Table of Contents

INTRODUCTION TO WINSOCK RSHD/NT

Winsock RSHD/NT is a Remote Shell Daemon service for Windows NT/2000/XP/2003 and higher. The job of a Remote Shell Daemon is to service the standard ***rsh*** and ***rcp*** commands. The *rsh* command allows remote program execution of non-interactive programs over a TCP/IP connection and the *rcp* command provides files transfers over TCP/IP, similar to the COPY command. It runs under Microsoft Windows NT 4.0, Windows 2000, Windows XP, and Windows 2003 (or higher).

Winsock RSHD/NT is similar to the Unix *rshd*, but has some special features for the Windows environment, such as:

- User Equivalency translation table, allowing you to map Windows users to remote users
- Sending keystrokes to GUI programs started with **rsh**
- Optional automatic conversion of text files between Unix and Windows format with **rcp** copies

Winsock RSHD/NT can service clients running other operating systems such as Unix or from other Windows systems, as long a standard *rsh* and/or *rcp* command is available.

Winsock RSHD/NT does not include the *rsh* and *rcp* commands for Windows clients. Windows NT, 2000, XP, and 2003 include the *rsh* and *rcp* commands, as do most versions of Unix. Windows 95/98/ME does not have its own *rsh* and *rcp* commands. If you need an *rsh* or *rcp* client for Windows 95/98/ME, use can use Denicomp Systems' Winsock RCP/RSH/REXEC package (a separate product).

REQUIREMENTS

Winsock RSHD/NT requires an Intel x86-based system running Windows NT 4.0, Windows 2000, Windows XP, Windows 2003 (or future compatible operating system) and a network, with TCP/IP configured. Winsock RSHD/NT will run under Windows NT 3.51, but some features will not be available.

Winsock RSHD/NT does not run under Windows 95/98/ME. There is a version specifically written for Windows 95/98/ME called Winsock RSHD/95. See www.denicomp.com for more details.

SECURITY – A WARNING

The purpose of a Remote Shell Daemon is to service the standard *rsh* and *rcp* commands. These commands were designed to be used on a network where users and systems were generally ***trusted*** and *convenience* was more desirable than *security*.

Since *rsh* and *rcp* do not require users to supply passwords before being granted access, there is the risk of unauthorized users being granted access to your system when running Winsock RSHD/NT (or any Remote Shell Daemon, for that matter). This risk is greatly increased if the system running Winsock RSHD/NT is publicly available over the Internet. Winsock RSHD/NT provides mechanisms for you to determine which clients and users are granted access to your system. Firewalls can also help to reduce this risk. But the risk of unauthorized access still remains even when these are used because of the lack of password validation.

By installing Winsock RSHD/NT, ***you are taking this risk of unauthorized access upon yourself***. By installing Winsock RSHD/NT, you are agreeing that you will not hold Denicomp Systems responsible for any damage caused by unauthorized access to your system through the standard *rsh* and *rcp* commands. Denicomp Systems did not design the *rsh* and *rcp* protocols; RSHD/NT simply implements the standard originally designed for Unix systems. Please review the license agreement included with Winsock RSHD/NT and do not continue with the installation or with using it if you do not accept its terms.

WINSOCK RSHD/NT INSTALLATION

As will be explained later, you can start Winsock RSHD/NT as a Windows ***service*** or as a Windows ***application***. A service is a special program that automatically starts in the background when Windows boots. It runs even when no one is logged on to the Windows console (where the screen and keyboard are attached). An application is simply a standard Windows program that you execute from the Start menu or from the Command Prompt. Applications only run when someone is logged in to the Windows console; when you log out, Windows will stop the application.

Normally, you will want to install Winsock RSHD/NT as a service. The only reasons you would not want to install RSHD/NT as a service would be if you either only wanted to have RSHD/NT running occasionally when you manually start it or you only want to experiment with it and you do not have permission to install it as a service.

If you are going to install RSHD/NT as a service, you **must** be logged in as a user who has permission to create services. Normally, you will want to do this while logged in as **Administrator**.

Installing from a CD or Diskette

Click on the ***Start*** button, then choose ***Run***, then type the drive letter of your CD or diskette drive followed by "SETUP", such as A:SETUP and press the Enter key. This can also be done from a Command Prompt if you prefer. Or you can use Windows Explorer or My Computer to browse the CD or diskette and double-click on SETUP to begin the installation.

Installing from a Licensed Download File

If you downloaded a ".exe" file from the Licensed Users section of www.denicomp.com (because you selected electronic delivery when purchasing or you are downloading an updated version), execute that file to install the software. If this is the first time you are installing the licensed version from a download, you will be asked for your installation password that you received when you purchased the software.

If you purchased through Denicomp's online ordering system, your password will be at the end of the e-mail you receive just after you placed the order.

If you are updating from a version you installed previously from a CD or diskette, you will need to enter your installation password, which can be found on the license sheet included with the package you received.

Once you enter the installation password, you will not have to enter it again for the installation of future updates downloaded from Denicomp's web site. The installer records your password and will automatically re-use it for future updates.

Installing on a System with Windows Terminal Services

If you are installing on a system with Windows Terminal Services installed, you must use run SETUP or the downloaded ".exe" file using the Add/Remove Programs icon in the Control Panel. Although the operation of

RSHD/NT is not affected by Terminal Services, Windows requires that setup programs be run through Add/Remove Programs when Terminal Services is present.

Automating Installations

If you have purchased multiple licenses, a Site License, or Corporate License, you can automate installations on multiple systems using these techniques.

If you are installing using a licensed ".exe" file you downloaded from Denicomp's web site, you can specify the installation password on the command line using a **/p:** option. Specify the password after the **/p:** (no spaces between the colon (:) and the password). For example:

```
rshdnt_reg.exe /p:install_pwd
```

where "install_pwd" is your installation password. This will install the registered version of Winsock RSHD/NT on the system without asking for the password. You could put this command in a batch file and use the batch file to do the install so you do not need to type the command line or password.

When you use the /p: option, the installation will not ask for the password, but you will need to answer all other installation questions. If you wish to do a totally silent install, add the **/s** option before the **/p:** option. For example:

```
rshdnt_reg.exe /s /p:install_pwd
```

The **/s** *must* appear before the **/p:**. With **/s**, you will see nothing on the screen **-** the software will be installed using all default options. This command installs into the directory \WRSHDNT on the system drive, as a service, starting automatically at boot time, with the "no initial security" option.

If you do not like the default options, or you wish to set additional configuration options that you would like to mirror on all systems, you can install the software on one system, make any necessary changes to the configuration, then export the registry section for the software to a file.

When you run the installer, it looks for a registry export file and will install those settings. To create the registry export file, click on Start/Run and run **REGEDIT**. Drill down to the following registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\DenicompSystems\WRSHDNT\Setup
```

Then click on Registry/Export Registry File… Export it to a file named **wrshdnt.reg**. Put this file in the same directory as the RSHD/NT installer (rshdnt_reg.exe for example – the download file will also include a version number in its name).

You can review the settings in the .reg files by viewing them (or changing them) with WordPad.

Then, to install with those settings, simply run the installation. It will not ask for a password and the answers to all questions will default to those found in the .reg file. If you do not want to answer **any** questions, you can do a totally silent install by adding **/s** to the end of the installer command line. For example:

```
rshdnt_reg.exe /s
```

The installation will run silently and it will install with all of the settings specified in the original installation if it finds the wrshdnt.reg file in the current directory. Since the registration password is stored in the .reg file, there is no need to specify it on the command line.

Removing Winsock RSHD/NT

To remove Winsock RSHD/NT, use the **Add/Remove Programs** icon in the Control Panel. Find Winsock RSHD/NT in the list of installed programs, click on it once, then click on the **Add/Remove** button (or **Change/Remove** on Windows 2000.

If you wish to remove Winsock RSHD/NT manually, you must first stop the Winsock RSHD/NT service, then remove the service from the Windows list of services before deleting the files. From the Windows Control Panel, double-click on the RSHD icon. Click on the Service Control tab and click on the Stop button to stop the service, then click on the Remove button to remove the service. You can then delete the Winsock RSHD/NT directory.

You may then also delete the Winsock RSHD/NT registry entries using **regedit**. The Winsock RSHD/NT registry entries are stored under the registry keys:

HKEY_LOCAL_MACHINE\SOFTWARE\DenicompSystems\WRSHDNT
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\RSHDNT

Running Winsock RSHD/NT as a Service

Usually you will want to run Winsock RSHD/NT as a service. It will start automatically when Windows boots and will not stop when users log in and out of the system.

Services are controlled (started and stopped) using either the **Services** icon in the Control Panel (which is in the Administrative Tools folder under Windows 2000) or using the Service Control tab in the RSHD Control Panel applet.

When using the Services icon, Winsock RSHD/NT is listed as the service named “Remote Shell Daemon”.

Running Winsock RSHD/NT as an Application

You can optionally run Winsock RSHD/NT as a Windows application instead of a service. This requires that someone log in to Windows to start RSHD/NT.

To start Winsock RSHD/NT as an application, execute the following command:

```
wrshdnt  /s
```

If you are executing this command from the Windows Command Prompt, you should use the command:

```
start wrshdnt  /s
```

If you do not use the **start** command, you will not have access to that Command Prompt window until Winsock RSHD/NT is stopped. Be sure that you do **not** already have Winsock RSHD/NT running as a service.

When Winsock RSHD/NT is started, it will display various messages in the window if you have specified a value in the Message Level field in the RSHD Control Panel applet. If you have not, nothing will display.

You can also include the **/m** or **/h** options along with **/s**. The **/m** option minimizes the RSHD/NT window and the **/h** hides the RSHD/NT window. You must include a space between the **/s** option and one of these. For example:

```
wrshdnt /s /m
```

To stop Winsock RSHD/NT, simply close the window or activate the window and press Control-C. (This is not possible if you used the /h option to hide the window. You can only stop RSHD/NT through the Task Manager if the window is hidden.)

WINSOCK RSHD/NT CONFIGURATION

Winsock RSHD/NT will work properly using its default configuration. You only need to configure RSHD/NT if you wish to change any of the available options, enable security, or use the logging capabilities. By default, no security is enforced and no logging is done

You configure RSHD/NT by using the Windows Control Panel. In the Control Panel, double-click on the **RSHD**. When configuring RSHD/NT, you can press <F1> at any time to display help information about the options available.

After changing most RSHD/NT Configuration options, you do not need to stop and restart Winsock RSHD/NT. It will recognize the change when you click the OK button or the Apply, unless you disable the monitoring of the registry (see below).

In any of the settings that require a filename to be specified, you can use environment variables by enclosing the variable name between percent signs (%). For example, if you wanted to put the Message File in the system's "temp" directory, you could specify that filename as %TEMP%\rshd.log. The environment used will be the environment the service inherits from the Service Manager; it will not use any user-specific environment variables or variables from the Environment Variable File. Additionally, you can use a special environment variable %RSHD%, which will be set to the RSHD/NT installation directory.

Security and Logs

Security File: (Default: None)

Specify the full path name of the Security File used by RSHD/NT to enforce security (allow and deny users and clients). The format of this file is explained in more detail later. If you do not specify a Security File, **all** users and clients will be granted access to execute programs and transfer files to and from your system, unless you enable the option that requires remote user names to exist as Windows users (see below). If you do not wish to enforce any security, do not specify a filename.

If you do specify a Security File and it does not exist or RSHD/NT cannot access it (because of insufficient permissions), ***no*** users or clients will be granted access. The Security File must exist on a local hard drive (not on a network drive). Also be sure to save the Security File as an ASCII text file; if you save it in some other format (such as Write, Word, or Unicode), RSHD/NT will not be able to understand the data in the file.

You can use the **Edit Security File** button to edit the Security File you specified. It will start the Windows Notepad editor. If your Security File does not exist, the file will be created with some comments in it about the format of the file for your convenience.

Must Remote Users be Valid Users on this System? (Default: Unchecked)

If this option is unchecked, user login names sent to RSHD/NT by *rcp* and *rsh* do **not** need to be valid users on this system. Security is enforced only through the Security File and the User Equivalency table.

If this option is checked, user login names from *rcp* and *rsh* **must** be valid users on this system. If the user is not valid, access will be denied. This is the standard behavior of a *rshd*. However, if the *rcp* and *rsh* commands are being executed from another operating system such as Unix, the user login names may not be the same between systems. If user logins are the same, you can then enable this option for security. If they are different, you should not enable this option.

Note: The Unix *rsh* and *rcp* commands have the ability to specify an alternate user (using the –l option for *rsh* and the *user@* construct with *rcp*). It is the standard behavior of the Unix *rsh* and *rcp* commands to transmit ***both*** the alternate name specified on the command line ***and*** the actual login name. Both of these will be validated by RSHD/NT.

Must Users be Set Up as Equivalent Users? (Default: Unchecked)

If this option is checked, remote users **must** be set up in the *User Equivalency* table (using the User Equivalency tab) to equate the remote user with a Windows user. If the user is not found in the User Equivalency table, access will be denied. See the section on User Equivalency for more details about setting this up.

If this option is not checked, access will be granted to remote users and clients as long as the Security File allows access. Commands will be executed in the security context of the user specified in the **Execute All Commands as User** option on the RSH Options tab or the **Execute All RCP Copies as User** option on the RCP Options tab. If no user is specified in those options, the command will be executed in the security context in which the service is running (usually the System Account).

Always Use Remote Login as Equivalent User? (Default: Unchecked)

If you are using the *User Equivalency* table to equate a remote user with a Windows user, normally the login of the remote user will be used to look up the user in the table, **unless** that user is overridden in the *rsh* command with the **–l** option or in the *rcp* command with the **user@** option. If the user override option is used, ***that override user will be used to look up the user in the User Equivalency table instead***.

This is potentially a security problem, since any user can override their login with another user name. For example, let's say your User Equivalency table is set up like this:

Remote User	Windows Equivalent
john	Guest
mary	Guest
root	Administrator

When the remote users "john" or "mary" issue *rsh* commands, they will be executed as Guest by RSHD/NT. However, if john adds the option **"-l root"** to his *rsh* command, his command will execute as Administrator – there is nothing to stop him from doing that, since *rsh* does not require passwords.

Checking this option basically disables the **–l** option of the *rsh* command when executing commands through RSHD/NT – the user specified after **–l** will be ignored for the purpose of looking it up in the User Equivalency table. This also applies the **user@** option of the *rcp* command.

This option **only** affects the lookup of users in the User Equivalency table. If you are specifying users in the Security File, both the remote login and the override user will be validated and if you checked the option **Must Remote Users be Valid Users on This System?**, both the remote login and the override user will be validated.

Message Log: (Default: None)

Specify the full path name of a file where any messages from RSHD/NT should be stored. The message file is optional**.** You should only enable the message log when you are trying to find the source of a problem or you need to closely monitor all access, since the message log can become quite large on an active system.

This option is used in conjunction with the **Message Level** option. If **Message Level** is set to a value greater than zero (0), RSHD/NT will output messages that provide information about its operation. These messages are mostly useful for problem determination.

The message file created is a text file that you can examine at any time using utilities such as TYPE or MORE, or editors such as Notepad. You can clear the message log at any time by simply deleting it or clicking the Truncate button.

Message Level: (Default: 0)

Specifies the level of detail of the messages stored in the file specified in the **Message Log** option. The default level is **0** (or blank), which will not write any messages to the Message Log file. Levels 1 through 9 will produce increasing amounts of detail (level 1 provides the least detail, level 9 provides the most).

Request Log: (Default: None)

This option allows you to log all requests (programs to be executed) in a file you specify. Each time someone attempts to execute a program through RSHD/NT, the date and time, the user name, the client hostname, and the program will be written to this file. If you would rather have this information logged as standard Windows events (viewed through the Event Viewer), specify ***EventLog** in this field.

Deny Log: (Default: None)

This option allows you to log all permission violations in a file you specify. Each time someone is denied permission to execute a program or copy a file through RSHD/NT, the date and time, the user name, the client hostname, and the program will be written to this file. If you would rather have this information logged as standard Windows events (viewed through the Event Viewer), specify ***EventLog** in this field.

Error Log: (Default: None)

This option allows you to log all program execution errors in a file you specify. Each time someone receives an error trying to execute a program through RSHD/NT, the date and time, the user name, the client hostname, the program, and error message will be written to this file. These are errors that occur after the user has been granted permission to execute the program. For example, an error would be logged if a program were to be run that did not exist. If you would rather have this information logged as standard Windows events (viewed through the Event Viewer), specify ***EventLog** in this field.

NOTE: Each of the log files may refer to the **same** file name if you wish. They will not overwrite each other. Each message is appended to the end of the file. You should be sure to periodically delete the log file(s) because they can get large over time on an active system. You can do this by simply deleting the files or by clicking the Truncate button in the configuration window. (The Truncate button has no effect if you specified ***EventLog** as the filename because the data is being logged in the Windows Event Log. Use the Event Viewer to clear the events.)

User Equivalency

The User Equivalency table allows you to associate remote user logins with Windows logins. Since the *rsh* command does not ask for a password; it assumes *user equivalency* between the client and the server. That is, it is assumed that the user “jsmith” on the client is the same user as “jsmith” on the server, so no password needs to be specified.

This works with the Unix *rshd* (the operating system where the *rsh* protocol originated) because under Unix, a privileged process running as the super-user (such as the Unix *rshd*) can impersonate any Unix user without knowing that user’s password. So when the client issues the *rsh* command while logged in as “jsmith”, the Unix *rshd* can run the command as “jsmith” without knowing jsmith’s password.

Windows does not have a similar capability. Unlike Unix, Windows ***requires*** that a password be supplied in order for any program to impersonate another user. So RSHD/NT cannot arbitrarily impersonate any user passed to it by the client without knowing that user’s password. But the *rsh* command does not ask for a password, so RSHD/NT cannot do the impersonation.

Instead, as a compromise, RSHD/NT allows you to set up a **User Equivalency** table where you can associate remote user logins with Windows logins and supply the Windows password for that login. RSHD/NT can then impersonate that user when a command is received from that user.

Keep in mind that there are some caveats with doing this. You must know the password for each user entered into the table. Also, if a user’s Windows password expires or changes, *rsh* commands from that user will fail if the table is not updated with the new password.

Alternatively, you can choose to not use the User Equivalency table. Instead, you can specify a single user and password on the RSH Options tab and the RCP Options tab and all commands will be executed in the Windows security context of that user, regardless of the user login received from *rsh*. You can still validate and/or restrict users who gain access using the Security File, but once access is granted, the user received from *rsh* becomes irrelevant.

For example, you can create a user “rshd” in Windows and give that user access to the files and resources that users will need when executing *rsh* and *rcp* commands. Then specify “rshd” in the field **Execute All Commands as User** on the RSH Options tab (and the RCP Options tab if necessary). Then, regardless of who issues the *rsh* or *rcp* command, the command will execute as the user “rshd”.

Of course, there are downsides to this method also. You may not want all users using *rsh* and *rcp* to have access to the same files and resources. In that case, it is best to use the User Equivalency table so the user can vary among commands.

To add a new user to the User Equivalency table, enter the remote login in the Remote User field. This will be the login used on the client system, unless the **–l** option is used in the *rsh* command to specify an alternate user, or the **user@** option is used in the *rcp* command to specify an alternate user.

You can simply specify the remote user's login and the record will be used for that remote user from any client. Or you can specify the remote user's login in the format **user@hostname** or **user@ipaddress**. If the at-sign (@) is used in the remote user name, the record will only apply to the remote user when the *rsh* originates at the hostname or IP address specified after the @. This allows you to give different Windows logins to the same remote user from different systems or allows you to restrict access to only a limited number of clients.

SECURITY NOTE: The **–l** option in the *rsh* command and the **user@** option in the *rcp* command will allow any remote user to impersonate any user in your User Equivalency table. For example, if you set up an entry that equates the remote user "root" with the Windows user "Administrator", ***any*** remote user will be able to use **–l root** in their *rsh* command and execute a command as Administrator. If you do not desire this capability, you can check the option **Always Use Remote Login as Equivalent User?** on the Security and Logs tab . Checking that option effectively disables the ability to override the actual user login in *rsh* and *rcp*. When that option is checked, **only** the user's actual login on the remote system is used to obtain an equivalent user.

After specifying the remote user, enter the equivalent Windows User, Windows Domain name if necessary (if the user is a Domain user and not a local user on this system), and the Windows Password for that Windows user. Then click the Add button to save it.

To modify an existing user in the table, find the user in the list and click on it once. You will see the information for that user appear in the fields. Update the necessary information and click the Update button to save it.

To delete a user from the table, find the user in the list and click on it once, then click the Delete button.

NOTE: If you use the User Equivalency table, you must run the RSHD/NT service in the **System Account** (the default setup for the service) and enable the option to interact with the desktop. If you do not want to run the RSHD/NT service in the System Account for some reason, you can specify an alternate user in the service startup, but that user must have the following Windows rights:

- Act as part of the operating system
- Replace a process level token
- Increase quotas

These rights can be assigned to the user through the Windows user setup. You only need to deal with giving a user these rights if you specify that user in the RSHD/NT service startup.

Also, when running the service as a user other than the System Account, you will no longer see any windows of any program started through RSHD/NT. Only the System Account has access to the Windows desktop (screen), so Windows will run the programs on an invisible desktop when running the service as another user.

RSH Options

Reject All Incoming RSH Commands: (Default: Unchecked)

If you check this option, all incoming *rsh* commands will be rejected, effectively disabling the *rsh* serving capability of RSHD/NT. This is useful if you only want to use RSHD/NT as an ***rcp*** server.

If a remote user attempts to issue an *rsh* command to this system, an error will be returned to the remote user stating that *rsh* has been disabled.

Attempt Redirection on Every Command: (Default: Checked)

If this option is checked (it is by default), RSHD/NT will assume that each program executed by *rsh* is a Windows Console application or MS-DOS program and attempt to send its standard output/standard error back to the *rsh* client and send the standard input from the *rsh* client to the program.

A side effect of this option is that it will cause the *rsh* command on the client to wait until the program completes. This is the standard behavior of a remote shell daemon.

If you do not want the *rsh* command on the client to wait until the program completes, uncheck this option. Or, if you only occasionally do not want the *rsh* command to wait for the program to complete, use the special <[WIN]> directive in the *rsh* command. See the section on Standard Input/Output/Error Redirection for more details.

Wait for Command to Complete: (Default: Unchecked)

If this option is checked, RSHD/NT will cause the *rsh* command on the client to wait until the program executed ends. This option is only available when the **Attempt Redirection on Every Command** is not checked.

If the **Attempt Redirection on Every Command** option is not checked and this option is not checked, the *rsh* command on the client will not wait for programs to complete; it will end as soon as the program is started on the system running RSHD/NT. By checking this option, it allows you disable redirection by unchecking the **Attempt Redirection on Every Command** option, but still have *rsh* wait for programs to complete.

Buffer Stdout/Stderr Until End of Command: (Default: Unchecked)

Check this option if you want RSHD/NT to buffer (store in a file) the standard output and standard error output of the commands you execute, and then send all of the output when the command completes. When this option is checked, standard input from the *rsh* command is **not** passed to the program executed.

Occasionally, this option is necessary when the program executed also uses redirection or pipes, but it is rarely necessary.

Execute All Commands through Command Shell: (Default: Unchecked)

If you check this option, RSHD/NT will automatically prefix every program you execute with the default command shell (usually **cmd /c**).

This is useful if you commonly execute batch files (.BAT or .CMD) or other command scripts for the shell you are using and you do not want to have to specify the shell command in every *rsh* command.

The default shell command is used as the prefix. This command can be specified in the **Default Shell Command** field; if no default shell command is specified there, **cmd /c** is used.

For example, if this option is enabled and you execute the following command from a remote system:

```
rsh server xyz.bat
```

RSHD/NT will execute the command as:

```
cmd /c xyz.bat
```

Disable Detection of Internal Commands: (Default: Unchecked)

When you execute a program through RSHD/NT, it examines the command to determine whether or not it is a command internal to the default command shell (interpreter). If it is, it automatically prefixes the command with the default command shell (specified in the **Default Command Shell** field).

Some commands are not actually programs; they are interpreted and executed internally by the command shell. In the Windows command interpreter (CMD.EXE), some examples of these are DIR, SET, and COPY. If you look on your hard drive, you will not find a DIR.EXE or COPY.EXE. They are part of the command interpreter, CMD.EXE.

So, if you tried to execute a DIR command through RSHD/NT, it would not find the program since it doesn't exist. You would have to tell it to use the command interpreter by executing the command *CMD /C DIR.*

By default, RSHD/NT examines the command and if it determines that the command is an internal command, it adds the shell command for you. You can disable this by checking this option. All commands will be executed as they are specified in the *rsh* command.

Disable Watch for Kill Signal from RSH: (Default: Unchecked)

RSHD/NT watches for special signals from the *rsh* client that tell it when the remote user pressed an "interrupt" or "quit" key. When one of these signals is received, RSHD/NT will then attempt to kill the program being executed. If you check this option, RSHD/NT will not watch for this signal and the interrupt or quit keys from the client will be ignored.

Generally, on Unix systems, the "interrupt" key is the Control-C key or the Delete key, and the "quit" key is Control-Backslash. However, if you are issuing the *rsh* command from a Windows system and you are using the native *rsh* command included with the operating system, it does **not** send the special signal to RSHD/NT when you press Control-C or Control-Break. It simply ends the *rsh* command and the process will remain running on the remote system.

If you need to be able to send the interrupt signal from Windows, see www.denicomp.com for our Winsock RCP/RSH/REXEC for Win32 package. It includes an *rsh* client that will send the proper signal when Control-C is pressed.

NOTE: RSHD/NT can only kill the process it created. If you start a program with RSHD/NT and it starts other programs, pressing the interrupt key will only kill the process originally started by RSHD/NT; the other programs started indirectly will not be killed, since RSHD/NT does not know about them.

Disable Killing Command on Client Disconnect: (Default: Unchecked)

RSHD/NT watches to see if the *rsh* client disconnects prematurely ***if*** stdin/stdout/stderr is being redirected back to the client. If it detects that the client has disconnected, it will kill the program being executed.

A standard *rsh* client will send an interrupt signal when the user on the client system presses the system interrupt key (see the **Disable Watch for Kill Signal from RSH** option for more details). However, some *rsh* clients (such as the native Windows *rsh* client) do not properly pass through the interrupt signal. Or if the *rsh* client aborts abnormally or is killed at the process level, it will not have the opportunity to send the interrupt signal to RSHD/NT.

In this case, RSHD/NT will detect that the client is no longer connected and will assume that it was killed and will then kill the program executing.

If you do not want it to do this, check this option. The program executed through RSHD/NT will continue to run even after the client has disconnected.

NOTE: RSHD/NT can only kill the process it created. If you start a program with RSHD/NT and it starts other programs, it will only kill the process originally started by RSHD/NT; the other programs started indirectly will not be killed, since RSHD/NT does not know about them.

Load User Profiles: (Default: Checked)

When this option is checked, RSHD/NT will load the user profile and registry hive of the user it is impersonating for the command. The user it impersonates either comes from the User Equivalency table or the user specified on the RSH Options tab in the field **Execute All Commands as User**.

If this option is checked, the registry key HKEY_CURRENT_USER will map to the user's registry settings (printers, environment variables, etc.).

If this option is not checked, the registry key HKEY_CURRENT_USER will map to the registry settings of the default user (HKEY_USERS\.DEFAULT). (This was the behavior of versions of RSHD/NT prior to version 2.20.)

If the remote commands you are executing do not require the information loaded from the user's profile, then unchecking this option will increase performance of RSHD/NT.

Load User-Specific Environment Variables: (Default: Checked)

When this option is checked, programs executed through RSHD/NT will have environment variables set to values specific to the user it is impersonating for the command (plus any settings in the Environment Variable File). The user it impersonates either comes from the User Equivalency table or the user specified on the RSH Options tab in the field **Execute All Commands as User**.

If this option is not checked, the environment variables will have values specified only in the System Environment Variables, plus any values set by the **Environment Variable File** option. (This was the behavior of versions of RSHD/NT prior to version 2.20.)

If the **Load User Profiles** option is not checked, then this option will not be available, since it requires the profile to be loaded.

Default Window Type for Commands: (Default: Normal)

This specifies the default window type to be used when executing commands through RSHD/NT using the *rsh* command. The options available are:

Normal	The window for the command will display at its normal size.
Minimized	The window for the command will be minimized (without focus).
Maximized	The window for the command will be maximized.
Hidden	The window for the command will be hidden.

The default window type is used when the special window type indicators (*<[NORMAL]>*, *<[MINIMIZE]>*, *<[MAXIMIZE]>*, *<[HIDE]>*, etc.) are not specified in the *rsh* command.

There are a few points you must consider when selecting the default window type:

- The **Minimized** or **Hidden** options are useful when the system running RSHD/NT is actively used and is not a standalone server. With the **Normal** or **Maximized** options, the person using the system will see a window appear each time a command is executed through *rsh*.
- You ***cannot*** send keystrokes to commands that are minimized or hidden. Therefore, if you select the minimized or hidden option as the default, RSHD/NT will automatically run any program in a normal window when keystrokes are specified in the *rsh* command, overriding this setting.
- Using the **Hidden** option can cause administrative problems. You won’t be able to use the mouse at the console to close a hung process when its window is hidden.
- The **Hidden** option uses less system resources because RSHD/NT will run the command on an invisible “desktop”. The other options require RSHD/NT to modify security information in the visible desktop so programs can display their windows. With the Hidden option, this step is not necessary.

List of Commands to Allow (File): (Default: None)

This option allows you to specify the name of a file that contains a list of commands that users are permitted to execute on this system through *rsh*. This allows you to provide strict control over the commands users can execute.

If no filename is specified here, all commands are permitted.

The file must be a plain text file, with each permitted command on a line by itself. Commands in the file should not contain any spaces. Comparison of commands is done only up to the first space or tab character.

When a user executes a command on this system through *rsh*, RSHD/NT will extract the first part of the command, up to the first space or tab character, and compare that to the lines in the file specified. If it does not exist in the file, the *rsh* command will be rejected.

Environment Variable File: (Default: Blank)

This allows you to specify the name or names of files that contain environment variables that should be made available when programs are executed by RSHD/NT through *rsh*.

When the option **Load User-Specific Environment Variables** is checked, programs executed through RSHD/NT have an environment made up of the System Environment Variables and the User Environment Variables, specified in the System applet in the Control Panel.

When that option is not checked, the environment comes from the System Environment Variables only. Those variables are inherited from the Windows Service Manager, so if the System Environment Variables are changed, you must reboot the system for them to be propagated to RSHD/NT (if you are running RSHD/NT as a service).

Alternatively, you can create a custom environment for RSHD/NT by entering the environment variables and values in a plain text file. Each line in the text file should have the format:

```
VARIABLE=VALUE
```

Each time a program is executed through RSHD/NT by *rsh*, a custom environment is built from the file or files specified in this parameter, based on the lines in those files.

You can specify a single filename or multiple filenames, with each separated by semi-colons (;). Each file is read in sequence and added to the System Environment Variables inherited by RSHD/NT to create a new environment for the program to be executed. If a variable name appears in multiple files, the last value read will be used.

You may reference previously set environment variables as you do in Windows batch files using **%VAR%**. For example:

```
PATH=%PATH%;C:\MYPROGS
```

The filenames should be full path names. There are three special keywords that you can use in the filenames if you wish:

%ruser% - Substitute the login name of the remote user
%luser% - Substitute the login name of the local user
%rhost% - Substitute the hostname of the remote client

The **%ruser%** substitutes the login name of the remote user. This will be the login name the user used to log into the remote client from which the *rsh* command is being issued, unless the **-l** option of the *rsh* command was used to specify a different user; then that user will be substituted.

The **%luser%** substitutes the login name of the local user on the remote client. Normally, it is the same as **%ruser%**, unless the **-l** option of the *rsh* command was used. Then, this will contain the actual user login used at the remote client.

For example, if you are logged in as “john” on a remote client and you issue the command “rsh -l mary server xyz”, the %ruser% will substitute “mary” and the %luser% will substitute “john”.

The **%rhost%** substitutes the hostname of the remote client, if it is available. That is, RSHD/NT must be able to find the name of the remote client based on its IP address, either by using the HOSTS file or DNS. If it is not found, the IP address will be substituted.
These special keywords allow you to have different environment files for different users if necessary. For example, if you specify the environment variable file:

```
    c:\env\%ruser%.env
```

When “john” issues an *rsh* command, RSHD/NT will get the environment from the file “c:\env\john.env”. When “mary” issues an *rsh* command, RSHD/NT will get the environment from the file “c:\env\mary.env”.

Also, using the capability to specify multiple files, you can have a single “master” environment, and then only modifications to it by user. For example, you can have a standard set of environment variables in the file “c:\env\master.env” and user-specific modifications in the file “c:\env\%ruser%.env”. Your environment variable file field would read:

```
    c:\env\master.env;c:\env\%ruser%.env
```

First the variables in master.env would be read, then those for the user in %ruser%.env.

The Special "new" Keyword

The format of the environment variable files must be:

```
VARIABLE=VALUE
```

But, with one exception. If you specify the word “**new**” on a line by itself in an environment variable file, it will purge all environment variables set up to that point.

The primary purpose of this would be to remove all variables inherited from the System Environment Variables or variables created by loading the User Environment Variables. It allows you to start with a “clean slate” and set all environment variables from scratch.

Default Command Shell: (Default: None, use CMD /C)

This allows you to specify the default command shell to be used when RSHD/NT detects an internal command or the command shell to be used if the option to Execute All Commands through Command Shell is checked.

You should use this option only if you are using an alternate command shell. By default, RSHD/NT uses the Windows command shell CMD.EXE. You must specify all necessary options to the command shell so that it can be prefixed to any command (internal or external).

If you end the shell command with a single quote (') or double quote ("), RSHD/NT will supply the closing quote. This is necessary if the shell program would interpret parts of the command executed as its own options. For example, if you are using a Windows version of the Korn shell, specify the following as the command shell:

```
ksh -c "
```

If you do not specify the trailing quote, the ksh command will interpret options to the commands you execute instead of passing them to the command.

Internal Command List: (Default: None, use CMD list)

If you specified a Default Command Shell that has different internal commands than those of the standard Windows command shell CMD.EXE, you can specify the internal commands for that shell here. Separate each command with a comma (,). Do not include any spaces. By default, RSHD/NT recognizes the internal commands of the Windows command shell.

Internal commands are commands that are interpreted and executed by the command shell. They do not exist as executable files (.EXE or .COM). If RSHD/NT does not recognize a command as an internal command, you must prefix it with the command shell.

You only need to specify this list if you want RSHD/NT to recognize commands internal to an alternative command shell and automatically prefix the command with the appropriate shell command.

For example, you could specify:

```
cd,dir,type
```

Whenever you execute the command:

```
rsh server dir
```

It would see that "dir" is in the list and prefix it with the Default Command Shell before executing.

Execute All RSH Commands As User/Password: (Default: None)

If you specify a user and password here, all commands run through RSHD/NT will be run in the security context of the specified user, if that user is not found in the User Equivalency table (found on the User Equivalency tab).

RSHD/NT first searches the User Equivalency table for a matching remote user. If one is found, it will use the Windows login information found there for the command. Otherwise, it will use the user/password found here.

If these fields are blank and the user is not found in the User Equivalency table, RSHD/NT runs all commands in the context of the user specified in the RSHD/NT service setup, in the service’s Startup options.

You will not be able to access these fields if you have checked the option **Must Remote Users be Setup as Equivalent Users?** When that option is checked, remote users must be present in the User Equivalent table, so these fields would never be used.

By default, RSHD/NT is installed to run all commands as the special user **System** (also known as **LocalSystem** or the **System Account**). This is a special user that has access to the desktop (screen) regardless of who is logged in. However, the System user does not have access to any network shares, such as drives or printers. If the programs you run using *rsh* require access to network resources, you can either change the RSHD/NT service setup to run the service as a different user or you can specify a user and password in these fields.

The user you specify here can be a normal user account or you can create a special account just for the purpose of *rsh*. It is recommended that you specify that the user’s password does not expire, however. If the user’s password expires, *rsh* commands will not function until you manually update the RSHD/NT setup with the new password. Windows will not update it for you.

The differences between specifying a user in these fields and changing the RSHD/NT service setup are:

- If you change the service setup to run as a non-System user, you will no longer see the window of any commands you run through RSHD/NT. Windows will run them on an invisible desktop. If you specify a user and password in these fields, you will continue to see the window of commands run.

- If you change the service setup to run as another user, access to files copied with *rcp* will also be done in the context of that user. If you specify a user and password in these fields, *rcp* copies will continue to be done as the user specified in the RSHD/NT service setup, unless a user is specified in the RCP configuration section (explained later).

If the user is a domain user, you must use the format **DOMAIN\User**. For example, to specify the user "remote" in the domain ABC, you would enter it here as ABC\remote.

IMPORTANT: If you specify a user and password in these fields, you must run the RSHD/NT service using the **System Account** and also enable the option to allow the service to interact with the desktop. This is the default service setup, so if you have not changed it from its initially installed state, no changes are required. If you do not want to run the service in the System Account for some reason, you can run the service as another user, but that user must have the following Windows rights assigned:

- Act as part of the operating system
- Replace a process level token
- Increase quotas

These rights can be assigned to the user through the Windows user setup. You only need to deal with giving a user these rights if you specify that user in the RSHD/NT service startup.

Also, when running the service as a user other than the System Account, you will no longer see any windows of any program started through RSHD/NT. Only the System Account has access to the Windows desktop (screen), so Windows will run the programs on an invisible desktop when running the service as another user.

RCP Options

Reject All RCP Copies to This System: (Default: Unchecked)

If this option is checked, all attempts to copy files to this system with the ***rcp*** command will be rejected with an error message stating that incoming copies are disabled. This allows you to make the system “read only” when using the ***rcp*** command. You can also reject copying from the system, essentially disabling the ***rcp*** capability of RSHD/NT.

Reject All RCP Copies to From System: (Default: Unchecked)

If this option is checked, all attempts to copy files from this system with the ***rcp*** command will be rejected with an error message stating that outgoing copies are disabled. This allows you to make the system “write only” when using the ***rcp*** command. You can also reject copying to the system, essentially disabling the ***rcp*** capability of RSHD/NT.

Preserve Case in Multi-File Copies: (Default: Unchecked)

Specifies whether RSHD/NT should preserve the case of filenames when files are copied **from** this system by *rcp* using wildcards or recursive copies. By default, when the remote system uses a wildcard or recursive copy to get files from this system, RSHD/NT will convert all directory and filenames to **lowercase** letters before sending them to the remote system.

Although the Windows filesystem is not case sensitive (ABC and abc are the same file), it can store the case of the filename. When copying files via *rcp* to operating systems that are case sensitive, such as Unix, it is usually most useful to convert all of the names to lowercase letters.
If you do not wish to have all of the names converted to lowercase letters, check this option. The *rcp* command will then create files in exactly the same case as the names appear in the directory under Windows.

Note that this affects **only** wildcard and recursive copies. When copying individual files, the files will be created in the case you specify in the *rcp* command.

Automatic End-of-Line Conversion: (Default: Never)

This specifies whether or not RSHD/NT should perform any end-of-line conversions on files transferred to or from the system using ***rcp***.

With Windows, lines of text are delimited by carriage return and newline pairs (ASCII 13 and ASCII 10). With Unix, lines of text are delimited by only newlines (ASCII 10). Often, when copying text files between the two operating systems, it is necessary to convert the end-of-line delimiters to the proper method. RSHD/NT provides a way to automatically do this.

When files are copied from the Windows system through RSHD/NT, carriage returns will be removed from all carriage return/newline pairs (i.e. converted to Unix format). When files are copied to the Windows system through RSHD/NT, carriage returns will be added to every newline character that is not already prefixed by a carriage return (i.e. converted to Windows format).

There are four options available. The option you select affects all ***rcp*** copies to this system and all ***rcp*** copies from the system. It does not affect the operation of the ***rcp*** command on the Windows system. It only affects the result of ***rcp*** commands that access files on this system from other systems.

- **Never - Copy all files as binary**
  No end-of-line conversion. All files will be transferred or received unmodified.

- **Always - Convert all files**
  Convert end-of-line characters of every file copied from or to this system through RSHD/NT.

- **Convert based on list of file extensions**
  Only convert end-of-line characters in files ending with the specified list of file extensions; any file not ending in the specified extensions will be copied without modification. You must then enter the list of file extensions. Separate each file extension by a comma. Do not include any spaces. You must include the "dot" (.). For example:
  .TXT,.C,.H,.PRN,.MAK

- **Convert based on contents of first block**
  RSHD/NT will examine the contents of the first block of the file to be sent or received and determine whether or not an end-of-line conversion is necessary. If the first block contains only text characters (letters, numbers, spaces, tabs, carriage returns, newlines, backspaces, escapes, and form feeds), RSHD/NT will perform an end-of-line conversion on the file. If the first block contains any other non-text data, it will be copied without modification. The size of the first block is specified in the **RCP Block Size** field.

RCP Home Directory: (Default: Blank)

Specifies the starting directory where files will be copied from or to when a relative path name is used in the ***rcp*** command (no initial slash or backslash).

This directory must exist if specified. The directory name specified can contain the following special keywords:

%ruser% - Substitute the login name of the remote user
%luser% - Substitute the login name of the local user
%rhost% - Substitute the host name of the remote client

The **%ruser%** substitutes the login name of the remote user. This will be the login name the user used to log into the remote client from which the ***rcp*** command is being issued, unless the **@** option of the ***rcp*** command was used to specify a different user; then that user will be substituted (e.g. user@host:filename).

The **%luser%** substitutes the login name of the local user on the remote client. Normally, it is the same as **%ruser%**, unless the **@** option of the ***rcp*** command was used. Then, this will contain the actual user login used at the remote client.

For example, if you are logged in as “john” on a remote client and you issue the command “rcp xyz mary@server:”, the %ruser% will substitute “mary” and the %luser% will substitute “john”.

The **%rhost%** substitutes the host name of the remote client, if it is available. That is, RSHD/NT must be able to find the name of the remote client based on its IP address, either by using the HOSTS file or DNS. If it is not found, the IP address will be substituted.

If you specify a directory here, you will then have the option of checking the sub-option **Restrict All RCP File Accesses to This Directory.** If you check this option, when a client uses ***rcp*** to copy files to or from this system, this directory will be treated as the “root” directory and any files referenced will be under the RCP Home Directory. This allows you to restrict ***rcp*** so it can only access files in specific locations.

For example, if you set the RCP Home Directory to C:\RCPDIR and a client issues the command:

```
rcp somefile ntserver:/dir/file
```

this will copy “somefile” to c:\rcpdir\dir\file.

This does **not** affect ***rsh*** commands. Windows does not provide the capability for RSHD/NT to externally restrict a program it executes to a specific directory.

RCP Block Size: (Default: 1024)

Specifies the number of bytes in a block of data that the Remote Copy (*rcp*) service of RSHD/NT processes at one time. When files are copied to the system, it reads from the network and writes to the disk in blocks of this size. When files are copied from the system, it reads from the disk and writes to the network in blocks of this size. Note that this is an internal block size only; it does not change any TCP/IP parameters.

RCP Spoofing Prefix: (Default: None)

This specifies the first characters of the *rcp* command sent by the remote client that RSHD/NT should use when "spoofing" the *rcp* protocol. With its roots in Unix, the *rcp* command actually internally executes an *rsh* command to start *rcp* on the remote client before transferring files. RSHD/NT "spoofs" or looks for *rcp* commands executed through *rsh* by the remote client and services the *rcp* transfer. By default, RSHD/NT looks for the command prefixes of:

```
rcp -
/usr/bin/rcp -
/usr/lib/sunw,rcp -
set vms_rcp = 1 ; rcp -
```

Some *rcp* commands (especially those on non-Unix and non-Windows systems) may send other commands to initiate the *rcp* protocol. If yours does, you should enter the command prefix (up to and including the first hyphen) here. The last character should **always** be a hyphen with nothing after it, including spaces.

Regardless of the spoofing prefix entered, RSHD/NT will continue to look for the above default prefixes.

Execute All RCP Copies As User/Password: (Default: None)

If you specify a user and password here, all files accessed through RSHD/NT by the *rcp* command will be done in the security context of the specified user, if that user is not found in the User Equivalency table (found on the User Equivalency tab).

RSHD/NT first searches the User Equivalency table for a matching remote user. If one is found, it will use the Windows login information found there for the command. Otherwise, it will use the user/password found here.

You will not be able to access these fields if you have checked the option **Must Remote Users be Setup as Equivalent Users?** When that option is checked, remote users must be present in the User Equivalent table, so these fields would never be used.

If these fields are blank and the user is not found in the User Equivalency table, RSHD/NT accesses all files in the context of the user specified in the RSHD/NT service setup, in the service's Startup options.

When RSHD/NT services the *rcp* command, it will essentially log in as this user. It will be able to read and write any files that this user can read and write. If new files are created, they will be owned by this user (not the user on the remote system). So you can control access to files through *rcp* by modifying the permissions of this user.

The user can be a normal user account or you can create a special account just for the purpose of *rcp* copies. It is recommended that you specify that the user's password does not expire, however. If the user's password expires, *rcp* copies will not function until you manually update the RSHD/NT setup with the new password. Windows will not update it for you.

Advanced Options

Installation Directory: (Default: C:\WRSHDNT)

This is the directory where RSHD/NT is installed. This will be filled in by the RSHD/NT installation.

Initial Working Directory: (Default: None, use Installation)

This specifies the directory that will initially be considered the current working directory for all commands executed using *rsh*. It will also be considered the current working directory for all files copied using *rcp*, unless an **RCP Home Directory** is specified. If this is blank, then the **Installation Directory** becomes the initial working directory.

When RSHD/NT starts, it changes to this directory and remains there, unless an *rsh* request is received to execute the **cd** or **chdir** command, which will change RSHD/NT's working directory.

RSHD/NT intercept **cd** and **chdir** commands when possible and handles them internally, so it can track the "current directory" by user.

Disable Multithreading in RSHD/NT: (Default: Unchecked)

Multithreading allows RSHD/NT to process multiple requests simultaneously. When multithreading is disabled by checking this option, RSHD/NT will accept and complete only one request at a time. Other requests received during this time will be queued and executed in the order in which they were received.

Normally, you will want multithreading enabled, but you can disable it, for example, to ensure that the system will not become bogged down with requests.

Disable Monitoring of Registry for Changes: (Default: Unchecked)

Normally, RSHD/NT monitors the Windows Registry for changes to the RSHD/NT configuration options and if any options are changed, re-reads the registry so that the new options take effect.

If this option is checked to disable the monitoring of the Registry, you must stop and start RSHD/NT manually (or reboot the system) for the Registry changes to take effect. You may want the Registry monitoring disabled for security purposes so that no RSHD/NT options are changed while the system is in operation.

Host IP Address (If Multi-Homed): (Default: None)

If your system is multi-homed (it has multiple network cards, each with its own IP address), you can specify which IP address RSHD/NT will use to listen for requests. If you leave this empty, it will accept requests from any of the IP addresses associated with the system. If you specify one of the addresses of one of the cards (in dotted-decimal format), it will only accept requests routed to that address.

Listen Port: (Default: 514)

Specifies the port number that RSHD/NT listens to for connections. The standard RSHD port is 514.

Listen Backlog: (Default: 100)

Specifies the number of requests that can be *backlogged* when RSHD/NT is listening for connections. The minimum is 1; the maximum is dependent on the version of Windows being used.

ENFORCING SECURITY

RSHD/NT uses the following steps before accepting a command from *rsh* or *rcp* from a client:

Is the option "**Reject All Incoming RSH Commands**" checked?
If yes, and the client issued an *rsh* command, access is denied.

Is the option "**Reject All RCP Copies To This System**" checked?
If yes, and the client issued an *rcp* command to copy files to this system, access is denied.

Is the option "**Reject All RCP Copies From This System**" checked?
If yes, and the client issued an *rcp* command to copy files from this system, access is denied.

Is the option "**Must Remote Users be Valid Users on this System**" checked?
If yes, it checks to be sure the user login received from *rsh* is a valid Windows user. If it is not, access is denied.

Is there a **Security File** specified?
If yes, and the file does not exist, access is denied. If the file does exist, RSHD/NT searches the file to find out if the user and/or client hostname or IP address is allowed access. The process it uses to do this is explained in more detail in the section on the RSHD/NT Security File. If the Security File does not allow the user and/or client, access is denied.

Is the option "**Must Users be Setup as Equivalent Users**" checked?
If yes, it checks the User Equivalency table and if the user does not appear in the table, access is denied.

Was the user was found in the **User Equivalency** table?
If yes, RSHD/NT logs in as the Windows User and Password associated with the remote user. If the stored user and password is invalid (for example, the password is incorrect or expired or the account is disabled), access is denied.

If the user was not found in the User Equivalency table, and the client issued an *rsh* command, is there a user and password specified in the fields **Execute All Commands as User/Password** on the RSH Options tab?
If yes, RSHD/NT logs in as that user and password. If that user and password is invalid, access is denied. If no user and password is specified in those fields, access is granted and the command is executed in the security context of the RSHD/NT service. By default, the service runs in the System Account.

If the user was not found in the User Equivalency table, and the client issued an *rcp* command, is there a user and password specified in the fields **Execute All RCP Copies as User/Password** on the RCP Options tab?
If yes, RSHD/NT logs in as that user and password. If that user and password is invalid, access is denied. If no user and password is specified in those fields, access is granted and files are accessed using the security context of the RSHD/NT service. By default, the service runs in the System Account.

Winsock RSHD/NT Security File

The Security File allows you to specify a list of client hostnames or IP addresses and optionally user names that are permitted or denied access through RSHD/NT. The name of this file is specified in the RSHD/NT Configuration in the **Security File** field. It is similar to the */etc/hosts.equiv* and *$HOME/.rhosts* files used by the Unix *rshd*, although not totally compatible.

The Security File is **only** used to determine which users or clients may access the system through RSHD/NT. It has **no effect** on the security attributes of commands executed through RSHD/NT once it determines that access is permitted.

Depending on your security needs and how you will use RSHD/NT, you may or may not need the Security File. The setup option **Must Users be Setup as Equivalent Users?** requires that remote users be set up in the User Equivalency table. If you check this option, and set up all remote users in the format **user@hostname** there, that may provide all of the security necessary to prevent unauthorized access. You need to evaluate your security requirements and how RSHD/NT will be used to determine which method is best to use or if both should be used.

If you specify a Security File name and the file does not exist or the file is empty, **all** clients and users are denied access.

Conversely, if you do not specify a Security File, all clients and users are granted access, assuming they are not denied based on some other option (see the section on Enforcing Security for more details). So if you do not wish to enforce security with a Security File, do not specify a Security File name in the configuration file.

You create the Security File using a text editor. If you are using the RSHD/NT Control Panel applet, you can click on the **Edit Security File** button to run the Windows Notepad editor to edit the Security File specified in the Security File configuration option. If the Security File does not exist when you click the **Edit Security File** button, the file will be created with some comments in it about the file format for your convenience, similar to the information in the table below.

It is highly recommended that you set the “read only” attribute on the Security File, so a remote user cannot overwrite the file or change it.

The following shows the format of lines that may be placed in the Security File:

#	Any line beginning with # is treated as a comment and is ignored.
+	A plus sign (+) on a line by itself specifies that ALL clients and users are granted access. This is useful if you wish to allow many clients and users, but deny only a few. Use the deny options on subsequent lines.
client	A line with just the client hostname or IP address grants access to **all** users on that client, unless you specifically deny those users on subsequent lines.
!client	A line with an exclamation point (!) followed by a client hostname or IP address (no spaces) will deny access to **all** users on that client, regardless of subsequent lines.
+user	A line with a plus sign (+) followed by a user name (no spaces) will grant access to that user from any client, as long as the client is not deny access by some other line. See below for an explanation of the source of the user name and how it is validated.
-user	A line with a minus sign (-) followed by a user name (no spaces) will deny access to that user from any client. See below for an explanation of the source of the user name and how it is validated.
+user@client	A line with a plus sign (+) followed by a user name, at-sign (@), and a client hostname or IP address (no spaces) will grant access to that user on that client only.
-user@client	A line with a minus sign (-) followed by a user name, at-sign (@), and a client hostname or IP address (no spaces) will deny access to that user on that client only.

When specifying the **client** on the lines in the Security File, you may use either the TCP/IP hostname or an IP address (in dotted-decimal format). If you use a TCP/IP hostname, that name must be resolvable by TCP/IP, either through the *hosts* file or through DNS.

You may use wildcard characters when specifying the user and/or client name in the Security File. The wildcard characters that can be used are:

*****	Matches multiple characters. Example: *****.aol.com
?	Matches a single character. Example: 192.72.124.**??**
[ ]	Matches a list of characters or range of characters. Example: 204.22.6**[5-9]**.*
[!]	Matches characters NOT in a list or range of characters. Example: 204.22.**[!5-9]**?.*

If the connection is coming from a Unix *rsh* or *rcp* command, the **user** name is the Unix login name of the user. If the connection is coming from non-Unix operating system, the method of specifying the user name is determined by the implementation of the *rsh* or *rcp* command you are using.

Keep in mind that the standard *rsh* command allows you to override the user login name with the **-l** option and the standard *rcp* command allows you to use "user@" before the client name to override the user login name. When the user login is overridden using these methods, the standard Unix *rsh* and *rcp* implementations send **both** the actual login name **and** the override name specified on the command line. If you are specifying user names in the Security File, keep this in mind because RSHD/NT will need to validate **both** users.

Using the Security File

To effectively use the Security File, you must first understand how RSHD/NT uses it. When RSHD/NT receives a request, it sequentially processes the lines in the Security File to determine whether or not the client and user are granted or denied access. It looks at each line in the Security File until it determines that either the client or user is specifically denied access.

It begins by assuming that access is denied for the request. It then examines the lines in the Security File to see if any of the lines affect this request. Once it finds a line that denies access to either the user or the client, it stops looking and denies access.

If it finds a line that grants access to the user and/or client, access is tentatively granted, but it continues to process the lines in the Security File to make sure a later line does not deny access.

If it processes the entire Security File and does not find a line that grants access to the user and/or the client, access is denied. If access was tentatively granted at some point and not denied later, access is granted.

For example, let's say that the following is the contents of the Security File:

```
jetty
booey
eib
192.56.42.3
+fred@mars
-gary@booey
-jackie
+robin
*.netcom.com
```

In this example, if any user on the client "jetty" makes a request, access will be granted, unless the user is "jackie", since "jackie" is denied access from all clients (-jackie).

If "jackie@jetty" makes a request, RSHD/NT reads through the Security File and finds the client name "jetty", and tentatively grants access. However, it continues reading the file and finds that the user "jackie" is denied from all clients (-jackie), so access is denied.

Also, if any user on the client "booey" makes a request, they are granted access unless the user is "gary", since "gary@booey" is specifically denied access (-gary@booey). All other users on the client "booey" are granted access except "jackie", since “jackie” is denied on all clients (-jackie).

The user "fred" on the client "mars" is granted access because of the line "+fred@mars". However, since the client "mars" does not appear on a line by itself, no other users on the client "mars" are granted access except the user "robin", who is granted access from *any* client (+robin).

Finally, all users making requests from systems whose client names end in ".netcom.com" are granted access, again, unless the user is "jackie", since "jackie" is denied access from all clients.

Troubleshooting the Security File

If you think you have properly set up the Security File, but you are being denied access when you think you should be granted access, the following provides some tips on determining the cause of the problem:

- Be sure that you have correctly specified the Security File name in the RSHD/NT Control Panel applet and that it actually exists.
- Be sure that you have saved the Security File as an ASCII text file. From a Command Prompt, use the TYPE command to view the contents of the Security File.
- If the editor used automatically appended an extension to filenames (such as Notepad, which appends a .txt), be sure that you have specified this extension on the filename specified in the Security File field.
- Be sure that the Security File is stored on a local hard drive, not a network drive.

- If you are running the RSHD/NT service as a non-LocalSystem user, be sure that the user has permission to read the Security File.
- If you have specified client names in the Security File, be sure that your system can resolve the names, either through DNS or through the HOSTS file. Try using the **ping** command to verify access to the host names specified in the Security File.
- If the client is using the **–l** option in the *rsh* command or the *user@* construct in the *rcp* command, keep in mind that the standard *rsh* and *rcp* commands send **both** the actual login name and the user specified on the command line, so both users will need to be validated by the Security File.

If these tips do not help, enter a filename in the Message File field and a **4** in the Message Level field in the RSHD/NT Control Panel applet. Then execute *rsh* or *rcp* from a system that should be given access, then examine the Message File. It will contain a trace of the steps RSHD/NT is using to validate the remote user and client.

RSHD/NT and Windows Security

Under Unix, the Remote Shell Daemon (rshd) implements security through *user equivalence*. That is, it is assumed that when the user “john” on a client system executes an rcp or rsh command, the user “john” exists on the Unix system and has the appropriate privileges.

Under Unix, the Remote Shell Daemon (*rshd*) runs as the super-user, *root*. Unix has a feature known as “set user id” or “setuid”. When you execute an *rsh* or rcp, the Unix *rshd* effectively “becomes” the user passed to it by *rcp* or *rsh*. Any process spawned (*rsh*) or files accessed/created (*rcp*) are done so as if that remote user were logged on to the system, taking on the security attributes of that user.

Windows has a similar capability (called “impersonation”). However, there is a basic, but important difference. In order for a process like RSHD/NT to “become” another user, that user’s password **must** be supplied. This is incompatible with *rcp* and *rsh* **-** they do not ask for a password. So it is not possible for RSHD/NT to mirror this feature of Unix’s *rshd*.

When RSHD/NT is installed, a Windows service is created and is configured to run RSHD/NT as the user **System** (also known as **LocalSystem** or the **System Account**), which is a built-in account with local Administrator rights. Any processes created or files accessed by RSHD/NT will be executed as the user **System**. This means that programs run through *rsh* will have unrestricted access to the local Windows system and *rcp* will have unrestricted access to the files on the local system.

Although the default **System** account has Administrator rights (meaning it has access to all resources on the system where RSHD/NT is running), the **System** account does **not** have access to any network resources, such as shared drives or printers. This is a Windows security feature. If you attempt to run a program through *rsh* that needs to access a network resource, such as a shared drive, or you need to *rcp* a file on a shared drive and the service is running as **System**, you will receive “access denied” or similar errors.

If you do not want RSHD/NT to run all commands as the **System** user, either for security purposes or because programs needs to access network resources, you have the following options:

1) Use the User Equivalence table in the RSHD Control Panel applet. This allows you to assign a Windows user and password to remote users. This option most closely reflects the way the Unix *rshd* operates.
2) You can specify a single Windows user and password for both *rsh* and *rcp* commands, and **all** commands will be executed as that user, regardless of the remote user.
3) You can change the service startup (using the Services Control Panel applet) and run the RSHD/NT service as a specific user.

Option 1:

You can specify a Windows user and password for each remote user. As explained above, Windows does not have a capability similar to that of Unix, where it can impersonate another user without that user's password. Since *rsh* and *rcp* do not ask for passwords, RSHD/NT somehow needs to find the user's password.

This is done with the User Equivalence table, found in the RSHD Control Panel applet on the User Equivalence tab. You specify a remote user, either just the user's remote login name or the user's remote login name and client hostname (to restrict the usage of the account to just one client). Then you specify and equivalent Windows user and password for that remote user.

Whenever that remote user executes an *rsh* or *rcp* command to this Windows system, RSHD/NT will execute it as the user specified in the User Equivalence table.

Keep in mind that if the Windows user's password expires or changes, you will need to manually update the User Equivalence table with the new password. Windows will not update it for you.

See the section on Winsock RSHD/NT Configuration for more details on setting up the User Equivalence table.

Option 2:

You can run **all** commands from *rsh* as a single Windows user, regardless of the remote user login, by specifying a Windows user and password in the RSHD/NT Control Panel in the option labeled **Execute All RSH Commands as User/Password**, found on the RSH Options tab.

You can access **all** files referenced through *rcp* as a single Windows user, regardless of the remote user login, by specifying a Windows user and password in the RSHD/NT Control Panel in the option labeled **Execute All RCP Copies as User/Password**, found on the RCP Options tab. All files accessed by *rcp* commands will be accessed as that user. If you restrict access to any files or directories for that user, they will also be restricted for all *rcp* commands. If you need access to shared drives on your network through *rcp*, this will also solve the problem, as long as you specify a user who has sufficient privileges to access the necessary shared drives.

Option 3:

You can change the setup of the RSHD/NT service using the Services applet in the Control Panel. You can specify an account other than **System** and RSHD/NT will execute with the security attributes of that user. All *rcp* and *rsh* commands serviced will be restricted to the privileges of that user.

The possible downside to this option is that only the **System** user has access to the Windows desktop. So if you run RSHD/NT as any other user, Windows executes the programs on an invisible desktop so you will not see any windows of programs executed through RSHD/NT. You will not be able to use the special "send keys" option supported in the *rsh* command. The *rsh* command will continue to function properly in all other cases, however. Options 1 and 2 do not have this problem because the service still runs as **System**, but the programs will run in the context of the user specified.

In all cases, you can specify the account name of an existing user. However, this can cause problems because you must also specify that user's password. If that user's password expires or changes, you must remember to manually change it in the setup at that time or RSHD/NT will not start. Windows will **not** update the Service setup automatically.

An alternative is to set up a new account(s) specifically for the purpose of running RSHD/NT. Specify that the account's password does not expire so it will not be forced to change, and that the account's password cannot change.

Using Mapped Drives and Printers through RSHD/NT

If you need to access a network resource (such as a shared drive or printer on another system) through RSHD/NT, you **must** follow the steps in the previous section RSHD/NT and Windows Security so *rsh* or *rcp* executes in the security context of a user who has access to those resources. By default, RSHD/NT runs as the **System** user, who does not have access to any network resources.

If you follow the steps in that section, you will then be able to access network resources using their **UNC** (Uniform Naming Convention) names. For example, if there is a directory shared as "CDRIVE" on the system named "SERVER2", you can access it using the path name \\SERVER2\CDRIVE. You do not need to map a drive letter to it; you can access it directly (assuming proper permissions) using its UNC name. For example, to get a directory listing of that directory, you could use:

```
rsh server dir \\server2\cdrive
```

This command would access \\SERVER2\CDRIVE (case is not significant) through RSHD/NT running on the system "server".

Ideally, you should **always** use UNC path names, but there may be times when it is necessary for you to map a drive letter or LPT port to a network resource through RSHD/NT. If mapped drives or printers are required, there are a few special considerations involved.

- Initially RSHD/NT will not know of any drive or printer mappings. When a service starts or logs in as another user (i.e. using the User Equivalence table or the RSH User or RCP User), Windows does not execute any login scripts or restore any persistent drive mappings.

- With Windows NT and Windows 2000, drive mappings are ***global*** to the system, but are accessible *only* by the login session that created them. This is **very** important to understand. Although Windows functions as a network server, it is not a multi-user operating system. If the interactive user (logged in using the keyboard attached to the Windows system) maps the drive letter F: to a network share, F: will only be accessible from that keyboard by that user. Other processes (such as those created through RSHD/NT or any other service) will see that F: is a mapped drive, but F: will not be usable from those processes; Windows will give an Access Denied error if you try to use it (*** -** see note below). You also cannot map F: to another network drive while it is in use by the interactive user (or another process).

  The above is not true for Windows XP, Windows 2003, and later Windows versions. Beginning with Windows XP, each login session receives its own set of mapped drive letters. While the *rsh* command's login session still will not be able to see the mapped drives of the interactive user, it will be able to map the same drive letters as the interactive user without any conflicts and may even map those drive letters to different locations.

  *** -** this is true for drives mapped to other Windows systems. It may not be true for drives mapped to other types of operating systems, such as Novell or Unix (through NFS or Samba). Depending on how those servers are set up, they may allow you to access mapped drives that have been mapped in other login sessions. It is also not true for drives created with the SUBST command, **if** the SUBST command was executed by a user with Administrator rights.

- You **cannot** use *rsh* to issue single NET USE commands to map drive letters. If you try to do so, you will *orphan* the drive letter and it will be unusable by any process until you reboot the system.

Windows tracks drive mappings by logon session. When you issue an *rsh* command, RSHD/NT creates a new individual logon session for *each* command. If you use *rsh* to execute a NET USE command to map a drive letter, the map will be created for that login session (which exists only for the duration of that command). When the command finishes, the login session will be ended but the drive will remain mapped. Since the login session that created it no longer exists, the drive letter will become unusable. Also, because the login session no longer exists, the map cannot be deleted because only the session that created it can delete it. So it becomes an "orphan" and unusable until you reboot.

You can use NET USE commands in batch files executed through RSHD/NT as long as you remember to delete the map(s) at the end of the batch file. For example:

```
REM MYTEST.BAT
NET USE F: \\SERVER2\CDRIVE
DIR F:
NET USE F: /DELETE
```

This is OK because it creates the map, uses it, then deletes it. Since the batch file is executed with a single *rsh* command, it is executed within a single logon session so the drive letter will be usable throughout the batch file. But if you omitted the NET USE F: /DELETE command, F: would become orphaned and unusable when the batch file ended. The next time you ran the batch file, it would fail.

NOTE: The above batch file could also fail for two other reasons, if running on Windows NT or 2000:

- The interactive user on the system mapped drive F: to some network resource (even the same resource as above). Again, since the interactive logon session "owns" F:, it cannot be used in another logon session.
- Two (or more) *rsh* commands were executed simultaneously to execute this batch file or some other batch file that mapped F:. The first one executed would succeed, but the others would fail because the first execution would "own" the F: drive until it deleted the map.

Again, these restrictions do not apply to Windows XP or higher because XP gives each login session its own set of drive letters to map and will not conflict with other login sessions.

Using AUTOMAP.INI to Automatically Map Drives/Printers

If you must use drive letters and LPT ports to access network resources through RSHD/NT and you cannot use UNC path names, your only options are:

- Do so in a batch file. At the beginning of the batch file, issue the necessary NET USE commands to map the drives/printers, execute the command(s), then issue NET USE commands to delete the maps. This can be inconvenient because you would need to create a batch file for every different command you wanted to execute.
- Use the RSHD/NT **AUTOMAP.INI** file, explained in this section.

The AUTOMAP.INI file allows you to set up drive and printer mappings for use in RSHD/NT. For each *rsh* and *rcp* command, RSHD/NT will read this file and map the specified drives and printers, execute the command, then automatically unmap the drives/printers for you so they are usable by the next command and so they do not become orphaned.

You can map drives to other systems or you can create “symbolic links” to directories on the local hard drive using the “subst” command.

The AUTOMAP.INI file is a text file that you can create with a text editor (such as Notepad or EDIT). It **must** be stored in the RSHD/NT Installation Directory (usually C:\WRSHDNT **-** check the RSHD Control Panel applet to be sure; look on the Advanced tab).

In the AUTOMAP.INI file, you can specify NET USE and SUBST commands to establish drive and printer mappings. The syntax of the NET USE command allowed in AUTOMAP.INI file is:

```
NET USE d: \\system\resource [password] [/USER:user]
```

`d:`	The drive letter or LPT port (i.e. LPT2:) to be mapped. Specify "*" if you only want to connect to the resource, but not map a drive letter.
`\\system\resource`	The UNC path name of the network resource
`password`	An optional password, if required to access the resource
`/USER:user`	An optional parameter that allows you to specify the user account to use when accessing the resource. If not specified, the current user account is used. That is, the use specified in the RSH User or RCP User fields in the RSHD Control Panel applet or the user specified in the service startup.

This syntax is the basically the same as the standard NET USE syntax, except:

- you may omit the password parameter when using /USER: if the password is not necessary
- the password can be placed after /USER: instead of before it if you wish

The syntax of the SUBST command allowed in AUTOMAP.INI file is:

SUBST d: x:\directory

`d:`	The drive letter to be created.
`x:\directory`	The drive and directory where d: is to point. You can also use a standard Windows device name, such as \device\cdrom0. It **must** be a local resource. If the name contains spaces, it must be enclosed in double quotes.

There are some special considerations when using SUBST that are explained later.

So, if you placed the following lines in AUTOMAP.INI:

```
NET USE F: \\SERVER2\CDRIVE
NET USE M: \\WESTCOAST\ACCT clancy2 /USER:tom
NET USE LPT4: \\PRSERVER\PRT1
SUBST W: C:\WINDOWS
SUBST X: “D:\My Documents”
```

For every *rsh* and *rcp* command, RSHD/NT would map F: to \\SERVER2\CDRIVE, M: to \\WESTCOAST\ACCT (using the user "tom" and the password "clancy2"), and LPT4: to \\PRSERVER\PRT1. It would create drive W: to point to C:\WINDOWS and X: to point to D:\My Documents.

It would then execute the command, then unmap the drives and printers automatically after the command ends. It will do this regardless of whether the command actually uses the mapped drives or printers, since RSHD/NT has no way of knowing whether it will or not.

RSHD/NT will **not** report a failure if it cannot create any of the maps. Your command may fail because the map is unavailable or inaccessible and that may cause an error to be reported back to you, but the mapping failure itself will not generate an error.

If RSHD/NT is running on Windows NT or Windows 2000, the map would fail if another process already has the drive letter or printer port name in use.

When using SUBST with Windows NT/2000, there is an important consideration – Windows allows a user with Administrator rights to **remap** a drive already created with SUBST. If the interactive user executes a SUBST to point drive X: (for example) to C:\DIR1 and the AUTOMAP.INI file is set up to SUBST drive X: to C:\DIR2, and RSHD/NT is executing rsh commands as a user *with Administrator rights*, the AUTOMAP.INI file will cause drive X: to point to C:\DIR2 for **both** the rsh command **and** the interactive user, since drive letters are global to the entire system under Windows NT/2000. This does not occur if the effective user does not have Administrator rights – a user without Administrator rights cannot remap a drive already created with SUBST – the subst command in AUTOMAP.INI would fail in this case. This does not apply to Windows XP and Windows 2003, since each login session receives their own set of drive letters. Commands executed through RSHD/NT would not see the drives created with SUBST by the interactive user.

You can also create "sections" in AUTOMAP.INI that allows you to specify different mapping schemes for different users and clients. Any maps that appear at the beginning of the file that are not under a section heading get mapped for *every* user from *every* client.

You can specify sections in the AUTOMAP.INI file for specific users, specific clients, or specific users on a certain client. You do this by enclosing the specification in square brackets ([ ]) and including the maps under it. The options are:

[+user]	- The maps in this section are only executed for "user"
[hostname]	- The maps in this section are only executed for *rsh/rcp* commands from "hostname"
[user@host]	- The maps in this section are only executed for *rsh/rcp* commands from user@host

The "user" referred to above is the user received from the client in the *rsh/rcp* command. This may be the logged on user on the client or it may be overridden with the **-**l option in *rsh* or the *user@host:* specification in the *rcp* command.

You may use wildcard characters (*, ?) if necessary and you may use IP addresses instead of the host name if you wish. Some examples:

```
[+john]
[myhost.com]
[10.0.1.*]
[alice@brady.com]
```

You can specify comments in the AUTOMAP.INI file by using "rem", "#", or ";" at the beginning of a line.

The following is an example of a slightly more complex AUTOMAP.INI file:

```
# These maps are for all users
NET USE F: \\SERVER1\C
NET USE G: \\SERVER1\CDROM

[10.0.1.*]
# These maps are only for users with local IP addresses
NET USE H: \\SERVER2\SECRET
NET USE LPT4: \\SERVER1\HPLASER
SUBST X: “C:\MY DOCUMENTS”

[honcho@oursys]
# Only the head honcho can access this
NET USE M: \\MGMT\DATA
```

NOTE: Although the NET USE and SUBST commands are standard Windows commands, AUTOMAP.INI is **not** a batch file. The NET USE/SUBST syntax is only used for the purpose of familiarity. RSHD/NT is reading and interpreting lines in the AUTOMAP.INI file **-** it is not actually executing NET USE and SUBST commands per se. So you cannot include any other commands besides NET USE or SUBST in AUTOMAP.INI and you cannot use any other options to NET USE or SUBST other than those shown here. If you try, the lines will be ignored.

Keep in mind that this is not a complete solution to the problem of using mappings. With Windows NT and 2000, you still must somehow deal with the fact that Windows is **not** a multi-user system and if another process uses a drive letter, it will not be usable by any other process until the process that created it deletes the map. There is nothing that RSHD/NT can do about this **-** it is the way Windows NT/2000 works.

EXECUTING COMMANDS

The *rsh* command executes the specified program on a remote server and returns the standard output and the standard error output to the *rsh* command. You can then see the output of the programs on your screen or it can be redirected to a file.

When you execute Windows console or MS-DOS programs using *rsh* through RSHD/NT, you will also see the standard output and standard error output of those programs. Most Windows console and MS-DOS programs are character-based and write output to the standard output/error, which RSHD/NT can capture and return to the *rsh* command.

For example, the following command would display a directory of the C: drive’s root directory on your screen (assuming RSHD/NT’s default configuration is being used):

```
rsh server dir c:\\
```

Note the double backslashes (\\). If this *rsh* command is being issued from a Unix system, Unix uses the backslash as an “escape” character. To send a single backslash to RSHD/NT, you must specify two. This would not be true if you were executing the *rsh* command from another Windows system.

With Windows graphical programs, there is no such thing as "standard output" and "standard error". Programs execute in graphical windows, so there is no way to return any output using *rsh*. Additionally, there are also some console and MS-DOS programs that write directly to the screen and their output cannot be returned either.

Therefore, when using *rsh* to run graphical programs on a Windows system, you will not see any output of those programs on your screen (where the *rsh* command was used). If the RSHD/NT service is running in the

System Account and the RSHD/NT configuration is not set to run programs in hidden windows, the window of the graphical program will display on the screen attached to the Windows system.

For example, if you used the following command:

```
rsh server excel
```

This would start Excel on the system named "server" (assuming ”excel.exe” was in the PATH). You would see nothing on the screen where the *rsh* command was run as a result of starting Excel. Excel would be running on the screen of the system named "server".

Standard Input/Output/Error Redirection

You can optionally capture the standard output and standard error output of Windows Console or MS-DOS programs through RSHD/NT with the *rsh* command. This allows you to display the output of these programs that output to the standard output or standard error on another screen or capture it to a file on another system. It also allows you to send the standard input from *rsh* to the remote program. This is known as *redirection*.

By default, RSHD/NT automatically attempts redirection on every program executed using *rsh*, to capture the standard output and standard error and send it back to the *rsh* command, and receive standard input sent to it by the *rsh* command and provide it to the program executed.

If you mostly execute Console or MS-DOS programs through RSHD/NT, this default behavior of automatic redirection is the most useful. But it does have the side effect of causing the *rsh* command to wait until the program ends.

If you do not want the *rsh* command to wait until the program ends on the remote server, you have two options:

Option 1. If you only occasionally need to use *rsh* to execute graphical programs or console/DOS programs where you do not want *rsh* to wait until they complete, you can add the special “<[WIN]>” directive to your *rsh* command. This tells RSHD/NT that it should not attempt redirection on this program only. For example:

```
rsh server "<[WIN]>" excel
```

This will start Excel on the system named “server” and RSHD/NT will not attempt any redirection. The *rsh* command will end immediately after Excel starts; it will not wait until Excel exits. Without the “<[WIN]>”, the *rsh* command would not end until you closed Excel.

Option 2. If you usually do not want to have *rsh* wait for programs to end, you can uncheck the option labeled **Attempt Redirection on Every Command** in the RSHD/NT Control Panel applet, on the RSH Options tab. If you uncheck this option, RSHD/NT will not automatically attempt redirection, unless you specify the special “<[CON]>” directive is used in the *rsh* command. This tells RSHD/NT that it should attempt redirection on this program only. For example:

```
rsh server "<[CON]>" net view
```

This will run the "net view" command on "server" and display the output on your screen. The "net view" command displays information on the standard output. The “<[CON]>” directive tells RSHD/NT that “net view” is a console program and you want to capture its output.

If you simply want the *rsh* command to wait until the program finishes executing, but you do not care about capturing its output, you can use the special "<[WAIT]>" directive in the *rsh* command. For example, to execute the command "bkgprint" and wait for it to finish, use:

```
        rsh server "<[WAIT]>" bkgprint
```

This will wait for “bkgprint” to finish, but nothing will be redirected back to the *rsh* command. This saves a little overhead on the remote server running RSHD/NT, since no redirection is necessary.

Specifying the Window Type

RSHD/NT also allows you to specify the window type of the application being run. Normally, the application is run based on the **Default Window Type** specified in the RSHD/NT Control Panel applet.

If you want to specify a different method of displaying the application's window, you can use a special directive on the command line, starting with “<[“ and ending with “]>” **-** for example, “<[MIN]>”. The directive goes before the command, like this:

```
        rsh server "<[MIN]>" command
```

There are two methods of setting the window type. You can use one of the words shown below or you can use a number. The options are:

Window Option	**Displays**
NORMAL or NORM	Normal Display as defined by the application
MINIMIZE or MIN	Shows the application as a minimized icon without focus
MINACTIVE or MINA	Shows the application as a minimized icon with focus
MAXIMIZE or MAX	Maximizes the application on startup
HIDE	Hides the application (no icon appears)
0	Same as HIDE
1	Same as NORMAL
7	Same as MINIMIZE
2	Same as MINACTIVE
3	Same as MAXIMIZE

For example, if you wanted to run some application that does some task and exits, you could run it minimized using:

```
        rsh server "<[MINIMIZE]>" bkgprint
```

The quotes are normally required to prevent command shells from interpreting the characters < and > locally.

If you want to run the Windows Notepad maximized, viewing the file "heyyou.txt" using keystrokes, you would type:

```
        rsh server "<[MAXIMIZE]%FOheyyou.txt~>" notepad
```

This runs the Notepad maximized, then "presses" Alt-F-O (File Open) and types the filename "heyyou.txt" and presses Enter to load it. Keystrokes are explained in more detail in the next section. This command illustrates how you can combine a window type directive with keystrokes.

Note that Windows **does not allow you to send keystrokes to a minimized or hidden application.** Therefore, "[MINIMIZE]", "[HIDE]", "[0]", or "[2]" should always appear alone between the "<" and ">". If you specify other keystrokes, the application will not receive them (Windows will beep at you for each keystroke).

Sending Keystrokes

RSHD/NT provides the ability for you to send keystrokes to a graphical Windows application you run using the *rsh* command. This provides you with some "remote control" over what the program you run does once it starts.

If you are running the RSHD/NT service as a user other than the default **System** account, you will not be able to use this function. This is a feature of the Windows security system – only the System user has access to the desktop (screen). When the RSHD/NT service is run as a non-System user, Windows creates the program's windows on an invisible desktop and they cannot receive keystrokes. A better option is to use the User Equivalency table or the "Execute All RSH Commands as User" option in the RSHD Control Panel applet and continue running the service in the System account. This will allow you to use keystrokes.

Also, due to a limitation in Windows, you **cannot** send keystrokes to an application started through RSHD/NT **if there is no one logged in to Windows**. The program will be executed, but it will **not** receive the keystrokes. This is again due to the Windows security system. When Windows is at a login, the login window is displayed on a special desktop.

If you are familiar with Microsoft Visual Basic or the Visual Basic for Applications (VBA) macro language, sending keystrokes s is very similar to the ***SendKeys*** capability of those programming languages.

The standard syntax of the *rsh* command is:

```
rsh hostname command
```

This will execute "command" on the server "hostname". RSHD/NT allows a slight modification of the *rsh* syntax to send keystrokes. This is compatible with all *rsh* commands. The alternative syntax for sending keystrokes is:

```
rsh hostname "<keystrokes>" command
```

If the first parameter after the host name begins with a less-than sign (<), that parameter is interpreted as keystrokes to be sent to the command specified in the next parameter. The keystrokes must end with a greater-than sign (>). You **must** also enclose the entire parameter in **quotes** so special characters and spaces are not interpreted by the operating system.

For example, if you wanted to run the Windows Notepad on the system named "server" and type "This is a test" on the first line, the command would be:

```
rsh server "<This is a test>" notepad
```

If you looked at the server's screen, you would see the Windows Notepad with "This is a test" on the first line.

You cannot send keystrokes to all applications. This is beyond Denicomp Systems control. Most applications will allow you to send them keystrokes, but there are a few that will not.

Sending Special Keystrokes

RSHD/NT also allows you to specify special keys in the keystrokes parameter that cannot normally be typed on a command line, such as embedded Enter keys, function keys, ALT keys, etc.

Keystrokes are sent sequentially as they appear between the "<" and ">". To send a single character, use the character itself. For example, to send the letter "X", use the letter "X". To send the word "hello", just specify those letters.

To specify keys combined with any combination of Shift, Ctrl, and Alt keys, prefix the regular key code to one or more of the following codes:

Shift	+
Control	^
Alt	**%**

For example, to send the Alt-F keystroke, specify "**%F**". To send Ctrl-Alt-D, specify **"^%D**".

To send the Enter key, use the **tilde** (~).

To specify that the Shift, Ctrl, and/or Alt keys should be held down while several other keys are pressed, enclose the key codes in parentheses ( ). For example, to have the Alt key held down while X and D are pressed, use "**%(XD)**". You could also use "**%X%D**", but if the Shift, Ctrl, and/or Alt keys need to be held down for a number of keystrokes, the parentheses can make the string shorter. Also, you would want to use the parentheses if the application detects the release of the Shift, Ctrl, and/or Alt keys and that is not desired.

The following characters have special meaning in the keystroke parameter, so they must be enclosed inside **braces** (**{ }**). Some of these special characters have not been explained yet.

Special Character	**Example**	**Special Character**	**Example**
+ (plus)	{+}	^ (caret)	{^}
% (percent)	{%}	~ (tilde)	{~}
< (less than)	{<}	> (greater than)	{>}
[(left sq. bracket)	{[}	] (right sq. bracket)	{]}
((left paren)	{(}	) (right paren)	{)}
{ (left brace)	{{}	} (right brace)	{}}
@ (at-sign)	{@}		

To send characters that are not normally displayed when you press a key (such as Enter or Tab) and keys that represent actions rather than characters, use the following special codes:

Key	**Code**	**Key**	**Code**
Backspace	{BACKSPACE} or {BS}	Break	{BREAK}
Caps Lock	{CAPSLOCK}	Clear	{CLEAR}
Del	{DELETE} or {DEL}	Down Arrow	{DOWN}
End	{END}	Enter	{ENTER} or ~
Esc	{ESCAPE} or {ESC}	Help	{HELP}
Home	{HOME}	Ins	{INSERT}
Left Arrow	{LEFT}	Num Lock	{NUMLOCK}
Page Down	{PGDN}	Page Up	{PGUP}
Print Screen	{PRTSC}	Right Arrow	{RIGHT}
Scroll Lock	{SCROLLLOCK}	Tab	{TAB}
Up Arrow	{UP}	F1 through F16	{F1} through {F16}

You can also specify that a key is to repeat itself a certain number of times, without repeating the key itself in the string. To repeat a keystroke, use the format:

{keystroke number}

Where "keystroke" is the key to repeat, followed by a single space, then the number of times to repeat the key.

For example, to press the down arrow key eight times, use "**{DOWN 8}**".

To type thirty asterisks (*), use "**{* 30}**".

Pausing Within Keystrokes

Under some circumstances, it may be necessary to pause for a specific time before sending keystrokes to allow a program operation to complete. This is usually necessary when a program ignores keystrokes that have been queued while a long operation takes place (perhaps generating print output, rewinding a tape, performing a complex calculation, etc.).

Within the keystroke list, you can specify pauses by using the special **{PAUSE #}** keystroke.

This is not actually a keystroke, in that it does not press any key, but it can be included anywhere within the keystroke list. It will pause the specified number of seconds.

For example, the following keystroke list will press Alt-F, P, wait 10 seconds, then press Alt-F, X:

<%FP{PAUSE 10}%FX>

You can specify multiple pauses in the keystroke list if necessary.

Keystroke Example

The following example, will start Microsoft Word, load a file, print it, then exit.

```
rsh server "<%FO\docs\invoice.doc~%FP~%FX>" word
```

The keystrokes are:

Keystroke String	**Sent**	**Description**
`%F`	Alt-F	Drops down the file menu
`O`	O	Selects Open
`\docs\invoice.doc`	`\docs\invoice.doc`	Types the filename.
`~`	Enter	Loads the File
`%F`	Alt-F	Drops down the file menu
`P`	P	Selects Print
`~`	Enter	Accepts the defaults on the Print dialog box
`%F`	Alt-F	Drops down the file menu
`X`	X	Selects eXit and Word exits

Note that if this example were being run from a Unix system, you would have to use two backslashes (\\) for every one desired backslash because the Unix shells interpret the backslashes as special characters. The command would then be:

```
rsh server "<%FO\\docs\\invoice.doc~%FP~%FX>" word
```

Keystroke Macro Files

If your keystroke strings get rather long or complex, you can store them in a keystroke ***macro file*** so you do not have to specify all of them each time you use the *rsh* command.

To create a keystroke macro file, you must use a text editor (or a word processor, but be sure to save as an ASCII file). Enter the keystrokes as you would on the *rsh* command line, with the following exceptions/reminders:

- Do not enter "<" as the first character in the file or ">" as the last character. All of the characters you enter in the file will be sent.
- You may press Enter in the file to enter the keystrokes on multiple lines. The line breaks have **no effect** on the keystrokes. They will be treated as if they were entered all on the **same line**. That is, you must remember to still use "~" or "**{ENTER}**" to "press" the Enter key. Pressing Enter in the file will **not** send the Enter key.
- You **cannot** nest keystroke macros. Your macro file **cannot** contain references to other keystroke macro files.
- The keystroke macro file **must** reside on the system running RSHD/NT. You can create the file on that system or use *rcp* to copy it to that system before executing the command.

To use a keystroke macro file, enter the **at-sign** (**@**) followed by the filename in braces (**{ }**) where you would normally specify keystrokes on the *rsh* command line.

You will most likely need to specify a full pathname of the keystroke file on the system running RSHD/NT, unless you know the working directory of RSHD/NT on the system running it and the keystroke macro resides in that directory. You may use forward slashes (/) instead of backslashes if you wish; this makes life easier for Unix users because the shell interprets the backslash characters.

For example, if you had a macro in the directory \kbmac\printss.mac on the system running RSHD/NT, you could use it with this command:

```
rsh server "<@{/kbmac/printss.mac}>" excel
```

This would run "excel" on server and send the keystrokes stored in the file \kbmac\printss.mac to it.
You can intermix command line keystrokes and macro file keystrokes. That is, you can specify some of the keystrokes on the command line and use some from a macro file. You can also use multiple macro files.

For example, let's say we want to print a file using *rsh* through a Windows application called "wintiff". We want to store the keystrokes in a macro file, but do not want to store the filename in the macro file because it can change.

To do this you can store the first set of keystrokes in one macro file, specify the filename on the *rsh* command line, then store the remaining keystrokes in a second file.

For example, let's say the file is "mypic.tif":

```
rcp mypic.tif server:/tmp
rsh server “<@{/kb/tif1.mac}\tmp\mypic.tif~@{/kb/tif2.mac}" wintiff
```

This example copies the file "mypic.tif" to the \tmp directory on server. Then it runs "wintiff" and first sends the keystrokes from the file \kb\tif1.mac. That macro ends when "wintiff" requires a filename. The keystrokes to "type" the filename come from the *rsh* command line since the tif1.mac has ended. Then it continues by sending the keystrokes in the file \kb\tif2.mac. That is:

`@{/kb/tif1.mac}`	Send keystrokes from \kb\tif1.mac
`\tmp\mypic.tif~`	Type \tmp\mypic.tif and press Enter
`@{/kb/tif2.mac}`	Send keystrokes from \kb\tif2.mac

Shutting Down and Rebooting Windows through RSHD/NT

RSHD/NT internally understands two special commands that allow you to remotely shutdown and reboot the system through the *rsh* command. These commands require the special *<[INTERNAL]>* or *<[INT]>* directives. The commands have the following syntax:

```
rsh server "<[INT]>" shutdown [#]

rsh server "<[INT]>" reboot [#]
```

The **[#]** are optional. You can specify a number of seconds to delay the shutdown or reboot in this parameter. If you do not specify a number of seconds, it is immediate.

If you are running the RSHD/NT service as a user other than **System,** you are using the User Equivalency table, or you have specified a user in the **Execute All RSH Commands as User** field, that user must have permission to shutdown or reboot the system.

For example:

```
rsh server "<[INT]>" shutdown 60    (Shutdown server in 1 minute)
```

COPYING FILES USING RCP

RSHD/NT also provides Remote Copy (RCP) Server capability. This allows you to copy files to and from a system running RSHD/NT using the *rcp* command.

The *rcp* command is commonly found on Unix systems and Windows NT/2000 and higher systems. Windows 95/98/ME do not include the *rcp* command; instead, you can use Winsock RCP/RSH/REXEC from Denicomp Systems if you require *rcp* under one of those operating systems.

The *rcp* command is described in more detail in your operating system manual (on Unix, type “man rcp”) or with the manual that comes with Winsock RCP/RSH/REXEC. However, here are a few examples of its use.

Important Note: Unlike the standard Unix *rcp* command and Denicomp Systems’ *rcp* command (found in the Winsock RCP/RSH/REXEC package), the Microsoft-supplied, native Windows *rcp* command copies all files with **ASCII end-of-line conversion by default**. Binary files **must** be copied using the **-b** option of the native *rcp* command. If you do not use the **-b** option on binary files, the contents of the file will be altered on the destination system.

To copy a file from the host named "server" to the current system from the system named “server”, use:

```
rcp server:yourfile myfile
```

The file "yourfile" is copied from the host named "server" to the file on your PC named "myfile". The host "server" could be either a Windows system running RSHD/NT or Unix. The syntax is the same.

To copy a file from your PC or Unix system to the system named "server", use:

```
rcp \lists\xmas.doc server:/word/lists
```

The file \lists\xmas.doc is copied from your system to the file xmas.doc in the directory /word/lists on the system named "server", which could be either a Windows system running RSHD/NT or a Unix system. With RSHD/NT, it does not matter whether you use the slash (/) or the backslash (\) after the colon (:) in the parameter where the hostname is specified. RSHD/NT will handle it properly.

To send the entire directory tree to "server", use:

```
rcp -b -r \share server:/
```

All of the files and subdirectories in the directory \share are copied to the system named "server". It will create a \share directory in the root directory (\) of server. This example assumes you are using the native Windows *rcp* command because the –b option is used to specify a binary copy, so files are not modified.

If the \share directory contained any subdirectories, they would be created on the other system and all the files in them would also be copied.

To copy all of the files ending with ".xls" from "server" to your PC, use:

```
rcp -b server:\sheets\*.xls .
```

This copies all of the files ending with ".xls" in the directory \sheets on "server" to the current directory (.) on your PC.

You can use drive letters if necessary. For example, to copy a file from the A: drive on the "server" to your PC:

```
rcp server:a:file.txt file.txt
```

This will copy "file.txt" from the A: drive on "srvpc" to the file "file.txt" on your system.

End-of-Line Conversion

RSHD/NT provides a method for you to have end-of-line characters automatically converted in files copied with ***rcp***. Unix systems use a single newline (ASCII 10) character as a line delimiter. Windows systems use carriage return (ASCII 13) and newline pairs for line delimiters.

When end-of-line conversion is enabled, RSHD/NT will convert all carriage return/newline pairs to single newlines when files are copied from the Windows system. It will add a carriage return to every newline character when files are copied to the Windows system.

RSHD/NT can automatically convert end-of-line characters in the following ways:

- Convert all files. You would use this option if you **only** copy ASCII files with *rcp*
- Convert based on a list of file extensions that you supply. For example, you can convert only files ending with .txt.

- Convert based on the contents of the first block of data in the file. RSHD/NT looks at the data in the first block of the file and if it contains only printable ASCII characters, it will convert the file. If it finds any non-printable characters, it will not convert the file.

By default, RSHD/NT does **no** end-of-line conversion. The data in files sent or received using *rcp* is not modified.

To perform end-of-line conversions through RSHD/NT, see the section titled Winsock RSHD/NT Configuration for the RCP Option labeled **Automatic End-of-Line Conversion.**

WINSOCK RSHD/NT SERVICE CONTROL

You can control the Winsock RSHD/NT service (start, stop, etc.) using the Services Control Panel applet or using the Service Control tab in the RSHD Control Panel applet.

RSHD/NT also includes another utility that allows you to control the service from the command line, named **ctrlrshd.exe**. This can be found in the RSHD/NT installation directory. Its syntax is:

```
ctrlrshd install [auto | manual [directory]]
ctrlrshd start
ctrlrshd stop
ctrlrshd remove
```

The **install** option installs the RSHD/NT service in the Windows list of services. It does not start RSHD/NT. You can optionally specify whether you want RSHD/NT to start automatically each time Windows is booted (**auto)** or if you want to start RSHD/NT manually when necessary (**manual)**. If you do not specify either, **auto** is assumed. The ***directory*** option is used when you are using the **install** option and RSHD/NT is not installed in the current working directory. If it is not, specify the directory name as the third parameter. Note that if you specify a directory, then you must also specify **auto** or **manual**.

The **start** option starts the RSHD/NT service. RSHD/NT will begin accepting requests.

The **stop** option stops the RSHD/NT service. It will no longer accept requests. This does not remove it from the Windows list of services; it only stops it until you decide to restart it using the **start** option.

The **remove** option removes RSHD/NT from the Windows list of services. It will no longer accept requests and will no longer be available to **start** or **stop.** Note that you must first use the **stop** option if RSHD/NT is running before removing it. The **remove** option does **not** delete any files; it only removes RSHD/NT from the list of available services. You can later add it back to the list using the **install** option.

You can also start and stop the RSHD/NT service with the Windows “net start” and “net stop” commands:

```
net start “Remote Shell Daemon”  or  net stop “Remote Shell Daemon”
```

SUPPORT

Support is available via e-mail and the web.

Internet:	support@denicomp.com
WWW:	http://www.denicomp.com

Also, see the following web page for a list of Frequently Asked Questions: http://www.denicomp.com/faq.htm.