Microsoft

Windows NT® Server

Server Operating System

Using and Developing Applications Compatibility Scripts with Windows NT® Server 4.0, Terminal Server Edition

White Paper

Abstract

Microsoft® Windows NT® Server 4.0, Terminal Server Edition, is a new technology that gives the Windows NT Server operating system the capability to serve 32-bit Windows® operating system-based applications to terminals and terminal emulators running on personal computer and other computer desktops. The Terminal Server environment is, by definition, a thin client architecture where all application processing occurs centrally on the server. Because Terminal Server terminal emulator clients will be available for many different desktop platforms (Macintosh, UNIX, and others), Terminal Server provides access to 32-bit Windows-based applications from virtually any desktop, and provides a bridging technology for organizations that are transitioning to a pure 32-bit desktop environment.

This paper provides guidelines for using and developing Applications Compatibility Scripts for the Terminal Server environment. This is a highly technical document and a thorough understanding of Windows NT Server 4.0 and Windows NT Server 4.0, Terminal Server Edition will be important for understanding the information in this document. For a more information about Windows NTServer 4.0, Terminal Server Edition see http:\\www.microsoft.com\ntserver\terminalserver

Microsoft Corporation • One Microsoft Way • Redmond, WA 98052-6399 • USA

0698

CONTENTS

INTRODUCTION

Microsoft® Windows NT® Server 4.0, Terminal Server Edition, is a new technology that gives the Windows NT Server operating system the capability to serve 32-bit Windows® operating system-based applications to terminals and terminal emulators running on personal computer and other computer desktops. The Terminal Server environment is, by definition, a thin client, 100 percent server-centric architecture. Because Terminal Server terminal em ulator clients will be available for many different desktop platforms (Macintosh, UNIX, and others), Terminal Server provides access to 32-bit Windows-based applic ations from virtually any desktop. Terminal Server allows you to roll out 32-bit Windows-based applications to a heterogeneous set of desktops while transitioning to a pure 32-bit desktop environment.

Unlike the traditional client/server environment, an application runs only on the server in the Terminal Server environment. The Terminal Server client performs no local processing of applications. The Terminal Server operating system transmits only the application presentation—the Graphical User Interface, or GUI—down to the client. Each user logs on and perceives only his or her *session*, which is transparently managed by the server operating system and is independent from any other client session.

From an application usage perspective, one of the biggest benefits of Terminal Server is that the majority of existing 16- or 32-bit application programs run "as is"—no programming changes are required to run them on a Terminal Server. This does not, however, mean that all existing applic ations run equally well under Terminal Server. As with the addition of any new operating system technology, understanding how to design applications that take advantage of these new capabilities is important. It's also important to understand how this new service can magnify the negative results of bad programming habits, and to point out specific areas to watch for.

Note that following these guidelines does not limit or compromise an appl ication's ability to function in the traditional client/server Windows NT environment—Terminal Server-optimized applications work well in both enviro nments.

Terminal Server Extends the Multi-user Capabilities of Windows NT Server

In addition to providing a means to serve Windows-based applications to te rminals and other thin client devices, Terminal Server extends the current multi-user capacity of Windows NT Server. Of course, Windows NT Server is inherently multi-user-capable in the following ways:

- User profiles stored on the server can be used to allow numerous users to see many desktops and run various applications when they log on.
- A sophisticated security system controls the capabilities and access rights of both local and remote client users.
- Operating system interfaces allow concurrent users to safely access common

files and databases stored on file servers throughout a distributed network.

The Terminal Server technology goes beyond the client/server multi-user services listed above. With Terminal Server, users share hardware and sof tware resources commonly found on a local Windows or Windows NT-based client. These resources include use of a central CPU, memory, and storage, as well as operating system resources such as the registry and other data structures.

This document is divided into several sections.

- The first section provides in-depth information about why application compatibility issues exist and where the main issues are and possible solutions for those issues.
- The second section examines the Applications Compatibility Flags, and how they are used to negate some applications issues.
- The third section shows how to set-up a user logon script and explains the purpose of a user script..
- The fourth section explains , the Application Compatibility Script itself and describes how to fix issues discussed in this paper..
- Appendix A contains a printed version of an example User.cmd file (script) that can be used as a starting point for designing your own compatibility scripts.
- Appendix B contains a printed version of the Microsoft Word97 script with explanations of what each section does. Again, this can be used as a starting point for developing your own scripts.
- Appendix C has a flow chart showing the process that occurs when the userlogon.cmd script executes and what occurs. This is to provide a visual representation of what is occurring.

APPLICATION COMPATIBILITY

Why Do Application Compatibility Issues Exist?

Ideally, applications would install and run on Terminal Server without requiring any special configuration. Unfortunately, this isn't always the case. Many applications, particularly older applications, were not designed to allow operation by multiple users. These applications make assumptions about the operating environment that are not valid in a multi-user operating system.

As a result, these applications must be tuned before they run properly on NTS/TSE. In the future, application compatibility will become less of an issue because applications will designed by the developers to correctly handle multiple users.

Problem Areas, Compatibility Issues and How They are Resolved

There are a few classes of issues that might need resolution and therefore an Application Compatibility Script. The following sections describe for each area what the issues are and how to resolve them.

REGISTRY SETTINGS

Applications keep configuration information in a system database called the registry. The two most important sections of the registry are called HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER.

Information that pertains to the entire system should be stored in HKEY_LOCAL_MACHINE. An application might store the following information in this section:

- The components of an application that were installed.
- The path used to load application components.
- In certain circumstances, the path to a shared database. For example, a Point of Sale application where all users reference the same database could put that path in HKEY_LOCAL_MACHINE.

Applications store user specific information in the HKEY_CURRENT_USER section. This section is part of the user profile. When a user logs on, his profile is loaded into the system's registry and it becomes HKEY_CURRENT_USER. When the user logs off, any changes to HKEY_CURRENT_USER are written back to his user profile. Applications might store the following types of information in HKEY_CURRENT_USER:

- Paths to custom dictionaries, mailboxes, configuration files, and temporary directories. Per-user paths are particularly important for multi-user operation.
- Settings which are per-user preferences. For instance, some users might want to enable background spell checking while others may want to disable it.

Per User Data in HKEY_LOCAL_MACHINE:

An application may put per-user data into HKEY_LOCAL_MACHINE instead of HKEY_CURRENT_USER. For example, an application may store the path to a mailbox in HKEY_LOCAL_MACHINE. This would cause all users to share the same mailbox, which probably isn't the desired result. The application should use HKEY_CURRENT_USER for per-user paths.

Resolution:

The most common problem deals with paths. By mapping each user's home directory to a drive letter, a common path can be specified which still maps to individual files for each user. This technique will be discussed in more detail in the Setting up Users sections on p. 14.

Other types of HKEY_LOCAL_MACHINE issues typically can't be corrected. For example, if the application stores color preferences in HKEY_LOCAL_MACHINE, then users won't be able to specify custom color preferences.

HKEY_CURRENT_USER information setup for single user only:

Resolution:

Terminal Server handles this situation by providing an install mode. You enter install mode by issuing the command “change user /install” at a cmd prompt or by selecting “All users begin with common application settings” from the Add/Remove Programs wizard. Once you have entered install mode, you can install the application. Each registry setting that is written to HKEY_CURRENT_USER will be mirrored into a special location within HKEY_LOCAL_MACHINE. When your installation is complete, the “change user /execute” command takes you out of install mode. If you’re using the Add/Remove Programs wizard, you are automatically taken out of install mode after you click the finish button. Terminal Server propagates the mirrored information in HKEY_LOCAL_MACHINE to each user’s HKEY_CURRENT_USER when the information is needed. The location of the mirrored registry entries are: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\TerminalServer\Install.

Application fails to create HKEY_CURRENT_USER settings on install:

An application may not create HKEY_CURRENT_USER registry settings during installation. Instead, it might wait until the first time a user runs the program. If the person who installs the application does not run it while install mode is still active, the HKEY_CURRENT_USER settings won’t get copied to HKEY_LOCAL_MACHINE and the automatic propagation won’t occur.

The first time each user runs the application, their HKEY_CURRENT_USER will be loaded with the default settings. Depending on the application, this may or may not be appropriate. If the default settings aren’t sufficient, then they would need to be corrected for each user individually.

Resolution:

Before leaving install mode, the administrator can run the application. If the administrator has already left install mode but hasn’t run the application, just re-enter install mode with “change user /install” and run the application. If the administrator has already run the application while in execute mode, create another administrator account. Log on with this new account, change to install mode, and run the application. In all cases, use “change user /execute” to leave install mode after running the application.

FILES

The files and directories an application uses can also cause problems. Primarily, the issue is with applications that use public locations for per-user files. Here are the issues:

Incorrect Pathname for per-user data:

An application may use the same pathname for a file that contains per-user data. In the best case, if one user updates this file, it affects all other users. More severe problems include preventing multiple users from running the application simultaneously and corruption of the file, which could cause the application to fail for all users.

It's important to note that most files don't contain per-user data and are safe to share. Some files could be classified either way, depending on the system administrator's preference. For instance, document template files could be made read-only and safely shared. Or, users could have their own set of template files that they can update as desired.

Resolution:

This is perhaps the most difficult configuration issue to resolve. The solution is to update the pathnames of per-user files so these files reside in each user's home directory. There are three issues. First, you may not immediately know there is a problem. Second, it is difficult to detect which files need to be made per-user. Third, it isn't always obvious where to change the pathname.

When you find that a problem exists, you often become aware of the conflicting file too. If you have difficulty finding the shared file, you can turn file auditing on. It is most helpful to audit successful writes to files.

- After finding the name of the offending file, you must determine where that path is stored. . Applications can store the pathname of per-user files in a configuration file or it can be hard coded within the application. The pathname can also be stored in HKEY_LOCAL_MACHINE and HKEY_CURRENT_USERS.

The first step is to use **regedt32** to search HKEY_LOCAL_MACHINE and HKEY_CURRENT_USERS for the file in question. Typically, applications store their values in the "Software" sub-tree. If you don't find anything, you might also want to check the product documentation. Sometimes the default path is hard-coded in the program, but registry settings can be added to override the default.

The second step is to check the Options or Preferences dialog in the application. It may provide a way for you to change the path. If so, you should then determine if the settings are stored in a configuration file. File auditing can tell you if any files get updated after you make the change in the program. The idea is to determine the configuration file so it can be propagated to other users. The technique for this can vary since the configuration file may contain other information that is specific to each user.

A third method is to contact the application vendor. They may be able to identify a method of updating the pathname in question.

Once you have found the location of the path that needs to be changed, what should you change it to? If the path is stored in HKEY_LOCAL_MACHINE, a special technique is used.

> **When a user logs on, a script (UsrLogon.cmd) will be run that maps a drive letter to their home directory using the "subst" command. For example, Z:\ will map to \\server\users\fredf for Fred Flounder and to \\server\users\barneyb for Barney Baitfish.**

With this technique, you can specify what appears to be a shared path, such as Z:\mail\mailbox.dat. Yet, the drive mapping causes each user to get a unique copy of the file in their home directory.

Some per-user files are automatically generated by the application. A mailbox would be an example of this type of file. Other per-user files, such as document templates, must be provided for the user. That is, you must copy the standard files into the user's home directory. If you specified the path as a subdirectory of the user's home directory, you may need to create the subdirectory too.

Per-user tasks like this can be performed in a script that is run during logon. The script can check the user's home directory to determine if the files or directories in question already exist. If not, it can copy them from a "master" copy. Using a script allows you update current users with a new application as well as automatically set up new users you may add later.

Files Missing when normal user logs in:

When a user runs the application, it can't find certain files that are located in or below %systemroot%. However, if an administrator runs the application, it finds the files.

Resolution:

Applications that use the GetWindowsDirectory() function to determine %systemroot% benefit from an automatic remapping. For users, the function returns the "windows" subdirectory of their home directory. Administrators, on the other hand, get the actual %systemroot% directory.

Normally, this is useful. It automatically converts a shared file into a per-user file. However, if the shared file is expected to exist, then it won't be found under the user's home directory. This can be corrected by using a logon script to copy the file if it doesn't already exist.

As an alternative, an application compatibility flag exists that can disable this automatic remapping. This is appropriate if the files under %systemroot% are only being used for read access. For more information about these flags, refer to the Application Compatibility Flags section of this Paper (p.12), …

Inaccessible files due to file security:

Some applications store files in public directories such as %systemroot%. By default, some public directories have read-only permission for non-administrative users. This prevents accidental modification of common files. However, if an application attempts to write a file into one of these shared directories, then it fails.

This type of problem frequently manifests as an application that works fine for an administrative user, but fails for non-administrative users. To help track down this type of problem, you can use file auditing to log writes that fail.

Resolution:

There are three ways to correct this situation. First, note that Terminal Server automatically maps %systemroot% to the user's home directory as mentioned above.

Second, you can treat this issue just like the previous issue. That is, you find the pathname that needs to be updated and you change it to reference data in the user's home directory.

Finally, you can add extra permission to a public directory or to individual files within a public directory. Be very careful if you add permission to the entire directory, because other programs may then be able to alter the behavior for the entire system. The decision to make public directories read-only was based on the theory that it is better for an application to fail than for the entire system to fail.

Applications accessing same cached files:

Some applications use temporary or cache directories to store information that relates to the current user. The application may not create a separate directory for each user. This causes the application to store temporary data for multiple users in a single location. If the application isn't intelligent about handling this data, one user can delete data which is still needed by another user.

Resolution:

The best solution is to update the path as discussed in issue 1. Keep in mind that the application may not automatically create the directory on an as-needed basis. Therefore, your logon script should check the user's home directory, and create the sub-directory if it doesn't already exist.

Icons for applications do not show up in the start menu:

During installation, some applications add their Start menu shortcuts to the menu of the user performing the install instead of to the All Users Start menu. Then, other users don't have the program on their start menu.

Resolution:

If you use “change user /install” or “All users begin with common application settings” from the Add/Remove Programs wizard, Terminal Server addresses this issue for you. When you enter install mode, the state of the Current User start menu is captured.

When you finish installing the application and return to execute mode, the Current User start menu is compared to the saved state. Any additions or changes are moved from the Current User start menu to the All Users start menu. Additionally, each shortcut that is moved has read access added for group Everyone.

OBJECTS

Microsoft Windows applications can create objects (such as Events, Devices, Semaphores, and Sections) which are used to communicate with other applications. Each object has a name which is globally visible on the system. This can create a problem similar to shared files. That is, two instances of an application might both reference the same object name. They want separate objects but instead end up sharing a single object. This can cause erratic or incorrect behavior.

To solve this problem, Terminal Server appends a colon and the logon ID to each object that gets created. Thus, if an application runs under logon ID 4, it creates object foo as "foo:4". Under logon ID 7, the same application would create "foo:7". This prevents object collision between several running copies of an application. Objects that are renamed are known as user global objects.

The console is treated differently. If an application that is running on the console creates an object, nothing gets appended to the name. Using the above example, object foo would actually be named "foo". Objects that do not get renamed are known as system global objects. In addition to programs explicitly run on the console, service programs also run in the context of the console logon.

Maintaining the name of the System Global objects:

Certain classes of applications require the system global objects. For example, Enterprise backup and defragmentation utilities often run a service on each system in the network. An administration program can then be run on one machine in the network. The administration program communicates with the service running on each machine.

The service will create system global objects. The administrator program, on the other hand, most likely runs on a client instead of the console. When it creates an object with the same name, the object gets renamed. As a result, the service and the administrator program actually use different objects, even though they intended to share a single object.

If an application works correctly when run from the console and it fails when run from a client logon, this problem may be present.

Resolution:

Keep the administrator program's objects from being renamed. By default, all objects are user global. There are several methods of telling Terminal Server that an object should be system global.

The easiest method involves registering the DLL or executable that creates the object. By default, every object created by a DLL or EXE is user global. That default can be changed with the command "register filename /system" which marks the file as one which creates system global objects. After this change, any object created by this DLL/EXE will not be renamed.

Finally, if the application is being developed in-house, the best method is for the software developer to append “\\SYSTEM” to system global object names when they is created in the application. This informs Terminal Server that the object is system global and should not be renamed.

COMPATIBILITY FLAGS

These values are set under HKEY_LOCAL_MACHINE\SOFTWARE \Microsoft\Windows NT\CurrentVersion\Terminal Server\Compatibility\ Applications\appname, where appname is the name of the application's executable file. For example, if the executable file name for an application is WINWORD.EXE, the key would be WINWORD.

DOS application	0x00000001
OS/2 application	0x00000002
16-bit application for Windows	0x00000004
32-bit application for Windows	0x00000008
16 or 32-bit application for Windows	0x0000000c

As discussed above, these values set the version to which you want the registry settings to apply.

Return username instead of computername **0x00000010**

Some apps use the computername as a unique identifier. In Terminal Server every user will have the same computername. So, this setting will tell the system to return the username to an application rather than the computername.

Return Terminal Server build number **0x00000020**

Normally, the Windows NT build number is returned. Some applications will be aware of Terminal Server and will need to know which version of Terminal Server is running. An example is HOTFIX, which is set to a value of 0x28 (Return TS build number plus 32-bit application). Terminal Server will have hotfixes separate from Windows NT.

Disable Registry mapping for this application **0x00000100**

Normally, registry settings are captured and cached during installation and then written to each user's registry when the user runs the application. This is not always appropriate and applications can simply get global settings from the cache area without the settings being written to the user's registry. An example is REGEDT32, which has a flag value of 0x108 (Disable registry mapping and 32-bit application).

Instruct application to create System Global Objects **0x00000200**

Normally, named objects are decorated with the Session ID. Setting this value tells

Multi-User Object Manager not to decorate objects created by this application. An example is SNASRV, the executable for SNA Server. All its objects need to be registered as system globals (non-decorated) so the flag value is set to 0x200.

Return systemroot, not user's WINDOWS directory 0x00000400

This is the most commonly used application flag aside from the version flags. Normally, in Execute mode applications are told that the user's Windows directory in the user's home directory is the systemroot. This will cause some applications to fail because they need to find something in the real systemroot. An example is CINMANIA, the executable for Cinemania.

Limit the reported physical memory (GlobalMemoryStatus) 0x00000800

Some applications fail if they are told that the default 32 megabytes (MB) of physical memory is available. Setting a value of 0x800 causes the system to report a smaller amount of available physical memory. An example is MSACCESS, which has a flag value of 0x80c (limit reported memory and 32-bit app).

Log object creation to file 0x00001000

When troubleshooting application problems concerning objects, it may be useful to create a log file of created named objects. Creation of semaphores, mutexes, events and sections will be recorded. You must also set a system environment variable called CITRIX_COMPAT_LOGPATH to a valid directory, and the log file will be written there. This value is not set for any application by default. A further example is located in the APPENDIX.

Don't put app to sleep on unsuccessful keyboard polling (WIN16 only) 0x20000000

Some 16-bit apps (and many DOS apps) rely on keyboard polling to discover keyboard activity. This continual keyboard check can greatly impact CPU utilization. Normally, the system limits this polling. However, some apps will fail if they are limited. None of the default applications use this setting.

HOW TO SET UP USERS UTILIZING LOGON SCRIPTS

For optimum application compatibility, when users log in they should run a script that maps their home directory to a drive letter. Additionally, the script should perform any other application specific tasks that are necessary, determined with the information above about users needed interaction with the application.

There are several methods of running a script when a user logs on. First, the script can be put into the user's Startup folder. This allows a unique script for each user, but requires a bit more effort on the part of the system administrator.

Second, the script can be placed in the All Users Startup folder. With this method, a single script is used for the entire system. It is the easiest method, but it somewhat less flexible.

Finally, the administrator can enter a Logon script name into each user's profile with User Manager for Domains. With this method, you can enter a different script name for each user. For example, you may have a user.cmd script for normal users and admin.cmd for administrators. Logon scripts need to be put into %systemroot%\system32\Repl\Import\Scripts.

Note: User Manager for Domains also lets you map a user's home directory to a drive letter for network home directories (i.e,, Connect Z: as \\server\users\username). Don't be misled. The drive letter is only mapped to the server name and share point. The drive letter does not reference directories beyond the share point.

At a minimum, the script must do the following:

- Set RootDrive=Z:
- Subst %RootDrive% /d
- Subst %RootDrive% %HomeDrive%%HomePath%

Additionally, the script may invoke application configuration scripts by appending "Call scriptname" to the minimal script listed above. Terminal Server includes application scripts for many popular applications. You may need custom application scripts for other applications you run. Application configuration scripts will be described in more detail in the following section.

NOTE: A sample logon script is included in %systemroot%\compat\user.cmd. Also, check Appendix A for a printed version of this script and discussion of what each point does.

APPLICATION CONFIGURATION SCRIPTS

Application tuning tasks must occur at two different times: when the application is installed and when a user logs into the system.

When the application is installed, the administrator may need to do several things:

- Update any paths which might cause problems.
- Register DLLs and Executables as System Global.
- Change Compatibility Flags.

Many of these tasks can be automated using installation scripts. As mentioned, Terminal Server already includes scripts for various products. You may need to update the scripts before running them, because they make assumptions about your environment. For example, Microsoft Office 97 assumes you installed the application into its default directory \Program Files\Microsoft Office. If your installation uses a different path, you will need to change the scripts. Note that the application compatibility user logon scripts are matched to the installation scripts. If you change information in the installation script, you will often need to make a corresponding change in the user logon script. Any changes that may be needed will be clearly indicated at the top of the script.

When running an installation script, you should be in install mode. If the system isn't currently in install mode (you can use "change user /query" to determine this), use the "change user /install" command to enter install mode. Run the appropriate script. When the script completes, use "change user /execute" to return to execute mode.

The install scripts which are included with Terminal Server may be found in the %systemroot%\Application Compatibility Scripts\install directory.

Each time a user logs on, in addition to mapping the user's home directory to a drive letter, the logon script may need to perform application specific tasks. Examples of these tasks include:

- Creating sub-directories in the user's home directory.
- Copying files from a "master" location to the user's home directory.
- Setting permissions on particular files.

The user logon scripts which are included with Terminal Server may be found in the %systemroot%\Application Compatibility Scripts\logon directory.

TOOLS FOR ADDITIONAL AUTOMATION FOR SCRIPTS IN TERMINAL SERVER

The Application Scripts use one final component to assist in optimizing previously written applications for operation in a Terminal Server environment. That component is a series of small programs/executables that are called from within the application compatibility script to retrieve certain values from the registry, change permissions on a set of files.

Some of these tools are designed specifically for Terminal Server, while others were originally part of the resource kit and you may already be familiar with them. Regardless, a full explanation of each .exe with syntax descriptions is provided in the following section. These files are all located in the %systemroot%/applications compatibility scripts directory

You can find good examples of how these programs are used in appendix b, which also includes all of the other application compatibility scripts that Microsoft provides when the Terminal Server product is installed.

Program Tools

1. ACREGL.exe

What is the purpose of this utility?
This tool looks up the desired key and value in the registry. It outputs the result as a SET statement to the indicated filename. Options allow some massaging of the value before it is written to the file.

Syntax: Acregl Filename EnvVarName Key Value Options

NOTE: Key must start with HKLM or HKCU. Often the key will be in quotes because it will contain embedded spaces.

Detailed information on command line options:

Value may be blank ("") to look up the default value for a key. The acregl tool only supports values that return a string type. REG_SZ is ideal. REG_MULTI_SZ would only contain the first item in the list. REG_EXPAND_SZ should work but it won't explicitly get expanded.

The options parameter can be blank ("") or it can be one of the two following choices:

> "**STRIPCHARxn**" will search from right to left. It strips the rightmost n instances of the character x. The count may be omitted

for a single character, but it is still recommended. Otherwise you need to be careful; a backslash will be treated as an escape character for the quote which follows.

Example:

"STRIPCHAR\3" changes C:\WINNT40\Profiles\All Users\Start Menu to C:\WINNT40

"**STRIPPATH**" removes the path from the value.

Example:

"STRIPPATH" changes C:\WINNT40\Profiles\All Users\Start Menu to Start Menu

Full Example:

```
ACREGL MyKey.Cmd INSTLOC
"HKLM\Software\Microsoft\Office\8.0\Common\InstallRoot" "" ""
```

(This command creates the file MyKey.Cmd with the contents)

```
SET INSTLOC=C:\Program Files\Microsoft Office
```

An application compatibility script can then use "CALL MyKey.Cmd" to retrieve the value of the registry key.

ACREGL returns an errorlevel of 1 for failure or 0 for success.

2. ACSR.exe

What is the purpose of this utility?
This command performs simple text replacement. It reads from the Input file and writes to the Output file. Each occurrence of the search string is changed to the replace string. Any of the parameters can contain environment variables if desired.

Syntax: Acsr Search Replace InputFile OutputFile

Example:

ACSR "#ROOTDRIVE#" "%RootDrive%" Template\Office97.key Office97.key

This example changes the text #ROOTDRIVE# to the value of the RootDrive environment variable. It reads the Office97.key from the template subdirectory and writes to an Office97.key file in the current directory.

ACSR returns an errorlevel of 1 for failure or 0 for success.

3. ACINIUPD.exe

What is the purpose of this utility?
The aciniupd tool is used to update .ini files. It has several modes and options.

Syntax: Aciniupd [/e | /k] [/u] [/v] ini_file section key new_value

Detailed information on command line options:

/e option is used to update the value for a given key and section. For example, the following will update the value of the USER-DOT-PATH key in the Microsoft Word section of WinWord6.ini:

```
Aciniupd /e "Winword6.ini" "Microsoft Word"
USER-DOT-PATH "%RootDrive%\OFFICE43\WINWORD"
```

/k option updates a key name with the new key name in the specified section. In the following example, the key WZTABLE.MDA in the Libraries section of MsAcc20.Ini is changed. Note, that this option changes the left side of the KEY=VALUE statement while the /e option changes the right side.

```
Aciniupd /k "Msacc20.ini" Libraries "WZTABLE.MDA"
"%RootDrive%\OFFICE43\ACCESS\WZTABLE.MDA"
```

/u option updates the .INI file in the user's windows directory instead of the system directory. This is accomplished by changing the system mode to execute if currently in install mode. The mode returns to its original state when the tool completes.

/v option is for verbose mode. In verbose mode, error and status messages are displayed as appropriate. By default, the tool operates silently.

ACINIUPD returns an errorlevel of 1 for failure or 0 for success.

4. CACLS.exe

What is the purpose of this utility?
Cacls.exe is used to display or modify access control lists (ACLs) of files.

Syntax:
CACLS filename [/T][/E][/C][/G user:perm][/R user [...]][/P user:perm [...]] [/D user [...

Example:

Cacls %systemroot%\system32\MsForms.Twd /E /P EveryOne:F

This example replaces the permissions on the msforms.twd file to full control for the group Everyone.

Detailed information on command line options:

Filename option displays ACLs for the specific file.

/T option changes ACLs of specified files in the current directory and all subdirectories.

/E option edits ACL instead of replacing it.

/C option continues on access denied errors.

/G user:*perm* option grants specified user access rights.
Perm can be: R Read
C Change (write)
F Full control

/R user option revokes specified user's access rights (only valid with /E).

/P user:*perm* option replaces specified user's access rights.
Perm can be: N None
R Read
C Change (write)
F Full control

/D user option denies specified user access.

Wildcards can be used to specify more that one file in a command. You can specify more than one user in a command.

5. FINDSTR.exe

What is the purpose of this utility?
Searches for strings in files using literal text or regular expressions. For more detail, see findstr in online help.
Syntax: findstr [/b] [/e] [/l] [/c:string] [/r] [/s] [/i] [/x] [/v] [/n] [/m] [/o] [/g:file] [/f:file] strings files

Detailed information on command line options:

/b Matches pattern if at the beginning of a line.
/e Matches pattern if at the end of a line.
/l Uses search strings literally.
/c Uses specified text as a literal search string.

/r Uses search strings as regular expressions. This switch is not required; findstr will interpret all metacharacters as regular expressions unless the /l switch is used.
/s Searches for matching files in the current directory and all subdirectories.

/l Specifies that the search is not to be case-sensitive.
/x Prints lines that match exactly.
/v Prints only lines that do not contain a match.
/n Prints the line number before each line that matches.
/m Prints only the filename if a file contains a match.
/o Prints seek offset before each matching line.
/g Gets search strings from the specified file.
/f Reads file list from the specified file.
Strings Text to be searched for.
Files Files to be searched.

NOTE: Use spaces to separate multiple search strings unless the argument is prefixed with /e.

Examples:

- findstr "hello there" x.y
  searches for "hello" or "there" in file x.y.

- findstr /c:"hello there" x.y'
  searches for "hello there" in file x.y.

6. REGINI.exe

What is the purpose of this utility
Regini.exe reads in .key files and updates the registry based on the values in the .key files.

Syntax: REGINI [-h hivefile hiveroot | -w Win95 Directory | -m \\machinename]
[-i n] [-o outputWidth]
[-b] textFiles...

Example:
Regini msword97.key

Portion of the msword97.key:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\8.0
    Common
    FileNew
    LocalTemplates
    = DELETE
    SharedTemplates
    = DELETE
```

The first part of the msword97 key deletes the default values inserted into the registry by the installation.

```
HKEY_CURRENT_USER\Software\Microsoft\Office\8.0
        Common
        FileNew
        LocalTemplates
        = "W:\Office97\Templates"
        SharedTemplates
        = "C:\Program Files\MS Office\Templates"
```

The second part of the msword97 key creates new values to be inserted into the registry, showing specific user information.

Detailed information on command line options:

-h specifies a specify local hive to manipulate.
-w specifies the paths to a Windows 95 system.dat and user.dat files
-m specifies a remote Windows NT-based machine whose registry is to be manipulated
-i n specifies the display indentation multiple. Default is 4
-o outputWidth specifies how wide the output is to be. By default the outputWidth is set to the width of the console window if standard output has not been redirected to a file. In the latter case, an outputWidth of 240 is used.

-b specifies that REGINI should be backward compatible with older versions of REGINI that did not strictly enforce line continuations and quoted strings Specifically, REG_BINARY, REG_RESOURCE_LIST and REG_RESOURCE_REQUIREMENTS_LIST data types did not need line continuations after the first number that gave the size of the data. It just kept looking on following lines until it found enough data values to equal the data length or hit invalid input. Quoted strings were only allowed in REG_MULTI_SZ. They could not be specified around key or value names, or around values for REG_SZ or REG_EXPAND_SZ Finally, the old REGINI did not support the semicolon as an end of line comment character.

textFiles is one or more ANSI or Unicode text files with registry data. The easiest way to understand the format of the input textFile is to use the REGDMP command with no arguments to dump the current contents of your NT Registry to standard out. Redirect standard out to a file and this file is acceptable as input to REGINI.

Some general rules are:

- Semicolon character is an end-of-line comment character, provided it is the first non-blank character on a line

- Backslash character is a line continuation character. All characters from the backslash up to but not including the first non-blank character of the next line are ignored. If there is more than one space before the line continuation character, it is replaced by a single space.

- Indentation is used to indicate the tree structure of registry keys The REGDMP program uses indentation in multiples of 4. You may use hard tab characters for indentation, but embedded hard tab characters are converted to a single space regardless of their position

- For key names, leading and trailing space characters are ignored and not included in the key name, unless the key name is surrounded by quotes. Imbedded spaces are part of a key name.

- Key names can be followed by an Access Control List (ACL) which is a series of decimal numbers, separated by spaces, bracketed by a square brackets (e.g. [8 4 17]). The valid numbers and their meanings are:

  1 - Administrators Full Access
  2 - Administrators Read Access
  3 - Administrators Read and Write Access
  4 - Administrators Read, Write and Delete Access
  5 - Creator Full Access
  6 - Creator Read and Write Access
  7 - World Full Access
  8 - World Read Access
  9 - World Read and Write Access
  10 - World Read, Write and Delete Access
  11 - Power Users Full Access
  12 - Power Users Read and Write Access
  13 - Power Users Read, Write and Delete Access
  14 - System Operators Full Access
  15 - System Operators Read and Write Access
  16 - System Operators Read, Write and Delete Access
  17 - System Full Access
  18 - System Read and Write Access
  19 - System Read Access
  20 - Administrators Read, Write and Execute Access
  21 - Interactive User Full Access
  22 - Interactive User Read and Write Access
  23 - Interactive User Read, Write and Delete Access

- If there is an equal sign on the same line as a left square bracket then the equal sign takes precedence, and the line is treated as a registry value. If the text between the square brackets is the string DELETE with no spaces, then REGINI will delete the key and any values and keys under it.

- For registry values, the syntax is:

  value Name = type data

- Leading spaces, spaces on either side of the equal sign and spaces between the type keyword and data are ignored, unless the value name is surrounded by quotes.

- The value name may be left off or be specified by an at-sign character which is the same thing, namely the empty value name. So the following two lines are identical:

  = type data
  @ = type data

- This syntax means that you can't create a value with leading or trailing spaces, an equal sign or an at-sign in the value name, unless you put the name in quotes.

- Valid value types and format of data that follows are:

  REG_SZ text
  REG_EXPAND_SZ text
  REG_MULTI_SZ "string1" "string2" ...
  REG_DATE mm/dd/yyyy HH:MM DayOfWeek
  REG_DWORD numberDWORD
  REG_BINARY numberOfBytes numberDWORD(s)...
  REG_NONE (same format as REG_BINARY)
  REG_RESOURCE_LIST (same format as REG_BINARY)
  REG_RESOURCE_REQUIREMENTS (same format as REG_BINARY)
  REG_RESOURCE_REQUIREMENTS_LIST (same format as REG_BINARY)
  REG_FULL_RESOURCE_DESCRIPTOR (same format as REG_BINARY)
  REG_MULTISZ_FILE fileName
  REG_BINARYFILE fileName

- If no value type is specified, default is REG_SZ

- For REG_SZ and REG_EXPAND_SZ, if you want leading or trailing spaces in the value text, surround the text with quotes. The value text can contain any number of imbedded quotes, and REGINI will ignore them, as it only looks at the first and last character for quote characters.

- For REG_BINARY, the value data consists of one or more numbers The default base for numbers is decimal. Hexadecimal may be specified by using 0x prefix. The first number is the number of data bytes, excluding the first number. After the first number must come enough numbers to fill the value. Each number represents one DWORD or 4 bytes. So if the first number was 0x5 you would need two more numbers after that to fill the 5 bytes. The high order 3 bytes of the second DWORD would be ignored.

CONCLUSION

The amount of work it will take to bring an application in line with the Terminal Server guidelines varies from one application to another. If an application was designed to operate in a multi-user network environment, chances are it will take very little tweaking to turn it into an application that is optimized for Windows NT Server 4.0, Terminal Server Edition.. This is what Application Compatibility scripts are designed for.

This document has presented a variety of guidelines and suggestions for using and developing applications compatibility scripts for the Terminal Server environment. All of these guidelines and suggestions should be taken into consideration when you develop 32-bit applications; however, the three main points you should remember about the Terminal Server environment are:

- **Registry Settings** - Applications keep configuration information in a system database called the registry. The two most important sections of the registry are called HKEY_LOCAL_MACHINE and HKEY_CURRENT_USER. Being aware of the differences between these two locations and the impact they have on the application is critical to optimal performance
- **Files** - The files and directories an application uses can also cause problems. Primarily, the issue is with applications that use public locations for per-user files. It is important to be able to find the files that are inaccessible for whatever reason and make them available to the application/user.
- **Object** - Windows applications can create objects (such as Events, Devices, Semaphores, and Sections) which are used to communicate with other applications. Each object has a name which is globally visible on the system. This can create a problem similar to shared files. Take this into account

By taking these points into consideration, System Administrators, developers, etc. can make the most of their a pplications in the Terminal Server enviro nment.

FOR MORE INFORMATION

Please refer to http://www.microsoft.com/win32dev under Guidelines for add itional information on porting 16-bit applications to the 32-bit Windows environment. For the latest information on Windows NT Server and Terminal Server Edition, version 4.0, check out our World Wide Web site at http://www.microsoft.com/ntserver/terminalserver.

APPENDIX A: USRLOGON.CMD

USRLOGON.CMD

```
@Echo Off

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 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

Net Use %RootDrive% /D >NUL: 2>&1
Subst %RootDrive% /d >NUL: 2>&1
Subst %RootDrive% %HomeDrive%%HomePath%

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

ROOTDRV.CMD

If Exist "%SystemRoot%\Application Compatibility Scripts\RootDrv2.Cmd" Call "%SystemRoot%\Application Compatibility Scripts\RootDrv2.Cmd"

USRLOGN1.CMD AND USRLOGN2.CMD

These two scripts do not exist until at least one compatibility script is run. These files contain only CALL statements for various user logon scripts.

APPENDIX B: MSWORD97.CMD

```
@Echo Off

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
######################################################################
..\ACRegL %Temp%\O97.Cmd O97INST "HKLM\Software\Microsoft\Office\8.0"
"BinDirPath" "STRIPCHAR\1"
If Not ErrorLevel 1 Goto Cont0
Echo.
Echo Unable to retrieve Word 97 installation location from the registry.
Echo Verify that Word 97 has already been installed and run this script
Echo again.
Echo.
Pause
Goto Done

:Cont0
Call %Temp%\O97.Cmd
Del %Temp%\O97.Cmd >Nul: 2>&1

Rem
######################################################################
Rem
Rem Allow read access for everybody on a system DLL that is
Rem updated by Office 97.
Rem
```

```
If Exist %SystemRoot%\System32\OleAut32.Dll cacls
%SystemRoot%\System32\OleAut32.Dll /E /T /G EveryOne:R >NUL: 2>&1

Rem
######################################################################
Rem
Rem Remove Find Fast from the Startup menu for all users.  Find Fast
Rem is resource intensive and will impact system performance.
Rem

If Exist "%systemroot%\Profiles\All Users\Start Menu\Programs\Startup\Microsoft
Find Fast.lnk" Del "%systemroot%\Profiles\All Users\Start
Menu\Programs\Startup\Microsoft Find Fast.lnk" >Nul: 2>&1

Rem
######################################################################
Rem
Rem Change Registry Keys to make paths point to user specific directories.
Rem

Rem If not currently in Install Mode, change to Install Mode.
Set __OrigMode=Install
ChgUsr /query > Nul:
if Not ErrorLevel 101 Goto Begin
Set __OrigMode=Exec
Change User /Install > Nul:
:Begin

..\acsr "#ROOTDRIVE#" "%RootDrive%" Template\MsWord97.Key MsWord97.Tmp
..\acsr "#INSTDIR#" "%O97INST%" MsWord97.Tmp MsWord97.Key
Del MsWord97.Tmp >Nul: 2>&1
```

```
regini MsWord97.key > Nul:

Rem If original mode was execute, change back to Execute Mode.
If "%__OrigMode%" == "Exec" Change User /Execute > Nul:
Set __OrigMode=

Rem
######################################################################
Rem
Rem Update Wrd97Usr.Cmd to reflect actual installation directory and
Rem add it to the UsrLogn2.Cmd script
Rem

..\acsr "#INSTDIR#" "%O97INST%" ..\Logon\Template\Wrd97Usr.Cmd
..\Logon\Wrd97Usr.Cmd

FindStr /I Wrd97Usr %SystemRoot%\System32\UsrLogn2.Cmd >Nul: 2>&1
If Not ErrorLevel 1 Goto Skip1
Echo Call Wrd97Usr.Cmd >> %SystemRoot%\System32\UsrLogn2.Cmd
:Skip1

Echo.
Echo   To insure proper operation of MS Word 97, users who are
Echo   currently logged on must log off and log on again before
Echo   running any application.
Echo.
Echo Microsoft Word 97 Multi-user Application Tuning Complete
Pause

:Done
```

APPENDIX C: FLOW CHART OF USRLOGON.CMD