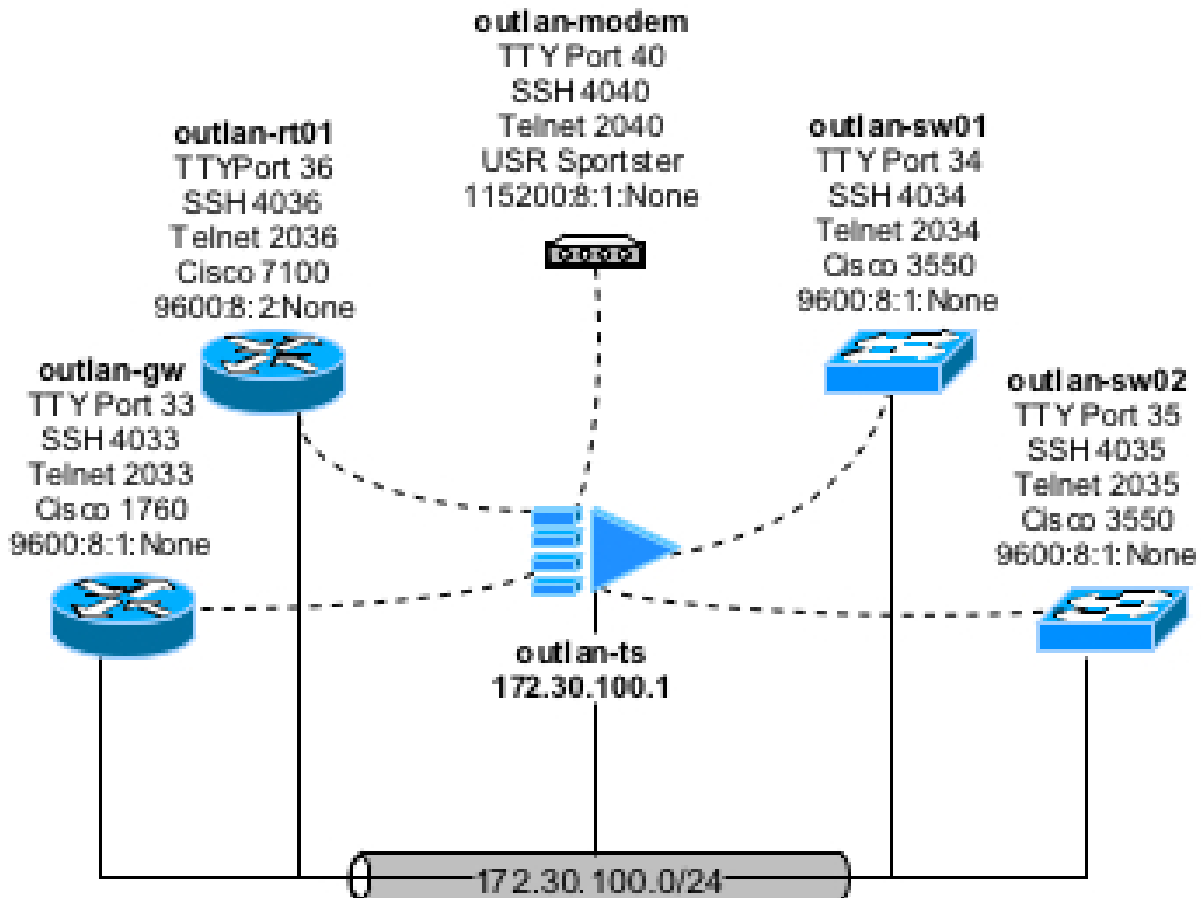


Configuring IOS Terminal Services

Michael J. Martin

This is the fourth and final installment in our series on implementing an asynchronous serial line terminal server using a Cisco 2511 with 16 fixed asynchronous serial ports or 26x0, 36x0 or 37x0 series modular routers using NM-16A or NM-32A. While the actual router configuration is quite straightforward, it is the "care and feeding" of the terminal server that causes the real headaches. To minimize some of this pain, you also need to know about Cisco async serial hardware, TTY management and TTY access. This month's article will focus on the IOS configuration of a Cisco router as a terminal server providing direct asynchronous access to other Cisco routers, along with modem access for dial-out services.

Getting Started



Above is a sample diagram for a Cisco terminal server topology. Before getting started with the terminal server configuration, it is a really good idea to map out the serial-to-rotary definitions and the devices to which they connect, along with the connected devices' line operation characteristics. By taking the time to label and document connections prior to the configuration, you will save time and the frustration of having to go back and trace the connections or figure out that the device on line 45 needs two stopbits to work correctly.

Once you have collected all of the operational parameters and the physical terminations between the terminal server (TS) and connected devices (CD) are in place, there are three configuration elements that need to be handled to get the terminal server up and running:

- Line configuration: This element sets up the behavior of the line, i.e., line speed, privilege-level, supported inbound and outbound access protocols. In addition to physical and administrative

Configuring IOS Terminal Services

Michael J. Martin

behavior, there are a number of informational queues that can be set to aid users' understanding about the device. We will review some of the common options and why they should be implemented.

- AAA services: Configured locally or using a TACACS server, this element defines access authentication and connected device access control and most importantly, server and line accounting. After all, you are providing a great degree of access to critical parts of your network, so you want to control who is using it and keep tabs on what they are doing. We will look at configuring both local and TACACS authentication configuration options.
- Access configuration: This element configures "how" the TS lines and the CD will be accessed. The IOS supports both local (from the terminal server EXEC shell) and remote (direct connection to the service port from a remote host) reverse telnet access. One or both options can be configured; each administrator must determine what best suits her environment. We will look a setting up both

There are two types of line configuration: DTE-to-DTE (terminal server to router/switch) and DTE-to-DCE (terminal server to modem for dial-in or dial-out).

Enabling Line Access

In order to access the device connected to the line, we need to enable either reverse telnet or reverse SSH. This is handled quite simply with the line configuration command `<transport-input {telnet | ssh | rlogin | all | none}>`. Here is a syntax example that enables both:

```
outland-ts(config-line)#transport input telnet ssh
```

I must mention two things about SSH support. First, SSH is only available on IOS versions that support SSH, and it must be enabled on the router. Second, SSH support requires that either AAA TACACS support or "local" username-based authentication is place. The line configuration command `<login>` is used to set the authentication behavior of the line. When AAA is not enabled, the command syntax is `<login {local | tacacs}>`, With AAA enabled, the syntax changes to `<login authentication {default | WORD}>`. And by extension, if no authentication method is defined on the line, the line will behave differently if AAA is or is not enabled. If AAA is enabled, then the "default" authentication scheme will be applied to the line interface. If AAA is not enabled, then no authentication is imposed to access the line. So, once the user establishes a reverse connection to the service port, the user has access to the connected device.

Configuring And Accessing The Line

The IOS terminal server supports two connection models: remote and local. In the remote or direct-connect model, a user can open a reverse telnet session to the IP address of the terminal server network interface or a loopback interface, followed by the line's corresponding rotary line number. For example, to connect to the device attached to line 34 (TTY rotary lines start at port 2000), a user would telnet to 172.30.100.98:2034.

```
Trinity:~ martin$ telnet 172.30.100.98 2034
Trying 172.30.100.98...
Connected to outlan-ts.
Escape character is '^]'.

```

User Access Verification

Username:

Configuring IOS Terminal Services

Michael J. Martin

To sever this connection, the user must terminate his telnet session using the ctrl-] or equivalent break option. This causes the telnet application to send a RST, instead of the standard FIN. While this approach works, it is not the most ideal method. The problem is that with reverse connections, the terminal server is just a pass-through to the connected device. The side effect is that there is no "channel" for the user to terminate the session with the terminal server gracefully.

There are other disadvantages. First, this "forced" session termination often leaves the line in a hung state, making any future connections impossible. When the line is hung, the connected device is locked out. This requires an administrator or user to connect to the terminal server and clear the line using the <clear line {line #}> EXEC command. The second issue, which is an extension of the first, is that since the terminal server functions as a passive proxy for these line sessions, there is no "support any session" control. Each rotary must be connected using a new session. Since no service channel exists between the user and the terminal server, it is not possible within an established session to change to a different line rotary. The third issue, which places the largest burden on the user, is the requirement that she know the rotary line each device she wants to connect to.

The alternative is the local connection model. In this approach, the user establishes a single session to the terminal server, from which sessions to the rotary lines can be established via reverse telnet. Thus, by adding the terminal server into the connection "path," the issues that surround the remote model are eliminated. There are also some added benefits. First, the IOS supports rotary line definitions as part of its IP host table definitions. These entries are defined using the configuration command <ip host {hostname} {port | mx | ns | srv |} {IP address}>. All IP devices have support for a host table, and IOS routers are no exception. A host table provides a way for administrators to define hostname to IP address translations on the local machine as an alternative to using a Domain Name Server (DNS) for IP and hostname resolution. The host table is commonly consulted prior to sending an nslookup request, making it possible for administrators to override DNS definitions and tailor hostname lookups on a per-host basis. On Unix (/etc/hosts) and Windows (C:WindowsSystem32Drivershosts) systems, the format for host file entries is "ip address:

```
hostname.domain-name: alias." Here is an example:
127.0.0.1 Loopback
172.30.100.49 lpr.outlan.net lpr
172.30.100.98 ts.outlan.net outlan-ts
```

IOS host definitions appear just after the AAA portion of the configuration. Here is an IOS host table example with and without a service port definition:

```
ip host outlan-gw 2033 172.30.2.33
ip host outlan-ts 172.30.100.98
```

This additional flexibility in defining host table entries makes connecting to line rotaries far more accommodating to users than opening sessions to specific service ports. All the user needs to know is the hostname of the device to which they want to connect. As this example shows:

```
outlan-ts#telnet outlan-gw
```

```
User Access Verification
Username:
```

In terms of implementation, a number of approaches can be followed. Hostname and line rotary entries can be mapped to the terminal server's network interface IP address. Here is an example host table:

Configuring IOS Terminal Services

Michael J. Martin

```
ip host outlan-gw 2033 172.30.100.98
ip host outlan-sw01 2034 172.30.100.98
ip host outlan-sw02 2035 172.30.100.98
ip host outlan-rt01 2036 172.30.100.98
ip host outlan-modem 2040 172.30.100.98
ip host outlan-ts 172.30.100.98
```

While functional, this has limitations. First, if the interface is down, none of the rotary lines can be reached. Second, line rotary connections will have the same destination IP address in their accounting log entries, limiting filtering and search options.

Hostname and line rotary entries can also be mapped as a single loopback interface IP address. Here is an example host table:

```
ip host outlan-gw 2033 172.30.2.1
ip host outlan-sw01 2034 172.30.2.1
ip host outlan-sw02 2035 172.30.2.1
ip host outlan-rt01 2036 172.30.2.1
ip host outlan-modem 2040 172.30.2.1
ip host outlan-ts 172.30.100.98
```

This approach addresses the issue of access, but still leaves the accounting log entry issue. That leads us to a third approach, which addresses both issues. Hostname and line rotary entries are each mapped to a unique loopback interface IP address. Here is an example host table:

```
ip host outlan-gw 2033 172.30.2.33
ip host outlan-sw01 2034 172.30.2.34
ip host outlan-sw02 2035 172.30.2.35
ip host outlan-rt01 2036 172.30.2.36
ip host outlan-modem 2040 172.30.2.40
ip host outlan-ts 172.30.100.98
```

Since the terminal server will refer to its locally defined host entries, before using DNS, the terminal server's host entries can mimic the legitimate hostnames in DNS, but be remapped locally. In this way, users can use the hostnames they already know to access the rotary lines.

With the access name issue addressed, there still remains the problem of session management. Most users use the exec command <telnet> to open a session with a remote host or rotary line. In addition to telnet, there is also the lesser known exec shell command <connect {ip address | hostname}>. The <connect> and <telnet> commands operate the same. Both are privilege-level 1 commands, and both open VTY sessions with remote hosts. However, if you type the ctrl-] break command, your session to the terminal server is severed, not the session to the remote host you opened from the exec shell. To end your remote connection and keep your session on the terminal server, you need to use the IOS escape sequence ctrl+shift+6, x, which drops you back to the exec shell. From there, you can manage the session using one of the following options:

End the session by typing <disconnect>:

```
outland-ts#disconnect
Closing connection to outlan-gw [confirm]
outland-ts#
```

Configuring IOS Terminal Services

Michael J. Martin

Suspend the session by executing a new exec command and return to the session by sending a <cr> at the exec shell prompt.

```
outland-ts#sh users
```

Line	User	Host(s)	Idle	Location
33 tty 33	martin	idle	00:00:50	outlan-gw
* 66 vty 0	martin	outlan-gw	00:00:02	trinity

Interface	User	Mode	Idle	Peer Address
-----------	------	------	------	--------------

```
outland-ts#
```

```
[Resuming connection 1 to outlan-gw ... ]
```

Suspend the session and open a new remote session using <connect> or <telnet>

```
outland-ts#connect outlan-sw01
```

```
User Access Verification
```

```
Username:
```

Suspend the session and list open sessions using <show sessions>

```
outland-ts#sh sessions
```

Conn	Host	Address	Byte	Idle	Conn Name
1	outlan-gw	172.30.2.33	25	1	outlan-gw
* 2	outlan-sw01	172.30.2.34	0	0	outlan-sw01

```
outland-ts#
```

Terminate a session using the <disconnect> command followed by the session number listed in the Conn field.

```
outland-ts#disco 1
```

```
Closing connection to outlan-gw [confirm]
```

```
outland-ts#
```

Suspend the current session and resume a previous session using the <resume> command followed by the session number listed in the Conn field.

```
outland-ts#resume 2
```

```
[Resuming connection 2 to outlan-sw01 ... ]
```

```
Your Connected to Outland-SW01
```

```
User Access Verification
```

```
Username:
```

As you can see, the IOS is quite capable of providing more than adequate session management. In terms of configuration, a loopback interface should be created to correspond with the line rotary, along with a local host table entry that maps a hostname to the loopback address and line rotary. Here is a syntax example:

```
outland-ts(config)#interface loop back 41
```

```
outland-ts(config-if)#ip address 172.30.2.41 255.255.255.255
```

Configuring IOS Terminal Services

Michael J. Martin

```
outland-ts(config-if)#description Interface for line rotary 41
outland-ts(config-if)#exit
outland-ts(config)#ip host outlan-ids 2041 172.30.2.41
outland-ts(config)#
```

Configuring the Line

Once access to the line is enabled, the line's speed, databits, parity, and stopbits should be set to accommodate the upstream device's line settings. The defaults are 9600:8:1:None. The line configuration commands are:

```
outland-ts(config-line)#speed 9600
outland-ts(config-line)#databits 8
outland-ts(config-line)#stopbits 1
outland-ts(config-line)#parity none
```

In most cases, the default settings are adequate and no configuration is needed. However, when configuring lines for modem support, the speed setting should be set to the highest supportable data rate, which will be either 115,200 kbps or 38,400 kbps. This sets the starting point for line speed arbitration. If left at the default, the line rate will start at 9600 kbps, limiting the connection speed regardless of the rate achieved between the modems over the PSTN.

Configuring Session Timers

Once the line has been configured for access, there are two session control options to consider:

- The <absolute-timeout {0-10000 in min}> session timer, which sets a time limit on the length of the session.
- The <session-timeout {0-35791 in min}> session timer, which sets how long the session will remain active if no activity detected.

None of these parameters is set by default, and their use is entirely discretionary. Discretionary, however, does not mean that they are not a good idea. Using session timeouts prevents two things from happening. First, they ensure that sessions are closed out correctly. This prevents users from "piggybacking" on other users' sessions that have authenticated on a connected device and have disconnected from the terminal server but have not logged off the connected device. Second, it makes sure that lines are not tied up indefinitely. Here is a syntax example that sets the session timeout at 30 minutes and the absolute timeout at an hour:

```
outland-ts(config-line)#absolute-timeout 60
outland-ts(config-line)#session-timeout 30
```

When using these commands on a line connected to a Cisco device, it is a good to have the timer values be identical to ensure both the devices behave the same.

Configuring Session Status Queues And Line Banners

The last step of the DTE-to-DTE line configuration is setting up the line description, connection banner, and busy message. The IOS supports a <description {text}> interface configuration command so administrators can provide notations in the configuration about the line. With line interfaces, the command is <location {text}>. Here is a syntax example:

```
outland-ts(config-line)#location Console Port For Outlan-SW01
```

Configuring IOS Terminal Services

Michael J. Martin

In addition to providing information on what is connected to the line, the <location> definition is also referenced by IOS informational banners. One of the annoying things about using IOS terminal servers is the problem of double authentication. Unless disabled, users are prompted for authentication by the terminal server when they connect to the line and by the device connected to the line. There are arguments for and against using double authentication; it's up to each administrator to decide. When double authentication is in place, it is hard for users to figure out which device is prompting them for authentication. The incoming connection banner, set with the <banner incoming {ctl char} {text} {ctl char}> addresses this issue. Here is a syntax example:

```
outland-ts(config)# banner incoming ^
You are connected to $(line-desc) on TTY line $(line)
```

To end this session type ctrl+shift+6 then x.

You may be required to authenticate on the connected device for console access.

^

The message text can be anything, but the IOS supports four system variable or tokens, which reference IOS configuration definitions that allow administrators to generate more descriptive banner messages:

```
$(hostname) displays the router's hostname
$(domain) displays the ip domain-name set on the router
$(line) displays the line number to which the user is connected
$(line-desc) displays the text set in the location field of the line configuration.
```

The last information queue to set is the busy message. The busy message is displayed when a user attempts a connection to a line that already has an active session. The busy message is mapped to a hostname entry, rather than the line. The message is set using the configuration command <busy-message {hostname | IP} {ctrl-chr} message {ctrl-chr}>. Here is a syntax example:

```
outland-ts(config)#busy-message outlan-gw ^
Enter TEXT message. End with the character '^'.
The line-rotary for Outlan-Gw is busy at this time.
```

^

```
outland-ts(config)#
```

Configuring Modem Support

Implementing dial-in/dial-out modems using an IOS terminal server provides a secure way to provide an insecure service. While the importance of modems has diminished with the explosion of broadband Internet access, modems are still widely used to provide OOB terminal access and backup network access when broadband access fails. Modems attached to async lines can be configured to support dial-in (over PPP or SLIP) dial-out or both.

To configure modem support, start with same initial configuration for DTE-to-DTE support. Keep in mind that the <speed> setting should be set at the highest supportable rate for the line and modem. The aptly named line configuration command <modem> is used to set up line parameters for modem support. In most cases, there are two configuration parameters that need to be set. The first option sets the line call support. There are two options: dial-in, which is set using <modem Dialin>, or dial-in/dial-out, which is set using the <modem InOut>. The second option sets up modem hardware

Configuring IOS Terminal Services

Michael J. Martin

support. This is set with the `<modem autoconfigure {discovery | type [type]}>`. Here is a syntax example:

```
outland-ts (config)#line 34
outland-ts (config-line)#speed 115200
outland-ts (config-line)#modem inOut
outland-ts (config-line)#modem autoconfigure discovery
```

The "discovery" option has the line test for modem type based on the IOS's modemcap database. If you want to view the results of discovery process, enable debugging using the `debug confmodem` command. If you are using a popular modem type, the "type" option allows you to set the modem type using one of the IOS's predefined modemcap files. Alternatively, you can create a custom modemcap file to support more esoteric modem brands. The IOS has modemcap files for the following modems (viewable using the `<show modemcap>` command):

```
outland-ts #sh modemcap
default
codex_3260
usr_courier
usr_sportster
hayes_optima
global_village
viva
telebit_t3000
microcom_hdms
microcom_server
nec_v34
nec_v110
nec_piafs
cisco_v110
microcom_mimic
mica
nextport
scm
```

To create your own, use the global configuration command `<modemcap edit <filename> miscellaneous {modem init string}>`:

```
outland-ts (config)#modemcap edit tester miscellaneous &ATZ
outland-ts (config)#
```

Once the line is configured for modem support, all that is left is to enable dial-in and/or dial-out. Dial-out is easy, and while it is fine to access each dial-out modem using its unique rotary line, the easier approach is to configure a rotary group for all of the dial-out modems. This way a user only has one session to a single hostname/rotary to use one of the modems in the pool. Here is the configuration syntax for five modem dial-out pool.

First, create a hostname entry for the rotary group:

```
outland-ts (config)#ip host modpool 3050 172.30.2.50
```

Then create a loopback interface for the rotary group binding:

Configuring IOS Terminal Services

Michael J. Martin

```
outland-ts (config)#interface loopback 500
outland-ts (config-if)#ip address 172.30.2.50 255.255.255.255
```

Add the modem lines to the rotary group; in this case group 50:

```
outland-ts (config)# lines 33 38
outland-ts (config-line)#rotary 50
outland-ts (config-line)# exit
```

Last, create a busy message to let users know if the pool is full:

```
outland-ts (config)#busy-message modpool ^
Enter TEXT message. End with the character '^'.
All of the dial-out lines are busy at this time.
^
outland-ts (config)#
```

To set up dial-in, in addition to the <modem Dialin> or <modem InOut> option being applied to the line, a group-async interface needs to be created that references the line or lines that will support dial-in.

Here is a group-async interface configuration that uses the same line pool used for dial-out.

```
outland-ts(config)#interface group-Async 1
outland-ts(config-if)#ip unnumbered FastEthernet 0/0
outland-ts(config-if)#ip tcp header-compression passive
outland-ts(config-if)#encapsulation ppp
outland-ts(config-if)#async mode dedicated
outland-ts(config-if)#async default routing
outland-ts(config-if)#group-range 33 38
outland-ts(config-if)#peer default ip address pool ppp-dial
```

Along with the async-interface, in order for dial-in to actually work, a local IP address pool must be created to assign the dial-in users IP addresses.

```
outland-ts(config)#ip local pool ppp-dial 172.30.71.67
```

Dial-in also requires that at least a minimum AAA configuration be in place:

```
outland-ts(config)# aaa authentication login default local
outland-ts(config)# aaa authentication ppp default local
outland-ts(config)# aaa authorization network default none
```

AAA provides the support for username/password authentication for PPP clients.

Access Control

The last thing we need to consider is access control. Our approach is dictated by two factors. First, if users access the line using a local or remote access model. Second, if AAA old-model (default) or AAA new-model services are in place. Here are the access control capabilities based on the AAA and line access model being employed:

- AAA old-model, remote line access model. No authentication is required to connect to the rotary line by default, to enable password checking, set the line option <login local> and create local user

Configuring IOS Terminal Services

Michael J. Martin

accounts using <username {name} password {password}>. Users may be prompted for authentication by the connected device, if configured. Per line access control can be implemented based on source IP address using <access-class {sacl #} {in | out}>.

- AAA new-model, remote line access model. The default authentication named-list will be applied to every VTY, CON, AUX and TTY line. Users may be prompted for an additional authentication round by the connected device, if authentication is configured. Per line access control can be implemented based on source IP address using <access-class {sacl #} {in | out}>.
- AAA old-model, local line access model: No authentication is required to connect to the rotary line by default. To enable password checking, set the line option <login local> and create local user accounts using <username {name} password {password}>. Users may be prompted for authentication by the connected device, if configured. User session control is available. No per-line access control is available. To prevent remote line access, an inbound filter list should be installed on the terminal server LAN interface that drops packets to line and group rotary ports. Here is an example filter list. (Note: The filter references a hostname, instead of an IP address. The ACL will translate the hostname to an IP address when it compiles the ACL.):

```
ip access-list extended tty-control
remark allow hosts to access TS
permit tcp any host outlan-ts eq 22
permit tcp any host outlan-ts eq telnet
remark TS to TACACS server communication
permit ip host trinity host outlan-ts
remark block incoming telnet connections to TTY service ports
deny tcp any host outlan-ts eq 2033
deny tcp any host outlan-ts eq 2034
deny tcp any host outlan-ts eq 2035
deny tcp any host outlan-ts eq 2036
deny tcp any host outlan-ts eq 2040
remark block incoming ssh connections to TTY service ports
deny tcp any host outlan-ts eq 4033
deny tcp any host outlan-ts eq 4034
deny tcp any host outlan-ts eq 4035
deny tcp any host outlan-ts eq 4036
deny tcp any host outlan-ts eq 4040
remark block incoming telnet connections to TTY rotaries
deny tcp any host outlan-ts range 3001 3099
remark permit all other traffic
permit ip any any
```

- AAA new-model, local line access model: The default authentication named list will be applied to every VTY, CON, AUX and TTY line. Users may be prompted for an additional authentication round by the connected device, if authentication is configured. Line access control is available on a per-user or per-group basis using AAA command authorization for privilege-level 1 commands. Restricting the hosts user/groups is permitted to telnet/connect/SSH to enforce access control. Here is a TACACS server configuration example that permits user oob to access the hosts outlan-rt01 and outlan-sw01:

#Controlling Access To Specific TS connected Devices.

```
user = oob {
login = cleartext oob
```

Configuring IOS Terminal Services

Michael J. Martin

```
cmd = telnet {
permit outlan-rt01
permit outlan-sw01
deny .*
}
cmd = ssh {
permit outlan-rt01
permit outlan-sw01
}
cmd = connect {
permit outlan-rt01
permit outlan-sw01
}
cmd = clear {
permit line
deny .*
}
cmd = show {
permit .*
}
cmd = resume {
permit .*
}
cmd = exit {
permit .*
}
cmd = disconnect {
permit .*
}
}
```

This approach will only allow the user oob to connect to the specified two rotary line interfaces. However, if this approach were to be restrictive, the configuration could be set up to use explicit "deny" clauses with a "permit" all at the end:

```
cmd = telnet {
deny outlan-rt02
deny outlan-sw02
deny outlan-rs01
deny outlan-rs02
permit .*
}
```

In addition to the TACACS server configuration, AAA authorization must be enabled on the terminal server:

```
outland-ts(config)# aaa authorization commands 1 default group tacacs+
```

As was the case in the previous scenario, to prevent remote line access, an inbound filter list must be installed on the terminal server LAN interface that drops packets to line and group rotary ports.

Configuring IOS Terminal Services

Michael J. Martin

That wraps up our IOS terminal server series. I hope that you have found this information helpful and it will enable you to improve your own networking environment. As always, any questions or concerns are appreciated. Stay tuned; next month we'll be implementing IOS menus