



realtimepublishers.com[™]

The Definitive Guide[™] To

Windows 2000 Terminal Services

NEW MOON) SYSTEMS[™]

Greyson Mitchem

| | |
|---|----|
| Chapter 1: Introduction to Windows Terminal Services..... | 1 |
| Terminal Services History..... | 1 |
| Implementing Win2K Terminal Services | 2 |
| Desktop Replacement..... | 2 |
| Elimination of End-Node Support..... | 3 |
| Rapid Deployment of New or Upgraded Software | 3 |
| Reduction in Power Consumption..... | 3 |
| Added Security..... | 3 |
| Limited Adaptability to One-Off Applications | 3 |
| Reduction in User Settings' Personalization | 3 |
| Limited Sound and Multimedia Capabilities | 4 |
| Increased Initial Deployment Costs | 4 |
| Remote Access | 4 |
| ASP..... | 4 |
| Remote Administration | 5 |
| The Client/Server Processing Model..... | 6 |
| The Boot and Logon Process | 8 |
| The Terminal Services Protocols | 10 |
| RDP | 10 |
| RDP versus ICA | 11 |
| Win2K Terminal Services Licensing | 11 |
| Terminal Services Licensing Service Enhancements..... | 13 |
| MetaFrame Licensing..... | 15 |
| Summary | 15 |
| Chapter 2: Installing and Configuring Terminal Services | 16 |
| Remote Administration Mode..... | 16 |
| Application Server Mode | 18 |
| Hard Disk Configuration..... | 18 |
| Physical Memory..... | 20 |
| Processors..... | 22 |
| Load Balancing | 22 |
| Putting It All Together | 23 |
| System Tuning For Terminal Services..... | 24 |

| | |
|---|----|
| Adjusting Server Settings..... | 25 |
| Setting up a Terminal Services License Server..... | 27 |
| Activating the License Server | 29 |
| Installing Licenses..... | 31 |
| How Does the Terminal Server Find the License Server? | 32 |
| Win2K Domain | 32 |
| NT 4.0 Domain..... | 33 |
| Workgroup Environment..... | 33 |
| Installing the Terminal Services Client..... | 33 |
| Setup.exe-Based Installation | 34 |
| Windows Installer-Based Installation | 34 |
| Handheld Device Installation..... | 35 |
| Configuring Client Connections..... | 35 |
| The Microsoft Terminal Services Advanced Client..... | 37 |
| Windows-Based Terminal Clients | 38 |
| Third-Party Clients..... | 38 |
| Summary | 39 |
| Chapter 3: Deploying and Managing Applications..... | 40 |
| Application Compatibility Mechanisms..... | 40 |
| Terminal Services Logon Scripts | 40 |
| USRLOGON.CMD | 41 |
| Example 1..... | 44 |
| Example 2..... | 44 |
| Additional Administrative Scripts..... | 46 |
| Application Installation..... | 46 |
| Registry Mapping..... | 47 |
| INI File Mapping..... | 48 |
| Install and Execute Modes | 49 |
| Application Compatibility Scripts..... | 49 |
| Examples of Terminal Services Application Installations | 51 |
| Simple Installation..... | 51 |
| Custom Installation | 51 |
| Application Compatibility Script Installation | 55 |

| | |
|--|----|
| Undocumented Application Installations..... | 56 |
| Real World Example..... | 57 |
| Terminal Services Compatibility Flags..... | 62 |
| Deploying Applications..... | 63 |
| Managing the Application Life Cycle..... | 63 |
| Summary..... | 64 |
| Chapter 4: Terminal Services Administration..... | 65 |
| Terminal Server Access Requirements..... | 65 |
| Log On Locally..... | 65 |
| Permissions on RDP..... | 67 |
| RDP Access Levels..... | 68 |
| Allow Logon to Terminal Services..... | 69 |
| Controlling Session Behavior..... | 70 |
| Session Timeouts..... | 71 |
| Environment Settings..... | 72 |
| Remote Control..... | 74 |
| UI Settings..... | 75 |
| Group Policy and Software Installation..... | 77 |
| Managing Group Policy..... | 77 |
| Standard Group Policy Processing Order..... | 78 |
| Loopback Group Policy Processing Order..... | 80 |
| Enabling Loopback..... | 81 |
| Home and Profile Directories..... | 82 |
| Terminal Services Profile Path..... | 82 |
| Terminal Services Home Directories..... | 84 |
| Managing and Supporting User Sessions..... | 84 |
| Terminal Services Manager..... | 85 |
| Remote Control..... | 87 |
| Registry Editing..... | 87 |
| Command-Line Utilities..... | 88 |
| Managing Printing..... | 88 |
| Summary..... | 90 |
| Chapter 5: Performance, Capacity Planning, and Availability..... | 91 |

| | |
|--|-----|
| Capacity Planning | 91 |
| The Testing Environment..... | 91 |
| The Tools..... | 92 |
| RoboClient (ROBOCLI.EXE) | 92 |
| RoboServer (ROBOSRV.EXE) | 93 |
| QueryIdle (QIDLE.EXE) | 94 |
| SimulatedClient (SMCLIENT.EXE) | 94 |
| Test Scripts..... | 95 |
| The Test Domain..... | 95 |
| Performance Monitoring and Troubleshooting | 95 |
| Hardware Monitors | 95 |
| Network Traffic..... | 96 |
| Scheduled Reboots | 96 |
| Hardware Fault Tolerance..... | 96 |
| System Monitors | 96 |
| Performance Monitor (PerfMon) | 96 |
| Logging and Alerts..... | 99 |
| Task Manager..... | 100 |
| Load Balancing | 101 |
| Installing Load Balancing | 102 |
| Affinity Mode..... | 106 |
| Controlling Load Balancing..... | 106 |
| Application-Level Load Balancing | 107 |
| Terminal Server Maintenance | 107 |
| Summary | 109 |
| Chapter 6: Securing Terminal Services..... | 111 |
| Using Encryption..... | 111 |
| Logon Security | 113 |
| Virus Protection..... | 113 |
| Critical Updates and Internet Explorer Maintenance..... | 113 |
| Access Through a Firewall..... | 113 |
| Implementing Terminal Services in an Extranet Environment..... | 114 |
| Application-Layer Security..... | 115 |

| | |
|---|-----|
| File and Registry Security | 117 |
| Remote Access | 117 |
| Advantages of Using Terminal Services for Remote Access | 118 |
| Summary | 118 |
| Afterword: The Future of Terminal Services..... | 119 |
| RDP 5.1 | 119 |
| Remote Desktop | 120 |
| Remote Assistance | 121 |
| Windows .NET Server | 121 |
| Third-Party Products | 122 |
| Summary | 122 |
| Appendix A: Terminal Services Clients..... | 133 |
| Configure a Workstation to Act Like a Thin Client..... | 133 |
| Appendix B: Important URLs | 135 |
| Appendix C: Registry Changes | 136 |
| Appendix D: Script Reference | 137 |
| USRLOGON.CMD | 137 |
| TSSHUTDOWN Wrapper..... | 138 |
| Maintenance Reboot Script..... | 139 |
| Appendix E: Terminal Services Command-Line Reference..... | 141 |
| Change User | 141 |
| Change Logon | 141 |
| Query Terminal Servers | 141 |
| Query Session..... | 141 |
| Query User | 142 |
| Query Process..... | 142 |
| Logoff..... | 143 |
| Message..... | 143 |
| Reset Session..... | 143 |
| Shadow | 144 |
| Terminal Services Profile..... | 144 |
| Terminal Services Shutdown | 144 |
| Appendix F: SMCLIENT Scripting Language Reference | 146 |

| | |
|-----------------|-----|
| Connect..... | 146 |
| Logoff..... | 146 |
| Disconnect..... | 146 |
| Start | 147 |
| Sendoutput..... | 147 |
| Senddata | 147 |
| Sendtext..... | 147 |
| Sleep..... | 148 |
| Clipboard..... | 148 |
| Loop | 148 |
| Check..... | 148 |
| Call | 148 |
| Job | 149 |
| Comments..... | 149 |

Copyright Statement

© 2001 Realtimepublishers.com, Inc. All rights reserved. This site contains materials that have been created, developed, or commissioned by, and published with the permission of, Realtimepublishers.com, Inc. (the "Materials") and this site and any such Materials are protected by international copyright and trademark laws.

THE MATERIALS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. The Materials are subject to change without notice and do not represent a commitment on the part of Realtimepublishers.com, Inc or its web site sponsors. In no event shall Realtimepublishers.com, Inc. or its web site sponsors be held liable for technical or editorial errors or omissions contained in the Materials, including without limitation, for any direct, indirect, incidental, special, exemplary or consequential damages whatsoever resulting from the use of any information contained in the Materials.

The Materials (including but not limited to the text, images, audio, and/or video) may not be copied, reproduced, republished, uploaded, posted, transmitted, or distributed in any way, in whole or in part, except that one copy may be downloaded for your personal, non-commercial use on a single computer. In connection with such use, you may not modify or obscure any copyright or other proprietary notice.

The Materials may contain trademarks, services marks and logos that are the property of third parties. You are not permitted to use these trademarks, services marks or logos without prior written consent of such third parties.

If you have any questions about these terms, or if you would like information about licensing materials from Realtimepublishers.com, please contact us via e-mail at info@realtimepublishers.com

Chapter 1: Introduction to Windows Terminal Services


Windows terminal services has been around since the early 1990s when Citrix Systems introduced WinView, a multi-user product based on the OS/2 operating system (OS). Many people view Windows terminal services as the result of computing architecture coming full circle, citing that server-based computing was the basis of the mainframe-computing model and pointing out that UNIX and X Windows provide a similar GUI-based server-centric computing model.

Although mainframes and UNIX systems still hold their places in the corporate computing world, Windows terminal services provides much greater flexibility for providing access to Win32 applications. Windows terminal services lets you provide access to an application without having to install the application on every computer in your environment. This book will focus on the native capabilities of Windows 2000 (Win2K) Terminal Services, its implementation, configuration, maintenance, and application integration.

Terminal Services History

Since its introduction, Windows terminal services has stayed in step with the evolution of Windows itself. After the success of Microsoft Windows 3.51, Citrix, a small Florida-based company, licensed the OS code from Microsoft to adapt the code for a multi-user application.

In 1995, Citrix introduced WinFrame, which provides terminal server users access to the full Windows 3.51 user experience. WinFrame brought Windows terminal services into the mainstream, and many large corporations began making it a standard for call centers, customer service representatives, and production-floor terminals. 1995 also saw the advent of the Windows terminal, or thin-client device, which were the first solid-state devices designed to provide access to a Windows terminal server. Wyse and Tektronix were the major players in this arena.

 A *Windows terminal*, or *thin-client device*, is a solid-state device (it has no moving parts) that has its OS burned into ROM. These devices are exclusively used to communicate with terminal servers, UNIX servers, or mainframes and have no native functionality of their own. They typically run Windows CE, but there are some models that run embedded Linux.

1996 brought another revolution to the Windows computing world—Windows NT 4.0. Once again, terminal services stayed in-stride with the Windows evolution through the introduction of two products—Windows NT Server 4.0, Terminal Server Edition (WTS), which uses Citrix MultiWin technology at its core, and Citrix MetaFrame. Both of these products came out in 1998. Unfortunately for terminal services, this period also saw an explosion of Win32 applications being ported from the Windows 3.1 and Windows 95 platforms. Many of these applications weren't designed for the specific needs of NT 4.0 or terminal services, so many of them don't work in a multi-user mode. This limitation, coupled with the fact that WTS is a separate OS rather than an NT 4.0 add-on, is the reason that many IT professionals still regard terminal services as difficult to deploy and limited in its application support. However, as Windows has matured, so have the applications being written for it, and today most programs will run in a terminal server environment with little or no modification.



Throughout this book, for generic references to the technology, I'll use terminal services. I'll refer to Windows 2000 Terminal Services as Win2K Terminal Services and NT 4.0, Terminal Server Edition as WTS.

Today, Win2K provides terminal services as a native component of every flavor of Windows Server—Win2K Server, Win2K Advanced Server, and Win2K Datacenter Server. This inclusion has given Win2K Terminal Services a hold in almost all implementations of Win2K. Citrix has also continued to improve and develop its terminal services products with the release of MetaFrame 1.8 for Windows 2000 and MetaFrame XP.

Although the evolution of terminal services is due mostly to the work of Citrix, Win2K Terminal Services' native capabilities are very robust and can be used for many implementations without incorporating MetaFrame. In addition, other third-party vendors are now releasing application service provider (ASP) management systems engineered around Win2K Terminal Services. If you're a large corporation looking to replace your users' desktop computers with Windows terminals or considering terminal services in a complex ASP model, you'll certainly want to consider third-party tools in your deployment plan. However, as I previously mentioned, this book will focus on the native capabilities of Win2K Terminal Services.

Implementing Win2K Terminal Services

In today's corporate world, IT professionals spend a great deal of time performing two major tasks—maintaining servers and deploying applications to end-users' desktop computers. Both of these tasks can be greatly streamlined with the implementation of Win2K Terminal Services.

There are four basic models for utilizing Win2K Terminal Services:

1. Desktop Replacement—Remove the standard Wintel PC from the user, and replace it with a thin-client device.
2. Remote Access—Provide users access to either a complete desktop environment or individual applications from a remote location over either a WAN link or RAS connection.
3. ASP—Provide access to individual applications to users at their regular workstation without installing the applications locally on users' PCs.
4. Remote Administration—Use Win2K Terminal Services as a remote control interface for Win2K servers.

Desktop Replacement

At its most pervasive and complete implementation, Win2K Terminal Services can allow an IT department to completely eliminate PCs from users' desks. This model provides many benefits, including elimination of end-node support, rapid deployment of new or upgraded software, reduction in power consumption, and added security. Depending on your corporate IT architecture, desktop replacement can also reduce bandwidth requirements and reduce the need for servers in remote offices.



Elimination of End-Node Support

Without a PC on users' desks, there is no longer any need to visit the workstation to install the OS, install or repair software, assist a user in configuring applications, or replace a defective hard drive. A Windows terminal's OS is burned into ROM, and applications are installed on the terminal servers. So Help desk personnel can provide user assistance via remote control of the terminal server session, and users can replace a damaged device by simply plugging in a new one.

Rapid Deployment of New or Upgraded Software

If you work in a large IT environment, you know how difficult and time consuming deploying software to your users can be. With Win2K Terminal Services desktop replacement, you simply install the new software on your terminal servers and overnight thousands of users will have access to it. There are even third-party utilities that will assist you in deploying software to all your terminal servers at once.

Reduction in Power Consumption

Because thin-client devices have no moving parts and are completely solid-state, they typically consume about 10 percent of the power that a normal Wintel PC consumes. With rising electricity costs today, this reduced consumption can provide a major cost-savings to your company.

Added Security

If a standard PC is stolen, you risk losing important and sensitive data stored on its local hard disk, and you must pay to replace the computer. With desktop replacement, there is no data stored on the end-node device, and the cost of replacement is about half that of a normal PC.

There are many players in the thin-client device market; for example, Wyse Technology's Winterm (<http://www.wyse.com>) and NeoWare System's Eon (<http://www.neoware.com>). These devices can use any embedded OS at their core—WindowsCE, embedded Linux, and so on.

There are, however, some potential drawbacks to the desktop-replacement model, including limited adaptability to one-off applications, reduction in user settings' personalization, and increased initial deployment costs.

Limited Adaptability to One-Off Applications

If you have a very small group of users who need an application, you'll no longer have the freedom to install the application on only those users' desktops. You'll be forced to integrate the application into your Win2K Terminal Services infrastructure.

Reduction in User Settings' Personalization

If your users are accustomed to personalizing their workstations with wallpaper and screen savers or have the ability to install their own software, you'll have a small battle on your hands when they're restricted from performing some of these customizations.

Limited Sound and Multimedia Capabilities

If your applications require sound or multimedia, running them via terminal services might not be a good fit as RDP doesn't support sound and has a difficult time with the numerous screen draws required by animation and multimedia. Third-party enhancements may be used to overcome some of these limitations.

Increased Initial Deployment Costs

If you already have a large user population that all have their own PCs, the initial setup costs of purchasing the thin-client devices and the robust servers needed for terminal servers can seem a little overwhelming. However, in the long term, the reduction in TCO will more than make up for the initial investment. Opening a new office or call center is a perfect opportunity to implement the desktop-replacement model.

Remote Access

If these drawbacks or your corporate culture eliminate desktop replacement as an option, the remote-access model may be a great alternative for you. Most big companies have a large population of remote or nomadic users—telecommuters, executives traveling to satellite offices, and so on. Although laptops provide the ability to work remotely, they don't address the needs of limited-bandwidth connections or remote support. In addition, laptops can be nearly double the cost of a desktop, so your finance department may balk at the thought of providing a laptop for users who only occasionally need remote access to applications.

The remote-access model can provide these users with the ability to access individual applications or even a complete corporate desktop from the Internet (by using the Terminal Services Advanced Client, a Web-based version of the Terminal Services Client) or from their home computers. In addition, the reduced bandwidth requirements of the protocols used for Win2K Terminal Services provide improved performance over running laptop-based applications over a slow-link.

Using Win2K Terminal Services as a portal to the corporate LAN can also shield your network from any viruses that may be on the remote computer.

As with any remote-access strategy, you must make security the top priority when considering the remote-access model. Be sure to take the time to educate your network design engineers in the specific needs of the Win2K Terminal Services protocols. In addition, implement a strategy to prevent the abuse of any changes you implement to accommodate the terminal server network traffic.


ASP

When looking at a large-scale deployment to your users of a vertical application, there are many factors to consider:


- Deployment method (for example, sneaker net, Systems Management Server—SMS, IntelliMirror)
- Workstation system requirements (for example, RAM, disk space, processing power)

- Support and recovery plan in case an installation goes awry
- Bandwidth requirements for client/server or database applications

If the application you're deploying doesn't have complex OLE integration with other applications on the users' desktops, the ASP model might be right for you. Win2K Terminal Services, especially when implemented with the Terminal Services Advanced Client, can give you the ability to provide users the applications they need quickly and easily without touching their workstations.

 Canaveral iQ from New Moon Systems (<http://www.newmoon.com>) and AppScape from SoftBlox (<http://www.softblox.com>) are add-on products for Win2K Terminal Services and MetaFrame, respectively, that will help you deploy applications quickly, track usage patterns, and manage application licensing.

In this model, the application is installed on terminal servers, and users launch it via a client application on their desktops or by using a Web browser. I'll go into the details of this model in Chapter 3.

 While you're in a mixed Win2K and NT environment, the ASP model is also a great way to provide your Help desk and support staff access to Win2K support tools without them needing two workstations.

Remote Administration

This terminal services utilization method is the most basic model, and it's the most commonly used in any Win2K deployment. Since Terminal Services is a built-in component of Win2K servers, you can enable it by simply selecting the associated check box in the Windows Components window of the Add/Remove Programs Control Panel applet, as Figure 1.1 shows.

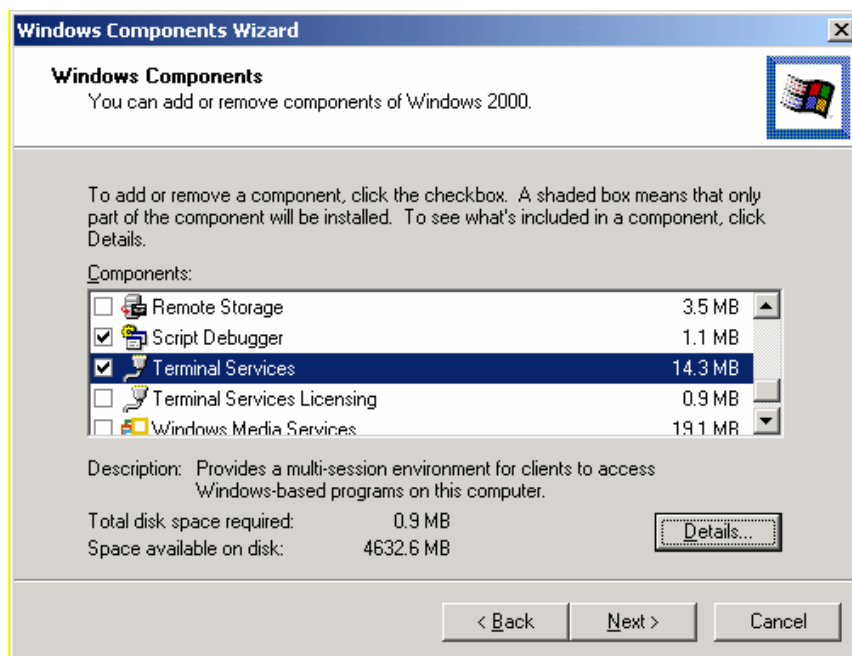


Figure 1.1: Enabling Win2K Terminal Services.

As Figure 1.2 shows, when you enable Win2K Terminal Services, the system asks whether you want to run Terminal Services in *application server mode* (the mode used for all three of the other models) or *remote administration mode*. Unlike application server mode, remote administration mode doesn't require that a user connecting to the server have a terminal services client access license (TSCAL), so you won't need to implement a Win2K Terminal Services licensing server for this model. I'll cover licensing in detail in the "Win2K Terminal Services Licensing" section later in this chapter.

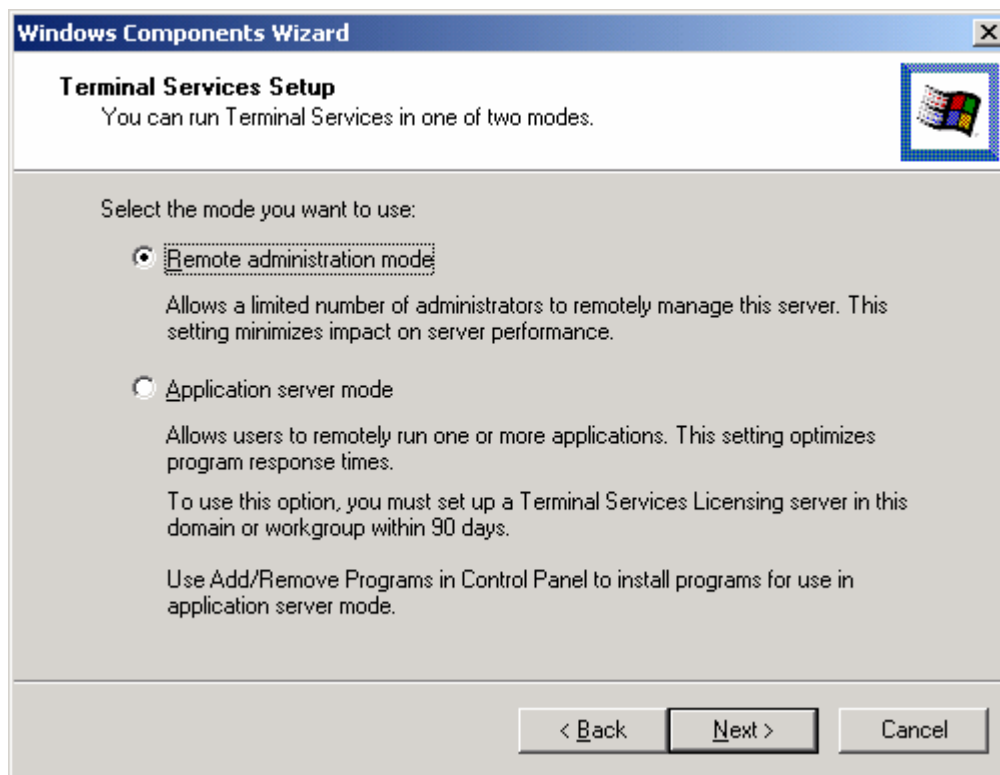


Figure 1.2: Selecting a Terminal Services mode.

Remote administration mode isn't used to provide multiple users access to applications, but rather to let systems administrators remotely control servers. Anything you can do at the console can be done over a Remote Admin connection. This model doesn't add undue load to the server, and it can be enabled on all servers in your environment. Only users with Local Administrator privileges on the server will be able to open a Remote Admin connection.

The Client/Server Processing Model

In traditional Windows-based computing, all processing is done at the client computer, as Figure 1.3 illustrates. Data can be stored on servers in a central location, but when documents are opened or executables are launched, the bits are transferred to the client, loaded into RAM and processed by the desktop's CPU.

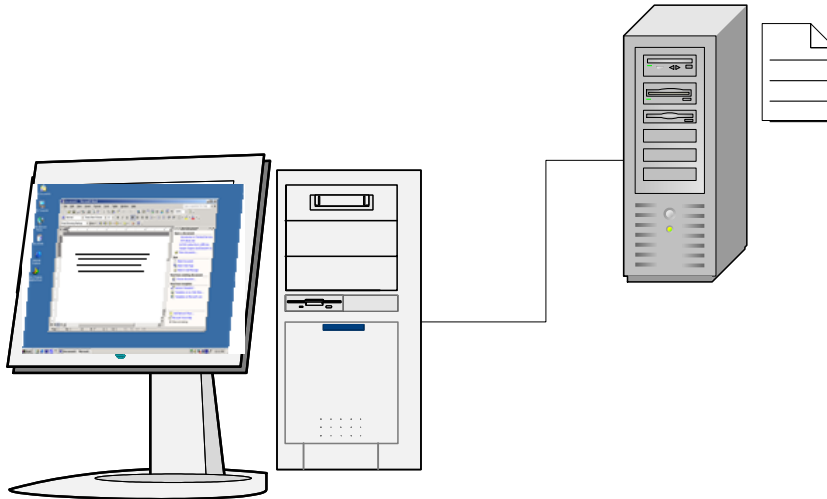


Figure 1.3: In the traditional computing model, data is copied to the client workstation and processing is performed there.

The client/server processing model is quite different than this traditional model. Instead of pulling the data to the client PC, only screen information is streamed to the user, as Figure 1.4 shows. The server handles all the processing.

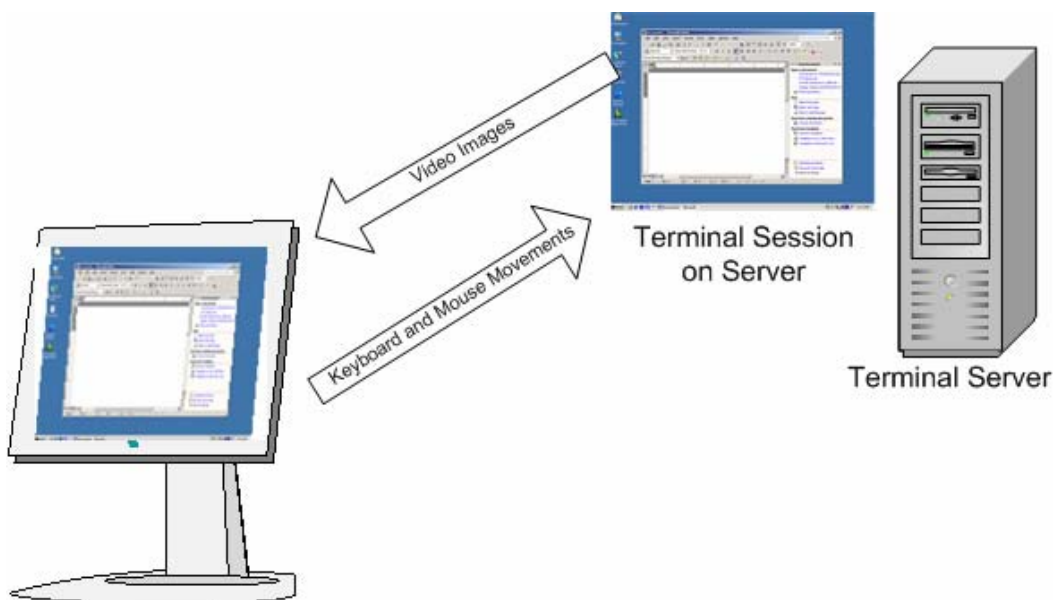


Figure 1.4: In the client/server processing model, all processing is done at the server, and only keyboard, mouse, and video information is sent to and from the client.

This shift in the computing model is accomplished by two mechanisms: MultiWin technology, which was developed by Citrix for WTS, then redesigned as a Windows service for Win2K Terminal Services; and a terminal services protocol, either remote desktop protocol (RDP) for native Microsoft terminal services or Independent Computing Architecture (ICA) for Citrix MetaFrame. (I'll discuss these protocols in detail later in the "The Terminal Services Protocols"

section of this chapter.) MultiWin combined with a terminal services protocol produce the ability for a server to simultaneously run multiple interactive user sessions and for users to interact with these sessions remotely over a network link rather than at a directly connected console.

WTS has a modified kernel to provide its MultiWin capabilities. Because of this modification, you can't install WTS over an existing installation of NT Server 4.0; you must perform a new installation of the OS and reinstall any applications on the server. Win2K Terminal Services gets its MultiWin functionality from a service called TERMSERV.EXE. For both Win2K Terminal Services and WTS, a new layer called SessionSpace is added to the processing model, as Figure 1.5 illustrates. SessionSpace creates a new virtual address space session for each user and loads a separate instance of WIN32K.SYS into each address space session. Each session has its own session ID and communication to and from the end-node device is targeted to its session ID only.

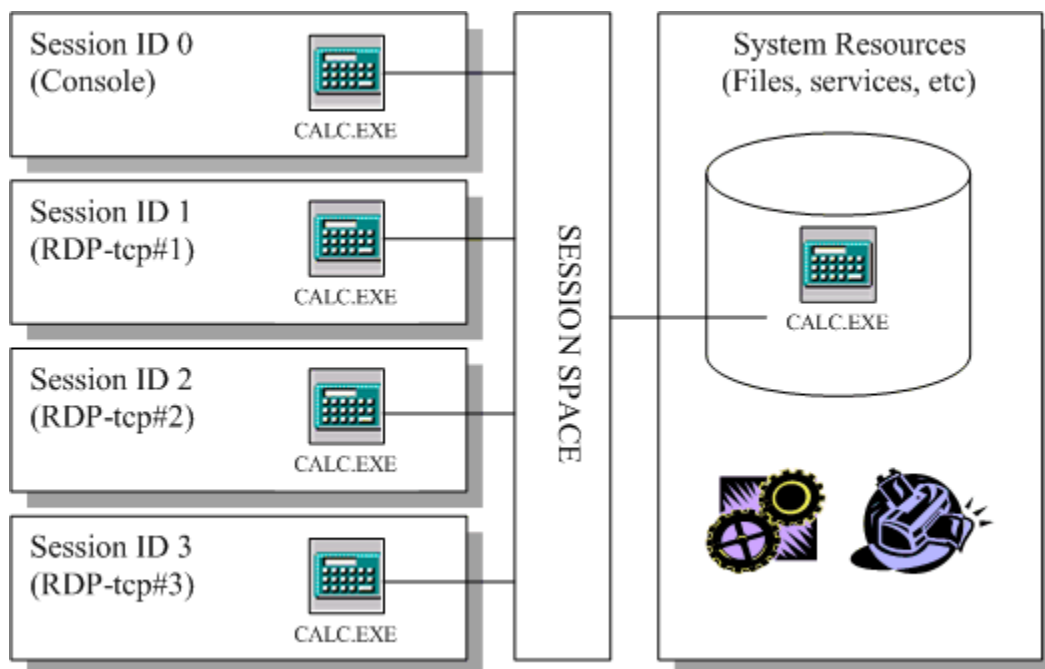


Figure 1.5: The SessionSpace model and example of memory-based code sharing.

Terminal Services also utilizes memory-based code sharing. This functionality allows a copy of an executable to be loaded into each virtual session, thus enabling all users to run the same application simultaneously without interfering with each other.

The Boot and Logon Process

When you boot a Win2K system, the session manager (SMSS.EXE) creates a console session that has a session ID of 0. If you have enabled Win2K Terminal Services, the new TERMSERV.EXE also loads and instructs the session manager to create two additional idle sessions. These idle sessions wait for a user to request a remote session.

👉 A common practice is edit the Registry to increase the number of idle sessions that the session manager creates so that the server can respond more quickly if multiple users simultaneously request remote sessions.

Each session gets a 2GB virtual address space for its user-mode processes. The system then loads copies of the Client Server Runtime Subsystem (CSRSS.EXE) and the Windows Logon Service (WINLOGON.EXE) into this space. These manage the user-mode processes and handle the logon request, respectively. Once these are in place, the new WIN32K.SYS loads to handle the graphics device interface (GDI) and USER roles of the WIN32 subsystem, as Figure 1.6 illustrates.

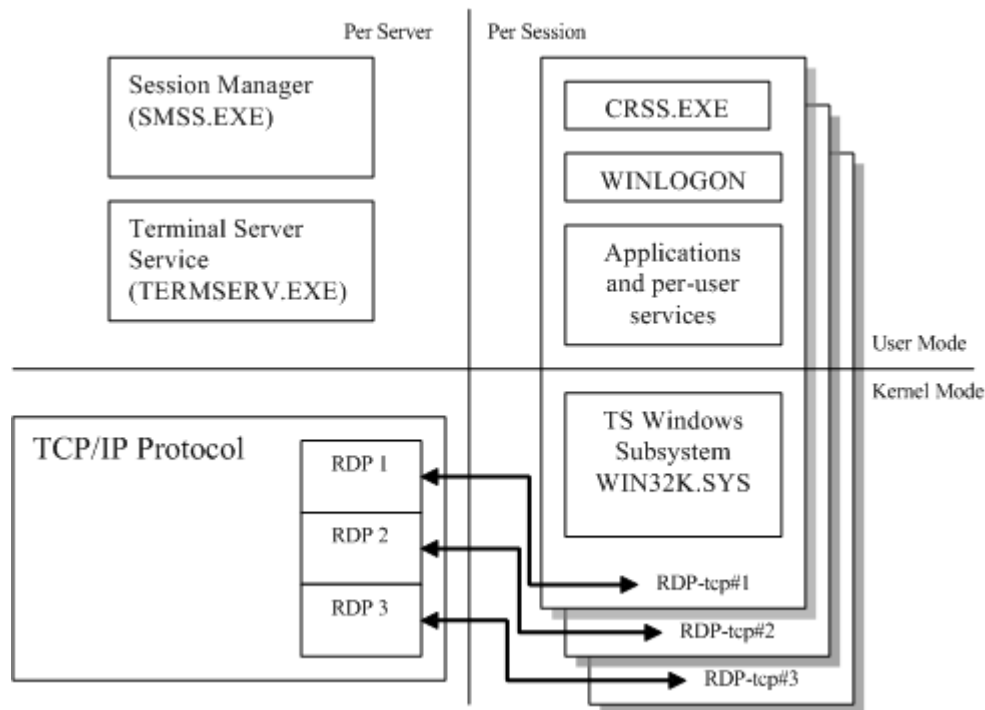


Figure 1.6: Services that create the multi-user environment.

When a user opens an RDP terminal connection to the server, he or she is assigned one of the idle sessions (which is immediately replaced with a new idle session for the next user) and WIN32K.SYS creates an instance of the GDI and generates an RDP Display Driver (RDPDD). This service and virtual device driver render the logon screen for the user and stream it back to the end-node device. After the user logs on and is authenticated, the GDI continues to stream all graphical information to the user and WIN32K.SYS accepts keyboard and mouse input from the user. The user is now ready to get to work.

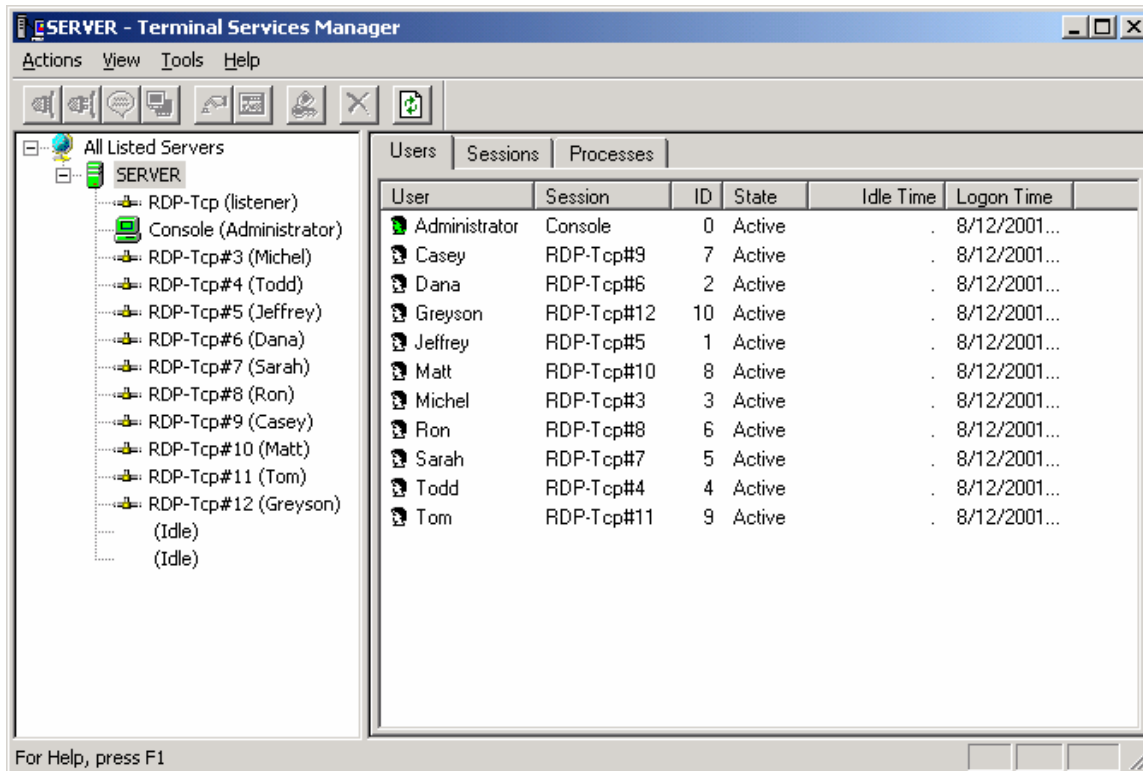


Figure 1.7: Active Sessions on a Terminal Server.

The Terminal Services Protocols

The traditional computing model relies on the standard TCP/IP protocol stack to transfer data back and forth between the workstation and server. The client/server processing model, however, has very specific needs for maintaining its network link between the client and server. Since only video information and keyboard and mouse movements are communicated and data isn't communicated, the protocol used must be robust and low in latency. Two main protocols have been developed to meet this need: RDP and ICA.

RDP

RDP was developed by Microsoft and is implemented in Windows' native terminal services. It is based on the International Telecommunication Union's T.120 protocol. RDP packets reside in the presentation layer of the Open Systems Interconnection (OSI) model and require a TCP/IP connection (IPX is not supported). Win2K Terminal Services uses RDP5, which provides additional features and improved compression over its predecessor, WTS' RDP4. RDP5's new features include support for

- Windows CE clients
- Remote control of user sessions
- Bitmap caching to disk (RDP4 supports caching only to RAM)
- Client clipboard mapping

RDP versus ICA


The ICA protocol was developed by Citrix and can be added to a Win2K terminal server by installing MetaFrame. These two protocols function in very similar ways, but have some important differences, which Table 1.1 outlines. (I'll cover the installation and configuration of RDP's features in Chapter 2.)

| Feature | RDP5 | ICA |
|------------------------------------|--|--|
| Supported client OSs | Win32, Win16, Windows CE, and PocketPC | Win32, Win16, Windows CE, PocketPC, MS-DOS, UNIX, Macintosh, Linux, Java |
| Browser-based client | Requires installation of Terminal Services Advanced Client on an Internet Information Services (IIS) server. | Supported natively by any Web server. |
| Transport protocol | TCP/IP only | TCP/IP, IPX/SPX, NetBEUI, direct serial |
| Sound | System beep only | Full stereo support available |
| Compression | Automatic | Automatic |
| Load balancing | Performed on a per-server level by the Windows Load Balancing Service (WLBS, which requires Win2K Advanced Server) | Performed on a per-application basis; requires purchase of additional load-balancing license |
| Automatic mapping of client drives | Requires the Drive Share resource kit utility | Automatic |
| Clipboard mapping | Automatic | Automatic |
| Remote control of user sessions | Automatic | Automatic |
| Mapping of local client printer | Automatic | Automatic |
| Seamless Windows Support | No | Automatic |

Table 1.1: Feature comparison of RDP5 and ICA.

Win2K Terminal Services Licensing

Under WTS, a common practice is to disable logging of TSCALs and simply purchase enough Client Access Licenses (CALs) to cover all your devices. Win2K Terminal Services doesn't allow you to disable license logging, so Win2K Terminal Services administrators are forced to install, configure, and maintain a Win2K Terminal Services License Server on their network.

 If you're using Win2K Terminal Services for remote administration only, you don't require a TSCAL and don't need a License Server.

The best way to explain Win2K Terminal Services licensing requirements is to let Microsoft do it. The following is an excerpt from the Microsoft End User License Agreement (EULA) for Win2K Terminal Services, which can be found in its entirety at <http://www.microsoft.com/windows2000/server/howtobuy/pricing/terminal.asp>:


In addition to the Windows 2000 server itself, Terminal Services has the following client licensing requirements when enabled in "Application Server" mode:

1. A Windows 2000 Server Client Access License (CAL) or BackOffice 2000 CAL is required to access the Windows 2000 server.
2. A Windows 2000 Terminal Services CAL or a license for Windows 2000 Professional is required to run Windows-based desktop and applications from a Windows 2000 server, regardless of the protocol or software used to interact with applications running on the server. As per the Windows 2000 Server End User License Agreement, "Terminal Services" means:
 - (i) using the terminal services feature of the Server Software to enable devices to use software residing on the Server, or
 - (ii) using other software in conjunction with the Server Software to provide similar services.

Appropriate application licenses. Licensing varies by independent software vendor (ISV). For instance, Microsoft Office is licensed on a per-device basis. This means that each end-device that displays Office applications requires a license to be assigned to it.

Note: Terminal Services is licensed on a per-device basis and is not available on a per-server or concurrent basis with the exception of the Terminal Services Internet Connector License explained below. Each device, whether it connects directly to the terminal server, or indirectly via another server, requires the appropriate licenses to be assigned to it.

Install a Win2K Terminal Services licensing server on your network, then contact the Microsoft Clearinghouse to purchase and install the Win2K Terminal Services CALs on the server. Both WTS and Win2K Terminal Services will grant devices running Win2K Pro access without requiring the purchase of an additional Terminal Services CAL. These devices are issued a special CAL under the Server and Desktop License Agreements. However, you still need to have a Terminal Services License Server available to issue this CAL to the Win2K Pro devices or the Win2K Terminal Services server will refuse connections after 90 days.

 You install the Terminal Server License Server through the Add/Remove Software Control Panel applet, under the Add/Remove Windows Components applet.

Microsoft issues Win2K Terminal Services licenses on a per-device, not per-user, basis. The license is tied to a Registry key written to the client device. The one exception is when you install the Win2K Terminal Services Internet Connector License. This license takes the place of both the server CAL and the Win2K Terminal Services CAL. You use this license to let as many as 200 concurrent users access the terminal server and server resources over then Internet when they're not covered by your company's existing CALs. Typically Internet Connector Licenses are used for customer access rather than employee access. The following excerpt from the Microsoft EULA for Win2K Terminal Services provides more information:


The Terminal Services Internet Connector license allows a terminal server to serve up to 200 concurrent connections. This replaces the need for a Terminal Services CAL and Windows 2000 Server CAL to be assigned to a specific device. Terminal Services Internet Connector licensing may only be used for anonymous connections from non-employees. The Terminal Services Internet Connector license is currently available under

the Microsoft Open, Select, and Enterprise Agreement volume licensing programs. A Windows 2000 Internet Connector license is not required for a Windows 2000 server using the Windows 2000 Terminal Services Internet Connector license.

Terminal Services Licensing Service Enhancements

Microsoft recently released enhancements to the Terminal Services Licensing Service (TSLs) that eliminate two of the headaches of Win2K Terminal Services Licensing — preventing licenses from being issued to devices in situations in which a user opened a connection but was never authenticated and reclaiming licenses that are no longer in use.

Before this upgrade, if a user connected to a Win2K Terminal Services server from a client device, that device would be allocated a license in the Win2K Terminal Services License Server's database. If that device were removed from the environment or had its OS re-installed, the only way to reclaim its associated CAL and have it issued to another device was to call the Microsoft Clearinghouse and request a license release. The TSLs enhancements allow unused licenses to be returned to the license pool after a random number of days (between 52 and 89 days). Be sure to install this update on your Win2K Terminal Services Licensing Server and all your terminal servers.

 Install the TSLs enhancements before any CALs are issued, as existing CALs will not return to the license pool.

The following excerpt from the Microsoft article “Terminal Services Licensing Enhancements” at <http://support.microsoft.com/support/kb/articles/q287/6/87.asp> describes these enhancements:

Post Logon License Token Issuance

Current Behavior

Windows 2000 Terminal Servers issue Terminal Services CAL (TS CAL) tokens to all clients after they connect by using the Terminal Services client. The TS CAL token is presented to the device before a user enters credentials and is granted or denied access to connect.

Enhanced Behavior

When an unlicensed client connects for the first time, the Terminal Server issues a temporary TS CAL token. After the user has logged into the session, the Terminal Server instructs the License Server to mark the issued temporary TS CAL token as being validated. The next time the client connects, an attempt is made to upgrade the validated temporary TS CAL token to a full TS CAL token. If no license tokens are available, the temporary TS CAL token will continue to function for 90 days.

This enhancement is designed to prevent TS CALs from being inadvertently allocated to devices that are not intended to be licensed for Terminal Services usage. To allocate a TS CAL token to a device, a successful logon to a Terminal Server must occur. However, this does not prevent users who are authorized to log on to a Terminal Server from logging on from devices that the organization does not intend to license. If this happens, a TS CAL token is still assigned to the device.

Automatic License Token Re-issuance

Current Behavior

TS CAL tokens are issued for each device, and are stored locally on each device that connects to a Windows 2000 Terminal Server. If a device loses this TS CAL token through hard disk failure, clean reinstallation, or other method, the TS CAL token remains assigned to that device. The only way to recover this TS CAL token is to place a phone call to the Microsoft Clearinghouse.

Enhanced Behavior

An expiration period has been added to each TS CAL token that is issued. This expiration period is a random number of days between 52-89 days of issuance. When a client connects to a Terminal Server, this date is checked. If the expiration is within 7 days, the Terminal Server connects to the License Server and renews the TS CAL token, giving it another expiration period of 52-89 days. If the License Server is not available, the TS CAL token functions as normal, with the Terminal Server attempting to replace it at each login. Any TS CAL token that has not been renewed is returned to the group of available license tokens by the License Server upon expiration.

For example, an unlicensed device connects and receives a TS CAL token with an expiration period set at the maximum of 89 days. The device's operating system is then reinstalled. The device then connects again. Because no other TS CAL tokens are available, the device is issued a temporary TS CAL token so it can connect for 90 days. On day 89, the original TS CAL token is returned to the group of available licenses. The next time this device connects, the Terminal Server presents the device with the full TS CAL token that was returned to the group of available license tokens.

With the addition of these fixes, it should not be necessary to call the Microsoft Clearinghouse to recover lost license tokens. If a device loses its license token, the administrator can be confident that license tokens that are issued after the enhancement was installed will be recovered automatically.

IMPORTANT: There are a few cases in which license tokens will not be recovered automatically:

License tokens are issued prior to the installation of this hotfix. Only TS CAL tokens that are issued after the installation of this fix will utilize the re-issuance logic. A TS CAL token that is issued to a device prior to the installation of this hotfix will remain assigned to that device. The Clearinghouse must be contacted to recover any TS CAL tokens that are issued prior to the installation of this hotfix. Because of this, it is important that this hotfix be installed on all Terminal Servers and Terminal Services Licensing Servers in an enterprise.

Catastrophic failure that results in the loss of the licensing database. In the event of a failure that results in the loss of the licensing database when a known good backup is not available, Terminal Services Licensing must be reinstalled and reactivated. The Clearinghouse will then need to reissue any previously issued License Key Packs. The License Key Packs that were originally issued are based on the License Server ID at the time of issuance. If the License Server ID changes, License Key Packs that are based on the old License Server ID cannot be installed.

Assuming that you've installed the TSLS enhancements, the license-issue process looks like the process outlined in Figure 1.8.

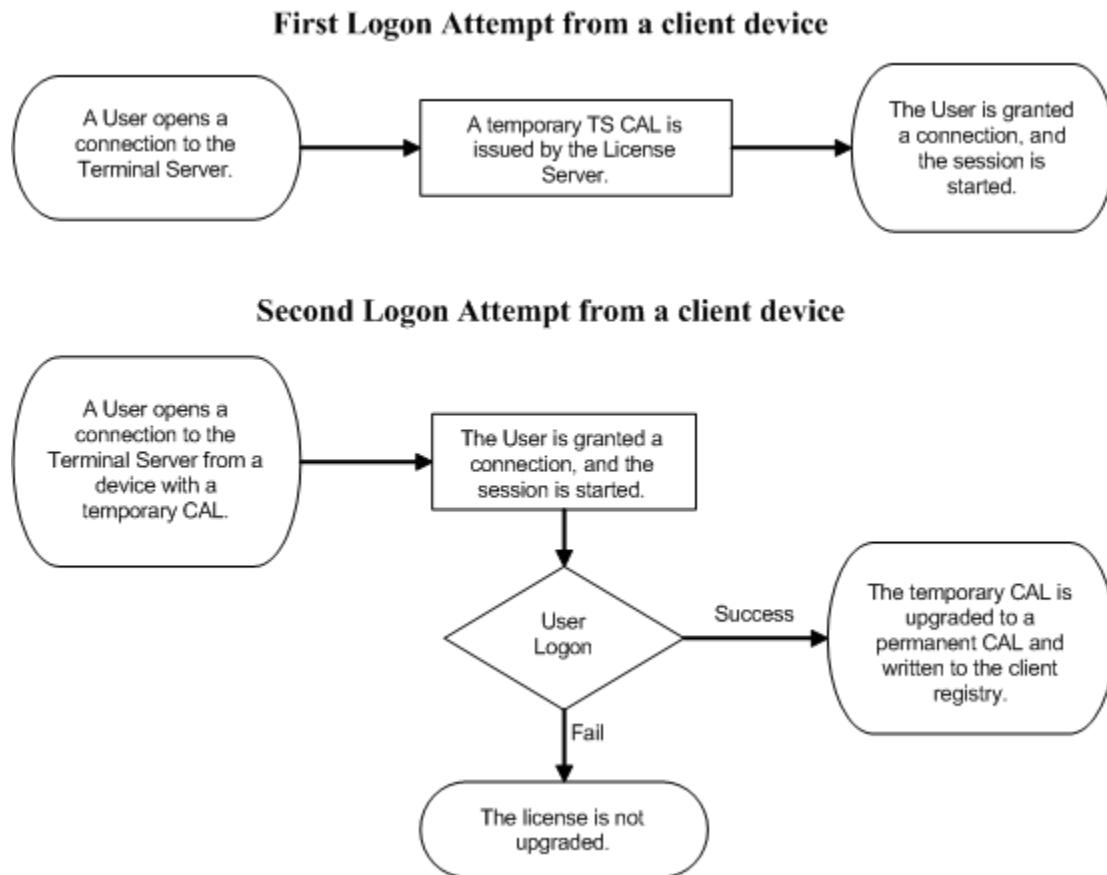


Figure 1.8: Terminal Services CAL-issuance process.

MetaFrame Licensing

If you've installed MetaFrame, and users are connecting to the terminal server over the ICA protocol, they'll need a Citrix CAL in addition to a Microsoft Terminal Services CAL. MetaFrame has its own tools for installing, issuing, and tracking licenses. Citrix CALs are issued on a concurrent-use, rather than a per-device, basis, so they're much easier to set up and maintain.

Summary

This chapter introduced you to terminal services. In it, I gave you an overview of the history and evolution of the technology as well as a general idea of the different ways that you can use terminal services in your Windows infrastructure. I then showed you the protocols and services that go into creating the software environment needed for a multi-user Windows system. Finally, I explained the various licenses that you need before you deploy Win2K Terminal Services. In Chapter 2, I'll discuss the installation and configuration of Win2K Terminal Services.


Chapter 2: Installing and Configuring Terminal Services

This chapter will focus on the installation and configuration of Win2K Terminal Services. I'll begin with the most basic configuration—remote administration—then move into application server mode. Keep in mind that you may need to modify some of the procedures outlined here to fit your infrastructure's standards and policies.

As with any deployment of new technology, you should begin by setting up a test server so that you can familiarize yourself with the process. Another good practice is to maintain a network share to store any scripts, utilities, and registry changes that you use so that you can replicate these onto additional servers as you build them.

Remote Administration Mode

Installing Terminal Services in remote administration mode is the most common and the simplest configuration. This mode gives systems administrators the ability to remotely control a Win2K server. Remote administration mode doesn't require a Terminal Services License Server, and applications installed on the server aren't tuned for multiple interactive users.

 Some systems administrators make the mistake of trying to use remote administration mode as a two-user Application Server because this mode doesn't require the purchase of additional TSCALs. But doing so can have unexpected results as applications running on the server aren't tuned for simultaneous users.

To enable remote administration mode on an existing Win2K server, go to the Add/Remove Programs Control Panel applet and select Add/Remove Windows Components. Select the Terminal Services check box, and click Next, as Figure 2.1 shows.

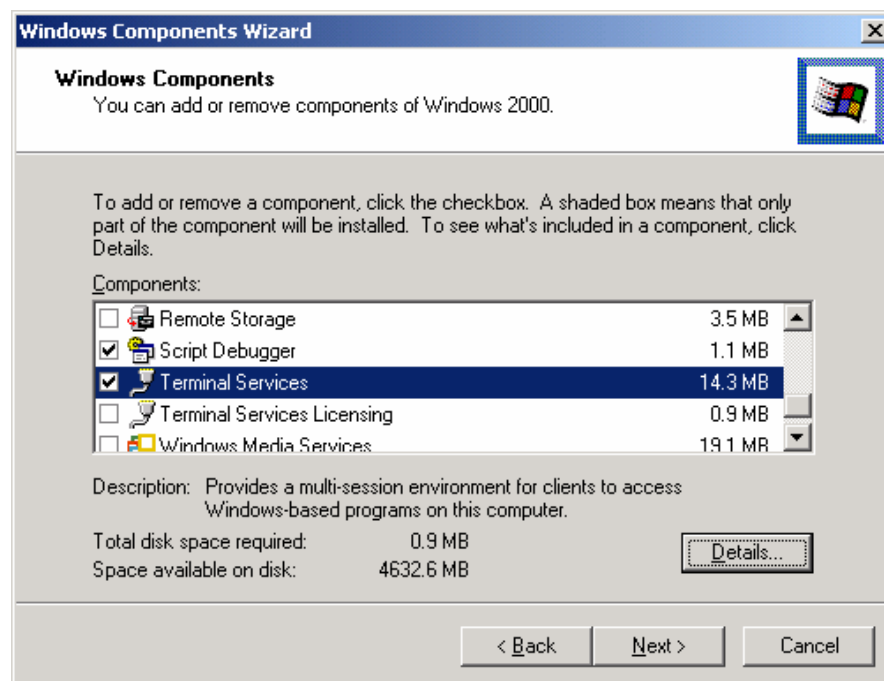


Figure 2.1: Installing Terminal Services.

Next, you'll be presented with the Terminal Services Setup window, which Figure 2.2 shows; select the remote administration mode radio button, and click Next.

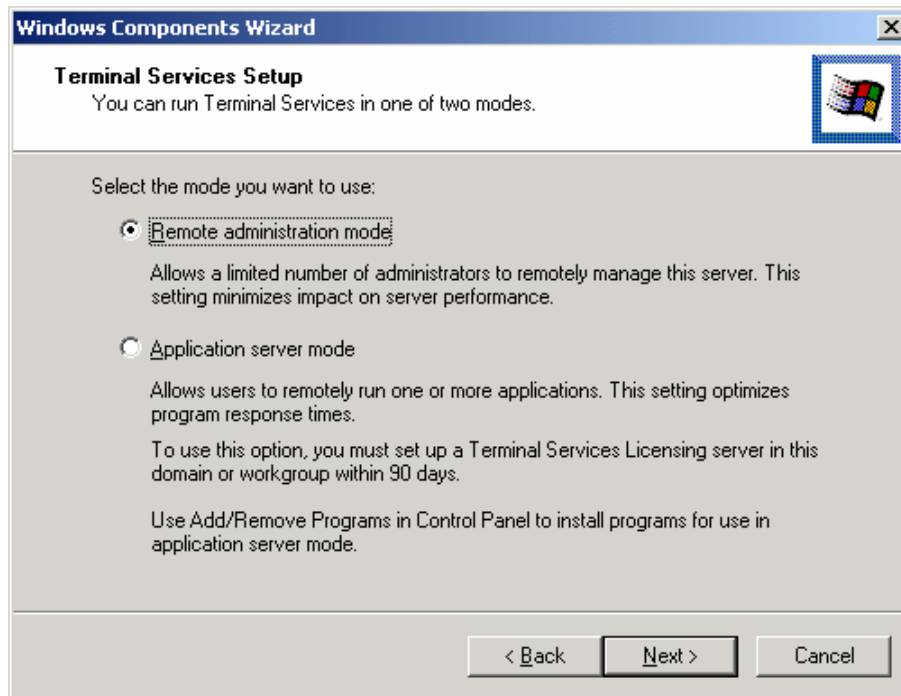



Figure 2.2: Selecting remote administration mode.

Windows will then install the components and ask you to reboot the server. After reboot, the server is ready for remote administration. To open a remote administration connection to the server, you need to be in the Local Administrators group on the server. Keep in mind that if you're installing Terminal Services in remote administration mode on a domain controller, this setup will require you to be in the Domain Administrators group on the server. Also, the server is limited to two concurrent connections.

 Because remote administration mode is limited to two concurrent connections, you may want to set an idle session timeout so that a connection is available when you need it. See "Managing Client Connections" in Chapter 4 for instructions about how to do so.

If you're planning to use remote administration mode to manage multiple servers in your environment, you may want to use the Microsoft Management Console (MMC) Terminal Services Connections snap-in rather than the traditional Terminal Services client. Figure 2.3 shows this snap-in.

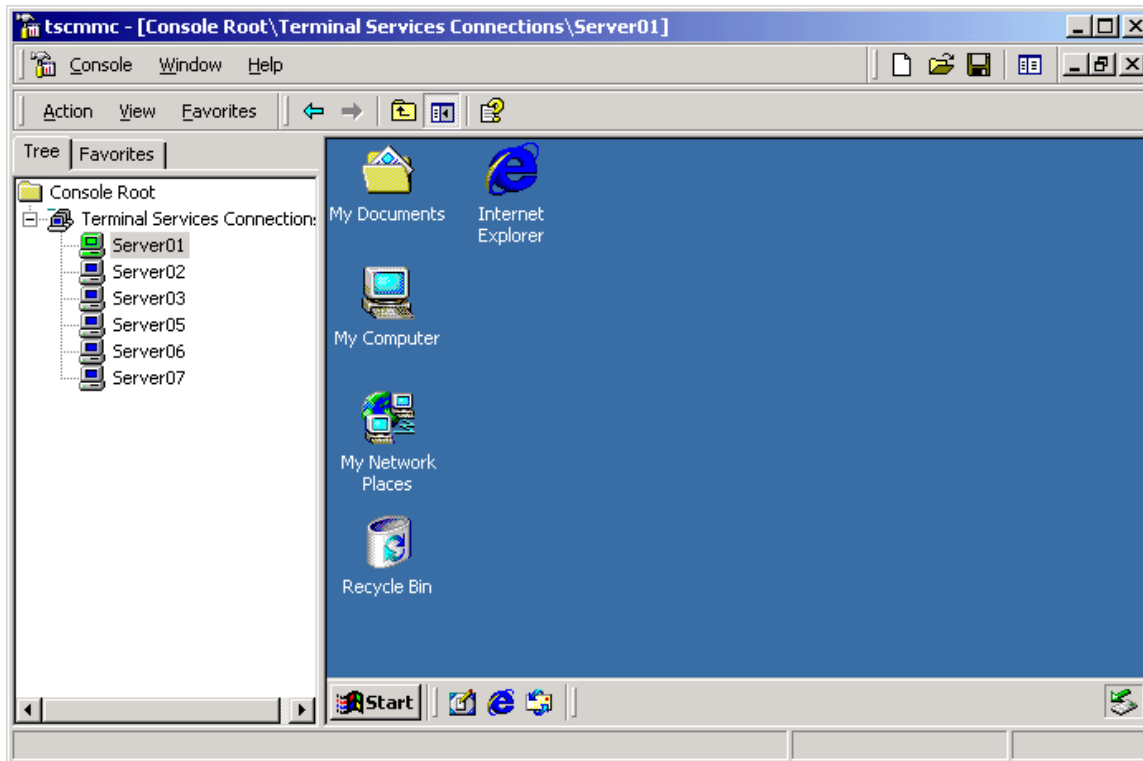


Figure 2.3: The MMC Terminal Services Connections snap-in.

The MMC snap-in client is designed for users running Win2K Pro and lets you easily open and manage multiple Terminal Services connections within a single application window. The MMC client has a two-pane interface, with available servers listed down the left pane, and the highlighted session appearing in the right results pane.

 You can download the Terminal Services Connections MMC snap-in from Microsoft at <http://www.microsoft.com/windows2000/downloads/recommended/TSAC/tsmmc.asp>.

Application Server Mode

Before enabling Terminal Services in application server mode, you must first consider server configuration and sizing. This consideration involves looking at disk size and partitioning scheme, the amount of physical memory required, and the number and speed of processors used.

Hard Disk Configuration

Terminal Services should typically be used to hold application executables only and not databases or user documents so that disk capacity is rarely an issue as long as you have adequate space for the OS, applications, swap file, and local copies of user profiles. The configuration of the disk partitions is very flexible. Keep in mind that Win2K doesn't have the 4GB partition limitation that NT 4.0 has, so you're able to make your system partition as large as you like. Also, Terminal Services Application Servers are more like workstations than other file and print servers, so you may want to partition them as you would a workstation instead of as a server.

Here are some options for Terminal Services Application Server partitioning, which Figure 2.4 illustrates:

- **Single partition**—If you have a fairly powerful server with adequate RAM and fast drives, there is absolutely nothing wrong with a single-partition model. Most applications are written to assume an installation directory of C:\Program Files, so keeping applications on the same partition as the OS will simplify your application integration.
- **Master partition with separate swap-file partition**—If you're concerned about the amount of swap-file activity that Terminal Services will produce, you may want to separate your swap file onto a separate physical disk. This separation will improve performance while still giving you the simplicity of a single system and application partition.
- **Separate system and application partitions**—Some systems administrators prefer to create a separate partition for applications rather than use the C:\Program Files directory. This setup allows systems administrators the freedom to back up and restore the system separately from the applications in the event of failure. Keep in mind, however, that most applications designed for Win2K maintain shared components and registry settings in the system partition, so the segregation isn't as clean as it was under Win16.

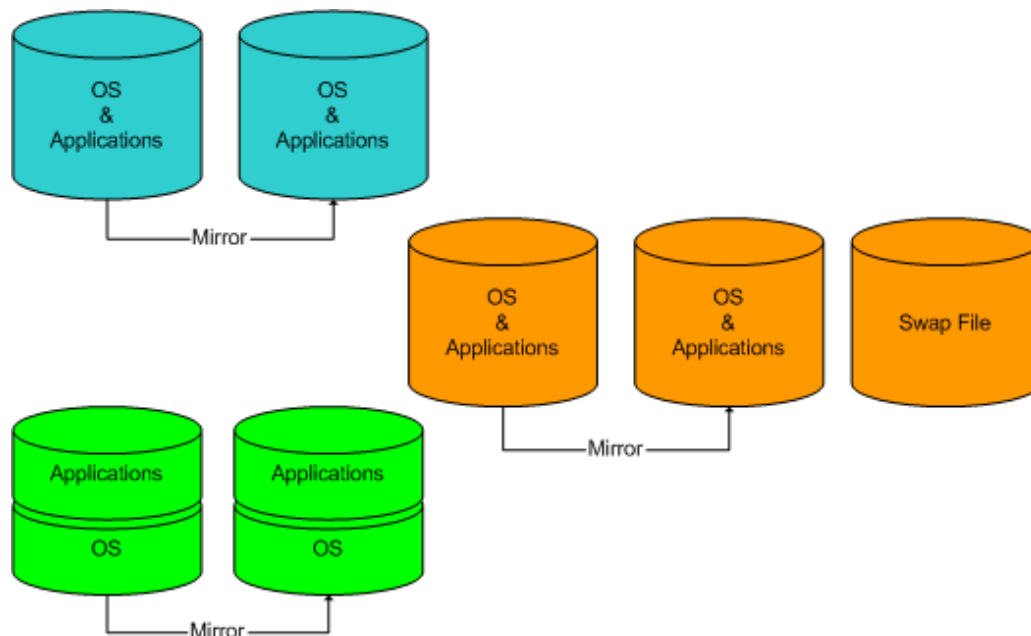



Figure 2.4: Terminal Services partitioning schemes.

You can accomplish these partitioning schemes by any combination of logical and physical disks as well as software or hardware RAID arrays. The type of server hardware that you're using as well as the number of physical disks available, will factor into your decision. If you're introducing Terminal Services to an existing server infrastructure and support team, you may need to provide some explanation as to why the partitioning scheme for Terminal Services should be different from other servers in the environment.

Physical Memory

Physical memory is the single most important factor in a successful Terminal Services Application Server design. With DRAM prices dropping all the time, you'll often see Terminal Services administrators install the maximum amount of RAM that the server will support. But you should be aware of the amount of memory that your server actually needs so you can better plan for expansion and scalability.

 Luckily, Microsoft, NEC, and Groupe Bull teamed up and produced a white paper to help you determine your servers' memory needs. You can find the complete text of the white paper "Windows 2000 Terminal Services Capacity and Scaling" at <http://www.microsoft.com/windows2000/techinfo/administration/terminal/tscaling.asp>

Before you can estimate your memory needs, you need to understand how Terminal Services allocates memory. In Chapter 1, you learned that each user session is allocated a 2GB memory space. This space is virtual and doesn't mean that you need 2GB of RAM per user. Instead, you need to look at which applications your users will be running as well as users' work habits.

First, consider the applications themselves. Are you installing basic productivity applications, such as Microsoft Office 2000, or are you using Terminal Services to provide users access to a centralized database application? Each scenario has its own RAM requirements.

As a baseline, let's look at a basic application such as Microsoft Word. While Word is running, you can look at Windows Task Manager to see just how much memory space Word needs—its *footprint*. Figure 2.5 shows an example Windows Task Manager window.

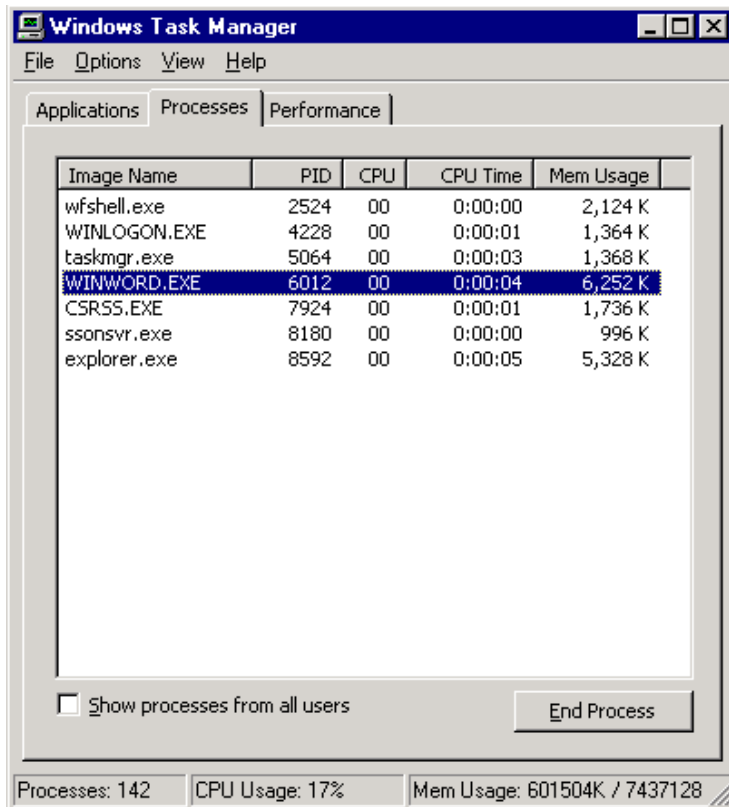


Figure 2.5: Memory footprint of WinWord.exe.

In my case, Word 2000 is using between 3MB and 20MB of memory depending on which functions I'm using. Keep in mind that even while I'm writing this book, Windows has swapped-out most of Word's footprint to the paging file, so you don't need to actually allocate 30MB of RAM for each instance of Word on your Terminal Services server. But Windows Task Manager is still a good reference point for unusual applications that you may come across. Also remember that 16-bit applications consume more memory than 32-bit applications because 16-bit applications require the additional memory usage of a virtual machine (wow.exe) to run. This requirement can add about 25 percent more memory than the 32-bit version of the same application.

To start off, Microsoft recommends 128MB for the server itself. Microsoft then created a testing environment in which the company could push Terminal Services to its limits under different types of user work styles. What the company came up with were some recommended memory specifications for three types of users, which Table 2.1 outlines.

| User Type | Memory Per User (MB) | System memory (MB) | Total Memory |
|------------------------|----------------------|--------------------|--|
| Structured task worker | 9.3 | 128 | System + (number of users × memory per user) |
| Knowledge worker | 8.5 | 128 | System + (number of users × memory per user) |
| Data entry worker | 3.5 | 128 | System + (number of users × memory per user) |

Table 2.1: Base memory recommendations from the Microsoft, NEC, Groupe Bull tests.

To interpret these results, you need to understand the definition of each user category and how it applies to your user community. The following list defines each user type:

- **Structured task worker**—Users who tend to use one application at a time in a sequential task-oriented process. These users are typically fast typists (about 60wpm) and are only partially dependant on computer availability. These users typically work in claims processing, account processing/accounts receivable, or customer service.
- **Knowledge workers**—Users who are dependent on their computers to get work done. They tend to use multiple applications simultaneously, toggling between them. Their workflow is more ad hoc rather than sequential. These users are the executives, project managers, or analysts.
- **Data entry workers**—Users who input data into computer systems such as transcription typists, order entry clerks, clerical workers, and manufacturing personnel.

So, as an example, let's assume that you are building a Terminal Services server for desktop replacement for 100 knowledge workers. According to the test results, you would need 128MB for the system plus 8.5MB per user. The total amount comes up as 950MB of RAM as a minimum requirement.

One thing to keep in mind when using these results as a baseline is that Microsoft conducted these tests exclusively with Microsoft applications—Office 2000 with Outlook to be precise.

Microsoft designed Office 2000 with Terminal Services in mind, so its component applications are very good at sharing resources. In the real world, you'll probably be using a combination of applications from different manufacturers, so I would recommend scaling the RAM requirements up by about 50 percent to be safe—about 12MB per user.

If you're sticking with mainstream software and have users with typical usage patterns, you can use the Microsoft, NEC, Bull results as a good guideline. But if you're deploying a unique or mission-critical application, you may want to do some benchmarking of your own. To do so, you'll need a testing environment—a few servers and some client workstations—and the same resource kit utilities that Microsoft used for its tests.

 The utilities that Microsoft used for the Terminal Services scaling white paper can be found at <http://www.microsoft.com/windows2000/techinfo/administration/terminal/loadscripts.asp>.

With these utilities, you can set up a large number of automated “roboclients” that connect to the terminal server, perform a sequence of tasks, then log off. You can then monitor the server for memory and processor usage and calculate the amount of RAM that you need.

Processors


Because of the multiple parallel tasks being performed on a Terminal Services system by all its users, processor load can increase very quickly. You'll see a much greater performance improvement by adding processors to a Terminal Services server than you would typically see in another type of server.

With additional processors, the server can unload CPU-intensive tasks, such as a single user crunching a large financial document, to a separate CPU rather than time-slicing his work among that of the other users on the server. In many cases, you'll gain more by adding processors to a Terminal Services server than by adding RAM. (You can review the Terminal Services scaling white paper to see the speed and number of processors that Microsoft used in its testing.)

Load Balancing

If you're building a Terminal Services infrastructure for a very large number of users or hosting mission-critical applications that require fault tolerance, you'll eventually hit a wall and you simply can't throw additional RAM and processors at a single server. At this point, you need to consider load balancing.

With Microsoft Network Load-Balancing Service added to your Terminal Services servers, you can configure multiple servers to act as one. The service will funnel users to the server that has the capacity to handle their user sessions, and if one server goes down, the other will continue to accept connections until you can correct the problem. See Chapter 5 for additional information about load balancing.

 Terminal Services is incompatible with Windows Clustering Services. Also, Load-Balancing Service is available only on Advanced and Datacenter Server.

Putting It All Together

After you've completed your server sizing work or you've opted for the easy way out and purchased 8-way 1GHz servers with 4GB of RAM, you're ready to install Terminal Services in application server mode and start integrating your applications. Begin by building your server (installing RAM and processors, configuring the drive arrays and partitioning the drives), then install Windows 2000 Server, Advanced Server, or Datacenter Server. Add any drivers and system utilities you need for the environment. Then install Terminal Services by going to the Add/Remove Programs Control Panel applet and selecting Add/Remove Windows Components. Select the Terminal Services check box, as Figure 2.6 shows, and click Next. In the Terminal Services Setup window, select application server mode.

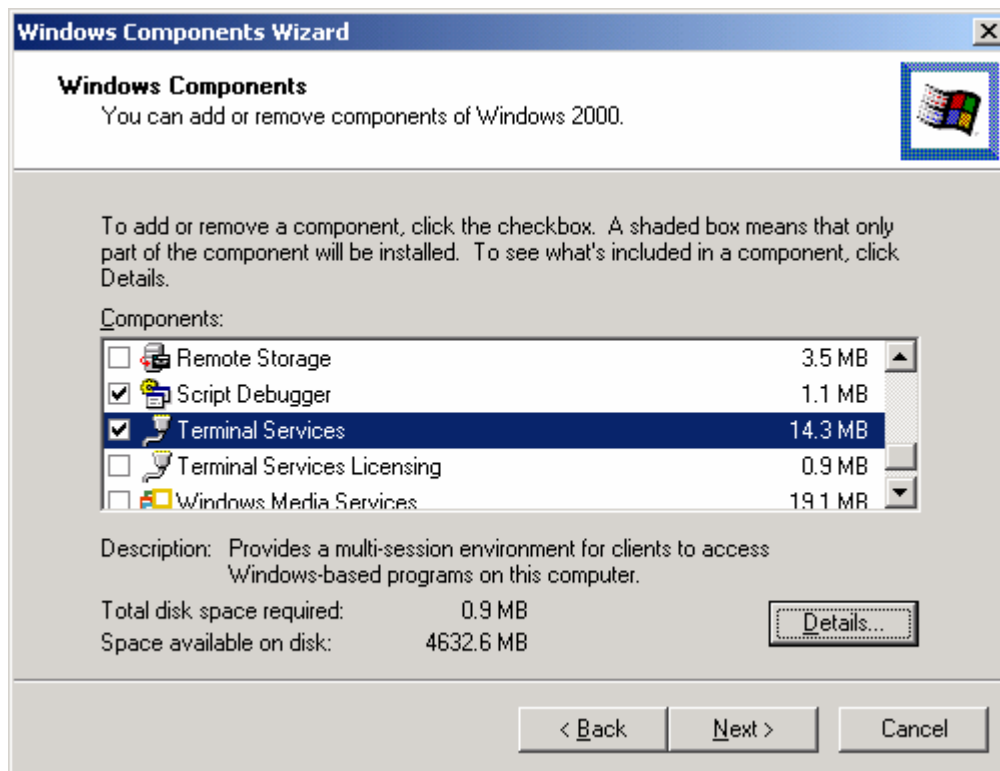


Figure 2.6: Installing Terminal Services through the Add/Remove Windows Components Control Panel applet.

If you've installed any applications on the server before this point, you'll now be informed that these applications may not work under Terminal Services. Occasionally, Windows will consider certain system utilities or drivers as applications and report them in this screen, you can ignore these. But if you have indeed installed any user applications, you're better off going back, uninstalling them, then reinstalling them after you enable Terminal Services.

The next screen in the Terminal Services Setup Wizard asks you which style of permissions you want to use on the terminal server. Figure 2.7 shows this window.

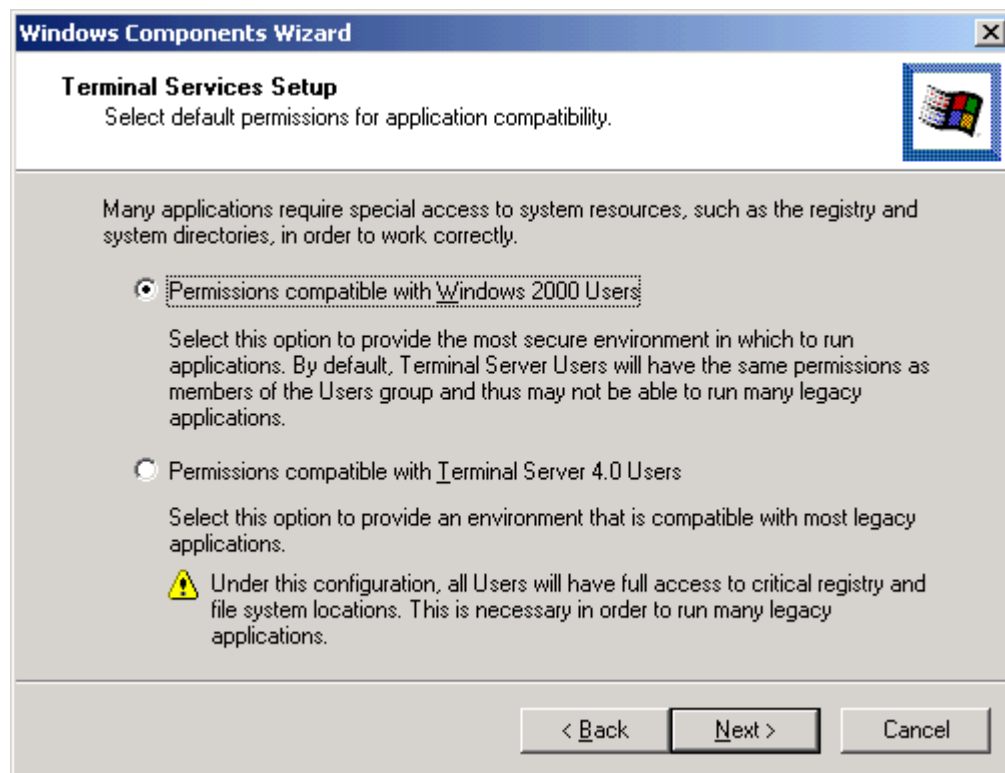



Figure 2.7: Selecting a permission mode for the application server.

As the wizard explains, the selection here affects permissions on both the registry and system files. A preferable practice is to use Win2K-style permissions, but if you're using any older applications (any application that doesn't explicitly state that it's compatible with Win2K is suspect), you may need to use Terminal Server 4.0-style permissions. If you're unsure, select Win2K, then change this setting if any of your applications report errors to see if the new setting fixes the problem. I'll further explain the differences in the permission modes and how to determine which you should use in Chapter 3.

 If you use Terminal Server 4.0 permissions, you'll need to keep careful watch on your server, as there are some very common shareware utilities that, if a user attempts to install them, will be able to add undesired registry keys and files to the server. Typically, the installation will fail, but you'll still have to perform housekeeping.

After Windows installs the components needed for Terminal Services, it will instruct you to reboot the server. Before installing applications, there are a number of tweaks you can make to the system that will greatly improve performance of your Terminal Services.

System Tuning For Terminal Services

Now that you've installed Terminal Services, there are a number of measures you can take to tune the system for optimal performance under the heavy load of numerous simultaneous users. I'll make the assumption that you're running Win2K with Service Pack 2 (SP2) installed.

The first step is to connect to Windows Update and install any “Critical Updates” and “Application Compatibility Updates”. These updates will ensure that you have the most up-to-date security and the most application-friendly configuration. After that, you should install any post-SP2 hotfixes that Microsoft recommends for Terminal Services. Be sure to install the “License Services Enhancements” hotfix on all terminal servers, even if they’re not running Terminal Services License Server. Some components of this hotfix must be on the terminal server itself to provide the enhanced license-issuance behaviors.

Adjusting Server Settings

The Terminal Services Configuration utility is a small nexus of very powerful settings. This program is found under Administrative Tools and is used to tune the servers Terminal Services settings as well as manage RDP itself.

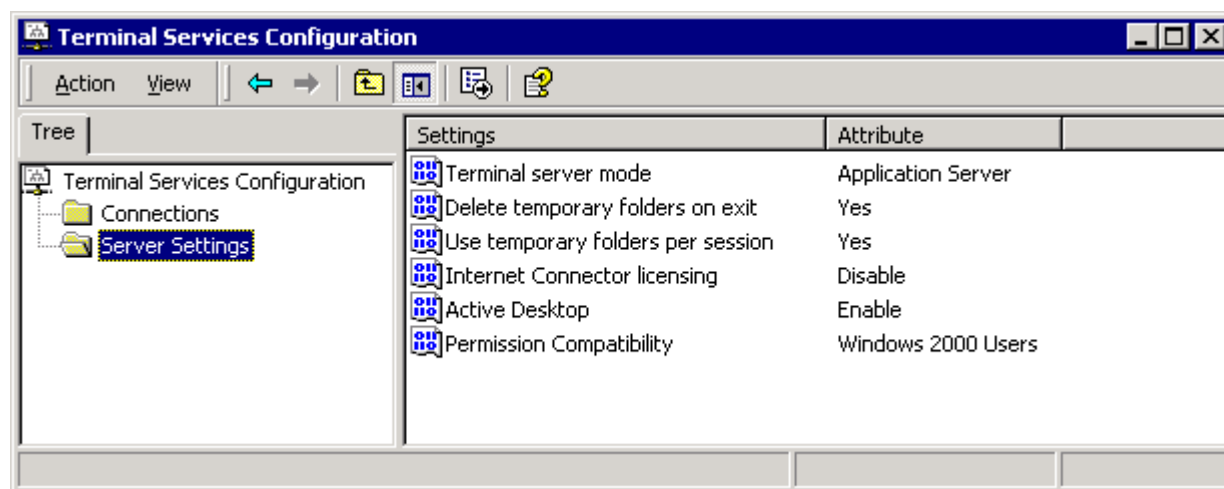




Figure 2.8: The Terminal Services Configuration utility.

Under Server Settings, you can change the mode of Terminal Services from application server to remote administration as well as change the permission mode that you selected when installing Terminal Services in application server mode. The other items here apply changes to the registry to improve server performance:


- **Delete temporary folders on exit**—This setting instructs the server to delete all files in the Temp directory when a user logs off the server.
- **Use temporary folders per session**—By default, the server will create each user an individual Temp directory in the Local Settings folder of the user’s profile. If you’re very low on disk space, you can disable this function and all users will share the C:\Winnt\Temp directory. This setting means that if two users are writing the same file to Temp, one file will overwrite the other.
- **Internet Connector licensing**—If you’ve installed the Internet Connector License on the terminal server, you can temporarily prevent anonymous logons here.
- **Active Desktop**—By default, the system will allow HTML content on the desktop over a Terminal Services connection. If bandwidth is a concern, you can improve performance by disabling this setting.

 The connections window of this tool is used to configure the user environment. This topic is covered in Chapter 4: Terminal Services Administration.

In addition to the settings provided here, there are a number of tweaks that have become standard practice among Terminal Services administrators. Be sure you understand each of these changes before you apply them so that you can evaluate their impact in your environment:

 If there is a Microsoft Knowledge Base article describing the registry change, I will list the article number in parenthesis. Also, all registry changes will be listed in the Appendix.

- Instruct Windows to ignore the Uniform Naming Convention (UNC) path contained within an LNK file by modifying the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer subkey LinkResolveIgnoreLinkInfo value to have a Dword-type value of 00000001. (Q195887)
- Disable the system debugger “Dr. Watson” by deleting the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug registry subkey. (Q188296)

 You should disable Dr. Watson only after you’ve completed installing and testing your applications. Without the system debugger, troubleshooting application issues might be difficult, especially if you’re using Microsoft’s support to assist you. If you need to re-enable Dr. Watson to assist in diagnosing an application issue, go to a command prompt and type

```
drwtsn32 -i
```

- Increase the number of idle RDP connections. I recommend increasing it to five by modifying the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server registry subkey’s IdleWinStationPoolCount value to have a Dword-type value of 00000005.
- Set user overrides for desktop settings. These settings override the user’s preferences when logging on over RDP to optimize performance and reduce screen refreshes. To do so, modify the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\UserOverride\Control Panel\Desktop\WindowMetrics registry subkey MinAnimate value to 0. This setting disables animation when resizing windows. You can also set user overrides for desktop settings by modifying the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp\UserOverride\Control Panel\Desktop registry subkey to include the values that Table 2.2 shows.

| Subkey | Value | Description |
|--------------|-------|---|
| AutoEndTasks | 1 | Automatically terminates programs that aren’t responding. |

| | | |
|----------------------|------------------------------|---|
| CursorBlinkRate | -1 | Prevents the cursor from blinking, which cuts down on screen redraws. |
| DragFullWindows | 0 | Doesn't show contents while dragging a window. |
| MenuShowDelay | 10 | Sets the delay for showing submenus. |
| WaitToKillAppTimeout | 20000 | The number of milliseconds to wait before terminating an application that isn't responding. |
| SmoothScroll | Dword-type value of 00000000 | Disables smooth scrolling. |
| Wallpaper | (none) | Disables wallpaper. |

Table 2.2: Registry values to set user override settings.

In addition to these registry changes you should change the following settings to tune the overall server performance:

- Tune the event logs. In Event Viewer, adjust the properties of each log so that its maximum size is 1MB, or larger if you want an extended history, and set it to overwrite as needed.
- In the Advanced Startup and Recovery Options of the System Control Panel applet, adjust the Write Debugging Information settings to save the debug dump to an appropriate location, set it to overwrite, and confirm that the server is set to automatically reboot.
- Also in the System Control Panel applet, under Performance Options on the Advanced tab, increase the maximum registry file size to 130MB to 150MB depending on your expected number of concurrent users.
- In the Add/Remove Programs Control Panel applet, under Add/Remove Windows Components, uninstall the Script Debugger.

If you're an experienced NT 4.0 WTS administrator or have done research on tuning Terminal Services, you'll know that there are a number of other changes that are typically made to NT 4.0 Terminal Services. Most of these standard changes involve disabling services. Be very careful when making changes to services under Win2K Terminal Services, as Windows now uses some services in ways that may cause unexpected problems if you disable them. For example, a common practice is to disable the telephony service on servers, but Outlook 2000 requires this service to autoforamt telephone numbers in users' contacts.

Also, many of the other common registry changes that NT 4.0 WTS administrators make are actually the default settings under Win2K Terminal Services, so be sure that you're not performing unnecessary changes. Always look before you hack.

Setting up a Terminal Services License Server

As you read in Chapter 1, each device that connects to a terminal server in application server mode needs a TSCAL. A Terminal Services license server is used to install, distribute and

manage these TSCALs. Without a license server, the terminal server will stop accepting connections after 90 days.

If you're in a Win2K domain environment, you should install the Terminal Services Licensing service on one of your domain controllers. All terminal servers in the domain will automatically find the license server. To install Terminal Services Licensing service, select the Add/Remove Programs Control Panel applet, select Add/Remove Windows Components, and select the Terminal Services Licensing check box, as Figure 2.9 illustrates.

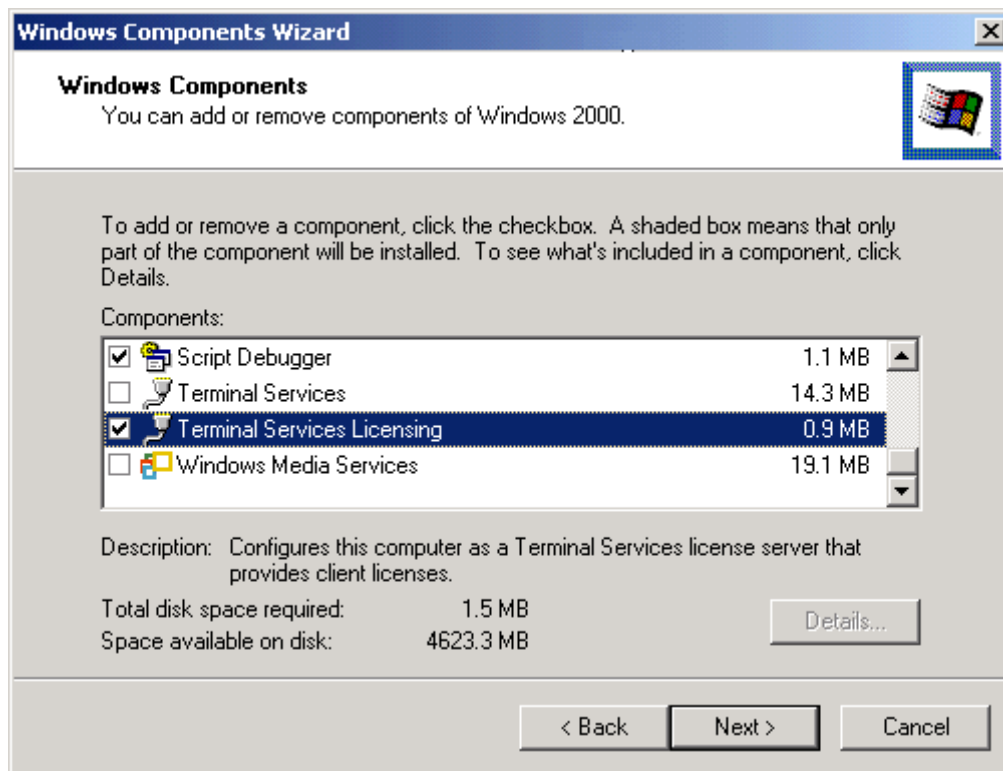


Figure 2.9: Installing the Terminal Services Licensing service.

When installing Terminal Services Licensing service, you can select whether you want the server to serve licenses to the domain or the entire enterprise (forest), as Figure 2.10 shows. To set up a license server for the domain, you must be a member of the Domain Administrators group. To create an enterprise license server to distribute TSCALs to the entire forest, you must be a member of the Enterprise Administrators group.

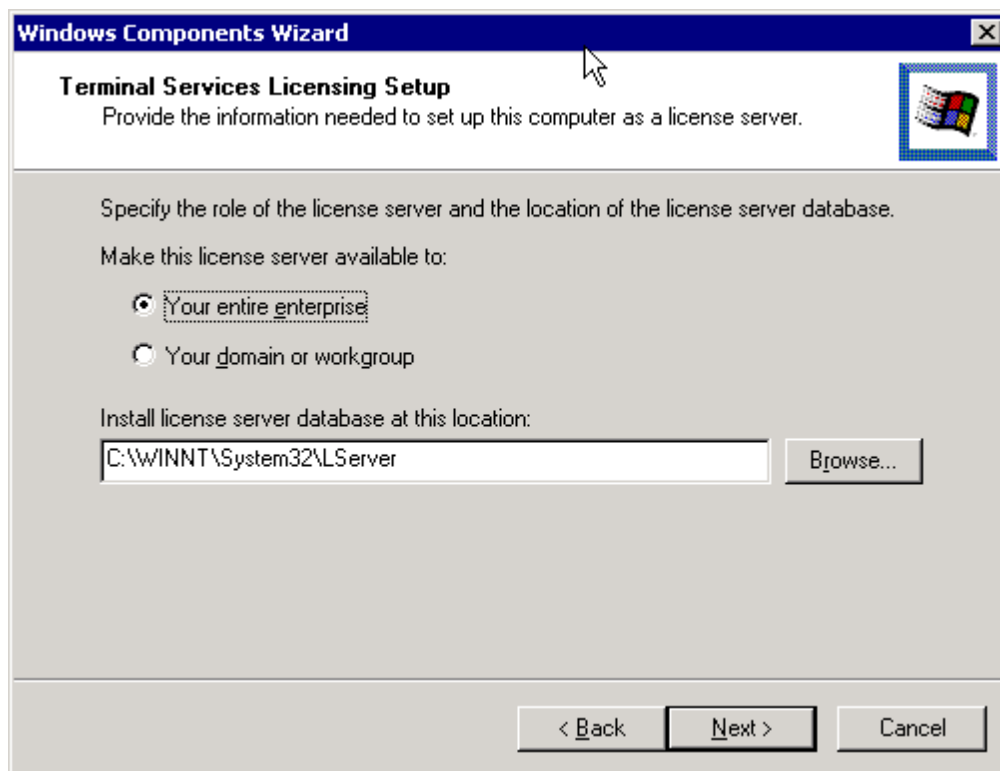


Figure 2.10: Choosing a domain license server or enterprise license server.

If you're in an NT 4.0 domain, you'll need to install Terminal Services Licensing service on a member server running a flavor of Win2K Server. When installing the service, you'll also need to specify a location for Windows to store the license database.

At this point, you should also install the Terminal Services Licensing service enhancements hotfix. This hotfix is described in and available from the Microsoft article "Terminal Services Licensing Enhancements" at <http://support.microsoft.com/support/kb/articles/q287/6/87.asp>.

Activating the License Server

After you've installed Terminal Services Licensing service on the appropriate server or servers, you'll need to activate the license server. This activation sets up the server to accept TSCALs that you purchase from Microsoft. To activate the Terminal Services Licensing service, launch the Terminal Services Licensing application from Administrative Tools and select Activate Server from the Action menu.

After the initial welcome screen, the wizard, which Figure 2.11 shows, will ask you to select a method for connecting to Microsoft. For any of these methods, you will need the product ID displayed on the welcome screen, but the wizard will provide this number when necessary.

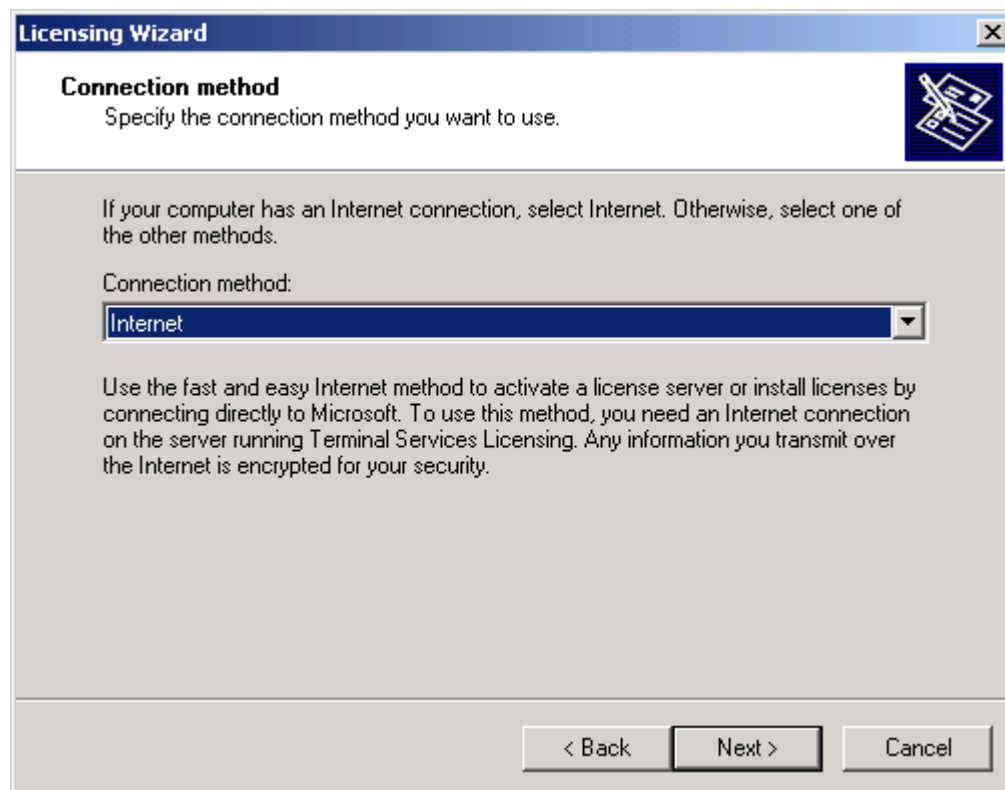


Figure 2.11: The Terminal Services Licensing Wizard.

The four connection methods are:

Internet—If you select this option, the wizard will then ask you to complete a form containing your company’s information and an email address. Your 35-character PIN is emailed to the address you provide.

World Wide Web—If you select this method, the wizard will instruct you to go to <https://activate.microsoft.com> and fill out your company information. Your PIN will be sent to you through email.

Fax—For this option, the wizard will have you select your country, then ask for your company information and fax number. Next, you’re instructed to print the form and fax it to Microsoft, who will respond with your PIN by fax.

Telephone—In this method, you select your country and are provided with the appropriate customer service number. A representative will take your company information and provide your PIN immediately.

After you select your connection method, the wizard asks how you purchase licenses from Microsoft, as Figure 2.12 shows. If you’re a Select or Open License customer, be sure to have your account number available when you attempt to install licenses. Select the Other radio button if you’re not a Select or Open License customer. You purchased your TSCALs through a retail channel and have the paper license numbers included in the package.

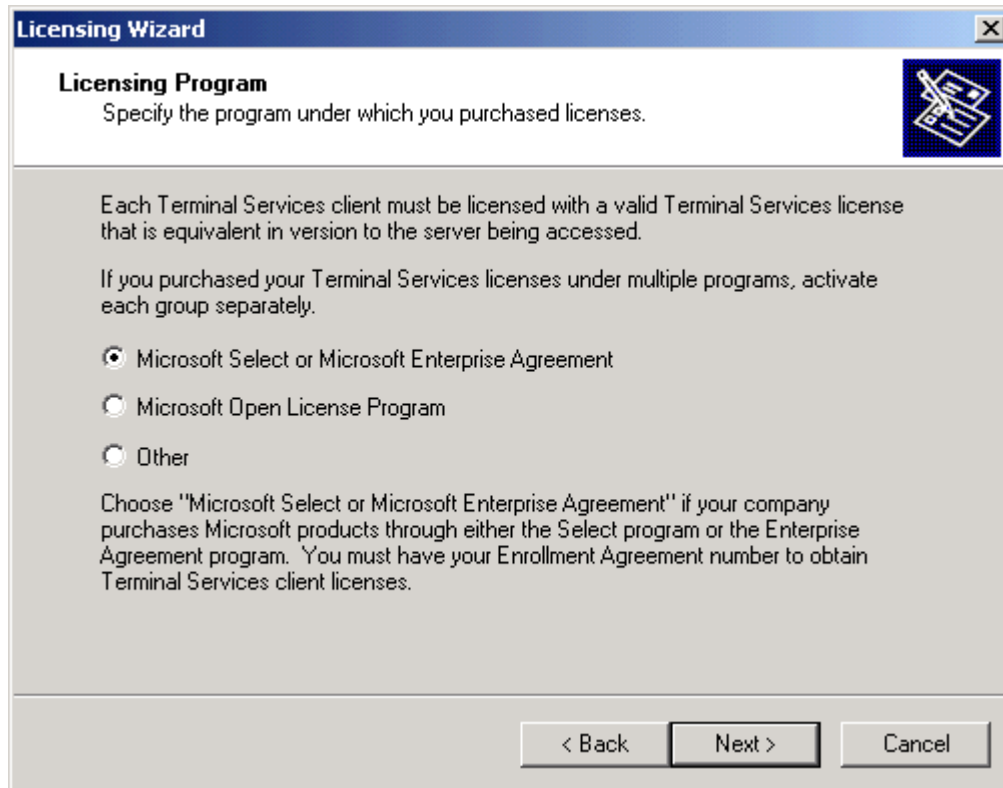



Figure 2.12: Selecting your licensing program.

After you complete the wizard, you'll be emailed, faxed, or told a 35-character PIN. After you enter the PIN, your Terminal Services Licensing service is active and ready to accept license packs.

 If you're using the telephone connection method, I recommend typing the PIN and license codes into the wizard while the representative is still on the line to be sure that you enter it without errors. If you write it down incorrectly, you'll have to call again.

Installing Licenses

After activation, the server will immediately begin handing out permanent CALs to Win2K devices. To issue permanent CALs to non-Win2K clients, you'll need to install license packs on the Terminal Services Licensing service. To do so, use the licensing wizard in the Terminal Services Licensing application.

The license program and connection method you selected when activating the server determines how you obtain license pack codes. If you need to change the connection method or license program, you can do so in the properties of the server in the Terminal Services Licensing application. The licensing wizard will ask you for the license code, then activate the number of CALs included in the code, as Figure 2.13 shows.

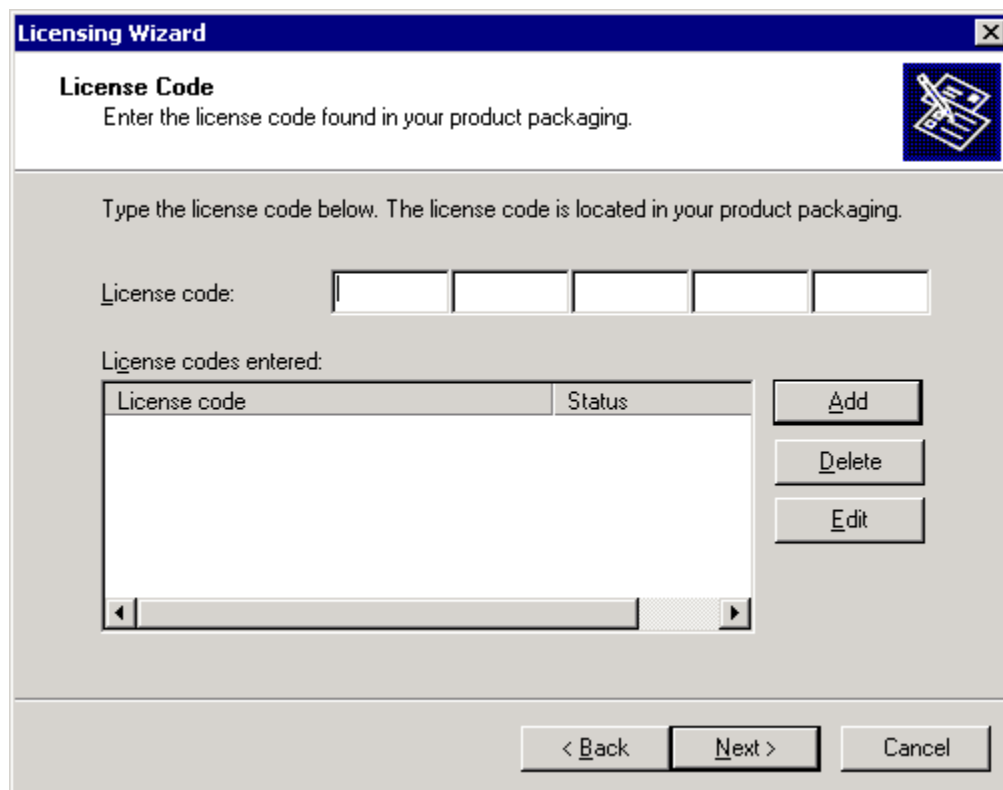


Figure 2.13: The Terminal Server Licensing wizard.

After CALs are installed on the server, non-Win2K clients will be issued a permanent license upon their second connection to the terminal server and Win2K clients will have their built-in license activated.

How Does the Terminal Server Find the License Server?

The Microsoft article “Description of Terminal Services License Server Discovery” at <http://support.microsoft.com/support/kb/article/q232/5/20.asp> explains how a terminal server finds a license server. The method used is different depending on the environment.

Win2K Domain

- The Terminal Services License Service must be installed on a Win2K domain controller in the domain.
- The Terminal Services server must be running on a Win2K domain controller or a Win2K member server with one or more Win2K domain controllers in the same domain.
- The Terminal Services-based computer looks for the license server by using a remote procedure call (RPC) to all Win2K-based domain controllers in the same domain and querying them for the Terminal Services Licensing service. The client chooses one of the license servers at random, then requests a license key pack from that license server. The license server passes the request to the enterprise license server if no license key pack is available.


- If the Terminal Services server cannot find a license server in the domain, the Terminal Services server queries the directory service for the enterprise license server.

NT 4.0 Domain

- The license server must be running on a Win2K member server and all domain controllers must be running NT 4.0.
- The Terminal Services-based computer issues broadcasts on a mailslot. All Terminal Services license servers that receive the broadcast respond. The Terminal Services-based computer selects one of the licenses servers at random.

Workgroup Environment

- The Terminal Services server and licensing server are in a workgroup.
- The Terminal Services-based computer issues broadcasts on a mailslot. All Terminal Services licensing servers that receive the broadcast respond. The Terminal Services-based computer selects one of the license servers at random.

 If you're running Terminal Services in a separate but trusted domain or forest, you can instruct your Terminal Services servers to use a specific license server by adding the following registry value:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\TermService\Parameters\DefaultLicenseServer

Create this value as a REG_SZ value and give it the NETBIOS name of the license server. The server needs to be able to find the Terminal Services license server by its NETBIOS name, so you may need to use an LMHOSTS file if you're not running WINS in your NT 4.0 domain.

Installing the Terminal Services Client

To connect to the terminal server, the Terminal Services Client is used. This client is available for a number of platforms:

- Win32 x86 OSs—Use this client to connect to the terminal server from a Windows 9x, NT 3.51, NT 4.0, or Win2K PC. This client provided on the terminal server.
- Win16 OSs—Use this client to connect to the Terminal Services server from a Windows 3.11 client. This client is also available on the terminal server.
- Windows Millennium Edition (Windows Me)—The Terminal Services client for Windows Me is included on the Windows Me CD-ROM.
- Windows CE Professional—These clients are used to connect to the Terminal Services server from a handheld device running CE Pro. This client can be found at <http://www.microsoft.com/mobile/downloads/ts-final.asp>.
- Microsoft Advanced Client—This client is an ActiveX-based client that will install and run from within Internet Explorer (IE) 4.0 or later on any 32-bit Windows OS.

- Windows Based Terminals (WBT) —Also called thin-client devices, these are dedicated terminals used to replace a desktop computer when using Terminal Services. These devices typically run a form of Windows CE.

Setup.exe-Based Installation

There are several methods used to install the Terminal Services Client. For the Win32 and Win16 clients, you can use the Terminal Services Client Creator Utility, which Figure 2.14 shows, in Administrative Tools to create installation diskettes to distribute to your end users. The 16-bit client will require four diskettes and the 32-bit client will require two.

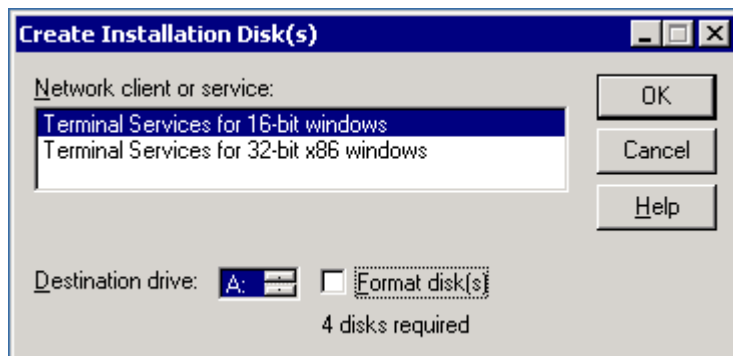



Figure 2.14: The Terminal Services Client Creator utility.

If you're distributing the client to workstations on your LAN, you can also copy the contents of the C:\winnt\system32\clients\tscient\net directory to a network share and install the client over the network. If you would like to automate the installation of the client, use one of the following switches:

- SETUP /Q—Installs the client with default options, but displays the progress gauge and the Exit dialog box when installation is complete.
- SETUP /Q1—Installs the client with default options. This switch will display the progress gauge but will not show the exit dialog box.
- SETUP /QT—Performs a completely silent and invisible installation with default options. No progress gauge or exit dialog box is shown.

Windows Installer-Based Installation

If you're installing the client on Win2K computers or have installed the Windows Installer on your Win9x or NT computers, you can also use the MSI package that Microsoft provides on the Terminal Services Advanced Client Web page. This package greatly simplifies installation if you're in a Win2K domain because you can easily publish the client through Group Policy.

 The MSI version of the 32-bit Terminal Services Client can be found at <http://www.microsoft.com/windows2000/downloads/recommended/TSAC/tsmsi.asp>.

After downloading the MSI client, you can run the setup program to extract the MSI package to a directory you specify. The default directory is C:\Program Files\Terminal Services Client MSI.

But don't let this process mislead you into thinking that you're installing the client itself; you're only extracting the files needed to perform the installation.

Handheld Device Installation

To install the Terminal Services Client on your WinCE Pro device, download the client from Microsoft and follow the instructions.

☞ At this time, Microsoft hasn't released a Terminal Services client for Pocket PC. To work around this limitation, download the WinCE Pro client and follow these instructions:

1. Put your Pocket PC in its cradle.
2. Run the HPCRDP.EXE installer program.
3. The installer will fail but .CAB files will be extracted to C:\Program Files\Windows CE Services\Terminal Server Client for Windows CE, Handheld PC Edition.
4. Copy to your Pocket PC the .CAB file that matches your Pocket PC's processor (MIPS = Casio, SH3 = HP, SA1100 = iPaq, MIPS = Aero).
5. Using the Windows Explorer program on your Pocket PC, click the .CAB file to start the installation process.
6. You'll be warned that the program isn't for your OS; just tell it to continue.

Note the .CAB file places the shortcut in the wrong location. Use file explorer to move it from \windows\programs to \windows\start menu\programs.

There is, however, a Terminal Services client for PocketPC 2002. If you can upgrade your device to 2002, you will have the new client.

Configuring Client Connections

The 16-bit and 32-bit clients have two components—the Terminal Services Client, which Figure 2.15 shows, and the Client Connection Manager. The Terminal Services Client is used to browse for servers in your domain or workgroup and connect to them by requesting a desktop session. The Client Connection Manager is used to store custom connections to servers or applications on Terminal Services servers.

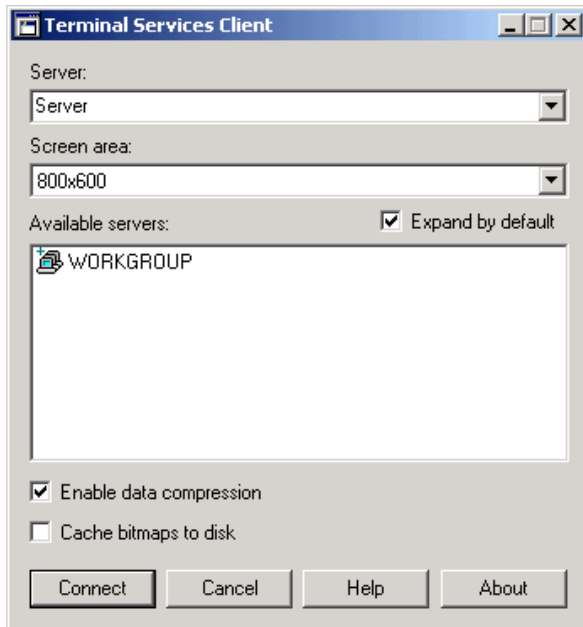


Figure 2.15: The Terminal Services Client.

In the Terminal Services Client window, you can type or browse the server you want to connect to, then specify the screen size you want your session to have. You then click Connect to start the session.

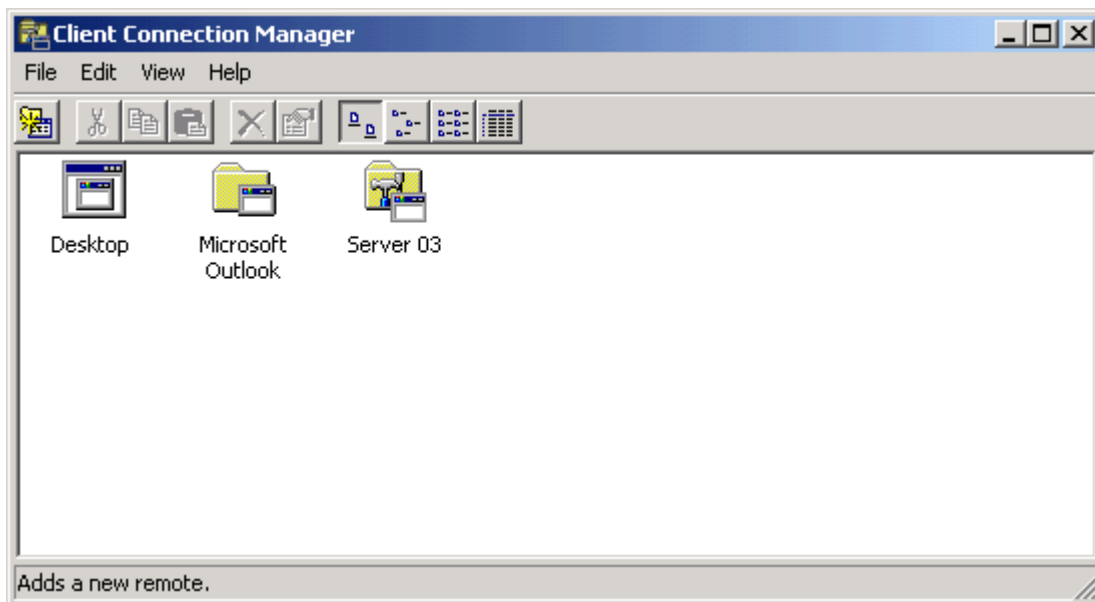



Figure 2.16: The Client Connection Manager.

In the Client Connection Manager, which Figure 2.16 shows, you use the New Connection button to open a wizard that will walk you through defining a connection to a server. The wizard will ask for the server name and the session screen size. It will offer to cache credentials to log onto the server, and ask if you want to launch a specific program rather than a complete desktop.

Icons for any connections you create here are automatically added to your Start menu in the Terminal Services Client folder.

If you want to deploy the Terminal Services Client with connections already configured for your users, you would first install the client on a test machine and configure in the Client Connection Manager the connections you want your users to have. Then under the File menu select Export All (or Export if you only want to deploy one of the connections listed in the manager). This action will create a CNS file that you can import to your users' Client Connection Manager.

 You can write a script that will silently install the Terminal Services Client and import the CNS file that you have created. See the Microsoft article "Terminal Services Client Command-Line Switches" at <http://support.microsoft.com/support/kb/articles/q255/8/98.asp> for the command-line switches needed to silently import connections. An example installation script will be given in the Appendix.

The Microsoft Terminal Services Advanced Client

The Terminal Services Advanced Client provides the same functionality as the 32-bit client, but offers many advantages if you're working in a Web-centric environment.

- Users don't have to manually install the client. It is automatically installed when the user goes to a Terminal Services Advanced Client-enabled URL.
- Administrators can quickly change a Web page to point users to a new or updated application on the same or different server.
- Users or administrators can roam to a different desktop and quickly access an application or desktop by simply knowing a URL.
- If the Terminal Services Advanced Client is ever updated, users will automatically pick up the new version when they navigate to the Web page.

You must install the Terminal Services Advanced Client on an IIS server running version 4.0 or later. This server doesn't have to be (and in most cases, should not be) a terminal server itself. To install the client, simply download the package from <http://www.microsoft.com/windows2000/downloads/recommended/TSAC/default.asp> and run it on your IIS server. The installation wizard will ask you for a Web directory to install the client to and will add the ActiveX control, the ActiveX Client Control Deployment Guide, and sample Web pages to the directory. The default Web site provided by the package, which Figure 2.17 shows, asks the user for a Terminal Services server name and a screen size to open the session.

With some skilled HTML and ActiveX programming, you can create custom Web pages that instantly connect your users to specific servers and applications by simply going to a URL that you provide. In addition to the RDP virtual channel, Terminal Services Advanced Client provides an additional virtual channel that can be used to provide OLE functionality between applications installed locally on the user's workstation and applications running on a Terminal Services server.

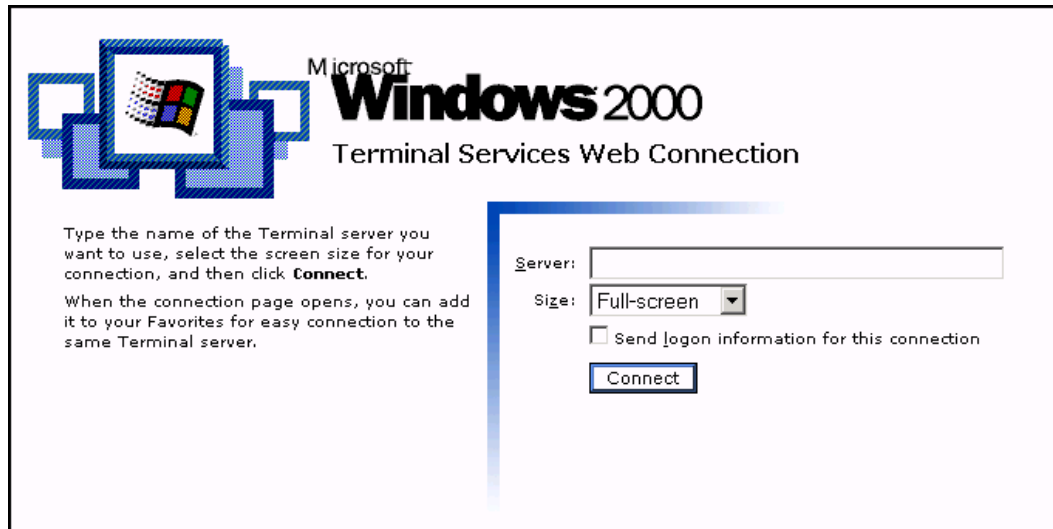



Figure 2.17: The default Web site created by Terminal Services Advanced Client.


 New Moon's Canaveral IQ leverages the Terminal Services Advanced Client to allow administrators to easily deploy, configure, and publish applications over the Web. Read more about this product at <http://www.newmoon.com/products/>.

Windows-Based Terminal Clients

Windows-based terminals (WBTs) provide instant access to terminal servers. With these devices, you can replace a user's desktop or provide KIOSK style access to applications on your Terminal Services servers. Most WBT manufacturers offer centralized management utilities to allow administrators to remotely configure these units for connection to specific servers and applications. Other than configuring the connections, these devices require little, if any, installation work.

There are many players in the WBT market. To compare products across manufacturers, go to <http://www.thinplanet.com/products/>. ThinPlanet focuses on thin-client and WBT technology and is a great resource for WBTs.

If you like the idea of a WBT, which allows easy access to a Terminal Services desktop or application, but you're not interested in purchasing a thin-client device, you can configure an NT or Win2K workstation to function as a WBT. This setup will give you the ability to test the WBT model and recycle older PCs.

 See Appendix A for detailed instructions about how to configure a workstation-based WBT.

Third-Party Clients

Microsoft doesn't provide RDP clients for non-Windows OSs—UNIX, Linux, Macintosh, and so on. If you're considering deploying Terminal Services in an environment in which you have multiple client OSs, you may want to take a serious look at Citrix MetaFrame, as this product supports clients for these platforms. If running the ICA protocol isn't an option, you can look at <http://www.hobsoft.com/> and test their Java-based RDP client, which may meet your needs.

Summary

In this chapter, I went over the installation and configuration of Terminal Services in both remote administration mode and application server mode. You're now ready to build a Win2K Terminal Services server. I also provided information about the various client applications that allow users to connect to a Terminal Services server.

In addition, you learned how to install, configure, and add licenses to a Terminal Services license server, which will manage TSCALs for your client devices. Be sure to include this step in your deployment plan, as without it, your Terminal Services servers will stop accepting connections after 90 days.

Now you're ready to dive into the real purpose of Terminal Services—providing applications. Chapter 3 will take you through the steps involved in installing an application on a Terminal Services server and tuning it for multi-user access. You'll also learn that scripting is a required skill for Terminal Services administrators, as you will use logon, logoff, and application-compatibility scripts extensively in a Terminal Services environment. So if you're not comfortable with scripting—in Shell Script, VBScript, KIX, or Jscript—you may want to start studying one or more of these languages.

Before deploying your Terminal Services servers, be sure to read Chapter 4 as well. This chapter will teach you how to manage your servers and Terminal Services-enabled users. You need a thorough understanding of how to limit access to Terminal Services to only authorized users as well as how to configure your users' accounts for Terminal Services.

Now is a great time to set up a test Terminal Services server. Familiarize yourself with the tools explained in this chapter, install one of the client applications, and make your first Terminal Services connection.


Chapter 3: Deploying and Managing Applications

In Chapter 2, you learned how to install, configure, and tune Win2K Terminal Services, so you're now ready to begin installing applications. If you're new to Terminal Services, you're lucky in that most current applications adapt very easily to the multi-user environment, so you probably won't need to spend hours analyzing registry changes and dissecting INI files. But you should become very familiar with the mechanisms involved in Terminal Services application compatibility so that you'll be prepared to troubleshoot a touchy application or integrate that one mission-critical legacy application that your company needs.

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, you must understand the mechanisms that Microsoft has put in place to assist you in making applications that weren't designed with Terminal Services in mind run on a terminal server. These include Terminal Services logon scripts, application compatibility scripts, install and execute modes, registry mapping, and INI file mapping.

 If an application carries the Certified for Windows 2000 logo, it will generally be compatible with Terminal Services. Microsoft's certification program assures that the application follows certain guidelines in its installation process, storage of data and settings, and is compatible with Win2K features. For more information about the certification program, see <http://msdn.microsoft.com/certification/appspect.asp>.

Terminal Services Logon Scripts

In addition to your Windows domain logon script, 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 3.1 illustrates the user-logon process.

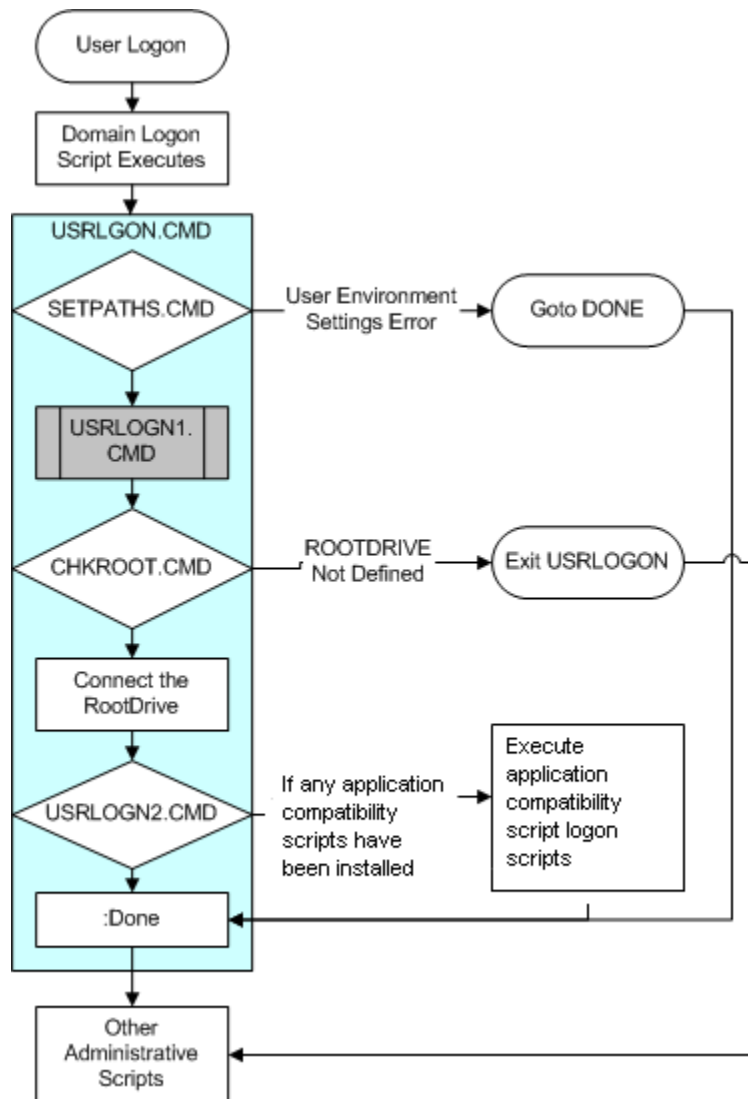



Figure 3.1: The user-logon process.

When you install Terminal Services, the system adds an entry to the `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\A ppsetup` 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 subscrips and explain a number of Terminal Services concepts along the way.

 All the scripts that Microsoft provides are written in shell script, but Win2K 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 3.1 shows this section.

```

REM USRLOGON.CMD
@Echo Off

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

```

Listing 3.1: 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 the all user variables are in the same subkey but under HKEY_LOCAL_MACHINE. If any of these variables aren't defined, a warning message is displayed, and USRLOGON.CMD aborts. Table 3.1 outlines 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 3.1: User environment values.

Next, as Listing 3.2 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 doesn't require a ROOTDRIVE. An example of this type of application compatibility script would be one that makes changes to the HKEY_CURRENT_USER subkey without referencing 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 3.2: 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 we can't use environment variables in this registry value, we need an absolute path that can be resolved for all users. Therefore, on a terminal server, we define a ROOTDRIVE.


USRLOGON.CMD uses a SUBST command to connect the ROOTDRIVE letter directly to the user's home directory upon logon. Listing 3.3 shows this section of USERLOGON.CMD.

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

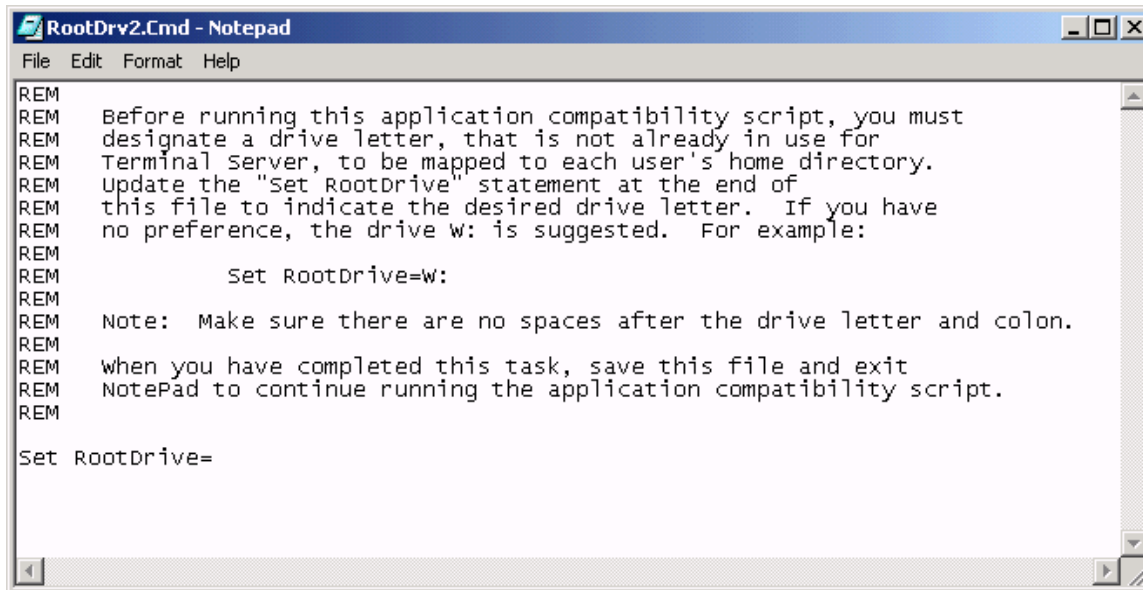
Rem
Rem Map the User's Home Directory to a Drive Letter
Rem

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 3.3: 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 does the ROOTDRIVE letter get 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 3.2 shows. After this reservation is set, the letter can't be used for any other mapping on the terminal server.



```

RootDrv2.Cmd - Notepad
File Edit Format Help
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 3.2: Setting the ROOTDRIVE letter.

The process that Microsoft uses to connect to the ROOTDRIVE was written for WTS, and doesn't take advantage of Win2K's enhanced drive-mapping capabilities. To explain this concept, let me walk you through two examples of how ROOTDRIVE works under WTS, 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 gets 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 link the W drive to %HOMEDRIVE%%HOMEPATH% or H:\Joe.User. Now the path W:\ is an alias for H:\Joe.User, and the registry can reference W:\ whenever we need to get directly to Joe's home directory. In addition, we can use this path 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 Win2K has the ability to map to the subdirectory itself, which would enable us to use 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 you can't take advantage of Win2K enhanced drive-mapping capabilities. Also, there is a bug in Win2K that causes an error message whenever you try to empty the Recycle Bin if you have any drives created by SUBST. There is a hotfix available for this bug, but 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 Win2K'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 need the SUBST hotfix, it can be found at <http://support.microsoft.com/support/kb/articles/q297/7/60.asp>, and 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/support/kb/articles/q195/9/50.asp>.

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 3.4 shows my suggested change to USRLOGON.CMD.

```
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 3.4: 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 3.5 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 3.5: 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 may want to run another logon script when your users log on to a terminal server. For example, you may want to create scripts that map additional network drives or connect to specific printers. To have the system execute an additional script, copy the script file to the \system32 directory of your terminal server, and modify 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 additional scripts, separated by commas. For more information, see the Microsoft article “How to Set Up a Logon Script Only for Terminal Server Users” at <http://support.microsoft.com/support/kb/articles/q195/4/61.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 certain guidelines laid out in the Application Specification for Windows 2000. This document instructs programmers to take advantage of a number of Windows component services

that would make the application natively compatible with Win2K 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 may 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 into Windows has his or her own HKEY_CURRENT_USER hive.

Applications that don’t use advertising typically write HKEY_CURRENT_USER registry information only into the hive belonging to the user that installed the application, so when another user logs into Windows, crucial registry settings may be missing from HKEY_CURRENT_USER. This shortcoming is usually not an issue 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 3.3 shows, any keys that are written to HKEY_CURRENT_USER are automatically copied to a special location under HKEY_LOCAL_MACHINE. They’re copied to the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Terminal Server\Install\Software subkey.

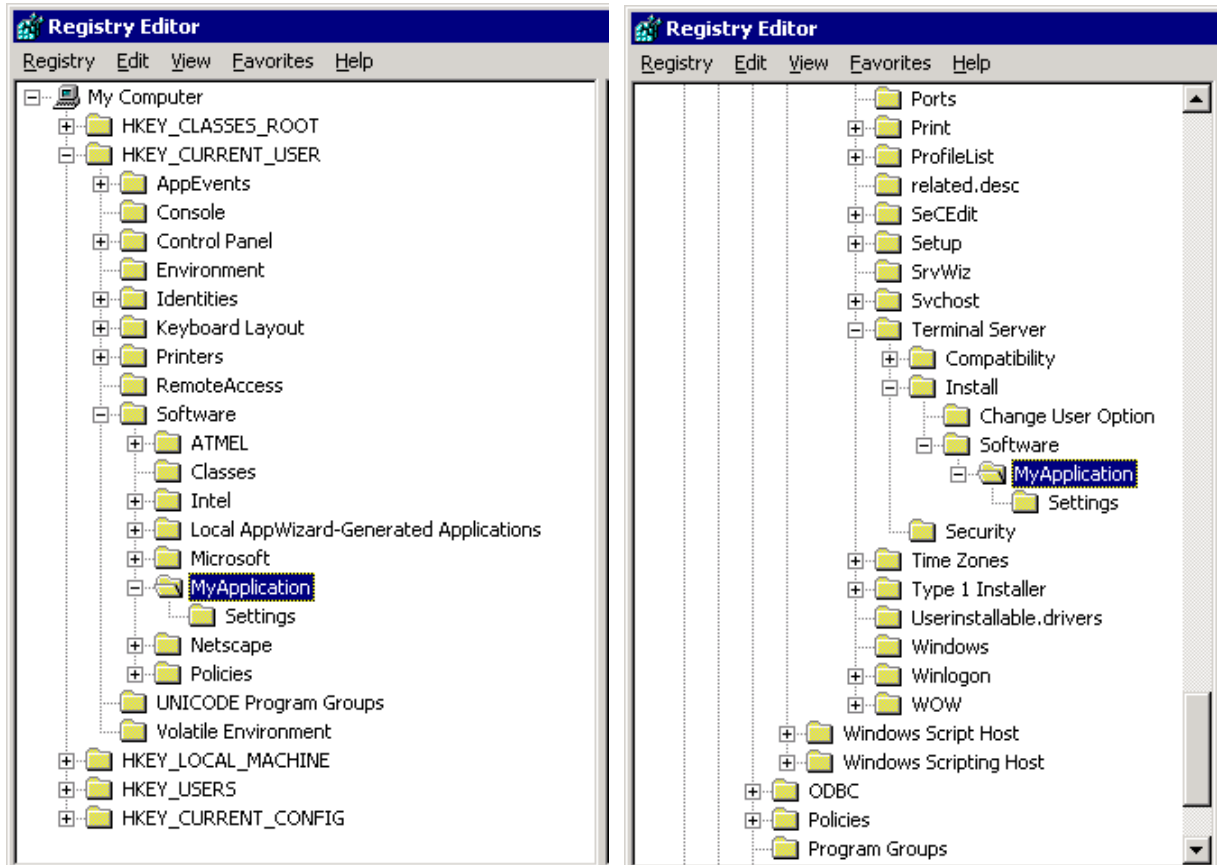



Figure 3.3: Registry mapping in install mode for a program called *MyApplication*.

-  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 may 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 if 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 3.4 shows.

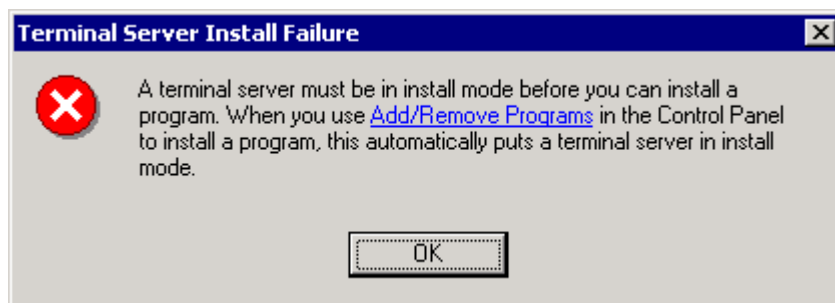



Figure 3.4: Attempting to run an installation program outside the Add/Remove Programs Control Panel applet.

 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 onto 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.

Microsoft provides application compatibility scripts for a number of common applications, as Figure 3.5 shows. You can find these scripts in `C:\WINNT\Application Compatibility Scripts`. The scripts are divided into three groups:

- **Install**—These scripts make system-level changes and add entries to `USRLOGN2.CMD` if the application also requires a logon application compatibility script for each user.
- **Logon**—These scripts are called by `USRLOGN2.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.

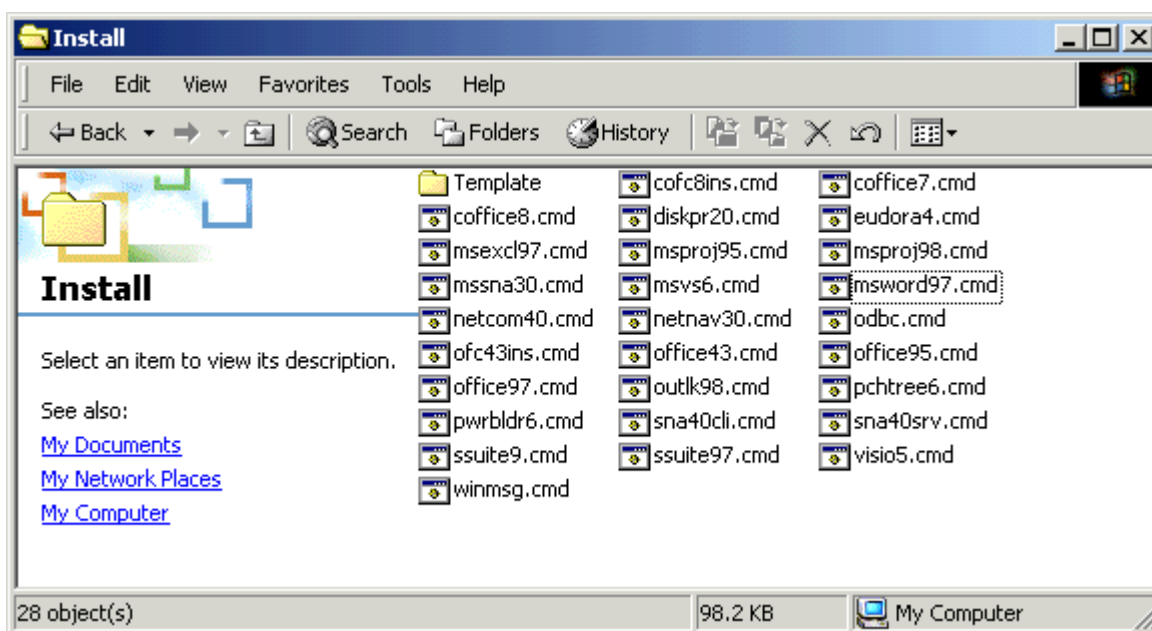



Figure 3.5: Application compatibility scripts provided by Microsoft.

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.

 As you can see in Figure 3.5, most of the application compatibility scripts that Microsoft provides are for older applications. If you're installing a non-logged-compliant application that doesn't have a Microsoft-provided application compatibility script, see "Undocumented Application Installation" 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

I'll walk you through the installation process for three applications: First for an application that has no problems running on Terminal Services, then for an application that comes with a special installation method for Terminal Services, and finally for an application that requires an application compatibility script.

Simple Installation

Adobe Acrobat Reader is a free utility used to open PDF files. This utility has no problems running on Terminal Services. So to install it, you simply use the Add New Programs Wizard in the Add/Remove Programs Control Panel applet, which Figure 3.6 shows. This wizard places the server into install mode.

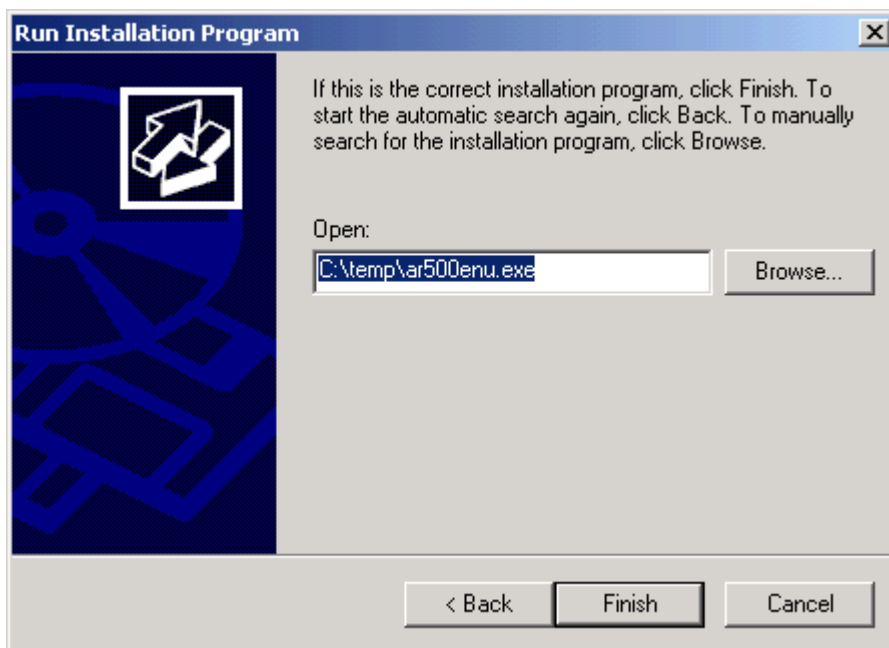


Figure 3.6: The Add New Programs Wizard.

After the installation is complete, click Finish. The server is now back in execute mode and Acrobat Reader is ready for your Terminal Services users. As an exercise, you should 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 Reader for the first time, the system will copy the keys into his or her HKEY_CURRENT_USER registry hive for you.

Custom Installation

Some software manufacturers take Terminal Services into account when writing their programs, and may provide you with instructions about how to install their application 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 program 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 you for your name and initials during setup, so all users of the terminal server would end up with your name. We need to enable the NOUSERNAME switch to force Office to ask each user for his or her name on first use of Office.
- **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 can be downloaded 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, then select 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 3.7 shows.

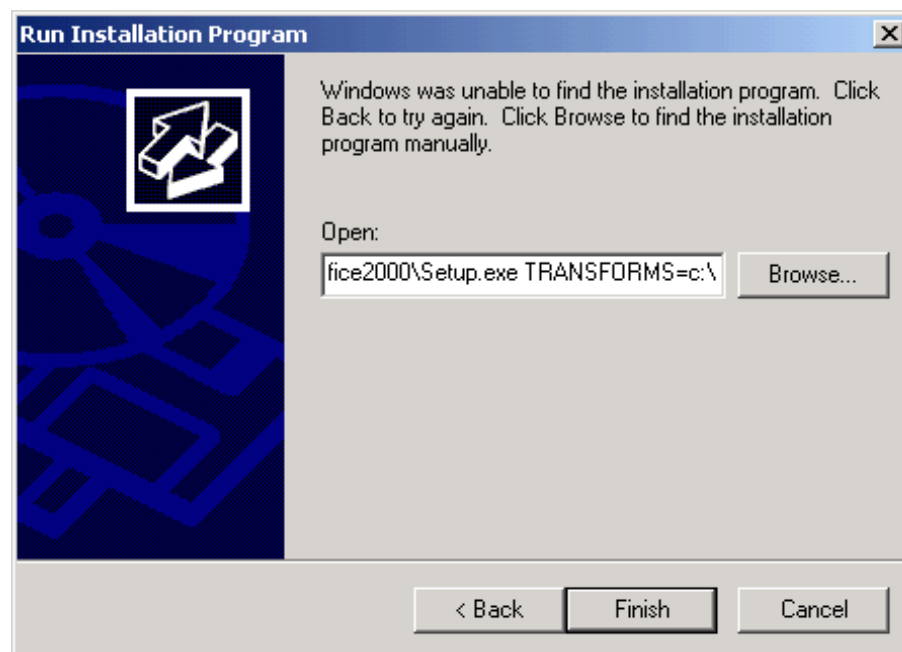


Figure 3.7: Installing Office 2000 with the Terminal Services transform.

After the installation is complete, click Finish to put the terminal 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 in the latest service release.

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. In Chapter 4, I'll discuss using Group Policy to manage your user settings in a Win2K domain.

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 3.8 shows, by typing

```
GPEDIT.MSC
```

in the Run text box.

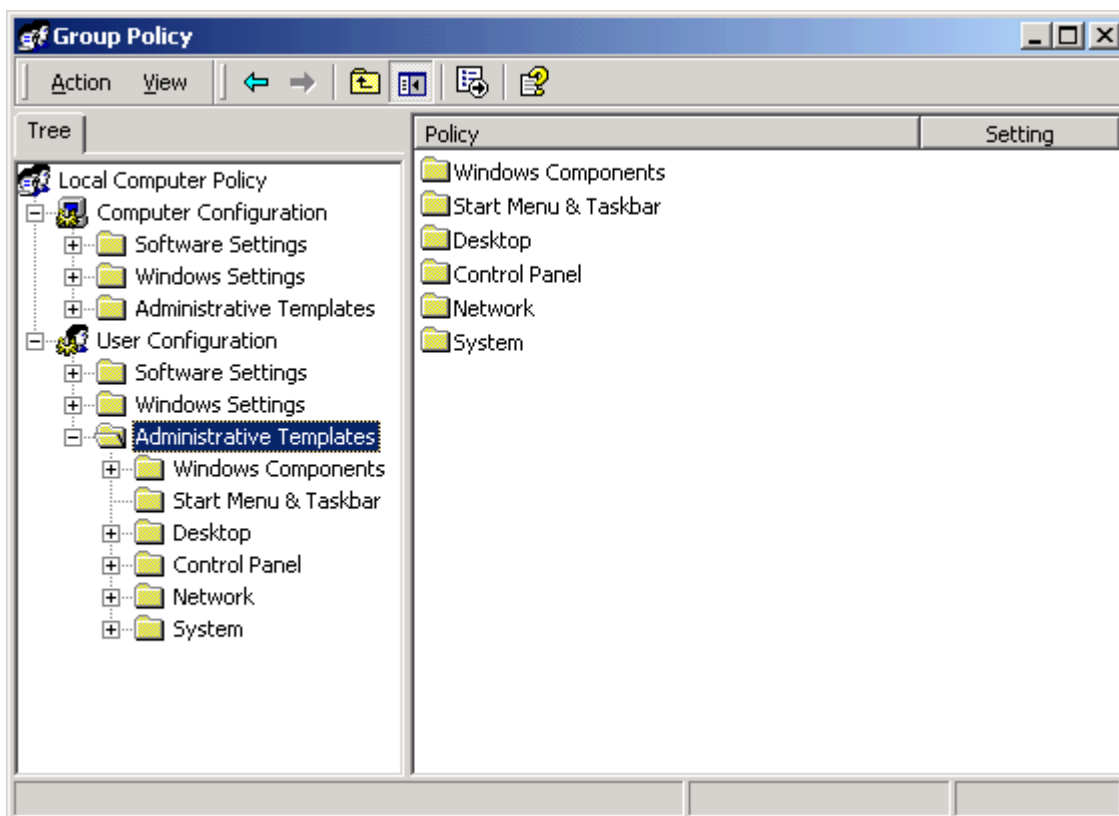


Figure 3.8: The Group Policy editor.

Next, right-click Administrative Templates, and select Add/Remove Templates. Add the ADM files for the programs that you have installed. Table 3.2 lists the ADM file 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 3.2: 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 3.9 shows.

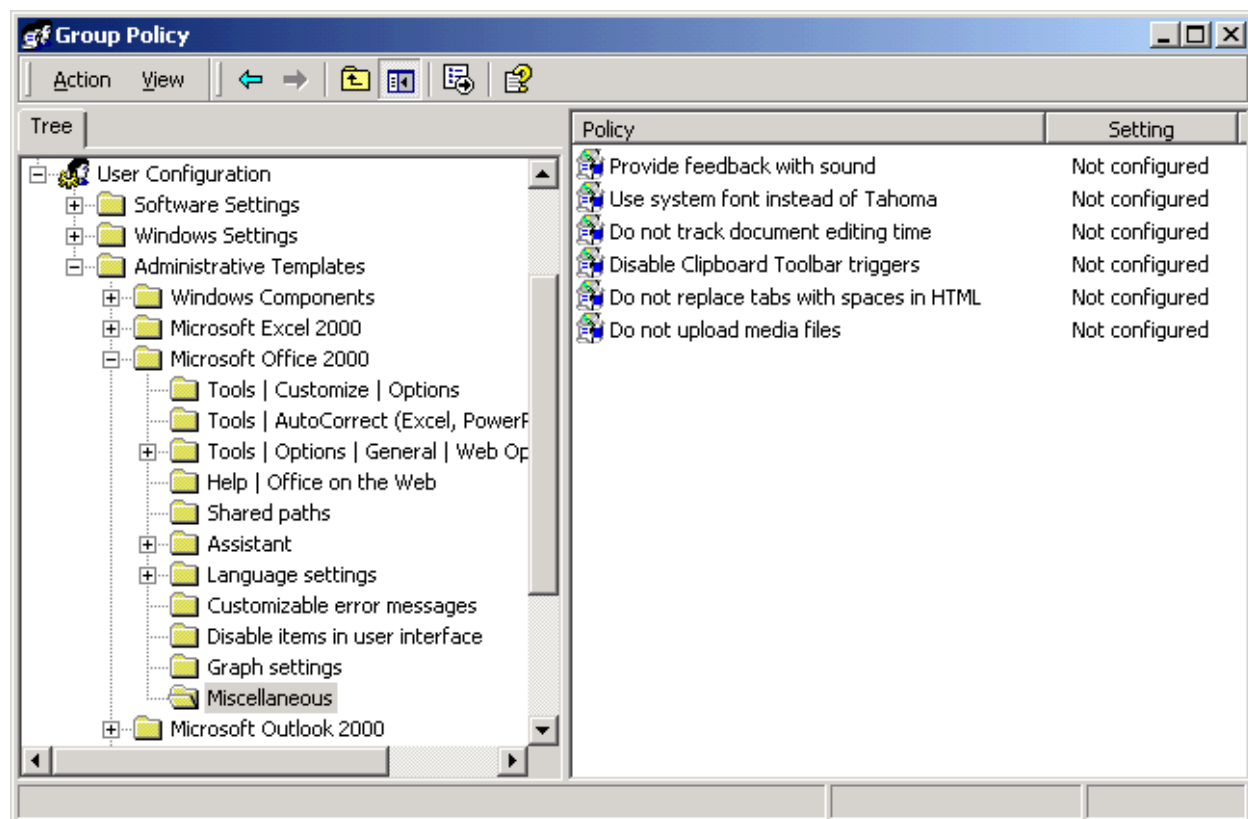


Figure 3.9: Settings available with the Office ADM file.

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 may 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.

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, 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 Navigator 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:

1. Checks to see whether you've defined your ROOTDRIVE letter and asks you to do so if you haven't.
2. Copies the Quick Launch icon from your profile to the Netscape program directory as a template for future users.
3. Copies the Netscape profile directory that the setup program created for you to be a template for other users.
4. Applies restrictive permissions to the Netscape profile-manager utility to prevent non-administrators from running it.
5. Modifies the application compatibility script logon script for Netscape to reflect the ROOTDRIVE letter that you have chosen.
6. 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 compatibility script logon script for Netscape) is called and performs the following actions:


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

Netscape is now ready for simultaneous users.

Undocumented Application Installations

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 need to be made or an application compatibility script needs to be written. This skill is an important one to have, but it's becoming 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 may 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.deja.com>

<http://www.thethin.net>

<http://www.thinplanet.com>

After collecting any information you find about your application, you should 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, you should 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, you should 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.

Now, look at the HKEY_CURRENT_USER\Software key and see if 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 to see whether the keys were mirrored in the Terminal Server section of the HKEY_LOCAL_MACHINE key. 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 onto 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, follow these steps:

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 modify 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. But take heart in the fact that you need to go through this process only for applications that aren't certified for Win2K.

Real World Example

Let me walk you through an example of an application that needs to be modified for terminal server use. I'll change the name to protect the publisher.

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 user's run it from within a Web page (using the Microsoft Terminal Services Advanced 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.

You begin by using the Add New Programs Wizard to run SETUP.EXE for WorkGroup. The installer doesn't require a reboot, so you immediately go to regedit, and examine HKEY_LOCAL_MACHINE\SOFTWARE for absolute paths under WorkGroup's key, as Figure 3.10 illustrates.

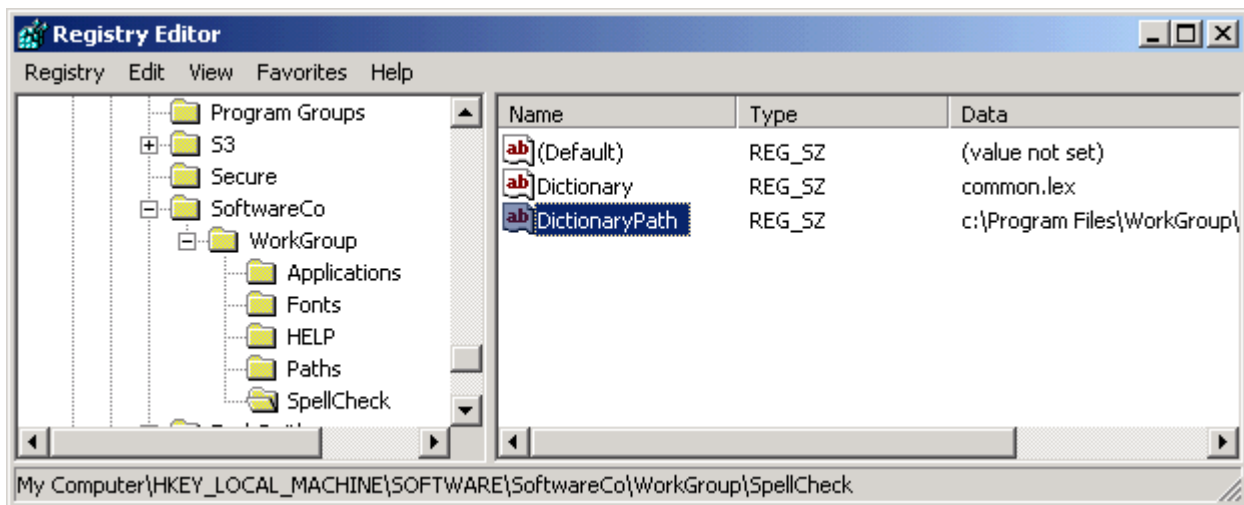


Figure 3.10: HKEY_LOCAL_MACHINE key for the WorkGroup application.

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. But after closer inspection, this value looks like it's the base lexicon file that all users will share, not a per-user file. You make a note of this key, just in case.

Now you 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 3.11 shows.

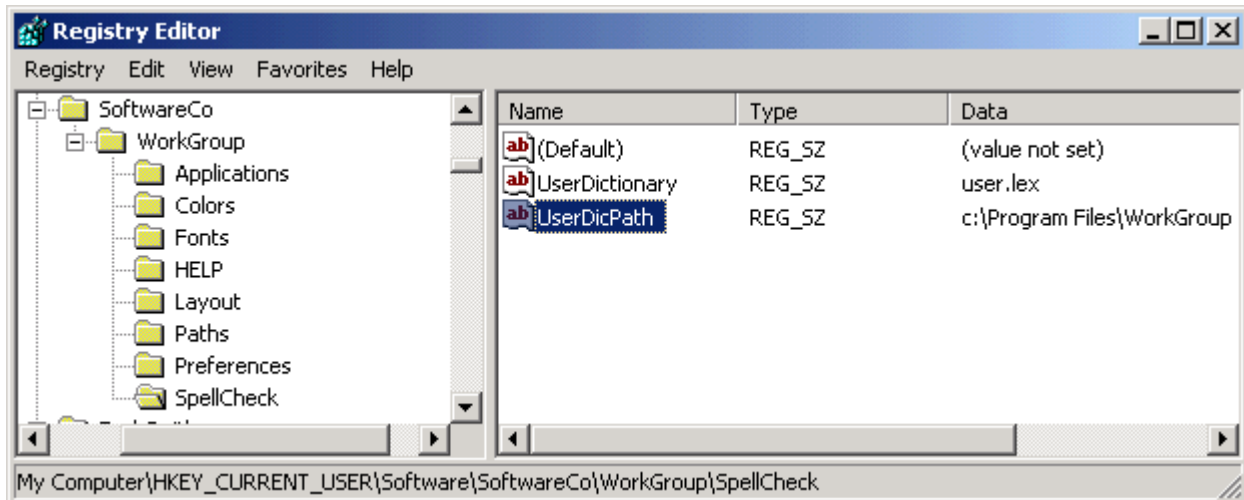


Figure 3.11: Example of a per-user file stored in a common location.

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\Install\Software key, as Figure 3.12 shows.

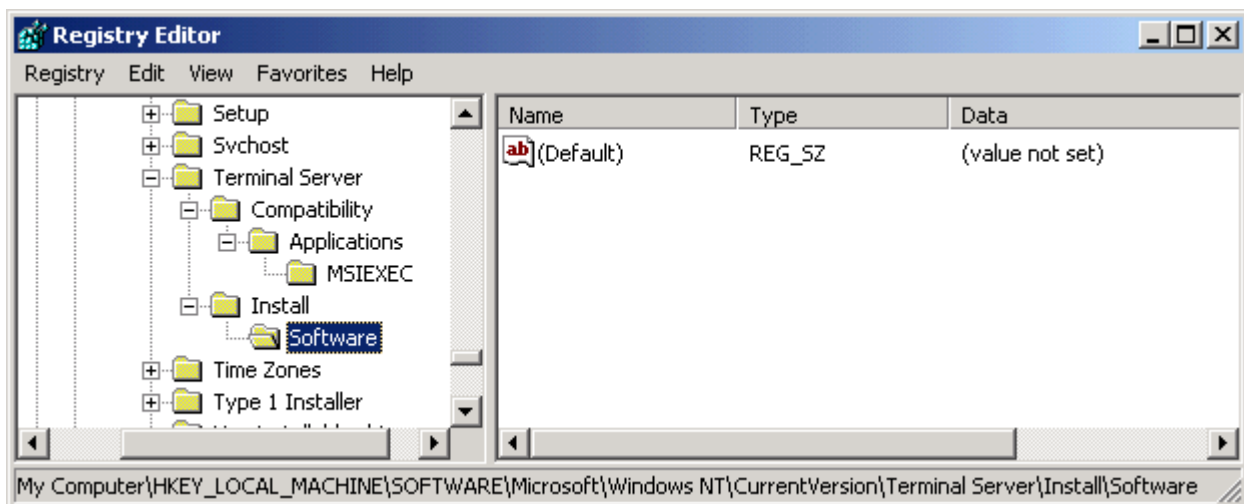


Figure 3.12: The location for mirrored registry keys.

As you see in Figure 3.12, 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 and see whether it will run under our installer account. Let's assume that you don't encounter any problems. We'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, you need to 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 H, so the change was successful.

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


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

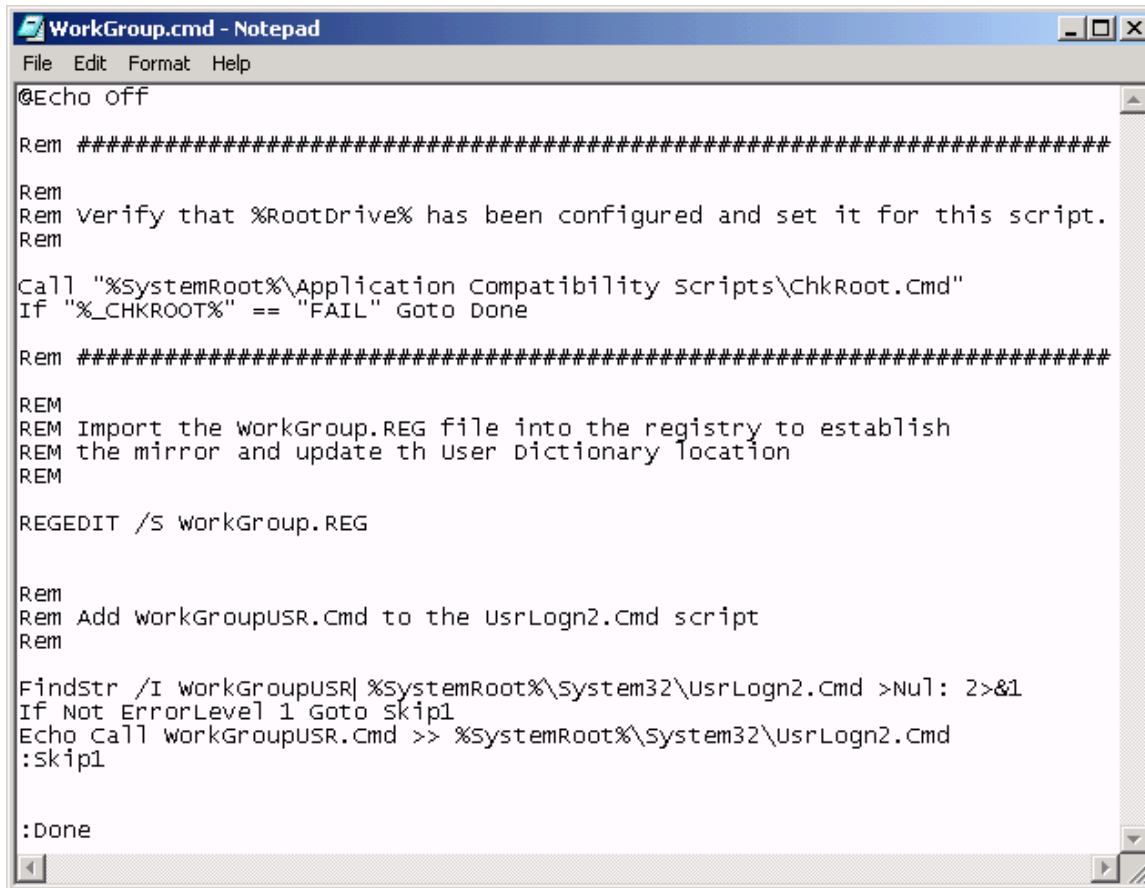
You should be able to make the registry change through registry mapping, but you've already discovered that you'll have to manually set up this mapping. The file copy will require an application compatibility script. Let's take each of these tasks in turn.

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 3.13 shows an example application compatibility script installation script.

 Manually importing the .REG file and editing USRLOGN2.CMD 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.



```

WorkGroup.cmd - Notepad
File Edit Format Help
@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

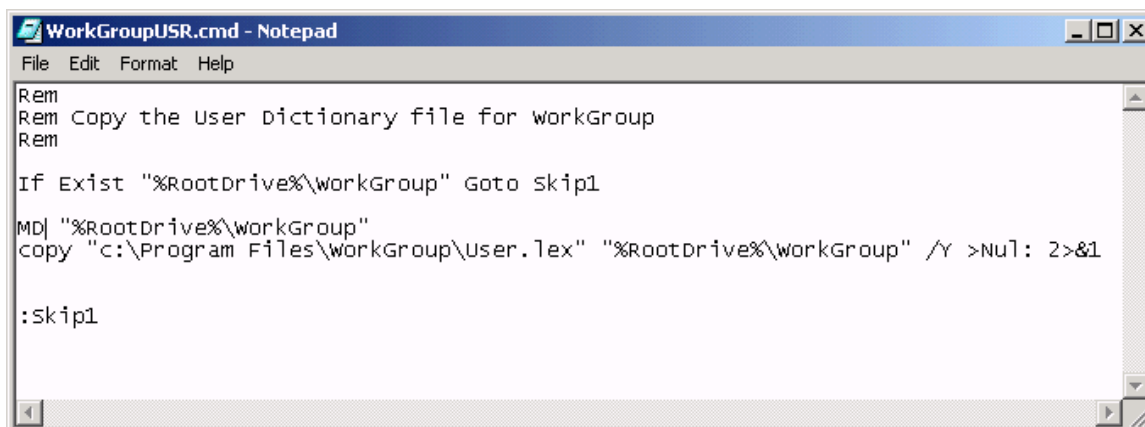
FindStr /I workGroupUSR| %SystemRoot%\System32\UsrLogn2.Cmd >Nul: 2>&1
If Not ErrorLevel 1 Goto skip1
Echo Call workGroupUSR.Cmd >> %SystemRoot%\System32\UsrLogn2.Cmd
:skip1

:Done

```

Figure 3.13: 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. Therefore, be sure to include logic in your scripts to check this. This script file should be saved in C:\WINNT\Application Compatibility Scripts\Logon. Figure 3.14 shows an example application compatibility script logon script.



```

WorkGroupUSR.cmd - Notepad
File Edit Format Help
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 3.14: 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 3.15 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 adjust this flag so that Terminal Services makes adjustments when the application is launched.

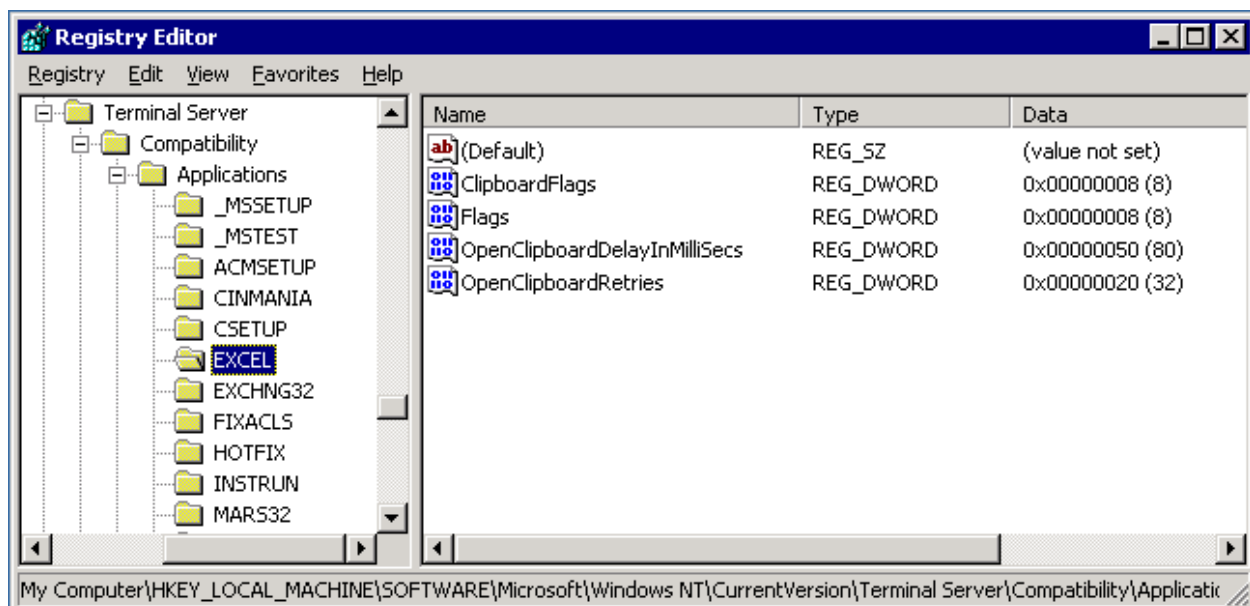


Figure 3.15: The compatibility flags registry values.

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 here.


 If you have a program that will not run on Terminal Services, read the Microsoft article “Terminal Server Registry Settings for Applications” at <http://support.microsoft.com/support/kb/articles/q186/4/99.asp>.

Deploying Applications

To deploy a new application to your users if you’re using Terminal Services as a desktop replacement, you simply add 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 terminal server client, you’ll need to create a CNS file that contains the server name and initial application path for the new application, and import the file into the Client Connection Manager on your users’ workstations. This import process can be automated by using a domain logon script.

If you want to deploy an application through the Microsoft Terminal Services Advanced Client so that your users will go to a URL to launch the program, you should reference the documentation that Microsoft provides with the advanced client to customize the Web page that hosts the application. The advanced 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.

 **New Moon’s Canaveral IQ product will create for your users a dynamic Web site that lists all ASP-based applications that you assign to them. This powerful add-on to Terminal Services enables administrators to easily deploy new applications and even provides load balancing so that you can host a program on more than one terminal server.**

Managing the Application Life Cycle

If you manage a large number of terminal servers, installing applications should be automated so that installation, upgrade, and retirement can be done smoothly with little user interruption. There are a number of software-management systems available, from Win2K’s native IntelliMirror abilities to SMS to third-party software suites. Whichever method you choose, be sure to confirm that the deployment mechanism takes Terminal Services into account so that the server can be placed into the proper mode for application installation and execution.

 **Timbale by Marimba is a software-management system designed specifically for terminal servers. This utility can be used to create automated software-installation packages, and install them across multiple terminal servers. See http://www.marimba.com/products/change_management/server/WTS.html for more information.**

Summary

In this chapter, I went over the mechanisms that Terminal Services uses to host applications for simultaneous users even when the application itself was not designed to handle them. I examined the Terminal Services logon script process and administrative modes and explained application compatibility scripts. I defined the ROOTDRIVE and explained how this concept can be modified to take advantage of network home directories and Win2K's enhanced drive-mapping capabilities. You're now ready to install applications on your terminal server, test them for multi-user compatibility, and write an application compatibility script if necessary.

In Chapter 4, I'll discuss how to configure your user accounts for Terminal Services access. I'll explain the difference between a Terminal Services home directory and a Windows home directory as well as how Terminal Services maintains its user profiles. I'll show you how to support your users by remote controlling their sessions, and how to troubleshoot session connectivity and performance problems.

Chapter 4 will also address how to manage your terminal servers with Group Policy, and how to assign a separate policy to a user when he or she logs onto a terminal server instead of a workstation. I'll also introduce you to a number of command-line tools that you can use to manage and monitor your servers.

Chapter 4: Terminal Services Administration

Congratulations! You now have the skills you need to set up a terminal server and install applications. Next, you need to learn how to enable or restrict access to the servers and configure your users' accounts for the optimal Terminal Services experience. This tuning process involves a number of settings—user rights on the terminal server, permissions on the access protocols, Terminal Services profile configuration, and so on.

You also need to know how to support your users in a Terminal Services environment through registry access and shadowing. This chapter will go over these topics and introduce you to Group Policy loopback, permissions available for RDP, and remote control configuration.

Throughout this chapter, I will be speaking from the perspective of a native Win2K domain infrastructure. Although Win2K Terminal Services can function as a member server in an NT 4.0 domain, you gain the most benefit from working with the enhancements that true Win2K AD provides. This chapter will still be helpful to you if you're deploying Win2K Terminal Services in an NT 4.0 environment, but you should also consult texts written about WTS.

Terminal Server Access Requirements

For a user to log on to a terminal server, three primary settings must be in place. Fortunately, the required settings are default settings in Win2K. Unfortunately, these default settings mean that everyone in the domain can now access your terminal server, which may not be desirable. The three settings are

1. The Log on locally right—By default on a Win2K server, only members of the local Administrators group have the right to logon locally. This setting prevents unauthorized users from accessing the server from the console. When you enable Terminal Services in application server mode, the Users group is also granted this right. The one exception to this automatic addition of the Users group is if you enable Terminal Services on a domain controller; in this case, the Log on locally right remains restricted to administrators.
2. Permission to use RDP—An administrator can set permissions on RDP through the Terminal Services Configuration utility. By default, the local Users group is given User Access permissions to the protocol.
3. The Allow logon to Terminal Server check box—In the properties of each user object in AD, there is a check box that will allow the user to log on to a terminal server. This check box is selected by default.

If a user receives a *Does not have permission to access this session* error message, one of these three settings is preventing the user from accessing the session. In the following sections, I will explain how and where to adjust these settings.

Log On Locally

You can control the *Log on locally* right through the User Rights Assignment section of the machine policy. When you install Terminal Services in application server mode, this right is changed from Administrators to Users, and the local Users group includes all authenticated users in the domain. To view or change this right, use the Microsoft Management Console (MMC)

Local Security Policy snap-in found in Administrative Tools. Drill down to Local Policies, User Rights Assignment. Figure 4.1 shows the default groups granted this right.

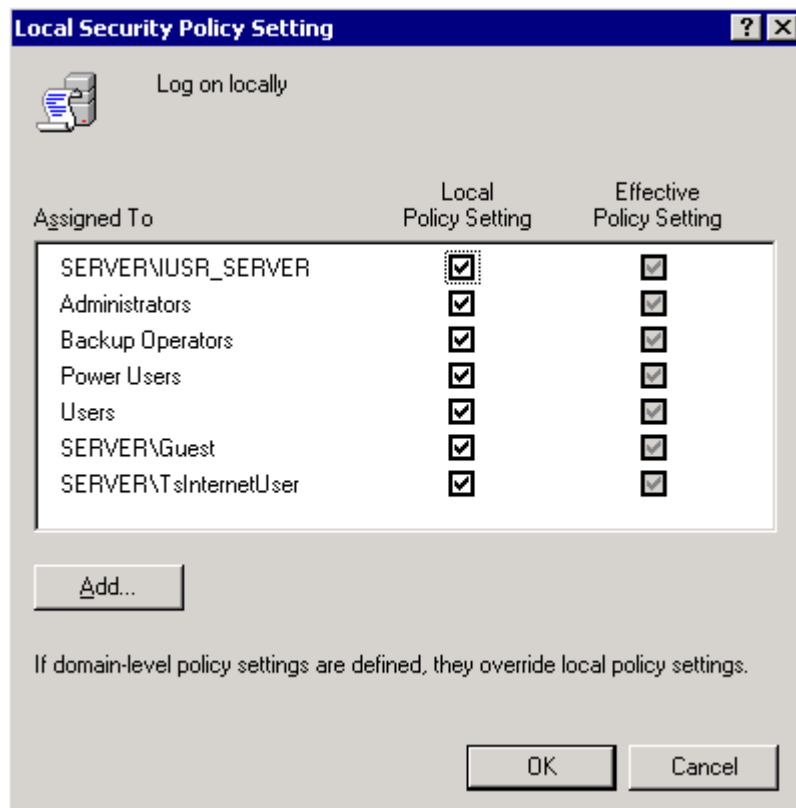




Figure 4.1: Groups assigned the *Log on locally* right.

 You can also access this setting by examining the local machine policy through the Group Policy Editor (GPE). From the Run text box, type `GPEDIT.MSC` and drill down to Computer Configuration, Windows Settings, Local Policies, User Rights Assignment.

All users using the terminal server will need this right, so you may want to leave this right set to the Users group and restrict access by setting permissions for RDP. Be aware that domain group policies can modify User Rights Assignment, so you need to be sure that a Group Policy Object (GPO) is not removing Users from this setting. To do so, check the Effective Policy Setting column, which Figure 4.1 shows. If the User group check box is clear in this column, you will need to examine your domain GPOs to find where the right is being restricted and adjust accordingly.

 If you are going to restrict the *Log on locally* right, be sure to grant it to the same groups that you are using to set permissions for RDP; otherwise you may have a difficult time determining which setting is preventing a user from logging on.

Permissions on RDP

In Chapter 2, I introduced you to the Terminal Services Configuration utility. At that point, we were only concerned with using it to tune the server for optimal performance, but you can also use this utility to set permissions on RDP as well as set overrides for user session settings. Figure 4.2 shows the connections window of the utility.

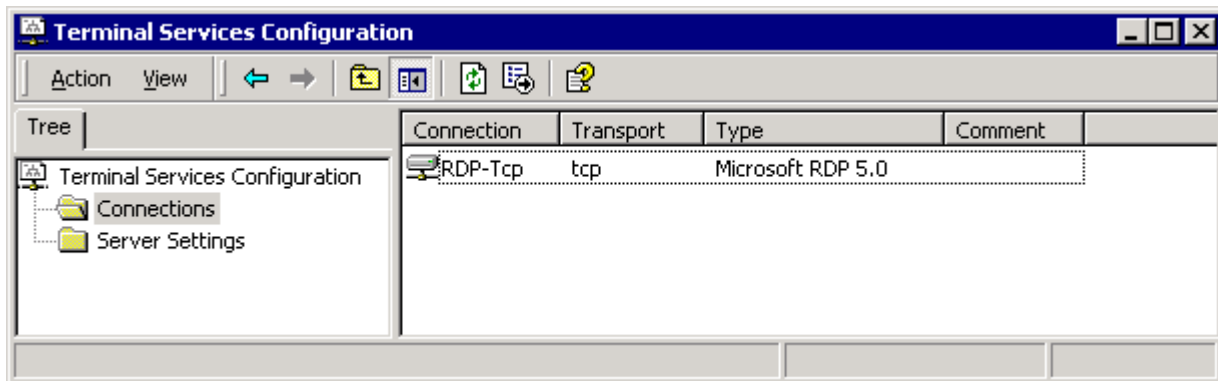


Figure 4.2: Connections settings in the Terminal Services Configuration utility.

For now, we are only interested in the permissions on the protocol itself. I will go into user settings and session overrides later. Double-click the RDP-Tcp connection, and select the Permissions tab, which Figure 4.3 illustrates.

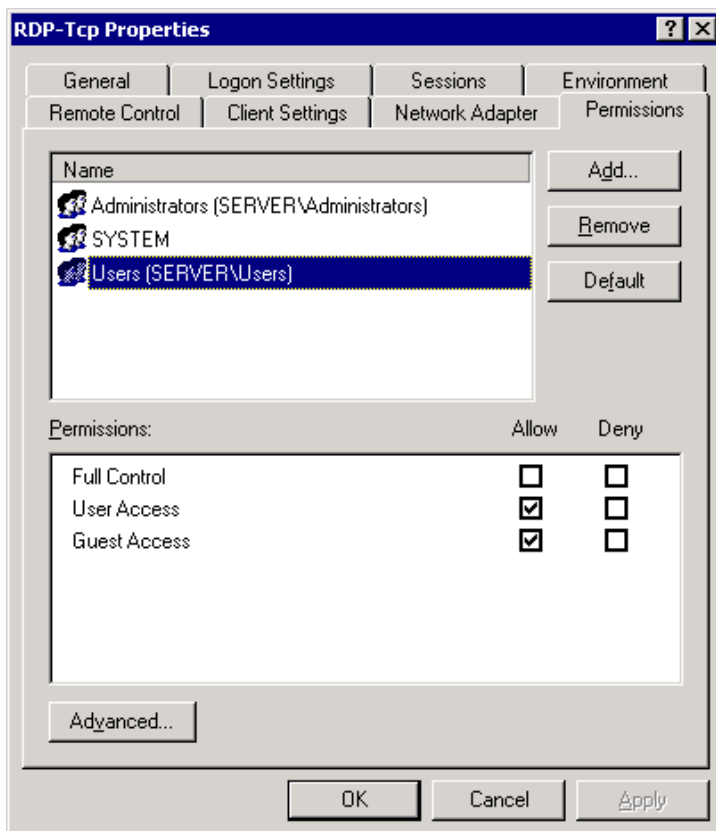


Figure 4.3: Setting permissions on RDP.

As you can see in Figure 4.3, the Users group is granted User Access by default. If you want to restrict access to the terminal server to a specific domain group, you would remove the Users group from this screen and replace it with another group that you have created to contain users that you want to be able to log on to the terminal server. To fully utilize the power of these permissions, you must first understand what each level of access provides, and what advanced settings are available.

RDP Access Levels

RDP provides three basic levels of access: Guest Access, User Access, and Full Control. The level assigned to a group determines the group's abilities when connected to the terminal server over RDP. Let's first examine the permissions available, then put them together into the basic access levels. Figure 4.4 shows the advanced ACL Editor's list of individual permissions, and Table 4.1 explains which abilities each permission setting bestows on the users.

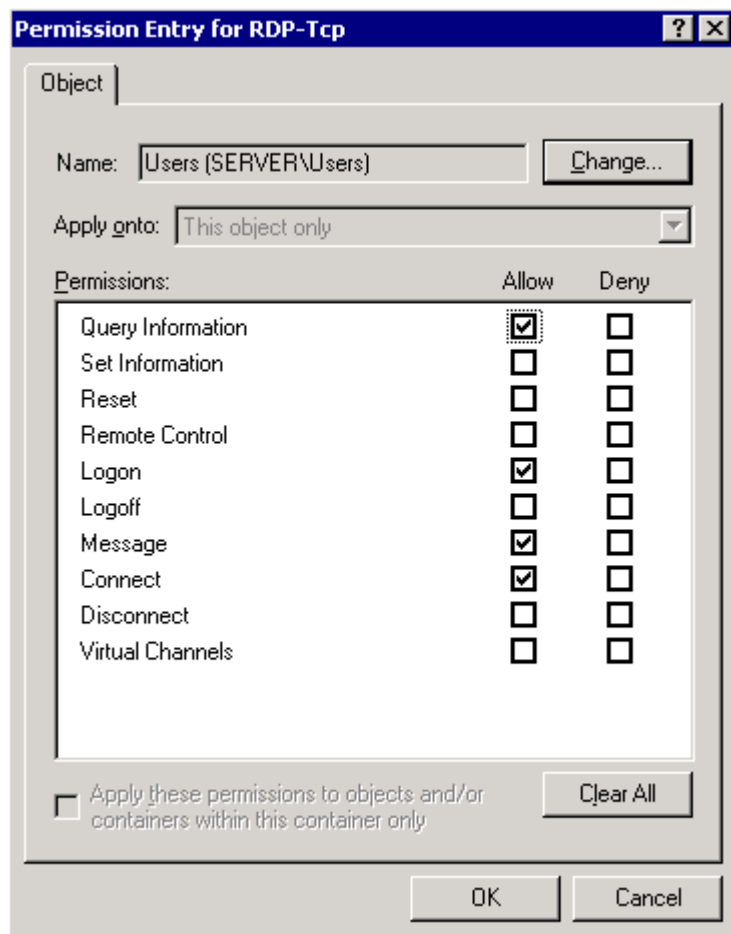


Figure 4.4: Permissions available for RDP.

| Permission | Ability |
|-------------------|---|
| Query Information | Query information through Terminal Services Administrator or at a command prompt using the QUERY command. |
| Set Information | Change settings and permissions for RDP. |

| | |
|------------------|--|
| Reset | Forcibly terminate another user's session. |
| Remote Control | View or actively control another user's session. |
| Logon | Log on to a session on the server. |
| Logoff | Force another user to logoff of his or her session. |
| Message | Send messages to other sessions on the server by using the Terminal Services Manager console or at a command prompt by using the MSG command. |
| Connect | Reconnect to a session that the same user left active on the server. |
| Disconnect | Forcibly disconnect another user from his or her session, leaving the session active on the server. |
| Virtual Channels | Use virtual channels, which are communication channels that developers can use to enhance the capabilities of RDP. As long as System (the local system account) has this permission, users will be able to use applications that take advantage of them. |

Table 4.1: Permissions available for RDP.

Table 4.2 shows the permissions assigned to each basic access level. You can also use the Advanced window of the ACL Editor to create special permission sets. For example, you may want your Help desk staff to have all the abilities of Full Control, except Set Information.

| Permission | Guest Access | User Access | Full Control |
|-------------------|--------------|-------------|--------------|
| Query Information | | X | X |
| Set Information | | | X |
| Reset | | | X |
| Remote Control | | | X |
| Logon | X | X | X |
| Logoff | | | X |
| Message | | X | X |
| Connect | | X | X |
| Disconnect | | | X |
| Virtual Channels | | | X |

Table 4.2: Basic access levels.

Allow Logon to Terminal Services

The last requirement to log on to a terminal server is a per-user setting. In the properties of each user object in the domain, there are a few tabs that are terminal server related. Most of the settings affect session behavior once a user is logged on. But if you leave the *Allow logon to terminal server* check box clear, the setting will prevent that user from logging on in the first place. Figure 4.5 shows the User properties tab that holds this setting. The *Allow logon to terminal server* check box is selected for all users by default.

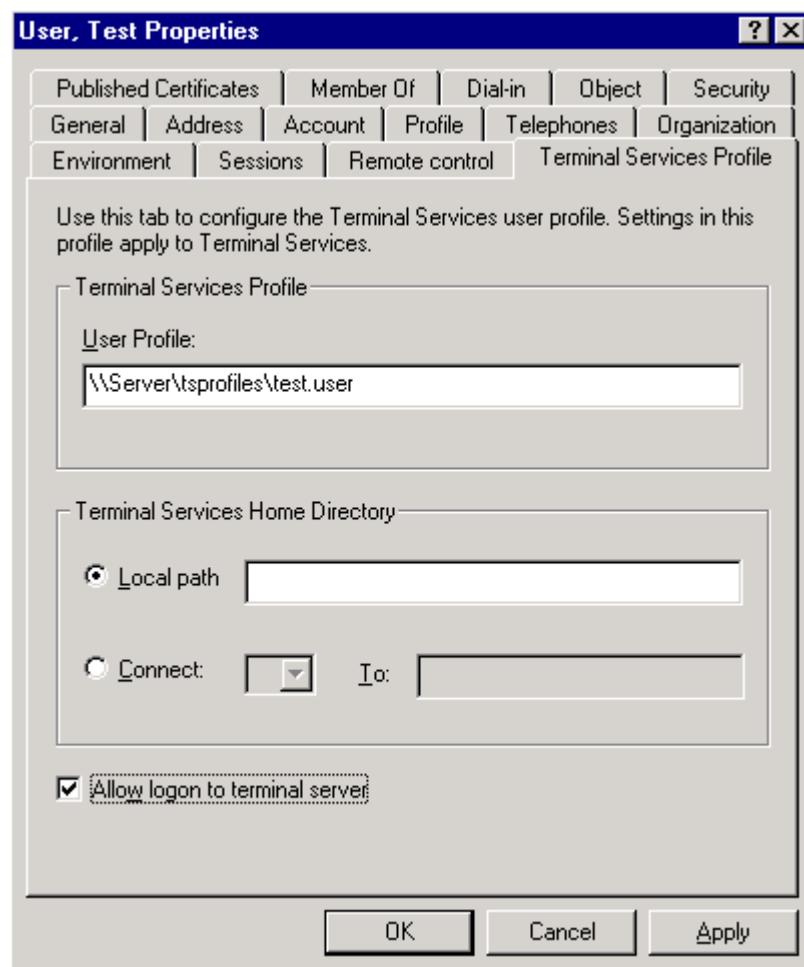



Figure 4.5: The Terminal Services Profile tab of a user object.

Controlling Session Behavior

Now that your users have permission to log on to the terminal server over RDP, you can control how their sessions behave. Almost all session-related settings can be set per-user or per-server. Per-server settings always override per-user settings, so you have the ability to make one terminal server behave differently from the rest.

 In W2K, each field in the user object properties (Full Name, Logon Script, Home Directory, and so on) is its own attribute in the AD schema. This setup gives administrators the ability to programmatically query and modify information and settings using ADSI scripting. Unfortunately, Microsoft did not include any of the Terminal Services settings in this model. All fields on the Terminal Services and Dial-in tabs are lumped together into one attribute—UserParameters.

Microsoft published a white paper about how to access these settings through the WTS32 API, but unless you are comfortable programming in C, you will find this task difficult. You can find the white paper at <http://www.microsoft.com/ntserver/zipdocs/TseApis.exe>.

There is a utility available on the terminal server, TSPROF.EXE, that will allow administrators to modify a user's Terminal Services profile path programmatically, but if you want to change other parameters, you will have to do so manually. Also, if you copy a user object to create a new user, the UserParameters attribute is not copied. So there is not even a way to create a template user who has the Terminal Services settings that you want to use in the domain. These shortcomings have caused most administrators to apply all session behavior settings from the server rather than on the user accounts. Keep these shortcomings in mind when you are planning your Terminal Services environment.

I will explain each of the settings and show you the interface that controls them from both the server properties and the user properties. You configure server settings by using the properties sheet of the RDP connection in the Terminal Services Configuration utility, and you configure user settings in the properties of each user object in the Active Directory Users and Computers MMC snap-in.

Session Timeouts

Figure 4.6 shows the Sessions tabs of the RDP and User properties windows. This tab controls timeouts and behaviors for Terminal Services sessions. By default, the User settings are configured for no timeouts and allow users to disconnect from the server while leaving the session active for them to reconnect to. The RDP properties are set to not override these settings, although in Figure 4.6, I have enabled user-setting overrides so that you can see the available timeouts.

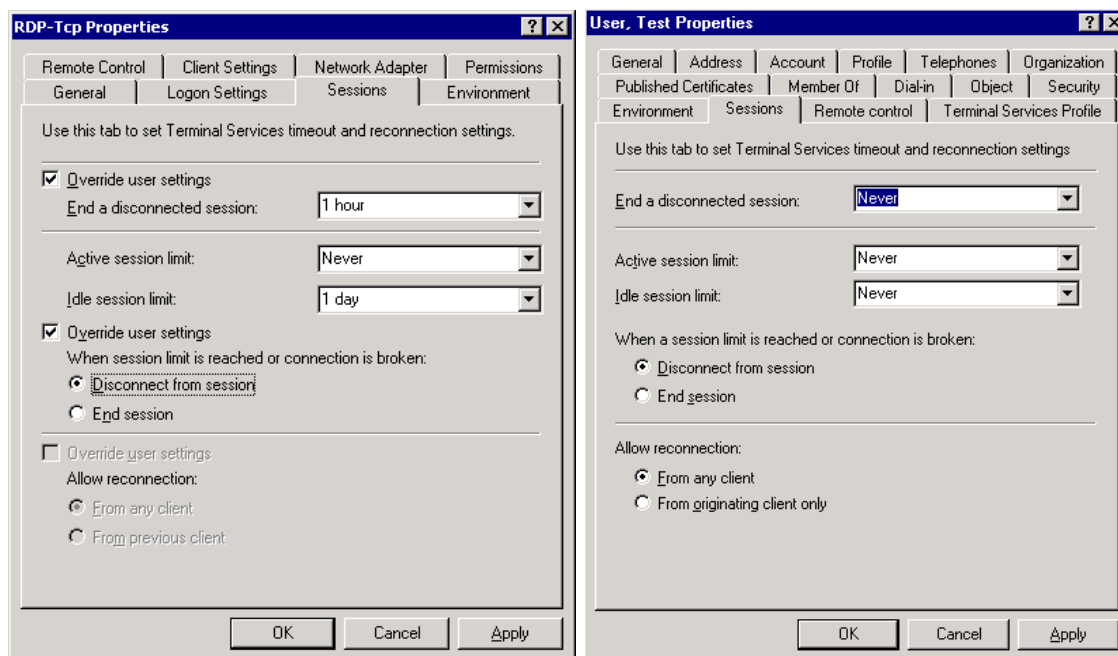


Figure 4.6: Session settings per machine (on the left) and per user (on the right).

Let's examine each of the options available:

- **End a disconnected session**—This setting defines how long a session will remain alive on the server after the user has disconnected from it without logging off.

- **Active session limit**—This setting limits the amount of time that a user can actively use a session. When the time limit approaches, users are given a warning message instructing them to save their work and log off.
- **Idle session limit**—When the server stops receiving keyboard and mouse input from the user but still maintains an open network connection with the client device, the session is said to be idle. This setting sets a time limit for idle sessions. When this limit approaches, the user is warned before the session is ended or disconnected.
- **Session limit behavior**—When the idle session limit is reached or the network connection is lost, this setting determines whether the session is immediately ended or placed into disconnected status.
- **Allow reconnection**—When a user disconnects from a session without logging off, either manually or by losing network connectivity, you can select whether to allow them to reconnect only from the same client device or from any client device.

Your particular environment will determine whether you use the user settings or the server settings or a combination. In my experience, a common practice is to leave the user settings alone and configure session timeouts from the server.

Environment Settings

The Environment tab of each interface determines the program to use as the shell of the session when the user connects. The default is to use EXPLORER.EXE to provide the user with a full desktop interface. Typically you will use the RDP client to configure a connection to a specific program. However, these interfaces allow you to override the client and either force all connections to a server or all connections from a specific user account to receive a unique program rather than a desktop or client-requested application. Figure 4.7 shows the default settings.

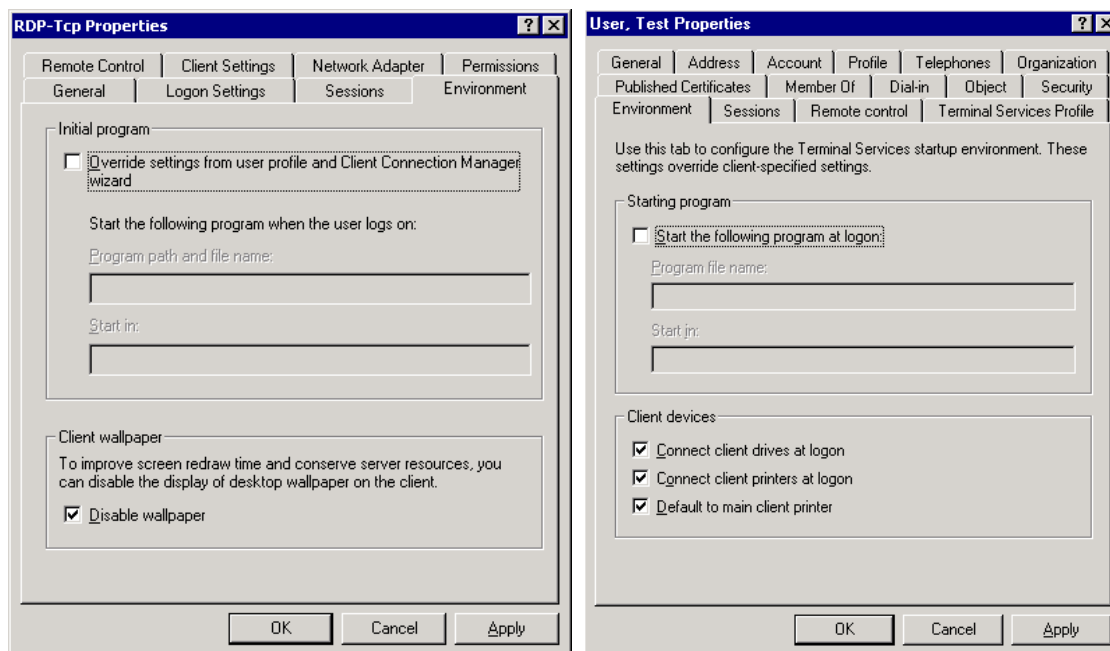


Figure 4.7: Default environment settings.

If applied per user, you can use this setting to configure a service account that can only access a specific program on the terminal server. Keep in mind, however, that if you configure this setting on the server side, even administrators will only be given the defined program and not a desktop.

The Environment tabs, in addition to the Client Settings tab of the RDP properties, which Figure 4.8 shows, also allow you to configure client mapping.

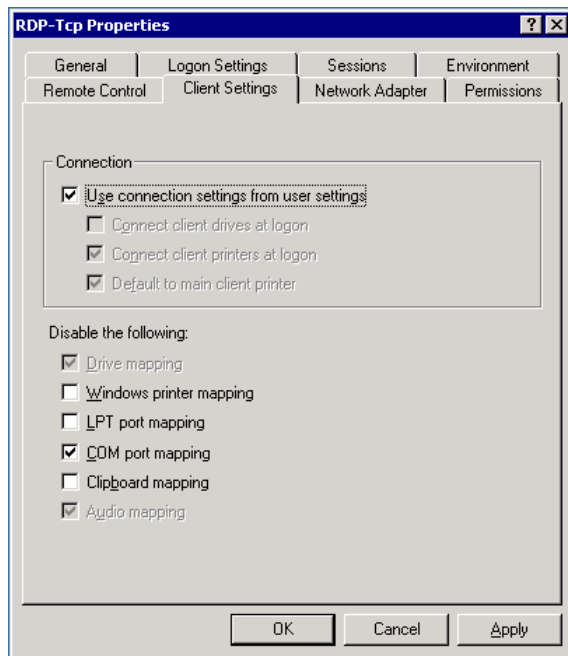


Figure 4.8: Client settings on the RDP properties sheet.

Any combination of settings can be used to enable or disable client functionality:

- **Connect client drives**—If you have installed the Drive Share resource kit utility, the client device's local drives are automatically mapped into the Terminal Services session at logon. You can clear this check box on the User object, or override this behavior from the server.
- **Connect client printers**—By default, parallel port printers connected to the client device will be added to the user's session (if the server has a driver for them). You can prevent this behavior by clearing this check box.
- **Default to main client printer**—If you are allowing printers to be connected, you can also configure the system to use the client device's default printer automatically.
- **Client wallpaper**—By default, the system will ignore any wallpaper defined in the user's profile to help improve performance. If you want to allow your users to have wallpaper, clear this check box. This setting is only available from the server.
- **Windows printer mapping**—This override will prevent any network printers that are mapped on the client device from being added to the session. Currently, this setting applies only to servers that have Citrix MetaFrame installed, but perhaps the next version of RDP will give us this ability as well.

- **LPT port mapping**—In addition to printers, Citrix’s ICA protocol can take advantage of other devices connected to the client’s LPT port. This setting will override this connection.
- **COM port mapping**—ICA can also access the client’s COM port. This override will prevent this behavior.
- **Audio mapping**—The current version of RDP does not support sound, unlike the ICA protocol. This override will disable sound from being sent to the client.

Remote Control

One of the most powerful support features of Terminal Services is remote control or *shadowing*. I will explain how to use shadowing later in the chapter. For now, let’s look at the settings that control how it behaves. Figure 4.9 shows the Remote Control tabs of each of the properties sheets.

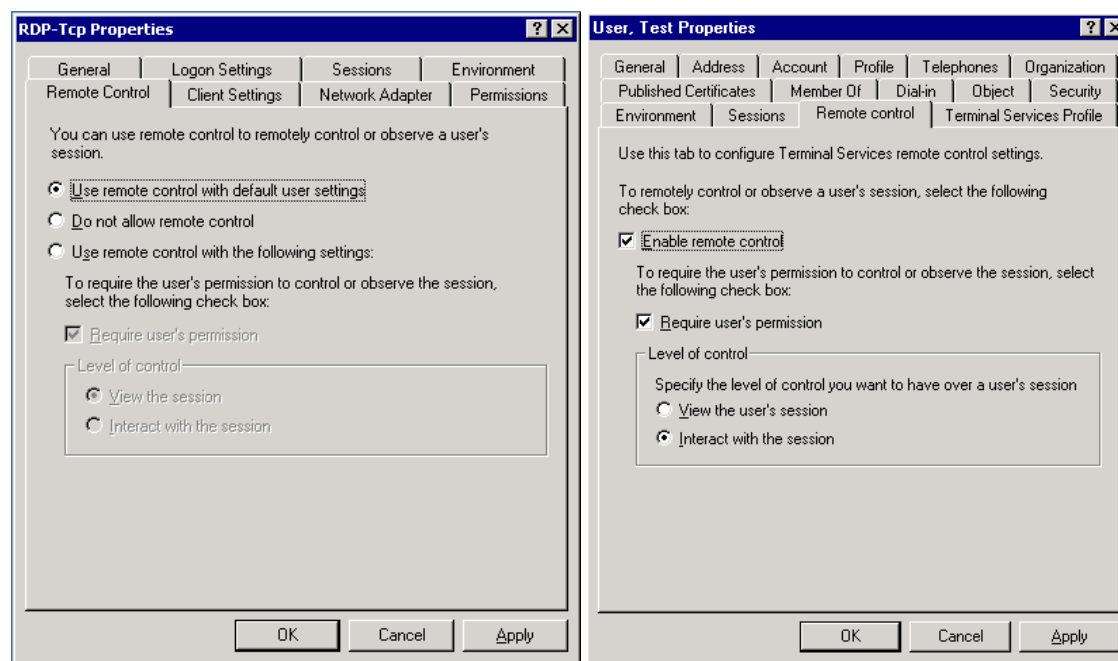


Figure 4.9: Remote control settings.

Like the other Terminal Services settings, the default remote control behavior is to obey the settings defined on the user object. The defaults are to allow a user’s session to be remote controlled and to require that the user be prompted to grant permission before an administrator or support person can view or interact with the session. Assuming that you do not disable remote control altogether, you can configure it with the following settings:

- **Require user’s permission**—If this check box is selected, when an administrator or support person wants to view or control a user’s session, the user is prompted for his or her permission. Figure 4.10 shows the request message box.

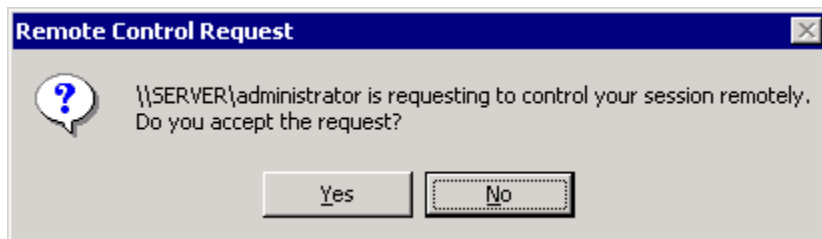


Figure 4.10: Remote Control request for a user's permission.

- **Control level**—You can configure the user account or the server to allow one of two levels of control when shadowing: View or Interact. When set to View, the support person can only observe the user, whereas, with Interact set, the support person can actually control the user's session as if the support person were sitting at the user's keyboard and mouse.

UI Settings

In addition to configuring permissions, timeouts, and remote control settings, you will also want to use system policies to lock down certain components of the UI. This topic is very volatile, as many users are accustomed to using features that systems administrators typically remove or disable in a Terminal Services environment, such as the command prompt, Run command, Task Manager, and so on. I would recommend using Microsoft article Q278295 "How to Lock Down a Windows 2000 Terminal Server Session" as a starting point. You can then add or remove restrictions based on your user's needs and behavior patterns.

Many of the policy settings are designed to eliminate confusion more than to secure the server. For example, you will typically remove the Shut Down command from the Start menu even though non-administrators do not have the rights required to shut down the server, so they would receive an error message if they accidentally select Shut Down.

Here are the policy settings that Microsoft recommends. I have separated Computer Configuration settings from the User Configuration settings.

Computer Configuration settings:

- Do not display last user name in logon screen
- Restrict CD-ROM access to locally logged-on user only
- Restrict floppy access to locally logged-on user only
- Disable Windows Installer - Always

User Configuration settings:

- Folder Redirection: Application Data
- Folder Redirection: Desktop
- Folder Redirection: My Documents
- Folder Redirection: Start Menu
- Remove Map Network Drive and Disconnect Network Drive

- Remove Search button from Windows Explorer
- Disable Windows Explorer's default context menu
- Hide the Manage item on the Windows Explorer context menu
- Hide these specified drives in My Computer (Enable this setting for A through D.)
- Prevent access to drives from My Computer (Enable this setting for A through D.)
- Hide Hardware Tab
- Prevent Task Run or End
- Disable New Task Creation
- Disable and remove links to Windows Update
- Remove common program groups from Start Menu
- Disable programs on Settings Menu
- Remove Network & Dial-up Connections from Start Menu
- Remove Search menu from Start Menu
- Remove Help menu from Start Menu
- Remove Run menu from Start Menu
- Add Logoff to Start Menu
- Disable and remove the Shut Down command
- Disable changes to Taskbar and Start Menu Settings
- Hide My Network Places icon on desktop
- Prohibit user from changing My Documents path
- Disable Control Panel
- Disable the command prompt (Set Disable scripts to No)
- Disable registry editing tools
- Disable Task Manager
- Disable Lock Computer

Obviously, if you use all the settings that are listed here, you will have a very restrictive and perhaps unusable environment. For example, Microsoft recommends removing the Map Network Drive command, but if you are in a distributed environment, your users may require this ability to access their documents. Nonetheless, this list provides is a good starting point.

I recommend starting with a very restrictive policy, then test what you can do as you re-enable certain features. Keep in mind that you should use Group Policy to eliminate confusion for your users and to help protect the system from undesired and inadvertent activity. Policies alone should not be relied upon to secure your server because there are many loopholes that a malicious user can exploit.


For example, if you use the *Prevent access to drives in My Computer* policy as the only method of protecting files on your server, a malicious user could very easily write a harmful batch file. Instead, you should implement restrictive NTFS permissions to protect the file system and use the Prevent Access policy as a way of preventing a user from mistaking the C drive of the server for the C drive of his or her client device. The good news is that if you are running the server in Win2K Permissions mode, most key areas of the file system and registry are protected by default.

You also need to be careful to create separate policies for your users and the systems administrators. Obviously, an administrator needs certain features that your users do not.

Group Policy and Software Installation

If you are familiar with Win2K Group Policy, you know that one of its very powerful features is the ability to install, upgrade, and manage software. Although you might be tempted to use this feature to push new software out to your terminal servers, you need to be very careful how you do so.

Most MSI packages are unaware of Terminal Service's Install mode, so if you attempt to use Group Policy to deploy user applications, you will not be able to take advantage of registry and INI file mapping. I have found that you are usually better off performing manual installations of all user software on a terminal server. Although this method sounds tedious, imagine how much time you are saving over installing that same software on 100 desktop computers, even by Group Policy!

 You can use the software installation component of the Computer Configuration settings to deploy system utilities to your terminal server—virus protection software, system monitoring agents, and so on. These types of software do not typically have user-based settings.

Managing Group Policy

After you have determined the proper set of policies for your users, you need a way to apply them. NT 4.0's system policy mechanisms are not nearly as flexible as Win2K's are. NT 4.0's system policy mechanisms require that terminal server administrators set up a separate NTCONFIG.POL for their terminal servers and manage it separately from the domain policy. The Win2K Group Policy process is very flexible and allows us to centrally manage all settings in the domain, and even maintain separate policies for users when they log on to a terminal server.

Win2K will still obey the NT 4.0 policy process. If you are using Win2K Terminal Services in an NT 4.0 domain, you should read Microsoft article Q192794 "How to Apply System Policies to Terminal Server" to learn how to configure a separate NTCONFIG.POL for you servers.

Group Policy is one of the most powerful and complex enhancements that Microsoft made in Win2K. I highly recommend that you become comfortable with how Group Policy works if you are going to be responsible for its design.

 For more information about Group Policy, I highly recommend Darren Mar-Elia's *The Definitive Guide to Group Policy* (Realtimedpublishers), available at <http://www.realtimedpublishers.com> and <http://www.fullarmor.com>.

If you are working in an environment in which your users exclusively use thin-client devices, and never log on to a Win2K Pro workstation, your Group Policy design can be very straightforward and will look identical to that of a non-terminal server design. If, however, your users log on to both workstations and thin clients, or you are using remote access or ASP model infrastructures, you need to learn how to apply separate policies for you users' desktop sessions and Terminal Services sessions. For this setup to work, loopback is the key.

Standard Group Policy Processing Order

Win2K Group Policy is a layered process. To determine the final settings that a user will see, you must look at all the GPOs that are being applied. GPOs are applied in a fixed order: local, site, domain, organizational unit (OU). Machine settings and user settings are processed separately, although they can both come from the same GPOs.

To understand how GPO processing works, let's look at a theoretical domain infrastructure and walk through GPO processing as it happens. Figure 4.11 shows our example domain and its GPOs. For simplicity, I have applied only one GPO at each level. In many implementations, you will have multiple GPOs at the site, domain, and OU levels. These GPOs would all have to be processed in order to produce a resultant set of policies.

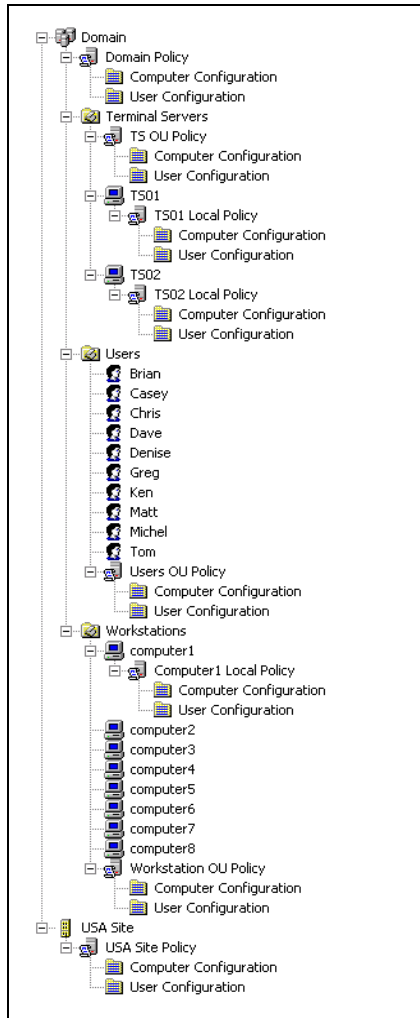


Figure 4.11: An example domain and its GPOs.

Let's start by looking at standard GPO processing by booting up computer1. During the boot process, security settings are applied for the computer itself in the following order:

1. **Local**—The Computer Configuration settings from computer1's Local Policy is applied.
2. **Site**—The Computer Configuration settings from the USA Site Policy is applied.
3. **Domain**—The Computer Configuration settings from the Domain Policy is applied.
4. **OU**—The Computer Configuration settings from the Workstations OU's Workstation OU Policy is applied. The OU policies are determined by the OU that contains the object in question. In this case, computer1 is in the Workstations OU, so policies linked to that OU will be processed.

Now computer1 has a complete configuration. Let's have Greg log on to computer1, and see how the system processes the User Configuration settings:

5. **Local**—The User Configuration settings from computer1's Local Policy is applied.
6. **Site**—The User Configuration settings from the USA Site Policy is applied.
7. **Domain**—The User Configuration settings from the Domain Policy is applied.

8. **OU**—The User Configuration settings from the Users OU’s Users OU Policy is applied. In standard processing mode, Greg will always receive his User Configuration from this policy regardless of which OU contains the computer he is logging on to.

Note that in standard processing mode, because there are no user objects in the Workstations OU, the User Configuration of the Workstation Policy is never applied. So, if you want Greg to receive a different set of policies when he logs on to TS01, you will need to use loopback.

Loopback Group Policy Processing Order

Loopback processing allows us to take advantage of User Configuration settings from GPOs linked to the OU that contains the computer that is being accessed. As Figure 4.12 shows, there are two modes of loopback processing: Replace and Merge. Merge mode instructs the system to first apply the User Configuration from the Users OU Policy (the standard processing order), then apply the User Configuration from the Computers OU Policy. Replace mode instructs the system to ignore GPOs from the Users OU altogether and only apply User Configuration settings from the Computers OU policy.

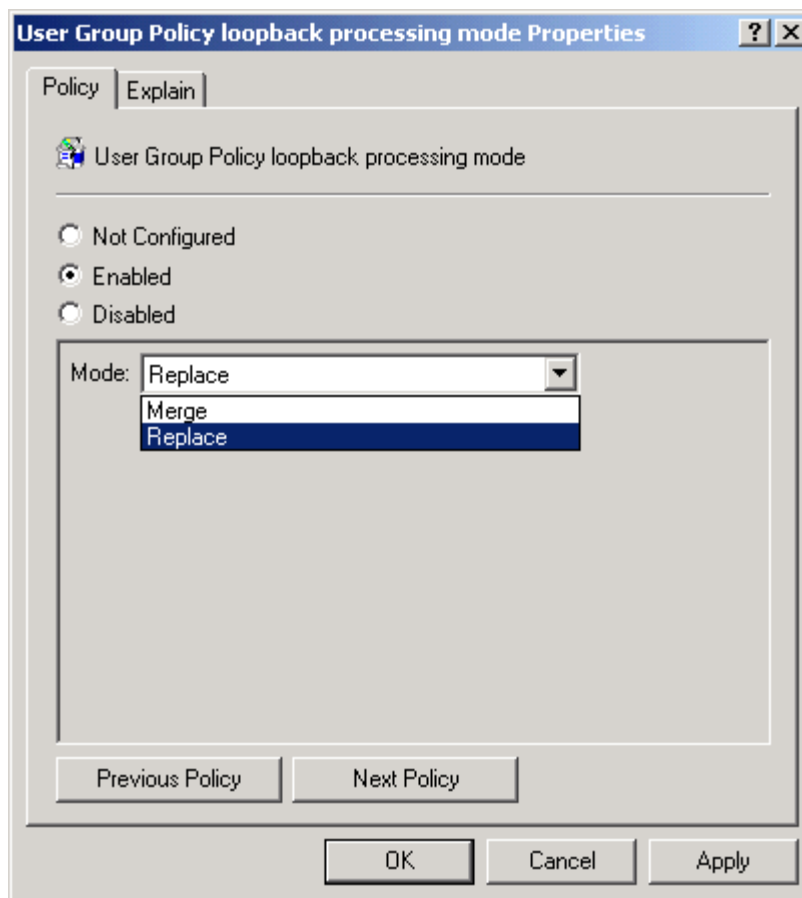


Figure 4.12: Loopback mode selection.

The Computer Configuration is applied as usual, but if we place TS01 into loopback merge mode, Greg’s User Configuration processing looks like this:

1. **Local**—The User Configuration from computer1’s Local Policy is applied.

2. **Site**—The User Configuration from the USA Site Policy is applied.
3. **Domain**—The User Configuration from the Domain Policy is applied.
4. **OU**—The User Configuration from the Users OU's Users OU Policy is applied.
5. **OU Loopback**—The User Configuration from the Terminal Servers OU policy is applied.

In this mode, Greg could receive his Internet Explorer (IE) Proxy Server settings from the Users OU Policy, but have the Shut Down command removed from his Start menu by the Terminal Servers OU Policy. Merge mode has the advantage of being able to place global settings in the Users OU Policy and only apply lockdowns in the Terminal Servers OU Policy. The disadvantage is that in this mode, you need to keep track of user settings in two GPOs.

In loopback replace mode, the User Configuration from the Users OU is ignored:

1. **Local**—The User Configuration from computer1's Local Policy is applied.
2. **Site**—The User Configuration from the USA Site Policy is applied.
3. **Domain**—The User Configuration from the Domain Policy is applied.
4. **OU Loopback**—The User Configuration from the Terminal Servers OU Policy is applied.

This mode simplifies GPO processing by placing all settings into the Terminal Servers OU Policy, but it will force you to keep some settings in this policy in sync with changes made to the Users OU Policy. For example, if you are using My Documents folder redirection to centrally store user files, you will want the setting to be the same when Greg logs on to either a workstation or a terminal server. If the domain administrator migrates Greg to a new file server, the setting will have to be changed on both policies.

Enabling Loopback

Loopback is enabled through the Computer Configuration section of the GPE shown in Figure 4.13. You can enable it either in the local machine policy on the terminal server, or through any of the GPOs being applied to your terminal servers.

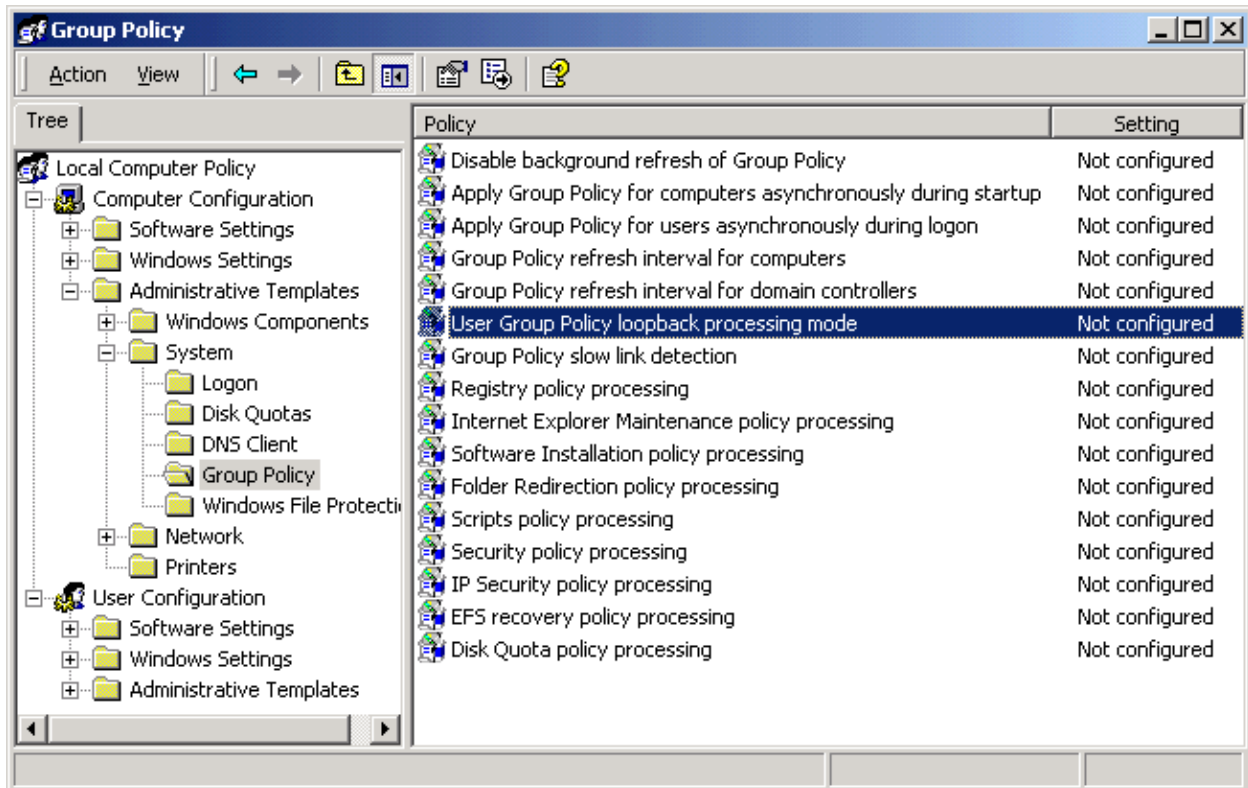


Figure 4.13: The GPE's loopback setting.

Home and Profile Directories

As a systems administrator, you are familiar with network home directories and roaming profiles. These features in Windows allow us to maintain central stores for our users' documents and profile settings so that they are available regardless of which computer users sit down at.

Terminal Services has the ability to maintain its own separate stores for home and profile data for the users. As with everything else in this chapter, how you utilize this ability depends on your environment and administration style.

Terminal Services Profile Path

When a user logs on to a workstation, the system checks the profile path attribute of his or her user object to see whether the user has a centrally stored profile. If he or she does, and it is newer than any locally cached copy that may exist, the profile is downloaded for the user. In the same way, when a user logs on to a terminal server, the system queries the UserParameters attribute and looks for a Terminal Services Profile path. Figure 4.14 shows the Terminal Services Profile tab of a user object.

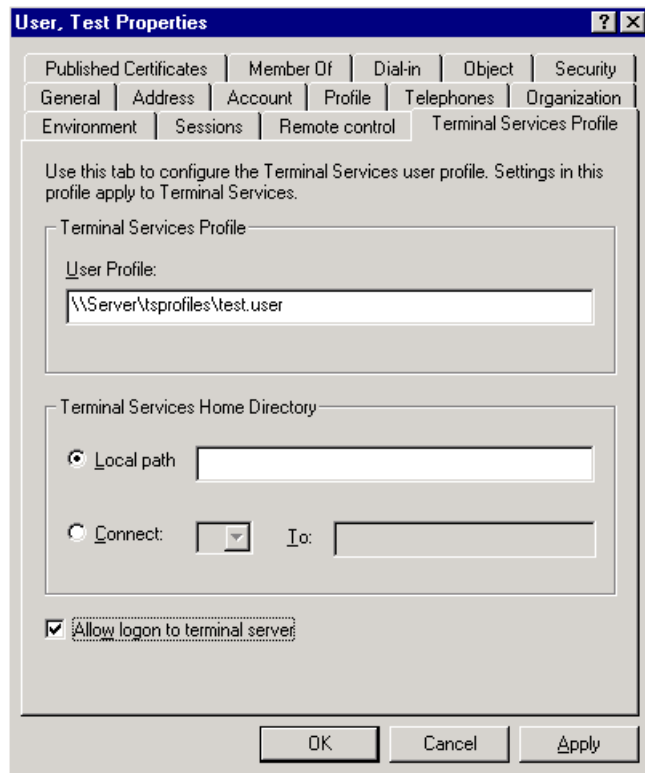



Figure 4.14: The Terminal Services Profile tab.

This separation enables administrators to maintain separate profiles for users depending on which type of computer they are accessing. In most cases, you will want to take advantage of Terminal Services profiles, as certain functions of Terminal Services make it difficult to not maintain them. I'll explain what I mean.

If you do not use roaming profiles for your users' workstations, you rely on the fact that the computer maintains a copy of the user profile. If your users do not log on to more than one PC, this setup is perfectly fine. On a terminal server, however, not using roaming profiles means that the terminal server would be maintaining the profiles for *all* your users. Also, if you need to scale your Terminal Services infrastructure and use load balancing to distribute your users among more than one terminal server, your users would be maintaining separate profiles on each server.


To alleviate the disk space problem, we can enable a system policy that deletes cached copies of roaming profiles. This way, after a user logs off of the terminal server, the disk space is reclaimed once the system copies the profile back to the central location.

 If you do not define a Terminal Services profile path but define a Windows roaming profile path, the terminal server will use the Windows profile. Also, if the Terminal Services profile is defined but unavailable, the system will fall back on the Windows profile. This behavior may have undesired results if you are using applications with ACSs on your server. See the appendix for a script that will detect whether the profile that is loaded is a Terminal Services profile and will log the user out before any ACSs run.

If you use roaming Windows profiles, using Terminal Services profiles can be even more critical, because if the system does not find a Terminal Services profile path in the user's

account, it will then look for a Windows profile path and use it instead. As you learned in Chapter 3, ACSs often make changes to registry settings under the HKEY_CURRENT_USER subkey to help tune applications for simultaneous users. If these changes are made to the user's Windows profile, the user might experience problems the next time he or she logs on to a workstation.

Geography may also be a factor in separating your profiles. You probably store your users' Windows profiles on file servers that are close to their workstations, but your terminal servers, given their low-bandwidth requirements, may be in a central data center. You would not want to have to pull a profile across a slow WAN link.

 When you enable Terminal Services, a utility is added to the system, TSPROF.EXE. This utility allows you to query, copy, and update Terminal Services profiles on your user objects without having to go into the Active Directory Users and Computers snap-in.


Terminal Services Home Directories

You are also able to configure your user accounts to use separate home directories when logging on to a terminal server. As you learned in Chapter 3, the system uses the user's home directory as its ROOTDRIVE and stores application compatibility files there. The intention of using a separate home directory when logging on to a terminal server was to keep these files out of the user's Windows home directory.

The problem is that if your users store their documents in their Windows home directory, they will need that same directory available when logging on to a terminal server. If you define a Terminal Services home path, it will be mapped instead of the Windows home path during logon. If you do not define a Terminal Services home path, the Windows home is mapped as usual, but ACD files will be written to it. This behavior poses quite the challenge.

If you are in a smaller environment, you can take advantage of My Documents folder redirection to centrally store your users' documents, and reserve the home directories for application data only. But folder redirection is based on security groups, so it does add administrative overhead in larger infrastructures.

Most users will not mind the few directories that ACSs create and will simply ignore them. If you want, you can modify your ACSs to flag these directories as hidden to keep them from bothering your users.

 As with profiles, if you do not define a Terminal Services home path, the system will use the Windows home path instead. So if you want to use the same home directory for both workstations and terminal servers, simply leave the Terminal Services home path blank.

Managing and Supporting User Sessions

Support is one area in which Terminal Services outshines a workstation environment. You have already seen how centralized application installation and the low-maintenance of thin clients greatly reduce TCO. In this section, I will show you how having users function within a Terminal Services environment also simplifies support.

If you are used to a workstation-centric infrastructure, you know how difficult the task is of remotely diagnosing and correcting problems for your users. You probably have used such tools as SMS remote control and Timbuktu to work on a user's computer. And you know how to connect to a network registry to modify settings for your users.

In a Terminal Services environment, these same tasks are made easier and more straightforward. Rather than tracking down a specific workstation on a remote node of your network, you and your users are both logged on to the same computer. And, since you are both utilizing RDP to transmit KVM data, you can easily tap into the stream to provide assistance. To show you the various support techniques, I will first introduce you to the utilities used.

Terminal Services Manager

When you launch the Terminal Services Manager administrative tool, you are presented with a list of all servers with Terminal Services enabled in the domain. Using this tool, you can easily see which servers users are connected to, which client devices they're accessing the servers from, and which processes and applications they are running in their sessions. Figure 4.15 shows you the Terminal Services Manager interface.

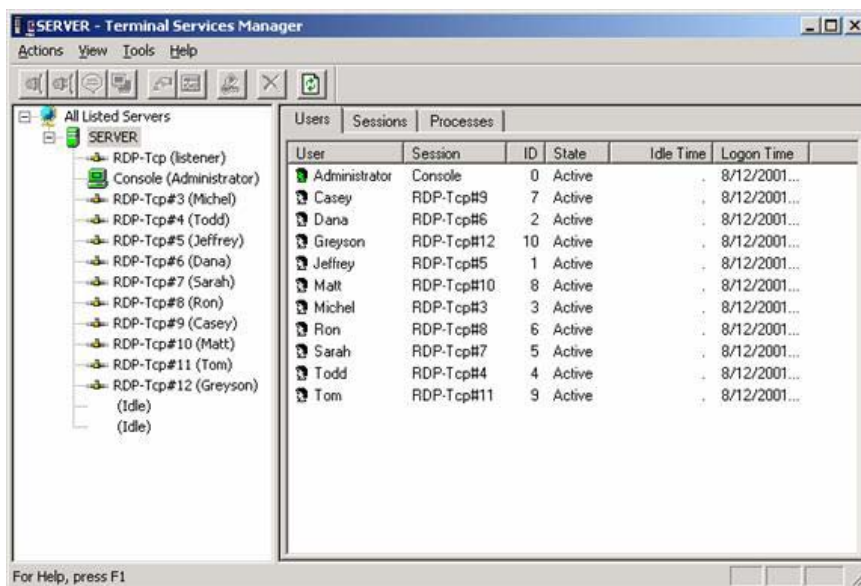


Figure 4.15: The Terminal Services Manager interface.

Once you highlight a specific server, the tool shows you a list of all user sessions on that terminal server. It also shows you each session's status:

- **Active**—The user is actively sending keyboard or mouse information to the server.
- **Idle**—The user has not moved the mouse or entered a keystroke in a set period of time.
- **Disconnected**—The user has disconnected from the server, but has left the session running for later reconnection.

If you highlight a user's session in the left pane, the results pane will show you all the processes that highlighted user has running on the server, as Figure 4.16 shows. Here you can end a hung process for a user.

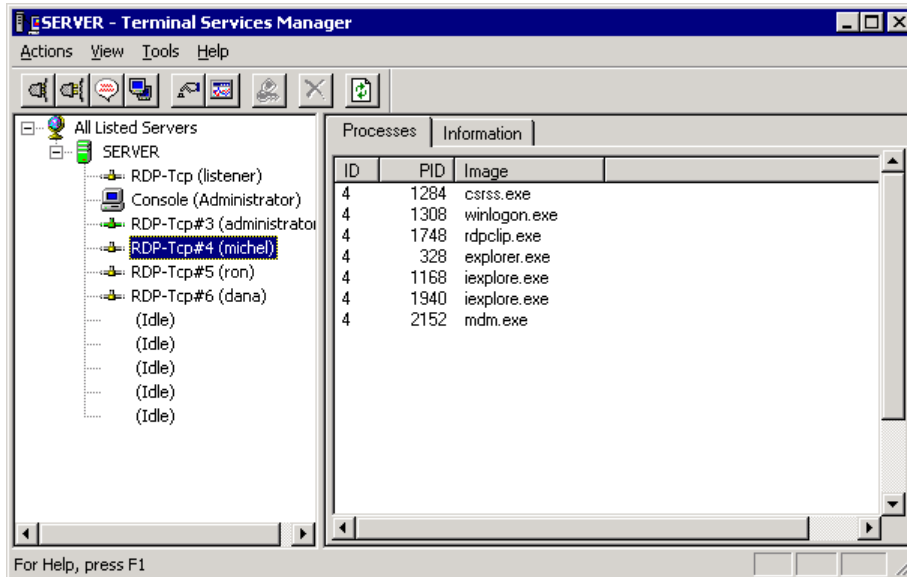


Figure 4.16: Active processes in a user's session.

The information tab will present you with the user's client device name and IP address as well as the version number of the RDP client that he or she is running, the screen resolution, and the encryption level. This information can assist you in troubleshooting connectivity problems. If you right-click a user's session, you are presented with the seven options that Figure 4.17 illustrates.

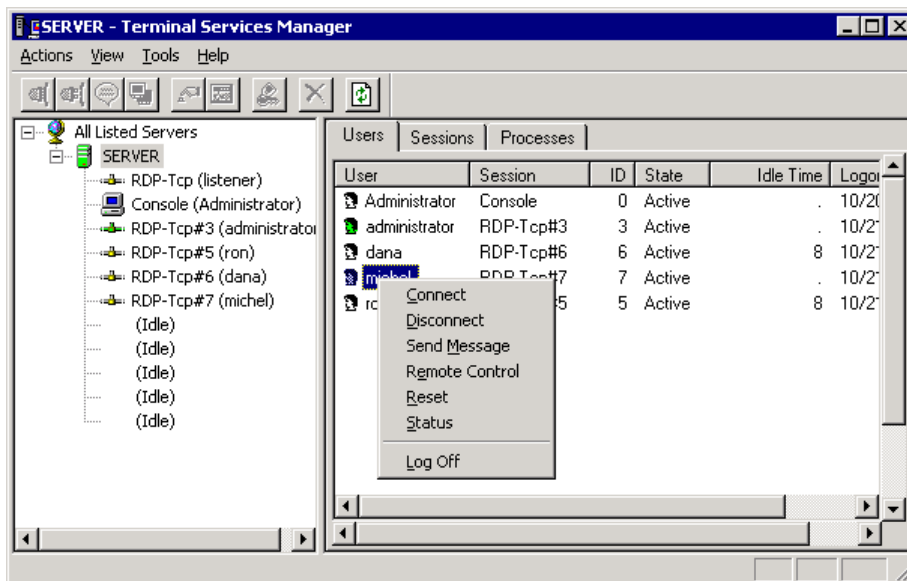



Figure 4.17: Options available to administrators from the Terminal Services Manager.

These commands allow you to interact with the user's session in several ways:

- **Connect**—Takes over a session from the user. The user's client device is disconnected, and you gain exclusive control of the session.
- **Disconnect**—Disconnects the user from the session, but leaves it running on the server.

- **Send Message**—Sends a popup message to the user.
- **Remote Control**—Allows you to view or interact with the user's session without disconnecting the user. The user sees any activity that you perform while remote controlling, and you can observe him or her as well.
- **Reset**—Kills the session.
- **Status**—Shows a status window of network activity between the server and client device.
- **Log Off**—Forces the user to log off of the session.

 The Log Off command will perform a graceful logout and upload the user's profile to the central profile directory. However, it will not give the user any opportunity to save any work in progress.

Remote Control

When you select Remote Control from the Terminal Services Manager interface, you temporarily disconnect from your session and are connected to the user's session. RDP now sends all video information to both you and the user's client device and receives keyboard and mouse movements from both of you, if you have configured remote control with interact privileges.

While remote controlling, the user can observe you while you launch applications, change settings, and so on, and you can observe the user to see an error that he or she is reporting. One thing to remember is that any restrictions that you have placed on the user through Group Policy are in effect while remote controlling, so if you have disabled registry editing for you users, you will not be able to launch REGEDIT either.

Registry Editing

There are some support issues that require you to edit a user's registry. In a workstation environment, this feature requires you to connect to the remote registry on the user's workstation. On a terminal server, you are sharing the same registry with your users. There is only one HKEY_LOCAL_MACHINE subkey for all users, and each session's HKEY_CURRENT_USER subkey can be seen under HKEY_USERS.

Each user's registry is listed by SID. The quickest way to find a specific user, assuming you don't know the user's SID, is to look in the Volatile Environment subkey for each user. This subkey contains the APPDATA variable, which will list the username in its path, as Figure 4.18 shows.

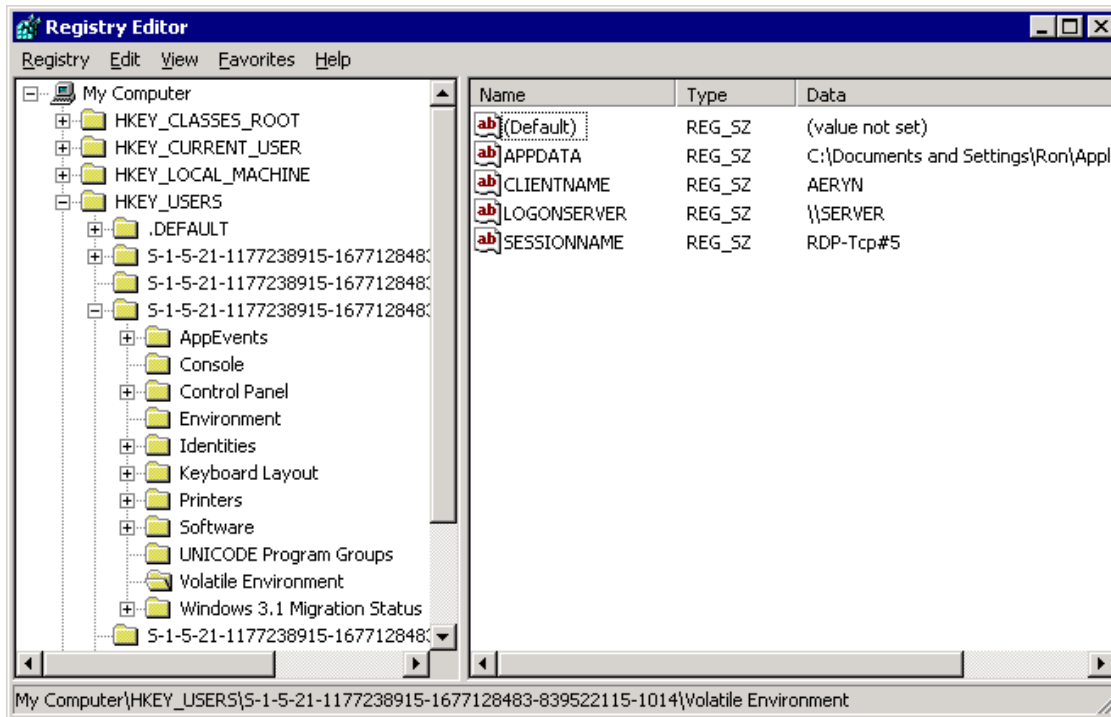


Figure 4.18: The HKEY_USERS subkey showing Ron's HKEY_CURRENT_USER registry subkey.

The user immediately sees any changes you make here.

Command-Line Utilities

There are a number of command-line utilities that you can use to help manage your servers. Most have direct counterparts in Terminal Services Manager, but the command-line utilities can be useful if you are writing administrative scripts.

- **CHANGE USER {/Install /Execute}**—The CHANGE USER command is used to switch the server between install and execute mode for application installation.
- **CHANGE LOGON {/Enable /Disable}**—The CHANGE LOGON command can disable any new logons from being accepted by the server.
- **Query {Process | Session | Termserver | User}**—The Query command presents similar information as the Terminal Services Manager tool presents: lists active processes by session, active sessions, available terminal servers, and current users.
- **TSSHUTDN**—The TSSHUTDN command is used to shut down or reboot a terminal server. This command, unlike the Shut Down command from the Start menu, gives your users a warning that the server is being shut down so that they can save their work. It then forces a logoff for each session, and finally shuts down the server.

Managing Printing

Printing is one of the most difficult things to manage in a Terminal Services environment. Win2K's increased driver set has made the task a little easier, but there are still a number of challenges.

First, by default, only administrators have the right to install printer drivers on a terminal server. So unless Windows has a native driver for a printer that your users will use, an administrator will have to manually install the driver on each terminal server in advance. You also need to be careful when installing drivers, as many new printers (especially multifunction devices and personal printers) will attempt to install software components in addition to driver files. These applications can sometimes be incompatible with a multiuser environment. Whenever possible, select a driver-only install.

Also, if your users have personal printers that they want to use, you have the additional challenge of trying to keep up with the ever-increasing number of printers available on the market. In addition, Terminal Services supports only LPT-connected and network printers, so any printer that uses a USB connection will not be recognized or supported.

Now for the good news—if you have a number of terminal servers in your environment, you can centralize printer-driver management by implementing a *trusted source* for drivers. This method will allow you to install new drivers on one terminal server, then automatically make them available to all the servers.

Start by selecting one server to be the trusted source. On this server, leave the *Prevent users from installing printer drivers* setting enabled (the default setting). This setting is found in the policy editor—either Group Policy or local policy—under Computer Configuration, Windows Settings, Security Settings, Security Options.

Next, share out that server's C:\WINNT\system32\spool\drivers\w32x86 directory. You can name the share whatever you like. This share point will be your trusted source.

Now, on all other terminal servers, perform the following actions:

1. Start a registry editor and locate the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Print\Providers\Lan Man Print Services\Servers subkey.
2. Select Add Value from the Edit menu, then add the LoadTrustedDrivers entry of type REG_DWORD with a value of 1. A value of 1 indicates that drivers can be installed only from the trusted print server shares specified by the TrustedDriverPath value.
3. Select Add Value from the Edit menu, then add the TrustedDriverPath entry of type REG_SZ = \\servername\sharename where servername is the name of the server you selected to be the trusted source, and sharename is the name of the appropriate share.
4. Use either local or Group Policy to disable the Prevent users from installing printer drivers security option.
5. Close the registry editor and restart the terminal server.

Now when you want to install a new printer driver, you need only install it on the trusted source server. In addition, your users have the right to then install these drivers on the other servers themselves, which is done with little or no action on their part.

- ☐ This technique greatly simplifies driver management, but it does not address USB printers or multifunction devices that your users may have attached to their workstations. If you have an environment in which this behavior is a problem—particularly difficult if your terminal servers are used for remote access, take a look at ScrewDrivers by triCerat Software at <http://www.tricerat.com>. This utility creates a redirector service that allows your users to print to their local print drivers, thus eliminating the need to install drivers on the server at all.

Summary

Every Terminal Services implementation requires its own unique set of permissions, rights, and policies. In this chapter, I have presented you with the information you need to begin your Terminal Services infrastructure design. You will need to seriously consider how your terminal servers integrate with the rest of your Win2K domain. With the flexibility of Group Policy and the powerful support methods that RDP and Terminal Services provide, you will certainly find the right combination to fit your needs.

In Chapter 5, I will go into depth about capacity planning, scalability, and performance benchmarking and troubleshooting. I will show you how to configure load balancing and implement preventative maintenance to keep your terminal servers available when your users need them.

Chapter 5: Performance, Capacity Planning, and Availability

With all the advantages that the server-based computing model provides, there are also a number of challenges. By centralizing your users' computing environment, you create a much greater need for fault tolerance, performance, and availability. In the traditional computing model, if a workstation goes down, it typically will affect only a single user, but in a terminal services environment, if a system becomes unavailable, it will impact a large number of clients.


As you saw in Chapter 4, careful use of permissions, user rights, and group policies will reduce the possibility that a user will be able to negatively impact the server or other users. But in addition to protecting the system, you need to implement a strategy to address capacity needs, fault tolerance, and system maintenance.

In this chapter, I will go into depth about the capacity planning tools that I introduced you to in Chapter 2. I will also introduce you to the Microsoft load-balancing service, which you can use to implement fault tolerance at a per-server level. Once you have the tools needed to design a fault-tolerant architecture that will handle your users' needs, we will dive into techniques for monitoring system health and maintaining top performance through preventative maintenance.

Capacity Planning

As server and memory prices continue to drop, capacity planning is not the daunting task that it once was. In Chapter 2, we went through the NEC and Groupe Bull study and established some baseline memory requirements for your terminal servers. In most cases, generous application of these guidelines will prove sufficient, and when you implement load balancing as part of your infrastructure, you can easily add a server to the cluster if your server hardware reaches maximum capacity.

I will introduce you to the tools that Microsoft, NEC, and Groupe Bull used in their study, but I should strongly caution you that these tools were developed exclusively for the study and were not originally intended for public release. Therefore, they are poorly documented and difficult to use. In most cases, I would recommend implementing a test server and using live QA testers to establish the load of an unusual application.

 To obtain the capacity-planning tools, download them from <ftp://ftp.microsoft.com/reskit/win2000/roboclient.zip>. The tools are also included in the Win2K resource kit, but the versions in the kit simply do not work. If you have the resource kit and want to patch the capacity-planning tools, you can download a hotfix at http://download.microsoft.com/download/win2000platform/Tscpt/1.0/NT5/EN-US/tscpt_hotfix.exe.

The Testing Environment

To use the capacity-planning tools, you will need to set up a testing environment similar to the one that Figure 5.1 illustrates. The lab environment will need to include the following computers:

- A terminal server with applications you want to test as well as the QueryIdle (QUIDLE.EXE) program.
- A test manager system to run RoboServer.
- A domain controller to authenticate your test's virtual user accounts.

- A number of test clients. These clients must be workstations and not thin client devices. Depending on the workstation hardware, you can run a number of test connections from each machine. Each client machine will need SimulatedClient and RoboClient installed in addition to the Terminal Services client program.

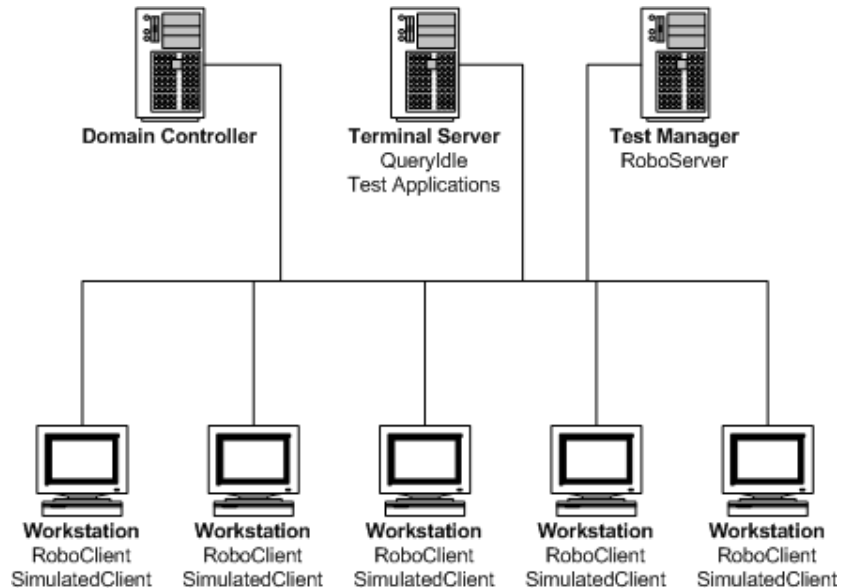


Figure 5.1: Capacity planning lab computers.

The Tools

As Figure 5.1 shows, there are four components to the capacity-planning toolkit. These components include RoboClient, RoboServer, QueryIdle, and SimulatedClient.

RoboClient (ROBOCLI.EXE)

The RoboClient program is installed on each test workstation. It communicates with the test manager system and controls execution of your test scripts. By default, when you launch ROBOCLI.EXE, it looks for a test manager system named `ts-dev`. If your test manager system has a different name, you can use an `-s` switch to specify another server name:

```
robocli.exe [-s servername]
```

(where *servername* is the name of the test manager system), or enter the name in the GUI, as Figure 5.2 shows.

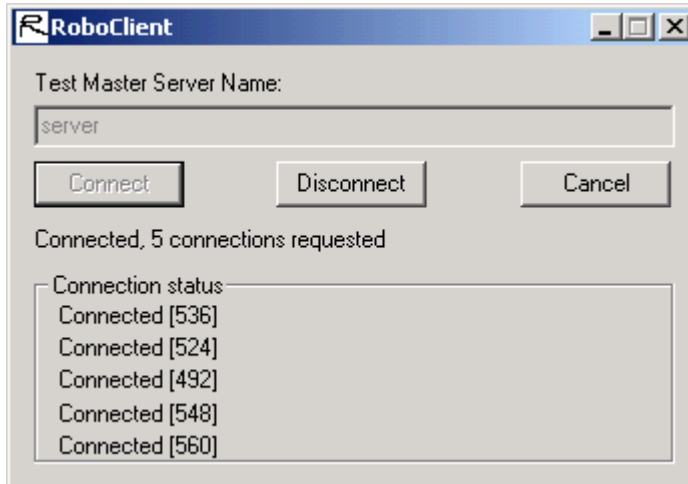


Figure 5.2: The RoboClient GUI.

RoboServer (ROBOSRV.EXE)

RoboServer is installed on the test manager system. This tool is the conductor of all activity during the test. RoboServer instructs each instance of RoboClient when to open a new connection to the terminal server and which test script to run in the session. From the GUI, you can specify the following parameters:

- The name of the terminal server to use
- The number of sessions each client computer should establish
- How many sessions make up a test set
- The delay time between test sets
- The delay between multiselect commands (if you instruct a number of sessions to begin a test script, RoboServer will wait this number of seconds between execution)

You can also specify the terminal server name and the number of sessions per client from the command line using an `-s` and `-n` switch, respectively:

```
robosrv.exe [-s TS_Name] [-n sessions_per_client]
```

Figure 5.3 shows the RoboServer GUI.

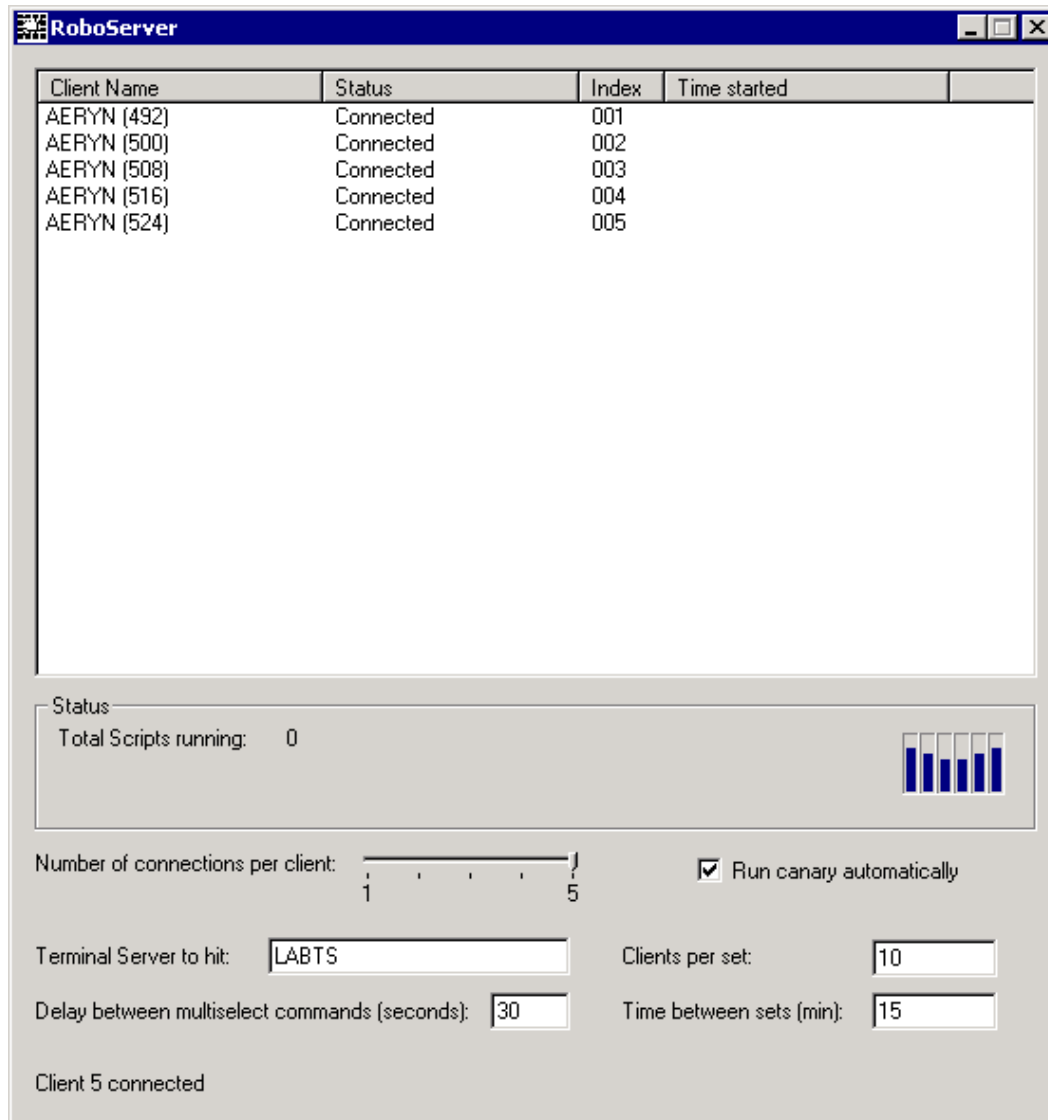


Figure 5.3: The RoboServer GUI.

QueryIdle (QIDLE.EXE)

The QueryIdle program is typically run from the console of the terminal server. This program monitors sessions and updates its display every 30 seconds. If a session becomes idle for 2 minutes, QueryIdle beeps and displays the username and computer name of the idle session so that you can reset the session or troubleshoot your test script. You launch QueryIdle from the command line; include a server name as an argument if you want to monitor a server other than the one on which you launch QueryIdle, as the following example shows:

```
QIDLE.EXE TS_Name
```

SimulatedClient (SMCLIENT.EXE)

The SimulatedClient program runs on each client system in the test. This program waits for RoboClient to kick it off, then opens a session on the terminal server and begins running a

specified script. During the test, SimulatedClient is completely controlled by RoboClient and RoboServer. However, if you are writing your own test scripts, you can use the following command to manually launch SimulatedClient to test your script before beginning an automated test:


```
smclient.exe -f file_name [-s TS_Name]
```

Test Scripts

Once each instance of RoboClient connects to RoboServer, you can right-click the connection and instruct that instance to execute a test script. Doing so passes the correct command-line parameters to SMCLIENT on that system and begins the test.

 You can download template testing scripts from <http://www.microsoft.com/windows2000/zipdocs/TSScale.exe>.

To write your own test scripts, you will need to learn the language that SMCLIENT uses—a fairly simple language that requires every action to be specified, right down to key ups and key downs. I will go through the command syntax of the scripting language in the appendix.

 The script names are hard-coded into RoboServer and match the template scripts that you downloaded. If you are writing your own scripts, you will need to use the same filenames that Microsoft used.

The Test Domain

You will need to set up user accounts on the test domain controller for your sessions to use. Once again, the tools expect a default set of usernames, so you should build accounts named smc001, smc002, smc003, and so on. For simplicity, you might want to use blank passwords for all accounts.

As you can see, the capacity-planning tools are fairly high-maintenance to use. Most terminal server administrators will find that setting up this type of test is far more work than it is worth. Instead they use the published figures as a guideline and carefully monitor the performance of the systems in the real world making capacity projections from there and implementing additional systems when necessary.

Performance Monitoring and Troubleshooting

As with any server implementation, monitoring the health and performance of your systems is critical. In most cases, your existing monitoring systems should be able to address the needs of your terminal server hardware, but there are some specific metrics that you might want to watch within the systems, especially if you are evaluating the capacity of the systems to determine when to implement additional processors, RAM, or parallel servers.

Hardware Monitors

There are many excellent server-monitoring systems. Depending on your server hardware and existing infrastructure, you should be able to incorporate your terminal servers into the existing systems.

Compaq Insight Manager, Hewlett-Packard OpenView, and NetIQ AppManager are all common system-monitoring systems and can all handle the unique requirements of terminal servers.

Rather than go into the specifics of hardware monitoring, as they are no different than with any high-availability Wintel servers, I will point out how terminal server monitoring is unique.

Network Traffic

If you have a large number of clients connecting to a terminal server, you may see elevated network traffic. You might want to implement either a teamed-NIC solution or a high-speed LAN connection (100Mb or gigabit Ethernet) to improve network performance.

Scheduled Reboots

As I will explain later, many terminal server implementations use regularly scheduled reboots to help maintain system performance. Be sure that your system monitor takes this behavior into account by either having “ignore periods” or adequate delays before issuing an alert.

Hardware Fault Tolerance

Most servers today have fault-tolerant components—multiple power supplies, dual NICs, and RAID disks. If any of these components fail during normal operation, the backup systems will take over and allow the server to continue to run. Typically failure will trigger an alert with the monitoring system. In a terminal server environment, backup systems are critical but pose an unusual challenge—many systems will stop the boot sequence if one of the redundant systems is down and wait for an administrator to OK the boot on backup systems. This behavior can be a problem with scheduled reboots. If you can, change the default setting in the server’s BIOS to continue the boot process without intervention.

System Monitors

In Win2K, Microsoft provides a number of utilities for monitoring and troubleshooting system performance. You are probably already familiar with most of these tools, but when you enable Terminal Services, a number of new options become available. You should always run performance monitors during your testing and staging phases to establish baselines for comparison once you migrate your Terminal Services-based applications to production systems.

In addition to the native Win2K tools, a number of third-party systems are available to help monitor the health of your systems and alert you in the event of a problem. NetIQ and Quest Systems both make excellent products.

Performance Monitor (PerfMon)

PerfMon is the tried and true native monitoring tool for Windows systems. As Figure 5.4 shows, you can use PerfMon to view, log, and create alerts based on performance counters. In addition to the counters that you may be familiar with—Free Memory, % Processor Time, and so on—Terminal Services also has a number of counters specific to it. I will define some of the key

counters to look at, both standard and Terminal Services-specific, then go over how to set up logging and alerts.

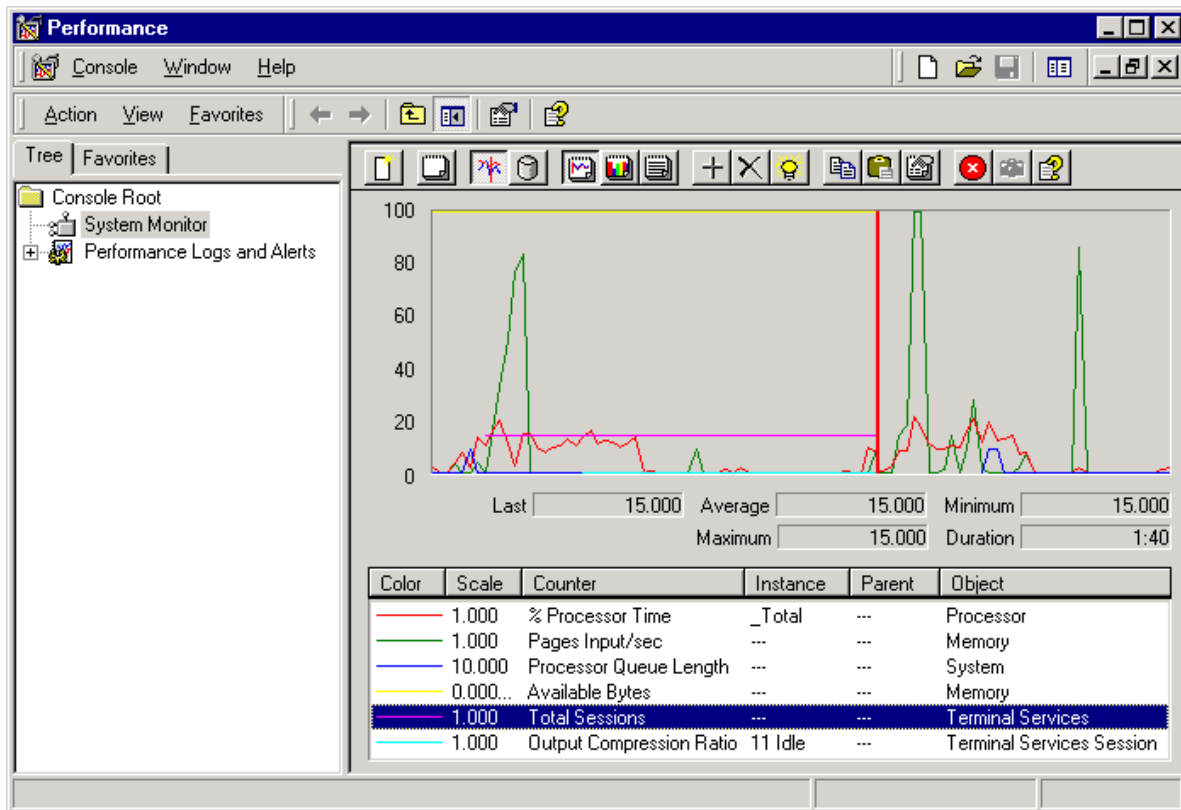


Figure 5.4: Example Performance Monitor counters.

To add counters to the PerfMon window, right-click the graph or click the plus (+) icon on the toolbar. This will bring up the Add Counters dialog box that Figure 5.5 shows. Here you specify the counter you want to add to the graph and the server you want to monitor. Clicking Explain is very helpful, as it adds a lower pane to the dialog box that gives a definition of the counter that you have highlighted.

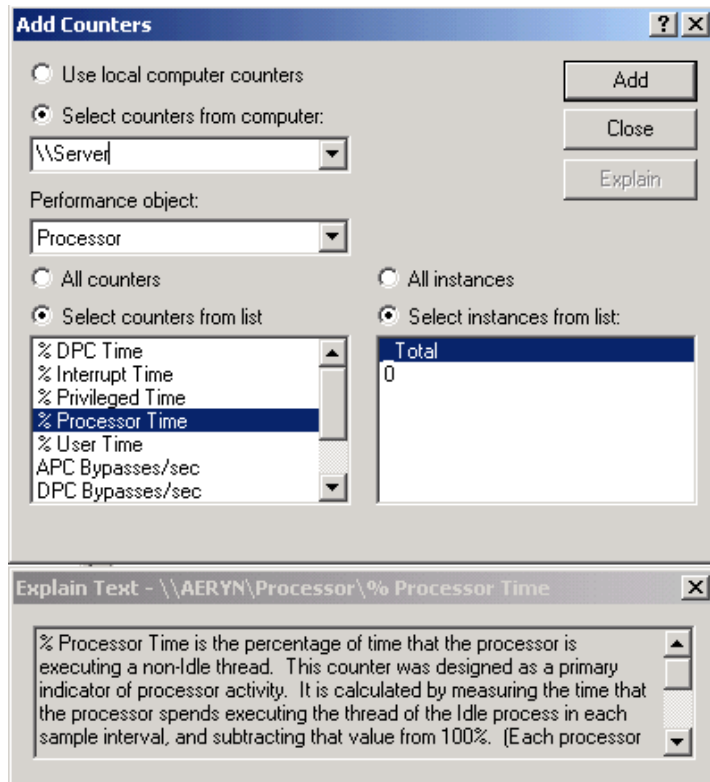


Figure 5.5: The Add Counters dialog box.

Here are some of the key counters to look at:

- **% Processor Time**—Displays the percentage of time that a processor is busy with a non-idle thread. In a multi-processor system, you can monitor an individual processor or the total. This counter is found in the Processor Object.
- **Processor Cue Length**—This counter is found in the System Performance Object and displays the number of threads waiting for an available processor. If you see a cue of greater than 2 over an extended period of time, you may have a congested system. In this case, you need to add processors or parallel servers to lighten the load.
- **Available Memory (in bytes, kilobytes, or megabytes)**—These three counters under the Memory Object display available physical memory on the system. Monitoring this level will help you determine whether additional RAM is needed.
- **Page Inputs/Sec**—This counter is key to measuring system performance. It measures the number of times per second that the system must go to the page file instead of physical memory. Also found under the Memory Object, this counter should always be looked at in conjunction with the available memory on the system.
- **Active Sessions**—When monitoring a terminal server, two new objects are available. Under the Terminal Server Object, you can monitor Active, Idle, and Total sessions. By using these counters as a baseline when looking at the other performance counters, you can establish a healthy total number of users to allow on each server.

- Terminal Services Session Object—This object isn't an individual counter but an object with a number of counters that can help you monitor performance of the session manager and RDP. This object includes counters for compression ratio, RDP input and output, and the efficiency of the client-side bitmap cache.

Logging and Alerts

Through PerfMon, you can set up log files to track system performance over time and alerts to inform you if the system becomes over taxed. To create a log file, in the left pane of PerfMon, expand the Performance Logs and Alerts node, then right-click Counter Logs. In the resulting dialog box, you create a log in the same way that you added counters to the live graph. Figure 5.6 shows the logging interface.

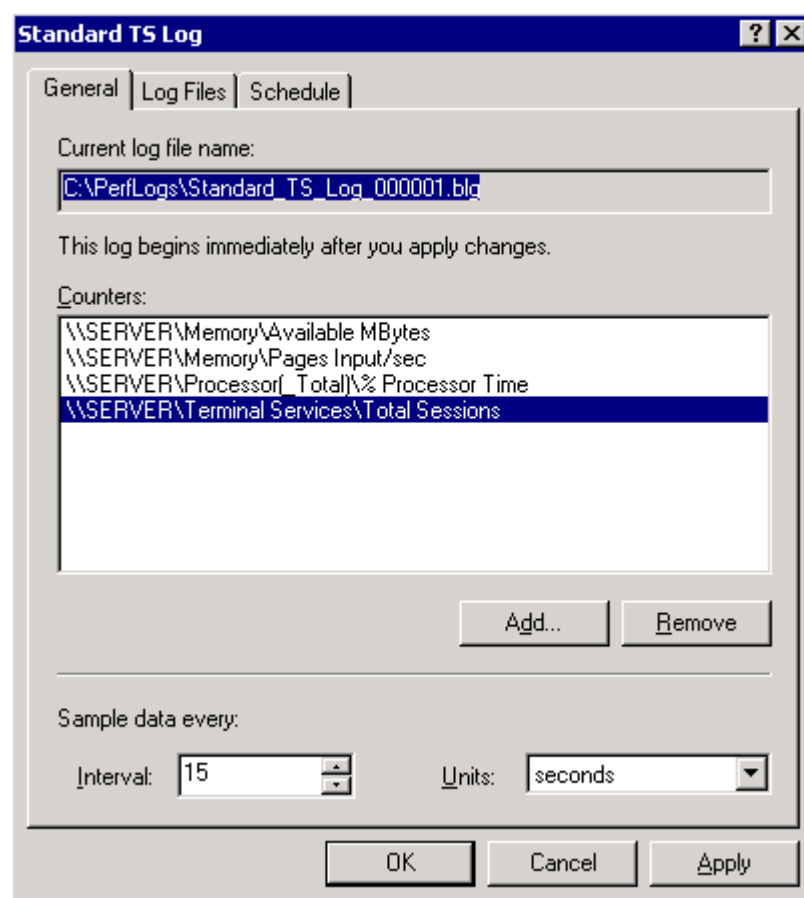


Figure 5.6: Adding a log to Performance Monitor.

The other two tabs in this window allow you to change the default filename and location for the log and control start and stop times for the log. On the Log Files tab, you can also select the format for the log file.

Also in the left-pane of the main PerfMon window, you can add administrative alerts. These alerts are used to trigger specific actions when a defined value for a counter is reached. Figure 5.7 shows the Add Alert dialog box.

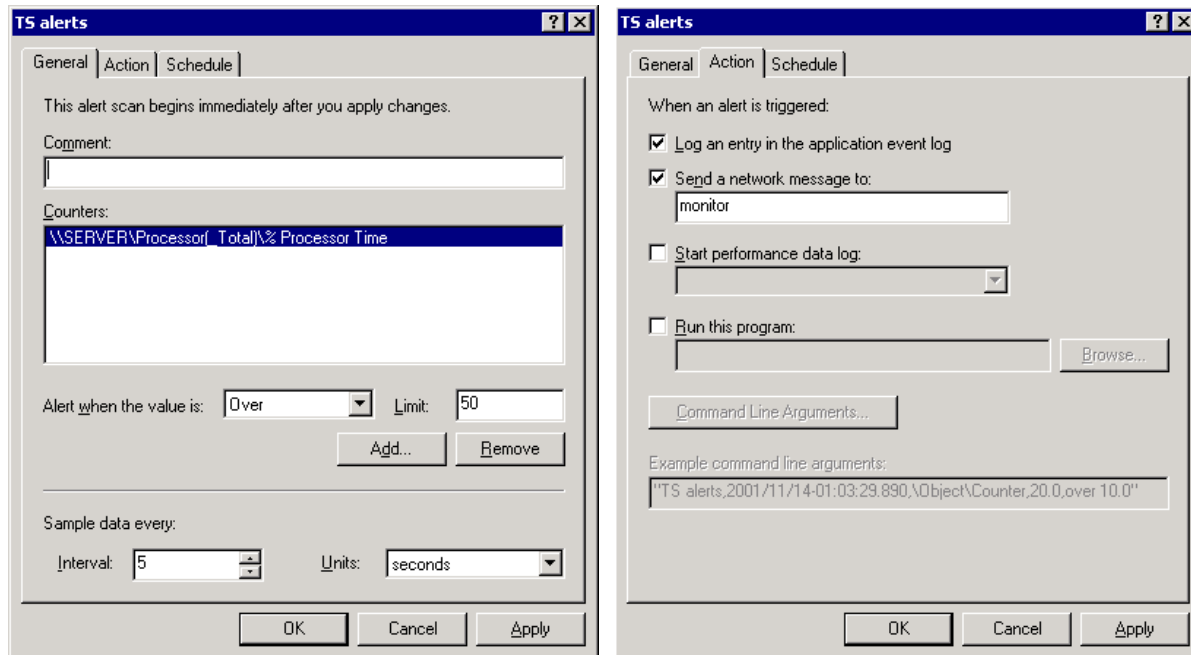


Figure 5.7: Setting up an alert in Performance Monitor.

In Figure 5.7, an alert is triggered when % Processor Time reaches 50 percent; an entry is added to the application event log, and a network message is generated to the computer named Monitor. You can also use events to start a specific PerfMon log or trigger a specific program to run on the server.

Task Manager

Task Manager is one of the most common tools used in Windows and is often considered a user tool; it offers a great deal of information in a very condensed format. On a terminal server, you can use Task Manager to display processes in either only your session or all sessions on the server. From here you can kill a hung process or quickly spot a leaky one.

In Figure 5.8, you can see the *Show processes from all sessions* check box. Once this check box is selected, you should select *Select Columns* from the *View* menu, and enable the *Username* and/or *SessionName* column. This information will easily identify which users are running a specific process.

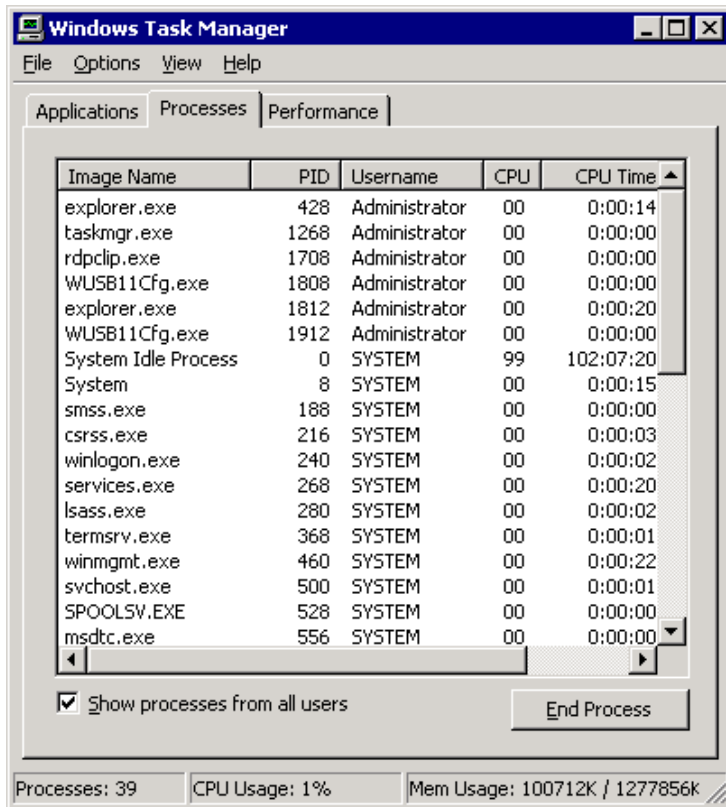



Figure 5.8: The Task Manager GUI.

Load Balancing

The Windows Load Balancing Service is the most powerful feature in a terminal server environment. With it, you can build server farms that can handle far greater user load than an individual server. The Windows Load Balancing Service also creates a layer of fault tolerance in your design—if one server goes down, users can connect to other servers in the cluster. Granted, any users logged onto the crashed system will be impacted, but they will be able to log back onto the terminal server without having to go looking for another hostname.

At a very basic level, the Windows Load Balancing Service creates a Virtual IP address (VIP) and hostname for a group of servers. When a request or connection is made to the VIP or hostname, the load-balancing service decides to which physical server to route the request based on the number of connections on each server.

 Don't confuse the Windows Load Balancing Service with Windows Cluster Service. Both services are only available on Advanced Server and Datacenter Server. The difference between the two services is that while the Windows Load Balancing Service is like a traffic cop directing sessions to one server in the group, Windows Cluster Service would create parallel connections to multiple servers and run all requests and applications on all servers, therefore creating a truly fault-tolerant design. Unfortunately, only Windows Load Balancing Service can be used with Terminal Services. Although you will often see the term "cluster" applied to either type of server group, I will use the term farm to describe a group of load-balanced servers.

Installing Load Balancing

To install Windows Load Balancing Service, go to the Network Control Panel applet and enable Network Load Balancing in the properties of the network connection definition. If Network Load Balancing is not listed, you can add it by clicking Install, then selecting Service. Figure 5.9 shows Network Load Balancing enabled.

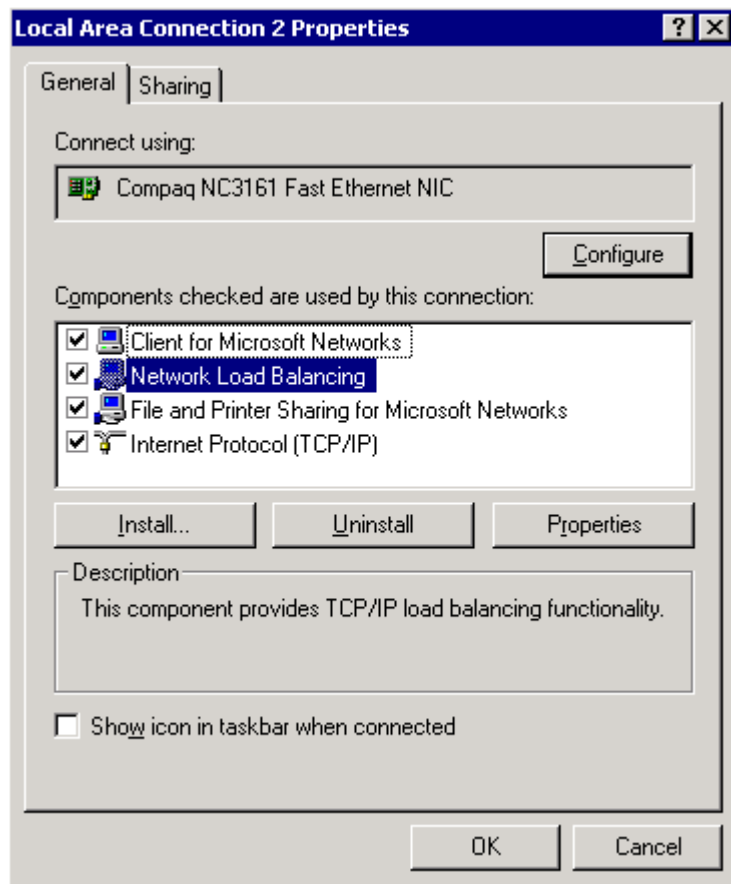


Figure 5.9: Enabling Network Load Balancing.

Before you can configure Windows Load Balancing Service, you will need to add an additional IP address to the TCP/IP protocol. To do so, highlight the Internet Protocol (TCP/IP) option, and click Properties. In the resulting Advanced TCP/IP Settings window, which Figure 5.10 shows, you will see the IP address for the server. The first address in the list should always be the actual physical IP address of the server. Click Add, and enter the VIP address you want to use for the farm. You will need to repeat this addition on all servers in the farm. The VIP you are adding will be the same for all servers in the group.

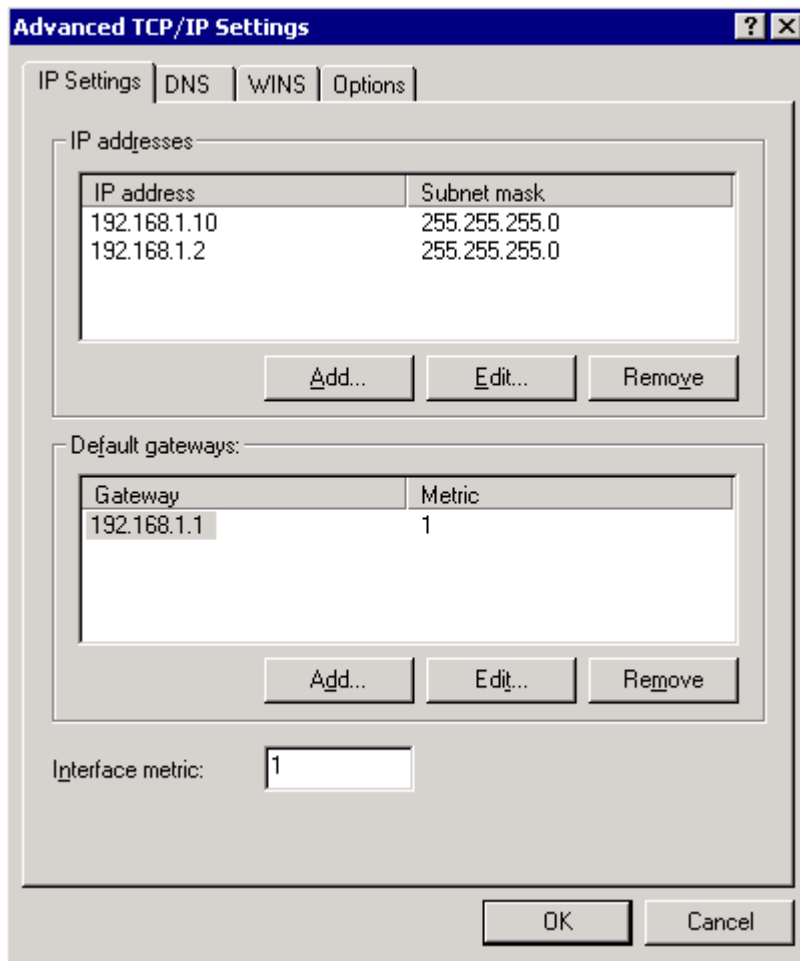
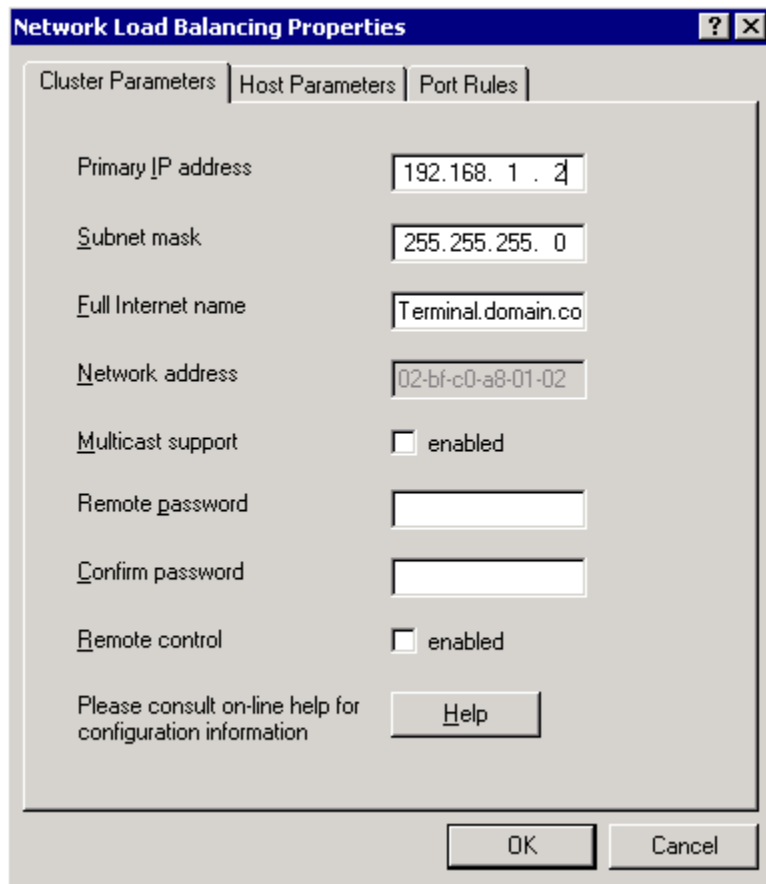


Figure 5.10: Adding the cluster's IP address to each server.

Once you have added the cluster IP to each server, you are ready to configure the Network Load Balancing Service. You do so by going to the properties of the service in the Network Control Panel applet of each server. On the Cluster Parameters tab, which Figure 5.11 shows, you enter information about the farm—the VIP address and hostname and a password if you want to be able to manage the cluster service remotely through a command-line utility. This page should be the same on all servers in the farm.



The screenshot shows a dialog box titled "Network Load Balancing Properties" with three tabs: "Cluster Parameters", "Host Parameters", and "Port Rules". The "Cluster Parameters" tab is selected. The dialog contains the following fields and options:

| | |
|---|----------------------------------|
| Primary IP address | 192.168. 1 . 2 |
| Subnet mask | 255.255.255. 0 |
| Full Internet name | Terminal.domain.co |
| Network address | 02-bf-c0-a8-01-02 |
| Multicast support | <input type="checkbox"/> enabled |
| Remote password | |
| Confirm password | |
| Remote control | <input type="checkbox"/> enabled |
| Please consult on-line help for configuration information | Help |

At the bottom of the dialog are "OK" and "Cancel" buttons.

Figure 5.11: Cluster parameters on the load balancing properties sheet.

Figure 5.12 shows you the Host Parameters tab, which describes the individual host that you are configuring. Here you enter the physical IP address of the server and give the server a priority number that must be unique for each server. The priority setting comes into play when a member of the cluster is not available. The service will then use the priority setting to select which server should take over new requests to the cluster: the lower the number, the higher the priority.

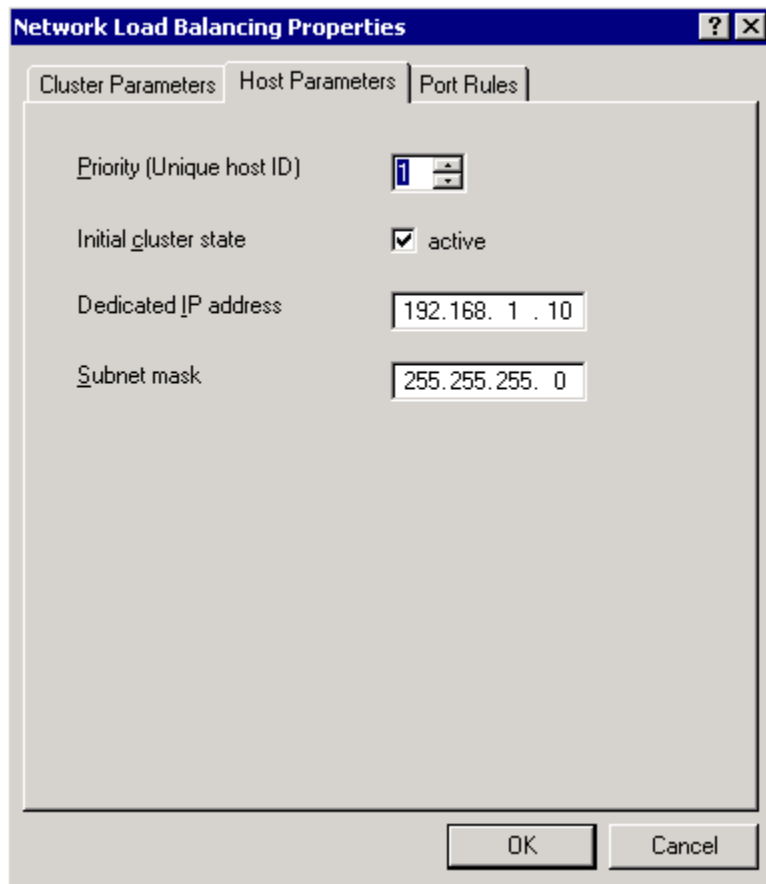


Figure 5.12: Host parameters of the load-balancing service.

The last tab in the properties sheet is the Port Rules tab. When using the load-balancing service for multiple network or database applications, this tab can be very complex. But if you are only worried about running Terminal Services, configuring port rules is quite simple. By default, the entire range of TCP ports is set to balance among the servers; although this setting will work for Terminal Services, you are better off restricting the load balancing to just RDP. This restriction will keep the service from having to evaluate every network request that is directed at the VIP. RDP uses port 3389, so edit the default port rule to filter only this TCP port number. Leave the filtering mode set to multiple hosts, and the affinity mode set to single.

If all servers in the cluster are running equivalent hardware, you can leave the load weight set to equal, but if you have some servers that are more powerful than others, you can specify a percentage of the requests to be handled by each host. Figure 5.13 shows these settings.

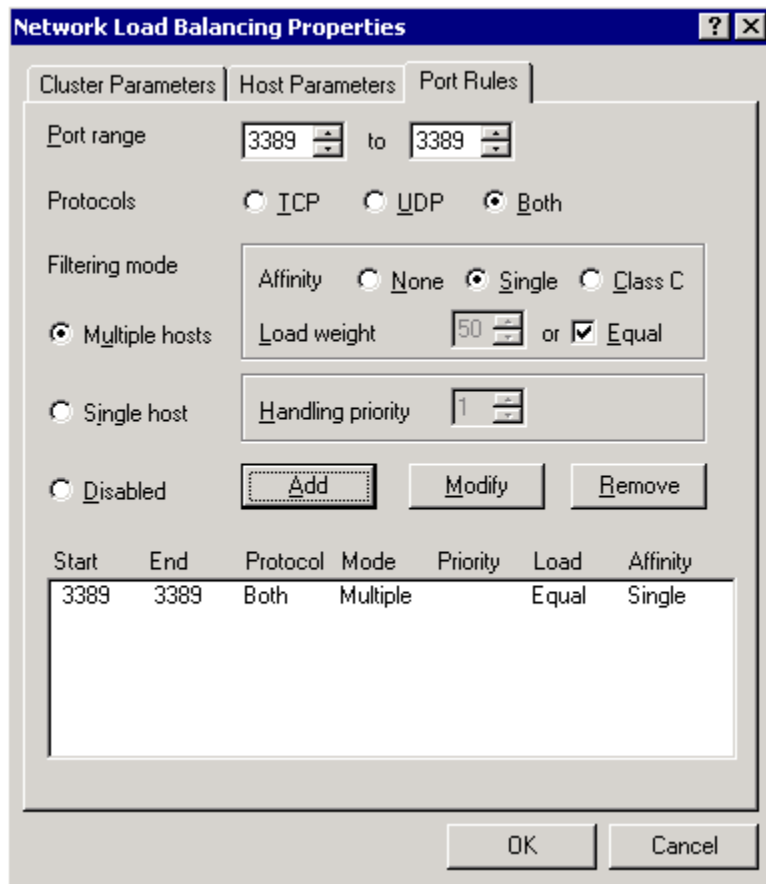


Figure 5.13: The Port Rules tab of a Terminal Services cluster.

Affinity Mode

Affinity mode is very important in a farm of terminal servers. Affinity remembers which server accepted the last request from a specific client IP address. This way, if a user loses his or her connection to the server, the user can reconnect to the same server in the cluster and re-establish the same session (assuming you are allowing a disconnected session to stay alive). One problem that this behavior might create, however, is if a user tries to reconnect from a different client device, the client's IP address will be different, and therefore affinity will not apply. Keep this functionality in mind when considering whether to use load-balancing service, especially for remote access, with which a user's IP can change often.

Controlling Load Balancing

Once you have installed and configured load balancing, you can control the farm from a command line. The Windows Load Balancing Service commands are used to control the service. With them, you can start, stop, enable, disable, or drain the server. The most common command is used to drain a server:

```
WLBS DRAIN ALL
```

This command tells the server to stop accepting new requests but doesn't affect existing connections. Once the last user on the server has logged off, you can perform maintenance or reboot the server without impacting users.


 The Help file for load balancing is very thorough and informative. To access it, click Help in the load balancing properties sheet or type WLBS HELP from a command line.

Figure 5.14 shows an example of a Terminal Services load-balancing group.

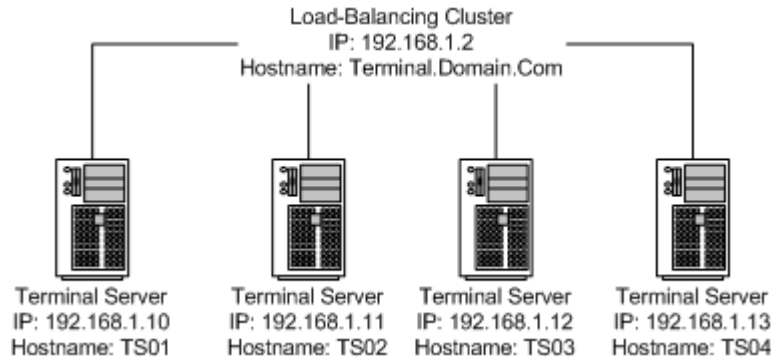


Figure 5.14: Example of a Terminal Services load-balancing cluster.

Application-Level Load Balancing

The major shortcoming of using Windows Load Balancing Service in a terminal server environment is that every server in the farm must have the same applications loaded on it. If you have an application that only a small number of users need, you will still need to load it on every server in the farm.

By using third-party products such as New Moon's Canaveral IQ or Citrix's MetaFrame, you will be able to perform load balancing on a per-application basis instead of at a per-server level. This way, you can publish your core applications from many servers, but offer lesser-used applications from only a few servers. The other major advantage to using a third-party add-on for load balancing is that these products do not require the additional expense of Win2K Advanced Server—you can implement them on the standard Win2K Server OS. Of course, these products offer many more advantages than just application-level load balancing, so be sure to consider them in your Terminal Services design.

Terminal Server Maintenance

In a perfect world, Windows systems would always perform as they are supposed to. Users would be able to log on, log off, open and close applications, and recover from hung applications without causing problems to the system or other users. In reality, however, most Terminal Services administrators have found that applications will occasionally cause memory leaks and user profiles will sometimes become locked on the terminal server. Although your goal should be to find the cause of such problems and correct them, a very helpful task is to implement a maintenance schedule for your terminal servers.



We all accept that we have to reboot a workstation occasionally to clear up a problem. Microsoft has made great strides with Win2K to reduce the frequency of these types of problems, but when you consider that a terminal server is actually many virtual workstations running in one box, you realize that weekly reboots are more for preventative maintenance than an indication of a problem. Please do not take this suggestion as an indication that all Windows systems require regular reboots—doing so is a terminal server-specific practice.

Under WTS, a common practice was to have servers with heavy utilization perform maintenance reboots nightly. Win2K is more robust, and most administrators have found that running weekly maintenance scripts is sufficient. You can perform any number of actions during a maintenance script, but the most common actions are to

1. Disable new logons.
2. Warn any users that are currently logged on that the system will be rebooting so that they can save their work and log out.
3. Force any remaining users to log out.
4. Delete any locally cached user profiles that were not deleted when the user logged off (you should only do so if you are using roaming Terminal Services profiles).
5. Clean the print spooler.
6. Reboot the server.

You can write your maintenance script in any scripting language that you are comfortable with, but because many of the commands you will use are native to Windows, Shell Script is perhaps the easiest to use. Here are a number of commands and utilities that are useful in writing maintenance scripts (these commands are not case sensitive):

- **MSG**—Native on Terminal Services, this command sends a popup message to a user on the terminal server. When followed by an asterisk (*), this command sends the message to all users on the server.
- **SLEEP**—This command is a resource kit utility. You should copy this utility to the \winnt directory of your server, as it is very useful in scripting. SLEEP pauses a batch file for a specified number of seconds.
- **DELPROF**—Native on Win2K, this command deletes local copies of user profiles. When followed by /I /Q, it will ignore errors and not wait for you to OK each deletion.
- **CHANGE LOGON /DISABLE**—Native on Terminal Services, this command disables new logons to the server.
- **LOGOFF RDP-TCP < YES.TXT**—Native on Terminal Services, this command logs off all users connected through RDP. Because there is no quiet switch, you have to pipe a yes command to LOGOFF from a text file containing a Y and a carriage return.
- **TSSHUTDN /reboot**—Native on Terminal Services, this command reboots the terminal server.
- **Net Stop Spooler**—Native on Win2K, this command stops the print spooler so that you can delete any orphaned spool files.

Listing 5.1 shows a sample maintenance script.



```

msg * Please save your work and log off. Weekly reboot in 10 minutes.
sleep 600
msg * Save your work now and log off. Nightly reboot in 5 minutes
sleep 300
msg * Nightly reboot in progress. You will be logged off in 30 seconds
sleep 30
logoff rdp-tcp < yes.txt
delprof /i /q
net stop spooler
del %systemroot%\system32\spool\printers\*. * /q
tsshutdn /REBOOT

```

Listing 5.1: An example maintenance script.

Occasionally, a user's profile will become locked in the registry of the terminal server. In this case, the DELPROF command will not be able to clear it until you either manually unload the user hive by using regedt32 or rebooting the server. There is a shareware utility floating around the Internet called WTS-REGUNLD that will unload all user hives so that you can remove them all with DELPROF.

Once you write and test your maintenance scripts, you can schedule them to run by using the Windows Schedule service (the AT command). If you are using load balancing, be sure to schedule each server at a different time or on a different day so that at least one member of the cluster is available at all times. To schedule the script to run every Monday at 2:00AM, you would use the following command:

```
AT 2AM /every:mon script_name
```

When implementing scheduled reboots, you must be careful to not leave a session logged on at the console of the terminal server. Certain system messages at the console will prevent the server from rebooting.

Be careful when using the TSSHUTDN command. If you type this command with no switches, the default behavior is to shut down the server without rebooting. In the appendix, I will give a VBS script that you can use as a wrapper for TSSHUTDN to prevent inadvertent shutdowns.

Summary

In this chapter, I went into detail about the capacity-planning tools that Microsoft provides. I also examined the Performance Monitor counters that will assist you in watching the health and capacity of your terminal servers. We also looked at the Windows Load Balancing Service—one of the most powerful options available for high-availability Terminal Services designs. Finally, we went over maintenance scripting and the common commands you will use in maintenance

scripts. All the tools and procedures available to you will help you design and maintain a stable, fault-tolerant Terminal Services infrastructure.

In Chapter 6, I will take you through how to secure your Terminal Services design. We will look at the network ports required for terminal server access and set up Terminal Services access through a firewall. I will also go over some common pitfalls that can occur when untrained users and administrators are given Admin rights on a terminal server.

Chapter 6: Securing Terminal Services

In today's world of hackers, crackers, and viruses, security is a top priority for any technology organization. Terminal Services provides IT departments a new set of tools for providing access to the corporate LAN and to vertical applications, but it also presents a unique set of security challenges. Fortunately, with proper planning and implementation, deploying a secure Terminal Services infrastructure can be accomplished.

In this chapter, we will look at a number of security issues that arise when using Terminal Services, and I will present you with options for addressing them. Keep in mind that as with most IT infrastructure designs, there are many ways to accomplish a given task. Your environment might have its own unique security requirements that can either eliminate some of the options I will offer, or present entirely new ways of securing your Terminal Services design. The topics we will cover in this chapter include encrypting RDP, accessing a terminal server through a firewall, deploying a terminal server in an extranet environment, and using remote access strategies.

Using Encryption

Your first line of security when using Terminal Services is encryption. Unlike the traditional computing model, which transfers entire files from the server to the client, Terminal Services typically sends only video data and metafiles to the user. So, even if the data stream could be decrypted, a hacker would only be able to see fragmented bits of images rather than capture a sensitive file. We are more concerned about protecting the keystrokes that the user is sending back to the terminal server—a hacker could theoretically reconstruct a password or even an entire document from this data stream. As a result of this possibility, the data sent from the client to the server is always encrypted, even in low encryption mode.

When you configure RDP on your server, you can select one of three levels of encryption for the server to use. The default encryption level is medium, as Figure 6.1 shows.

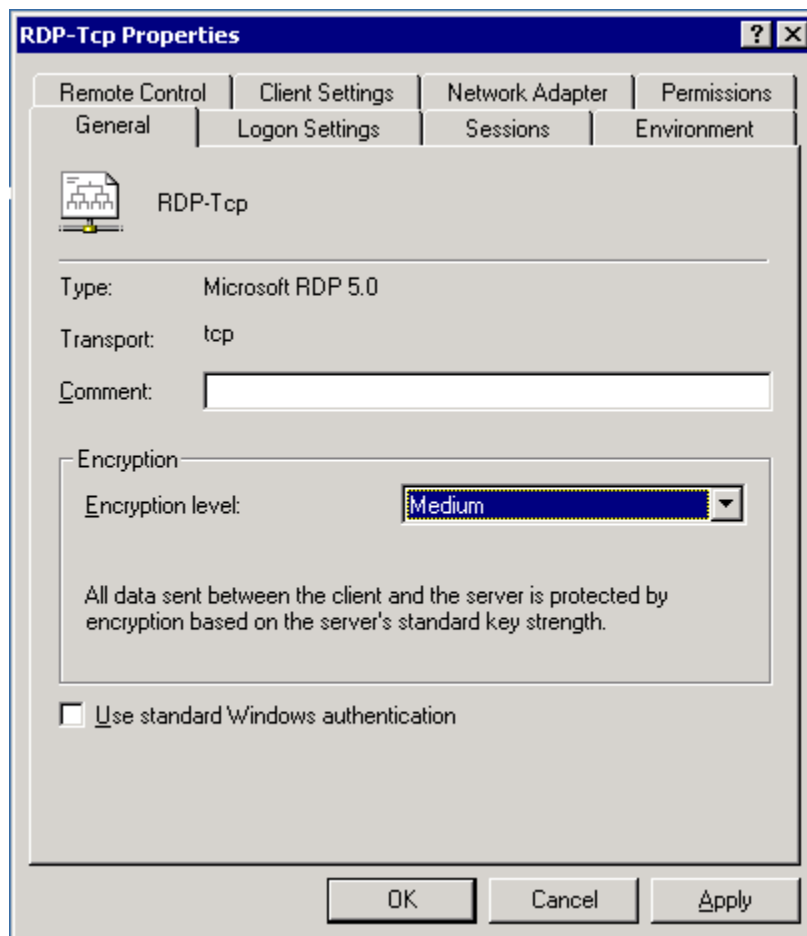


Figure 6.1: Setting the encryption level for RDP.

Microsoft defines the three encryption levels as follows:

- **Low**—With low encryption, traffic from the client to the server is encrypted using the RC4 algorithm and a 56-bit key (40-bit key for RDP 4.0 clients); whereas traffic from the server to the client is unencrypted. Low encryption protects sensitive data such as password entry and application data. The data sent from the server to the client is screen refreshed, which is difficult to intercept even when unencrypted.
- **Medium**—With medium encryption, traffic in both directions is encrypted using the RC4 algorithm and a 56-bit key (40-bit key for RDP 4.0 clients).
- **High**—With high encryption, traffic in both directions is encrypted using the RC4 algorithm and a 128-bit key. In the export version of Terminal Services, high encryption uses the RC4 algorithm and a 56-bit key (40-bit for RDP 4.0 clients).

The bandwidth overhead that is added by using medium encryption is negligible. Thus, unless you have unique networking conditions, I recommend using the default setting even when working within the safety of a corporate network.

Logon Security

If you have implemented some kind of token-based or biometric security in your Windows environment, you must deal with the fact that Terminal Services does not natively support this type of authentication. So, if your security guidelines require it, you will have to place the terminal server behind an authentication server or find a third-party authentication product that supports Terminal Services.

Virus Protection

At one time, an acceptable practice was to not install virus protection software on terminal servers. The software products that were available were not designed for terminal servers and tended to develop memory leaks in a multi-user environment. So administrators did their best to protect the systems using very restrictive access control lists (ACLs) on the file system and registry. Unfortunately, viruses have evolved to a point at which ACLs are no longer a sufficient barrier. Many viruses take advantage of security holes in the OS and applications to function under the system context. Luckily, virus protection software has also evolved, and today there are a number of products available that will function well on a terminal server.

☞ When selecting virus protection software, be sure to choose a product that will allow you to update the virus definition files without rebooting. Otherwise, you will be forced to interrupt your users if you need to apply an emergency update during operating hours. Also, many products use a system-tray icon as a status monitor. Having this monitor program run in each user's session isn't necessary, so you might want to contact the software publisher and see if it can be disabled.

Critical Updates and Internet Explorer Maintenance

If names like Melissa and NIMDA send chills down your spine, you are already aware of the pains we systems administrators go through to keep our systems secure. As new security holes are discovered, Microsoft is quick to release patches and updates for both Win2K and Internet Explorer (IE). Using terminal servers greatly simplifies the deployment of these critical updates. Installing a security patch on a few servers is much easier than deploying the patch to a thousand desktops. To stay up to date on security patches, subscribe to Microsoft's security newsletters at <http://www.microsoft.com/security>.

Most critical updates require the system to be rebooted, so be sure to disable new logons and force any existing users to log off before installing the update. You can disable new logons from a command line using the command

```
CHANGE LOGON /DISABLE
```

Access Through a Firewall

RDP communicates over the TCP port number 3389. If you need to access a terminal server on the other side of a firewall, this port must be open. Also, you must be able to find the server—either by hostname or IP address, so the server must have a routable IP address or utilize port forwarding on your router. (You'd be surprised how many people forget that.) You might want

to only allow inbound port 3389 traffic to the specific IP addresses of your terminal servers or allow outbound traffic on port 3389 only to trusted terminal servers on the outside. Figure 6.2 shows the data flow when accessing a terminal server on either side of a firewall.

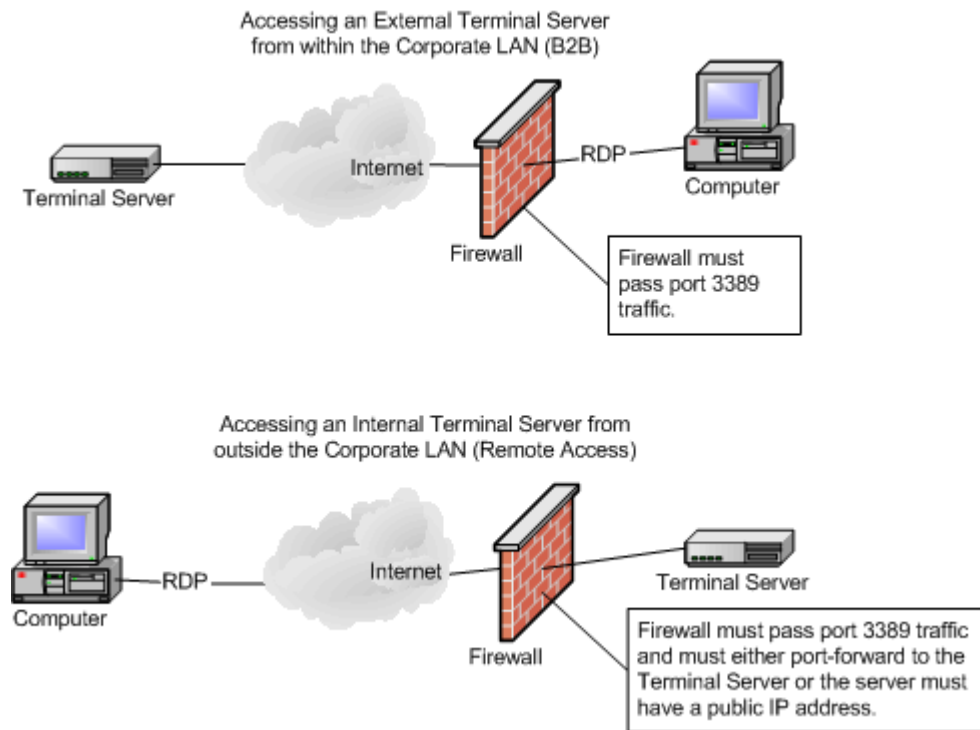


Figure 6.2: Accessing a terminal server through a firewall.

Another, often more tempting, option when your users need to access a terminal server on the outside of a firewall is to use a Windows Sockets (SOCKS) proxy. Unfortunately, using a SOCKS proxy is only possible when you use the Citrix Independent Computing Architecture (ICA) protocol. At this point, Microsoft’s RDP client does not support connections by way of SOCKS.

Implementing Terminal Services in an Extranet Environment

As a result of the many complex data flows required by today’s business-to-business (B2B) marketplace, many enterprises are building extranets at the periphery of their LANs. This setup enables them to place systems in a “buffer zone” in which they have control over access from both the external networks as well as the intranet. If you are deploying a vertical application to which both internal and external users require access, then an extranet may be the perfect solution.

Think of an extranet as a network segment with firewalls at both the trusted and untrusted sides. One major advantage to this model when using Terminal Services is that once an outside user has established a session on the terminal server, the user still does not have full access to the corporate LAN—only to systems, IP addresses, and ports that you have specified on the trusted-side router. Figure 6.3 shows a simplified version of this model.

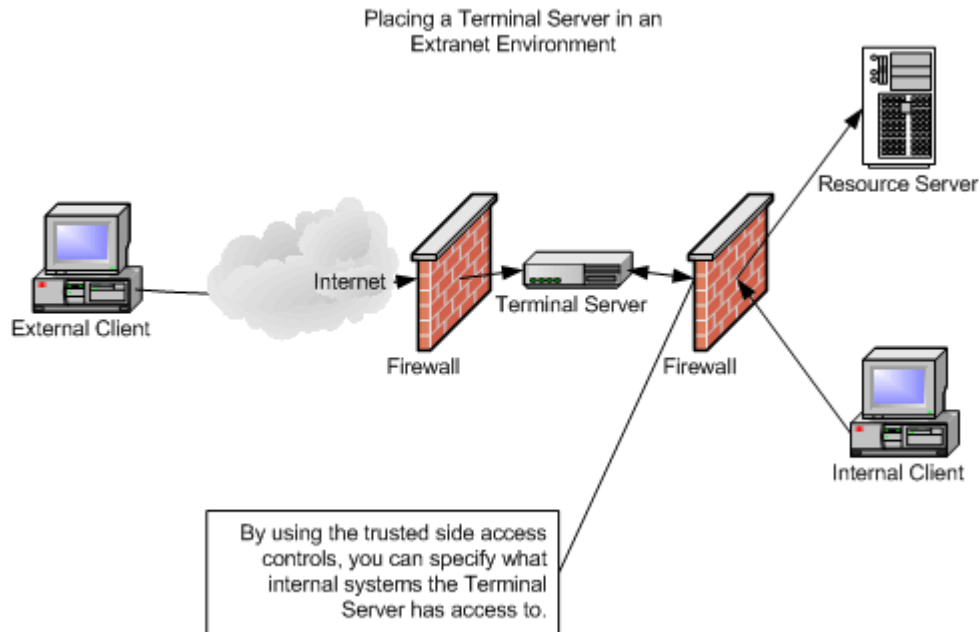


Figure 6.3: Placing a terminal server in an extranet.

👉 If users of a terminal server in the extranet require access to internal domain resources, you will need to define data flows for domain authentication on the internal access controls.

Application-Layer Security

After you have determined how your users will gain access to the terminal server itself, you can implement application-layer security on the terminal server. You do so by using a resource kit utility called the Application Security tool, or AppSec, and implementing a Group Policy—either domain or local—that restricts which executables users of the terminal server can invoke.

💣 The version of AppSec that is included with the Win2K resource kit is missing required files. To update your kit, download AppSec.zip from <ftp://ftp.microsoft.com/reskit/win2000>.

With AppSec, you define a list of executables that non-administrators can invoke. AppSec monitors the CreateProcess application programming interface (API) thread and only allows specified processes to be created. When you first launch AppSec, you will see a predefined list of applications that are allowed. These executables are required for the system to run, and you should not remove them without serious testing. Figure 6.4 shows the AppSec interface.

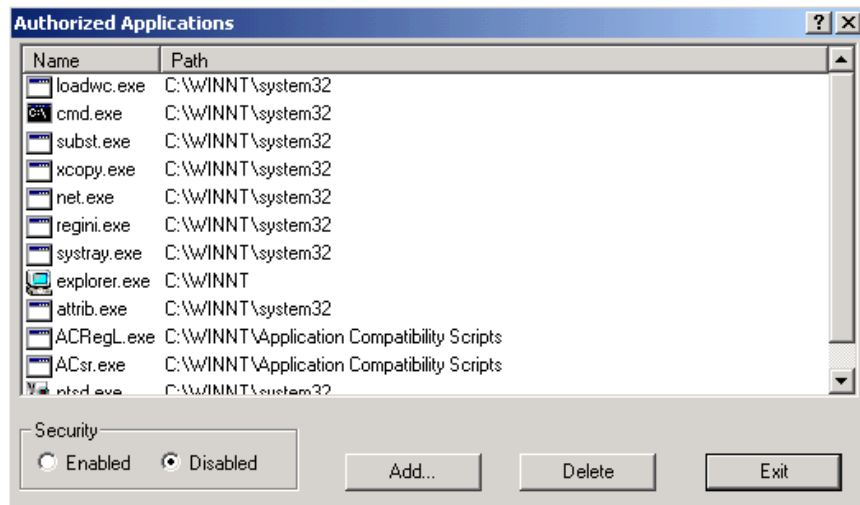


Figure 6.4: The AppSec utility.

When you enable AppSec security, 16-bit applications are disabled by default. If your users need access to any 16-bit programs, you must add NTVDM.EXE to the access list.

In addition to enabling application security, you will also want to apply the *Run only allowed Windows applications* policy. This policy prevents users from seeing disallowed applications in their Start menu, as well as adding a second layer of security by preventing the system from invoking other application from Explorer or a Run command. This setting is found in the Group Policy Editor under User Configuration, Administrative Templates, System. Figure 6.5 shows the editor for this policy.

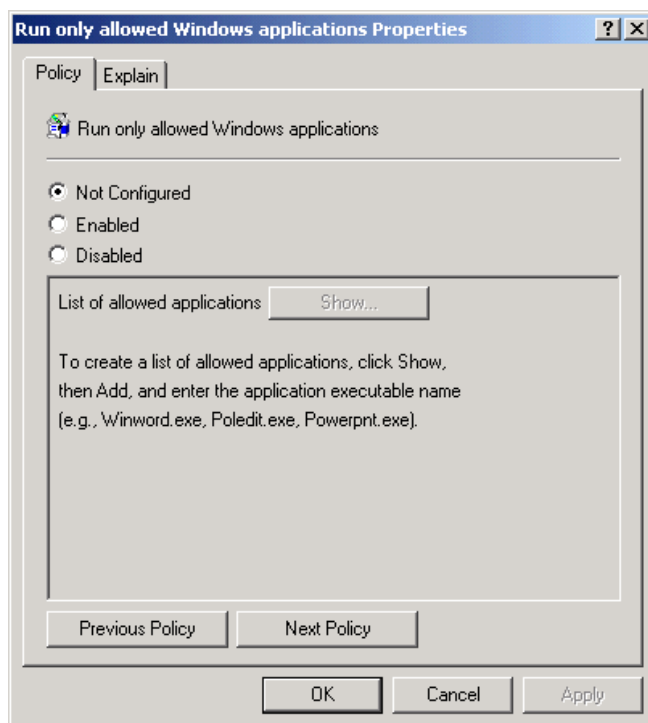


Figure 6.5: Setting the Run only allowed Windows applications Group Policy.


You should be aware that if you do not take precautions by restricting file-level access, a malicious user could bypass both AppSec and the policy by renaming an executable, then invoking it. Both security measures monitor threads by name, so by renaming WINWORD.EXE to SYSTRAY.EXE, a user could start Word because SYSTRAY is an allowed program.

File and Registry Security

Back in Chapter 2, you saw that there are two options for local file and registry permissions—Win2K permissions and NT 4.0 WTS permissions. You can select the permission mode when installing Terminal Services, and change it using the Terminal Services Configuration administrative tool.

Microsoft does not provide any explicit documentation about the exact differences between the two modes other than stating that in NT 4.0 compatibility mode, users have full access to critical registry and file system locations. Thus, if you are required to run a terminal server in NT 4.0 compatibility mode because of a legacy application, you should take extra care to be sure that registry-editing tools and access to the hard drive of the terminal server are disabled in your users' Group Policies.

Better yet, use registry and file-access monitoring tools to pinpoint the exact keys and files to which the legacy application needs access and manually grant permission to those files. That way, you can run the terminal server in Win2K mode and protect the rest of the HKEY_LOCAL_MACHINE registry key and the file system.

 Winternals Software publishes excellent file and registry monitoring tools that are available at (<http://www.winternals.com>).

For example, if an application errors out under Win2K permissions mode for a user who is not in the Administrators group, and the application runs correctly under NT 4.0 compatibility mode, you would leave the server in NT 4.0 compatibility mode and run the application while using a file-access monitoring tool. The logs generated by the monitoring utility may indicate that the application needs write access to the program's application directory. You would then grant the Authenticated Users group write permission on that specific directory under C:\Program Files, leaving the rest of the Program Files directories protected.

Now change the server back to Win2K permission mode and test the application again using a non-administrative account. If the application now runs, you can add this ACL change to your installation documentation and continue to benefit from the safety of Win2K permission mode. You can diagnose and correct registry permission problems in the same way.

Remote Access

If you are implementing Terminal Services as a remote access option for your users, often, a desirable option is to use security measures that are independent of the terminal server. This way, you can treat the server as just another workstation within the corporate LAN.

There are many options for a secure remote-access model, and most organizations will already have one in place. Terminal Services will function as long as your Remote Access Service (RAS) model supports TCP/IP connectivity and will pass port 3389. You can use a virtual private

network (VPN), Point-to-Point Tunneling Protocol (PPTP), IP Security (IPSec), HTTP over Secure Sockets Layer (HTTPS), and any of the token-based security measures that are common today. Once a user has been granted trusted access to the LAN, connecting to the terminal server and logging onto the domain become easy.

Advantages of Using Terminal Services for Remote Access

When designing your remote access infrastructure, you might want to consider using the terminal server as a portal to the corporate LAN. This way, in addition to the security measures that your RAS solution provides, you can further restrict access by only allowing port 3389 and requiring that all remote users work through Terminal Services.

Most security administrators see Terminal Services as a great tool for locking down the access that external users have. The terminal server, or cluster of terminal servers, becomes a single point of entry which can be easily monitored, audited, and disabled if necessary.

Summary

Terminal servers centralize your computing environment, often making it easier to secure. They also provide new options for your remote-access strategy. If proper care is taken when designing and deploying Terminal Services, you can create an environment that is highly stable and secure.

If you take Terminal Services security in layers—starting with RDP encryption to communicate with the terminal server and moving outwards in your design from registry, file, and application security—by the time you reach the secure boundary of your corporate LAN, you can easily implement a routing or remote access solution that meets your needs. Be sure to work closely with your network engineers to come up with a design that pleases everyone.

Afterword: The Future of Terminal Services

There are many changes coming in the landscape of Terminal Services technology. New products are being released to take advantage of centralized computing and enhancements to RDP are being made. Microsoft has even integrated Terminal Services technology into their new desktop OS—Windows XP. I'll show you a few of the new options that are on the horizon so that you can keep them in mind as you design your Terminal Services infrastructure.

RDP 5.1

Perhaps the greatest change to the Terminal Services landscape is RDP 5.1. This new protocol is available today as part of Windows XP, and you can download the new Remote Desktop client from Microsoft's Web site at

<http://www.microsoft.com/windowsxp/pro/downloads/rdclientdl.asp>

This new client supports both the Remote Desktop functionality of Windows XP as well as RDP 4.0 and 5.0 connections to existing terminal servers. The client also includes support for connection files (text files with a file extension of .RDP), which allow you to save, edit, and distribute connection definitions without having to edit the registry. Figure A.1 shows the new interface.

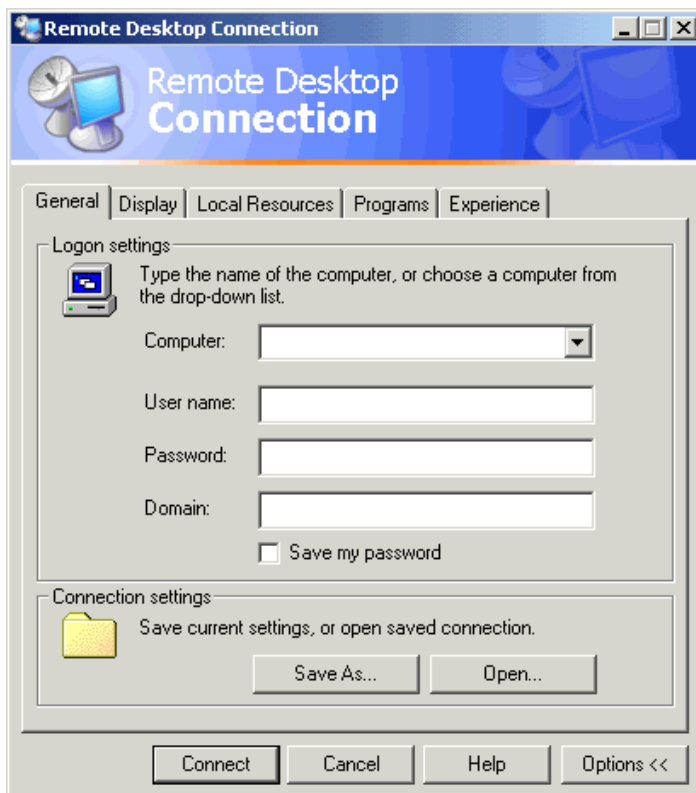


Figure A.1: The new Remote Desktop connection client interface.

RDP 5.1 will also be available on Microsoft's upcoming Windows .NET Server platform—the successor to Win2K Server. The new version of the protocol adds native support for client drive

mapping, audio, and a new connection bar that users can “pin” to the top of their screen during full-screen Terminal Services sessions. The connection bar lets you easily determine which server or Remote Desktop you are working on.

Video resolution and color depth are also enhanced in RDP 5.1. The protocol now supports as high as 24-bit color (although the client GUI offers only as high as 16-bit support). Figure A.2 shows the Display configuration tab of the new client.

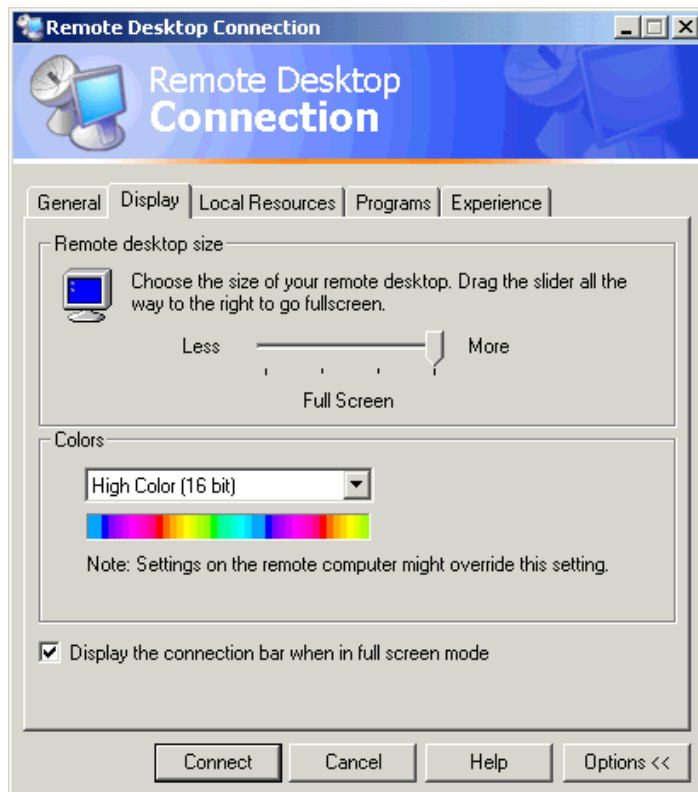


Figure A.2: The Display configuration tab of the RDP 5.1 client.

Remote Desktop

Windows XP Professional includes a new feature called Remote Desktop. This functionality takes advantage of Microsoft’s Terminal Services technology and the new RDP 5.1 protocol to allow you to remotely connect to your workstation. With Remote Desktop enabled, you can redirect your video, keyboard, and mouse from the local console to the RDP stream, thus making your computer available from anywhere on the network.

This ability might completely change how we use Terminal Services for remote access and mobile enterprise users. Why connect to a terminal server that offers a generic set of applications when you can connect directly to your personal computer and have the exact same experience and applications that you have at your office?

Terminal Services will certainly maintain its place as the platform of choice for the application service provider (ASP) model and B2B connectivity. And the advantages of thin client devices will still play a role in larger homogeneous computing environments because these devices

virtually eliminate end-node support and increase the hardware lifecycle far beyond that of a traditional PC.

Remote Assistance

If you have supported users in a Terminal Services environment, you know the advantages of using Remote Control. With this ability, you can assist your users in configuring their applications, demonstrate processes to new users, and correct technical problems without visiting the user's desk. Windows XP's Remote Assistance offers these same abilities on the desktop platform. Remote Assistance also takes advantage of Terminal Services technology and the RDP 5.1 protocol to enable support personal to shadow their users. Remote Assistance requires that the computers at both ends of the connection be using Windows XP.

Both Remote Desktop and Remote Assistance are configured through the System Control Panel applet on Windows XP, as Figure A.3 shows. In an enterprise, you can also configure these features through Group Policy. You can specify which users and groups can request or provide Remote Assistance, and select users and groups that can open Remote Desktop connections.

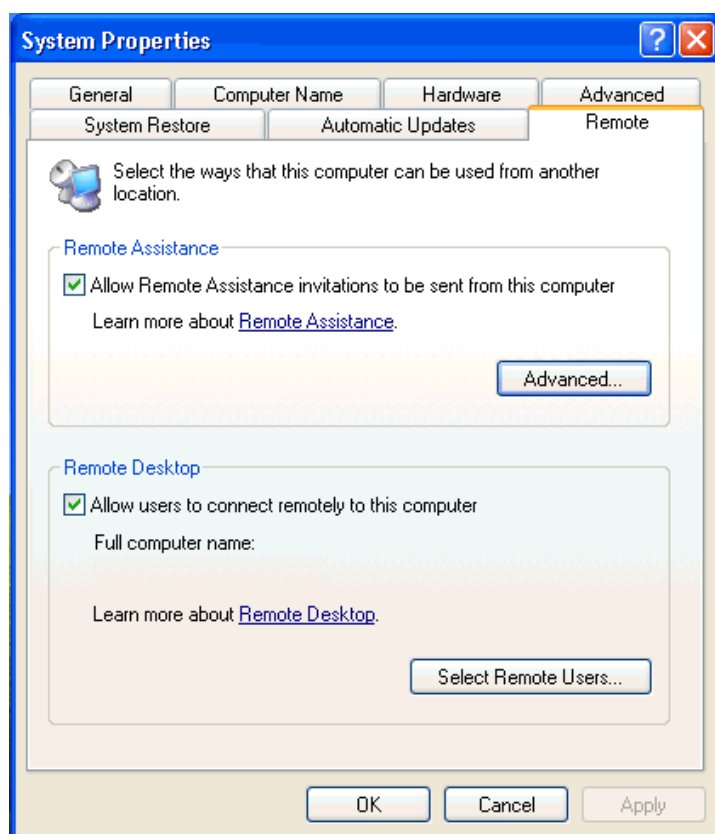


Figure A.3: Configuring Remote Assistance and Remote Desktop in the System Control Panel applet.

Windows .NET Server

The current beta of Microsoft Windows .NET Server offers a number of enhancements to Terminal Services. First, in .NET Server, Microsoft has separated all Terminal Services-related

parameters of user accounts into their own attributes, so administrators can now populate, query, and modify these settings through standard Active Directory Service Interfaces (ADSI) calls.

In addition, .NET Enterprise Server and .NET Datacenter Server feature a new technology called Terminal Services Session Directory, which will enhance a user's ability to reconnect to an active session on a terminal server cluster.

All versions of .NET Server also include the new RDP 5.1 protocol and a new advanced client.

 For more information about Windows .NET Server, see <http://www.microsoft.com/windows/netserver/>

Third-Party Products

As Microsoft releases these new technologies, third-party application vendors will be enhancing their products as well. I'm sure in the near future we will see such technologies as integrated file associates for applications provided by a Terminal Services ASP solution. Imagine being able to double click on a .DOC file on a network share and have your system automatically spawn a connection to a terminal server that provides you with Microsoft Word.

Hardware vendors are also keeping in stride. There are now thin client devices integrated into flat-panel monitors and tablet devices that use 802.11 wireless LAN to connect you to a terminal server. New Pocket PC devices can natively communicate with terminal servers, so we may soon see Win32 applications designed specifically for that form factor.

Summary

As we look to these new technologies over the next year, the gap between locally installed applications and Terminal Services/ASP applications will become much smaller. We, as Terminal Services administrators, will have an easier time selling the benefits of Terminal Services when there are fewer differences for the end user.

Also, in a desktop replacement environment, more and more application vendors are complying with the Microsoft Windows Logo specification, so application integration on Terminal Services is becoming easier. I am already seeing more and more terminal servers without a single application compatibility script, and I spend less and less of my time creating wrappers and special installation instructions for legacy programs.

If you work in a mixed Terminal Services and workstation environment, be sure to get involved in the Windows XP deployment. You will need to take XP's new features into account when planning for the future of your terminal servers. And your experience and knowledge with Terminal Services technologies will make the integration of Remote Desktop and Remote Assistance much easier for your enterprise. It is an exciting time for Windows technologies, and Terminal Services will play an important role as computing becomes more mobile and pervasive.

I'd like to thank you all for reading *The Definitive Guide to Windows 2000 Terminal Services*. I have enjoyed writing it. I'd like to thank the people that helped me make this book as accurate and detailed as it is—Scott Hill, my technical editor; Laurie Nocella, my style editor; everyone at New Moon; my partner, Michel, for all the encouragement, and of course, Sean Daily and David Templeton of Realtimerepublishers for making it all happen.

Appendix A: Terminal Services Clients

NEW: RDP 5.1 client for Terminal Services and Remote Desktop:

<http://www.microsoft.com/windowsxp/pro/downloads/rdclientdl.asp>

Terminal Services Advanced Client (Web-based client):

<http://www.microsoft.com/windows2000/server/evaluation/news/bulletins/tsac.asp>

RDP 5.0 32-bit client in MSI format:

<http://www.microsoft.com/windows2000/downloads/recommended/TSAC/tsmsi.asp>

Terminal Services Connection MMC snap-in client:

<http://www.microsoft.com/windows2000/downloads/recommended/TSAC/tsmmc.asp>

NEW: Terminal Services Client for Pocket PC (2000 and 2002):

<http://www.microsoft.com/mobile/pocketpc/downloads/terminalservices/default.asp>

Terminal Server Client for Windows CE Pro:

<http://www.microsoft.com/mobile/downloads/ts-final.asp>

Configure a Workstation to Act Like a Thin Client

The following steps walk you through how to configure an NT 4.0 Workstation or Win2K Pro system to act like a thin client.

1. Install the Terminal Services Client—for this, you must use the RDP 5.0 client.
2. Create a new local machine account and place it in the local administrators group.
3. Add
`c:\progra~1\termin~1`
to the path.
4. Log out and log on again as the new account you created in Step 2.
5. Launch the Client Connection Manager and create or import your Terminal Services connection definitions.
6. Apply the following registry settings (I used regedit4 because these settings work on both NT 4.0 and Win2K):

REGEDIT4

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon]
"AutoAdminLogon"="1"
"Shell"="conman.exe"
"DefaultUserName"="<username of the account you created>"
"DefaultPassword"="<password of the account you created>"
"DontDisplayLastUserName"="0"
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Pol
icies\System]
"DisableLockWorkstation"=dword:00000001
[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Pol
icies\Explorer]
"NoRun"=dword:00000001
```

7. Remove the account you created from the local admin group.
8. Reboot.



To bypass the automatic logon, hold down the Shift key as Windows starts. You can then log on as local administrator. Use Ctrl+Alt+Delete to bring up Task Manager, and launch regedit from the new task button. Now you can change the WINLOGON/Shell value back to EXPLORER.EXE (the default).

Appendix B: Important URLs

Terminal Services end-user license agreement:

<http://www.microsoft.com/windows2000/server/howtobuy/pricing/terminal.asp>

Terminal Services licensing enhancements hotfix:

<http://www.microsoft.com/windows2000/downloads/critical/q287687/default.asp>

Microsoft, NEC, and Groupe Bull Terminal Services sizing white paper:

<http://www.microsoft.com/windows2000/techinfo/administration/terminal/tscaling.asp>

Terminal Services sizing sample scripts:

<http://www.microsoft.com/windows2000/techinfo/administration/terminal/loadscripts.asp>

Terminal Services sizing utilities hotfix for the Win2K resource kit:

http://download.microsoft.com/download/win2000platform/Tsct/1.0/NT5/EN-US/tsct_hotfix.exe

Terminal Services licensing Web site:

<https://activate.microsoft.com>

Microsoft's "Certified for Windows" logo specifications:

<http://msdn.microsoft.com/certification/appspec.asp>

Terminal Services APIs white paper:

<http://www.microsoft.com/ntserver/zipdocs/TseApis.exe>

Microsoft application security utility hotfix for the Win2K resource kit:

<ftp://ftp.microsoft.com/reskit/win2000>

Appendix C: Registry Changes

Listed here are all the registry changes described in Chapter 2. I have formatted them to allow you to copy and paste them into a .REG file for easy import.

```
Windows Registry Editor Version 5.00
```

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\
Policies\Explorer]
```

```
"LinkResolveIgnoreLinkInfo"=dword:00000001
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server]
```

```
"IdleWinStationPoolCount"=dword:00000005
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp\UserOverride\Control Panel\Desktop]
```

```
"AutoEndTasks"="1"
```

```
"CursorBlinkRate"="-1"
```

```
"DragFullWindows"="0"
```

```
"MenuShowDelay"="10"
```

```
"WaitToKillAppTimeout"="20000"
```

```
"SmoothScroll"=dword:00000000
```

```
"Wallpaper"="(none)"
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-Tcp\UserOverride\Control
Panel\Desktop\WindowMetrics]
```

```
"MinAnimate"="0"
```

Appendix D: Script Reference

USRLOGON.CMD

Listing A.1 is the modified USRLOGON.CMD which takes advantage of Win2K's ability to map the home directory to the user's folder. Deletions from the original USRLOGON.CMD are in ~~STRIKETHROUGH~~ and additions are in **BOLD**. To use this script, you should set your ROOTDRIVE to the same drive letter that you use for your users' network home directories.

```
@Echo Off

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

Rem
Rem This is for those scripts that don't need the RootDrive.
Rem

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

:cont0

Rem
Rem Determine the user's home directory drive letter.  If this isn't
Rem set, exit.
Rem

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

Rem
Rem Subst the user's profile directory onto the ROOTDRIVE
Rem if it is not already mapped as the Home Directory
Rem

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
```

```

Rem
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 A.1: Modified USRLOGON.CMD.

TSSHUTDOWN Wrapper

TSSHUTDOWN.EXE is the native utility you use to shut down or reboot a terminal server. This utility warns users that the terminal server is about to shut down so that they have time to save their work. The program then executes a logoff command within each session, and finally shuts down or reboots the server. Unfortunately, the default behavior of the utility is to *shut down* the server in 30 seconds rather than *reboot* the server.

I wrote the wrapper that Listing A.2 shows to prevent accidental shutdowns. Copy the following code, and save it as a VBS file in the SYSTEM32 directory. To launch the wrapper, type the VBS file's name at a command prompt.

```

' AUTHOR: Greyson Mitchem, The Definitive Guide to W2K TS
'
' This VBS file is a safety wrapper for the native
' TSSHUTDOWN.EXE that shuts down or reboots a TS.
' This script will collect the parameters for
' the tsshutdn command.

set oShell=CreateObject("Wscript.Shell")
set oNet=CreateObject("Wscript.Network")

returnkey=msgbox("This script collects the parameters needed
to"&VBCRLF&"correctly shutdown or reboot a Terminal
Server."&VBCRLF&"You may abort the process at any time by hitting
CANCEL."&VBCRLF&VBCRLF&"Do you wish to continue?", VBOkCancel +
VBIInforamtion, "TS Shutdown")
IF returnkey=VBCancel THEN Wscript.Quit(1)

Server=InputBox("Enter the name of the server you wish to
Reboot/Shutdown:", "Server Name", oNet.ComputerName)
if trim(Server)="" then Wscript.Quit(1)

bRestart=MSGBox("Do you wish to have the server
reboot?"&VBCRLF&VBCRLF&"Clicking NO will PowerDown the system without
rebooting.", VBYesNoCancel + vbQuestion, "Reboot or PowerDown")
Select Case bRestart
    Case vbYes

```

```

        sOption="/REBOOT"
    Case vbNo
        sOption="/POWERDOWN"
    Case vbCancel
        Wscript.Quit(1)
End Select

Wait=InputBox("Please enter the number of seconds to give users to
finish working before forcibly logging them off:", "Wait Time", "60")
if trim(Wait)="" then Wscript.Quit(1)

Delay=InputBox("Please enter the number of seconds to wait after all
users have logged off before shutting down the system:", "Wait Time",
"30")
if trim(Delay)="" then Wscript.Quit(1)

' All Parameters have been collected. Now we build the comand line.


ShutdnCMD="tssshutdn.exe "&wait&" /server:"&Server&" "&sOption&"
/Delay:"&Delay&" /v"

oShell.RUN(ShutdnCMD)

Wscript.Quit(0)

```

Listing A.2: TSSHUTDN wrapper.

 If you want to create a shortcut to the wrapper, make the target
 "wcript %systemroot%\system32\

Maintenance Reboot Script

Listing A.3 shows an example of a shell script that you can use to perform a maintenance reboot on a terminal server. Your circumstances—number of users, roaming profile environment, applications installed, and so on—will determine the frequency that you may need to schedule maintenance reboots. For this script to run correctly, you need two additional files in the directory with the script:

- The Sleep.EXE resource kit utility.
- A text file named yes.txt with the letter Y as its contents. Copy this code and paste it into a .BAT or .CMD file.

```

REM
REM Sending a message to any currently logged-on users
REM warning them that a maintenance reboot will occur
REM in 10 minutes.
REM

msg * Please save your work and log off. Maintenance reboot in 10
minutes.

REM Pausing for 5 minutes
sleep 300

```

```
REM 5 minute warning
msg * Save your work now and log off. Maintenance reboot in 5 minutes

REM Pausing for 5 minutes
sleep 300

REM 30 second warning
msg * Maintenance reboot in progress. You will be logged off in 30
seconds

REM Pausing for 30 seconds
sleep 30

REM Logging all users off
logoff rdp-tcp < yes.txt

REM Stopping the Print Spooler service and deleting any
REM orphaned files.
net stop spooler
del %systemroot%\system32\spool\printers\*. * /q

REM Rebooting the TS
tsshutdn /REBOOT
```

Listing A.3: An example maintenance reboot script.

Appendix E: Terminal Services Command-Line Reference

Use the following commands for Terminal Services administration from the command line.

Change User

This command toggles the system between install and execute modes. To switch the terminal server to install mode:

```
CHANGE USER /install
```

To switch the terminal server to execute mode:

```
CHANGE USER /execute
```

Change Logon

This command enables or disables new logons to the terminal server; it does not affect currently logged on users. To enable new logons:

```
CHANGE LOGON /enable
```

To disables new logons:

```
CHANGE LOGON /disable
```



When you reboot a terminal server, logons are automatically enabled; even if they were disabled when you shut down the system.

Query Terminal Servers

This command lists all active terminal servers in the current or specified domain:

```
QUERY TERMSERVER [servername] [/domain:domain] [/address]
[/continue]
```

where

- *servername* is the name of a specific terminal server that you want to query.
- */domain:domain* is the name of the domain you want to query. The default is to query the current domain.
- */address* includes the IP address of each server in the output.
- */continue* does not pause between screens of output.

Query Session

This command lists all current sessions on a specific terminal server:

```
QUERY SESSION [sessionname | username | sessionid]
[/server:servername] [/mode] [/flow] [/connect] [/counter]
```

where

- *sessionname* is the name of a specific session that you want to query.



- *username* is the name of the specific user you want to query.
- *sessionid* is the ID of the specific session you want to query.
- */server:servername* is the name of the server you are querying (the default is the server you are logged on to).
- */mode* outputs the current line settings.
- */flow* outputs the current flow control settings.
- */connect* outputs the current connection settings.
- */counter* outputs the counter information for the server.

Query User

This command lists all current users with sessions on a terminal server:

```
QUERY USER [username | sessionname | sessionid]
[/server:servername]
```

where

- *sessionname* is the name of a specific session that you want to query.
- *username* is the name of the specific user you want to query.
- *sessionid* is the ID of the specific session you want to query.
- */server:servername* is the name of the server you are querying (the default is the server you are logged on to).

Query Process

This command lists processes running on the terminal server and can be filtered to a specific session:

```
QUERY PROCESS [* | processid / username | sessionname | /id:nn |
programname] [/server:servername] [/system]
```

where

- *** lists all processes on the terminal server.
- *processed* lists information about only the specific process ID.
- *username* lists processes running under the context of a specific user.
- *sessionname* lists processes running under the context of a specific session.
- */ID:nn* lists processes running in the session with the specified session ID number.
- *programname* lists all processes started by the specified executable.
- */server:servername* is the name of the server you are querying (the default is the server you are logged on to).
- */system* lists processes running under the system context.

Logoff

This command logs off a user from the terminal server and deletes the session. If no arguments are included, the command logs you out:

```
LOGOFF [sessionid | sessionname] [/server:servername] [/v]
```

where

- *sessionid* is the ID of the session you want to log off.
- *sessionname* is the name of the session you want to log off.
- */server:servername* specifies the name of server on which the session you want to log off is running (the default is the server you are connected to).
- */v* displays verbose information about actions being performed.

Message

This command sends a popup message to a user or users on the terminal server:

```
MSG [username | sessionname | sessionid | @filename | *]
[/server:servername] [/time:seconds] [/v] [/w] message
```

where

- *username* is the name of the user to whom you are sending the message.
- *sessionname* is the session name to send the message to.
- *sessionid* is the ID number of the session to send the message to.
- *@filename* is the name of a text file containing user names, session names, or session IDs to send the message to.
- *** sends the message to all users on the current or specified server.
- */server:servername* specifies the server to which recipients of the message are connected.
- */time:seconds* the number of seconds to display the message before the popup message box closes itself.
- */v* displays information about the message as it is sent.
- */w* causes the popup window to wait for the user to click OK before closing.
- *message* is the text of the message to send.

Reset Session

This command terminates a user's session without warning the user, and without performing a graceful logoff; can be used to terminate a hung session:

```
RESET SESSION [sessionname | sessionid] [/server:servername] [/v]
```

where

- *sessionname* is the name of a specific session that you want to reset.
- *sessionid* is the ID of the specific session you want to reset.

- `/server:servername` is the name of the server on which the session resides.
- `/v` displays verbose information about actions being performed.

Shadow

This command begins a remote control session:

```
SHADOW [sessionname | sessionid] [/server:servername] [/v]
```

where

- `sessionname` is the name of a specific session that you want to shadow.
- `sessionid` is the ID of the specific session that you want to shadow.
- `/server:servername` is the name of the server on which the session resides.
- `/v` displays verbose information about actions being performed.

Terminal Services Profile

This command populates the terminal service profile path of a specified user; you can also use it to copy the terminal server profile contents from one user to another. This command requires Administrator rights:

```
TSPROF /update [/domain:domainname | /local] /profile:path  
username
```

```
TSPROF /copy [/domain:domainname | /local] [/profile:path]  
src_user dest_user
```

```
TSPROF /q [/domain:domainname | /local] username
```

where

- `/update` populates the `domainname\username` of the user's Terminal Services profile path with `path`.
- `/copy` copies the Terminal Services profile from `src_user` to `dest_user` and, if specified, updates the Terminal Services profile path of `dest_user` with `path`.
- `/q` displays the Terminal Services profile path for the specified user.

Terminal Services Shutdown


This command shuts down or reboots a terminal server in an orderly fashion, giving any current users the opportunity to save their work and log off.

```
TSSHUTDN [wait_time] [/server:servername] [/reboot] [/powerdown]  
[/delay:logoffdelay] [/v]
```

where

- `wait_time` is the number of seconds to wait after notifying the users that the terminal server is about to shut down before forcibly logging them off. The default is 30 seconds.

- `/server:servername` is the name of the server to reboot/shutdown. The default is the server you are connected to.
- `/reboot` reboots the server.
- `/powerdown` powers down the server after Windows has shutdown. The server's BIOS must support this command.
- `/delay:logoffdelay` the number of seconds to wait after logging off all users before shutting down the system. The default is 30 seconds.
- `/v` displays verbose information about actions being performed.

 See Appendix D for a VBS wrapper for TSSHUTDOWN to make entering the parameters easier.

Appendix F: SMCLIENT Scripting Language Reference

SMCLIENT is the simulated client utility included in the Terminal Services capacity planning toolkit. The following syntax is from the SMCLIENT.DOC file included with the tools. The following identifiers are reserved for use as client simulation keywords, and may not be used otherwise:

- check
- connect
- disconnect
- job
- logoff
- loop
- senddata
- sendoutput
- sendtext
- sleep
- start
- call

Connect

A connection between a user and a terminal server is made using a connect statement:

```
connect (<user_name>, <password>, <domain>, <xResolution> *  
<yResolution>);
```

where

- *user_name* is the user's name attempting to connect to the terminal server.
- *password* is the user's password.
- *domain* is the user's domain.
- *xResolution* * *yResolution* specifies the connection resolution (for example, 640 * 480)

Logoff

To log off a user from a terminal server use the logoff statement. The user is logged off from the last opened connection:

```
logoff ();
```

Disconnect

To disconnect a user from a terminal server use the disconnect statement. The user is disconnected from the last opened connection:



```
disconnect ();
```

Start

To launch an application in the last opened connection, use the start statement:

```
start (<application_command_line>);
```

where *application_command_line* is the application command line.

Sendoutput

To send events (keyboard and mouse events) to the terminal server, use the sendoutput statement. Events are sent through the last opened connection:

```
sendoutput (<events_file>, <milliseconds>);
```

where

- *events_file* is the file that contains the keyboard and mouse events to send to the terminal server.
- *milliseconds* specifies the time, in milliseconds, for which to suspend execution after each command sent from the *events_file* file.

Senddata

To send a specific event (keyboard or mouse) to the terminal server, use the senddata statement. The specified event is sent through the last opened connection:

```
senddata (<message_name>, <wparam>, <lparam>);
```

where

- *message_name* is the message name to be sent to the terminal server. The only supported messages are the mouse and keyboard messages. For a complete reference of the messages supported by this application see the SpyHydra documentation.
- *wparam* depends on the *message_name* message.
- *lparam* depends on the *message_name* message.

Sendtext

To send a string (no special characters) to the terminal server, use the sendtext statement. The specified string is sent through the last opened connection:

```
sendtext (<string>, <milliseconds>);
```

where

- *string* is the string to be sent to the terminal server.
- *milliseconds* specifies the time, in milliseconds, for which to suspend execution after each character sent from the *string*.

Sleep

To suspend the execution of the current job for a specified interval use the sleep keyword:

```
sleep (<milliseconds>);
```

where *milliseconds* specifies the time, in milliseconds, for which to suspend execution.

Clipboard

To put something on or get something from the clipboard use the clipboard keyword:

```
clipboard (<operation>, <file_name>);
```

where

- *operation* specifies the clipboard operation (copy or paste).
- *file_name* is the file that contains what to copy or paste to or from the clipboard.

Loop

In a loop statement, the statement list is executed until the value of the counter becomes zero. If the initial value of the counter is zero, then the loop is executed forever:

```
loop (<counter>)
{
  <statement_list>
}
```

where

- *counter* specifies how many times the loop is run.
- *<statement_list>* lists the set of commands that you want to loop.

Check

To verify that something worked OK on the terminal server, use the check statement:

```
check (<check_function>, <function_params>);
```

where

- *check_function* is a null-terminated string that contains the check function name. This function should be exported from the Check.dll library.
- *function_params* is a null-terminated string that contains the check parameters. It is the responsibility of check function to parse this string.

Call

Call statement is used to call a function from a DLL not specified in smclient.ini:

```
call (<DLL_library>, <function_name>, <function_params>);
```

where

- *DLL_library* is a null-terminated string that contains the DLL name.

- *function_name* is a null-terminated string that contains the function name. This function should be exported from the *DLL_library* library.
- *function_params* is a null-terminated string that contains the function parameters. It is the responsibility of the function to parse this string.

Job

Job is like an anonymous function; it contains one or more statements that are executed in sequence. The job keyword is not a statement; therefore, it cannot be used in a loop or another job:

```
job
{
    <statement_list>
}
```

where

- <statement_list> lists the set of commands that you want to loop.

Comments

The characters // start a comment, which terminates at the end of the line on which they occur. The comment characters // have no special meaning within a // comment and are treated just like other characters. For example

```
//this is a comment
```