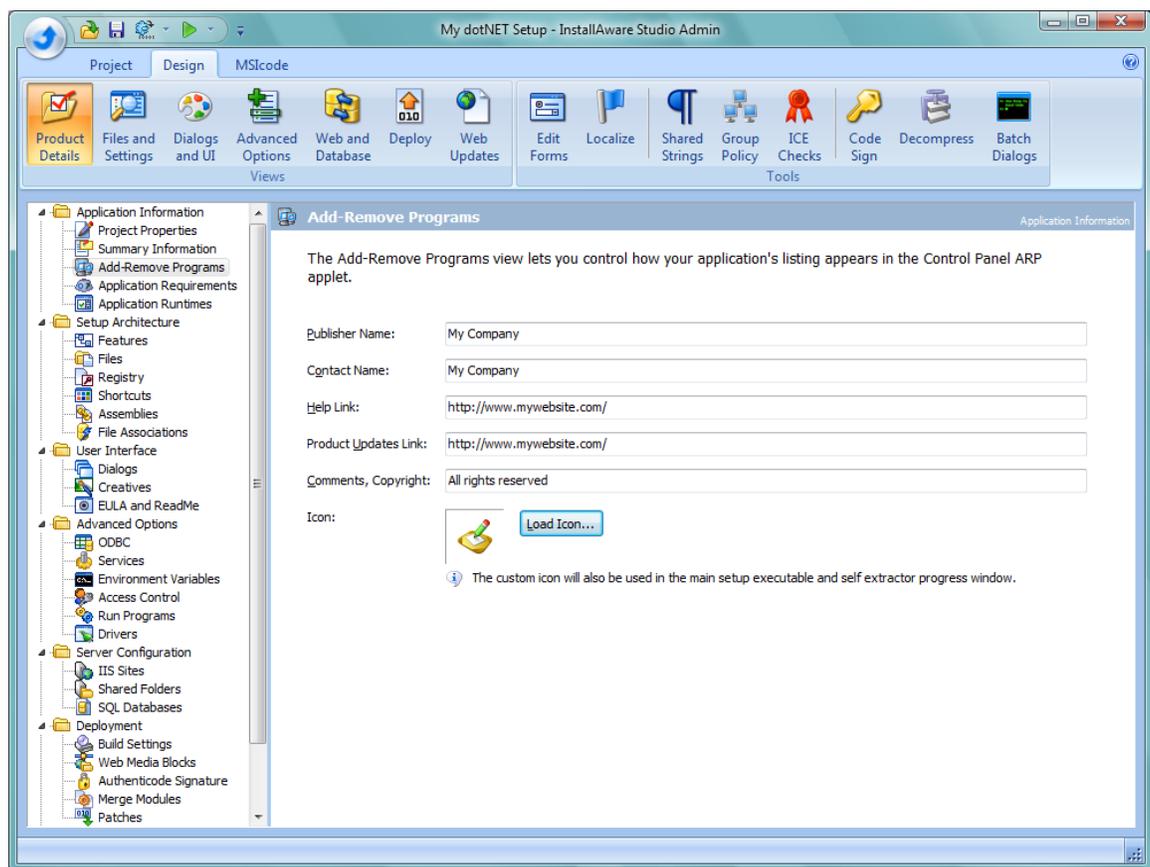
6. The **Project Properties** designer allows you to make high level changes to the installation, such as choosing the default **Target Folder**, and the default **Shortcut Folder**. The `$PROGRAMFILES$` variable is displayed in the **Target Folder** field. It represents the location of the **Program Files** directory on the destination computer. InstallAware denotes variables by enclosing them within dollar signs, in the form `$VARIABLE$`. Variables are defined and modified in the script that remains accessible from the **MSIcode** view.



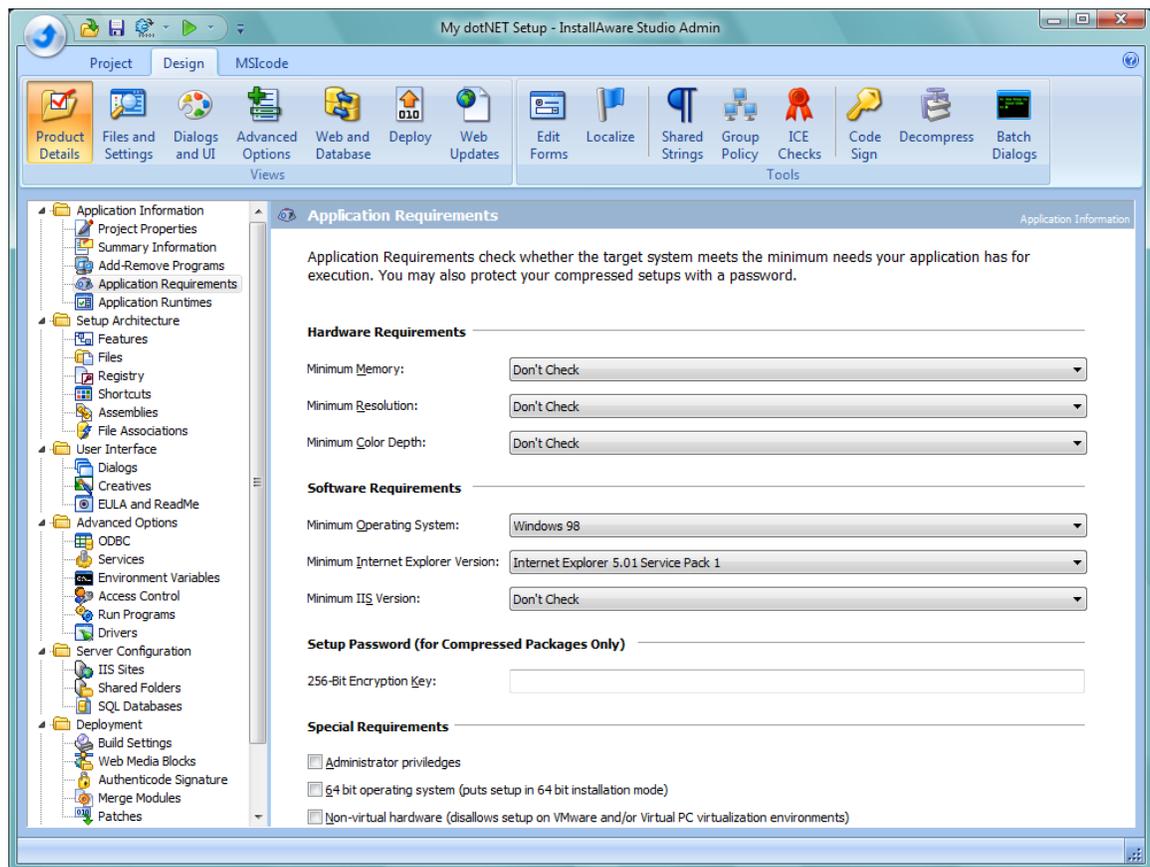
In InstallAware 7, the **Receive Application Updates** check-box lets you serve unlimited application updates to your user base after you ship your product. If you are interested in this functionality, check this box and choose the **Web Updates** heading for more information.

Also remember that help in the InstallAware IDE is never more than one button away. At any time you are confused or need more information, just press **F1**. Context sensitive help guides you through the current design view, **MSIcode** command, or whatever else you may be doing at the moment.

- The remainder of the design views may be accessed by clicking their names on the left of the main IDE window. You may skip the **Summary Information** designer for this project. The summary information is recorded directly into the MSI database (Windows Installer package) that will be created by InstallAware, and is visible in Windows Explorer when the MSI file is right-clicked. You may wish to update the default values provided by the .NET setup template to ensure that they accurately reflect your company and product information.
- Now click **Add-Remove Programs** to switch to this designer which customizes how the application is shown in the **Control Panel | Add-Remove Programs** applet.

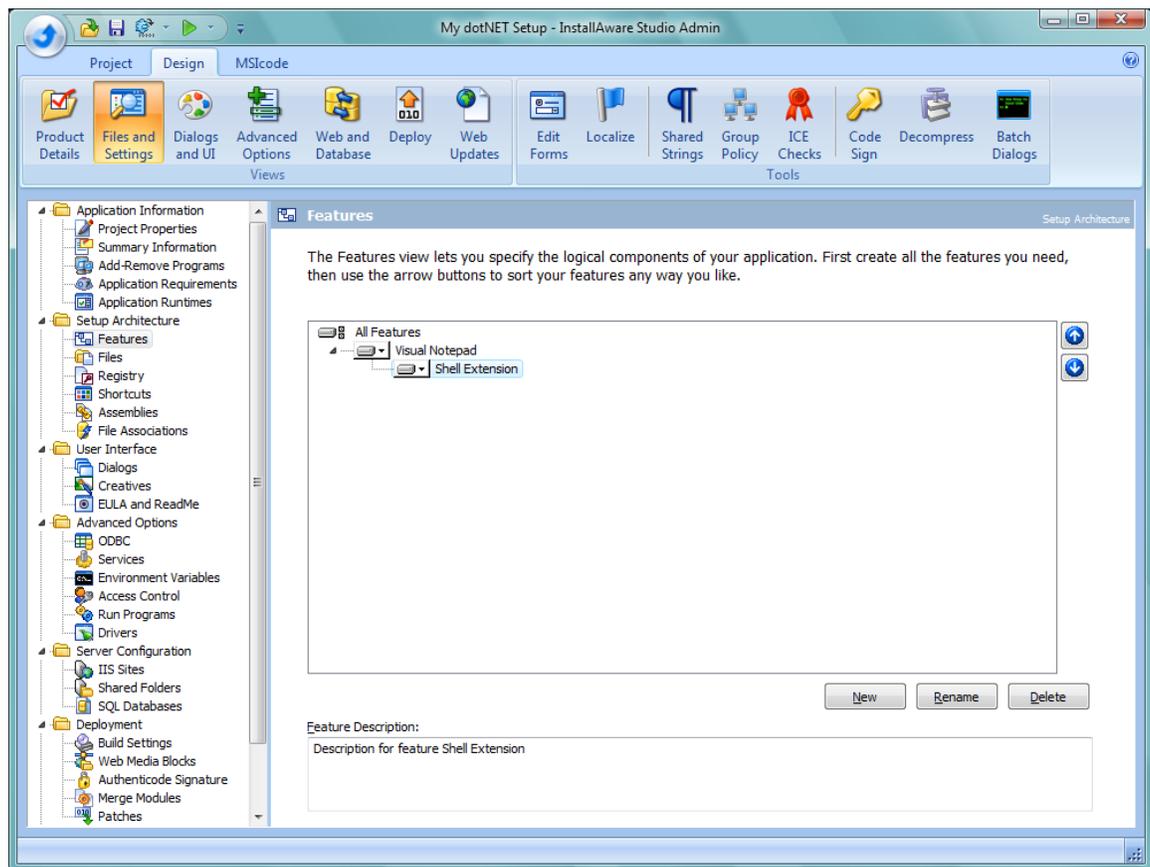


You may also customize the icon of the setup package here. Click the **Load Icon** button to load any icon that represents your application. You are free to use icons of any color and pixel depth that are supported by Windows. Edit the other fields as necessary to reflect your company and website information.

9. Click **Application Requirements**.

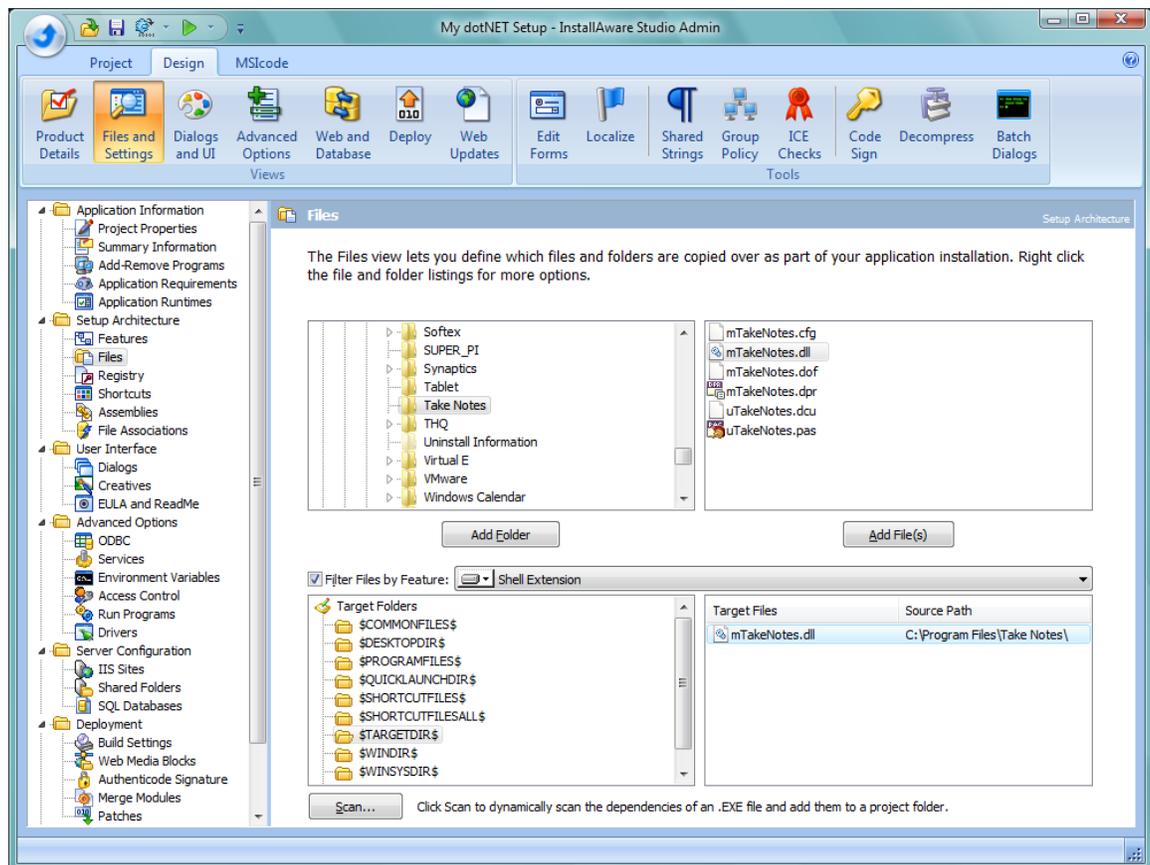
The **Application Requirements** designer helps you enforce any run-time hardware or software requirements that your application may have. Windows 98 has already been selected as the **Minimum Operating System**, since the .NET Framework is not supported on Windows 95. Internet Explorer 5.01 Service Pack 1 has been selected under **Minimum Internet Explorer Version**, which is another requirement of the .NET Framework. If your .NET application has other requirements — such as memory, resolution, color depth, or IIS version, set them here. You may also encrypt your entire setup payload with a global password on this screen. Simply provide a password for the **Setup Password** field, and InstallAware encrypts your application files and data using strong 256-bit AES encryption. In InstallAware 7, the **Special Requirements** section lets you configure additional options for your setup. You may force your application to install only on real hardware – disallowing instances on Virtual PC or VMware virtual machines. You may put setup in native 64 bit installation mode, requiring native 64 bit hardware. You may also check **Administrator Privileges**, if your application requires an administrator account to run successfully. However, InstallAware installations work without administrator privileges, even if they copy files to protected system folders.

11. Click **Features**. This designer takes us under the **Setup Architecture** heading.



The **Features** designer lets you define the logical components of your product, as will be seen by end-users when they are installing your product. For the hypothetical application here, two features have been defined: **Visual Notepad**, corresponding to the main application, and **Shell Extension**, an optional sub-feature of **Visual Notepad**.

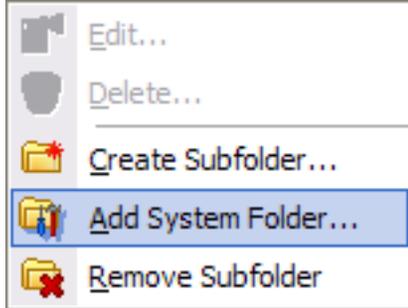
Define the logical components of your .NET application here, along with their descriptions. You may also use the Up/Down arrow buttons to reorder the appearance of your setup features. If your application does not have any configurable features, simply skip this designer. In that case, you may want to remove the pre-defined feature (called **My dotNET Setup** in our template project), and uncheck the feature selection and setup type dialogs (called **componentstree** and **setuptype**) in the **Dialogs** designer, so they are not shown during the installation wizard sequence.

12. Click **Files**.

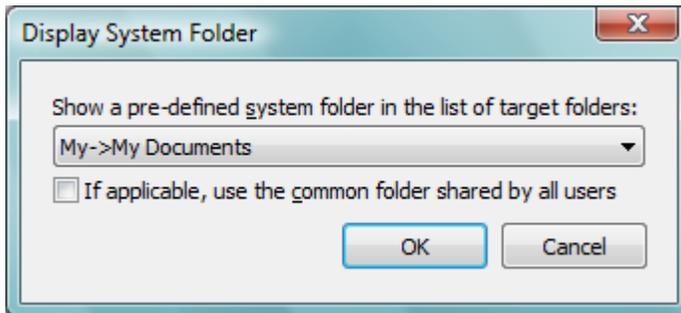
Files may be added to a particular feature of your setup, or unconditionally to the entire installation. Check the **Filter Files by Feature** check-box to activate the feature selection combo-box on this screen. Then choose **Feature Independent** to unconditionally install files, or select the particular feature you wish to add files into.

To add an entire folder to your setup, browse to that folder using the directory browser, and click **Add Folder**. To add files, first browse into the folder containing the files to add, and then highlight all the files to add, clicking **Add Files** when ready. In InstallAware 7, you may also drag-drop files directly from Windows Explorer into any folder to install them there.

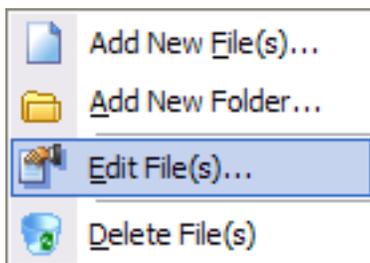
Notice the list of variables that are displayed under the **Target Folders** list. `$TARGETDIR$` is a special variable which indicates the destination directory chosen by the user at install time. You may also choose another pre-defined location, such as `$WINSYSDIR$` for the system directory or `$WINDIR$` for the Windows directory.



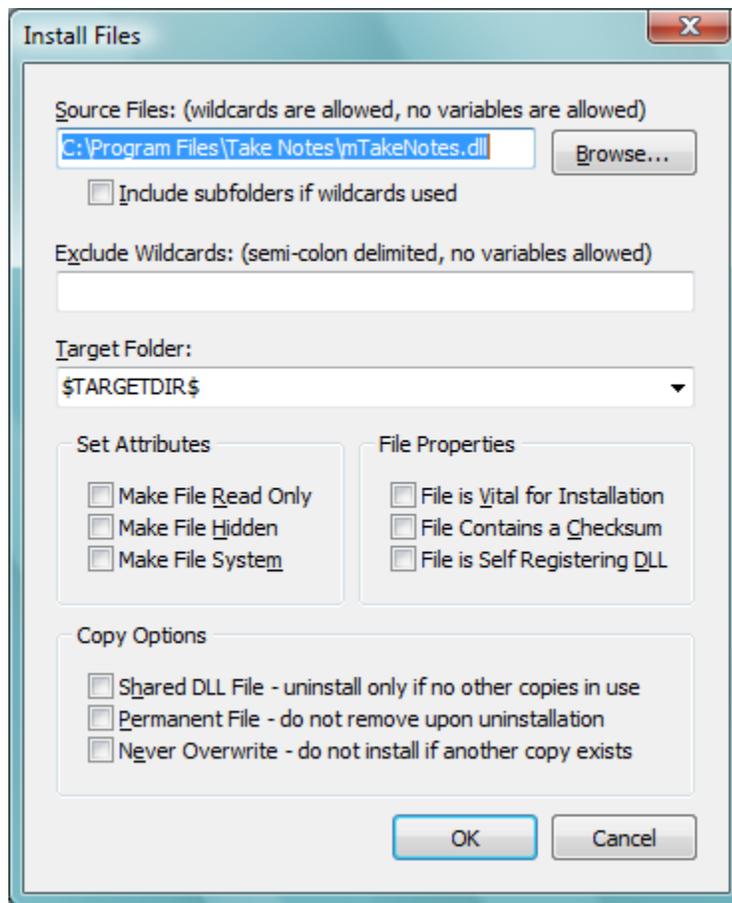
You may display more pre-defined folder locations on this list. Right-click the view and choose **Add System Folder**. Then, choose the special folder to add, such as the **My Documents** folder of the end-user. This way, you may directly install files to a variety of pre-defined system folders on the target machine.



Once you've added some items to the **Target Files** listing, it's easy to make further changes. Just double-click the item to update, or highlight multiple items, right-click, and choose **Edit Files**.



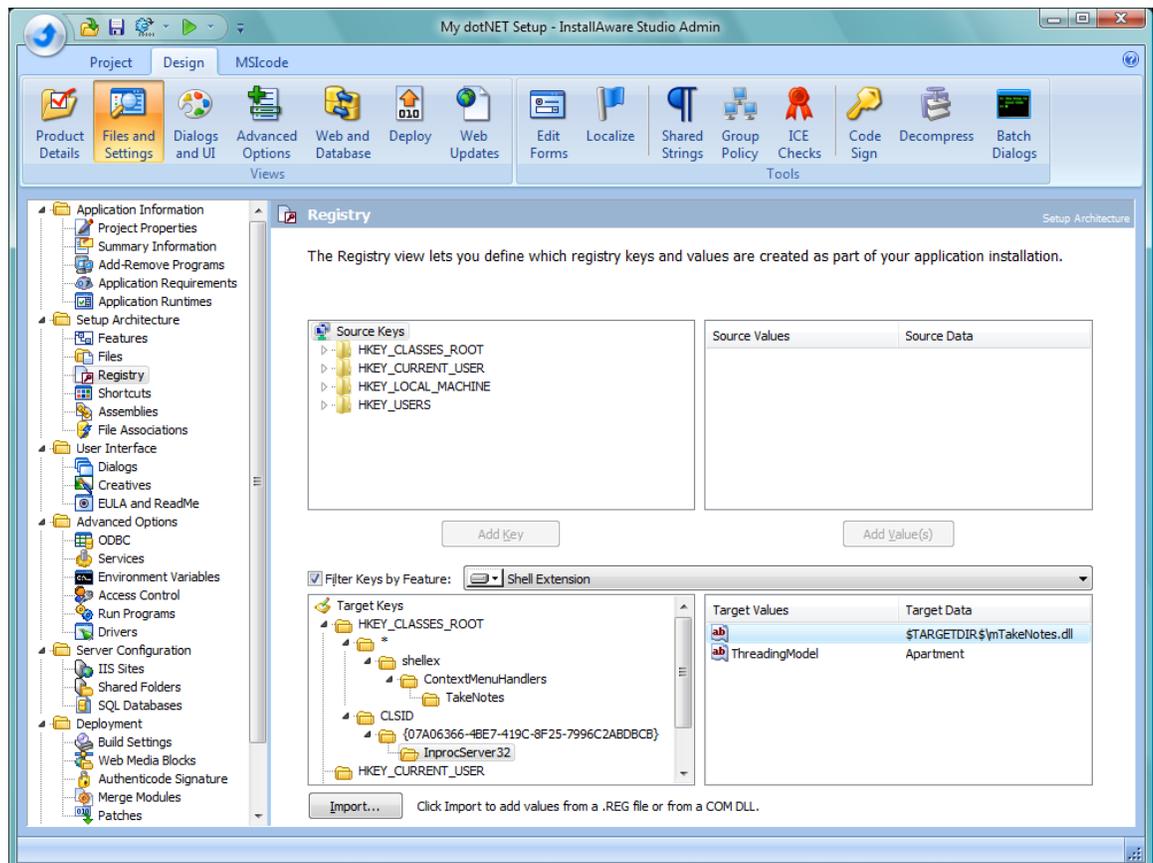
The **Install Files** dialog box appears, allowing you to fine tune the settings of the items being installed.



This dialog box controls many advanced settings for installing files. You may edit the **Source Files** field and specify any wildcard pattern for installing files (for any folders you added above, this field contains the mask `*.*` which means “everything”). You may edit the **Exclude Wildcards** field and specify a mask of files to ignore when building your setups. You may edit the **Target Folder** field and change the destination directory of files being installed. For instance, to copy files to a subfolder of the user chosen directory, you would enter `$TARGETDIR$\subfolder` in this field. You may self-register files at install time by checking the **File is Self Registering DLL** check-box. If the files you are installing are shared by more than one application, be sure to check the **Shared DLL File** field in this dialog box. Review the other available options and check the ones as required by your application.

When you’re done making your changes to your files, click **OK** to continue.

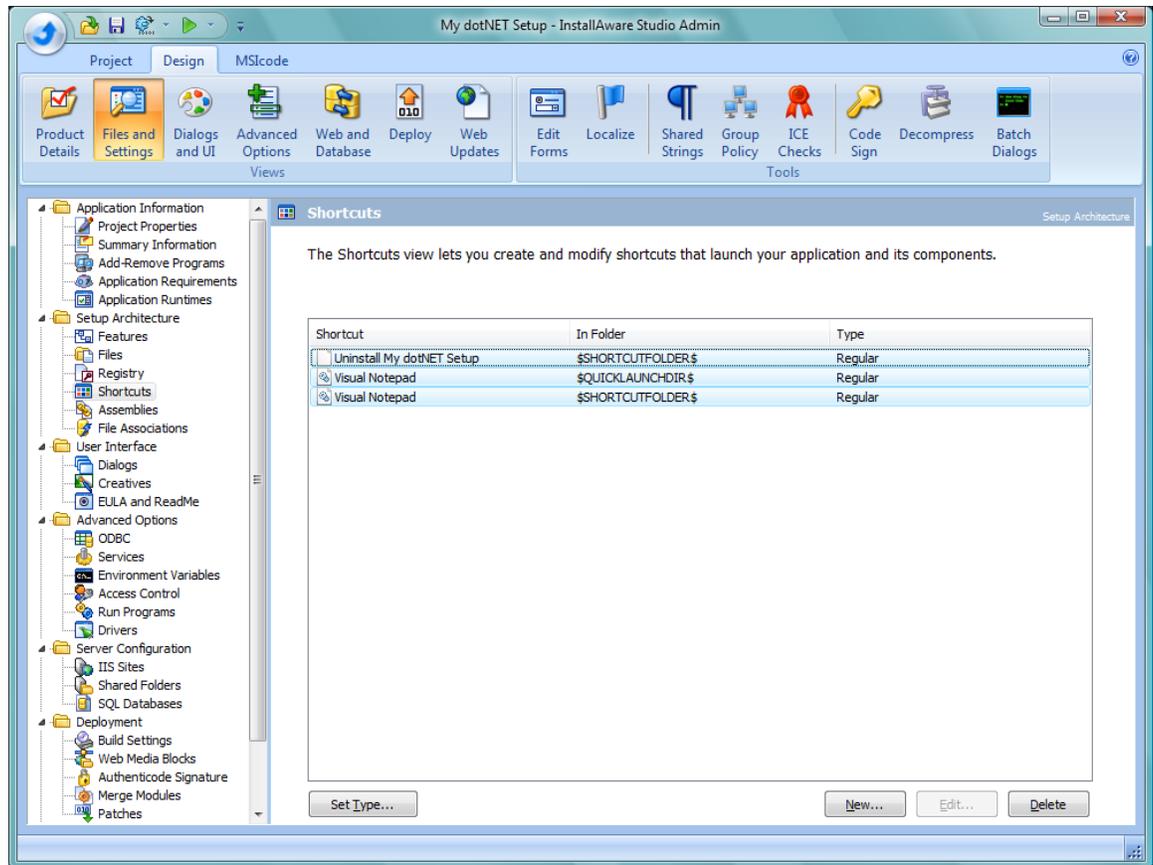
13. Next focus on the **Registry** designer to create the registry entries required by your application. Click **Registry**.



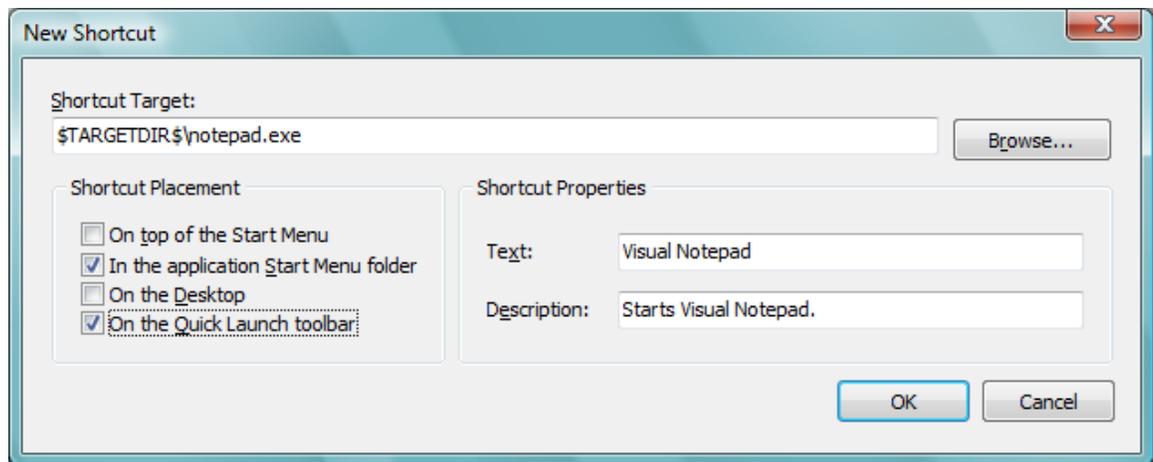
The hypothetical application has some registry entries defined under the Shell Extension feature. These were imported from a self registering COM server using the **Import** button. Use the **Import** button to import registry data from a **.REG** file or a self registering **COM** server (of course, you may also elect to check the **File is Self-Registering DLL** check-box in the **Install Files** dialog to register your servers).

The remainder of the **Registry** designer works much the same way as the **Files** designer. Expand keys under the **Source Keys** heading, click the **Add Key** button to add entire keys together with their subkeys, and click the **Add Values** button to add a selection of values. Customize keys and values by double-clicking the item to update, or highlight multiple items, right-click, and choose **Edit Value**. Adding registry elements to specific features of your application is again possible using the **Filter Keys by Feature** check-box.

14. Click **Shortcuts**.



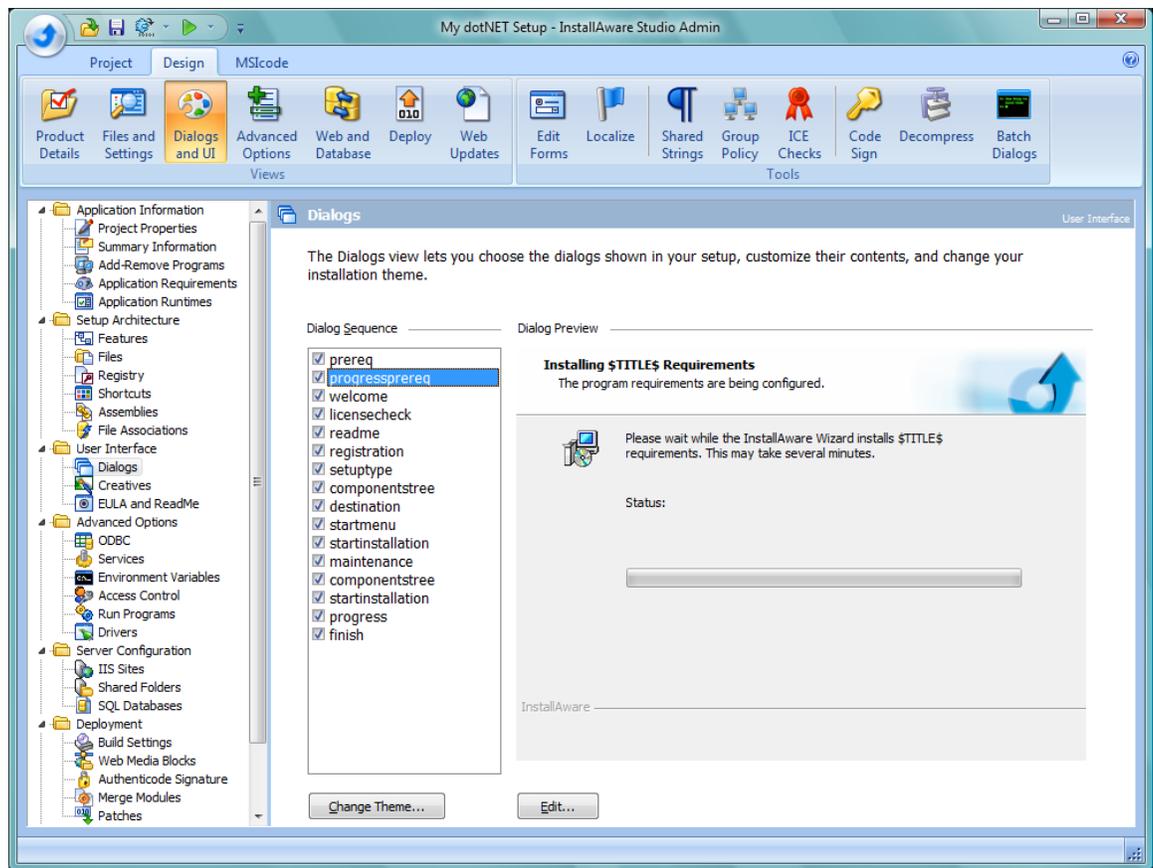
The **Shortcuts** designer displayed here already contains one entry for uninstalling the example application. Existing entries are removed using the **Delete** button, and customized using the **Edit** button. To add a new shortcut, click the **New** button.



The **New Shortcut** dialog box allows to easily define shortcuts, and can place them in frequently used system locations such as the **Quick Launch** toolbar and the **Desktop**, at the click of a single check-box. You don't need to memorize your application paths to populate the **Shortcut Target** field either – just click the **Browse** button to visually locate the file that the shortcut should point to.

You may type any value in the **Shortcut Target** field – you are not limited to the list of files you are installing with your application (as shown by the **Browse** button). You may type a path to a pre-existing file on the target system, or even enter a URL. To create a shortcut that launches a URL in the default system browser, simply type in the full URL path inside the **Shortcut Target** field, instead of a file path. For instance, if your website is <http://www.mysite.com/>, and you wish to create a shortcut to your site, enter <http://www.mysite.com/> in this field. Click **OK** to save your changes.

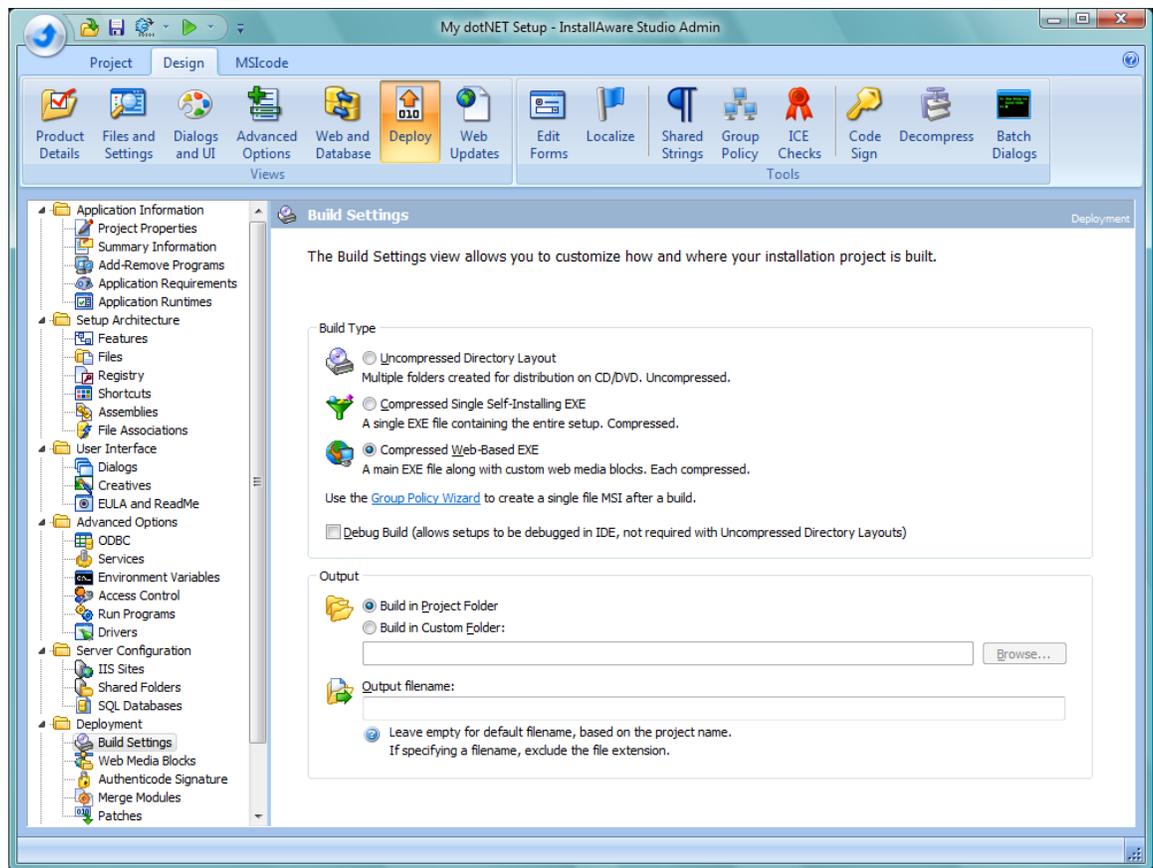
15. We'll skip the remaining two designers under this heading. Choose the **Dialogs** designer, under the **User Interface** heading.



The **Dialogs** designer lists each wizard panel that will be displayed by your installer at run-time. Clicking the **Edit** button customizes a dialog in the dialog editor (every aspect of each dialog is fully customizable). The check box option next to each dialog name determines if the dialog is to be shown during the installation. The **Change Theme** button lets you instantly switch to a different set of dialogs — 15 unique themes are available, and you may build your own.

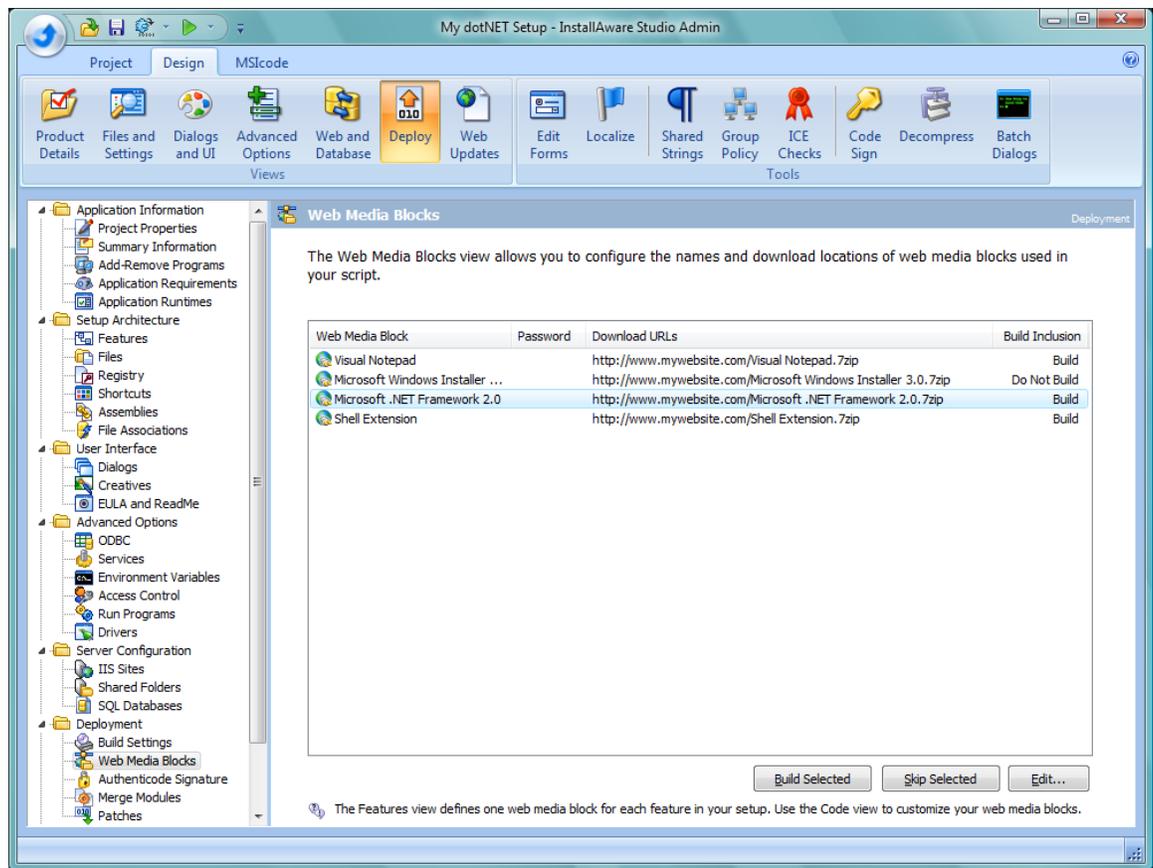
Feel free to explore the remaining designers on your own –our hypothetical application does not need them. We now skip ahead to the **Deployment** section.

16. Choose **Build Settings** under the **Deployment** heading.



InstallAware offers three build options, the last of which is especially useful for deploying .NET applications. The **Uncompressed Directory Layout** option builds your setup files without compressing them, and creates multiple directories in the build folder as part of the process. This option is ideal for CD/DVD distributions. The **Compressed Single Self-Installing EXE** option packages the application files inside a single downloadable EXE file, so you do not have to worry about distributing multiple folders. The **Compressed Web-Based EXE** option enables **Web Media Blocks**, which enable you to optimize your download package by taking out the optional components from the main setup file. Once you've chosen this option, switch over to the **Web Media Blocks** designer to explore this build option in detail.

17. Click **Web Media Blocks**.



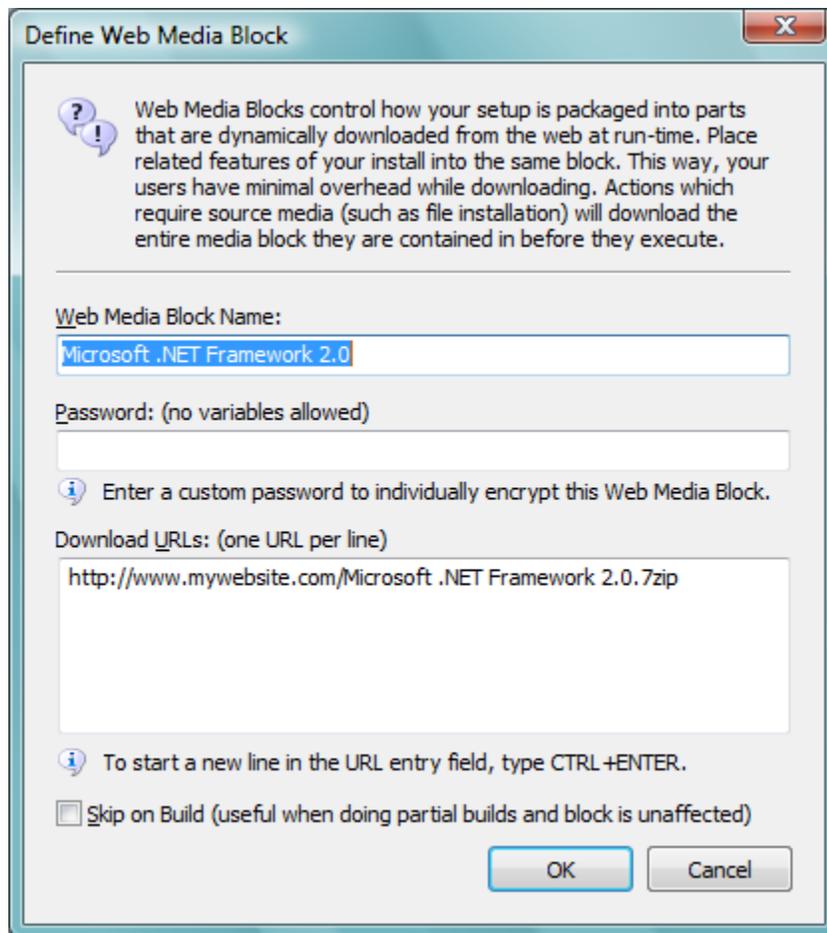
The **Web Media Blocks** designer lets you define the different parts that your setup file can be split into. This is a unique and critical benefit of InstallAware, and makes **Partial Web Deployment** (also called **WebAware Installation**) possible. In the .NET template project, there are two pre-defined Web Media Blocks, in addition to the main setup file. The **Microsoft Windows Installer 3.0** web media block seamlessly downloads and installs the Windows Installer runtime, if it is not found on the target system. Similarly, the **Microsoft .NET Framework 2.0** web media block downloads and installs the .NET Framework. Files added to the setup using the **Feature Independent** are included with the main installer file — allowing the installer to always install those files, even if an Internet connection is unavailable. The remaining Web Media Blocks correspond to the features that you have created for your setup. When setup is running, it will attempt to download Web Media Blocks only if the corresponding runtimes are missing and/or the connected features have been chosen to be installed by the end-user.

18. On the **Project** tab, in the **Build** group, click **Web Deploy** and wait for the build process to complete. Then click **Output Folder Folder** in the same group. You should observe that the following three files have been created as the result of the build:

- [My dotNET Setup.exe](#): This is the main setup executable, which contains the installer and the feature independent files for the setup project.
- [Microsoft .NET Framework 2.0.7zip](#): This is the Web Media Block that contains the .NET Framework 2.0.
- [Microsoft Windows Installer 3.0.7zip](#): This is the Web Media Block that contains Windows Installer 3.0.

Other Web Media Block files may also exist; based on the features you defined for your setup, and whether or not they contained any files.

To tell the installer where to find the Web Media Blocks at runtime, double-click a Web Media Block in the **Web Media Blocks** designer. The **Define Web Media Block** dialog box appears.



Populate the **Download URLs** list to point the installer to one or more actual download locations at run-time. Web Media Block files do not have any special server requirements. Simply make them available for download as ordinary files at the indicated URLs, and the installer will obtain them on-the-fly. Using more than one URL increases scalability and reliability. If your web server does not serve 7zip files due to security restrictions, you may rename the extensions of the created Web Media Block files to an approved extension, and upload them accordingly. Just make sure the **Download URLs** list points to the exact locations of the files on the server, including the file extensions – the initial file name of the Web Media Block as created by the build process is not important. You may also specify FTP sites, network locations, or other file sources (not necessarily Internet locations) among the URLs to be used. Your InstallAware installation will obtain the Web Media Blocks successfully if the target machine can “download” them from one of the specified URLs. Using Internet Explorer (or your favorite browser) on the specified URLs is a great way to test that Web Media Blocks are uploaded correctly and being served.

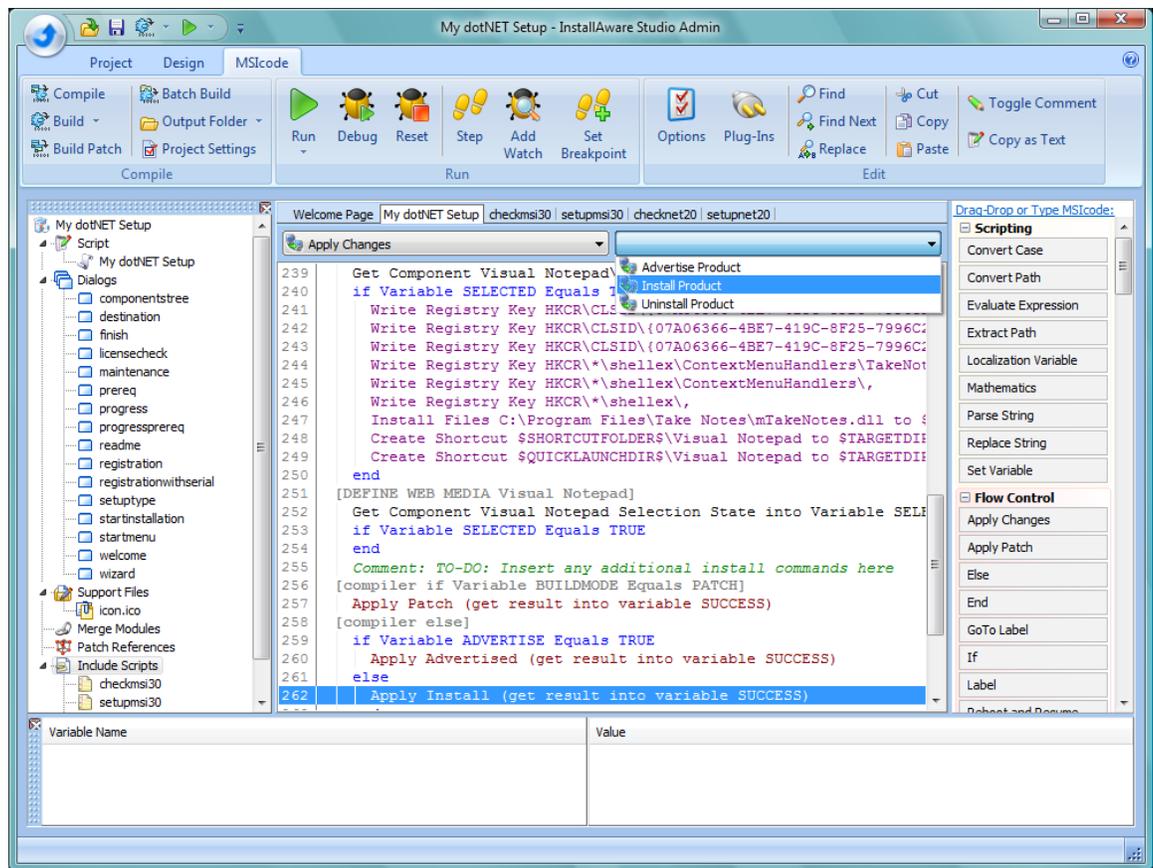
If you would like to save time during builds, you may also check the **Skip on Build** check-box. This is a very convenient option, especially for application runtimes that do not change in between different builds of your own application. When **Skip on Build** is checked, InstallAware will not copy any files belonging to that Web Media Block, nor will it compress any data in that Web Media Block, effectively eliminating that part of the setup from the build process, and therefore improving build speed. As long as you have a pre-built copy of that web media block, you may skip it during builds and save time.

InstallAware 7 also offers the option to individually encrypt each Web Media Block using 256 bit strong AES encryption. Just populate the **Password** field to turn on encryption. Individually encrypting each Web Media Block makes it harder for your setups to be compromised with the loss of a single global password, as each Web Media Block can potentially be using a unique password.

Coupled with the superior compression provided by the InstallAware engine, **Compressed Web-Based EXE** builds provide the industry's leading solution for online software distribution. With the new individual encryption options and URL lists in InstallAware 7, online software distribution becomes even more reliable, scalable, and secure.

You may notice that when you perform a CD or Single File build, Web Media Blocks are not created, even though they have been defined; this makes it easier to build for multiple media types when generating your setups. In fact, you can click **Batch Build** in the **Compile** group on the **MSIcode** tab to build for multiple media types in one pass. Just select each desired build type.

19. Finally take a peek at the setup script, which has been automatically generated and maintained for you so far by the visual designers. Choose the **MSIcode** tab, and then choose the [My dotNET Setup](#) page next to the **Welcome Page**.



On the top left combo box, choose **Apply Changes**. On the top right combo box, choose **Install Product**. This takes you right to the place in your script where most of the install actions take place.

Look around the script. This script is a genuine programming language, called **MSICode**. **MSICode** provides complete branching code execution for Windows Installer, another unique benefit of InstallAware, also called **Genuine Scripting for Windows Installer**. Not only does this feature make developing setups easier — it also eliminates the design overhead by providing a simpler development model than is ordinarily possible with Windows Installer. You simply author your human readable **MSICode** script, using either the **Design Views** or the **MSICode** view, and InstallAware produces a logo compliant MSI for you automatically at build-time. **MSICode** is the easiest way to describe the content and logic of an MSI database, significantly simplifying the setup development process, and reducing development, maintenance, and support costs for your setups.

Also notice the **Install Files**, **Create Shortcut**, and **Write Registry** commands in the script. You will recall them from the edits we made in the **Files**, **Registry** and **Shortcuts** designers. The InstallAware IDE features two-way integrated code editing, so that when you make changes in any of the **Design Views**, the changes are always reflected in the **MSIcode** view, and vice versa. This allows you to rapidly prototype your setup in the **Design Views**, and then switch to the **MSIcode** view to fine-tune your installation. When you build your setup, the resulting installer will always be a logo certifiable MSI installer database — InstallAware takes care of the necessary MSI table insertions and MSI sequences for you. Absolutely no custom action authoring is required with InstallAware!

A screenshot of a code editor window. The text 'Run Program' is visible, with 'Program' highlighted in blue. To the right of the text is a small blue square containing a white downward-pointing arrow, indicating a dropdown menu.

There are many more **MSIcode** commands to try – just scroll the list on the right of the script editor. Each command executes a specific installation task, saving you countless hours of custom development effort. You may drag-and-drop commands from there into the setup script, or select a line in the script editor and simply type the first few letters of a command – this activates smart code completion. The script editor features all modern conveniences such as copy/paste, find/replace, and more. You may even use code folding and define your custom code folding regions to organize your setup script.

20. This concludes the creation of your .NET application installation. Make ample use of the available **Design Views** and **MSIcode** to refine your setup, and ship when ready!

Deployment Scenarios

Independent Software Vendors (ISVs)

Isolate Application Runtimes

Most ISVs distribute their setups on the Internet. This means downloads have to be as small as possible — smaller files mean more downloads and less bandwidth consumption. Especially for .NET applications, the standard 23 megabyte size of the .NET Framework can be a huge problem. InstallAware works around this problem by both stripping runtimes from the main product download, and also by using superior compression to reduce the .NET Framework size to only 12 megabytes.

Secure Installers with Authenticode Code Signing

Especially with the release of Windows XP Service Pack 2, and the upcoming Windows Vista, users are cautioned strongly against installing packages that aren't digitally signed. InstallAware supports Authenticode code signing technologies to verify the integrity of downloaded content.

Protect Application Features

ISVs can use Web Media Blocks to strip crucial application components from their public downloads. Using scripting, the installer can query user and system information to verify that the end-user is eligible to install protected features. Only if the user is authenticated (by, for example, querying a web page with user credentials), will the installer proceed to download the protected Web Media Blocks.

Deploy Unlimited Web Updates

In InstallAware 7 is the ability to deliver unlimited updates to your users over the web. The update experience closely resembles that of Windows Update – displaying standard message balloons in the system tray (even on old Windows versions that do not support tray balloons), and customizable dialogs that configure the update schedule, list available updates, and install them.

Improve the End-User Experience

InstallAware present a single, unified user interface during the entire setup process. Separate install wizards or progress windows are never shown when launching custom third party installers and/or configuring application runtimes. InstallAware even natively captures and displays the progress of lengthy runtime installs like .NET Framework 2.0 in your InstallAware setup wizard, improving the user experience.

Leverage One-Click Incremental Patching

InstallAware provides a one-click patching mechanism that works by building a patch against older versions of the installer. This is done by simply showing the setup project where the older installer files of the product are found. When a patch is built, it contains only the minimum incremental data required to bring a product up-to-date with the latest installation files. Incremental patching saves developers and end-users time and effort in updating their current installers.

Build Logo Certifiable Installers

Microsoft's Designed for Windows logo program provides ISVs with a way to certify the stability, reliability, and quality of their solutions. Most enterprises today will not even consider software titles that do not have a logo certified MSI installer, for reasons of automating and securing their corporate deployment. For any product to pass testing for the Designed for Windows logo programs, an MSI installer for that product, with the following features, is required:

- No ICE (MSI database internal consistency evaluator) errors must be found
- MSI databases should support advertising

InstallAware setups are automatically logo compliant. No matter how complex or simple the setup script, the output MSI files always pass ICE tests, and support advertised installations.

Rapidly Customize Installers for Product Editions

ISVs typically produce several editions of their products, tailored for different market and user segments. Because these products rely on the same core, their installers are often very similar. Typically the highest edition features the most complex install, with files and logic being stripped away progressively as the product editions are traversed. Instead of building and maintaining multiple installs for each edition, InstallAware lets developers define compiler variables, which work much like `IFDEF` statements found in programming languages. Based on the values of these compiler variables, parts of the setup script can be excluded from the build process, thereby emitting customized installers with absolutely no manual effort.

Communicate with End-Users

InstallAware makes it very easy to build custom dialogs that capture any kind of information from the end-user at install time. Simply use the dialog editor and customize an existing dialog, or build your own from scratch. Use edit boxes, drop-down combo boxes, radio buttons and check-boxes to interact with end-users and survey them as an integral part of your installation – not on a web form where they can walk away from your download. Seamlessly submit captured information to a web server. Ask end-users where they found about your product, why they are uninstalling, and more!

Enterprise Information Technology Scenarios

Allow Partners to Customize Their Own Installers

InstallAware features a COM automation library, along with an ASP scripting object that enables access to that library from ASP web pages. Using COM automation, enterprises could programmatically emit complete setup projects and scripts, and then build them on-the-fly in response to partner requests. For instance, consider an enterprise product that is made available to a large number of resellers, each with its own language and branding requirements. A web site could be built where resellers could upload their custom text and branding images. Then, when the Build Installer button is clicked, InstallAware automation could be invoked to dynamically build a customized installer, which would make the custom setup available for download as soon as the build process finished. This would reduce the burden on an enterprise when supporting its resellers.

Lock Down Installers to Only Run Inside a Network

InstallAware's Web Media Blocks are able to download files not only from the Internet, but also UNC paths. Where sensitive application and data are involved, an installer could be built that placed all its contents inside Web Media Blocks, and those Web Media Blocks placed on locations accessible only inside the corporate network. Even if these installers were to be "stolen", they would not actually work, since they would attempt to download Web Media Blocks from a location that would not exist on an outside site.

Use Silent Installs for Automated Software Distribution

Installing and maintaining applications deployed onto thousands of computers on a corporate network can rapidly escalate into a maintenance nightmare. Because InstallAware installations are Windows Installer based, they conform and interoperate well with large scale corporate deployment scenarios. Applications can be silently installed, removed, and monitored — all without user intervention or knowledge. Log files can be created from installations to provide additional debugging data in case deployment problems arise.

Group Policy Deployment

InstallAware setups are fully compatible with Active Directory deployment using Group Policy, as well as other automated deployment technologies. Push installers onto thousands of desktops with a single click.

Shell to and Repackage Installs

Typically, when deploying older software, corporate administrators are required to convert old and non-standard package formats into the MSI package format. Even when working with MSI based installers, packages may need to be installed with a precisely defined set of configuration settings, to ensure proper operation. InstallAware addresses both these needs. InstallAware's **Shell to MSI** technology enables the execution of any MSI package, with a specified set of configuration settings, as an integral part of a larger software installation process. In fact the **Shell to MSI** process is so efficient that it even captures the progress of the spawned installation. With non-MSI packages, the PackageAware tool that ships with InstallAware provides a migration path: PackageAware takes before and after snapshots of the state of the system as a legacy package is installed. It then encapsulates the delta of the system snapshots in a ready-to-build, customizable InstallAware setup project. Both these technologies enable seamless and painless installations of large numbers of interdependent packages across the entire corporate network.

Allow Non-Administrative Installs

Since InstallAware setups are Windows Installer based, they automatically leverage the ability of Windows Installer to perform low level changes to the system even when administrator rights are not present. As long as the **Always Install Elevated** group policy has been defined by the corporate administrator for user accounts, all InstallAware installations will successfully install inside locked-down user environments, making it unnecessary for a system administrator to manually log on and install the desired product on the system.

Use Advertising for Install-On-Demand and Self-Healing

Advertising is a Windows Installer term for two similar concepts:

1. An installation that runs in advertised mode creates shortcuts, registers file types, and creates registry entries required by the product — overall creating the impression that the application has been installed. However, it does not actually copy any files, until the application has been invoked for the first time, from either the Start Menu, or by double-clicking its associated file in Explorer.
2. Advertised shortcuts and file types both enable the above behavior, and also repair a damaged application by copying missing files whenever they are invoked. They do this by checking that all required files are present when they are being invoked.

InstallAware setups fully support both of these advertising technologies. Plus, InstallAware adds the capability of storing each feature of an advertised installation as a Web Media Block to seamlessly download it on-demand from any desired URL.

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.

Copyright© 1996-2007 InstallAware Software Corporation. All rights reserved. All InstallAware brand and product names are trademarks or registered trademarks of InstallAware Software Corporation in the United States and other countries. Microsoft, Windows, and other Microsoft product names are trademarks or registered trademarks of Microsoft Corporation in the U.S. and other countries. All other marks are the property of their respective owners. Corporate Headquarters: 336 Guerrero Street, San Francisco CA 94103-3332 · 415-358-4094 · www.installaware.com · sales@installaware.com