realtimepublishers.com™

The Definitive Guide™ To

Windows Server 2003 Terminal Services

triCerat Software®

Gresyon Mitchem

Copyright Statement

Chapter 5: Application Installation and Compatibility

We've explored how to enable the Terminal Services role, create and manage a Session Directory farm, and integrate terminal servers into your AD environment. All of these factors are vital to a successful Terminal Services infrastructure, but it is the user applications that can make or break your terminal server deployment.

As Terminal Services evolved, user applications have also evolved. Today, the majority of applications will install natively on Terminal Services and function in a multi-user environment without modification. However, there will always be a mission-critical program that is either legacy or from a vendor that does not follow the Microsoft Windows Logo specifications, so it is important to become familiar with the various features available in Windows Server 2003 to help address application compatibility.

This chapter will focus on installing, deploying, and managing applications in a Terminal Services environment. You'll learn how to tune applications for simultaneous users, write an application compatibility script, and use Terminal Services registry flags to handle legacy applications. I'll discuss the various administrative modes that you'll use and how to test an application for Terminal Services compatibility.

Application Compatibility Mechanisms

Before we dive into the application-installation process, let's explore the mechanisms available to assist in making applications that weren't designed for Terminal Services run on a terminal server. These mechanisms include Terminal Services logon scripts, application compatibility scripts, install and execute modes, registry mapping, and INI file mapping.

> If an application carries the Designed for Microsoft Windows XP logo, it is usually compatible with Terminal Services. Microsoft's certification program assures that the application follows certain guidelines in its installation process and storage of data and settings, and is compatible with WS2K3's features. For more information about the certification programs, go to http://www.microsoft.com/winlogo/software/windowsxp-sw.mspx.

Terminal Services Logon Scripts

In addition to any Windows domain logon scripts, Terminal Services uses a sequence of scripts that run upon user logon to assist you in making any adjustments to your applications so that they'll run in a multi-user environment. Figure 5.1 illustrates the user-logon process.

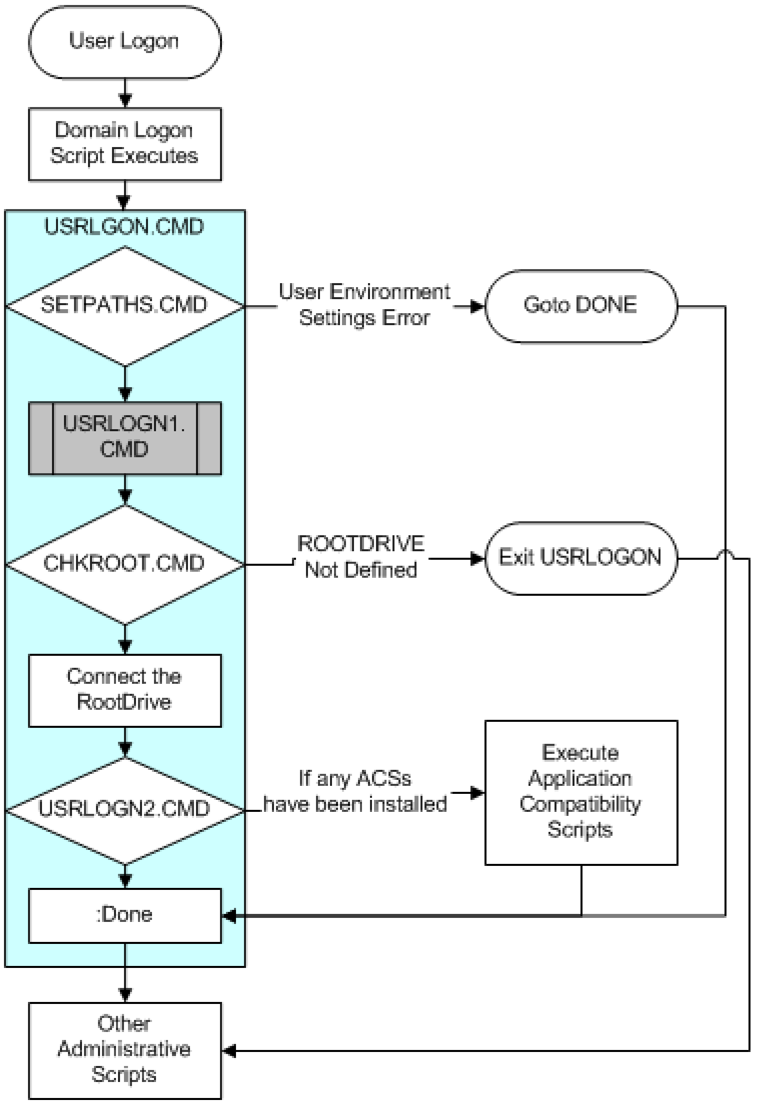

User Logon
Domain Logon Script Executes
USRLGON.CMD
SETPATHS.CMD
User Environment Settings Error
Goto DONE
USRLOGN1. CMD
CHKROOT.CMD
ROOTDRIVE Not Defined
Exit USRLOGON
Connect the RootDrive
USRLOGN2.CMD
If any ACSs have been installed
Execute Application Compatibility Scripts
:Done
Other Administrative Scripts

Figure 5.1: The user-logon process.

When you install Terminal Services, the system adds an entry to the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\Appsetup registry subkey that will execute the USRLOGON.CMD script that is stored in the \System32 directory. USRLOGON.CMD is the heart of the Terminal Services logon process, and it calls a number of additional scripts in its processing. I'll walk you through this script and its subscripts and explain a number of Terminal Services concepts along the way.

All the scripts that Microsoft provides are written in shell script, but WS2K3 also has the ability to use VBScript and JavaScript through the Windows Script Host (WSH).

USRLOGON.CMD

The first section of the USRLOGON.CMD script calls SETPATHS.CMD. Listing 5.1 shows this section.

```
REM USRLOGON.CMD
@Echo Off

Call "%SystemRoot%\Application Compatibility Scripts\SetPaths.Cmd"
If "%_SETPATHS%" == "FAIL" Goto Done
```

Listing 5.1: The first section of USRLOGON.CMD.

The SETPATHS.CMD subscript checks to make sure that the registry keys for the user's application environment are in place. The registry keys for the current user variables can be found in the HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders subkey, and the keys for all user variables are in the same subkey under HKEY_LOCAL_MACHINE. If any of these variables aren't defined, a warning message is displayed, and USRLOGON.CMD aborts. Table 5.1 provides the user environment values.

Environment Component	**Registry Value**
All Users: Startup	COMMON_STARTUP
All Users: Start Menu	COMMON_START_MENU
All Users: Start Menu\Programs	COMMON_PROGRAMS
Current User: Start Menu	USER_START_MENU
Current User: Startup	USER_STARTUP
Current User: Start Menu\Programs	USER_PROGRAMS
Current User: My Documents	MY_DOCUMENTS
Current User: Templates	TEMPLATES
Current User: Application Data	APP_DATA

Table 5.1: User environment values.

Next, as Listing 5.2 shows, the script checks for the existence of a script called USRLOGN1.CMD, and calls it. By default, this script doesn't exist and would only be used if you have an application that requires an application compatibility script that does not use a ROOTDRIVE. This type of application compatibility script would be one that makes changes to the HKEY_CURRENT_USER hive without referencing any user-specific file locations.

```
If Not Exist "%SystemRoot%\System32\Usrlogn1.cmd" Goto cont0
Cd /d "%SystemRoot%\Application Compatibility Scripts\Logon"
Call "%SystemRoot%\System32\Usrlogn1.cmd"

:cont0
```

Listing 5.2: USRLOGON.CMD checks for the existence of and calls USRLOGN1.CMD.

The ROOTDRIVE is a drive letter that the administrator reserves as an absolute path for the user's home directory—either network or local—that is the same for all users.

The next section of the script is designed to create a ROOTDRIVE. The concept of the ROOTDRIVE was created because most registry keys can't reference environment variables. For example, MyApplication.EXE may have a registry value called UserTemplates that defines the path used to store user-modifiable templates. The best option for this path would be to use %HOMEDRIVE%%HOMEPATH%\MyTemplates so that each user can be directed to his or her network home directory (if one exists) or to his or her profile directory on the terminal server. Because you can't use environment variables in this registry value, you need an absolute path that can be resolved for all users. Therefore, on a terminal server, you define a ROOTDRIVE.

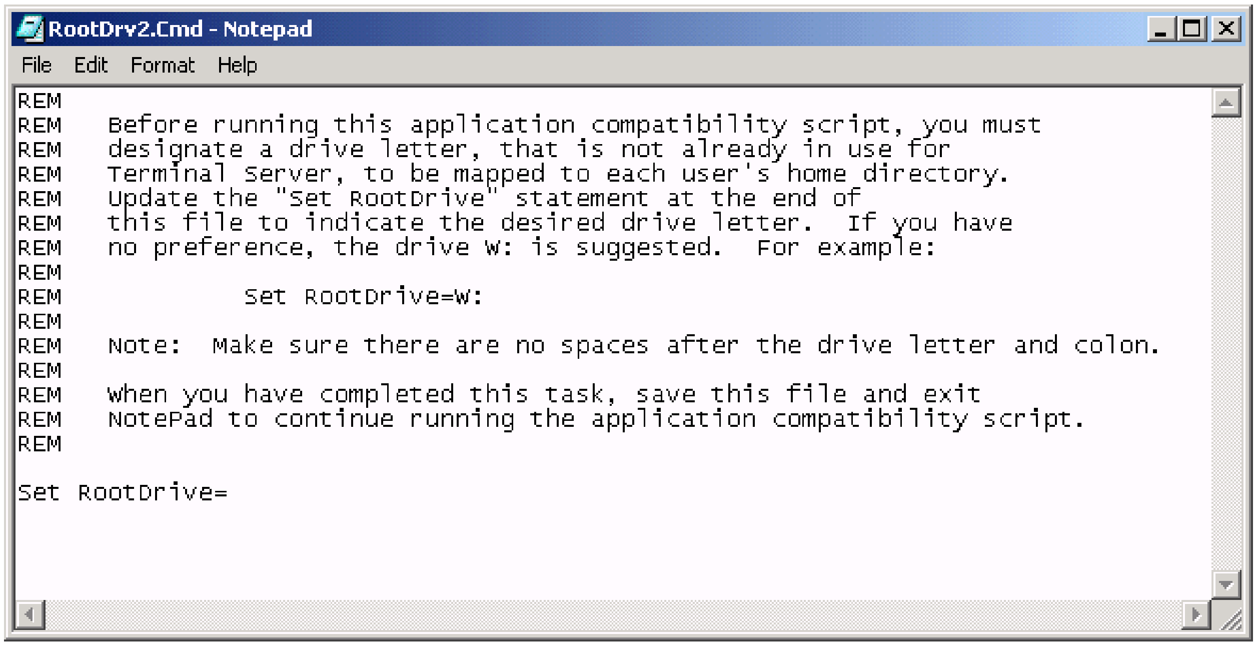
USRLOGON.CMD uses a SUBST command to connect the ROOTDRIVE letter directly to the user's home directory upon logon. Listing 5.3 shows this section of USRLOGON.CMD.

```
Cd /d %SystemRoot%\"Application Compatibility Scripts"
Call RootDrv.Cmd
If "A%RootDrive%A" == "AA" End.Cmd

Rem
Rem Map the User's Home Directory to a Drive Letter
Rem

Net Use %RootDrive% /D >NUL: 2>&1
Subst %RootDrive% "%HomeDrive%%HomePath%"
if ERRORLEVEL 1 goto SubstErr
goto AfterSubst
:SubstErr
Subst %RootDrive% /d >NUL: 2>&1
Subst %RootDrive% "%HomeDrive%%HomePath%"
:AfterSubst
```

Listing 5.3: USRLOGON.CMD connects the ROOTDRIVE letter to the user's home directory.

The SUBST command is used in Windows to map a drive letter to an absolute path, unlike NET USE, which maps a drive letter to a universal naming convention (UNC) path. Thus, SUBST W: C:\WINNT\FONTS would make the W drive an alias for the fonts folder.

How is the ROOTDRIVE letter selected and defined? When an administrator installs an application compatibility script that references a ROOTDRIVE, the script's installation script will automatically ask the administrator to edit a batch file called ROOTDRV2.CMD to define the letter the administrator wants to reserve, as Figure 5.2 shows. After this reservation is set, the letter can't be used for any other mapping on the terminal server.



```
REM
REM   Before running this application compatibility script, you must
REM   designate a drive letter, that is not already in use for
REM   Terminal Server, to be mapped to each user's home directory.
REM   Update the "Set RootDrive" statement at the end of
REM   this file to indicate the desired drive letter.  If you have
REM   no preference, the drive W: is suggested.  For example:
REM
REM            Set RootDrive=W:
REM
REM   Note:  Make sure there are no spaces after the drive letter and colon.
REM
REM   When you have completed this task, save this file and exit
REM   NotePad to continue running the application compatibility script.
REM

Set RootDrive=
```

Figure 5.2: Setting the ROOTDRIVE letter.

The process that Microsoft uses to connect to the ROOTDRIVE was originally written for Windows NT 4.0, Terminal Server Edition and doesn't take advantage of WS2K3's enhanced drive-mapping capabilities. To explain this concept, let me walk you through two examples of how ROOTDRIVE works under NT 4.0, one example using a network home directory, and one using a local profile directory.

Example 1

Joe User has a home directory defined in his user account that maps H to \\Server01\Home\Joe.User. Upon logon to the terminal server, H is mapped to \\Server01\Home (NT 4.0 can only map to the share level). In Joe's session, the %HOMEDRIVE% variable is now set to H, and the %HOMEPATH% variable is set to \Joe.User.

Joe uses an application that allows him to create personal templates for his documents, and there is a registry key that defines the path to store these template files. The registry can't reference environment variables, only absolute paths, so we cannot use %HOMEDRIVE%%HOMEPATH%. If we try to use H for UserTemplates, Joe's templates will be created in the root of the home share instead of in his directory.

If the administrator has used ROOTDRV2.CMD to set the ROOTDRIVE variable to W, then USRLOGON.CMD uses a SUBST command to map W to %HOMEDRIVE%%HOMESHARE% or H:\Joe.User. Now the path W:\ is an alias for H:\Joe.User, and the registry can reference W:\ whenever access directly to Joe's home directory is needed. In addition, this path can be used for the location of his templates.

Example 2

Jane Doe doesn't have a network home directory, so her templates should be stored in her user profile folder. In Jane's session, the %HOMEDRIVE% variable resolves to C and %HOMEPATH% resolves to \WTSRV\Profiles\Jane.Doe. Once again, we can't use variables in the registry key, so we don't have an easy way to reference Jane's profile directory.

The Administrator has set ROOTDRIVE=W: in the script, so once again USRLOGON.CMD connects W directly to Jane's profile directory, and we can use W:\ in the registry.

In Joe's example, the entire process is used to get around the fact that NT 4.0 can't map a network drive to a subdirectory of a share, so

```
      Net Use H: \\Server01\Share\Directory
```

would connect H to \\Server01\Share. But WS2K3 has the ability to map to the subdirectory itself, which enables using the home drive as the ROOTDRIVE.

However, without modification USRLOGON.CMD always clears any existing mapping on the ROOTDRIVE letter before performing the SUBST command, so it doesn't take advantage of WS2K3's enhanced drive-mapping capabilities. Why would you want to use two drive letters to reference the exact same directory path? You're better off modifying USRLOGON.CMD to take advantage of WS2K3's abilities. You still need to compensate for any users who don't have network home directories, but you can easily do so.

If you would like to read Microsoft's explanation of the ROOTDRIVE, see the Microsoft article "How and why ROOTDRIVE is used on Windows Terminal Server" at http://support.microsoft.com/default.aspx?scid=kb;[LN];195950.

Let's assume that you use network home directories, and you map these on the H drive. If you set your ROOTDRIVE to H as well, you can avoid mapping two drive letters altogether. Listing 5.4 shows my suggested change to USRLOGON.CMD (the suggested changes are in red).

```
Cd /d %SystemRoot%\"Application Compatibility Scripts"
Call RootDrv.Cmd

If "A%RootDrive%A" == "AA" goto done

REM If the user has a network Home Directory already mapped
REM on the ROOTDRIVE, we do not need to do anything.

if /I "%rootdrive%" == "%homedrive%" goto NoSubst

:DoSubst
Net Use %RootDrive% /D >NUL: 2>&1
Subst %RootDrive% "%HomeDrive%%HomePath%"
if ERRORLEVEL 1 goto SubstErr
goto AfterSubst
:SubstErr
Subst %RootDrive% /d >NUL: 2>&1
Subst %RootDrive% "%HomeDrive%%HomePath%"
:AfterSubst

:NoSubst
```

Listing 5.4: Modified USRLOGON.CMD.

☞ If you aren't installing any applications that require application compatibility scripts, the ROOTDRIVE will never be created, and your logon process is greatly simplified.

After the ROOTDRIVE has been established, USRLOGON.CMD calls any application compatibility scripts that have been installed, then exits, as Listing 5.5 shows.

```
Rem Invoke each Application Script.  Application Scripts are
automatically
Rem added to UsrLogn2.Cmd when the Installation script is run.
Rem

If Not Exist %SystemRoot%\System32\UsrLogn2.Cmd Goto Cont1

Cd Logon
Call %SystemRoot%\System32\UsrLogn2.Cmd

:Cont1

:Done
```

Listing 5.5: USRLOGON.CMD calls any application compatibility scripts, then exits.

When you install an application compatibility script, a call to it is added to the USRLOGN2.CMD script so that all application compatibility scripts can be called in turn without modifying USRLOGON.CMD.

Additional Administrative Scripts

In addition to USRLOGON.CMD and application compatibility scripts, you might want to run an additional script when your users log on to a terminal server. For example, you might want to create scripts that map additional network drives or connect to specific printers. If you are in an AD environment, you can add logon scripts to a User Group Policy Object (GPO).

On a workgroup mode terminal server, you can have the system execute an additional script by copying the script file to the \System32 directory and modifying the AppSetup value of the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon registry subkey. By default, this key lists USRLOGON.CMD, but you can add scripts that are separated by commas.

For more information about adding scripts, see the Microsoft article "How to Set Up a Logon Script Only for Terminal Server Users" at http://support.microsoft.com/support/kb/articles/q195461.asp.

☞ Rather than edit the AppSetup registry subkey, you can call additional scripts from within USRLOGON.CMD. If you aren't using a ROOTDRIVE, insert the call after :Cont0. If you are using a ROOTDRIVE, insert the call at the end of the script.

Application Installation

Now that you understand the process that Terminal Services uses to create an environment that can assist applications in running for simultaneous users, we can look at how the actual software-installation process works on a Terminal Services server. In a perfect world, all applications would follow the guidelines laid out in the Designed for Microsoft Windows XP logo specification.

This specification instructs programmers to take advantage of several Windows component services that would make the application natively compatible with WS2K3 and Terminal Services. The following list quotes and describes the elements of the specification that are of particular interest to the Terminal Services administrator:

- *Do not read from or write to Win.ini, System.ini, Autoexec.bat, or Config.sys on any Windows operating system based on NT technology*—Programs that don't obey this rule might store per-user settings in these per-machine configuration files.
- *Install using a Windows Installer-based package that passes validation testing* and *Ensure that your application supports advertising*—The Windows Installer service uses a process called *advertising* to ensure that per-user registry keys and files are installed for each user of a computer and not just the user who installed it.
- *Default to My Documents for storage of user-created data*—Compliance with this item ensures that per-user files (documents, macros, templates, and so on) are stored in a per-user location, not in the program's directory.

In the real world, however, systems administrators have to deal with many applications—both current and legacy—that don't adhere to these guidelines or were written before the specification was established. To assist in integrating these types of applications, Terminal Services utilizes registry mapping, INI file mapping, and application compatibility scripts.

Registry Mapping

The Windows registry is divided into two main sections: HKEY_CURRENT_USER and HKEY_LOCAL_MACHINE. HKEY_LOCAL_MACHINE is used to store global system-configuration information such as network settings, hardware configuration, and software settings that are the same for all users. HKEY_CURRENT_USER stores per-user settings such as cosmetic settings, user preferences, and per-user software configuration. Each user that logs on to Windows has his or her own HKEY_CURRENT_USER hive.

When you install an application, the HKEY_CURRENT_USER registry keys are created by the installation program. When other users log on to the computer, they will also need those HKEY_CURRENT_USER registry keys in order for the application to run as expected. Some applications use *self registration* to create these keys. When the application is launched, it looks for the keys it needs under HKEY_CURRENT_USER. If those keys do not exist, a subroutine is called within the program to create the keys and set them to default values.

If an application uses the Windows Installer Service, the application can use *advertising* to create the HKEY_CURRENT_USER keys. When an advertised application is launched from the start menu, the Windows Installer Service is invoked and performs a per-user repair on the application. The repair process creates all required user registry keys and any per-user files that the application may need.

Applications that don't use self-registration or advertising only write HKEY_CURRENT_USER registry information into the hive belonging to the user that installed the application, so when another user logs on to Windows, crucial registry settings may be missing from HKEY_CURRENT_USER. This shortcoming is usually not evident on a workstation, as the computer is often used by only one person. A terminal server by its very nature, however, is designed to be shared by multiple users.

To guarantee that all users receive the proper registry settings, Terminal Services uses a process called registry mapping. When installing an application, the Terminal Services server is placed into install mode. This mode causes the server to monitor the installation program for any writes to HKEY_CURRENT_USER. As Figure 5.3 shows, any keys that are written to HKEY_CURRENT_USER are automatically copied to a special location under HKEY_LOCAL_MACHINE—they are copied to the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software subkey.

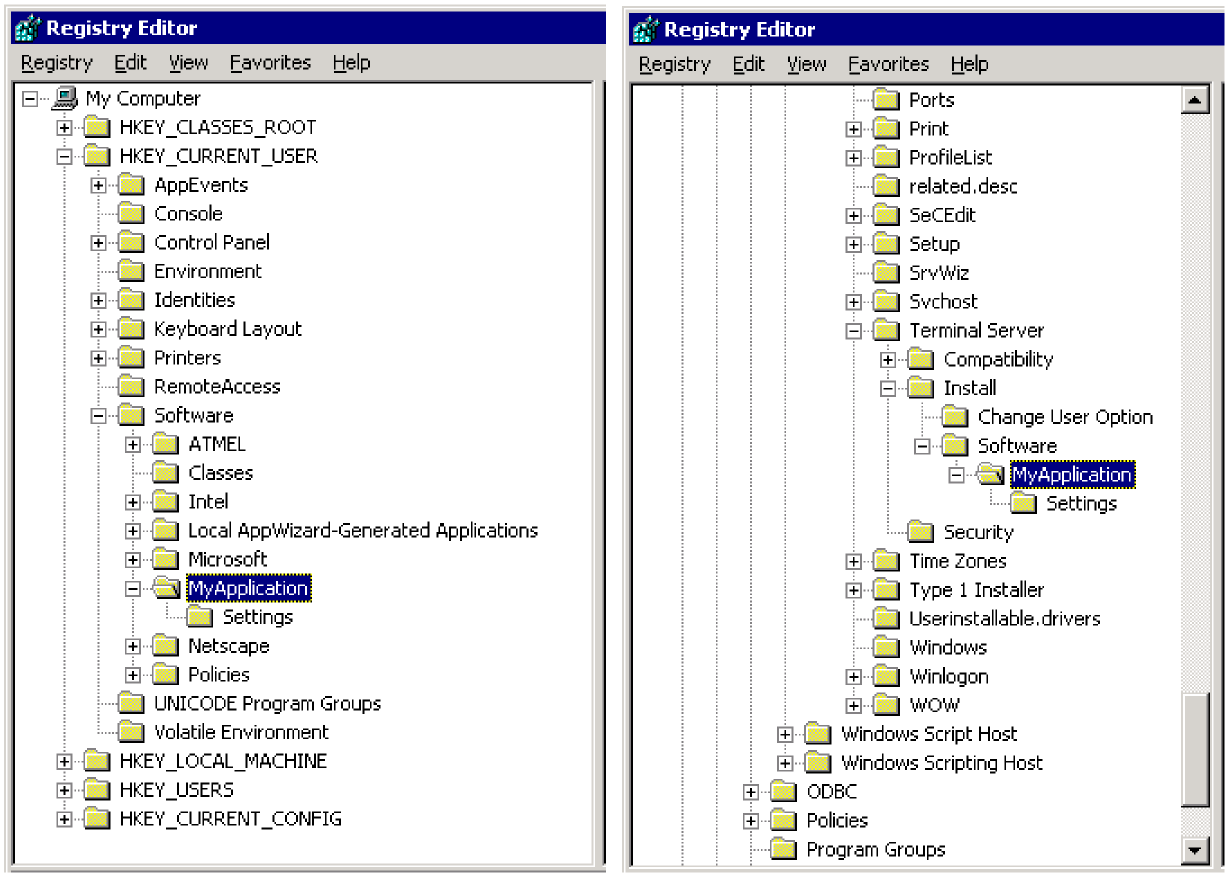


Figure 5.3: Registry mapping in install mode for a program called MyApplication.

This mapping of registry keys works only if the installation program uses standard API calls to write to the registry. Always check the registry after installing an application and before launching it for the first time to make sure that the mirror has been created. If not, you might need to manually copy the keys.

After application installation is complete, the server is put into execute mode. In this mode, if an application tries to read a registry key from HKEY_CURRENT_USER, and the key isn't there, the system will automatically check whether the key exists under the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software subkey. If it exists there, the system will copy the key to HKEY_CURRENT_USER.

INI File Mapping

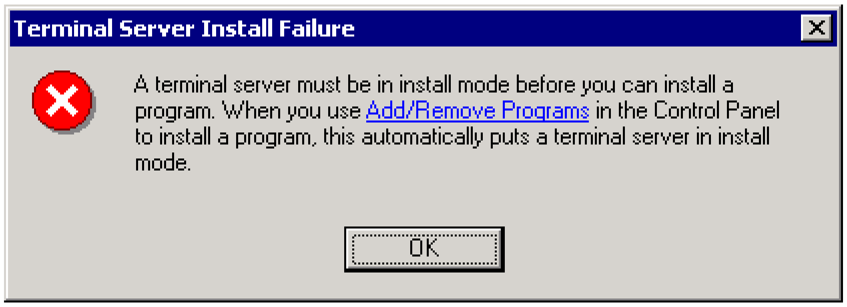
Before Windows 95, all settings for the system and applications were stored in INI files (initialization files). Legacy applications will occasionally use these files today. Unlike the registry, which has separate areas for per-user and per-machine settings, INI files are global, so changes made by one user would affect everyone on the terminal server. In addition, most terminal servers are configured so that non-administrative users don't have adequate rights to make changes to these system-level files, so applications that reference the files would not run under the user's context.

To compensate for this behavior, Terminal Services creates copies of the system INI files and stores them in each user's home directory. When an application tries to read from or write to a system INI file, Terminal Services redirects the call to the user's copy instead of the original.

While in install mode, any changes that the installer program makes to these INI files are made to the original copies in the %SYSTEMROOT% directory. Upon logon, the system checks the user's copies of the system INI files against the ones in the WINNT directory, and if the user's copies of the system INI files are older, the system updates the user's files. You can disable this version-checking process by modifying a registry setting, which I'll discuss later.

Install and Execute Modes

For registry and INI file mapping to work, the system must be in the proper mode for either application installation or execution. To switch the server into install mode, simply invoke the Add New Programs Wizard in the Add/Remove Programs Control Panel applet. To switch back to execute mode, close the wizard. If you try to run a SETUP.EXE program from outside the Control Panel, the terminal server will correct you by sending the error message that Figure 5.4 shows.


Terminal Server Install Failure

A terminal server must be in install mode before you can install a program. When you use Add/Remove Programs in the Control Panel to install a program, this automatically puts a terminal server in install mode.

Figure 5.4: Attempting to run an installation program outside the Add/Remove Programs Control Panel applet results in the presentation of this warning box.

The terminal server won't catch all installation programs, so be sure to get in the habit of always using the Control Panel to install new applications.

Alternatively, you can toggle between install and execute mode from a command line using the following commands:

```
CHANGE USER /INSTALL
CHANGE USER /EXECUTE
```

These commands come in handy if you want to script your application-installation processes.

Application Compatibility Scripts

Many applications don't take Terminal Services into account and store user-customizable components on the C drive of the computer. These components can include templates, macros, and dictionary files. On a workstation, if a user modifies any of these files, the change would also affect any other user that logs on to the same computer. But on a terminal server, changes would also affect any other users logged on simultaneously. This behavior can create problems when these files are in use by one user and can't be accessed by the instance of the application running in another user's session.

Application compatibility scripts compensate for this problem by copying these components to a location that is unique for each user (usually the home directory), then instructing the application about where to find the new copies. You can also use application compatibility scripts to grant user access to certain file and registry locations that are restricted under Terminal Services.

The fact that most application vendors are now complying with the Windows Logo specification is demonstrated by the reduced the number of application compatibility scripts included with Windows. You can find the remaining scripts in C:\WINNT\Application Compatibility Scripts (see Figure 5.5). The scripts are divided into three groups:

- Install—These scripts make system-level changes and add entries to USRLOGON2.CMD if the application also requires a logon application compatibility script for each user.
- Logon—These scripts are called by USRLOGON2.CMD upon user logon; they copy user components to the ROOTDRIVE and make HKEY_CURRENT_USER registry changes.
- Uninstall—These scripts remove the calls to the logon scripts from USRLGON2.CMD when you no longer need the application compatibility script to run.

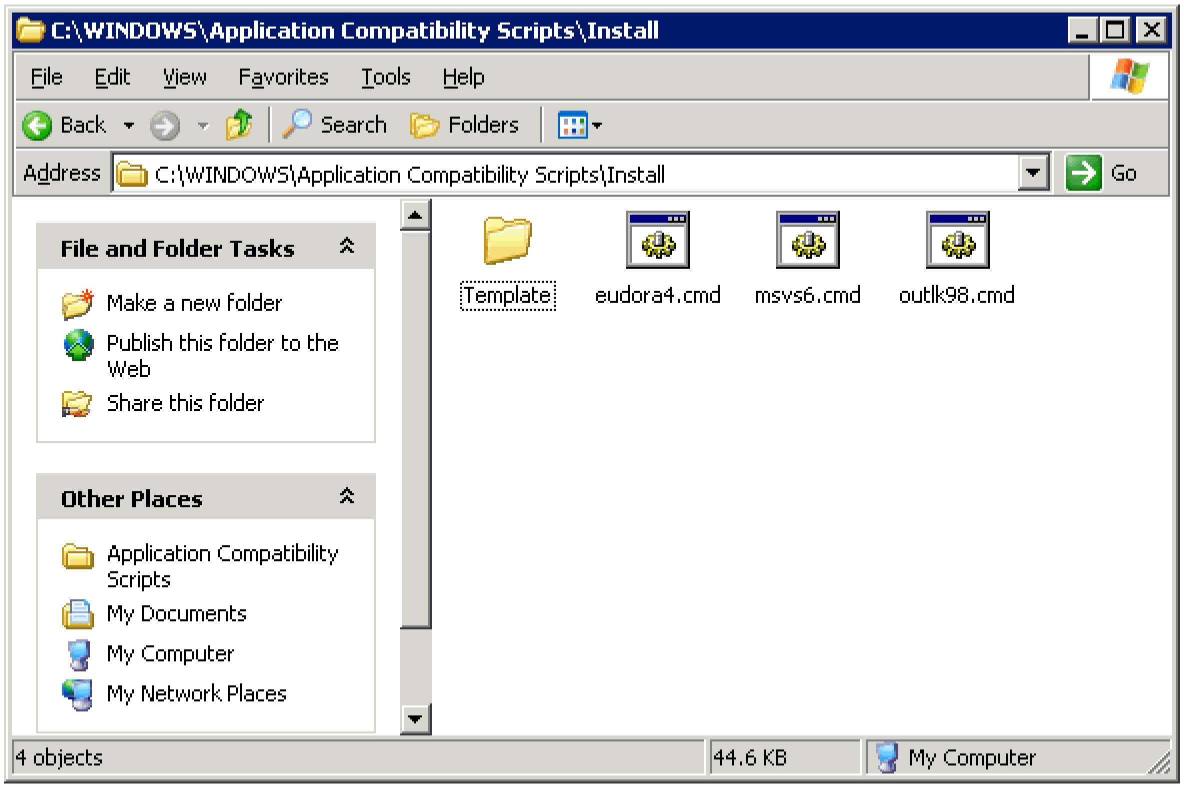


Figure 5.5: Application compatibility scripts provided by Microsoft.

As you can see, Microsoft only includes application compatibility scripts for Eudora 4, Visual Studio 6, and Outlook 98. If you are still using other legacy applications in your environment (such as Office 97, Project 95, and so on) you might want to copy the application compatibility scripts folder from a Win2K terminal server to a network share for reference.

When you install an application that requires an application compatibility script, you run the installation script after installing the application itself. This installation script makes any system changes that are needed and instructs USRLOGN2.CMD to run the application compatibility script logon script when users log on.

If you're installing a non–logo–compliant application that doesn't have a Microsoft-provided application compatibility script, see "Installing Undocumented Applications" later in this chapter for information about how to determine whether you need an application compatibility script and how to write one.

Examples of Terminal Services Application Installations

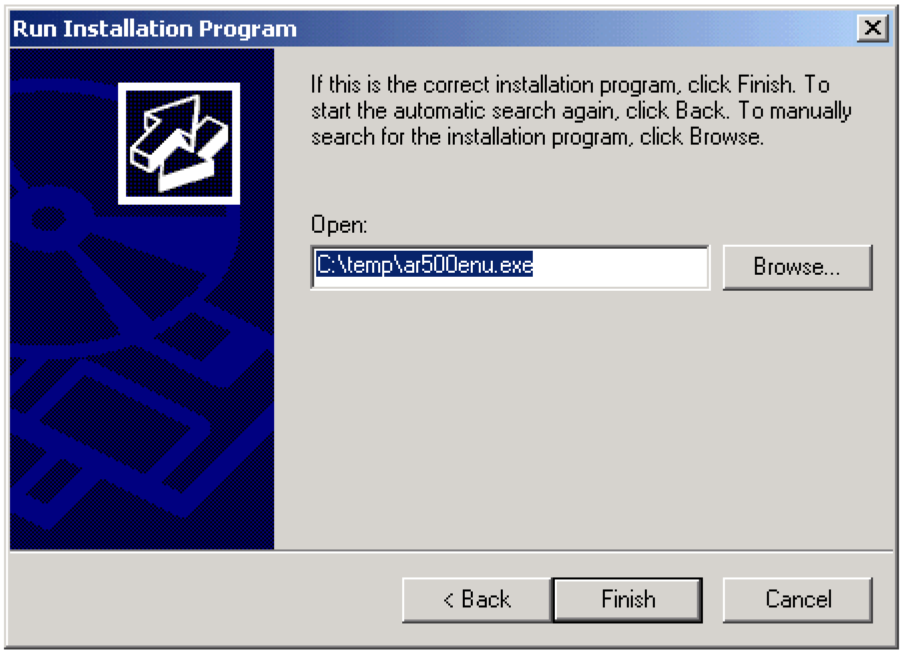
Let's walk through the installation process for three applications: an application that has no problems running on Terminal Services, an application that comes with a special installation method for Terminal Services, and an application that requires an application compatibility script.

As I mentioned earlier, most current applications are natively compatible with Terminal Services, so you might find that you can build a successful Terminal Services infrastructure without ever creating a transform or writing an application compatibility script. However, these skills are important to learn in case you encounter a mission-critical application that does not run under Terminal Services out of the box.

The three applications I'll use for these examples are "old"—Acrobat Reader 5, Netscape Navigator 4, and Office 2000—but many organizations still run these applications. In addition, they provide useful examples that you can apply to other applications.

Simple Installation

Adobe Acrobat Reader is a free utility used to open PDF files. This utility has no problems running on Terminal Services. To install it, you simply use the Add New Programs Wizard in the Add/Remove Programs Control Panel applet, which Figure 5.6 shows. This wizard places the server into install mode.


Run Installation Program

If this is the correct installation program, click Finish. To start the automatic search again, click Back. To manually search for the installation program, click Browse.

Open: C:\temp\ar500enu.exe

Figure 5.6: The Add New Programs Wizard.

After the installation is complete, click Finish. The server will return to execute mode and Acrobat Reader is ready for your Terminal Services users. As an exercise, open the registry editor and compare the settings in HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software\Adobe to those in HKEY_CURRENT_USER\Software\Adobe to see the registry mapping in action. When a new user launches Acrobat for the first time, the system will copy the keys into the user's HKEY_CURRENT_USER registry hive.

At the time of this writing, Adobe has just released Acrobat Reader 6. This new version is packaged in MSI format. It also does not require any special steps for terminal server installation.

Custom Installation

Some software manufacturers take Terminal Services into account when writing their programs and provide you with instructions about how to install their applications on a terminal server. As an example, let's go through the process of installing an application that was designed with Terminal Services in mind—Microsoft Office 2000. Office 2000 in its raw form has a number of problems when running on Terminal Services:

- Install on First Use—Because users don't have the rights to install components on the terminal server, the Install on First Use option in the Office setup isn't appropriate. Instead, we need to set all components to either Run from My Computer or Not Available.
- User's name and initials—The Office installer asks for the user's name and initials during setup, so all users of the terminal server would end up with the installer's name. You need to enable the NOUSERNAME switch to force Office to ask each user for his or her name on the first use.
- Animation—The frequent screen redraws required by animation are challenging in a Terminal Services environment, so animation should be avoided where possible. The Office Assistant is a major culprit and should not be installed.

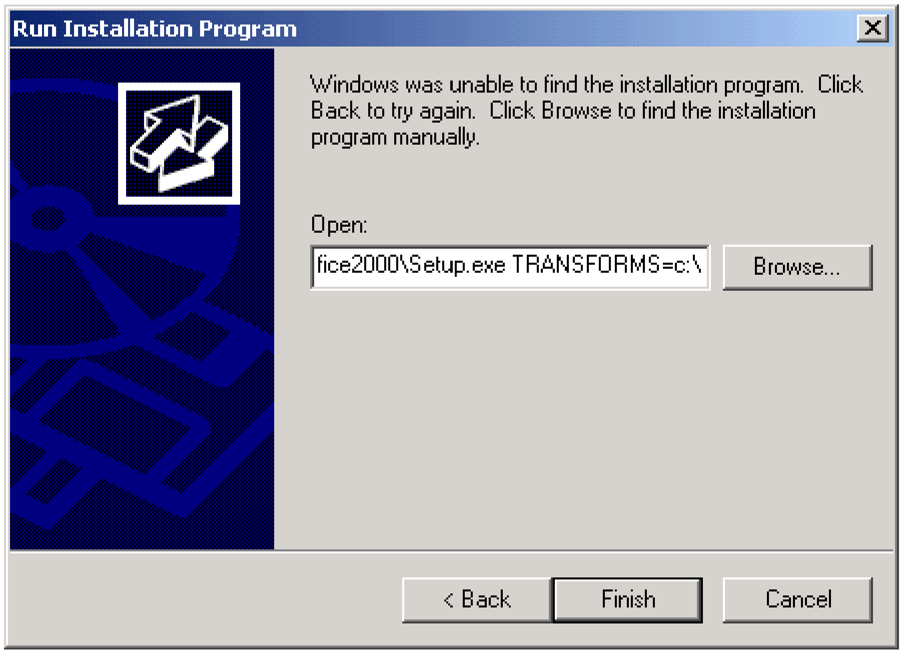
These problems are addressed by using the Office 2000 Terminal Services transform (TermSrvr.mst). This transform is a special file that instructs Windows Installer to configure Office for Terminal Services and to omit certain features of Office 2000 that are problematic.

☞ The Terminal Services transform can be found in the core tool set of the Office resource kit, which you can download from http://www.microsoft.com/office/ork/2000/appndx/toolbox.htm#orktools. Microsoft also provides a detailed white paper about installing Office on a terminal server at http://www.microsoft.com/office/ork/2000/two/30t3_2.htm.

After you have the MST file, you are ready to install Office 2000. Begin by opening the Add/Remove Programs Control Panel applet, and selecting Add New Programs. Click CD or Floppy, browse to your Office source files, and select SETUP.EXE. Now you must instruct the installer to use the transform. To do so, add

```
TRANSFORMS=path\TermSrvr.mst
```

to the command, as Figure 5.7 shows.


Run Installation Program

Windows was unable to find the installation program. Click Back to try again. Click Browse to find the installation program manually.

Open:

fice2000\Setup.exe TRANSFORMS=c:\

Figure 5.7: Installing Office 2000 with the Terminal Services transform.

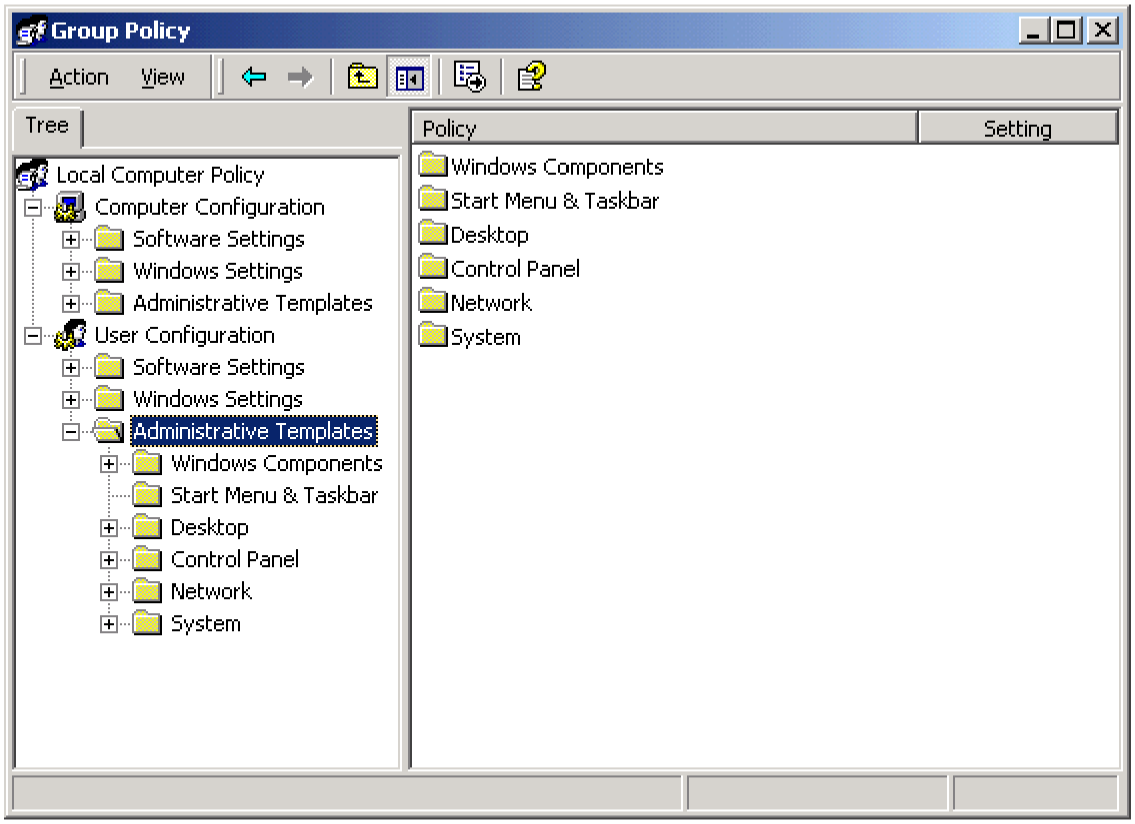
After the installation is complete, click Finish to put the Terminal Services server back into execute mode. I recommend installing the latest service release for Office, as there are some bugs that affect Terminal Services users that have been corrected.

You should now take the time to set up some custom settings for your users. You can do so by using the Office Profile Wizard or the Office 2000 ADM files in conjunction with the local policy editor or a domain Group Policy. If you want to use the Office Profile Wizard, follow the documentation that Microsoft provides. If you're in an AD domain and are using Group Policy to manage your user settings, see Chapter 4.

To use local policy, find the ADM files that were included in the resource kit (which you downloaded to get the MST file), then launch the local policy editor, which Figure 5.8 shows, by typing

```
GPEDIT.MSC
```

from the Start menu's Run text box.



Local Computer Policy
- Computer Configuration
  - Software Settings
  - Windows Settings
  - Administrative Templates
- User Configuration
  - Software Settings
  - Windows Settings
  - Administrative Templates
    - Windows Components
    - Start Menu & Taskbar
    - Desktop
    - Control Panel
    - Network
    - System

Policy	Setting
Windows Components	
Start Menu & Taskbar	
Desktop	
Control Panel	
Network	
System	

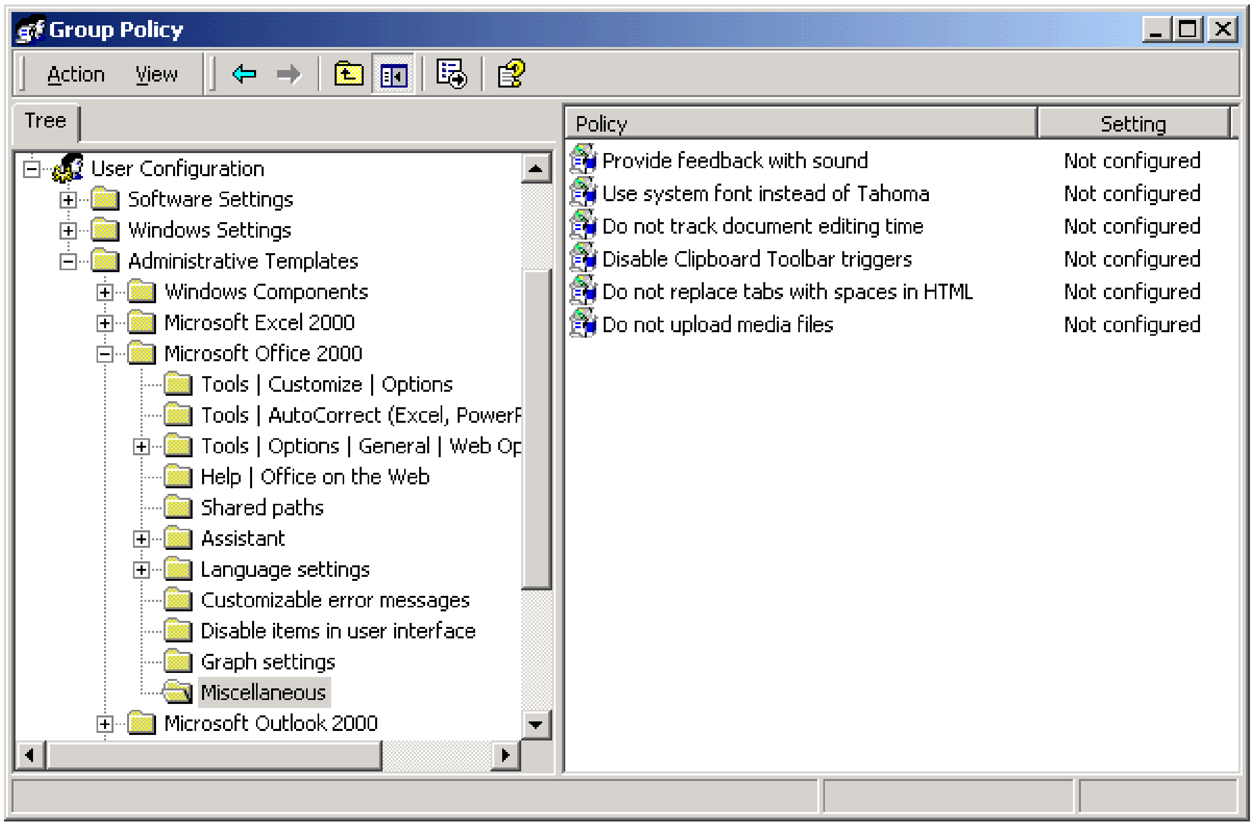
Figure 5.8: The Group Policy editor.

Next, right-click Administrative Templates, and select Add/Remove Templates. Add the ADM files for the programs that you have installed. Table 5.2 lists the ADM files associated with Office programs.

Office Program	**ADM File**
General Office Settings	OFFICE9.ADM
Word 2000	WORD9.ADM
Excel 2000	EXCEL9.ADM
PowerPoint 2000	PPOINT9.ADM
Outlook 2000	OUTLK.ADM
FrontPage 2000	FRONTPG4.ADM
Access 2000	ACCESS9.ADM
Publisher 2000	PUB9.ADM

Table 5.2: Office ADM files for local or Group Policy.

After you have added the ADM files, you have access to a large number of settings that you can preconfigure or disable for your users, as Figure 5.9 shows.



User Configuration
- Software Settings
- Windows Settings
- Administrative Templates
  - Windows Components
  - Microsoft Excel 2000
  - Microsoft Office 2000
    - Tools | Customize | Options
    - Tools | AutoCorrect (Excel, PowerF
    - Tools | Options | General | Web Op
    - Help | Office on the Web
    - Shared paths
    - Assistant
    - Language settings
    - Customizable error messages
    - Disable items in user interface
    - Graph settings
    - Miscellaneous
  - Microsoft Outlook 2000

Policy	Setting
Provide feedback with sound	Not configured
Use system font instead of Tahoma	Not configured
Do not track document editing time	Not configured
Disable Clipboard Toolbar triggers	Not configured
Do not replace tabs with spaces in HTML	Not configured
Do not upload media files	Not configured

Figure 5.9: Settings available with the Office ADM file.

You should go through these settings and customize Office for your users. You'll want to disable any features that use sound or animation. Remove the Detect and Repair command from all the Office products. You might also want to disable the Check Spelling as you Type and Check Grammar as you Type options in each program, as these features are very processor intensive, especially when you have multiple users.

You should be aware that if you use the Local Machine Policy to configure your settings, the settings apply to all users, including administrators. But, for settings within Office programs, this behavior is usually not a problem.

With the release of the Windows XP generation of Office programs, Microsoft has begun to include the required modifications for terminal servers in the native MSI package. When you install Office XP on a terminal server, Windows Installer recognizes that you are on a terminal server and makes the changes to the setup parameters automatically. A transform is no longer needed.

Even with these new enhancements in Office XP setup, there are still some post-installation changes you will want to make using the Office XP ADM files. For details, see http://www.microsoft.com/office/ork/xp/one/deph02.htm.

Application Compatibility Script Installation

Some applications require post-installation modifications or on-the-fly changes upon user logon. These changes are made by using an application compatibility script. You have already seen the list of application compatibility scripts that Microsoft provides for Terminal Services, and I'll use one of these applications as an example—Netscape Communicator 4.5.

We begin, as with any other application, with the Add New Programs Wizard. Install Netscape as you normally would, then click Finish to close the wizard. Now we must install the application compatibility script. Before we do so, let's examine why Navigator 4.5 requires an application compatibility script:

- User Profiles—By default, Navigator 4.x stores user profiles (favorites, settings, history) in the program's directory. Users on a terminal server don't have write access to the Program Files directory, so we need to direct Netscape to store profiles in the user's home directory instead of on the terminal server, and automate the profile setup for each user upon logon.
- User Profile Manager—Navigator has a profile-manager program that, in its native state, allows users to modify or delete other profiles on the terminal server. We need to restrict access to this utility.
- Quick Launch Toolbar—Users will want an icon for Netscape in their Quick Launch toolbar. The application compatibility script can create this icon for them upon logon.

To run the script, browse to C:\WINNT\Appliation Compatibility Scripts\Install and run NETCOM40.CMD. The script performs the following actions:

- Checks whether you've defined your ROOTDRIVE letter and asks you to do so if you haven't.
- Copies the Quick Launch icon from your profile to the Netscape program directory as a template for future users.
- Copies the Netscape profile directory that the setup program created for you to be a template for other users.
- Applies restrictive permissions to the Netscape profile-manager utility to prevent non-administrators from running it.
- Modifies the application compatibility script logon script for Netscape to reflect the ROOTDRIVE letter that you have chosen.
- Adds a call to the application compatibility script logon script for Netscape to USRLOGN2.CMD so that the application compatibility script will run upon logon for all users.

When a user logs on, COM40USR.CMD (the application logon script for Netscape), is called and performs the following actions:

- Copies the template User Profile to the user's ROOTDRIVE if it doesn't already exist.
- Modifies the Netscape section of HKEY_CURRENT_USER to point Netscape to the profile in the ROOTDRIVE.
- Creates the Quick Launch icon if it doesn't exist.

Netscape is now ready for simultaneous users.

Installing Undocumented Applications

You can now install any application that is either natively compatible with Terminal Services or comes with Terminal Services-compatibility instructions. But what do you do when you encounter an application that has no Terminal Services documentation? You need to learn how to determine whether any post-installation changes are necessary or an application compatibility script needs to be written. This skill is important but is less frequently used as more and more applications follow Microsoft's certification specifications.

Begin by checking the application publisher's Web site and search the online support area for references to "terminal server" or "Citrix" (even though we're installing the application on a Terminal Services server without MetaFrame, you might find useful information indexed under Citrix, as it has been the leader in terminal services for so long). If you can find nothing there, turn to online discussion forums and query for the name of the application AND "terminal server" OR "Citrix." Keep in mind that anyone can post to most forums, so you need to be very careful with any advice you find.

The following Web sites are useful in searching for help with installing an undocumented application:

http://www.thethin.net

http://www.thinplanet.com

After collecting any information you find about your application, install the application on a test server. Your test server should have the same configuration as your production boxes—service packs, hotfixes, logon scripts, and so on.

Never install an untested application on a production server!

As with any other application, install through the Add New Programs Wizard. If the application requires a post-installation reboot, do that as well. Before launching the application for the first time, examine the registry. Begin in the HKEY_LOCAL_MACHINE\SOFTWARE key and find the subkey for the application. Look through the values and pay careful attention to any data that contains an absolute path. Values such as InstallPath or ApplicationSource are fine, as these would be the same for all users, but values that reference user settings, such as UserDictionary or UserHome, may be problematic. Make notes of these keys.

Next, look at the HKEY_CURRENT_USER\Software key to determine whether the application configured any registry keys. If so, look for the same types of values here and make notes of any that you find. Also check whether the keys were mirrored in the Terminal Server section of the HKEY_LOCAL_MACHINE hive. If not, you should create a .REG file of the application's HKEY_CURRENT_USER\Software key by selecting regedit's Export Registry File option from the Registry menu. You will need this file later if you need to manually mirror the keys.

Be sure to export the application's HKEY_CURRENT_USER\Software key before you launch the application for the first time. You don't want to capture any current user settings that are generated at first launch.

Now launch the application under the same account that was used to install it. Go through the application's functions and look for problems—error messages, incorrect behaviors, and so on. Then log on to the terminal server with a test account. This account should not be an administrator on the terminal server and should be configured like your regular users. Launch the application and run through the same tests. If no errors are found, close the application, and check the registry for this user.

Confirm that the application's HKEY_CURRENT_USER keys were properly copied from the Terminal Server key in HKEY_LOCAL_MACHINE or were automatically generated by the application. If there are keys missing when you compare them to those under the installer account, you should mirror the keys yourself. To do so:

1. Open in Notepad the .REG file that you previously created, and select Replace from the Edit menu to change all occurrences of HKEY_CURRENT_USER\Software to HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software.
2. Import the .REG file back into the registry.

Now delete the profile (both local and roaming) of your test account, and log on. Launch the application and confirm that the mirrored registry keys were copied. If they were, and you determine that you need to make any registry changes for your users, you can make them under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software and the new settings will be copied when your users launch the application for the first time.

Now look at your notes for any registry keys that you found that point to user-specific files. If you found any references to user files stored on the local drives of the terminal server, you should try copying these files to a subdirectory of your ROOTDRIVE, then modifying the registry key to point to the new location. If the key is in HKEY_LOCAL_MACHINE, just modify it there. If the key is in HKEY_CURRENT_USER, change it there AND in the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software subkey. Be sure to document any changes you make so that you can make them again when you install the application on your production terminal server. Launch the application and confirm that it doesn't have a problem with the new location.

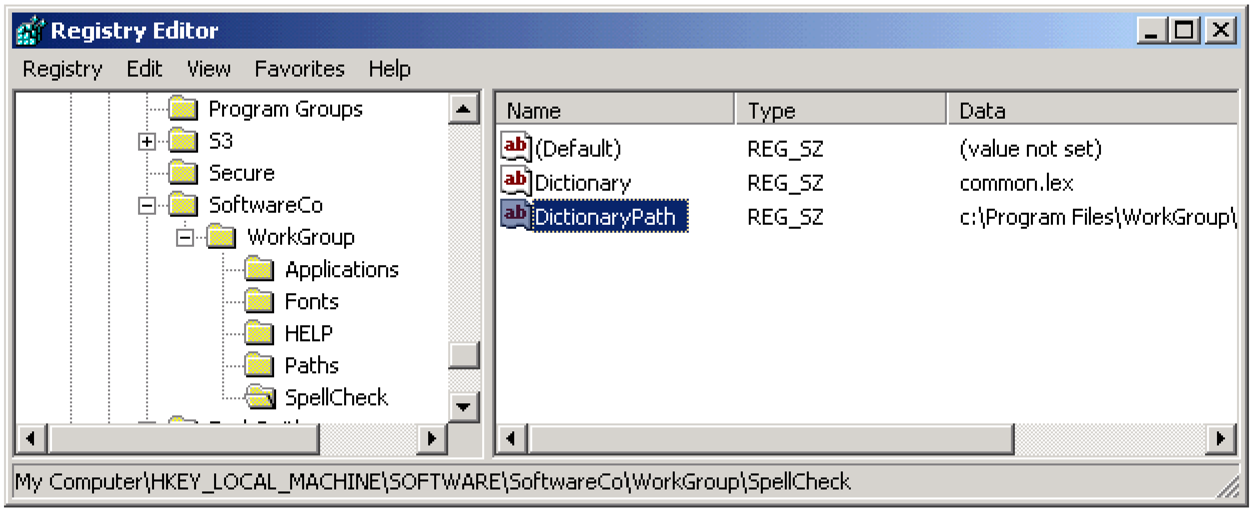
This process is mostly trial and error and is different for each application you install. Take heart in the fact that you need to go through this process only for applications that aren't certified for Windows.

A Real World Example

Let's walk through an example of an application that needs to be modified for terminal server use. Let's assume that WorkGroup is a problem-tracking system that your company uses to manage projects. It references a SQL database to store project descriptions, timelines, and team members' notes. You want to install WorkGroup on your terminal server so that you can have your users run it from within a Web page (using the Remote Desktop Web Connection client) instead of installing it locally on every workstation.

The publisher's Web site makes no reference to terminal server, and when you call the company's technical support number, you're informed that the vendor "Does not support the application on Terminal Services." You also check the usual newsgroups, but as this application is uncommon, you find no mention of it. You'll need to test the application yourself to determine whether it's compatible with Terminal Services and requires an application compatibility script.

Begin by using the Add New Programs Wizard to run SETUP.EXE for WorkGroup. The installer doesn't require a reboot, so immediately go to regedit, and examine HKEY_LOCAL_MACHINE\SOFTWARE for absolute paths under WorkGroup's key, as Figure 5.10 illustrates.



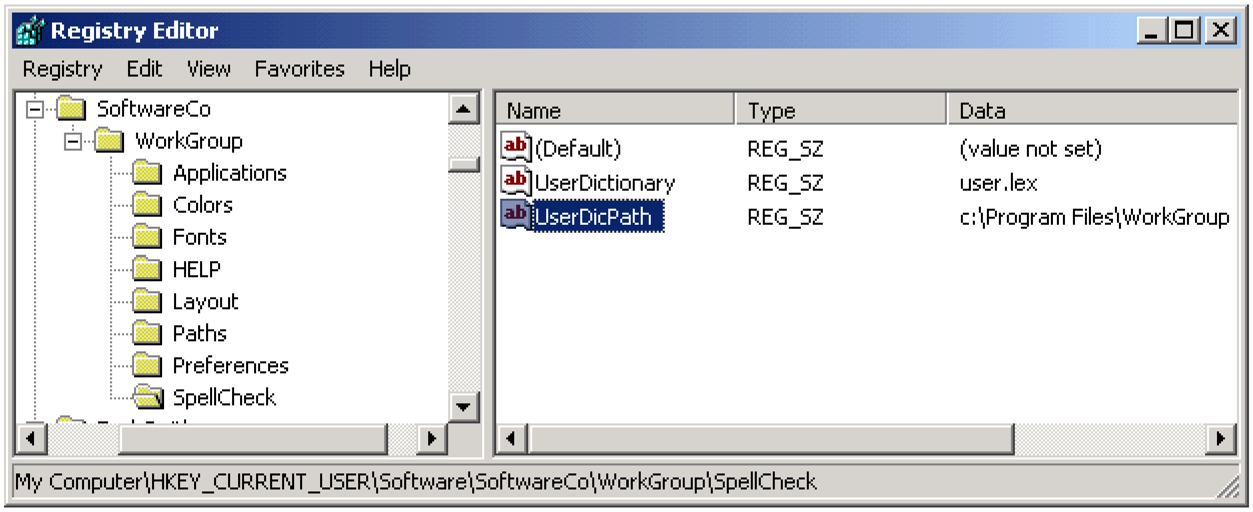
Name	Type	Data
(Default)	REG_SZ	(value not set)
Dictionary	REG_SZ	common.lex
DictionaryPath	REG_SZ	c:\Program Files\WorkGroup\

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\SoftwareCo\WorkGroup\SpellCheck

Figure 5.10: HKEY_LOCAL_MACHINE key for the WorkGroup application.

Under the Paths key, you find the location of the SQL database, which will be the same for all users, so this key shouldn't be a problem for Terminal Services. Under SpellCheck, however, you find a value that may pose a problem—DictionaryPath. After closer inspection, this value looks like it's the base lexicon file that all users will share, not a per-user file. Make a note of this key, just in case.

Next, look to HKEY_CURRENT_USER\Software for similar values. Here you find another SpellCheck key, and it contains the location for the user's custom dictionary file—C:\Program Files\Workgroup, as Figure 5.11 shows.



SoftwareCo
WorkGroup
Applications
Colors
Fonts
HELP
Layout
Paths
Preferences
SpellCheck

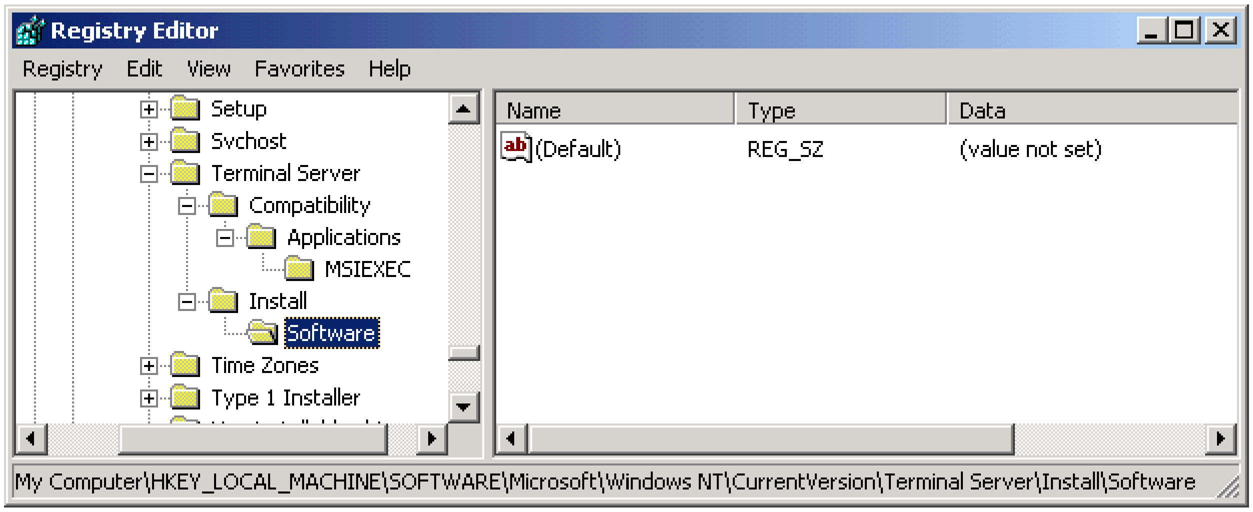
Name	Type	Data
(Default)	REG_SZ	(value not set)
UserDictionary	REG_SZ	user.lex
UserDicPath	REG_SZ	c:\Program Files\WorkGroup

My Computer\HKEY_CURRENT_USER\Software\SoftwareCo\WorkGroup\SpellCheck

Figure 5.11: Example of a per-user file stored in a common location.

This key raises a red flag for Terminal Services. Each user should have his or her own USER.LEX file for WorkGroup, but this file, by default, is stored on the terminal server, not in each user's home directory. Luckily, WorkGroup gives you a modifiable registry key for the file's location, so the problem should be easy to fix. You make a note that this problem will have to be addressed.

Next, you determine whether registry mapping has worked by comparing this HKEY_CURRENT_USER key to the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server key, as Figure 5.12 shows.



Setup
Svchost
Terminal Server
Compatibility
Applications
MSIEXEC
Install
Software
Time Zones
Type 1 Installer

Name	Type	Data
(Default)	REG_SZ	(value not set)

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software

Figure 5.12: The location for mirrored registry keys.

As you see in Figure 5.12, registry mapping hasn't occurred, so you might need to perform the registry mapping yourself (if you determine that you need to preconfigure HKEY_CURRENT_USER settings for your users). In case you do, you should create a .REG file from HKEY_CURRENT_USER\Software\SoftwareCo\WorkGroup now.

You now launch WorkGroup to determine whether it will run under the installer account. Let’s assume that you don’t encounter any problems. You’re able to connect to the database, read and enter data, and perform queries. So now you do the same test under a non-administrative test account. Once again, the application launches without a problem, but when you try to add a word to the custom dictionary, the program crashes—the test account doesn’t have write access to the USER.LEX file stored on the C drive. You need to attempt to compensate for this shortcoming.

First, test whether WorkGroup will let you move the USER.LEX file to a new location. Logged on as the installer account once again, you copy the file from C:\Program Files\WorkGroup to H:\WorkGroup, and change the UserDicPath value under HKEY_CURRENT_USER to H:\WorkGroup. Now launch WorkGroup and add a new entry to the dictionary. By checking file timestamps, you can confirm that the word was added to the copy on the H drive, so the change was successful.

To replicate this change to all users on the terminal server, you’ll need to perform two tasks:

- Change the UserDicPath value for the each user.
- Copy the USER.LEX file to the ROOTDRIVE when each user logs on.

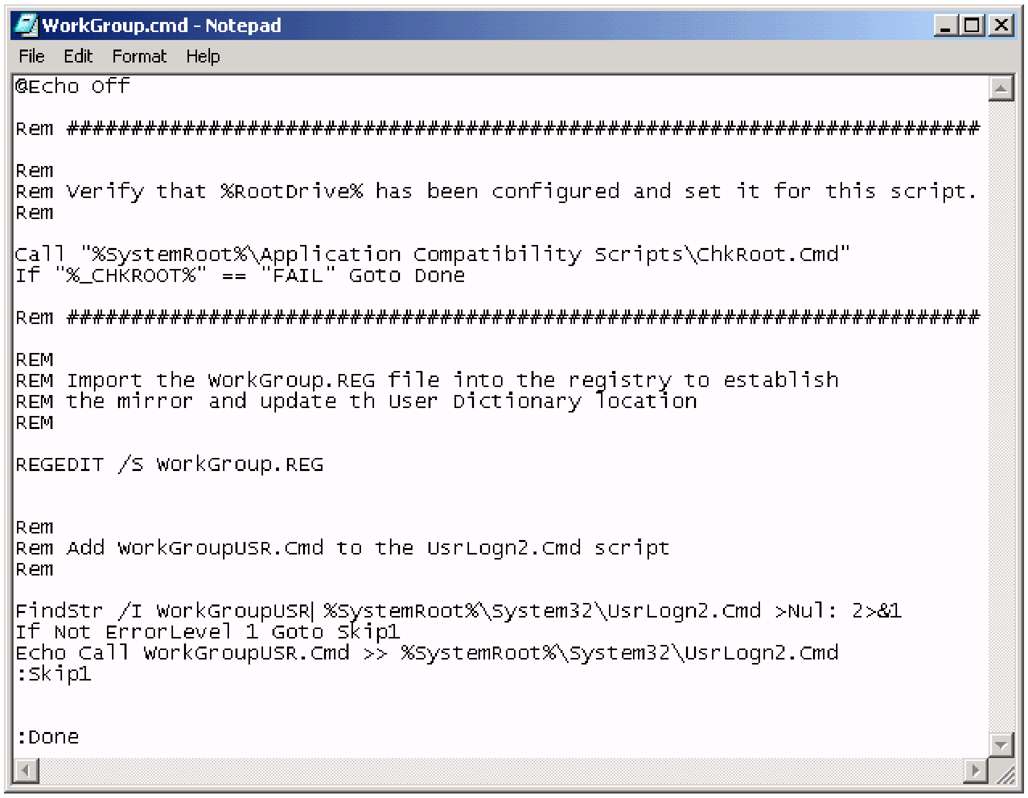
You should be able to make the registry change through registry mapping, but you’ve already discovered that you’ll have to set up this mapping manually. The file copy will require an application compatibility script. Let’s take each of these tasks in turn.

To set up registry mapping, edit the .REG file you created from HKEY_CURRENT_USER by right-clicking the file and selecting Edit. Use the Replace command to change all instances of HKEY_CURRENT_USER\Software to HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software.

Also, find the UserDicPath value and change its data to H:\WorkGroup in the .REG file. Save the updated file to C:\WINNT\Application Compatibility Scripts\Install. You’ll confirm that this new setting gets copied from HKEY_LOCAL_MACHINE to HKEY_CURRENT_USER for the test account after we take care of the application compatibility script.

To accomplish the file copy, you need to write a pair of application compatibility scripts—one installation script and one logon script. For this application, both scripts will be very simple. To copy the file to the proper location, you’ll need to have a ROOTDRIVE defined, so the installation script should confirm that this drive has been established. Next, the installation script should import the .REG file that you created to set up registry mapping and add a call to the application compatibility script logon script to USRLOGN2.CMD. Figure 5.13 shows an example application compatibility script installation script.

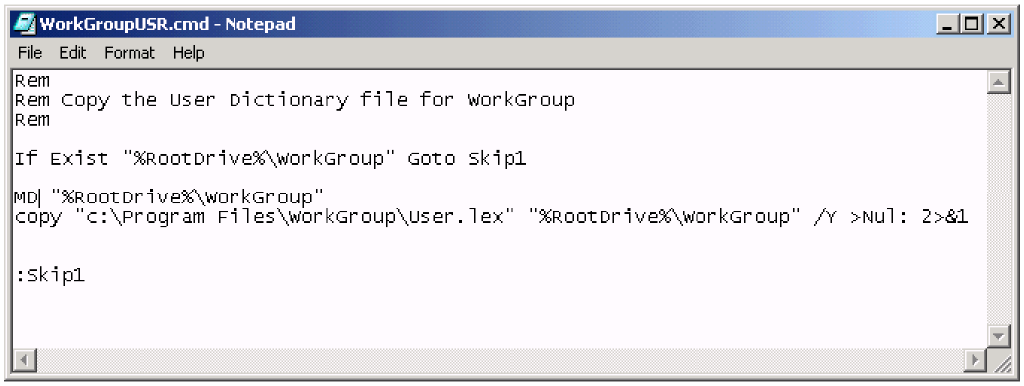
Manually importing the .REG file and editing USRLOGN2.CMD yourself would be easy, but by using an application compatibility script installation script, we make the process easily repeatable when we install the application on the production terminal server. Make sure that you maintain a central location for all application compatibility scripts that you create so that you can replicate them onto new servers when you build them or install new applications.



```
@Echo off

Rem #########################################################################

Rem
Rem Verify that %RootDrive% has been configured and set it for this script.
Rem

Call "%SystemRoot%\Application Compatibility Scripts\ChkRoot.Cmd"
If "%_CHKROOT%" == "FAIL" Goto Done

Rem #########################################################################

REM
REM Import the WorkGroup.REG file into the registry to establish
REM the mirror and update th User Dictionary location
REM

REGEDIT /S WorkGroup.REG


Rem
Rem Add WorkGroupUSR.Cmd to the UsrLogn2.Cmd script
Rem

FindStr /I WorkGroupUSR %SystemRoot%\System32\UsrLogn2.Cmd >Nul: 2>&1
If Not ErrorLevel 1 Goto Skip1
Echo Call WorkGroupUSR.Cmd >> %SystemRoot%\System32\UsrLogn2.Cmd
:Skip1


:Done
```

Figure 5.13: An example application compatibility script installation script.

Before running the installation script, you need to write the application compatibility script logon script that will perform the file copy for each user. This script will be called every time a user logs on, but it should copy the file only if the file doesn't already exist in the user's ROOTDRIVE, so be sure to include logic in your scripts to check this. This script file should be saved in C:\WINNT\Applicaiton Compatibility Scripts\Logon. Figure 5.14 shows an example application compatibility script logon script.



```
Rem
Rem Copy the User Dictionary file for WorkGroup
Rem

If Exist "%RootDrive%\WorkGroup" Goto Skip1

MD "%RootDrive%\WorkGroup"
copy "c:\Program Files\WorkGroup\User.lex" "%RootDrive%\WorkGroup" /Y >Nul: 2>&1


:Skip1
```

Figure 5.14: An example application compatibility script logon script.

Now that all the pieces are in place, execute the application compatibility script installation script for WorkGroup. You should now test the application compatibility script process by logging on as your test account. Be sure to delete this account's profile—both roaming and local—before logging on so that you have a clean testing environment.

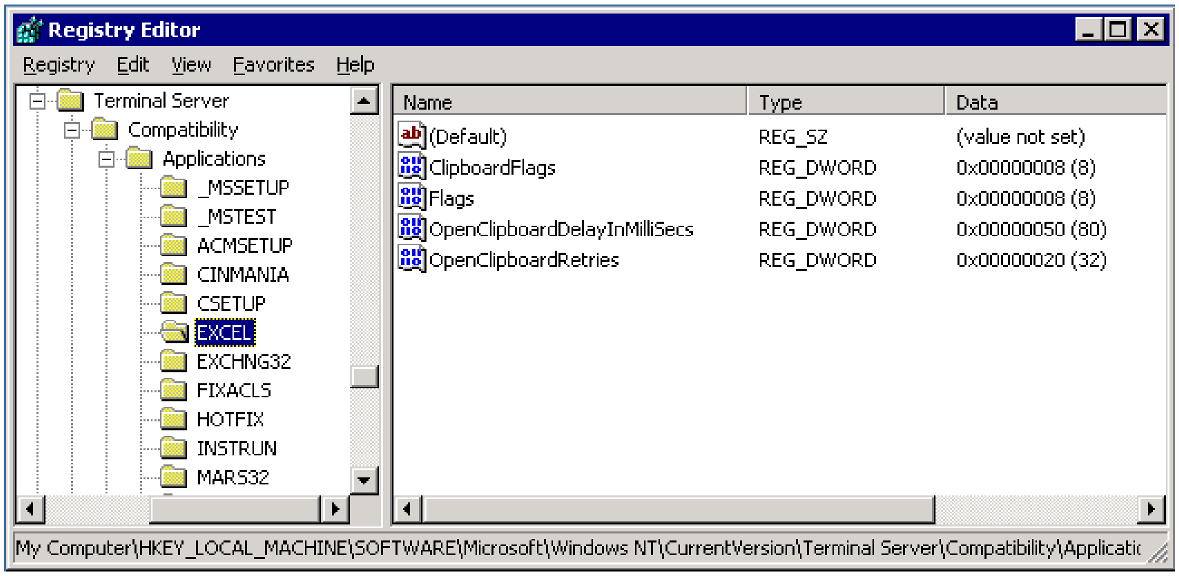
After logging on, first look in the test account's home directory to confirm that the USER.LEX file was created in the WorkGroup subdirectory. If not, go back and confirm that your application compatibility script installation script correctly added the call command to USRLOGN2.CMD, then debug your application compatibility script logon script.

Next, launch WorkGroup, and confirm that all registry keys, especially the modified UserDicPath value, were copied into HKEY_CURRENT_USER for the test account. Add a word to the custom dictionary for the test account, and confirm that the word was added to the copy in the home directory.

After all of this process has been tested, you should also perform a test on WorkGroup by running it under several test accounts simultaneously. If you're satisfied with its performance, you're ready to repeat the installation on your production server.

Terminal Services Compatibility Flags

When you install an application, Terminal Services creates a compatibility flag registry key, which Figure 5.15 shows, that instructs Terminal Services about which type of program the application is (MS-DOS, 16-bit, 32-bit). If you're installing a legacy application that will not run on Terminal Services, you can change this flag so that Terminal Services makes adjustments when the application is launched.



Name	Type	Data
(Default)	REG_SZ	(value not set)
ClipboardFlags	REG_DWORD	0x00000008 (8)
Flags	REG_DWORD	0x00000008 (8)
OpenClipboardDelayInMilliSecs	REG_DWORD	0x00000050 (80)
OpenClipboardRetries	REG_DWORD	0x00000020 (32)

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Compatibility\Applicati

Figure 5.15: The compatibility flags registry values.

The flags that can be set here affect memory allocation and registry and INI file mapping. They can also be used to instruct Terminal Services to return the %USERNAME% value whenever the application queries the %COMPUTERNAME%. This feature can come in handy if the application uses the hostname of the computer as a unique identifier. Typically, only older legacy applications require adjustments.

If you have a program that will not run under Terminal Services, read the Microsoft article "Terminal Server Registry Settings for Applications" at http://support.microsoft.com/default.aspx?scid=kb;[LN];186499.

Installing Applications Through Group Policy

If you are managing a large terminal server farm or need the ability to expand your farm quickly, you will want to use an automated software installation technology. In an AD environment you can use Group Policy to assign applications to your servers. When the server boots, the machine policies are processed and any applications that are assigned will be installed.

To use Group Policy to install applications, your applications must be in MSI format. Group Policy also supports a script format called ZAP, which can trigger a non-MSI installer. However, the use of this format is not recommended with terminal servers because such applications do not support advertising, so your users may not receive necessary HKEY_CURRENT_USER registry keys.

You can repackage software into the MSI format by using a third-party utility such as Wise for Windows Installer. Be sure to thoroughly test the application on Terminal Services before and after repackaging to determine whether any modifications need to be made for terminal server compatibility.

The basic steps required for Group Policy-based software installation are:

1. Create a network share to store your MSI packages.
2. Create Administrative Installations of your MSI packages and place them on the share.
3. Add the packages to a GPO that applies to the terminal server computers in AD.
4. Modify the permissions on the packages in the GPO if you want to filter which terminal servers receive each package.
5. Reboot the terminal servers.

Create a Share

To assign or publish an application via Group Policy, you need a central location to reference for the application source files. The path to this location must be resolvable and accessible by all computers to which the policy applies. The easiest way to accomplish this is to create a share on a file server and copy the source files to it.

Make sure that the machine accounts of your terminal servers have read and execute rights on the share. The Authenticated Users and Everyone groups both include machine accounts.

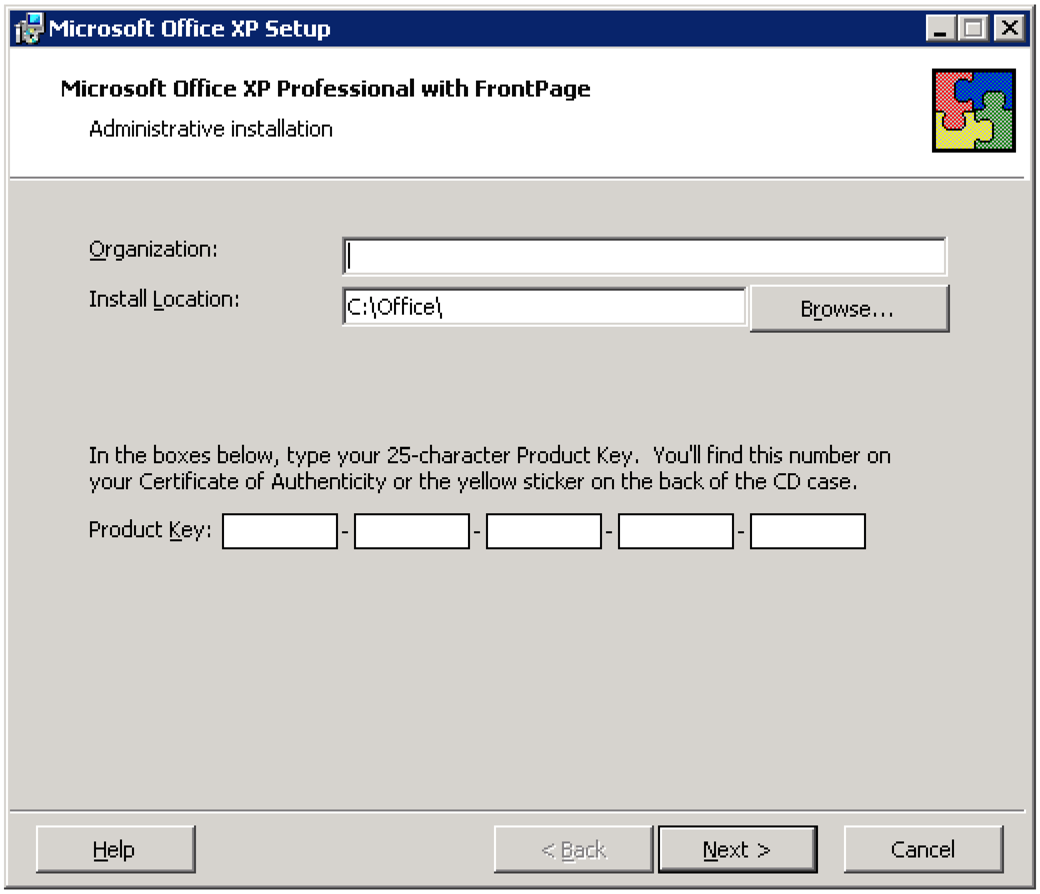
Create Administrative Installations

MSI packages typically come with all of the files needed for the application compressed into CAB files. To optimize the installation of the software, uncompress the files in advance by creating an Administrative Installation.

To create an Administrative Installation, execute the Windows Installer Service with a “/a” switch, and specify the path and name of the MSI package you want to uncompress:

```
msiexec /a d:\proplus.msi
```

A wizard, like the that Figure 5.16 shows, should appear asking you for the destination directory for the Administrative Installation. If the application requires a license key for installation, the wizard will prompt you for this as well. The key is then encrypted into the Administrative Installation so that installs performed from that source will not be prompted for the key.


Microsoft Office XP Setup
Microsoft Office XP Professional with FrontPage
Administrative installation

Organization:

Install Location: C:\Office\

In the boxes below, type your 25-character Product Key. You'll find this number on your Certificate of Authenticity or the yellow sticker on the back of the CD case.

Product Key: - - - -

Browse...
Help
< Back
Next >
Cancel

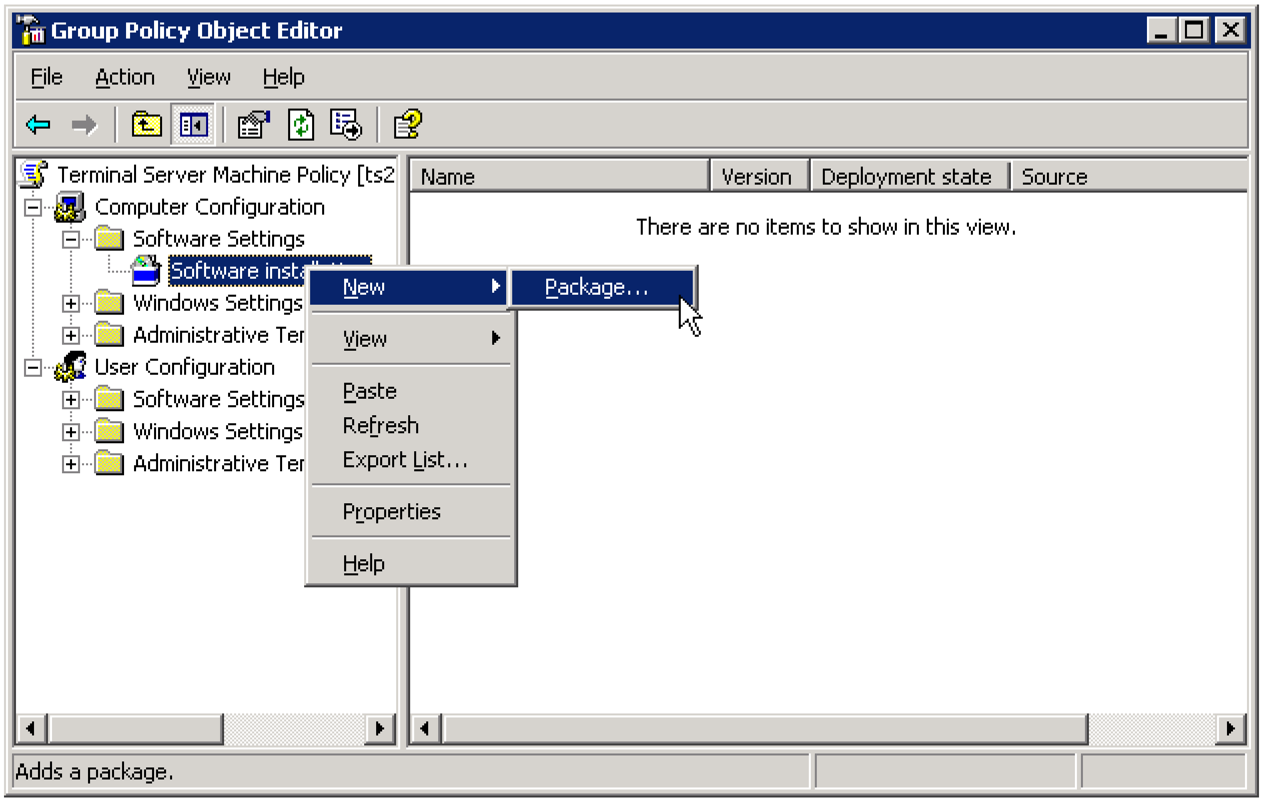
Figure 5.16: An Administrative Installation wizard.

Once the Administrative Installation is complete, copy the new source files to your share.

Add the Packages to a GPO

In Chapter 4, we created an OU for terminal servers and linked three GPOs to that OU. One of the GPOs was the Terminal Server Machine Policy. This GPO contained the machine configuration for your terminal servers, so it already is configured to apply to your terminal servers. You can add your software packages to this GPO, and they will be installed on all computers in the TerminalServers OU.

To add a package to a GPO, edit the policy and expand Computer Configuration, Software Settings. Right-click *Software installation*, and select New, Package (see Figure 5.17).



There are no items to show in this view.

Adds a package.

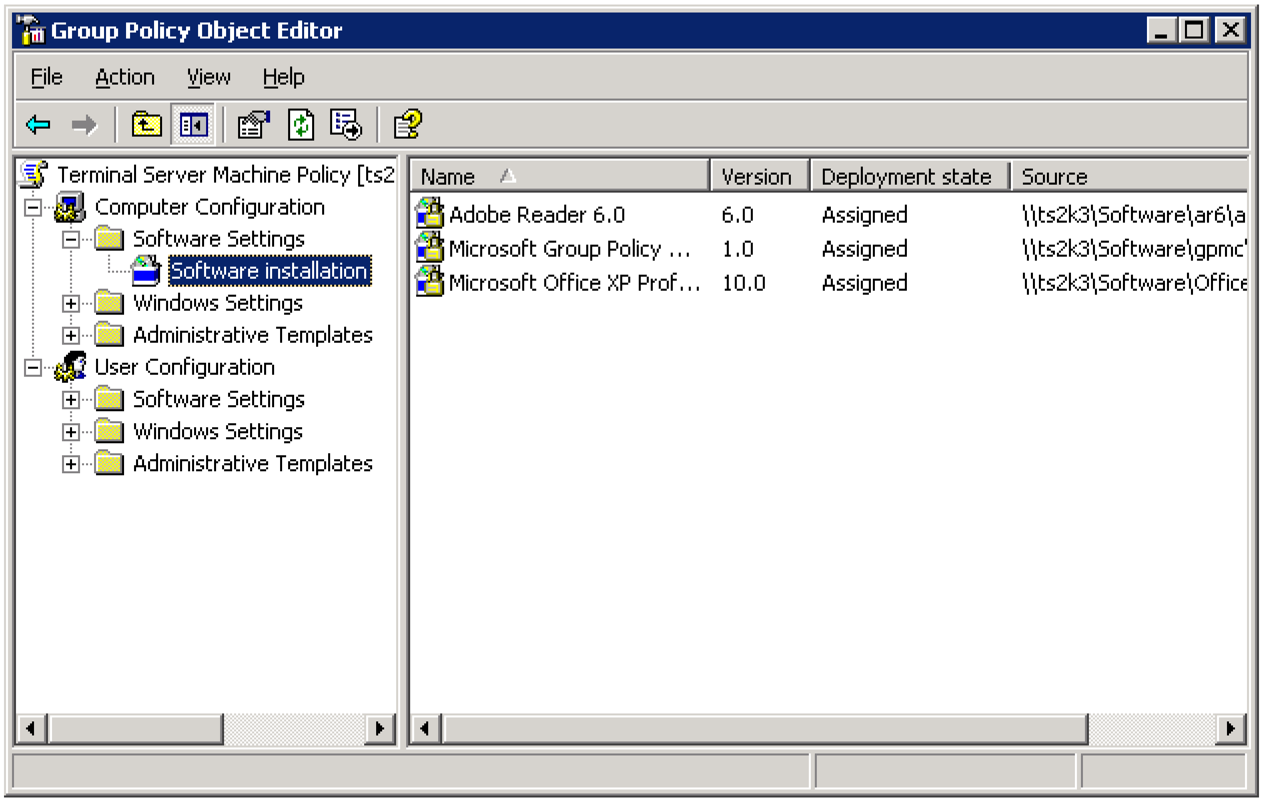
Figure 5.17: Adding a software package to a GPO.

You will be prompted for the MSI package that you want to add. Navigate to your share and select the MSI file that you want to install. You will then be asked whether you want to assign the package or open the Advanced Settings interface. In most cases, you can select Assign and accept all of the default options. If, however, you need to specify a transform to be applied during the installation, select Advanced.

A transform (MST file) specifies options or makes changes to the default settings of a Windows Installer package (MSI file). You can create transforms by using the Microsoft Office Custom Installation Wizard or with a third-party utility.

Filtering Applications

You can use a single GPO that applies to all of your terminal servers and still filter which applications are installed on each server by using security filters on the packages in the GPO. Figure 5.18 shows a GPO that contains three software packages—Adobe Acrobat Reader 6, Office XP, and the Microsoft Group Policy Management Console.

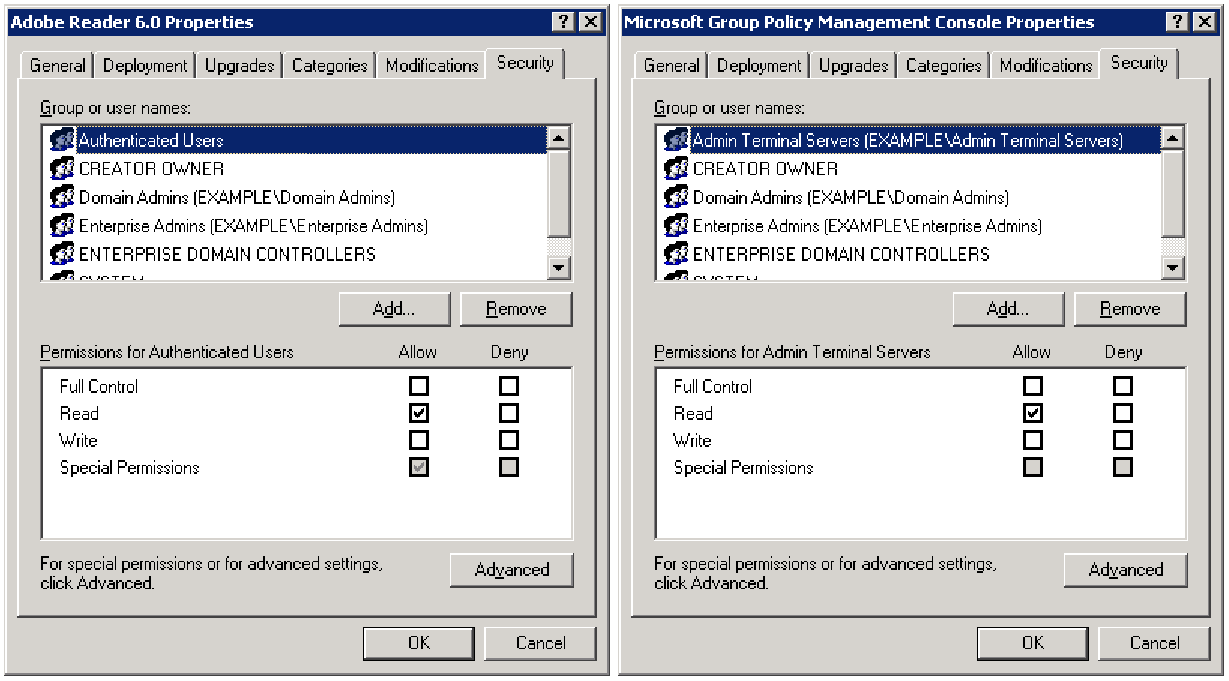


Name	Version	Deployment state	Source
Adobe Reader 6.0	6.0	Assigned	\\ts2k3\Software\ar6\a
Microsoft Group Policy ...	1.0	Assigned	\\ts2k3\Software\gpmc'
Microsoft Office XP Prof...	10.0	Assigned	\\ts2k3\Software\Office

Figure 5.18: A Group Policy that has three software packages applied.

Let's assume that you want to install Acrobat Reader and Office XP on all terminal servers to which the GPO applies, but you only want to install the console on the terminal servers that administrators use. By default, the packages will inherit security settings from the GPO, so any computers that have read permission on the GPO will also have read permission on the package, and will therefore install the software during boot up.

You can change the permissions on the package and limit the ability to read it to only a specific group of computers. By doing so, all computers in the OU will process the policy, but only those with read permission on the package will install it. Figure 5.19 shows the default permissions on a package and the permissions after they have been modified so that only the Admin Terminal Servers group can read it.



Adobe Reader 6.0 Properties

General | Deployment | Upgrades | Categories | Modifications | Security

Group or user names:

- Authenticated Users
- CREATOR OWNER
- Domain Admins (EXAMPLE\Domain Admins)
- Enterprise Admins (EXAMPLE\Enterprise Admins)
- ENTERPRISE DOMAIN CONTROLLERS
- SYSTEM

Add... Remove

Permissions for Authenticated Users	Allow	Deny
Full Control	☐	☐
Read	☑	☐
Write	☐	☐
Special Permissions	☑	☐

For special permissions or for advanced settings, click Advanced. Advanced

OK Cancel

Microsoft Group Policy Management Console Properties

General | Deployment | Upgrades | Categories | Modifications | Security

Group or user names:

- Admin Terminal Servers (EXAMPLE\Admin Terminal Servers)
- CREATOR OWNER
- Domain Admins (EXAMPLE\Domain Admins)
- Enterprise Admins (EXAMPLE\Enterprise Admins)
- ENTERPRISE DOMAIN CONTROLLERS
- SYSTEM

Add... Remove

Permissions for Admin Terminal Servers	Allow	Deny
Full Control	☐	☐
Read	☑	☐
Write	☐	☐
Special Permissions	☐	☐

For special permissions or for advanced settings, click Advanced. Advanced

OK Cancel

Figure 5.19: Filtering security on a software package.

For this security filter to work, you will need to create a Domain Security group—Admin Terminal Servers, for example—and place the servers you want to receive the console into that group.

To modify the permissions of a package, you have to break inheritance on the GPO's permissions. To do so, click Advanced, and clear the *Allow inheritable permissions* check box. You can then choose to copy the existing permissions and use them as a starting point for your modifications.

Reboot the Terminal Servers

A major limitation of Group Policy-based software installation is that it only is processed during boot up. If you add a new package, you must reboot your servers to install it. Using GPO-based installation is still useful in that new servers that are built and placed in the Terminal Servers OU will automatically install the required packages, saving you the time and effort of installing them on every new server.

You can also use GPOs to upgrade and uninstall software. Check out Darren MarElia's *The Definitive Guide to Windows 2000 Group Policy* (Realtimepublishers.com) available from a link at http://www.realtimepublishers.com for an in-depth look at Group Policy-based software management.

There are several software management systems available that can install software on a scheduled basis without requiring a reboot. Microsoft Systems Management Server and Citrix MetaFrame XPe both have the ability to schedule installations and manage the software outside the Group Policy process.

Deploying Applications to End Users

If you're using Terminal Services as a desktop replacement, deploying a new application to your users is as simple as adding an icon for the program to the All Users Start menu on your terminal server. If, however, you want to deploy a single application in an application service provider (ASP) model, your deployment method will be determined by the type of terminal server client you're using.

If your users access Terminal Services applications through the local Remote Desktop Connection client, you'll need to create an RDP file that contains the server name and initial application path for the new application, then distribute it to your users.

If you want to deploy an application through the Remote Desktop Web Connection Client so that your users will go to a URL to launch the program, you should reference the documentation that Microsoft provides with the client to customize the Web page that hosts the application. The Web Connection Client is very powerful and customizable, so with a little HTML and ActiveX scripting, you can create a very robust portal for your ASP-based applications.

Summary

In this chapter, we explored the mechanisms that Terminal Services use to host applications for simultaneous users, even when the application is not designed to handle them. I examined the Terminal Services logon script process and administrative modes, and explained application compatibility scripts. We defined the ROOTDRIVE and how this concept can be modified to take advantage of network home directories and WS2K3's enhanced drive-mapping capabilities. This information enables you to install applications on your terminal server, test them for multi-user compatibility, and write an application compatibility script if necessary.

Finally, we walked through the process of using Group Policy to automatically install software on your servers. This functionality will help streamline the process of building new servers, allowing you to scale your terminal server farm quickly and easily.

In Chapter 6, we'll explore a very hot topic—patch management. With the number of viruses and worms over the past few years, it is critical that you keep your servers up to date with the latest security updates and software patches. I'll be covering some options that make this process easy and automatic.