Extreme Installers: Slash The .NET Runtime Size in Half, Register Users Online

[reprint of article that appeared on the October 2005 issue of MSDN Magazine Europe]

European Content Inside!
The Microsoft Journal for Developers
msdn magazine europe
Extreme Installers
Slash The .Net Runtime Size In Half, Register Users Online
High Availability
Keep Your Code Running With The Reliability Features Of The .NET Framework
ASP.NET 2.0
Personalize Your Portal With User Controls And Custom Web Parts
Custom Cultures Extend Your Code's Global Reach With New Features In The .NET Framework 2.0
Stay Alert Use Managed Code To Generate A Secure Audit Trail
Microsoft
msdn

This article uses the following technologies:
Microsoft Windows Installer, Microsoft .NET, InstallAware Script

This article discusses:

- Reducing Size of the .NET Runtime
- Partial Web Deployment of Application Runtimes
- Using InstallAware Script and Dialog Designer for Creating User Feedback Forms During the (Un)Installation

Sinan Karaca was raised in diverse parts of the world. He went to primary school in Kabul, Afghanistan, and received university education in Austin, Texas, where he majored in Computer Sciences and Philosophy. Sinan currently lives in Turkey, where he directs MimarSinan International, a software company focusing on file and installation technologies for the Microsoft Windows and recently .NET platforms. Sinan can be reached at **mimarsinan@gmail.com**.

Extreme Installers

Installers are the first impression that end-users have with your software, and as such are a very critical part of your application; however they are generally forgotten until the very last minute when managers and developers alike are struck in panic, not sure how to go about getting their installers done in time. Fortunately, a wide variety of third party tools are available to help with the task, and especially with Microsoft's introduction of the Windows Installer standard in 1999, building installers has become a standardized process.

In this article we will explore InstallAware™, a fairly new installation development environment targeting Microsoft Windows Installer. InstallAware has a couple of unique extras that help especially with deploying .NET Framework applications. InstallAware also makes it very easy to customize every aspect of the end-user experience in the wizard dialogs, which we will be leveraging to create a custom dialog that captures user registration information, and posts it to a web server, all as an integral part of the main installation.

.NET Framework Deployment

While deploying .NET applications is much easier compared to traditional applications (no more DLL hell to suffer, thanks to assemblies), there are still some issues left over that merit using an installer, as much as we would like to do plain XCOPY deployment. The biggest of these is deployment of the .NET Framework itself. Version 1.1 of the framework is a 23 MB download; Version 2.0 is even larger at 26 MB. Especially considering that our applications may be a lot smaller than that, typically coming in at less than 1 MB for a rich client application, the burden of the runtime becomes especially apparent.

InstallAware helps us with that, in a couple of unique ways. The first is to reduce the size of the runtime itself. InstallAware utilizes a form of lossless compression that exceeds the compression used in the standard .NET Framework installer. This compression algorithm is so efficient that it reduces the .NET Framework installer down to 12 MB, and compared to the already compressed size of 23 MB, that is almost 50% savings!

If you have not done so already, visit www.installaware.com and download InstallAware Studio first. Then, open the InstallAware IDE, and choose **File->New->.NET Script** to start a new .NET Framework installer project. InstallAware features visual editors that you can use to rapidly specify the files, registry entries, shortcuts, assemblies, and any other items that comprise your application. These views work similarly in most installer environments, so we won't cover them here; but feel free to make use of them to flesh out your product installation first. Then, click the **Build Settings** button under the **Deployment** heading, and notice InstallAware offers three build types:

- **Uncompressed Directory Layout**: Best for installers distributed on CD/DVD media, where space is not usually a limiting factor. InstallAware builds your setup using multiple folder and file structures.
- **Compressed Single Self Installing EXE**: InstallAware builds a single self-installing executable file. This is similar to the monolithic installers that contain all application files and runtimes in a single package that works without requiring an Internet connection.
- **Compressed Web Based EXE**: InstallAware builds a self-installing EXE, and strips away runtimes (along with other optional application features) into other files downloadable from the Internet on-the-fly during an installation.

Choose the **Compressed Single Self Installing EXE** option, and build your setup by choosing **Project->Build**. If you build an empty project this way, the resulting installer file will be approximately 15 MB in size. Notice that this file already includes the .NET Framework, and the Windows Installer runtime – along with our setup MSI files! We can already see InstallAware's great compression at work here – typically the size of both runtimes weighs in quite heavily at 26 MB.

But we can do better than that – by isolating both these runtimes from our main installer, and putting them on the web for download on demand, using web deployment. Web deployment does have the downside that everything goes on the web – even our essential application files, thus mandating an Internet connection every time the install is run. Fortunately InstallAware offers a unique partial web deployment option that removes this limitation on web deployed installers.

Partial Web Deployment

Now build the same setup using the **Compressed Web-Based EXE** option, which will build our setup for partial web deployment on the web. After the build is complete, choose **File->Open Build Folder** and explore the output:

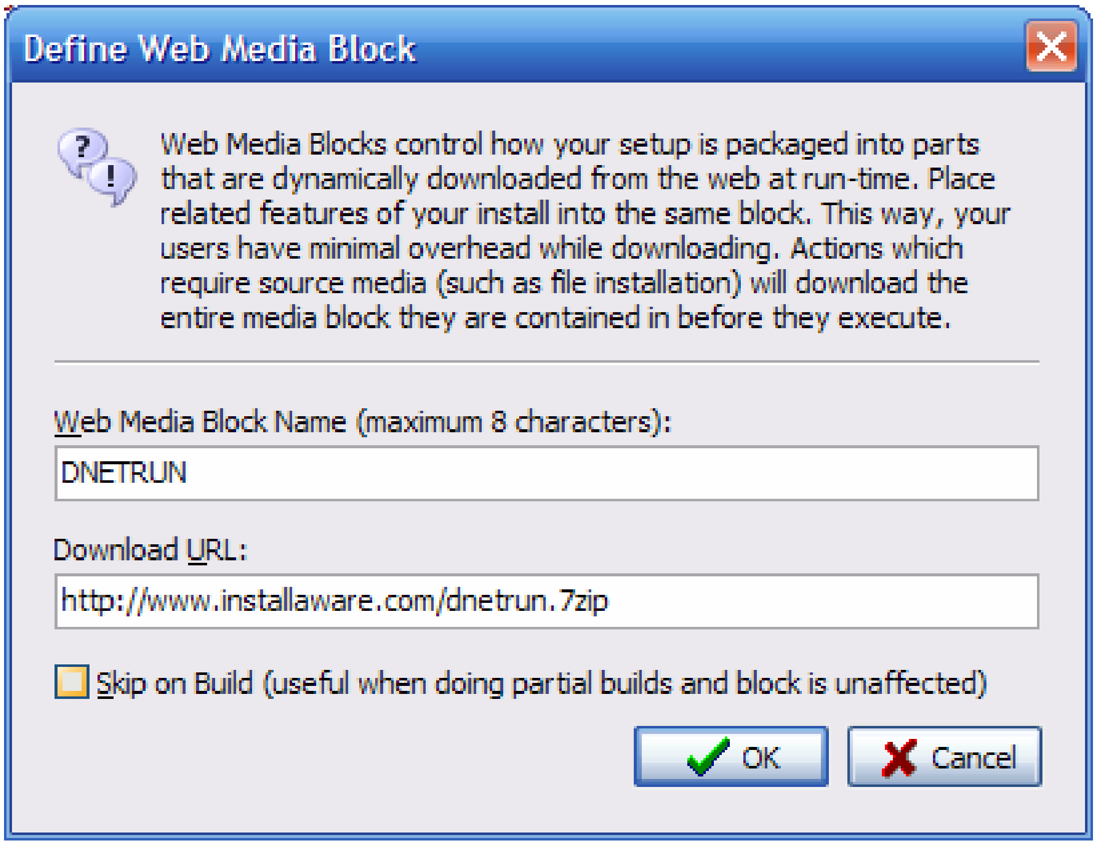
- **setup.exe**: This is the main installation file, and contains your essential application files, along with the setup MSI files. Note that this file may be named differently based on the name you chose for your setup project.
- **DNETRUN.7zip**: This is the .NET Framework partial web deployment file, called a `web media block` in InstallAware. Notice the size – KB – almost half the regular size of the runtime, at only 12.1 MB.
- **MSIRUN.7zip**: This is the Windows Installer runtime, and again, weighs in at half the regular size, at 1.83 MB.

While setup.exe is running, if all the runtimes are already installed on the target system, it will not connect to the Internet to download anything, as essential application files are already inside the setup.exe. This is what partial web deployment offers over and above traditional web deployment schemes, which force you to put even your vital application files on the Internet, needlessly requiring an Internet connection after you have already downloaded your setup once already.

If some of the runtimes are required, setup will first try to find the web media block files in the same folder as itself. If they are found there, it will use them from that location to prevent connecting to the Internet. If they are not found, it will connect to the Internet, and download them automatically, as a seamless part of the main installation. If the download is interrupted (or corrupted) for any reason, all we have to do is try again – setup.exe will resume from where it left off, and not re-download the entire file. The process is also proxy friendly – it will automatically detect proxy server configuration on the target system.

The setup engine that InstallAware uses is Internet Explorer's native download engine. InstallAware supports installations on every version of Windows – from the Windows 95 Gold release to the latest Windows Server 2003 system, so Internet Explorer may not be pre-installed on earlier versions of Windows (Windows 95 and Windows NT 4.0 in particular), however in that case InstallAware will fall back to its secondary, native download engine to make sure your installs succeed without a hiccup.

But how does our setup know where it is to obtain the web media block files from? Click the **Web Media Blocks** button under the **Deployment** heading. You will see a list of web media block files, including `DNETRUN` and `MSIRUN`. Double-click one of these, and the **Define Web Media Block** dialog opens.



Web Media Blocks control how your setup is packaged into parts that are dynamically downloaded from the web at run-time. Place related features of your install into the same block. This way, your users have minimal overhead while downloading. Actions which require source media (such as file installation) will download the entire media block they are contained in before they execute.

Web Media Block Name (maximum 8 characters): DNETRUN

Download URL: http://www.installaware.com/dnetrun.7zip

☐ Skip on Build (useful when doing partial builds and block is unaffected)

Figure 1 Defining a Web Media Block

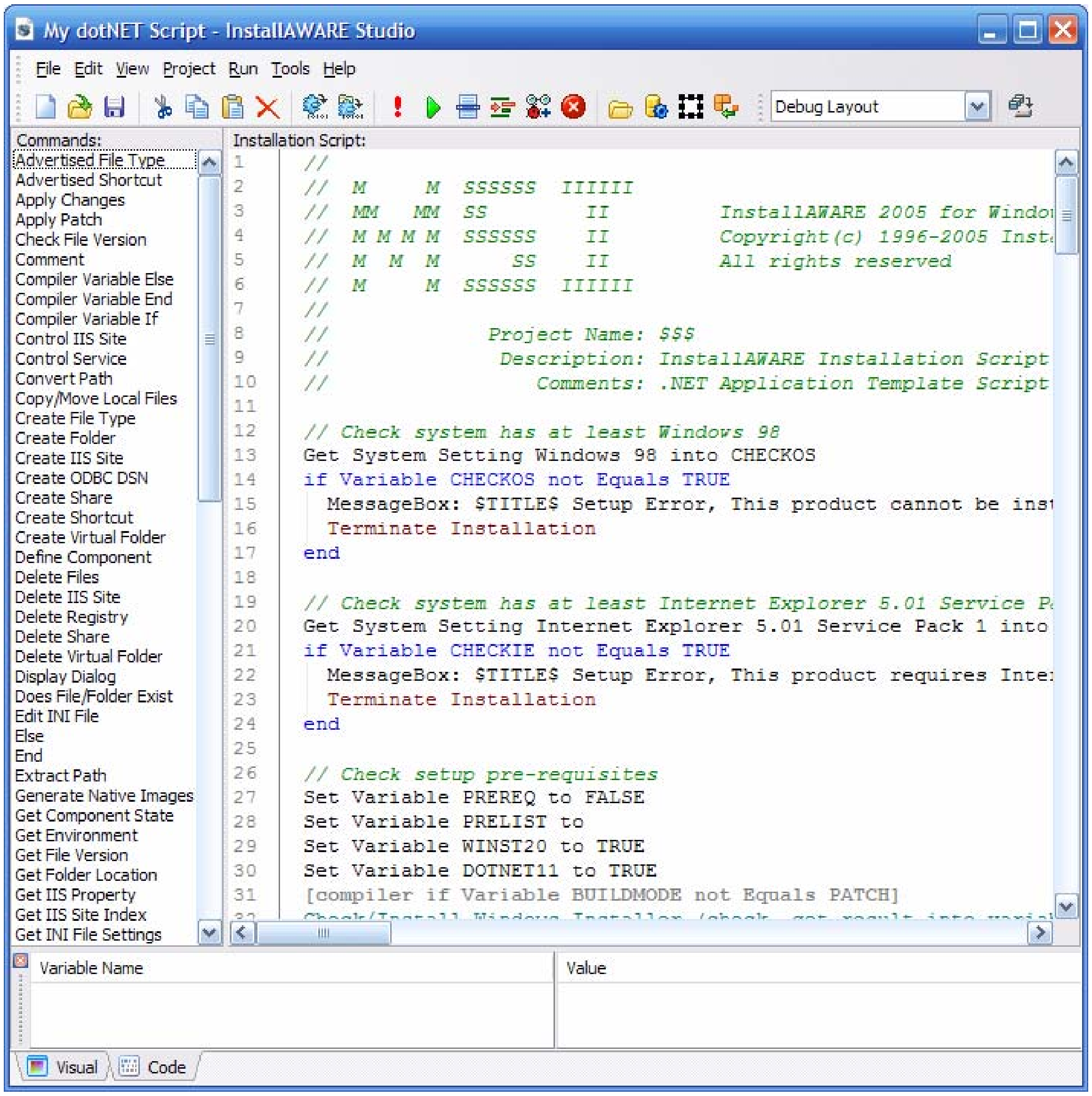
The **Web Media Block Name** field determines the name of the web media block file that is created during the build process. The **Download URL** field tells the setup program where it can download the web media block file from at runtime. All you need to do is upload your web media block file to a web server, and type in the download URL for that location in here. The rest is all automatic, and there are no specific server requirements for web deployment. Just publish the web media block file as an ordinary file to the location described in the **Download URL** field.

You can also enter non-Internet locations in the **Download URL** field, such as a corporate network share of the form \\server\share\path\weblock.7zip. This has the additional side effect of protecting your installers from running at off-site locations, since the installer will never find a file that resides inside your corporate network while you are not physically connected to the network.

InstallAware also lets you use the web media block mechanism to precisely split your setup into any number of online and offline parts. For instance, you could put rarely used or bulky application parts, such as clip-art collections for word processors, inside their own web media blocks, and separate them from the main setup. We can precisely control what parts of our application go where – and to do that we will switch to the code view.

The Code View

Choose **View->Code** to switch to the InstallAware Script view. What we see is a conditionally flowing, traditional setup script; with commands like `Install Files`, `Create Shortcut`, and the like. Feel free to take a quick look at script (or print it for a hardcopy reference) to get a sense for how the setup logic flows. Of course, you can change everything – the script we're looking at is just the template script that the IDE created for us when we started our new project.



```
//
//  M     M  SSSSSS  IIIIII
//  MM   MM  SS        II          InstallAWARE 2005 for Windo
//  M M M M  SSSSSS    II          Copyright(c) 1996-2005 Inst
//  M  M  M      SS    II          All rights reserved
//  M     M  SSSSSS  IIIIII
//
//              Project Name: $$$
//               Description: InstallAWARE Installation Script
//                  Comments: .NET Application Template Script

// Check system has at least Windows 98
Get System Setting Windows 98 into CHECKOS
if Variable CHECKOS not Equals TRUE
  MessageBox: $TITLE$ Setup Error, This product cannot be ins
  Terminate Installation
end

// Check system has at least Internet Explorer 5.01 Service P
Get System Setting Internet Explorer 5.01 Service Pack 1 into
if Variable CHECKIE not Equals TRUE
  MessageBox: $TITLE$ Setup Error, This product requires Inte
  Terminate Installation
end

// Check setup pre-requisites
Set Variable PREREQ to FALSE
Set Variable PRELIST to
Set Variable WINST20 to TRUE
Set Variable DOTNET11 to TRUE
[compiler if Variable BUILDMODE not Equals PATCH]
```

Figure 2 The InstallAware Script

Whenever you make changes in the visual view, as we had been doing so far, the IDE updates the underlying script for us automatically. Likewise, after making changes in the code view, if we switch back to the visual view, the visual view updates itself accordingly, making RAD setup development possible. The code view is the heart of the installation and contains all the commands that drive our setup, and of course, at build time, the InstallAware Script code is compiled into a logo compliant MSI database, capable of running directly without requiring a third party scripting engine on the target system.

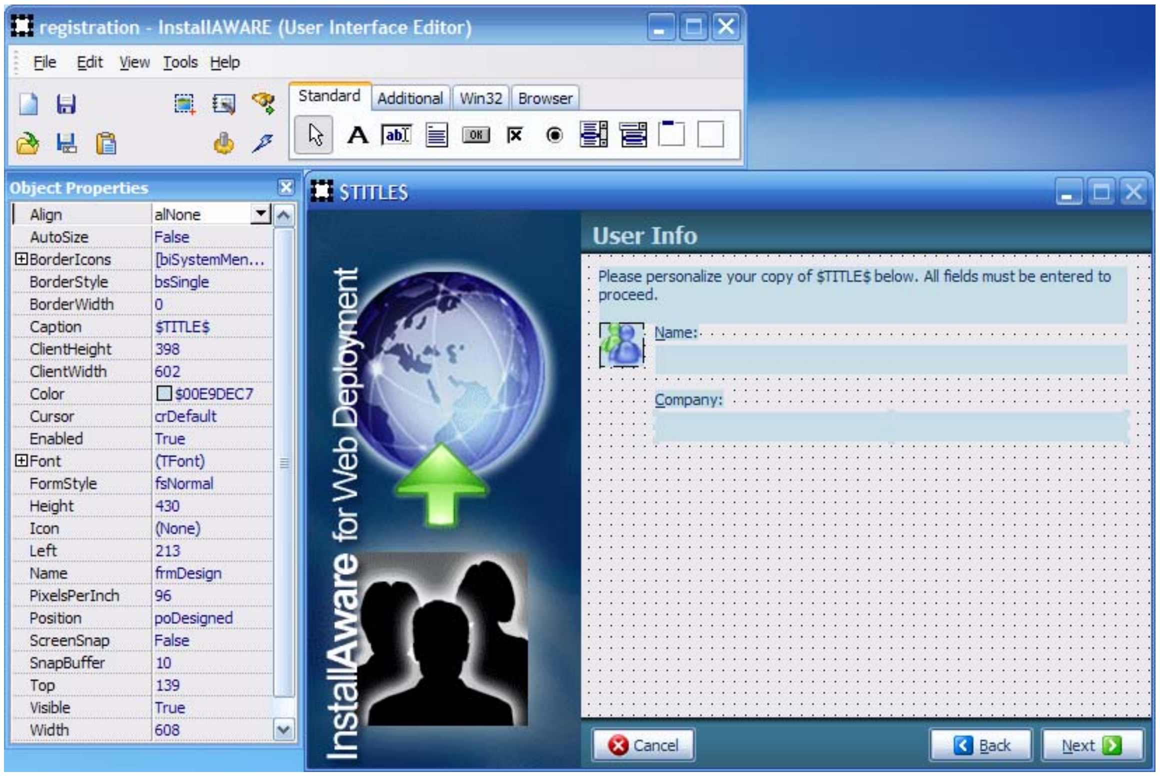
Type `CTRL+F` and search for `Define Web Media`. The editor highlights the web media blocks that have been defined in the script, including `DNETRUN` and `MSIRUN`. The script can contain any number of `Web Media Block` statements. To define a new web media block, simply double-click the `Web Media Block` command from the list of commands listed on the left of your setup script. The **Define Web Media Block** dialog we saw earlier in the visual view opens. Every subsequent installation command (such as `Install Files`) that follows the web media block statement will put its contents inside the named web media block, until you define another web media block in your script.

If you leave the web block name empty in the **Define Web Media Block** dialog, and click **OK** to return to the script, InstallAware inserts the command `OFFLINE CONTENT` into your script. If you search for `OFFLINE CONTENT`, you will also find the initial one that was already inserted for us when we first started the project. Files that follow the offline content directive are placed directly inside the setup.exe file, so if you have parts of your application that are always to be installed, placing them after an offline content statement ensures that they will always be available with the main setup.exe file.

Using the `Web Media Block` commands, you can precisely craft your setups into any number of offline and online parts. You can even share web media blocks across different setups, as long as they contain shared files. The script editor gives us full freedom in deciding what goes where.

Posting User Information

Let's now explore the InstallAware Dialog Editor, and capture user information which we will post during the install to a web server. First switch back to the visual view by choosing **View->Visual**, and then click the **Dialogs** button under the **User Interface** heading. From the **Dialog Sequence** listing double-click the **registration** dialog which we will be customizing to capture an email address, in addition to the name and company of the user.


registration - InstallAWARE (User Interface Editor)
File Edit View Tools Help
Standard Additional Win32 Browser
Object Properties
$TITLE$
InstallAware for Web Deployment
User Info
Name:
Company:
Cancel
Back
Next

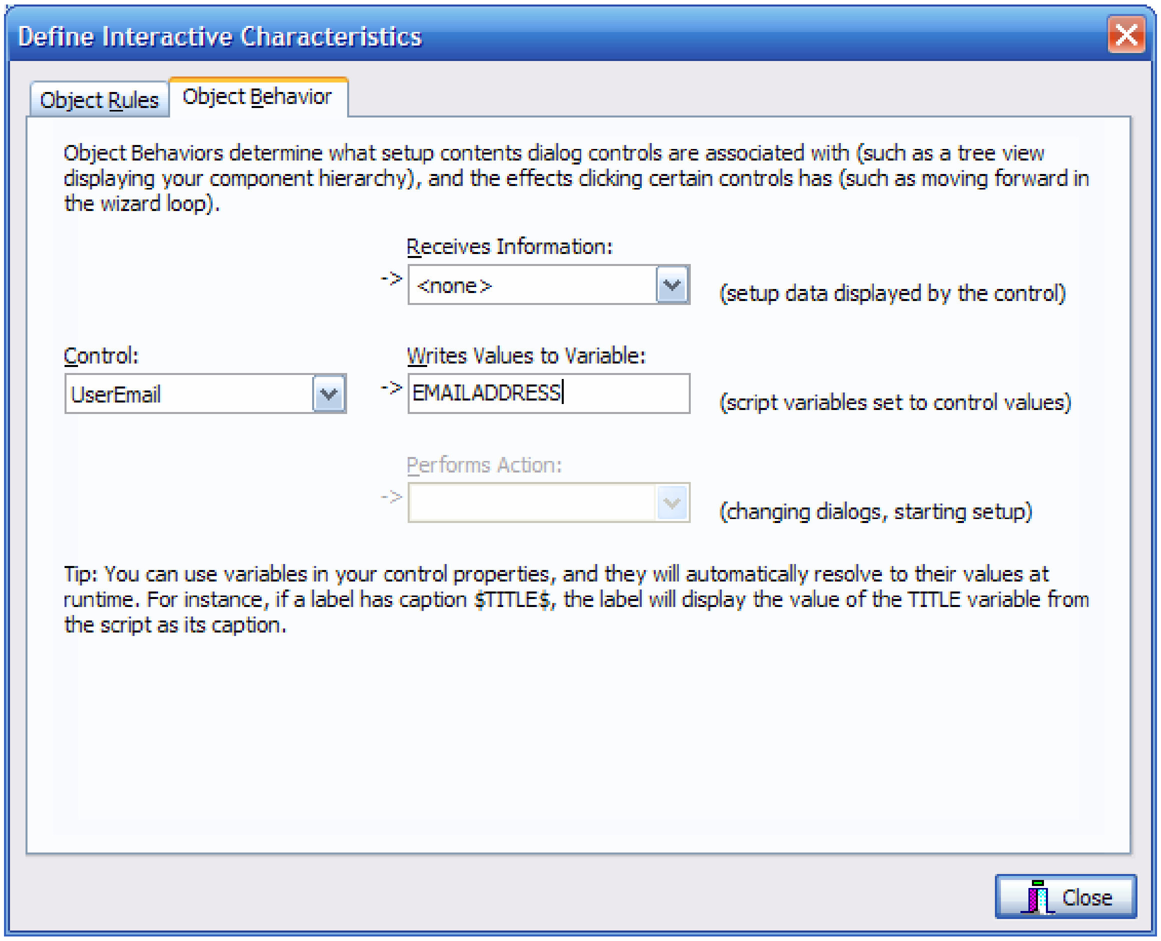
Align	alNone
AutoSize	False
⊞BorderIcons	[biSystemMen...
BorderStyle	bsSingle
BorderWidth	0
Caption	$TITLE$
ClientHeight	398
ClientWidth	602
Color	☐ $00E9DEC7
Cursor	crDefault
Enabled	True
⊞Font	(TFont)
FormStyle	fsNormal
Height	430
Icon	(None)
Left	213
Name	frmDesign
PixelsPerInch	96
Position	poDesigned
ScreenSnap	False
SnapBuffer	10
Top	139
Visible	True
Width	608

Please personalize your copy of $TITLE$ below. All fields must be entered to proceed.

Figure 3 The Dialog Editor

The dialog editor opens with the default registration dialog. We have an **object properties** window that we can use to customize our dialog properties, along with a rich **component palette**, including **HTML** and **Flash** containers that can be used for interactive billboards during an installation. For now, place a text label and an edit control on the form. Edit the `Caption` property of the **text label** to read `&Email Address`, and edit the `Text` property of the **edit control** to leave it blank. Set the `Name` property of the **edit control** to `UserEmail` and remember to set the `FocusControl` property of the **text label** to `UserEmail`, so users can use keyboard shortcuts if necessary. We also want to make registration optional, so users are submitting information only if they want to. Add a **check-box** to the form, set its `Name` property to `RegisterMe`, `Caption` to `&Register on the Internet`, and `Checked` to `TRUE`.

Now its time to link these controls with our setup script. Double-click the **edit control** to show the **Define Interactive Characteristics** dialog, and choose the **Object Behavior** tab.


Define Interactive Characteristics

Object Behaviors determine what setup contents dialog controls are associated with (such as a tree view displaying your component hierarchy), and the effects clicking certain controls has (such as moving forward in the wizard loop).

Receives Information: -> <none> (setup data displayed by the control)

Control: UserEmail

Writes Values to Variable: -> EMAILADDRESS (script variables set to control values)

Performs Action: -> (changing dialogs, starting setup)

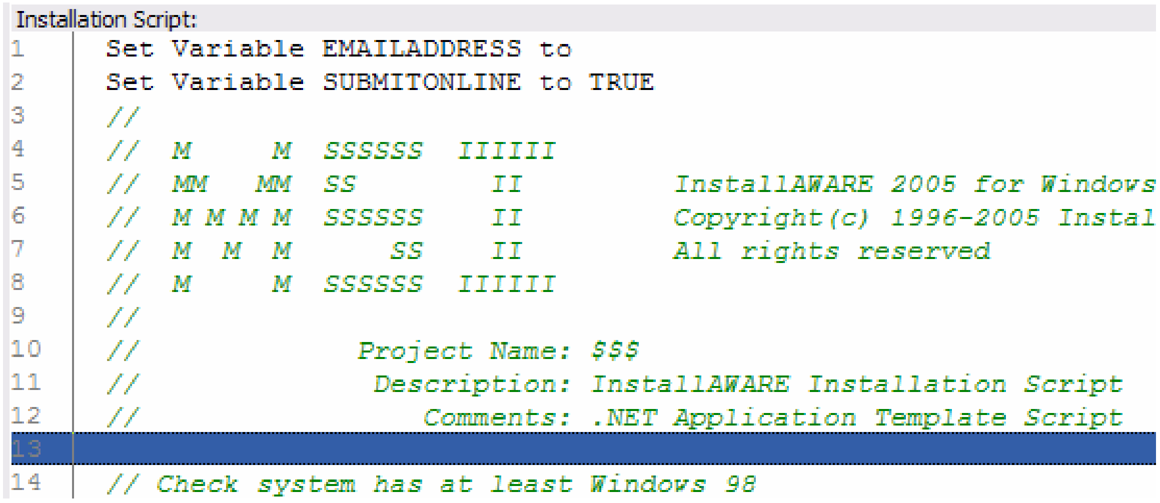
Tip: You can use variables in your control properties, and they will automatically resolve to their values at runtime. For instance, if a label has caption $TITLE$, the label will display the value of the TITLE variable from the script as its caption.

Figure 4 Assigning Script Variables to Dialog Controls

Enter EMAILADDRESS in the **Write Values to Variable** field. This way, we will be able to access the text entered by the user into our edit control from our InstallAware Script by referencing the EMAILADDRESS script variable. Then click the **Control** drop-down box and choose RegisterMe. Tell this control to write to the variable SUBMITONLINE.

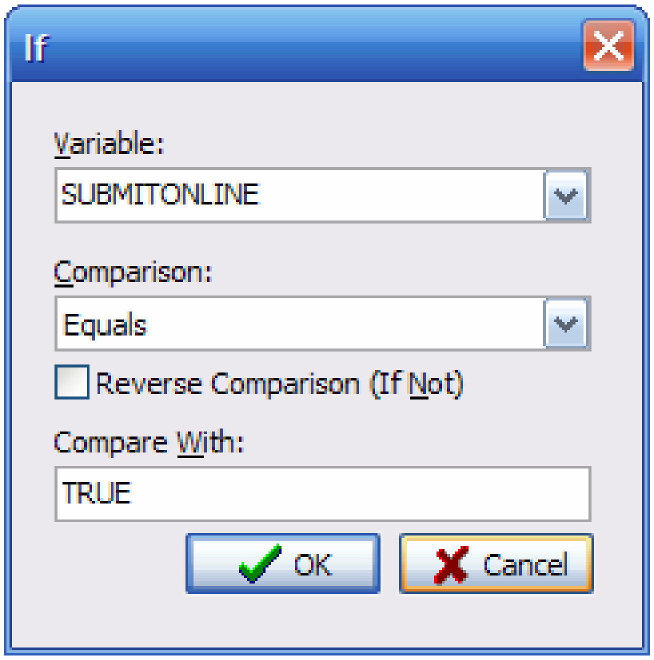
You may also choose the Name and Company controls from the **Control** drop-down to see that the dialog is already configured to write the user name and user company information into the USERNAME and USERCOMPANY variables respectively. Click **Close**, the choose **File->Save** to save the changes we made to our dialog. Remember that you can always customize the appearance of the dialog later by playing with the **Object Properties** window.

Now switch back to the Code view in the main InstallAware IDE. Scroll to the top of the script, and insert a Set Variable command to set the default value of the EMAILADDRESS variable to an empty string, and insert another Set Variable command to set the default value of the SUBMITONLINE variable to TRUE.


```
Installation Script:
Set Variable EMAILADDRESS to
Set Variable SUBMITONLINE to TRUE
//
//  M     M  SSSSSS  IIIIII
//  MM   MM  SS        II         InstallAWARE 2005 for Windows
//  M M M M  SSSSSS    II         Copyright(c) 1996-2005 Instal
//  M  M  M      SS    II         All rights reserved
//  M     M  SSSSSS  IIIIII
//
//              Project Name: $$$
//               Description: InstallAWARE Installation Script
//                  Comments: .NET Application Template Script

// Check system has at least Windows 98
```
Figure 5 Initializing Script Variables

Now type `CTRL+F` and search for the text `Apply Install`. We will use custom scripting to submit the user information right before processing the installation. First insert an `IF` command to test the value of the `SUBMITONLINE` variable, so we respect the end-user's wish on submitting their personal information. The `&Register on the Internet` **check-box** in the registration dialog will write Boolean values to this variable depending on user selection at run-time.



Variable: SUBMITONLINE

Comparison: Equals

☐ Reverse Comparison (If Not)

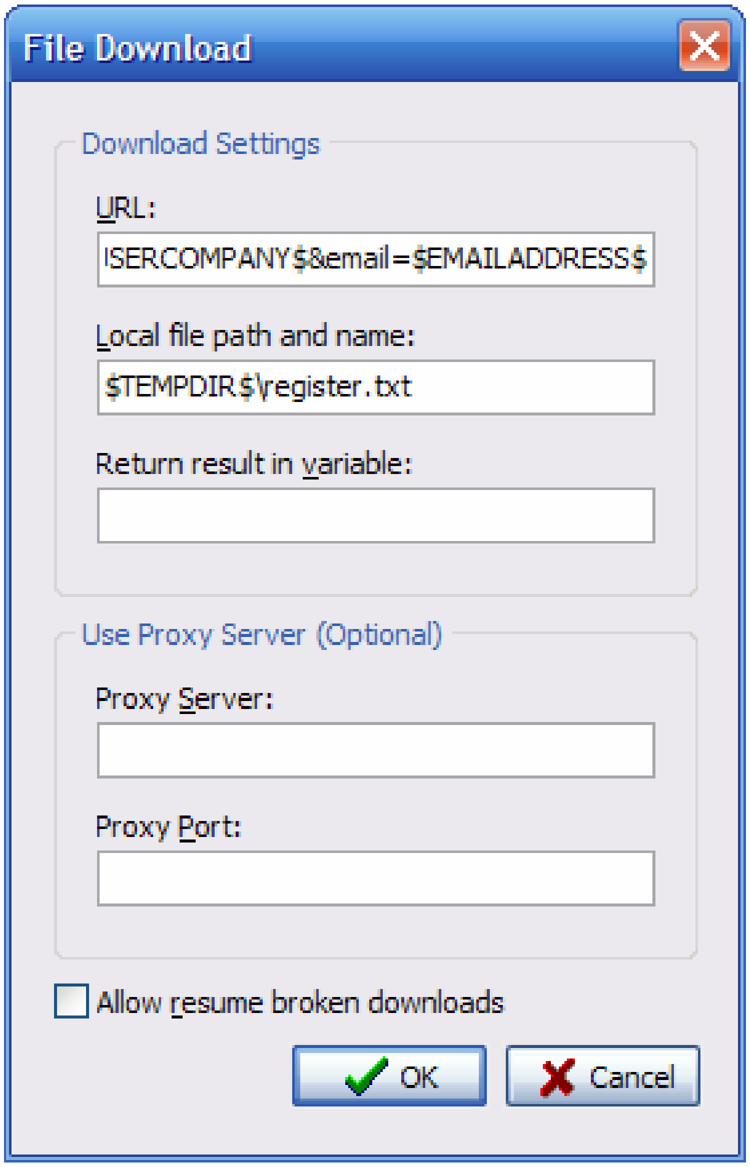
Compare With: TRUE

Figure 6 Inserting a Command into the Script

Next, insert a `Download File` command after the `if Variable SUBMITONLINE Equals TRUE` statement. If you cannot find the `Download File` command, just scroll to the bottom of the commands list, which is where plug-in provided commands are displayed (InstallAware is plug-in extensible, and posting to web servers is provided by the **Download File plug-in**). In the URL field, enter the location of the web server to post user information to, using a form like:

http://www.mimarsinan.com/postuser.asp?name=$USERNAME$&company=$USERCOMPANY$&email=$EMAILADDRESS$.

Notice that variables are dereferenced in InstallAware Script by enclosing them with dollar signs. If you have a variable called VAR, using VAR in your script dereferences the value of that variable. Populate the remainder of the fields in the **Download File** command as you deem appropriate, such as in the figure below.



URL: ISERCOMPANY$&email=$EMAILADDRESS$

Local file path and name: $TEMPDIR$\register.txt

Return result in variable:

Proxy Server:

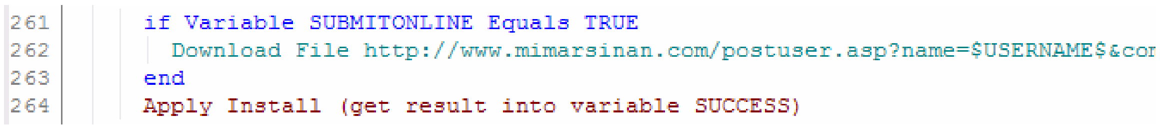
Proxy Port:

☐ Allow resume broken downloads

Figure 7 The File Download Plug-In

In figure 7 above, we used the pre-defined script variable `$TEMPDIR$` to hold the returning file from the post operation. InstallAware Script pre-defines other useful variables that you may use in your script, to see a full list, just search for `Pre-Defined Variables` in the **help index**. Also note that we didn't have to do anything extra to obtain the value of the new edit control we popped on the registration dialog – just referencing the variable is enough.

Finally drag-drop an `End` command from the commands list after the `Download File` statement, so we're closing our conditional `IF` block.



```
if Variable SUBMITONLINE Equals TRUE
  Download File http://www.mimarsinan.com/postuser.asp?name=$USERNAME$&cor
end
Apply Install (get result into variable SUCCESS)
```

Figure 8 Script Commands for Posting User Information

With the custom script commands in place, we're good to go. Let's build and test our setup. Choose **Project->Options**, and then select **Uncompressed Directory Layout** under the **Build** node (we don't want to spend time compressing while we're still developing and testing our setup). Click **OK**, and then choose **Project->Build** to make the setup. Finally, choose **Run->Run Without Debugging** to see our setup in action. Of course, if you want to step through the setup line by line, you can choose **Run->Step**, or set breakpoints and choose **Run->Run** for debugging.

Conclusion

With partial web deployment and powerful scripting, InstallAware provides a great way to deploy .NET Framework applications. The visual views assist in rapidly prototyping your setup, and the code view lets you fine tune existing behavior, such as we did for submitting custom user information. Let your creativity soar with the dialog designer and the script view – making intelligent installers that provide a lasting first impression is now quick and enjoyable.