realtimepublishers.com™

The Definitive Guide™ To Windows Installer Technology for System Administrators

Darwin Sanoy and Jeremy Moskowitz

Introduction

By Sean Daily, Series Editor

Welcome to *The Definitive Guide to Windows Installer Technology for System Administrators*!

The book you are about to read represents an entirely new modality of book publishing and a major first in the publishing industry. The founding concept behind Realtimepublishers.com is the idea of providing readers with high-quality books about today's most critical IT topics—at no cost to the reader. Although this may sound like a somewhat impossible feat to achieve, it is made possible through the vision and generosity of corporate sponsors such as Wise Solutions, who agree to bear the book's production expenses and host the book on its Web site for the benefit of its Web site visitors.

It should be pointed out that the free nature of these books does not in any way diminish their quality. Without reservation, I can tell you that this book is the equivalent of any similar printed book you might find at your local bookstore (with the notable exception that it won't cost you $30 to $80). In addition to the free nature of the books, this publishing model provides other significant benefits. For example, the electronic nature of this eBook makes events such as chapter updates and additions, or the release of a new edition of the book possible to achieve in a far shorter timeframe than is possible with printed books. Because we publish our titles in "real-time"—that is, as chapters are written or revised by the author—you benefit from receiving the information immediately rather than having to wait months or years to receive a complete product.

Finally, I'd like to note that although it is true that the sponsor's Web site is the exclusive online location of the book, this book is by no means a paid advertisement. Realtimepublishers is an independent publishing company and maintains, by written agreement with the sponsor, 100% editorial control over the content of our titles. However, by hosting this information, Wise has set itself apart from its competitors by providing real value to its customers and transforming its site into a true technical resource library—not just a place to learn about its company and products. It is my opinion that this system of content delivery is not only of immeasurable value to readers, but represents the future of book publishing.

As series editor, it is my raison d'être to locate and work only with the industry's leading authors and editors, and publish books that help IT personnel, IT managers, and users to do their everyday jobs. To that end, I encourage and welcome your feedback on this or any other book in the Realtimepublishers.com series. If you would like to submit a comment, question, or suggestion, please do so by sending an email to feedback@realtimepublishers.com, leaving feedback on our Web site at www.realtimepublishers.com, or calling us at (707) 539-5280.

Thanks for reading, and enjoy!

Sean Daily

Series Editor

Copyright Statement

Chapter 1: Meet Windows Installer: Introduction, Features, and Benefits

by Jeremy Moskowitz

You’re an administrator. You’re the person who comes in early to reboot the servers on Sunday. You’re the person who stays late to ensure that the backup job has really kicked off successfully. You’re setting up servers and hauling around the occasional desktop, and you’re the lucky one who all-too-frequently has the honor of losing weekends due to a poorly planned “move, add, change.” We created this book for you.

If you’re familiar with the technology and history behind Windows Installer, you know that historically, discussions about this technology have had a bit of a developer slant. However, there is another side to these discussions, and the administrator’s side to the Windows Installer story is an exciting one that might get you a weekend or two back.

We’re going to try hard in this book to show you why Windows Installer is useful for you—the administrator. Throughout the book, we will be getting into some of the meaty internals of Windows Installer. If you have a little bit of a developer background, that’s great; but those who do not will be just as comfortable and find much useful information. With that in mind, let’s start our journey.

Defining the Need for Windows Installer

Before we dive into an introduction to Windows Installer and an exploration of how it works, let’s define the need for this technology. Why was it developed and what does it offer that will benefit you?

Saved Time and Effort Through Automated Installs

How are you installing your software today? If you’re like many administrators, you’re still tracking down the installation CD-ROM media (or you’ve made the installation available on a network source), running from machine to machine, and shooing the user aside for 30 to 60 minutes to install and test the application. This method works just fine for about 10 machines but quickly falls apart once the number of users and applications starts to increase. In the past, additional manpower has been brought in to shoulder the load—more people are hired to slog from computer to computer and load applications. Although hiring more people can ease the load, you and the other administrators must still answer, for each installation, the same 10 questions—such as the location of the program files, the default location of the data files, and which portions of the application should be loaded.

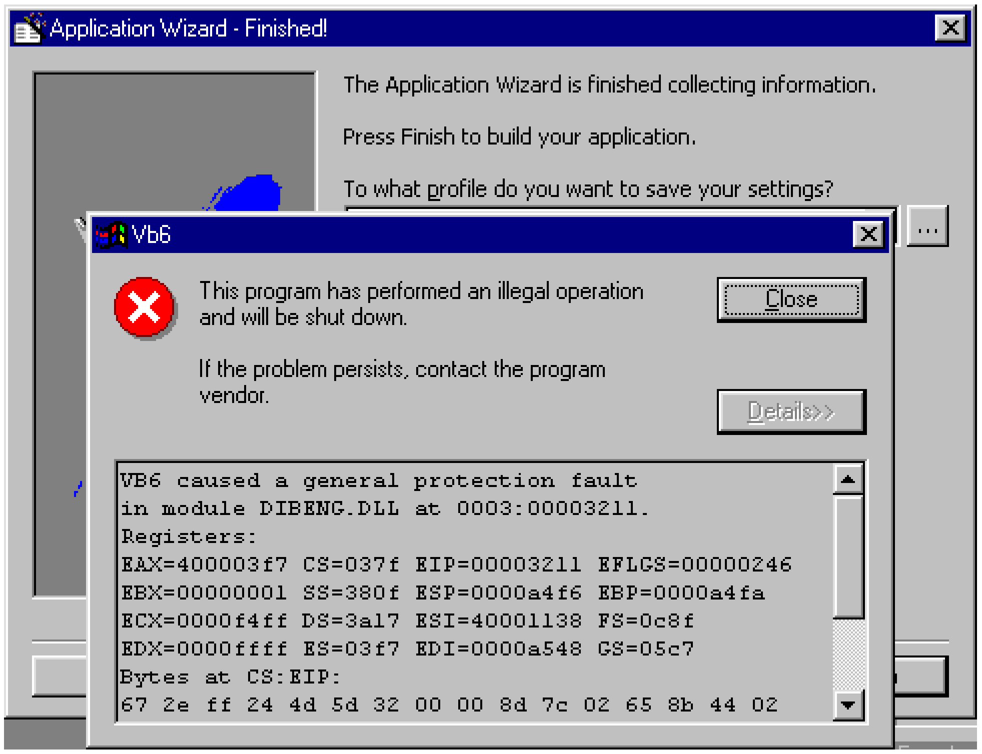
In addition to being a “poor administrative experience,” this tack is expensive. Imagine the figure you would come up with if you had to calculate all the time that you and your staff spent ensuring that each desktop had exactly the same hand-crafted software installation settings. Think of all the other stuff you could be doing if you and your team weren’t trotting to each desktop.

Enter Windows Installer technology. After we've covered all the pieces of this technology—from the Windows Installer basics to the MSI repackaging process to the actual deployment—you'll have gained the knowledge, skills, and tools necessary to save yourself and your staff tons of time that can then be devoted to more productive endeavors.

Many IT departments try some form of automated software distribution. Oftentimes, the software distribution job is seen as "secondary work" and isn't given a dedicated person. When the new implementation starts destabilizing as a result of a lack of dedicated personnel, IT managers become frustrated and determine that it's best to just throw in the towel rather than continue throwing money at a new project that already appears doomed. The result is that the project is completely scrapped and bad feelings are felt toward those who wanted to give it a shot in the first place. We'll be talking about how to prevent this chain of events through an exploration of distribution methods and helpful tips in Chapter 6.

Application and Operating System Stability

How often have you loaded a desktop or server system, then returned months later to load yet another application only to discover the bitter taste of application incompatibility? As Figure 1.1 shows, applications can be destabilized by other applications that load on top of required components.


Application Wizard - Finished!
The Application Wizard is finished collecting information.
Press Finish to build your application.
To what profile do you want to save your settings?
Vb6
Close
Details>>

This program has performed an illegal operation and will be shut down.

If the problem persists, contact the program vendor.

```
VB6 caused a general protection fault
in module DIBENG.DLL at 0003:00003211.
Registers:
EAX=400003f7 CS=037f EIP=00003211 EFLGS=00000246
EBX=00000001 SS=380f ESP=0000a4f6 EBP=0000a4fa
ECX=0000f4ff DS=3a17 ESI=40001138 FS=0c8f
EDX=0000ffff ES=03f7 EDI=0000a548 GS=05c7
Bytes at CS:EIP:
67 2e ff 24 4d 5d 32 00 00 8d 7c 02 65 8b 44 02
```

Figure 1.1: Evidence of application incompatibility.

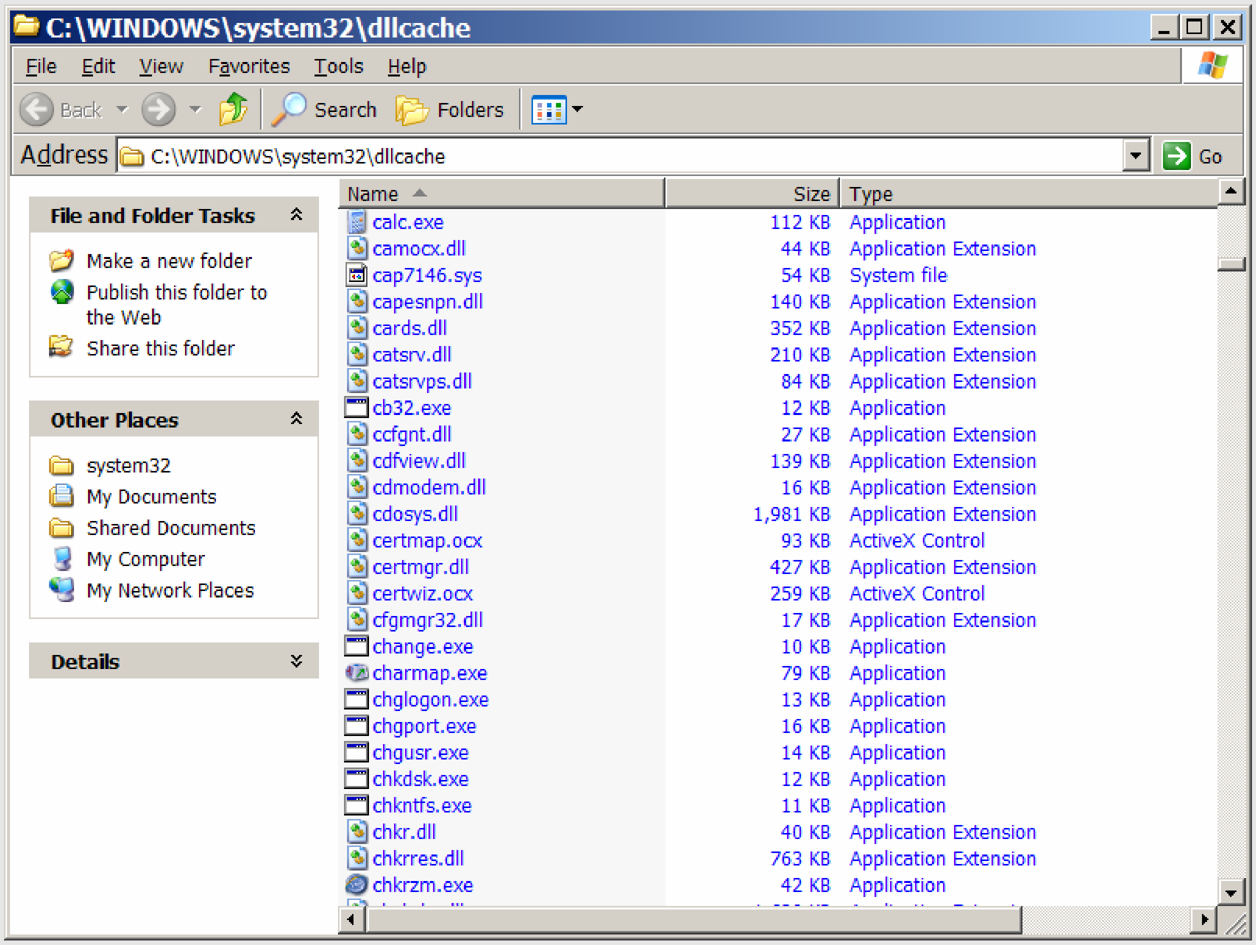
This incompatibility can occur because the technology used to package (or re-package) the application or manipulate its components for installation was simply not aware of other components already loaded on the system. Likewise, the operating system (OS) components can find themselves in harm's way.

Figures 1.1 appears here with permission from David Joffe and his Microsoft Crash Gallery Web site at http://www.scorpioncity.com/mscrash.shtml.

In Windows ME, Windows 2000 (Win2K), and Windows XP and later, new strides have been taken to ensure that, at least, the OS has an increased layer of protection. Read about it in the following sidebar “Windows File Protection.”

Windows File Protection

The Windows OS has had a reputation for crashing, but a new feature in Windows 98, Win2K, and Windows XP tries to put an end to this problem. The new feature is called Windows File Protection (WFP.) The goal of WFP is to ensure that if a critical system file, such as a .DLL or .EXE, is compromised, a secret copy can be brought back from a hidden “cache” on the disk. This copy ensures that misbehaving applications cannot take over the OS. Take a look into Windows’ secret directory called *dllcache* under the %systemroot%\system32 directory, usually C:\windows or C:\winnt (see Figure 1.2).



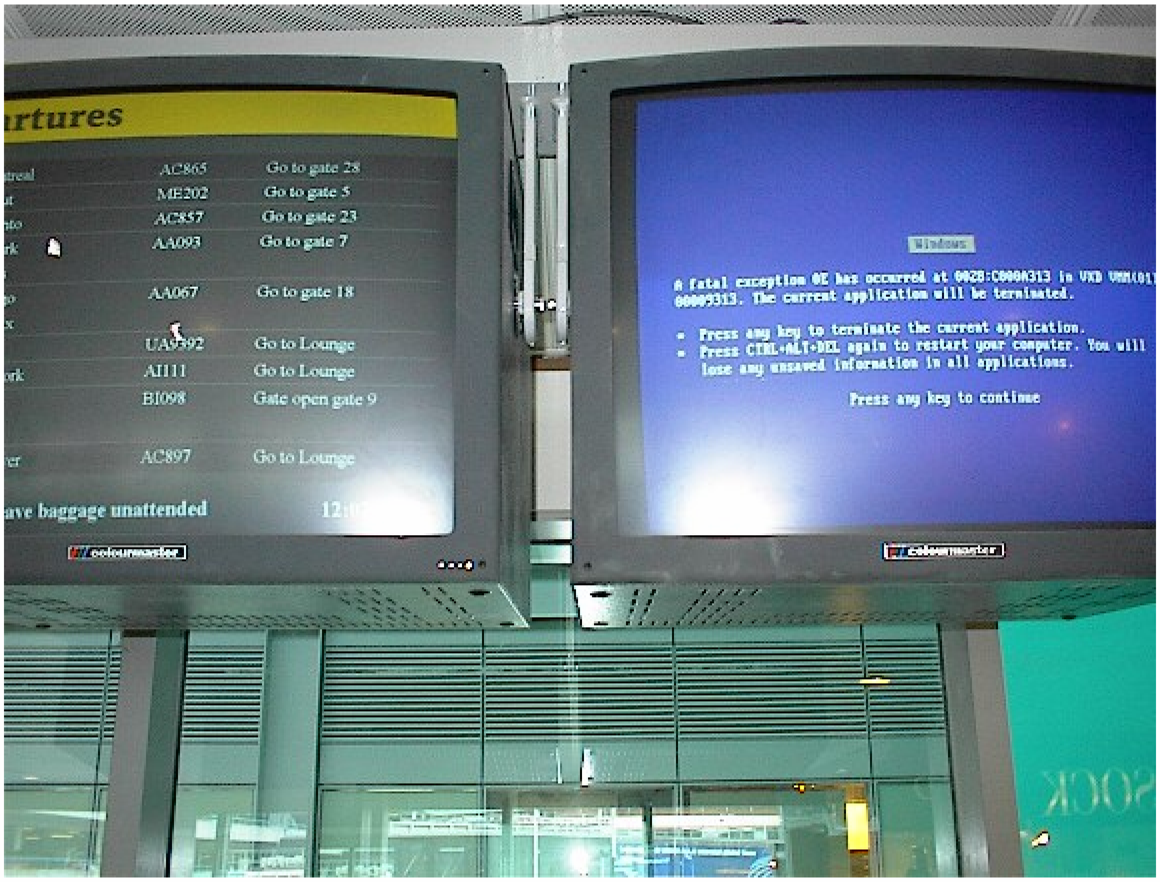
Name	Size	Type
calc.exe	112 KB	Application
camocx.dll	44 KB	Application Extension
cap7146.sys	54 KB	System file
capesnpn.dll	140 KB	Application Extension
cards.dll	352 KB	Application Extension
catsrv.dll	210 KB	Application Extension
catsrvps.dll	84 KB	Application Extension
cb32.exe	12 KB	Application
ccfgnt.dll	27 KB	Application Extension
cdfview.dll	139 KB	Application Extension
cdmodem.dll	16 KB	Application Extension
cdosys.dll	1,981 KB	Application Extension
certmap.ocx	93 KB	ActiveX Control
certmgr.dll	427 KB	Application Extension
certwiz.ocx	259 KB	ActiveX Control
cfgmgr32.dll	17 KB	Application Extension
change.exe	10 KB	Application
charmap.exe	79 KB	Application
chglogon.exe	13 KB	Application
chgport.exe	16 KB	Application
chgusr.exe	14 KB	Application
chkdsk.exe	12 KB	Application
chkntfs.exe	11 KB	Application
chkr.dll	40 KB	Application Extension
chkrres.dll	763 KB	Application Extension
chkrzm.exe	42 KB	Application

Figure 1.2: A list of some of the files in dllcache.

With WFP, no applications (except hotfixes or service packs) can overwrite these files—either here or in the files’ actual locations (normally Windows or system32). Go ahead and try to delete a file listed in dllcache, such as calc.exe, from its actual location. It will come right back from the cache!

Figure 1.3 shows a famous picture taken at London Heathrow Airport by Alan Cox. As this picture shows, application incompatibilities can lead to an unstable system with disastrous results. Although this picture illustrates a humorous example of the result of an application incompatibility, your goal should be to provide as stable of a system as possible.

✎ The picture in Figure 1.3 was reproduced with permission, courtesy of Alan Cox.



```
A fatal exception 0E has occurred at 0028:C000A313 in VXD VMM(01)
00009313. The current application will be terminated.

*  Press any key to terminate the current application.
*  Press CTRL+ALT+DEL again to restart your computer. You will
   lose any unsaved information in all applications.

                    Press any key to continue
```

Figure 1.3: London Heathrow Airport's system has a problem.

But how does Windows Installer relate to system stability? Before Windows Installer, packaged applications haven't had the ability to sense what was going on around them, which resulted in application and OS stability problems when a new application was installed on the system. Thus, Microsoft needed to step in and release a technology that was vendor-neutral and helped increase OS and application stability. That technology is called Windows Installer. In the following sections, we'll explore how Windows Installer improves stability as well as other benefits this technology provides.

The Benefits of Windows Installer and MSI

Windows Installer works as a result of the marriage between Windows Installer and a new package type called the Microsoft Installer (MSI) file type, which I'll discuss later in this chapter, and because of the behind-the-scenes action that takes place when Windows Installer encounters the new package type. Before we get too far along and talk about the technology behind Windows Installer and MSI and the stuff you can do with that technology, let me introduce its basic benefits. Table 1.1 provides just a few of the myriad benefits that Windows Installer and the corresponding MSI technology bring to the table.

✎ Some documentation refers to MSI packages as Medium Scale Integration files rather than Microsoft Installer files.

Windows Installer and MSI Benefit	**Description**
Application is installed via OS service	On Win2K and Windows XP and later, the application is installed in an administrative context. I'll explore this technology later in this chapter.
MSI provides a standard package format	A new format, the MSI package and its .MSI extension, is the new standard to interface with the Windows Installer technology.
Transactional install and rollback	Windows Installer packages can be made to either fully install the way the author intended, or if there is a failure during the install (for example, because you run out of disk space part-way through), the failed install can simply undo all the changes it has made up to that point in the installation to bring the system back to its previous state.
Self-healing (or self-repair) of corrupt or deleted critical files	As we'll explore in detail later, certain files can be *keyed* for detection of failure. If a critical file (a .DLL or .EXE file, for example) that is part of the distribution is corrupt or is deleted, the user can be prompted to repair the installation by presenting the original .MSI distribution. Additionally, if the installation media is available (for example, on a network share), the repair simply happens automatically.
Served installs	Because MSI files can be housed in a share point and delivered via a server, you can keep your installation files all in one place or move them around—closer to your users if necessary.
Install on demand	Windows Installer–deployed applications can be offered to clients at any time. Once offered, their installation can be triggered when a user clicks a corresponding registered extension. For instance, clicking a .DOC file prompts the installation of Word for Windows. Once chosen, the application is downloaded in a Just in Time (JIT—see the following entry) fashion.
JIT installation	After an application is offered to a user, it isn't *actually* installed. Instead, the application's icon appear, and when the user decides to run the application, it is installed from the media (or downloaded from the server) in a JIT fashion, and presents itself as ready to the user in a matter of moments.
Packages can utilize transform files	An application's package can be developed such that a *base* or *administrative* install is prepared for general distribution. A *transform* file can overlay the base, letting you customize specific installations. I'll discuss this benefit later in this chapter.
Packages can utilize patch files	After a package is on the machine, you might need to fix the source files if a bug is found or an update

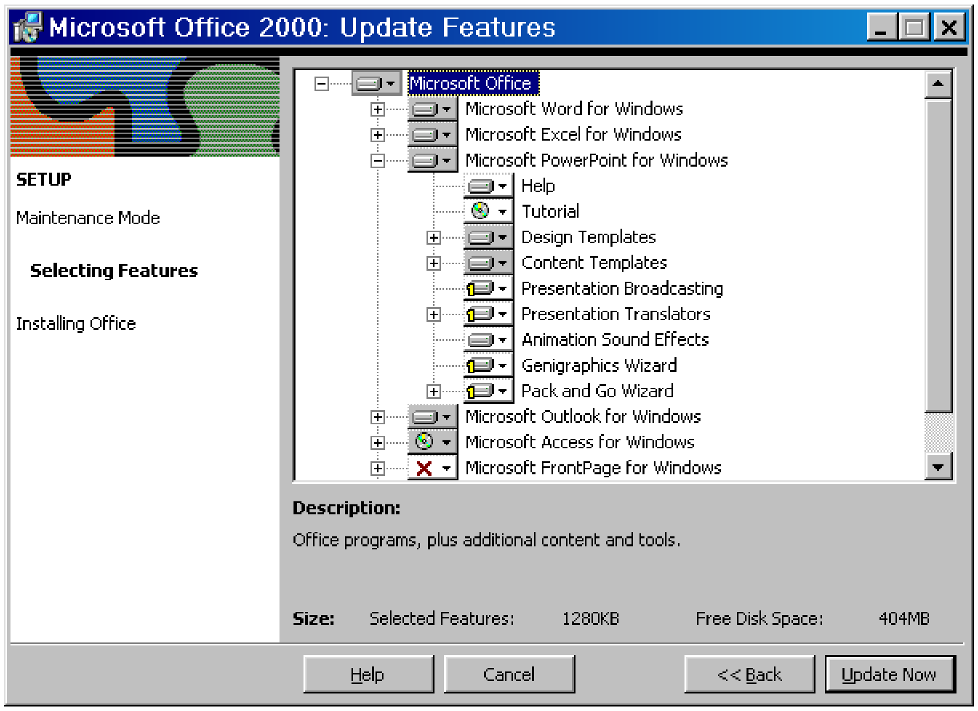
	is needed. Windows Installer defines a clear path to rectify these problems. I'll discuss this benefit later in this chapter.
State management	In the past, it's been difficult to know whether an application is installed on a machine. You would have to query for a .DLL with a specific version number or determine whether an .EXE file with a specific name was present. Windows Installer provides an application programming interface (API) that lets programmers and administrators see whether a specific application is installed on a machine.
Administrative privileges are not required for installations	Previously, you might have found that applications needed to be loaded through the local administrator account. Windows Installer eliminates this requirement.
Scriptable API	With a little elbow grease, you could whip together a VBScript to help you with your MSI file manipulations. The API to manipulate MSI files is so powerful that it can create packages, validate packages, update packages, trigger installs and uninstalls, examine the MSI repository data on computers, and perform some custom actions. If you have to repeat the same function, scripting is the way to go.
Packages can be managed using the MSIEXEC command-line tool	The command-line tool MSIEXEC is a very powerful tool that lets you manage your MSI applications. We'll be exploring some features of the MSIEXEC command-line tool a bit later in this chapter.

Table 1.1: Benefits of Windows Installer and MSI technology.

In the rest of the chapter and in the upcoming chapters, we'll explore these benefits in detail. A roadmap of the remaining chapters can be found at the end of this chapter.

Your First Windows Installer Encounter

The Windows Installer technology has been around for a while, so it's likely that you've already had some experience with the technology, perhaps without even being aware of it. For instance, you might have casually encountered it when installing the first Windows Installer–ready application, Office 2000, as Figure 1.4 shows.



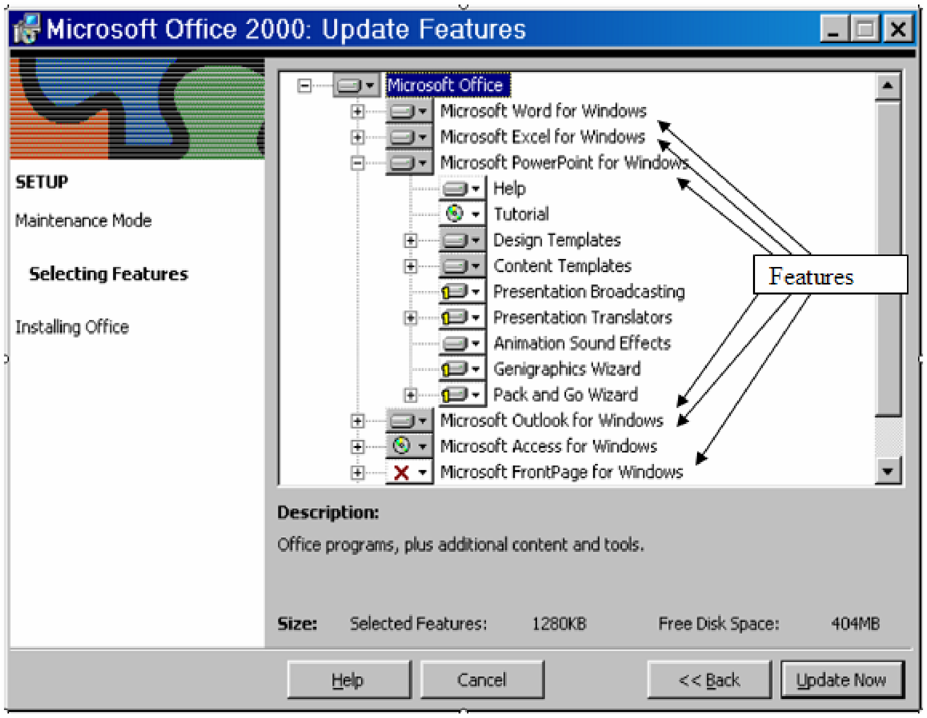
Microsoft Office
Microsoft Word for Windows
Microsoft Excel for Windows
Microsoft PowerPoint for Windows
Help
Tutorial
Design Templates
Content Templates
Presentation Broadcasting
Presentation Translators
Animation Sound Effects
Genigraphics Wizard
Pack and Go Wizard
Microsoft Outlook for Windows
Microsoft Access for Windows
Microsoft FrontPage for Windows

Description:
Office programs, plus additional content and tools.

Size: Selected Features: 1280KB Free Disk Space: 404MB

Figure 1.4: Office 2000 was the first application to ship as a Windows Installer–ready application.

Take a close look at the icons within the graphic. First, you'll notice that a product's installation is viewed in a hierarchical fashion. At the top of the hierarchy, we can see Windows Installer telling us which *product* we are installing. In this case, the product is Microsoft Office. Below the product, is a subset called *features*, as Figure 1.5 details.



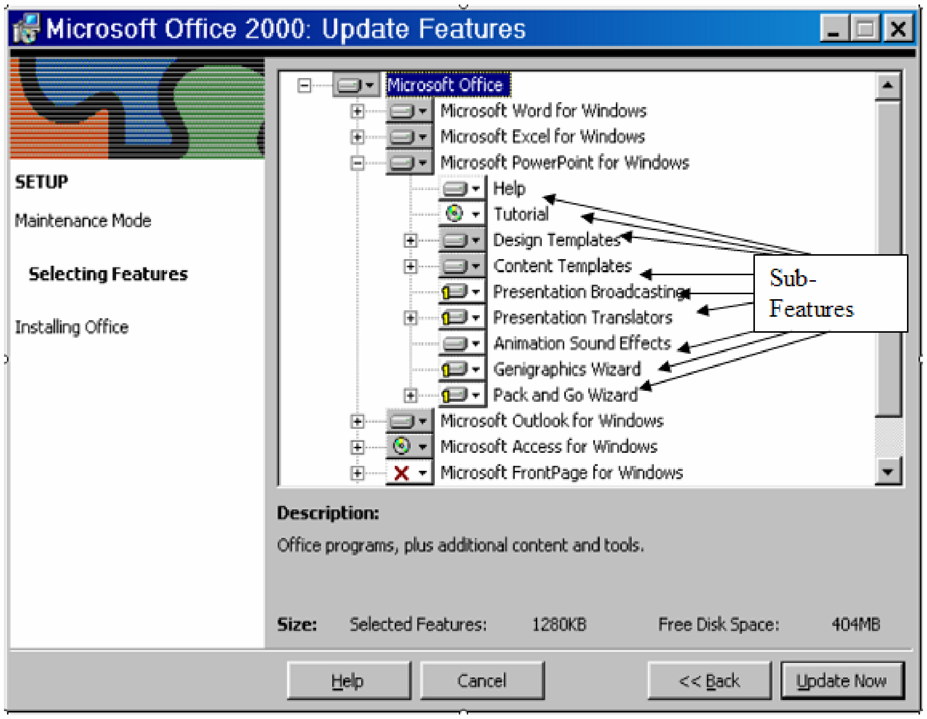
Microsoft Office
Microsoft Word for Windows
Microsoft Excel for Windows
Microsoft PowerPoint for Windows
Help
Tutorial
Design Templates
Content Templates
Presentation Broadcasting
Presentation Translators
Animation Sound Effects
Genigraphics Wizard
Pack and Go Wizard
Microsoft Outlook for Windows
Microsoft Access for Windows
Microsoft FrontPage for Windows

Description:
Office programs, plus additional content and tools.

Size: Selected Features: 1280KB Free Disk Space: 404MB

Figure 1.5: Highlighting a product's features.

The features make up the product, and there can be one or more features per product. Each feature in the hierarchy can have *sub-features*, as Figure 1.6 shows. In Office 2000, most of the components are located in the sub-features.



Description:

Office programs, plus additional content and tools.

Size: Selected Features: 1280KB Free Disk Space: 404MB

Figure 1.6: Features can have sub-features.

Occasionally sub-features are also called features, so these terms can be misleading. Common use defines features as the first level below product, and sub-features as the second level below product (as well as all additional levels below the feature level).

As you can see in the previous graphics, four possible installation states exist for any specific subsection of Office 2000:

- Solid gray means that the feature or sub-feature will be installed and available for use as soon as the install is complete.
- White with a yellow “1” icon means that the feature or sub-feature will be installed at first use. When the feature is installed, this sub-feature isn’t really installed. When the user tries to use this sub-feature for the first time, it is pulled from the installation media and installed in a JIT fashion. A benefit of this installation state is that it saves space—this feature will only be installed (and consume disk space) if users are going to use it.
- The CD-ROM icon means that the feature or sub-feature will not be installed directly on the hard drive. The feature or sub-feature will be accessible and able to run when the source CD-ROM is present in the drive or through the network if a network connection is available. This installation state is handy, for example, for features that consume a lot of space, such as multimedia files and reference files, that will be used occasionally, if ever. In the previous figures, for example, Microsoft Access has been set to run solely from the installation media—either a CD-ROM drive, network share, or other source.

- The red X icon means that the feature or sub-feature will not be installed and won't be accessible during normal use of the product. If this feature is desired, the installation setup program needs to be re-run and this feature's state needs to be changed to one of the other three states.

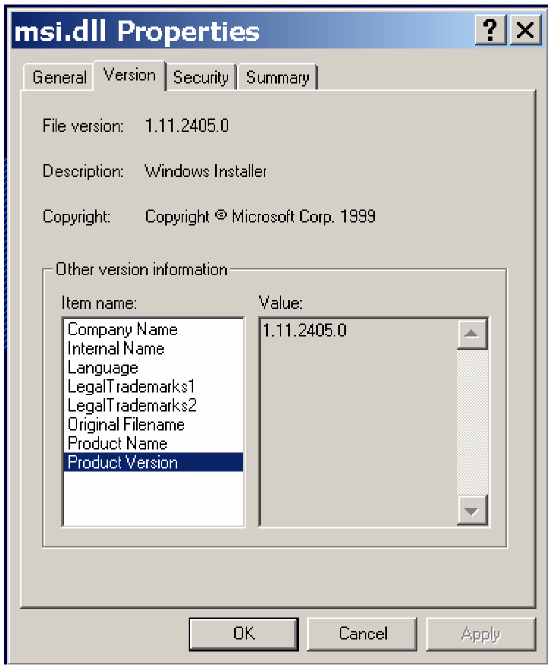
Windows Installer Version Numbers

Like most software products, Windows Installer has versions associated with it. There are major revisions (such as 1.0) and minor revisions (such as 1.1). Windows Installer is unique in that it is versioned for each platform. That is, Windows NT, Windows 9x, and so on each has its own Windows Installer version number. However, automatic updates can occur without a user's knowledge, so users in your environment could possibly have different Windows Installer versions.

What Is Your Windows Installer Version Number?

To find out which version of Windows Installer is on your machine, you can simply use Windows Explorer to navigate to the %systemroot% directory (usually \Windows or \Winnt), enter the system32 directory, right-click MSI.DLL, and select Properties. Doing so reveals the window that Figure 1.7 shows.

☞ You can also run the command-line tool MSIEXEC from the Start menu Run text box to discover the version number.



File version: 1.11.2405.0

Description: Windows Installer

Copyright: Copyright © Microsoft Corp. 1999

Company Name
Internal Name
Language
LegalTrademarks1
LegalTrademarks2
Original Filename
Product Name
Product Version

1.11.2405.0

Figure 1.7: Inspecting your Windows Installer version number.

The file version provided on the Version tab of the properties window should match the product version (in Figure 1.7, they both have a value of 1.11.2405.0). The 2405.0 entry is simply the specific build number. The Windows Installer filer version is 1.11. So why would your machine have a different file version?

The Internals of Version Numbers

As one might expect, Windows Installer started with version 1.0. As I previously mentioned, Office 2000 was the first Windows Installer–ready application. Being the first presented a problem: How would the system perform the Office 2000 installation if Windows didn't yet have the Windows Installer *bits* (the executables necessary for Windows Installer to install the program)?

The Office 2000 development team came up with a brilliant little plan: Before actually loading Office, load a little piece of code that loads the bits required to perform a Windows Installer–type install, then perform the rest of the Office 2000 install as a Windows Installer install. (Current applications will attempt to install the latest version of the Windows Installer using a similar method.) So, the bits for Windows Installer 1.0 are included on the Office 2000 CD-ROM and are automatically installed when the setup routine is run.

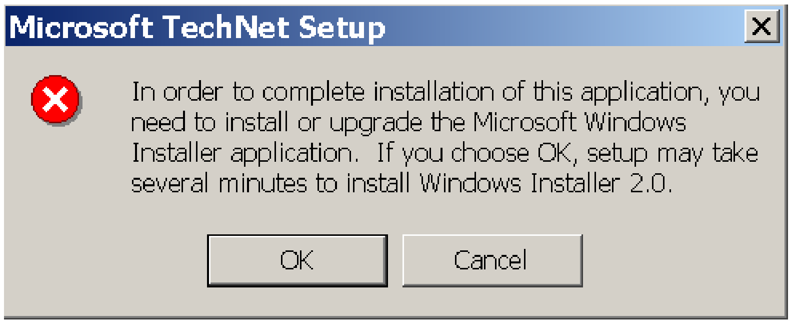
Let's take a look at the Windows Installer versions from then until now:

- Version 1.10 was the first version included in Win2K directly (build 1.10.1029.0). For other platforms, there was a version made available for download from the Microsoft Web site (build 1.10.1029.1).
- Version 1.11 appeared in Service Pack 1 (SP1) for Win2K (build 1.11.1314.0). Another, later build of Version 1.11 appeared in SP2 (build 1.11.2405.0).
- Version 1.20 appeared in Windows ME (build 1.20.1410.0) and was available for download from the Microsoft Web site (build 1.20.1827.1).
- Version 2.0 and later will be discussed in the next section.

Note the pattern of version numbers. Specifically, versions that end in .0 ship with and are built into the OS. Those that end in .1 are downloads.

Windows Installer Version 2.0

Windows Installer 2.0 is the latest major release for Windows Installer. You might casually encounter the upgrade in a manner similar to the original Office 2000 installation. That is, you might simply double-click to install your latest application acquisition, and you'll be presented with the pronouncement that Figure 1.8 shows.



Microsoft TechNet Setup

In order to complete installation of this application, you need to install or upgrade the Microsoft Windows Installer application. If you choose OK, setup may take several minutes to install Windows Installer 2.0.

Figure 1.8: The installation of Windows Installer 2.0.

You must have Windows NT SP6 installed before you are able to install Windows Installer 2.0.

Windows Installer 2.0 provides some new features, which the following list describes. These features make Windows Installer a much more efficient engine.

- Hash-based calculations—Windows Installer 2.0 is a lot smarter than previous versions about recognizing when files need to be replaced—for either a repair, patch, or upgrade to an existing package. Should an application need a repair, upgrade, or patch, Windows Installer 2.0 performs a much faster hash-based calculation (rather than comparing each installed file to the original source installation, which is a really slow process) to determine whether a file needs to be replaced. The added bonus is that you might not need the original source media available unless there is a problem with a file that specifically needs to be replaced. Thus, you can more quickly perform a repair, patch, or an upgrade.
- Delayed reboot—When a new version of the Windows Installer bits becomes available, you're prompted to install the new version (as Figure 1.8 shows). Previous versions of Windows Installer required that you install the new bits and reboot the system before you could progress with the installation of your application. Windows Installer 2.0 lets you delay the reboot (which completes the Windows Installer upgrade) until all your MSI packages are fully installed.

An upgrade from Windows Installer 1.x to 2.0 requires a reboot, but you can delay the reboot until you're ready. In addition, you must be logged on as a local Administrator to perform the upgrade.

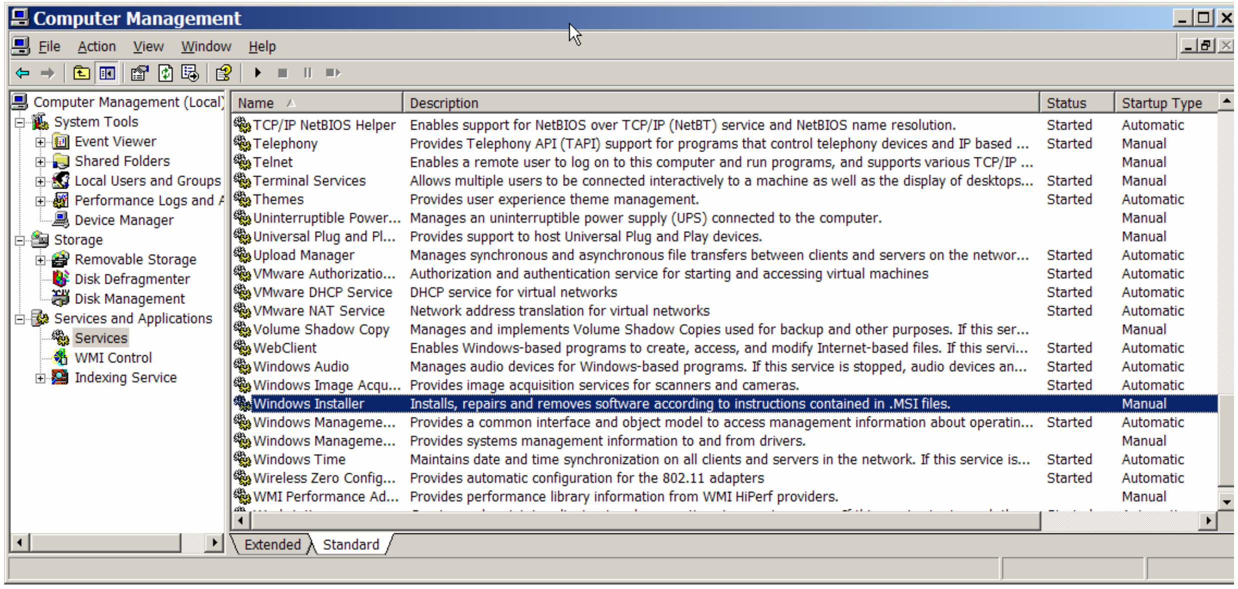
- Improved logging—Windows Installer 2.0 provides better logging in the event log and when files are installed. Each error receives an ID (the error codes for previous versions of Windows Installer all fell within two or three non-unique event IDs), which greatly improves how you can search for and filter Windows Installer events.
- Increased security and multiple user isolation—Previously, an MSI application was installed for one user; however, another user might be able to use that application. Windows Installer 2.0 makes a great effort to ensure that each install is a personal and customized experience so that when Joe sits at Sally's machine, Joe cannot run an application that Sally installed just for Sally.
- Digital signature support—Files can now be digitally signed within an MSI file (as well as .MSP and .MST files) to ensure that the file came from a trusted source.

- 64-bit service with 64-bit Windows—Windows .NET server and Windows XP will each have 64-bit versions that will run on the Itanium II processor. These OSs will be able to take advantage of the new 64-bit Windows Installer services.

> ☞ To download Windows Installer 2.0, go to www.microsoft.com/msdownload/platformsdk/sdkupdate/psdkredist.htm.

Windows' Relationship to Windows Installer

Let's take a closer look at how Windows Installer and Windows are related. Recall that the Windows Installer bits are built into Win2K and Windows XP; moreover, they run as a service. To see the service, simply right-click My Computer, select Manage, and click the Services entry under Services and Applications. The Windows Installer service is highlighted in Figure 1.9.



Name	Description	Status	Startup Type
TCP/IP NetBIOS Helper	Enables support for NetBIOS over TCP/IP (NetBT) service and NetBIOS name resolution.	Started	Automatic
Telephony	Provides Telephony API (TAPI) support for programs that control telephony devices and IP based ...	Started	Manual
Telnet	Enables a remote user to log on to this computer and run programs, and supports various TCP/IP ...		Manual
Terminal Services	Allows multiple users to be connected interactively to a machine as well as the display of desktops...	Started	Manual
Themes	Provides user experience theme management.	Started	Automatic
Uninterruptible Power...	Manages an uninterruptible power supply (UPS) connected to the computer.		Manual
Universal Plug and Pl...	Provides support to host Universal Plug and Play devices.		Manual
Upload Manager	Manages synchronous and asynchronous file transfers between clients and servers on the networ...	Started	Automatic
VMware Authorizatio...	Authorization and authentication service for starting and accessing virtual machines	Started	Automatic
VMware DHCP Service	DHCP service for virtual networks	Started	Automatic
VMware NAT Service	Network address translation for virtual networks	Started	Automatic
Volume Shadow Copy	Manages and implements Volume Shadow Copies used for backup and other purposes. If this ser...		Manual
WebClient	Enables Windows-based programs to create, access, and modify Internet-based files. If this servi...	Started	Automatic
Windows Audio	Manages audio devices for Windows-based programs. If this service is stopped, audio devices an...	Started	Automatic
Windows Image Acqu...	Provides image acquisition services for scanners and cameras.	Started	Automatic
Windows Installer	Installs, repairs and removes software according to instructions contained in .MSI files.		Manual
Windows Manageme...	Provides a common interface and object model to access management information about operatin...	Started	Automatic
Windows Manageme...	Provides systems management information to and from drivers.		Manual
Windows Time	Maintains date and time synchronization on all clients and servers in the network. If this service is...	Started	Automatic
Wireless Zero Config...	Provides automatic configuration for the 802.11 adapters	Started	Automatic
WMI Performance Ad...	Provides performance library information from WMI HiPerf providers.		Manual

Figure 1.9: Locating the Windows Installer service.

Windows Installer is installed as a service, so it's capable of intercepting shortcuts and file extensions that link to applications and prompt their install from the source media. For instance, you might have Office 2000 loaded, but set to load PowerPoint upon first use (that's the icon with the little '1' we saw in Figure 1.4).

The version that runs on Win2K and Windows XP and later is also capable of receiving and executing instructions when running in an Active Directory (AD) environment. Specifically, Win2K and Windows XP and later can take instructions to load MSI applications via Group Policy. This functionality allows for applications deployed via AD to run in an administrative context—allowing applications to be loaded when administrators want systems protected, and preventing regular users from installing applications they shouldn't. We'll be discussing this method of installation in Chapter 6.

Windows Installer on Downlevel Clients

Services are incapable of running on Windows 9x machines, so you won't be able to perform the aforementioned procedure. You could see the Windows Installer service on NT clients; however, downlevel clients can't receive Group Policy, so they are incapable of receiving instructions from AD telling them to install or upgrade an MSI file.

Although downlevel clients can't receive Group Policy, and hence, are incapable of receiving MSI installation instructions, the clients are still capable of using nearly all the remaining Windows Installer features. For example, recall that files can be keyed for proper operation, and if a required file should get damaged, the application can simply prompt for the original installation source. This capability is present with downlevel clients and is a major win for system stability on older clients.

☞ Microsoft has recently provided a very good FAQ about Windows Installer at http://www.microsoft.com/windows2000/community/centers/management/msi_faq.asp.

MSI File Foundations

In this section, we'll explore how an administrator might typically encounter MSI files. Not every MSI application will be interfaced in the same way, but this section will provide you with general exposure to the process.

Setup or MSI?

As we've discussed, you might already be familiar with MSI files because a vendor has delivered some of your applications in the MSI format. Interestingly, though, some vendors, including Microsoft, still include a legacy setup.exe program. This setup.exe program is often simply checking the system for the presence of MSI bits, then calling the corresponding .MSI file to launch. Oftentimes setup.exe is provided as a backward-compatibility measure; that is, those who don't know to click the .MSI file will simply use the setup.exe program, which then calls the .MSI file.

For example, as Figure 1.10 illustrates, the Win2K Support Tools installation comes with both a setup.exe program and a file named 2000RKST.MSI. Clicking either SETUP or 2000RKST will ultimately launch the MSI file, and start the installation.

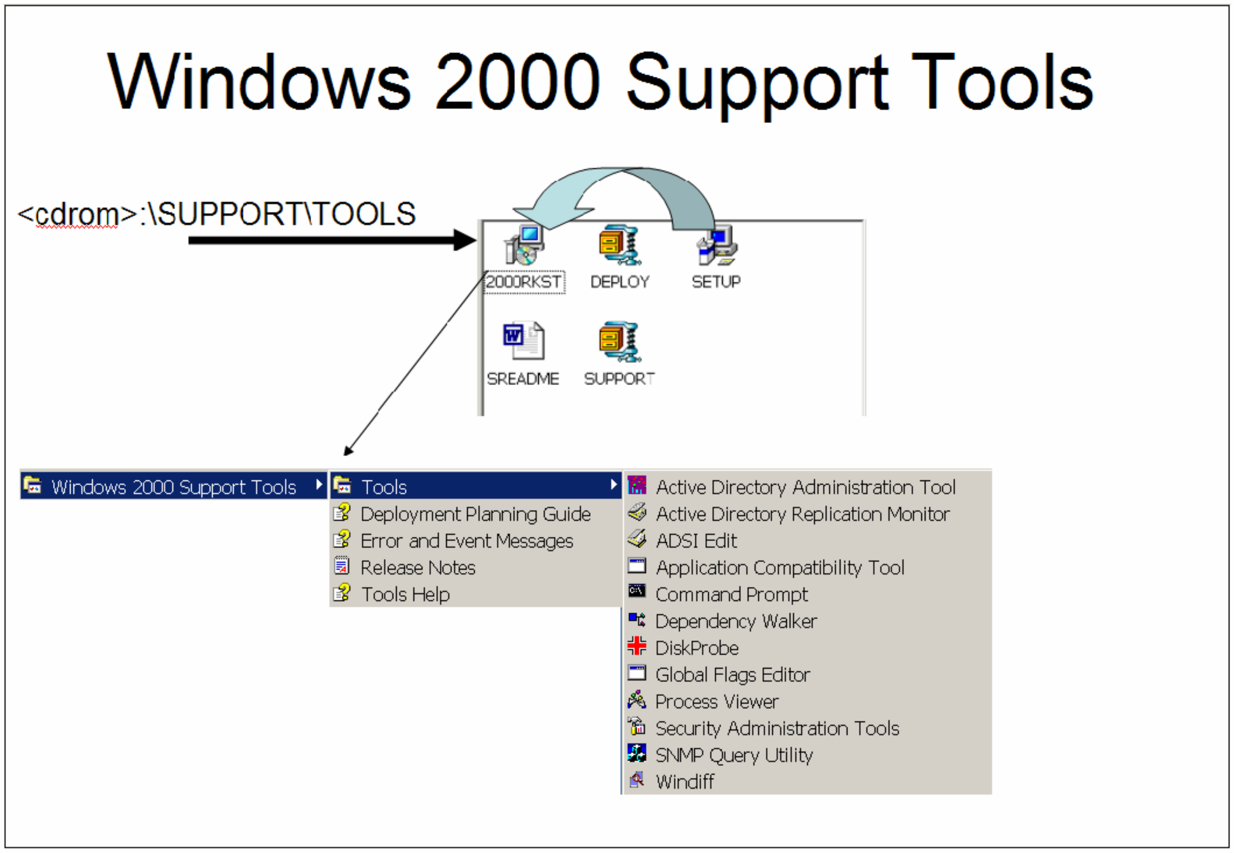


Figure 1.10: The Win2K Support Tools can be launched in either fashion.

Base Installations, Transforms, and Patches

We can build on this foundation of knowledge about MSI and Windows Installer so that you understand how to manage applications using all the new MSI functions. When you receive an application package from a vendor, what is the actual process you're supposed to follow—after you get the software, what do you do?

Base Installations

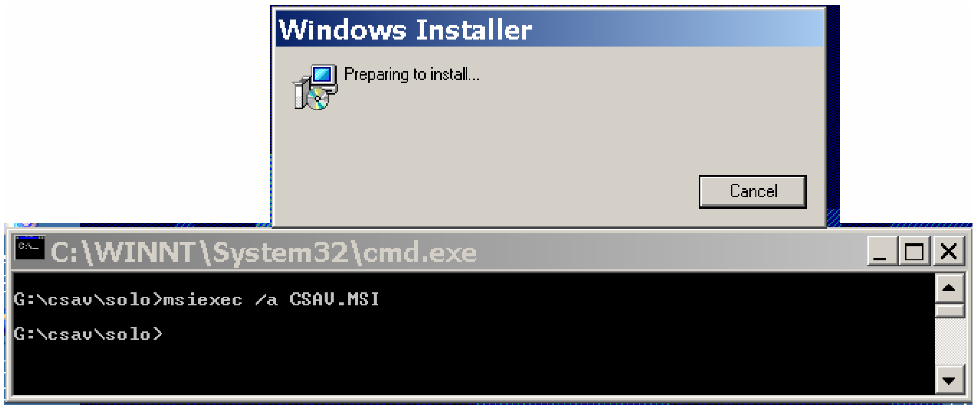
First, you'll need to understand that the application will come from a vendor and ship as a set of *base* installs. The base installs are the bits that are downloaded from the Web or the bits on the CD-ROM that constitute the original distribution of the software. (In just a moment we'll compare base installations to transforms and patches.)

To make use of the base installation, you might need to *prepare* the installation, creating an *administrative installation*. In an administrative installation, the base installation's files are basically yanked from the packed source (in this case, the MSI file) and placed in the format that the application needs in another, alternative directory structure that is suitable for file sharing. (This technique is similar to the way that Windows 95 and Office 97 rollouts were prepared.) Not all .MSI packages must be prepared as an administrative installation; check your vendor's documentation to be sure.

When an administrative installation is required to manipulate a vendor-supplied package, you'll usually use the built-in Win2K MSIEXEC command to create the administrative installation point. Oftentimes, an administrative installation is performed by running the MSIEXEC utility with the /a switch, as follows:

```
msiexec /a {packagename}.MSI
```

When you do, the familiar Windows Installer window will pop up, showing you that the command is working, as Figure 1.11 shows.



```
G:\csav\solo>msiexec /a CSAV.MSI

G:\csav\solo>
```

Figure 1.11: Run MSIEXEC to prepare the installation.

After this command has executed, you'll typically get a wizard that asks where you want to place the administrative installation. Simply place the files into a shared folder, and you're most of the way done. Your users could then connect to the administrative point by mapping a drive or via logon script or one of the many alternative methods, and run the installation.

The problem with running the installation in this fashion is that it's simply not customized or tailored for the many users who might want to install and use the software. Indeed, all users who connect back to either the base bits or the administrative installation are usually presented with the default settings as the MSI package is coded. If an end user isn't savvy, he or she could be faced with many potential installation choices, as we saw in Figures 1.5, 1.6, and 1.7. That is where transforms come in.

Transforms

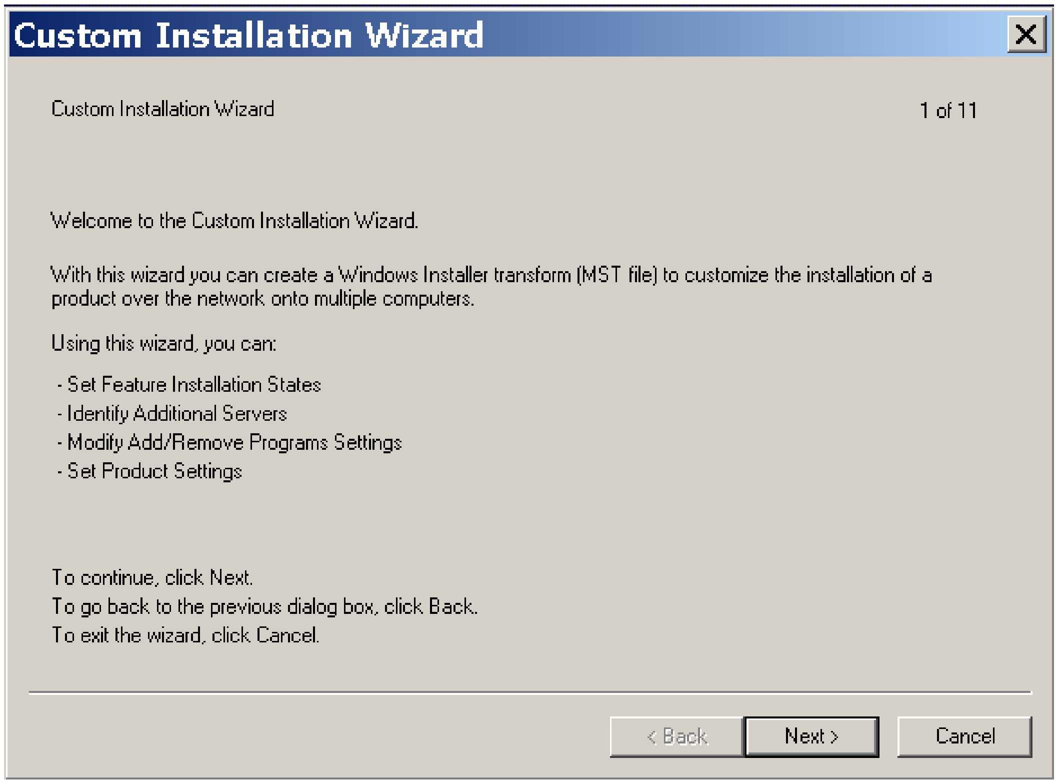
So, although both the base and administrative installations are useful, the installations that they create can simply be too broad. If a client were to double-click the setup's .MSI in the base installation, the client would be prompted to install every default option described in the package, which might not be the ideal installation for the client. You might want to specify that certain users get some options in a package and ensure that others do not, as well as specify the default installation directory, the default *Save as* location, and so on.

To do so, the MSI format allows you to create *transforms*. A transform filters and shapes what your MSI file will look like for a specific user base. A transform file has an .MST extension.

Transforms can be handy in a variety of situations. For instance, you might choose to have a base MSI installation package that loads the DogFoodMaker 5.0 application. However, you might want a customized installation for the sales group that places the default installation point on the D drive, a customized installation for the marketing group that places the default installation point on the E drive, and a customized installation for the nurses that has only one feature.

Vendor-Supplied Transform-Generation Tools

Transforms can be created for a specific MSI file in multiple ways. One way is that a vendor supplies a standalone tool that examines their product's setup .MSI file, allows for user input, then spits out a customized .MST file. Figure 1.12 shows an example of a custom–installation–creation wizard.



Welcome to the Custom Installation Wizard.

With this wizard you can create a Windows Installer transform (MST file) to customize the installation of a product over the network onto multiple computers.

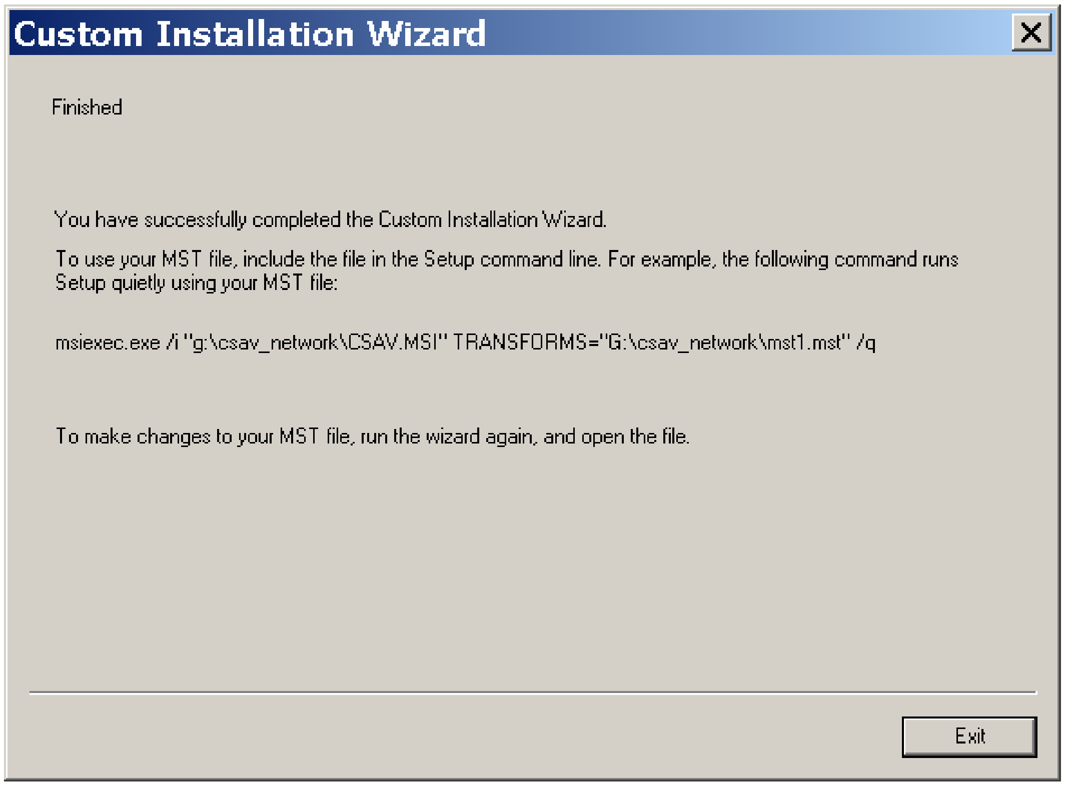
Using this wizard, you can:

- Set Feature Installation States
- Identify Additional Servers
- Modify Add/Remove Programs Settings
- Set Product Settings

To continue, click Next.
To go back to the previous dialog box, click Back.
To exit the wizard, click Cancel.

Figure 1.12: A vendor's custom–installation–creation wizard for creating transforms.

Such a tool uses the vendor's MSI file as an initial starting point, then walks you through which settings you can change to customize an installation. After you've chosen your customizations, the tool spits out both an MST file and instructions for its use (as Figure 1.13 shows).



Finished

You have successfully completed the Custom Installation Wizard.

To use your MST file, include the file in the Setup command line. For example, the following command runs Setup quietly using your MST file:

```
msiexec.exe /i "g:\csav_network\CSAV.MSI" TRANSFORMS="G:\csav_network\mst1.mst" /q
```

To make changes to your MST file, run the wizard again, and open the file.

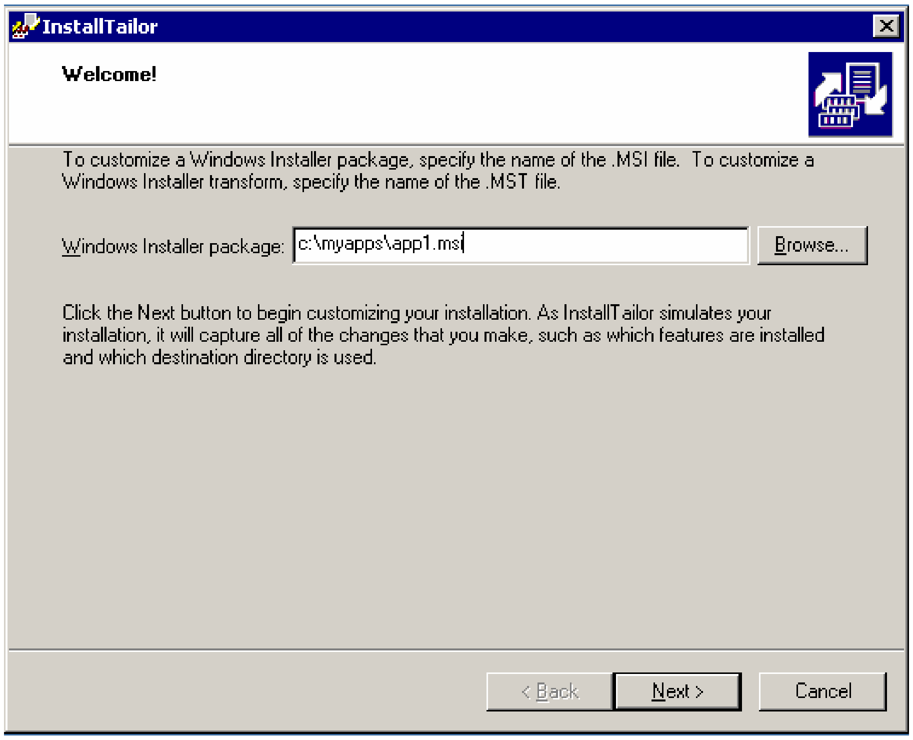
Figure 1.13: The final window of a custom–installation–creation wizard.

Don't run the package just yet. In the following sections, I'll explore third-party transform-generation tools, then discuss what to do with the customized installation file.

> Figure 1.13 shows output that uses a specific drive letter—the G drive. For you to use the command that Figure 1.13 shows, G would actually need to be mapped to the administrative installation. If you don't want to worry about drive mappings, consider substituting using universal naming convention (UNC) pathnames.

Third-Party Transform-Generation Tools

Although many vendors are starting to produce their software as MSI files, a lot of them don't yet offer a standalone tool to generate MST files. If you have applications that fall into this category, you'll need to find a third-party tool that can crack open an existing MSI file and help you generate an associated MST file, such as the InstallTailor tool included in Wise Solutions' Wise Package Studio. As Figure 1.14 shows, you simply point InstallTailor to an existing MSI file (either one that you create or a vendor-supplied MSI file).



InstallTailor

Welcome!

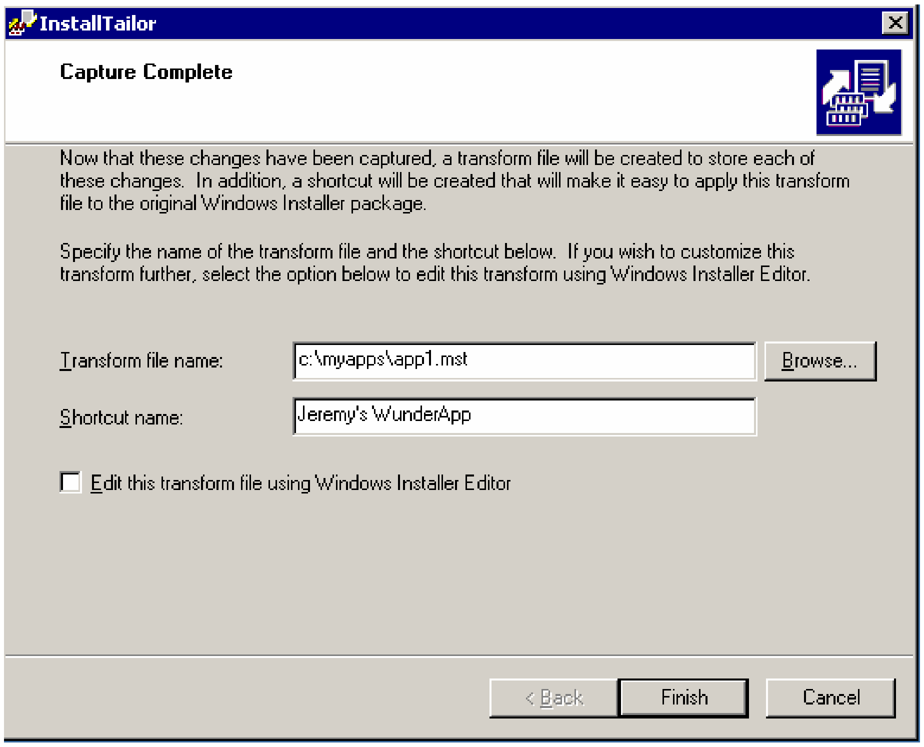
To customize a Windows Installer package, specify the name of the .MSI file. To customize a Windows Installer transform, specify the name of the .MST file.

Windows Installer package: c:\myapps\app1.msi

Click the Next button to begin customizing your installation. As InstallTailor simulates your installation, it will capture all of the changes that you make, such as which features are installed and which destination directory is used.

Figure 1.14: Opening an existing MSI file to create an MST file.

You'll then be asked to "simulate" the MSI file's installation. Make your installation choices as if you were actually installing the MSI package. When you're finished, the tool will create the MST file, as Figure 1.15 shows.



InstallTailor

Capture Complete

Now that these changes have been captured, a transform file will be created to store each of these changes. In addition, a shortcut will be created that will make it easy to apply this transform file to the original Windows Installer package.

Specify the name of the transform file and the shortcut below. If you wish to customize this transform further, select the option below to edit this transform using Windows Installer Editor.

Transform file name: c:\myapps\app1.mst

Shortcut name: Jeremy's WunderApp

☐ Edit this transform file using Windows Installer Editor

Figure 1.15: Creating an MST file for your applications.

Executing MSIs with Transforms

After you've created a transform by using either a vendor-supplied standalone application or third-party transform-generation source, you're ready to run. If you want a client to get the MSI package with your MST customizations, simply execute MSIEXEC in the following fashion:

```
msiexec /I {packagename}.MSI TRANSFORMS={transform.mst}
```

This command line runs the MSI package and applies the changes that you included in the MST file.

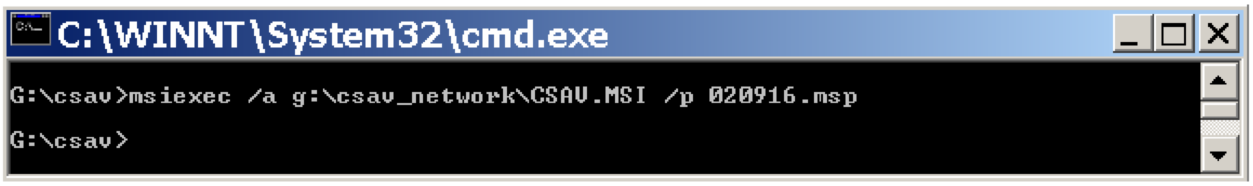
Patches

Getting the base bits online via the administrative installation is important. It's also important to make use of transform files, as they let you specify which bits that you want to make their way onto the client desktop. But what if the original bits need to be fixed in some way? That's where the MSI file definition has room to be adjusted.

If an update or a fix is available for a current MSI installation, you can *patch* the original installation with the latest bits to ensure that the most modern bits are being used. These files come in the form of .MSP files, signifying that they are patch files.

After you download the MSP file, you'll need to run the MSIEXEC command to update the MSI with the latest patch file. To do so, use the following command syntax, as Figure 1.16 shows:

```
msiexec /a {packagename}.MSI /p {patchfile}.MSP
```



```
G:\csav>msiexec /a g:\csav_network\CSAV.MSI /p 020916.msp

G:\csav>
```

Figure 1.16: Execute the patch against the MSI file.

Depending on the application, you might be prompted for the path of the administrative installation. After the file installs, all users who run the MSI will have the latest updates.

The good news about using MSP files for patching is that you're actually modifying the source MSI file. Thus, all new installations that spring forth from this MSI will be up to date with the latest patches. The bad news about using MSP files for patching is that you're stranding the installations that used the previous (un-patched) version of the MSI. If a client that used this un-patched MSI has a problem and must pull down a file or two from the source, it won't be able to because the MSI file is now different.

Therefore, you have to instruct all systems that used the previous version of the MSI that a *new* MSI file is available and to reinstall from that source. A typical command line to do so might look like the following example:

```
msiexec /fvomus package.msi REINSTALL=ALL
```

This command instructs the application to perform a full reinstallation. After running this command, if a re-installation is required or even just one file is damaged, the target system matches the administrative installation.

☞ The /f option in MSIEXEC simply specifies that you want to do a repair. The v, o, m, u, and s options essentially overwrite all previous files and registry entries if they are encountered.

Roadmap for the Rest of the Text

At this point, you should have a good handle on what the Windows Installer technology does and why the MSI file type is necessary. With this knowledge, we can start committing to the idea of consistently using the MSI file type so that every application we deploy is delivered as an MSI package.

In future chapters, we'll show you how to start creating your own packages and make use of all that the MSI technology has to offer. We'll do so via both free and third-party tools that help in MSI file generation, as we'll show you in Chapter 2.

In this chapter, we touched on the relationship between Windows Installer and MSI files. But to really make the best use of the technology, we'll need to dig deeper. In Chapter 3, we'll tackle the package structure in greater detail, talk about how to further manipulate packages using transforms, and learn how to secure our packages. Then, we'll move on in Chapter 4 to the secret world of the MSI software development kit (SDK).

Many companies are moving toward Win2K and AD, but many are staying put with either Novell or NT for the foreseeable future. If you're not planning on going to AD anytime soon or you simply don't want to make use of the built-in Win2K software deployment features, Chapter 5 will be a must-read for you.

Finally, we'll wrap things up in Chapter 6 by exploring various ways to distribute the packages you've learned to make while repackaging and authoring. Have a small user base? Have a giant user base? Have Win2K and AD? Chapter 6 will highlight various methods for you to get the right package to the right people.