realtimepublishers.com™

The Definitive Guide™ To

Windows Installer Technology

for System Administrators

Darwin Sanoy and Jeremy Moskowitz

Chapter 6: MSI Deployment Roundup 147
MSI Deployment for Free 147
Sneakernet 148
Batch File Installation 150
Microsoft MSI Deployment with Group Policy 153
MSI Deployment with Third-Party Tools 156
Deploying with the Assistance of Third-Party Repackaging Tools 156
Third-Party Distribution Methods 158
Microsoft SMS and MSI Deployment 159
MSI Deployment and Management with Altiris Client Management Suite 161
MSI Deployment and Management with ON Technology's ON Command CCM 164
MSI Deployment with ONDemand Software's WinINSTALL 166
MSI Deployment with Mobile Automation 2000 170
Summary 173

Copyright Statement

Chapter 6: MSI Deployment Roundup

by Jeremy Moskowitz

They say that getting there is half the fun—if such is the case, you've already experienced half the joy of your MSI deployment journey. Up to this point, you've been through the ins and outs in getting your MSI file "just right." In Chapter 1, you learned about what an MSI file is and why you would want to use one. In Chapter 2, you discovered the tools you can use to create MSI files, and in Chapter 3, became familiar with the internals of the files. In Chapter 4, we explored the best practices for building MSI files, and in the last chapter, you learned how they work inside and outside AD environments. Now, you're ready for the last leg of the journey—the deployment to your client systems of the MSI files you've created. In this chapter, we'll explore myriad free, cheap, and third-party deployment options for getting the MSI file from the administrative workstation where you developed the package onto the plate of each of your systems. Let's start with MSI deployment methods that are free.

> ☞ While reading this chapter, try to determine how and where to standardize. You will be much happier with your day-to-day MSI deployments if you can find one "tried and true" method for your environment and stick with it. All of the following options are approaches to solving the same problem—getting the MSI file onto the user's desktop and installing it. With so many possibilities, you'll benefit from standardizing on one method, if possible. If that's not an option, you can standardize on an option for each specific type of problem. For instance, you might use one method for the home office and another for branch offices.

MSI Deployment for Free

When it comes to deploying your MSI file to your client base, you'll need some sort of deployment mechanism to get the MSI tool to the desktop. Free is good, but be wary as oftentimes you get what you pay for. The free deployment methods each have pros and cons (the pros generally being that the method is free), but on a limited budget, these methods might just be the best or the only option.

Sneakernet

Sneakernet is the tongue-in-cheek description of, basically, running around to each workstation to perform the same task. After you have developed your MSI file, you could, theoretically, burn a copy to CD-ROM and roam the halls, hopping from desktop to desktop to perform the installation. This method is as simple as popping the CD-ROM or floppy into the user's system, and double-clicking to start the installation. However, if the user does not have Administrative rights on his or her workstation, a simple MSI package could generate a host of miscellaneous installation or runtime problems, as Figure 6.1 shows.

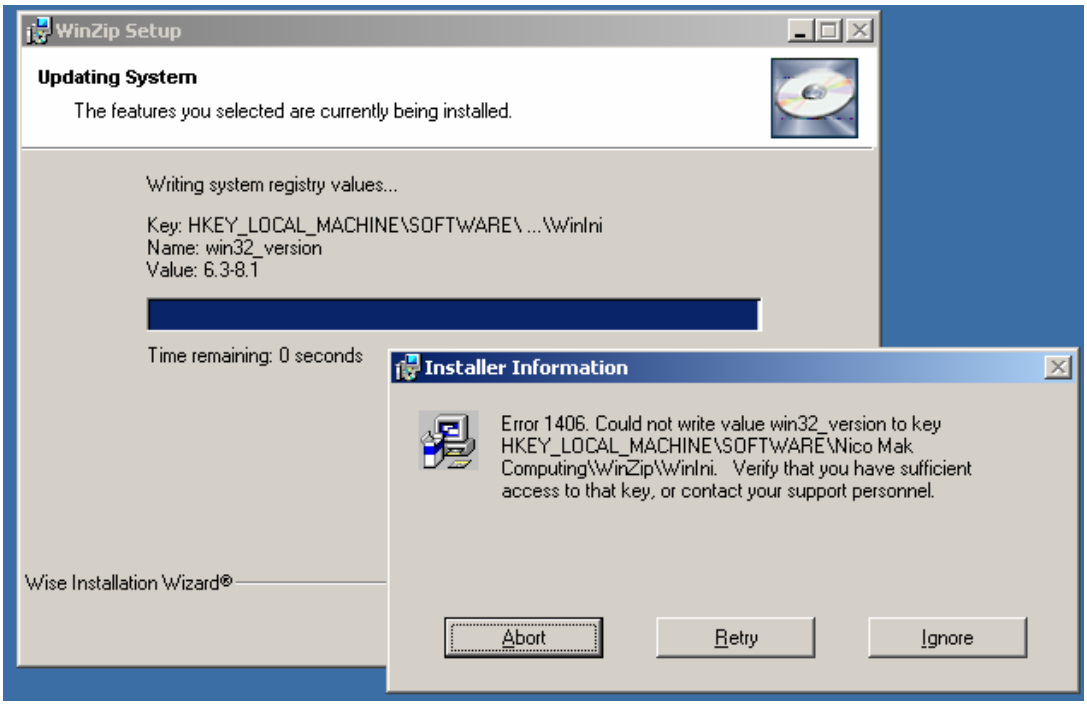

Figure 6.1: Most packages require Administrative privileges.

This scenario represents most environments, in which users don't have administrative rights on their workstations. Thus, you will need to perform the installation on behalf of each user through Sneakernet. To do so, you'll need to elevate the user's installation rights, and perform an installation on behalf of a user by using the runas command. For information about runas, see the following sidebar "The Runas Command."

The Runas Command

The `runas` command has the ability to allow a particular process to run in the context of another user; say, the administrator of the local workstation. Once the rights are in an elevated state, the administrator has the control required to accurately perform the installation. The runas command takes the following form:

```
runas /user:{DOMAIN}\{username} command.exe.
```

After you run this command, the system then prompts you for the password of the user to elevate. If the command you want to run has any spaces, you need to put the command into quotation marks.

To launch an MSI file in an administrative context, you can't simply specify the .MSI file on the command line; instead, you need to call the parent application for an MSI file—MSIEXEC. As you might recall from Chapter 1, one of the command-line syntaxes of MSIEXEC—specifically msiexec /i—will perform an installation. You can use runas and msiexec together, as Figure 6.2 shows.

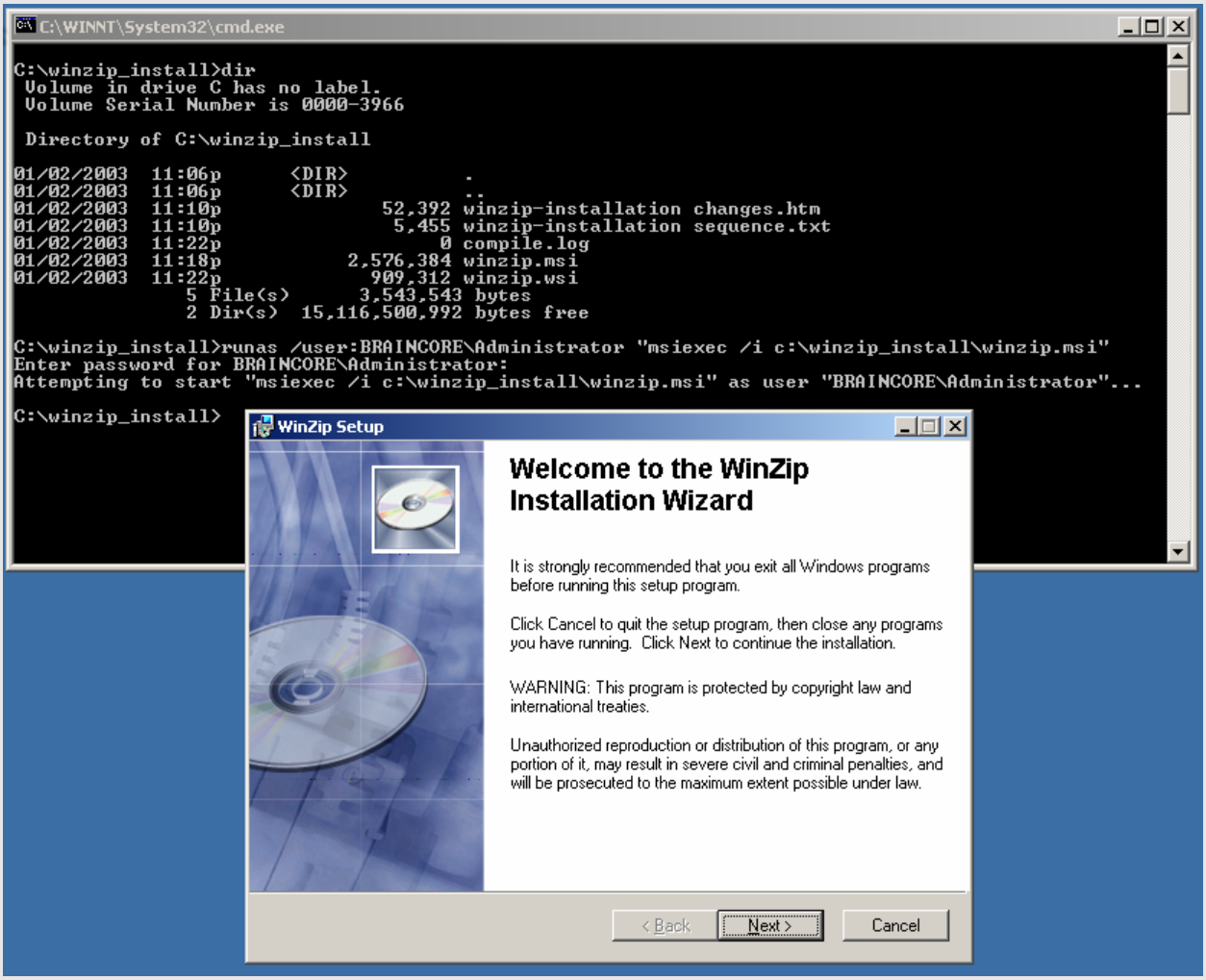

Figure 6.2: The runas command can help install packages that require administrative privileges

Sneakernet will work best for only the smallest environments. After about the tenth workstation, the installations will become very tedious. In addition, only the most finely tuned MSI packages will be successful by using Sneakernet because, in general, your MSI packages will usually ask for qualifying information, such as the directory to install to, the features desired, and so forth. If there is more than one person performing the installation, there's likely going to be more than one method of installing the software. More methods equals more errors equals more problems. Even with the best documentation, the best staff, and the best of intentions, there is still the great chance of error during installations. Unfortunately, even with the best of intentions and direction, when you give an .MSI file to 100 people, you could potentially wind up with 100 different installations.

Sneakernet has the following cons:

- No guarantee that the population to which you're distributing has the software installed in the same manner
- No guarantee that the population has the correct hardware to run software
- No guarantee that the staff loading the software has the correct knowledge to do so

However, Sneakernet does have pros:

- Increased contact between IT staff and end users
- Free one-on-one training opportunities after post-load

Batch File Installation

One method that does the job of deploying MSI packages is the good ol' batch file. Although this technique is a bit raw in practice, many systems administrators have honed it to a fine art. The idea is simple: modify your network logon scripts to determine whether a user is in a particular group, and have the batch file kick off the installation for the user.

To successfully pull off this trick, you can choose from many tools that will help you determine the group membership of the currently logged on user. One such tool is easy to find as it's available in the Win2K resource kit—IFMEMBER.EXE. You can call this tool in a basic batch file or use the more powerful KiXtart, which is a full scripting language and can be substituted for normal logon scripts.

KiXtart 95 is located in the Win2K resource kit. For Win2K, it has been updated to allow for conditional branching based on Win2K AD sites. For instance, if the computer that runs the batch file is in the New York site, then perform some actions, and if the computer that runs the batch file is in the Philadelphia site, then perform some other actions. An even newer version of KiXtart, KiXtart 2001, is available at http://www.kixtart.org/.

For an example, we'll use IFMEMBER.EXE, which will raise the environment variable error-level code to 1 if the membership is met. For instance, suppose you had an NT or Win2K domain-based group named ITGROUP with a member named John, and John logged on, then the error-level code would be set to 1. With this little bit of knowledge, you can get much accomplished. Listing 6.1 shows an example logon script addition that begins to achieve our goal.

```
ifmember DOMAIN\ITGROUP
if not errorlevel 1 goto NOT_ITGROUP
   rem load IT Group's software
EXIT
:NOT_ITGROUP
   rem load other groups' software
EXIT
```

Listing 6.1: Example IFMEMBER logon script addition.

In each part of the branch, you would run an MSIEXEC command with the software you wanted to install. However, if you ran this script upon next logon, once again, MSIEXEC would kick off and reload the software. To prevent this from happening, one option is to enhance the batch file to place "flag files" in strategic locations of the file system for each loaded piece of software. For instance, if you were loading DogFoodMaker5.msi, you might use the logon script to place a 0 byte DogFoodMaker5.txt file on the C:\ drive. In addition, we need one more line of code to jump over the MSIEXEC installation if we've placed the file there before, and presumably we've performed the installation. Therefore, we enhance our logon script a little bit by adding the lines that Listing 6.2 shows.

```
If exist c:\DogFoodMaker5.txt goto BRANCH2
Echo software_loaded > c:\DogFoodMaker5.txt
MSIEXEC /I \\server\share\dogfoodmaker5.msi
:Branch2
rem check to see if next flag file exists
```

Listing 6.2: Additional example text to add to a logon script.

This code will use the command interface's `if exist` command construct to verify that the flag file exists. If it does exist, then jump to another branch of the batch file or exit. If not, create the flag file by using the `echo` command, and create a flag file on the fly—in this case DogFoodMaker5.txt with the works "software_loaded" inside it. Finally, execute MSIEXEC with the /i switch to load the software.

> ☞ You might also choose to make the flag file a hidden file (via the attrib command) to ensure that it's not readily seen by prying eyes.

However, if the MSI installation fails, this addition to the logon script could be a disaster because the flag file is already written but the MSI installation didn't fully complete. You might use other additional checks, such as the known path to the .EXE file, or use additional tools to search the registry for the MSI's GUID.

As we learned in the Sneakernet section, you might run into situations in which you need to launch the MSIEXEC command in the context of the administrator. To do so within a batch file, you would need to expose the administrator password in the batch file with a command such as SU (found in the resource kit), causing a security risk by having the password totally exposed. To combat that, you might want to inspect myriad batch file compilers, which will wrap up batch files into .EXE files, which cannot be read as the logon script goes by. Although this method isn't the most secure for accomplishing this task, it at least presents a deterrent.

Both NT and Win2K domains can optionally execute .EXE files instead of just .BAT files for the logon process if desired.

One such batch file compiler is called Winbatch + Compiler. You can find this tool at http://www.winbatch.com/wb-compiler.html.

As you can see, the process of creating a batch file is easy but not simple. Every time a package changes, you need to keep on top of the script or scripts that are called to do the work. You have a new package? You change the logon script. However, most troubling, is that if something goes wrong during the MSI installation, users might not be able to interpret the feedback to help you assist with troubleshooting. The free nature of batch files might make them a tempting option, but they are usually difficult to work with when you have many MSI files and many machines to deploy to.

Batch file deployment has the following cons:

- No guarantee that the population has the correct hardware to run the software
- No centralized reporting to administrator if something goes wrong
- Batch file maintenance could be burdensome

However, batch file deployment has the following pros:

- Batch file creation is easy
- With enough elbow grease, you could have one logon script controlling all software deployments
- Theoretically, guarantees that a specific population of users (NT/Win2K group) has the same software

Microsoft MSI Deployment with Group Policy

In Chapter 1, we discussed the Windows Installer service built-in to Win2K and newer clients. As a refresher, the Windows Installer service installed in Win2K can be seen in Figure 6.3.

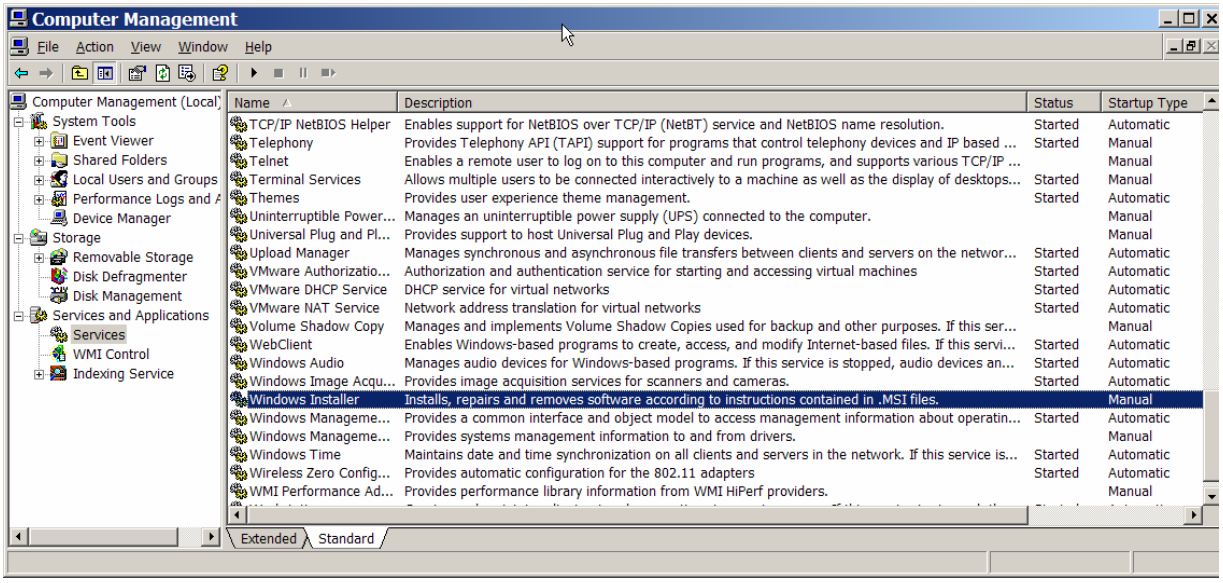

Figure 6.3: The Windows Installer service is built-in to Win2K and Windows XP.

It's not the service itself that's exciting, rather what can be done with the service. This service really comes to life when the Win2K client is used within a Win2K AD. The reason is that the service enables you to centrally deploy MSI files without having to worry about the administrative context of the user.

To connect to and leverage the Win2K service, you'll need to work a little bit. For small to midsized environments, Microsoft suggests that you check out a new technology built-in to Win2K AD. That technology is software deployment via the Group Policy mechanism. This mechanism is part of the Win2K feature set called IntelliMirror, whose goal it is to maintain system state from machine to machine as well as maintain overall system state health.

The software deployment features are very cut and dry—that is, they work under a very specific set of circumstances. First, Group Policy software deployment is typically meant to work with MSI files, though .EXE files are supported under very limited circumstances. At this point, you've wrapped up your applications as MSI files, as we covered in Chapter 2. To use Group Policy to deploy these MSI files, you must have a Win2K AD domain (this environment is likely an uphill battle to achieve). In addition, Group Policy applies only to Win2K and Windows XP clients—so the clients to which you want to deploy software must be running Win2K or Windows XP. Again, this requirement might prove difficult in larger environments. However, if all these conditions are met, you can start to experiment with Win2K Group Policy software deployment.

> 🕮 For an in-depth view of how to deploy each IntelliMirror feature, learn how Group Policy works (and how to troubleshoot it if it doesn't), and additional helpful tips and tricks for software deployment with Group Policy, check out *Windows 2000: Group Policy, Profiles, and IntelliMirror* (Sybex).

> In this section, we won't explore every Group Policy option with regard to software deployment. For your beginning practice, you'll want to start simply with a Win2K AD and a single Win2K Pro workstation.

After you create a new Group Policy object (GPO), you can edit it. When you do, you'll see that there are two software installation settings sections (as Figure 6.4 shows)—one for computer configuration and one for user configuration.

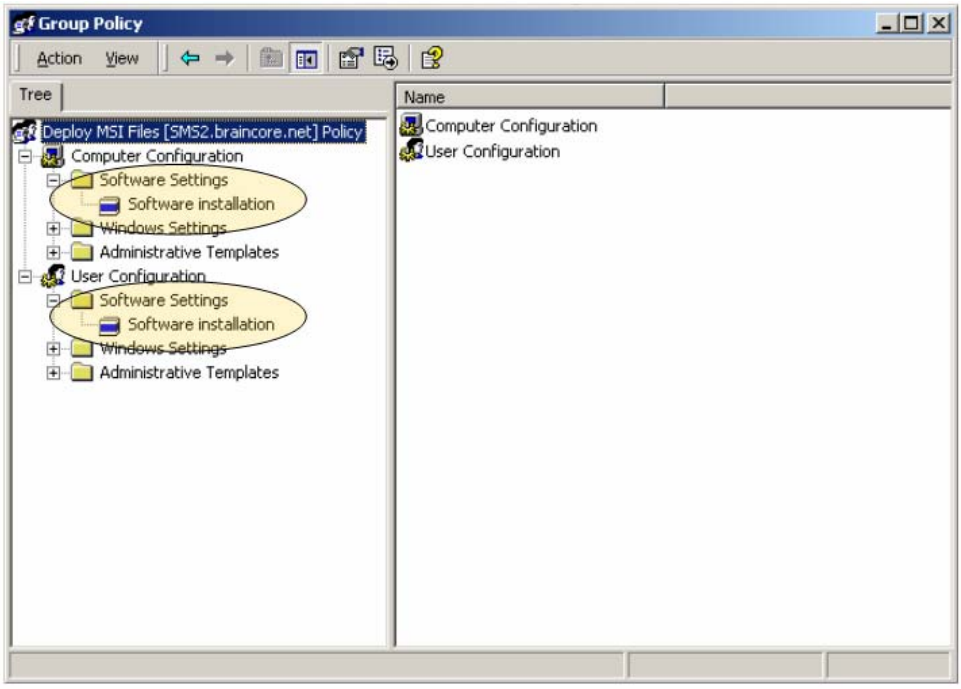

Figure 6.4: You can deploy packages to either computers or users.

At this point, you need to make a determination about how you want your software to be deployed. You can select to distribute a new Package, as Figure 6.5 illustrates.

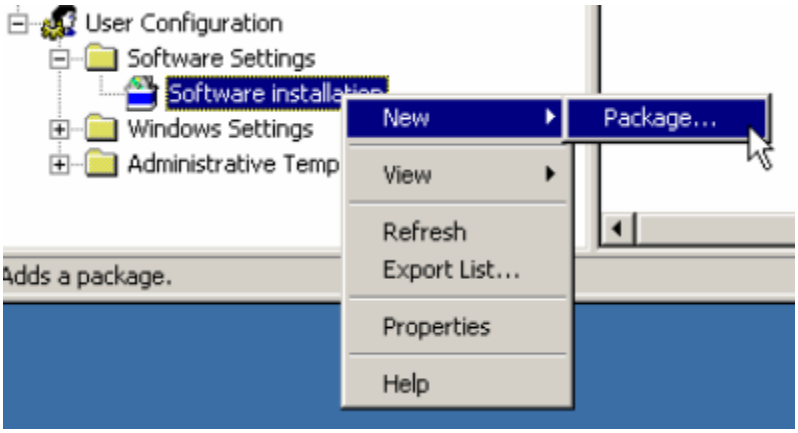

Figure 6.5: Choose either computer or users, and right-click to select New, Package.

You'll then be prompted for the star of the show—the MSI file that you created in the previous chapters. Figure 6.6 shows this window.

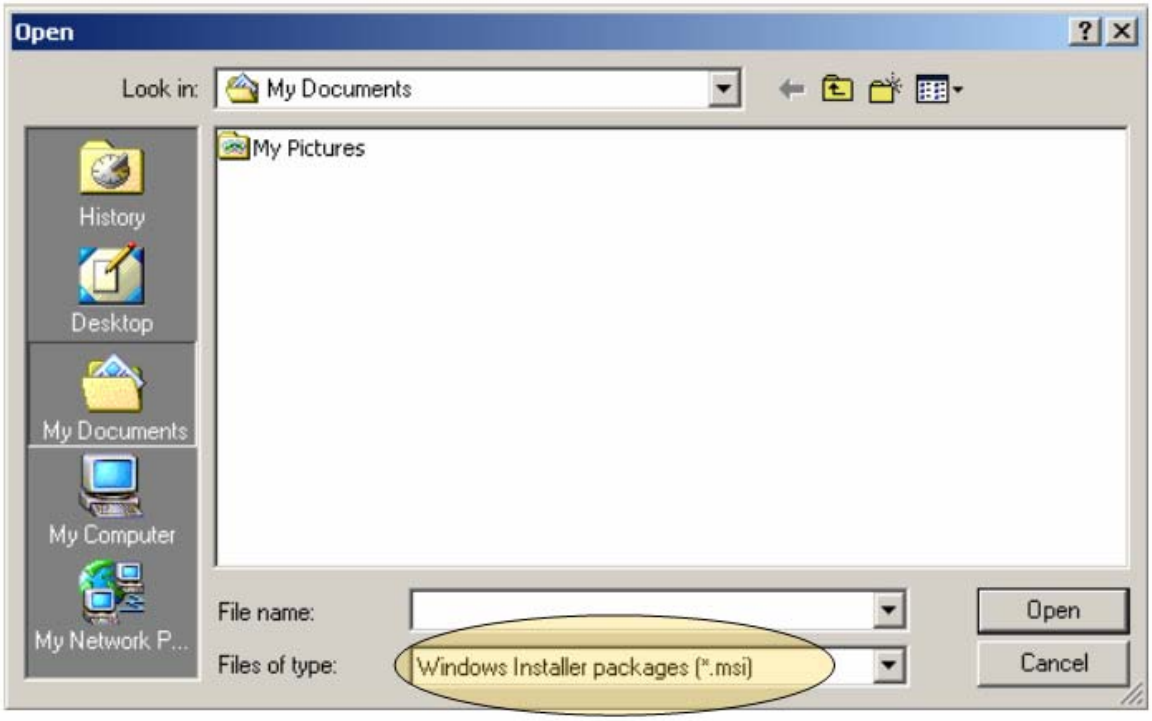

Figure 6.6: Select your MSI file for deployment.

For example, suppose you complete this set of tasks to deploy an MSI package to a user. In such a case, only the icon representing the software will make its way to the user's desktop. The icon is representative of a component that triggers the installation of its parent feature. A single feature can contain any number of advertised shortcuts. Once the user selects that icon, the required components to support that feature are installed in a just-in-time fashion. This installation usually incurs some delay from when the client clicks the desired icon and the application starts. However, this delay is only present for the first installation of the software; subsequent clicks of the icon result in quick loading the installed version of the software from the hard drive.

If you deploy your MSI package through the computer option (rather than user) all features in the MSI are loaded, and they appear loaded for each user who uses the computer. This method gives whoever is using the machine no options about which components are to be installed.

Group Policy software deployment has the following cons:

- Requires Win2K AD
- Requires Win2K or Windows XP clients
- Can be difficult to manage with a large user population as a result of a complex OU structure
- No choice in which components are potentially available

Group Policy software deployment does have its share of pros:

- Software can be deployed to either users or computers via Group Policy in AD
- Deployment is easy
- Theoretically guarantees that a specific population of users (an AD OU, for example) has the same software

MSI Deployment with Third-Party Tools

Another option for deployment of your MSI packages is to use third-party tools. The following sections explore how you can use the assistance of your MSI repackaging tool to hand off your package to a third-party deployment mechanism. We'll also discuss how to deploy packages even if you don't have a third-party package-creation tool to assist you.

Deploying with the Assistance of Third-Party Repackaging Tools

In Chapter 2, you learned about the various tools you could use to create your MSI packages. In that chapter, we discussed several free or cheap options to get the job done; however, to get the job done with finesse, I suggested that you look into purchasing a third-party tool. Indeed, the initial cost of a third-party tool can be mitigated if you use enough of the features it has to offer.

When we discussed the tools, we talked mostly about how tools wrap the packages into proper MSI files for future deployment. Some third-party MSI wrap-up tools have something special: the ability to deliver that MSI in a format that's "ready" for a third-party deployment mechanism to pick up. In other words, the third-party MSI creation tool doesn't directly have the ability to *deliver* the MSI it wraps up; rather it can pass the wrapped file directly to your company's third-party deployment choice. The goal is to have a seamless transition between the tool you use to create your MSI package and the tool you use to deploy your MSI.

To accomplish this task, you can use any number of tools, including Wise Package Studio and InstallShield AdminStudio. InstallShield has partnered with Marimba to offer repackaging and deployment functionality. In this example, I'm using the Wise Package Studio to transition my package to any number of supported third-party tools. To do so, I'll use the Package Distribution wizard, which Figure 6.7 shows.

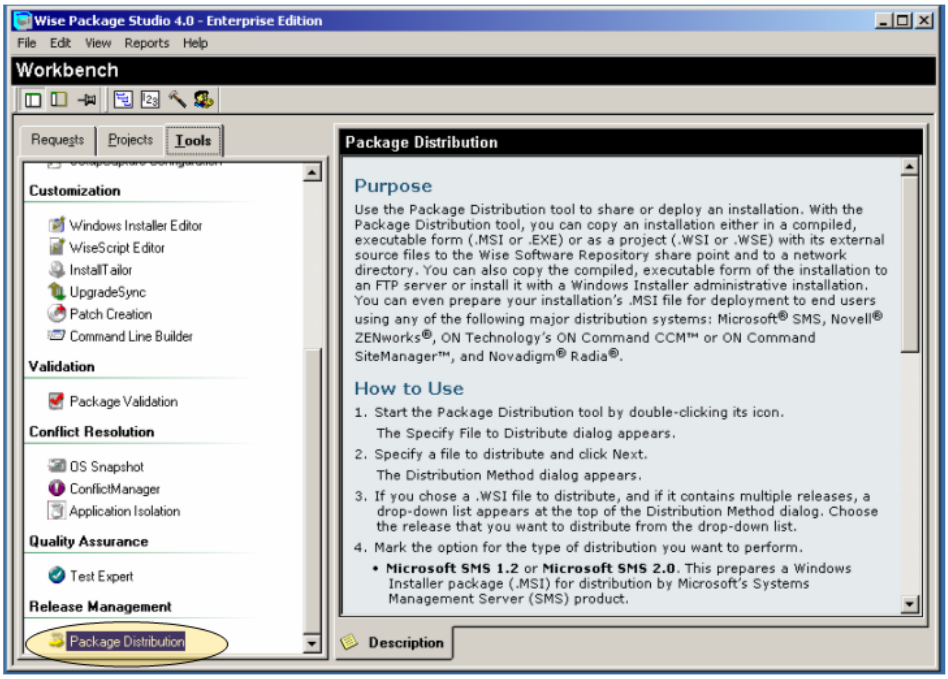

Figure 6.7: The Package Distribution wizard hands off the MSI to any number of distribution applications.

Once launched, the Wise Package Studio automatically detects which third-party distribution programs you have installed. It then presents you with the ability to push to the various third-party programs or select alternative methods to get the package out the door (see Figure 6.8).

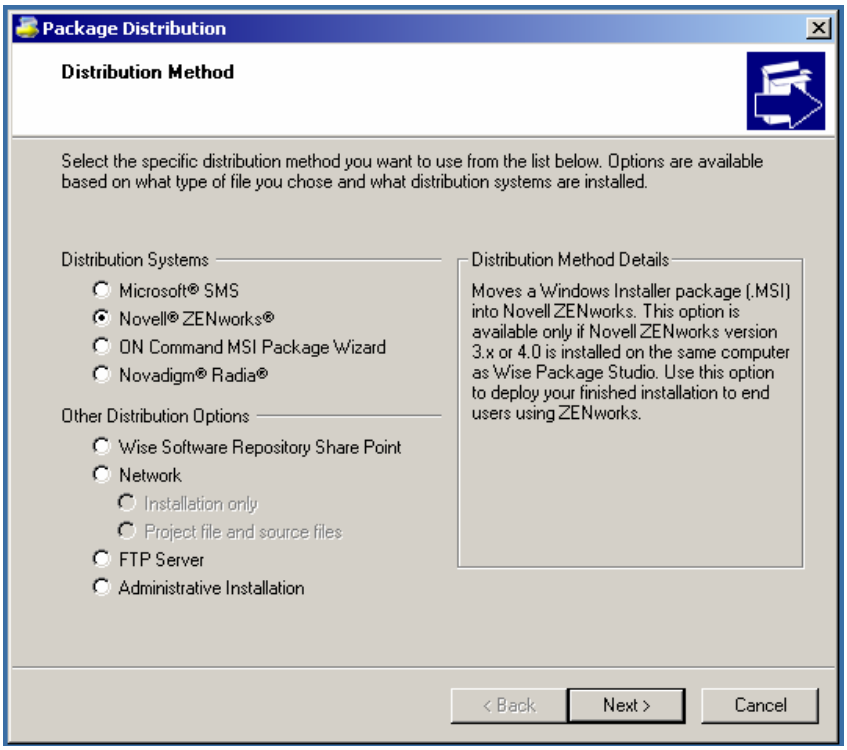

Figure 6.8: Popular distribution systems and other deployment options are presented with this MSI repackaging tool.

For example, selecting Novell ZENworks brings up the ability to easily migrate the package to that distribution method. This feature supports all the normal Novell features, such as Tree Name and Context, as Figure 6.9 shows.

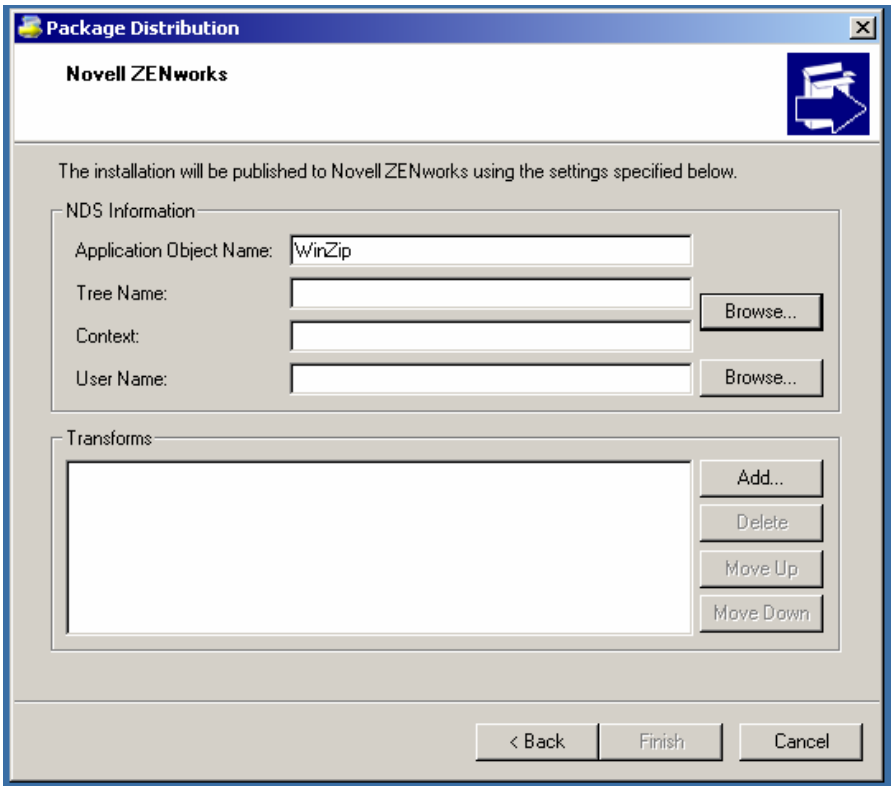

Figure 6.9: One possible way to distribute your package is through a third-party tool such as ZENworks.

Other hand-off functions are similar in that they target the specific third-party tool with exactly what's required. Each hand off is a bit different, so be sure to test each option if you have multiple third-party distribution methods in conjunction with an MSI repackaging tool such as this.

Third-Party Distribution Methods

You might not have a third-party MSI creation tool to assist you in handing off to your third-party distribution method. For instance, you might have used one of the free MSI creation tools—such as WinINSTALL LE or the SMS Installer—that don't offer the ability to hand off the resulting MSI file to any distribution mechanism. In either case, you need to learn about how to deploy your MSI file. This section examines several third-party distribution methods for deploying MSI files.

As I talk about third-party tools, my goal is educate you about some of the possibilities available to your for deploying MSI files. The programs we'll explore offer myriad functionality—software inventory, software metering, remote control, and even OS deployment features! However, I'll stick to discussing the MSI-centric features of these solutions.

Each tool takes a slightly different angle toward how they handle MSI file deployment and management. Hopefully, one of the tools I discuss will give you ideas about how to best deploy MSI packages in your organization.

> 🕮 I simply don't have enough space to cover each tool that can deliver MSI files. However, you can find a list of deployment (and repackaging) tools at http://appdeploy.com/tools.

Microsoft SMS and MSI Deployment

SMS is Microsoft's offering to deploy any and all files to your Windows desktop. SMS is robust in many ways, including the ability to deploy files to any 32-bit Windows desktop. I've categorized it as a third-party MSI distribution solution because it is an add-on product (you cannot do most of what SMS does "right out the box" with just Win2K).

A little background about SMS: Out of the box, SMS 2.0 wasn't originally intended to deploy MSI files. Rather, it was mostly meant to deploy .EXE files. Indeed, for years, the SMS repackaging tool (the SMS Installer tool) didn't create MSI files, it only created .EXE files. Only recently, as we explored in Chapter 2, did the SMS Installer repackaging tool offer MSI file creation functionality.

> As mentioned in Chapter 2, not every MSI function works correctly when creating MSI files using the SMS Installer tool. Therefore, you might want to consider using a full-featured third-party repackaging tool to create your MSI files. Then you can use the information here to continue to deploy that repackaged MSI file using SMS.

Deploying MSI files with SMS is somewhat different than deploying .EXE files with the tool. You need to prepare a little before doing wholesale deployment of your MSI files with SMS. First, you need to deploy the latest Windows Installer bits to your PCs, which can be a monumental task. Like an MSI, you must deploy Windows Installer as an .EXE deployment.

> 🕮 Get the bits you need for Windows 9x and NT 4.0 at http://www.microsoft.com/msdownload/platformsdk/instmsi.htm. The bits for Win2K and Windows XP are already installed, but you can update them if you want. Check out Chapter 1 for a refresher of which versions of Windows Installer are available.

When it comes to actually deploying your MSI files, you've got a little more work to do. First, you need to set up a share point with read access to everyone. Next, you need to perform an Administrative Installation of your MSI file into that share. Recall from Chapter 2, that an Administrative Installation unpacks the guts of the MSI file and dumps the necessary contents to a directory.

> ☞ Recall that you can run an Administrative Installation using MSIEXEC /a such as
>
> ```
> MSIEXEC /a myfile.msi
> ```

In the example that Figure 6.10 shows, I've performed an administrative install in the directory called e:\csav_distry, and created a new package pointing at the distribution source.

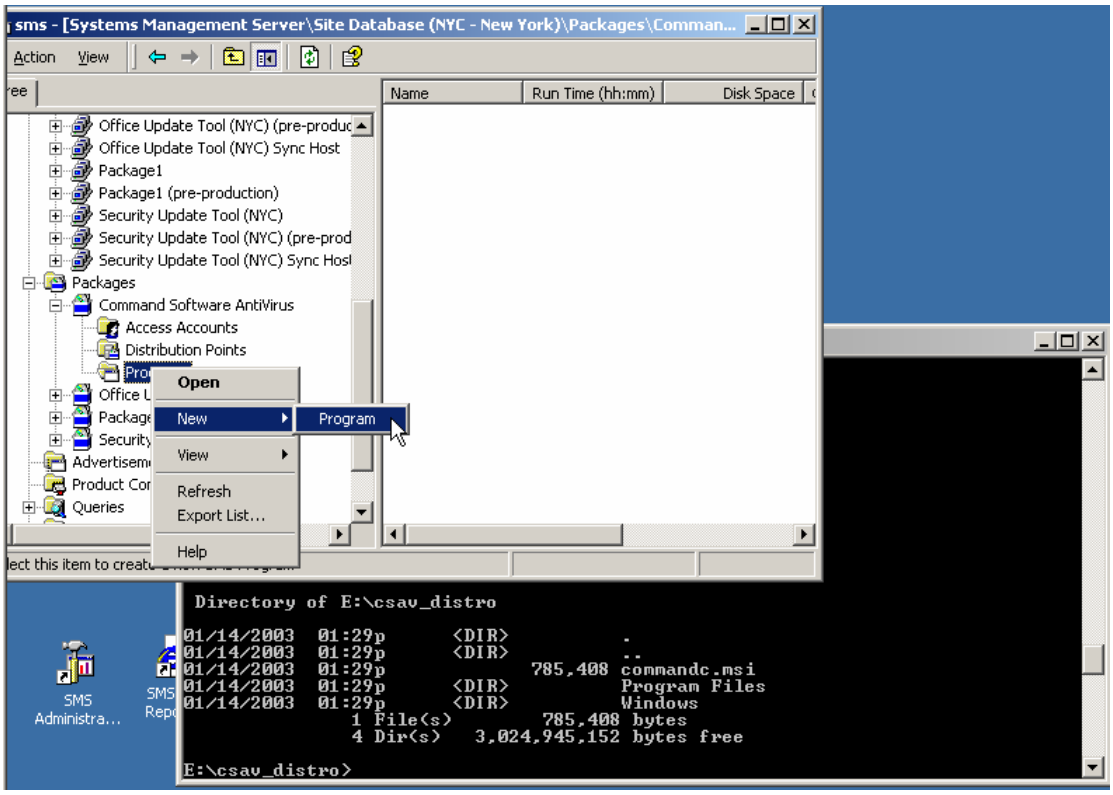

Figure 6.10: After creating your Administrative Install, you're ready to deploy MSI packages with SMS.

Once the files are in the directory and the SMS package is defined, you can continue to leverage your knowledge of the MSIEXEC syntax to deploy the packages to your SMS clients. You'll create a new program, as Figure 6.10 shows. Recall that to perform an installation of a particular MSI file, you use the msiexec /i <filename.msi> syntax (you must also specify a transform and/or command-line switch to automate the installation). This syntax is the same syntax that you pump into an SMS program to perform the installation, as Figure 6.11 illustrates.

> In SMS, a "program" is defined as the command line used to kick off a package.

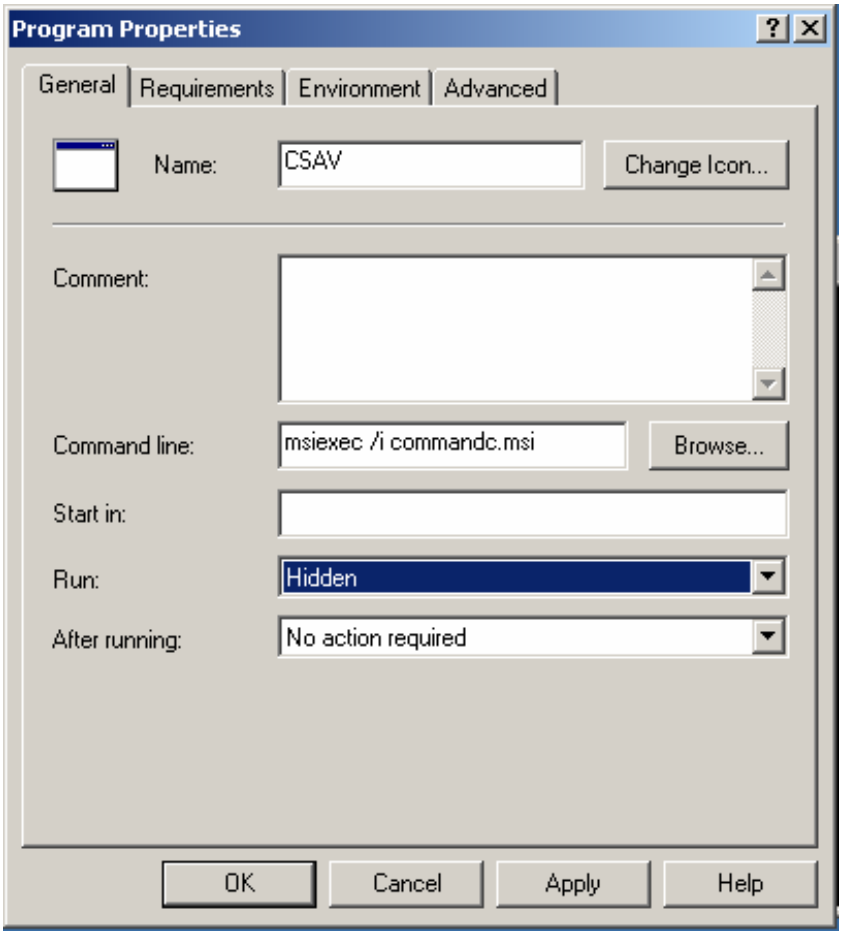

Figure 6.11: Use the MSIEXEC command line to assist with your deployment using SMS.

There are many additional options at your disposal for configuring how to run an MSI file at an SMS client. For example, if you supply an /M switch, you can generate a status MIF file, which SMS will pick up and inventory to let you know what happened during the install. Using the information you already know about MSIEXEC and Windows Installer, it's a lot easier to perform the task.

> ☞ You might want to check out Microsoft's additional notes about deploying MSI files when using SMS and Windows Installer at http://www.microsoft.com/smserver/techinfo/deployment/20/deployosapps/deploymsi.asp.

MSI Deployment and Management with Altiris Client Management Suite

Altiris' Client Management Suite is another third-party method to distribute and manage your MSI files. The Client Management Suite has many layers; we'll focus our attention on its MSI hot spots. Three parts of the Client Management Suite deal with MSI files:

- The deployment solution can send out an entire OS image. Once the image is deployed and finished, the deployment solution can pull an MSI package and install it before the user logs on for the first time.

- The software delivery solution is the suite's main software delivery mechanism. It's similar to SMS in functionality and terminology. For instance, to deploy MSI files, you use the MSIEXEC command when you set up your packages and programs.
- The application management solution lets administrators scan machines to determine the health of their distributed packages (either distributed via Altiris or some other distribution mechanism). This process can perform a scan of the machine, then report about the general MSI health of the system, as Figure 6.12 shows.

> Altiris also offers an MSI/EXE repackaging tool similar to the SMS Installer that creates both .EXEs and MSI files. This tool is called Rapid Install, and similar to earlier versions of the SMS Installer, it requires an external .EXE to .MSI conversion utility.

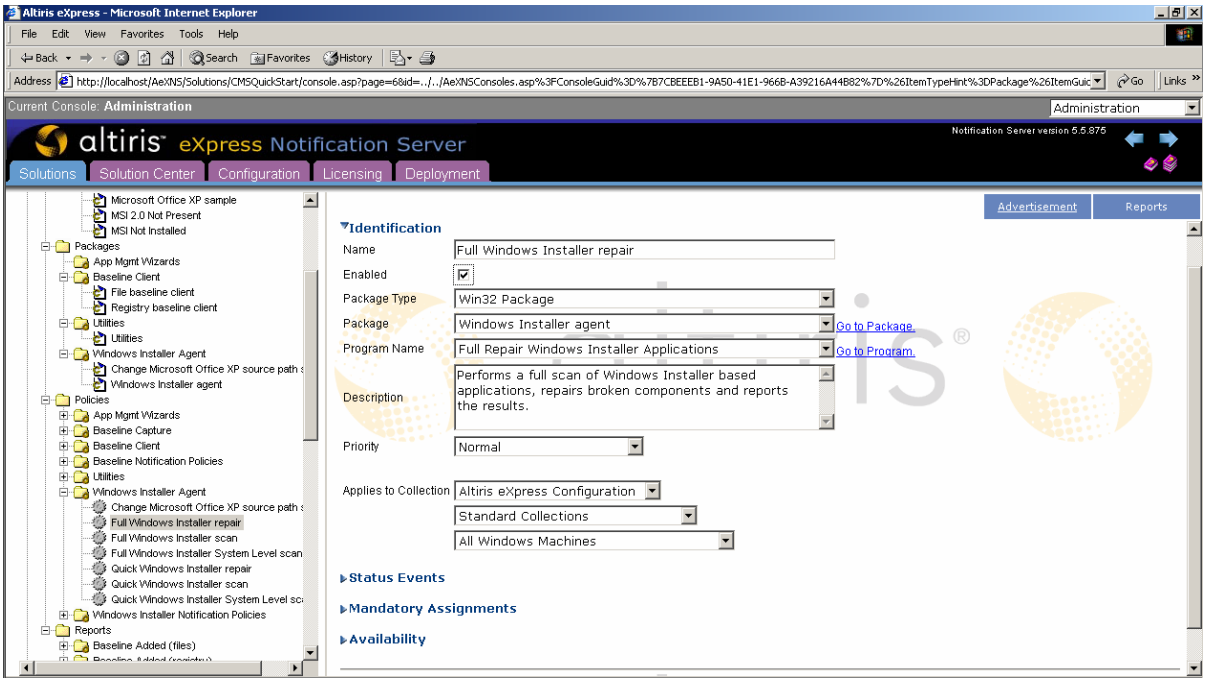

Figure 6.12: Altiris' suite can help manage and fix MSI files once you deploy them.

If the deployment of one package has broken another package, the Windows Installer might not know about this mishap and won't fix it until the original program is launched. The suite's application management solution can scan the system and learn which MSI packages are currently installed on the machine. It can then run through all the MSI interdependencies and links to determine what is broken. Finally, it can trigger the Windows Installer service to perform an automatic repair. Either quick or full scans can be chosen, as Figure 6.13 shows. A quick scan examines the MSI's key path and attempts a fix. A full scan tracks down all associated components and DLLs of an MSI and attempts to ensure that every bit is back in order.

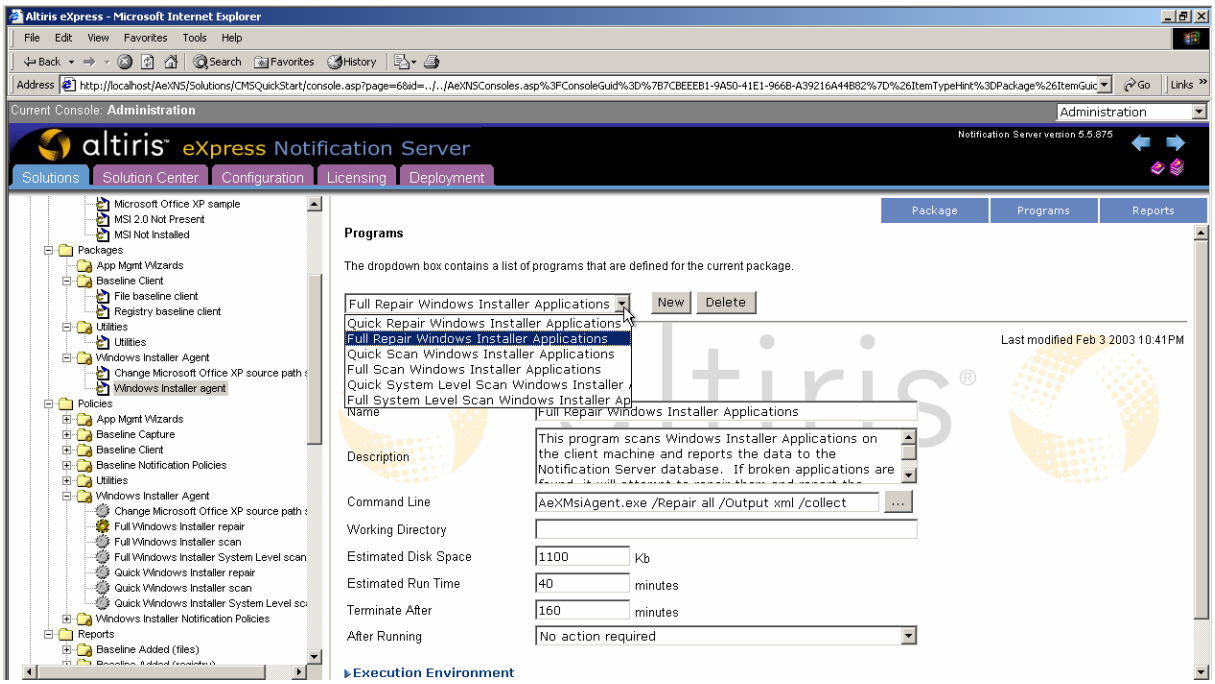

Figure 6.13: The suite's repair option allows for "Quick" or "Full" MSI scans.

Also available is the ability to create a baseline of a known good local machine, then do comparisons against a broken machine. You can compare which MSI components are registered in the registry. Simply find the broken component, and you're off to the troubleshooting races.

Altiris also provides several reports that can help explain how packages are interacting with the Windows Installer service on each client system, as Figure 6.14 illustrates. For instance, you can get a quick view of *Most frequently broken products* or *All repair attempts in the last N days*.

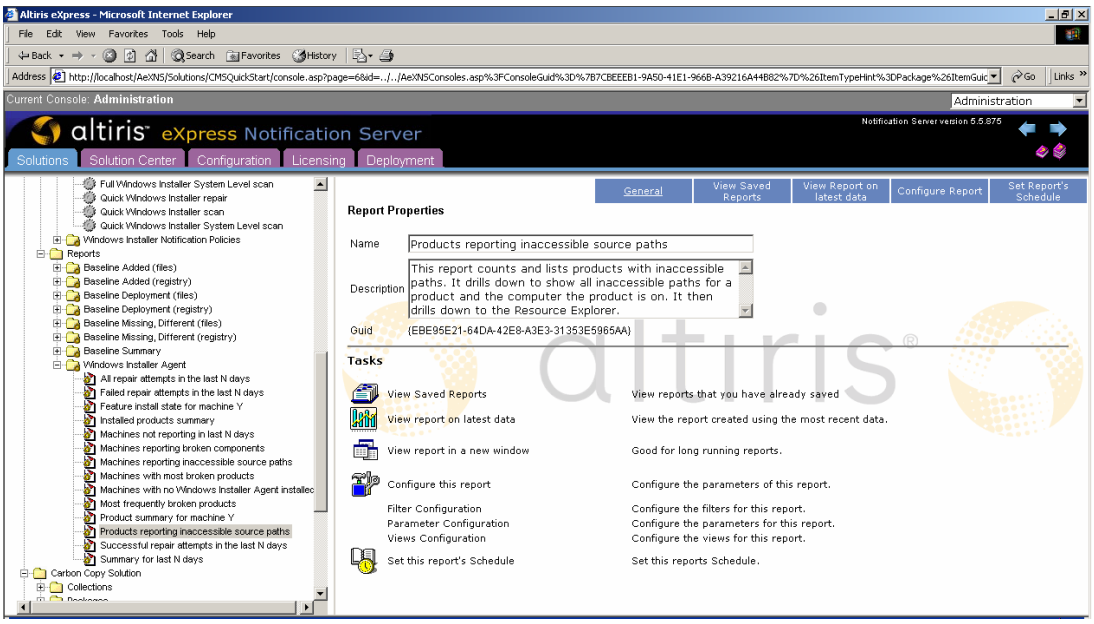

Figure 6.14: Altiris has facilities to run reports about the status of the Windows Installer service.

Altiris' suite continues with a rigorous MSI inventory. For instance, it can obtain a list of all MSI features not installed. For example, perhaps when a user loaded an MSI file, the user chose not to install a particular feature—say, the Help file. Altiris can examine which machines are missing the MSI feature, then simply send a new job to the client computers to pull the feature from the source to ensure that all clients are the same.

MSI Deployment and Management with ON Technology's ON Command CCM

Another third-party technology that has the ability to push MSI files is ON Technology's ON Command CCM (change and configuration management). This product provides unique MSI deployment functionality. For instance, CCM has the unique ability to bypass the entire requirement for an Administrative Installation and lets you distribute complex MSI packages directly from the source.

For example, in the windows that Figure 6.15 shows, I'm telling CCM to deploy Microsoft Office XP with FrontPage. The MSI file is called PROPLUS.MSI (in this example, I'm grabbing it directly off the CD-ROM).

Figure 6.15: CCM allows for you to bypass the administrative installation step.

CCM goes one step further: It has the ability to add a transform file you create right into the definition as well as essentially create customized transforms on the fly! If you do not choose an MST file, CCM will crack open the MSI file and locate the changeable variables. Instead of going through the lengthy customized MST creation process for your application (or by using a third-party MST creation tool), you can bypass it all and use CCM (see Figure 6.16).

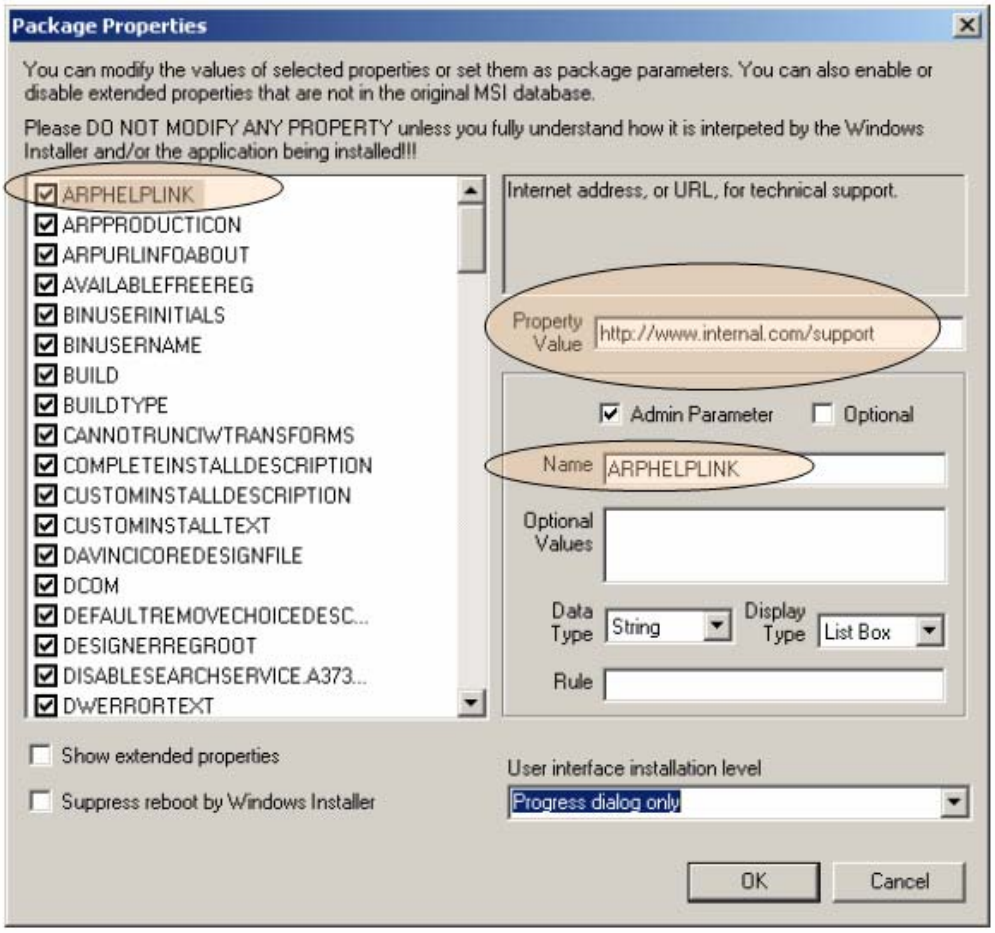

Figure 6.16: You can tap directly into an MSI's properties and change them on the fly.

In my example, I've changed the value of Office's ARPHELPLINK, which defines the location of online Help. Usually, it has a link to Microsoft. However, using CCM, I'm changing the location to my own company's internal support Web site.

When it comes time to target the package to your client systems, CCM has an additional notable feature. That is, it allows you to target the machines you want, and, in spreadsheet fashion, simply select which systems will get which MSI features once they hit the target machines. As Figure 6.17 illustrates, each of these three systems (gx150x, nx1, and MINI) will have the Office XP features in one of the MSI potential installation states.

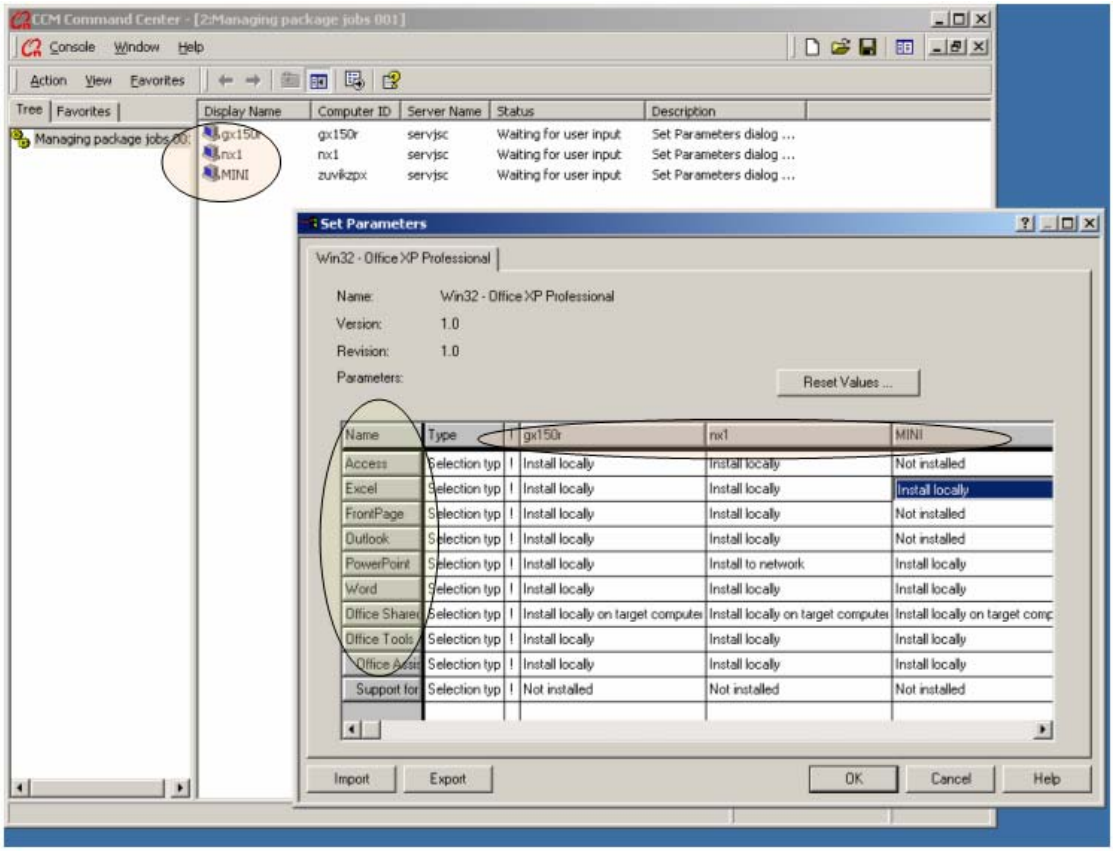

Figure 6.17: You can specify which computers get which options on the fly in spreadsheet form.

MSI Deployment with ONDemand Software's WinINSTALL

Another product that can help deploy your MSI packages is ONDemand Software's WinINSTALL (formerly owned by VERITAS). As we saw in Chapter 2, WinINSTALL has a companion product called WinINSTALL LE that can help create MSI packages, which you can then deploy via any distribution mechanism you desire—not just through WinINSTALL. Like its companion product, WinINSTALL has gone through a long history of ownership. It was developed by ONDemand Software, then Seagate bought WinINSTALL and WinINSTALL LE, then VERITAS bought Seagate. Because VERITAS wanted to focus more on backup and recovery, WinINSTALL and WinINSTALL LE were sold back to a newly reformed ONDemand Software—and the products are back in development. WinINSTALL's strength with MSI packages is that you are able to view and manipulate all the features and components of an MSI package that you create, as Figure 6.18 shows.

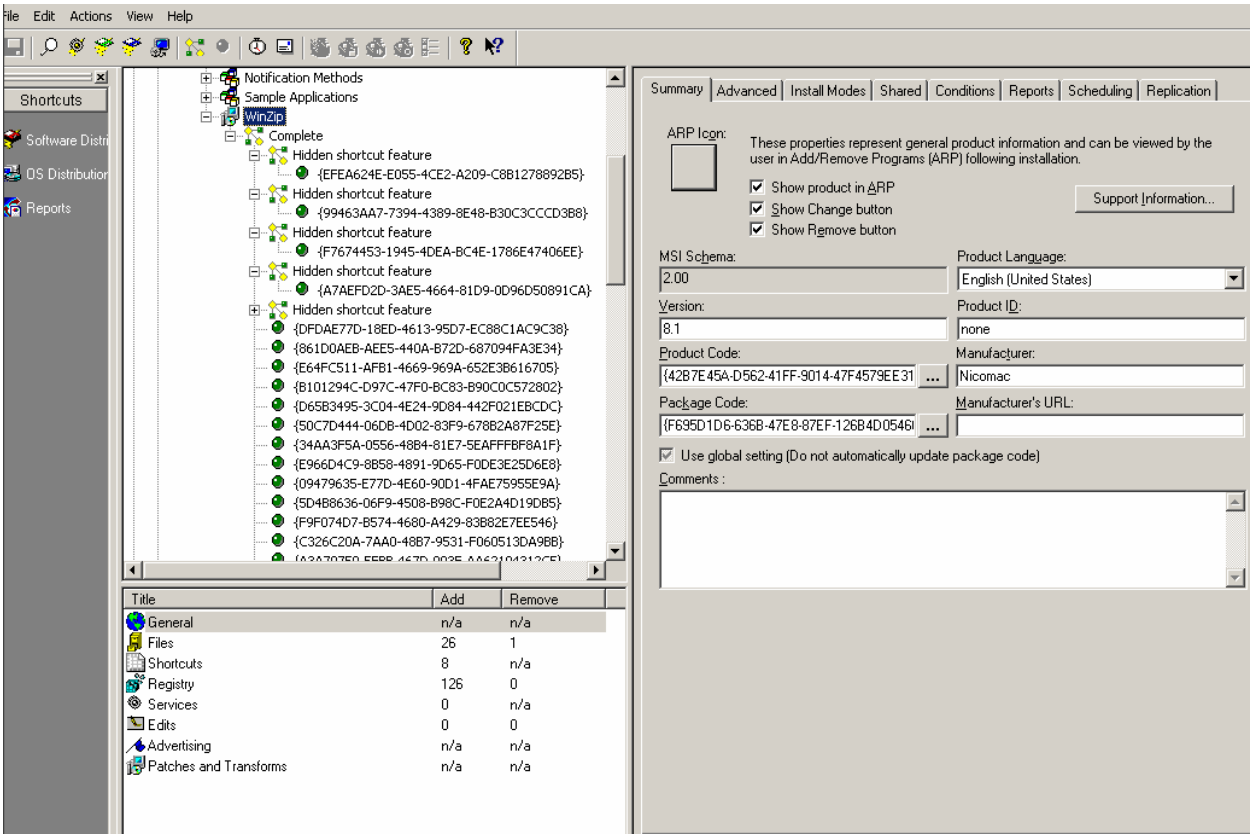

Figure 6.18: WinINSTALL displays all features and components of each deployable MSI file.

Once an MSI file is loaded into the WinINSTALL Console, you can see each of the features (represented in Figure 6.18 with the little DNA-type icon) and each component (represented with a green dot). In addition, at the feature level, you can easily make the initial determination whether a feature should be installed on first use, loaded and run from the hard drive, or an alternative option (see Figure 6.19).

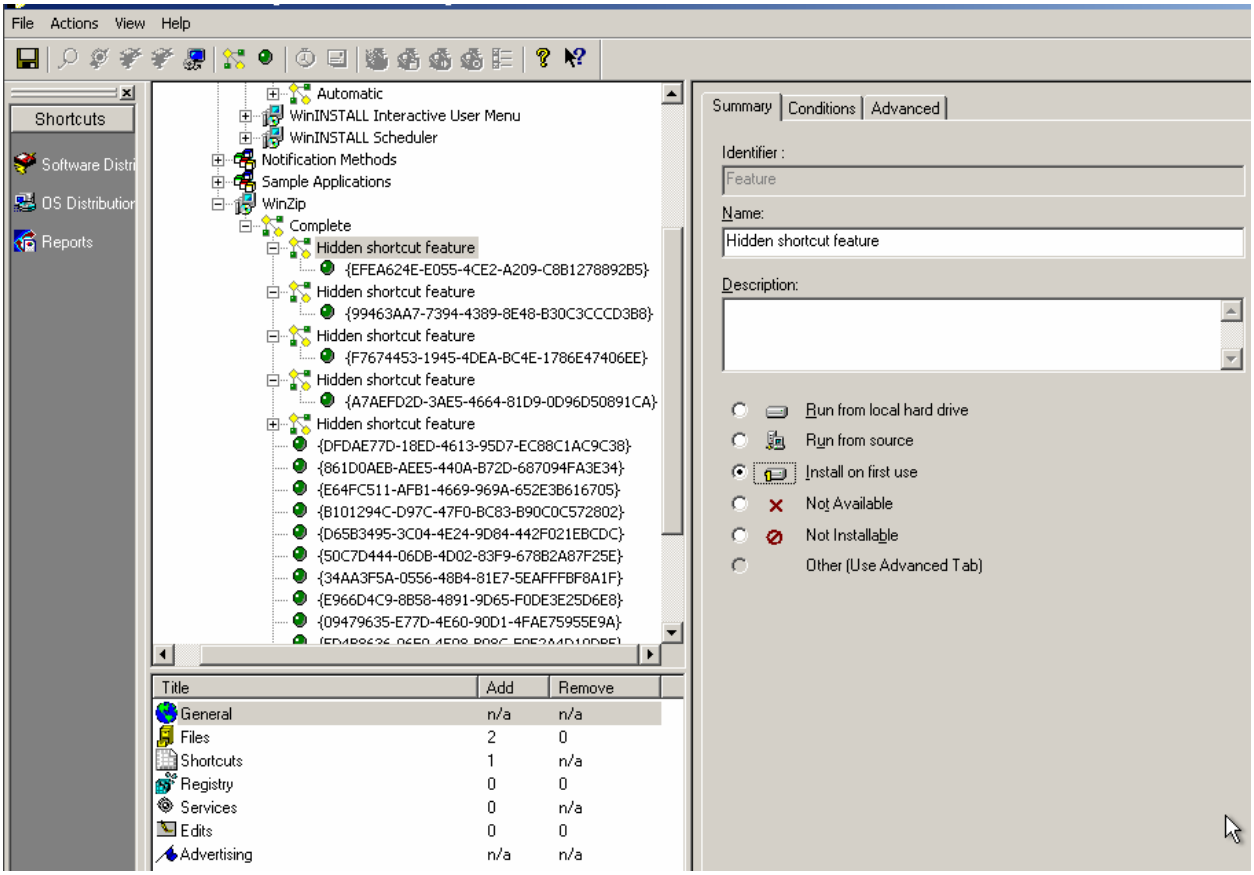

Figure 6.19: You can easily control the MSI installation state.

You can even dive in and take it a step further by determining which features and components are going to be installed in what manner. For instance, you might only decide to install the component related to the Help file (for example, to save space or eliminate confusion) of an application if the ProductLanguage was English and if the machine was at SP1 or higher. By drilling into the component and specifying the criteria you want to match, you can determine which components will be installed (see Figure 6.20).

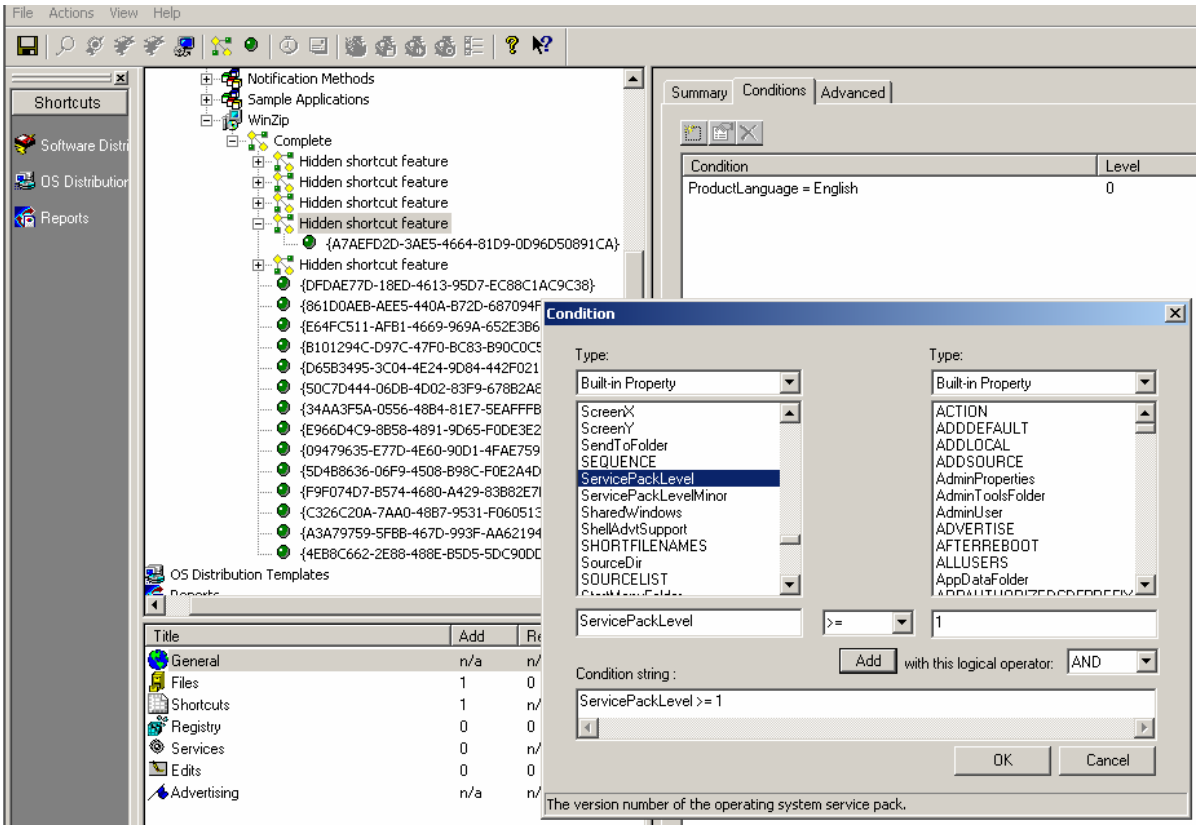

Figure 6.20: You can control how each feature and component is installed.

In addition, WinINSTALL can do a compression upon your MSI file based on the features you chose to install, as Figure 6.21 shows. That way, you're only sending the necessary data to the clients that need it; not the entire package.

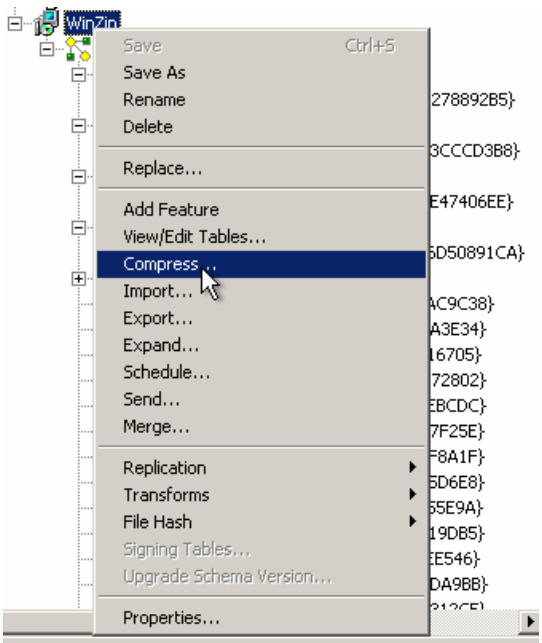

Figure 6.21: You can compress your MSI file before shipping it off to destination servers.

MSI Deployment with Mobile Automation 2000

Mobile Automation 2000's claim to fame is that it was the first kid on the block to perform a "trickle down" deployment, through which users on slow links can download a package to install in "drips and drabs," then once fully downloaded, the job kicks off and runs. Imagine trying to download, say, Office XP over a 56k modem in one fell swoop, and you can understand how this feature might come in handy. Mobile Automation 2000 also has the ability to send software to all sorts of Microsoft and non-Microsoft devices, such as Windows CE, Palm, and Blackberry, in addition to standard Windows-based PCs. Mobile Automation 2000 builds upon its history and is now a robust MSI package deployment mechanism.

Mobile Automation 2000 allows for a full script of any sort of install—including MSI files. In the example that Figure 6.22 shows, the Execute MSI Package command is pulled off the Command List as a possible first command to start an MSI deployment.

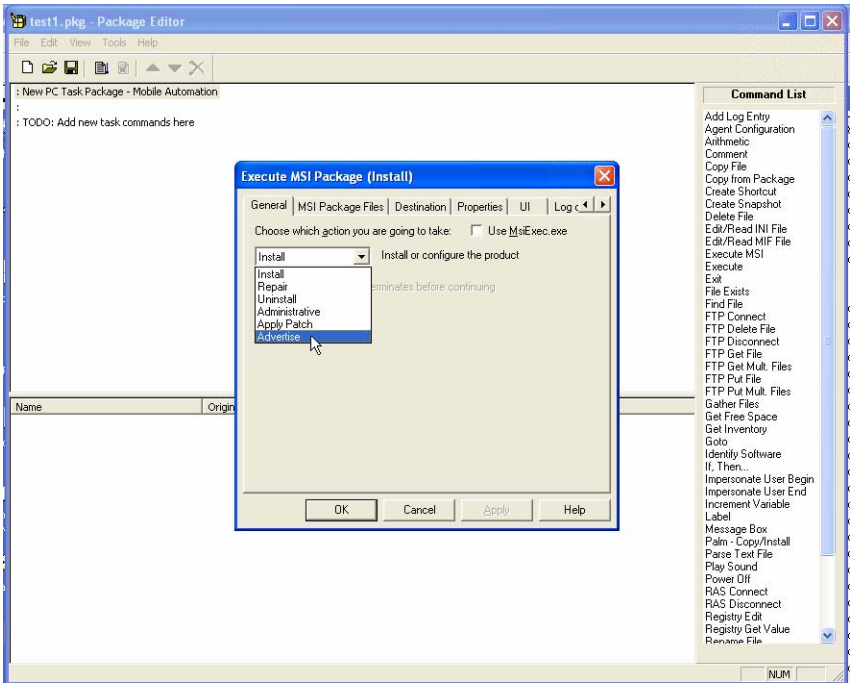

Figure 6.22: Mobile Automation 2000 allows for pre and post scripting during an MSI installation.

Note that the Use MsiExec.exe check box is clear by default. The reason is that Mobile Automation 2000 has the unique ability to install packages in two ways. When the check box is selected, Mobile Automation 2000 will use MSIEXEC to install the application. With the check box clear, Mobile Automation will use the direct Windows Installer APIs to perform the installation.

Mobile Automation 2000 also has the ability to grab the state of certain attributes of the package, such as if the package requires a reboot. In the example that Figure 6.23 shows, if the package requires a reboot, you could make some additional changes before actually going forth and performing that reboot. Doing so could save time during each and every deployment. Imagine how much time you could save if you could squelch every MSI reboot until absolutely necessary, perform the changes you wanted (which could also require a reboot), then reboot just once when you're ready.

Figure 6.23: You can inspect the state of several MSI variables, such as whether a reboot is needed.

Finally, because Mobile Automation 2000 has the ability to use the direct Windows Installer API calls (instead of just calling MsiExec.exe) to install the package, there is a huge jump in the level of detail provided about precisely what happens during the install time of a package. Every possible status code result is directly returned from the API function call and handled as a branch condition, as Figure 6.24 shows. (The API error code list is quite long and only a fraction can be seen in the drop-down box in the figure.)

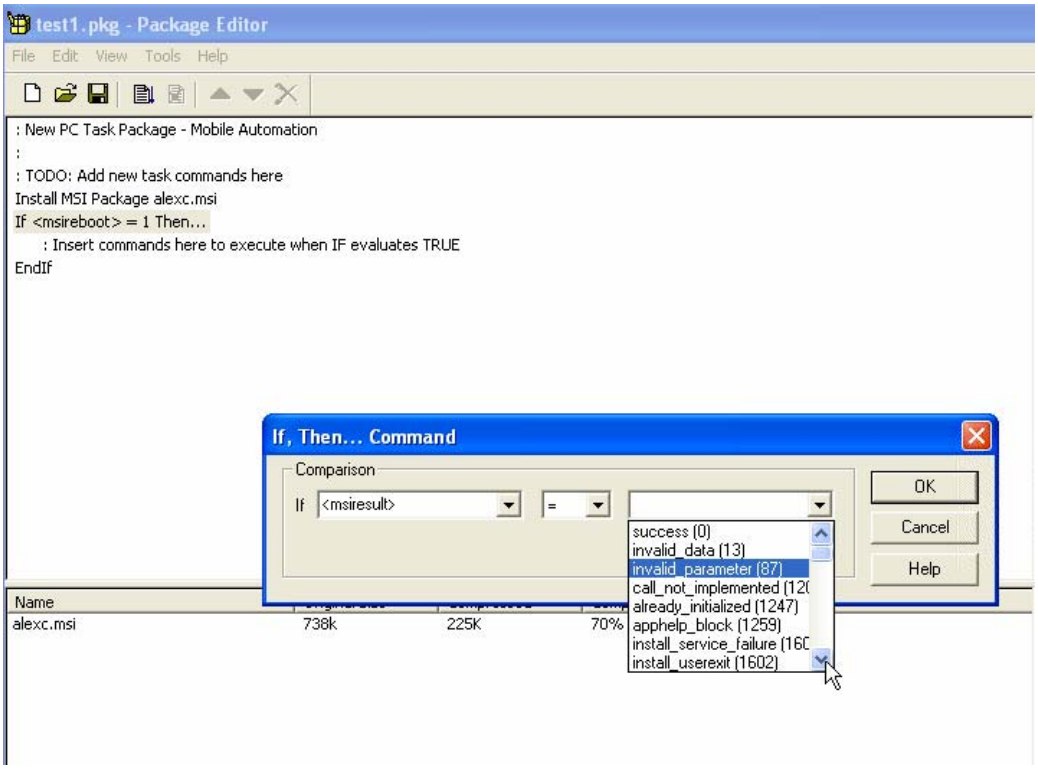

Figure 6.24: Mobile Automation 2000 can branch on the precise error returned from the Windows Installer API function for installation.

Once you know the error code, you can conditionally branch to handle specific conditions or simply make note of the error in a log file to help with MSI deployment troubleshooting. In the example that Figure 6.25 shows, I'm adding a warning flag to my Mobile Automation 2000 log file if the error condition known as <msiresult> is anything other than "success (0)."

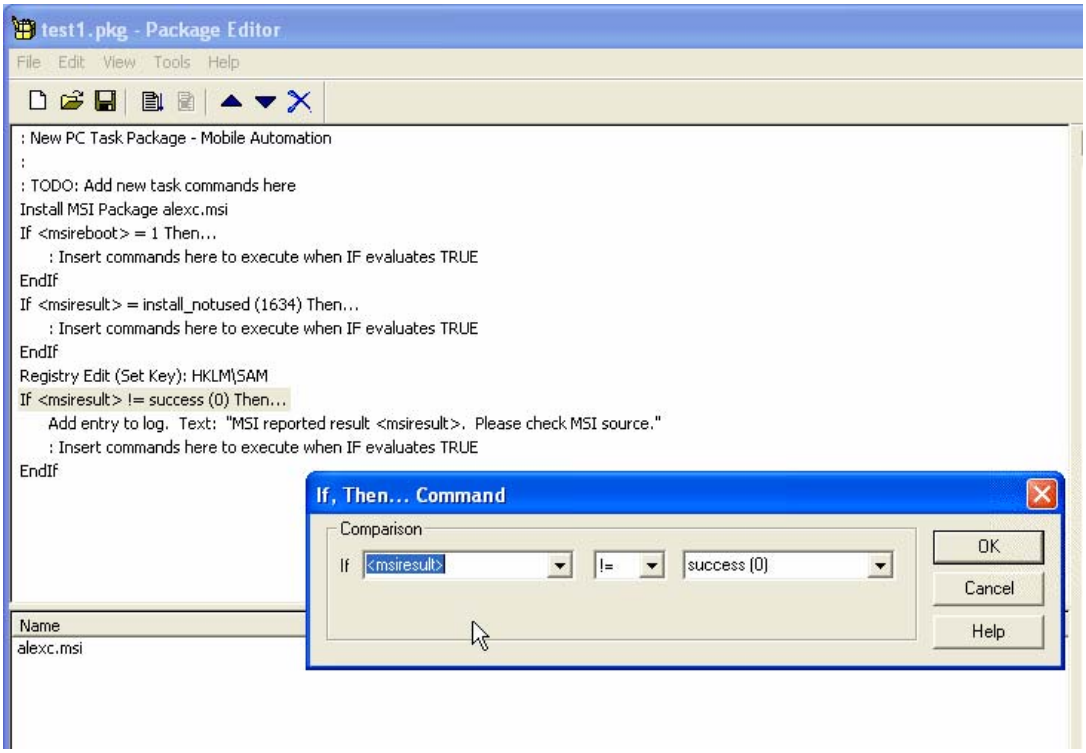

Figure 6.25: You can customize the error logs to demonstrate specifically what went wrong with an MSI installation.

Additionally, Mobile Automation 2000's inventory function can grab MSI data directly from the desktop—such as how often certain MSI packages are being used (see Figure 6.26). Finally, when the package is ready for uninstall, it can be uninstalled with a proper MSIEXEC uninstall string.

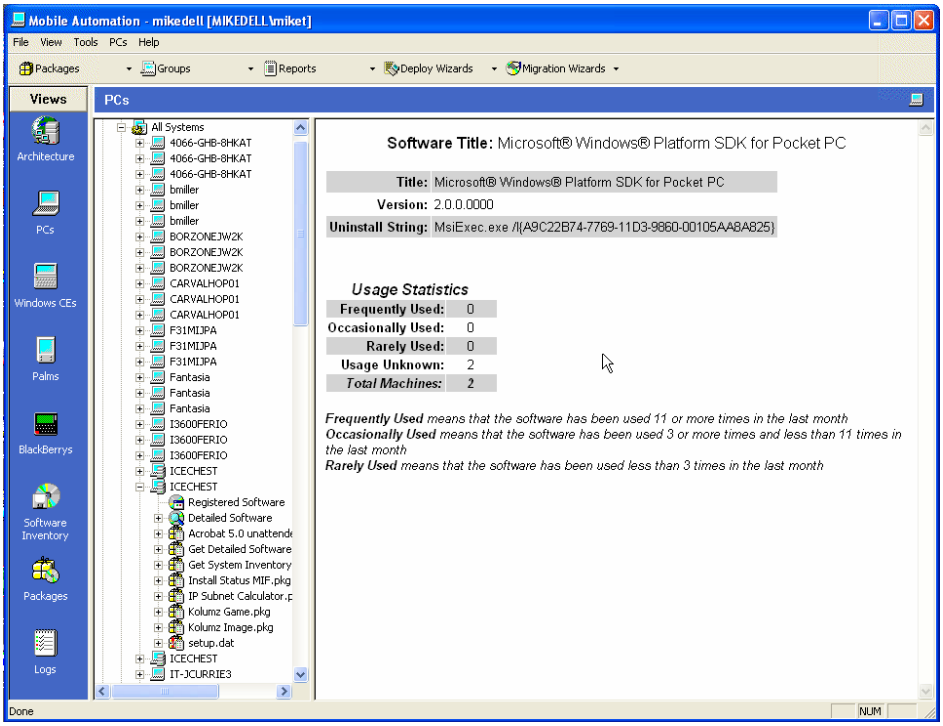

Figure 6.26: You can find out how popular the packages you deployed are by inspecting the usage statistics.

Summary

In this chapter, you learned about the free, the cheap, and the third-party ways to get your MSI files out the door and onto your users' plates. If you have a small environment, it's possible you can get away with Sneakernet or batch files. Midsized environments could make decent use of Win2K's Group Policy deployment features. However, the largest environments will likely need an industrial-strength solution to get those MSI files out the door and fully managed day to day.

Windows Installer technology has been out and about for several years now, and much has been written about it. Administrators leverage Windows Installer from a unique perspective, and they need resources that directly address their IT management concerns with sensitivity to their learning styles. To that end, we've tried to address Windows Installer technology management issues with practical tips and techniques you can apply immediately to your packaging efforts.

It is our hope that this book will be such a valuable resource that it will become very "virtually" worn and tattered as you reference it again and again.

Sincerely,

Jeremy Moskowitz and Darwin Sanoy