Chapter 44

# Secure Shell

# Introduction

This chapter describes the Secure Shell protocol, support for Secure Shell on the router, how to configure the router to act both as a Secure Shell server and client, and how to use Secure Shell to manage the router.

Secure management is increasingly important in modern networks, as the ability to easily and effectively manage routers and the requirement for security are two almost universal requirements. Traditionally, routers are managed using either remote terminal sessions via the Telnet protocol or SNMP. Both methods have serious security problems. They are protected by plaintext reusable passwords that are vulnerable to wiretapping and password guessing, which could result in sensitive information, even the passwords themselves, traversing the network as plaintext.

The Secure Shell (SSH) protocol is similar to the Telnet and rlogin protocols in providing encrypted and strongly authenticated remote login sessions. SSH provides sessions between a host running a Secure Shell server and a machine with a Secure Shell client.

The router implements both a Secure Shell server and a Secure Shell client to enable network managers to securely—with the benefit of cryptographic authentication and encryption—manage the routers over an insecure network:

- Secure Shell replaces Telnet for remote terminal sessions. Secure Shell is strongly authenticated and encrypted.
- Remote command execution allows manager commands to be sent to a router securely and conveniently, without requiring a terminal session on the router.
- Secure Shell supports Secure Copy (SCP), which provides a way of securely copying files between the router and remote machines.
- A network manager logged onto a router may connect to another router or a secure host using Secure Shell.

# Secure Shell on the Router

The router implementation of the Secure Shell protocol is compatible with the Secure Shell protocol described in the Internet Draft, *The SSH (Secure Shell) Remote Login Protocol*.

The protocol depends on other technologies, including DES encryption and RSA public key encryption, that may require licencing in some regions, and may be subject to ITAR export controls.

Secure Shell supports the following features:

- inbound Secure Shell connections (server mode) and outbound Secure Shell connections (client mode).
- file loading to and from remote machines using Secure Copy, using either the SSH client or SSH server mode.
- RSA public keys with lengths of 512–2048 bits. Keys are stored in a format compatible with other Secure Shell implementations, and mechanisms are provided to copy keys to and from the router.

- Single DES and Triple-DES (3DES) encryption for Secure Shell sessions. 3DES encryption requires a special feature licence.
- remote non-interactive shell that allows arbitrary router commands to be sent securely to the router, possibly automatically.

All functions work whether or not a PCI Accelerator Card (PAC) card is installed. If a PAC card is installed, it improves performance.

Secure Shell does **not** support the following features:

- IDEA or Blowfish encryption.
- Compression of Secure Shell traffic.
- Non-encrypted Secure Shell sessions. The purpose of Secure Shell is to provide encrypted and strongly-authenticated terminal sessions so the Secure Shell server refuses to operate in non-encrypted mode.
- Tunnelling of TCP/IP traffic. Instead IP encryption should be implemented using Security Associations.

Log messages are generated to record the following events:

- addition, deletion or modification of a Secure Shell user
- acceptance or rejection of a Secure Shell connection
- enabling or disabling the Secure Shell server
- regeneration of the Secure Shell server key
- disconnection of Secure Shell connections
- enabling or disabling Secure Copy service
- reasons for failed or aborted file transfers using Secure Copy

The Secure Shell implementation is functionally similar to existing implementations on various platforms, such as ssh-1.2.26 (UNIX) and Datafellows F-Secure SSH (Windows), and interoperates with these implementations.

# Configuring Secure Shell

## Configuring the Server

The Secure Shell server must be enabled before connections from Secure Shell and Secure Copy clients are accepted. If a firewall policy exists on the Secure Shell server, you must add a rule that allows access to the Secure Shell server on port 22 before Secure Shell connections are accepted. When the Secure Shell server is disabled, connections from Secure Shell clients are rejected.

To add a firewall rule that accepts Secure Shell connections, use the **add firewall policy rule** command on page 46-82 of Chapter 46, Firewall.

To enable or disable the Secure Shell server, use the commands:

```
enable ssh server hostkey=key-id serverkey=key-id
   [expirytime=0..168] [logintimeout=1..600]
   [scp={enabled|disabled}]

disable ssh server
```

To modify the Secure Shell server configuration, use the command:

```
set ssh server [hostkey=key-id] [serverkey=key-id]
   [expirytime=0..168] [logintimeout=1..600]
   [scp={enabled|disabled}]
```

Secure Copy can be disabled or enabled on the SSH server using the **scp** parameter in the **enable ssh server** and **set ssh server** commands. This allows you to reject Secure Copy file transfer requests, while still allowing Secure Shell connections.

To display the current configuration of the Secure Shell server, use the command:

```
show ssh
```

## Configuring the Client

The **set ssh client** command allows you to specify timeout and file modification options when the router is acting as an SSH client. Use the command:

```
set ssh client idletimeout=[0..4294967295]
   [logintimeout=1..600] [preservemtime={enabled|disabled}]
```

To initiate a Secure Shell connection from the router to a remote router or Secure Shell server, use the command:

```
ssh ipadd user=username {password=password|keyid=key-id}
   [command=string]
```

To display the current configuration of the Secure Shell client, use the command:

```
show ssh
```

## Configuring Users

The Secure Shell server accepts only connections from registered users. When there are no registered users, the router does not accept Secure Shell or Secure Copy connections even if the Secure Shell server is enabled. The following commands add and delete registered users:

```
add ssh user=username {password=password|keyid=key-id}
   [ipaddress=ipadd] [mask=mask]

delete ssh user=username
```

To modify the configuration for an existing registered user, use the command:

```
set ssh user=username {password=password|keyid=key-id}
   [ipaddress=ipadd] [mask=mask]
```

To temporarily disable and enable a registered user, use the commands:

```
disable ssh user=username
```

```
enable ssh user=username
```

A user that is disabled cannot make Secure Shell connections to the Secure Shell server. If a Secure Shell user makes 5 consecutive incorrect logins, their status changes to disabled. To display a list of registered users and their current status, use the command:

```
show ssh user[=username]
```

Secure Shell encryption key management is implemented by the ENCO module. RSA public keys can be imported and exported to/from the single-line ASCII format used by all Secure Shell implementations.

## Using Secure Copy

Secure Copy can copy files to and from the router, from either the router's location or from a remote machine. To allow client SCP sessions to connect to the router, enable SCP service on the server:

```
set ssh server scp=enabled
```

To use the router's SCP client to load files onto the router, use the command:

```
load method=scp [delay=delay] [destfile=destfilename]
   [destination={cflash|flash|nvs}]
   [{file|srcfile}=filename]
   [{keyid=key-id|password=password}]
   [server={hostname|ipadd|ipv6add}] [username=username]
```

To use the router's SCP client to upload files onto a remote machine, use the command:

```
upload method=scp [destfile=destfilename] [file=filename]
   [{keyid=key-id|password=password}]
   [server={hostname|ipadd|ipv6add}] [username=username]
```

For further information on loading files using Secure Copy, see the sections "Loading from a Remote Device using Secure Copy" on page 5-8, and "Using Secure Copy" on page 5-12 in Chapter 5, Managing Configuration Files and Software Versions.

## Managing Sessions

You can monitor the current status of Secure Shell and Secure Copy connections by using the **show ssh session** command. This displays both uploads and downloads, and whether the router is acting as a client or server. Use the command:

```
show ssh session[={all|scp|ssh}]
```

To delete Secure Shell and Secure Copy sessions without disabling the SSH server, use the command:

```
delete ssh session={session-id|all}
```

To see detailed information about the Secure Shell and Secure Copy counters, use the command:

```
show ssh counter[={all|scp|ssh}]
```

### Debugging Secure Shell and Secure Copy

Information which may be useful for troubleshooting SSH and SCP connections is available using the SSH debugging function. To enable or disable debugging on either SSH, SCP, or both, use the commands:

```
enable ssh debug[={ssh|scp|all}]
```

```
disable ssh debug[={ssh|scp|all}]
```

# Configuration Examples

## Configuring Secure Shell on the Router

The example in this section shows how to create encryption keys, configure the Secure Shell server, and register users to make Secure Shell connections to the router.

1. **Create a Security Officer login on the server.**

   A Security Officer must be configured before putting the router in security mode. Add a user with Security Officer privilege to the User Authentication Database:

   ```
   add user=login-name login={true|false|on|off|yes|no}
      password=password privilege=securityofficer
      [other-user-parameters]
   ```

   See the **add user** command on page 41-33 of Chapter 41, User Authentication.

2. **Enable security mode.**

   To store the Secure Shell configuration and other security information, such as encryption keys when the router is restarted, enable the security mode as follows:

   ```
   enable system security_mode
   ```

   When the router is in security mode, only a user with Security Officer privilege can configure security-sensitive aspects of it. See "Operating Modes" on page 41-7 of Chapter 41, User Authentication.

   **Important** Sensitive data, such as encryption keys, created on a router that is not in security mode is destroyed when the router is restarted. Enable security mode before creating encryption keys.

3. **Create encryption keys.**

   Two RSA private keys are required to enable the Secure Shell server. The first, is the *host key*, which is the router's own RSA key. The recommended length of this key is 1024 bits. The second key, the *server key*, is a randomly created key, which is re-generated after the specified timeout. This key must be 128 bits shorter than the host key, but should be at least 512 bits.

   Before creating keys, first define a user with Security Officer privileges. Defining a security officer lets that user enable security mode when logging in. The security officer definition process must be carried out on all routers that use the keys (i.e., on the head office router and the remote office router).

   ```
   create enco key=0 type=rsa length=1024
      description="Host Key" form=ssh
   ```

   ```
   create enco key=1 type=rsa length=768
      description="Server Key" form=ssh
   ```

To verify the key creation, use the command:

```
show enco key
```

4. **Optionally add a firewall rule that accepts Secure Shell connections.**

If you have a firewall enabled on the Secure Shell server, you must add a rule that accepts Secure Shell connections to the server. For example, to add Secure Shell access over port 22 with a public IP address of 200.200.200.1, a private IP address of 192.168.1.1, a public interface of ppp0, and a remote IP address of 200.200.200.5, use the command:

```
add firewall policy=main rule=1 action=allow interface=ppp0
   protocol=tcp port=22 ipaddress=192.168.1.1
   gblip=200.200.200.1 gblport=22 remote=200.200.200.5
```

5. **Enable the Secure Shell server.**

Once the required host and server keys have been generated, the Secure Shell server can be enabled using the command:

```
enable ssh server hostkey=0 serverkey=1 expirytime=1
   logintimeout=60
```

In this case, the server key expiry time has been set to 1 hour. Because routers are always busy (unlike a UNIX box), it is recommended that the expiry time be set to ensure that RSA keys are re-generated in off-peak times. One method of doing this is to set the expiry time to 0 (never) and then use the router script facility to destroy and recreate the key.

To verify the SSH server status, use the command:

```
show ssh
```

6. **Create SSH users.**

In order to connect and execute commands that require manager or security officer privilege on the router, users must be registered in the SSH user database, and in the User Authentication Database of the router. It is possible to create an SSH only user, however they will have very limited abilities.

**Create a user in the User Authentication Database**

To create a user in the User Authentication Database, use the command:

```
add user=john password=secret privilege=manager
```

The password configured for this user is not important; it does not have to match the password for the SSH user. For further details see the **add user** command on page 41-33 of Chapter 41, User Authentication.

**Create an SSH user**

SSH users cannot connect unless they can be authenticated. There are two ways to authenticate an SSH session: password authentication, and RSA private/public key authentication.

**To assign password authentication**

To create the user “john” with the password “secret”:

```
add ssh user=john password=secret
```

**To assign RSA authentication**

Create an RSA public key for the user. This RSA public key must be the public part of the RSA private key that the user sends when opening the SSH session.

To create the public key on the server you must:

(a) Create the public key file. This can be done in a number of ways by using a variety of SSH client programs. It can also be done on the router. In this example, we are creating the public key on the client router. (The remote router in this example is called the server router.)

First, create a private key on the **client** router:

```
create enco key=5 type=rsa length=768 description="Example
   private key"
```

The length of the key is flexible, it does not have to be 768 bits.

Then, from the private key, create a public key in a file

```
create enco key=5 type=rsa file=user.key form=ssh
```

The key number entered in this command must be the same as that used in the previous command because the purpose of this command is to create a public key from the private key. The file name must have a .key extension.

Once the key file has been created, it can be uploaded to a TFTP server or zmodem server so that it can be loaded onto the SSH server:

```
upload server=ipadd file=user.key
```

or

```
upload po=0 method=zmo file=user.key
```

(b) Load the public key file onto the **server** router.

```
load server=ipadd destination=flash file=user.key
```

or

```
load po=0 meth=zmo destination=flash
```

(c) Create an ENCO key from this file on the **server** router:

```
create enco key=7 type=rsa file=user.key
   description="user's public key" form=ssh
```

Now that the public key is created on the server, it is possible to create the SSH user:

```
add ssh user=name keyid=7
```

#### Other SSH user parameters

An IP address or subnet can also be added to force the user to login from that address. To require the user "john" to attach from the network IP 192.168.2.0, use the command:

```
set ssh user=john ipaddress=192.168.2.0 mask=255.255.255.0
```

Verify the list of registered Secure Shell users:

```
show ssh user
```

**7. Create an outgoing SSH terminal session.**

Use the authentication method (password or RSA) that matches the method from the previous step to create users.

#### Password authentication

To create an outgoing terminal session using password authentication, follow this procedure:

A user "john" has been configured with password authentication (password "secret") on a remote server with the IP address 192.168.2.5. To login from a client router to the remote server, use the command:

```
ssh 192.168.2.5 user=john password=secret
```

The first time that you make an SSH connection from a client router to a server using password authentication, the server sends the public part of its host key to the client router. Because the client router has not seen that host key before, it presents the message:

```
Host key not recognised - saved as ssh.key

SSH. Session closed.
```

At this point it is necessary to create an ENCO key from the saved public key file.

To create the ENCO key, use the command:

```
create enco key=6 type=rsa file=ssh.key
    description="B host key" form=ssh
```

You can then successfully create an SSH connection to the server with the command:

```
ssh 192.168.2.5 user=john password=secret
```

**RSA authentication**

To create the outgoing terminal session using RSA authentication, follow this procedure:

A user "Admin" has been configured with an RSA public key (matching RSA private key 5 on the local router) on a server router with the IP address 192.168.2.5. To use Secure Shell to login to the server router as user "Admin", on the client router, use the command:

```
ssh 192.168.2.5 user=Admin keyid=5
```

## Connecting and Executing Commands After Configuration

It is possible to make an SSH connection that simply connects to the server router, executes a command, and disconnects again, by using the command:

```
ssh 192.168.2.5 user=john password=secret
    command="act scr=test.scp"
```

In order to execute commands that require manager or security officer privilege, the server router must be configured with a manager or security officer user corresponding to the SSH user. If the SSH user is a member of the router's User Authentication Database, then the user also has the associated privileges.

For example, having created the SSH user “john”, it is necessary to also create “john” in the User Authentication Database on the router, by using the command:

```
add user=john password=any_old_thing privilege=manager
```

The password configured for this user is not important; it does not have to match the password for the SSH user.

# Command Reference

This section describes the commands available on the router to enable, configure, control, and monitor the Secure Shell client and server.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page lxv of About this Software Reference in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of messages and their meanings.

# add ssh user

**Syntax**

```
ADD SSH USER=username {PASSword=password|KEYid=key-id}
   [IPaddress={ipadd|ipv6add}] [MASk=mask]
```

**Description**

This command adds an entry to the list of users who are permitted to connect to the router via Secure Shell. If the user list is empty, no Secure Shell connections are allowed. If the Secure Shell user is also a member of the router user database, then that user has the associated privileges.

This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

| Parameter | Description |
|---|---|
| USER | The username that the user must supply when connecting to the router using a Secure Shell client. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and decimal digits (0–9). The string cannot contain spaces.<br>Default: no default |
| PASSword | The password that the user must supply when connecting to the router using a Secure Shell client. The *password* is a string 1 to 31 characters long. Valid characters are any printable character. If the string contains spaces, put double quotes around it.<br>Default: no default |
| KEYid | The RSA key being used to authenticate the user. This parameter enables Secure Shell RSA authentication. The specified key must exist. The *key-id* is a decimal number.<br>Default: no default |
| IPaddress | The range of IP addresses from which the user may connect.<br>Default: no default (all connections are accepted from any IP address)<br>*ipadd* — An IPv4 address in dotted decimal notation.<br>*ipv6add* — A valid IPv6 address in hexadecimal notation. |
| MASk | The range of IP addresses from which the user may connect. Valid only when **ipaddress** is an IPv4 address.<br>Default: the natural mask for the network, based on whether it is a class A, B, or C network |

**Example** To add a Secure Shell user with password authentication and a specified IP address, use the command:

```
add ssh user=john pass=secret ip=192.168.2.1
   mas=255.255.255.0
```

To add a Secure Shell user with RSA authentication, use the command:

```
add ssh user=john key=5
```

**Related Commands**

**delete ssh user**
**set ssh user**
**show ssh user**

# delete ssh session

**Syntax** `DELete SSH SEssion={`*`session-id`*`|ALL}`

**Description** This command deletes Secure Shell and Secure Copy sessions that are currently active on the router. This can include both server and client sessions. The deleted sessions are closed.

The *session-id* is the number assigned to each connection. Use a comma-separated list to specify more than one *session-id*. To see a list of current SSH sessions with their *session-id* numbers, use the **show ssh session** command. If a *session-id* number is specified, that session is closed. If **all** is specified, all connections are closed, except the sessions that are listening on the TCP port for new SSH connections.

**Example** To delete the SSH sessions with the ID numbers 2, 4 and 5, use the command:

```
del ssh se=2,4,5
```

**Related Commands** **show ssh session**

# delete ssh user

**Syntax** `DELete SSH USER=`*`username`*

**Description** This command deletes the specified user from the list of users permitted to connect to the router via Secure Shell.

This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

The **user** parameter specifies the user name being deleted. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and decimal digits (0–9). The string cannot contain spaces.

**Example** To delete a Secure Shell user use the command:

```
del ssh user=john
```

**Related Commands** **add ssh user**
**set ssh user**
**show ssh user**

# disable ssh debug

**Syntax** `DISable SSH DEBug[={ALL|SCP|SSH}]`

**Description** This command disables the SSH server debugging facility. Debugging is disabled by default.

| Parameter | Description | |
|---|---|---|
| DEBug | Whether debugging is disabled for either Secure Shell, Secure Copy, or both.<br>Default: **ssh** | |
| | ALL | Debugging for both Secure Shell and Secure Copy is turned off. |
| | SCP | Debugging is turned off for Secure Copy. |
| | SSH | Debugging is turned off for Secure Shell. |

**Example** To disable debugging of SCP, use the command:

```
dis ssh deb=scp
```

**Related Commands**
**enable ssh debug**
**show ssh**

# disable ssh server

**Syntax** `DISable SSH SERver`

**Description** This command disables the Secure Shell server. When the Secure Shell server is disabled, connections from Secure Shell and Secure Copy clients are not accepted.

The Secure Shell server is disabled by default. Secure Shell and Secure Copy sessions may be initiated from the router to another host, but inbound connections are not accepted.

This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

**Example** To disable the Secure Shell server, use the command:

```
dis ssh ser
```

**Related Commands**
**enable ssh server**
**set ssh server**
**show ssh**

# disable ssh user

**Syntax** `DISable SSH USER=`*`username`*

**Description** This command disables a Secure Shell user so that they cannot login. By default, users are enabled when they are added to the list of permitted users.

This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

The **user** parameter specifies the user name being deleted. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and digits (0–9). The string cannot contain spaces.

**Example** To disable a Secure Shell user, use the command:

```
dis ssh user
```

**Related Commands**
**add ssh user**
**delete ssh user**
**enable ssh user**
**set ssh user**
**show ssh user**

# enable ssh debug

**Syntax** `ENAble SSH DEBug[={ALL|SCP|SSH}]`

**Description** This command enables the SSH server debugging facility. Debugging is disabled by default.

| Parameter | Description | |
|---|---|---|
| DEBug | Whether debugging is enabled for either Secure Shell, Secure Copy, or both.<br>Default: **ssh** | |
| | ALL | Debugging for both Secure Shell and Secure Copy is turned on. |
| | SCP | Debugging is turned on for Secure Copy. |
| | SSH | Debugging is turned on for Secure Shell. |

**Example** To enable debugging of SCP, use the command:

```
ena ssh deb=scp
```

**Related Commands**
**disable ssh debug**
**show ssh**

# enable ssh server

**Syntax**

```
ENAble SSH SERver HOSTKey=key-id SERVERKey=key-id
   [EXPirytime=0..168] [LOGintimeout=1..600]
   [SCP={ENAbled|DISabled}]
```

**Description**

This command enables the Secure Shell server. Once enabled, connections from Secure Shell and Secure Copy clients are accepted. Secure Copy connections can be disabled, while still allowing Secure Shell connections, using this command. If a firewall is configured on the router, you must add a rule that accepts Secure Shell connections before you can enable the Secure Shell server.

The Secure Shell server is disabled by default. Secure Shell and Secure Copy sessions may be initiated from the router to another host but inbound connections are not accepted.

This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

| Parameter | Description | |
|---|---|---|
| HOSTKey | The key that must be used for the router host key. The specified key must exist. The *key-id* is a decimal number.<br>Default: no default | |
| SERVERKey | The key that must be used for the Secure Shell server key. The specified key must exist. The *key-id* is a decimal number.<br>Default: no default | |
| EXPirytime | The time in hours after which the Secure Shell server key expires and is regenerated. If 0 is specified, the key does not expire.<br>Default: **0** | |
| LOGintimeout | The time in seconds the server waits before disconnecting an un-authenticated client. This timeout cannot be disabled.<br>Default: **60** | |
| SCP | Whether the SSH server supports SCP connections.<br>Default: **enabled** | |
| | ENAbled | Allows SCP connections |
| | DISabled | Does not allow SCP connections |

**Example**

To enable the Secure Shell server, use the command:

```
ena ssh ser hostk=0 serverk=1
```

**Related Commands**

**disable ssh server**
**set ssh server**
**show ssh**

# enable ssh user

**Syntax** `ENAble SSH USER=`*`username`*

**Description** This command enables a Secure Shell user. By default, users are enabled when they are added to the list of permitted users. However, if a user fails to login five times in a row, the user is disabled. This command enables the user again.

This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

The **user** parameter specifies the user name being deleted. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and digits (0–9). The string cannot contain spaces.

**Example** To enable a Secure Shell user who has been disabled, use the command:

```
ena ssh user=john
```

**Related Commands**
**add ssh user**
**delete ssh user**
**disable ssh user**
**set ssh user**
**show ssh user**

# set ssh client

**Syntax**

```
SET SSH CLIent [IDLEtimeout=0..4294967295]
   [LOGintimeout=1..600]
   [PREservemtime={ENAbled|DISabled}]
```

**Description**

This command modifies the configuration of the Secure Shell client. When the router is in security mode, this command requires a user with Security Officer privilege.

| Parameter | Description | |
|---|---|---|
| IDLEtimeout | The period of time, in seconds, set for the SSH client's idle timer. If the specified time period lapses since the last time an SSH session received data from the remote server, the session is terminated. This applies from the moment that the SSH session becomes established, regardless of whether the user has logged in or not. If the SSH client idle timeout period is modified while there are established SSH sessions, the idle timers for those sessions are reset so that they use the new timeout value. Any idle time accumulated by those sessions prior to the modification is lost.<br>Default: **0** | |
| | 0 | The idle timer remains off, and the session must be terminated by the user. |
| | 1..4294967295 | The idle timer is active, and the session terminates when the idletimeout limit is reached. |
| LOGintimeout | The time in seconds that the client waits for the SSH session to establish. This cannot be turned off.<br>Default: **30** | |
| PREservemtime | Whether the SCP client preserves the modification time of the source file.<br>Default: **enabled** | |
| | ENAbled | Files copied to and from the router keep the same modified time as the source file. |
| | DISabled | Files copied to and from the router show the time of being copied as the modified time. |

**Example**

To set the SSH client idle timer to three minutes, and the login timer to 10 seconds, use the command:

```
set ssh client idle=180 log=10
```

**Related Commands**

**disable ssh server**
**enable ssh server**
**set timezone** in Chapter 4, Configuring and Monitoring the System
**show ssh**
**show ssh session**
**ssh**

# set ssh server

**Syntax**

```
SET SSH SERver [HOSTKey=key-id] [SERVERKey=key-id]
   [EXPirytime=0..168] [IDLEtimeout=0..4294967295]
   [LOGintimeout=1..600] [MAXSessions=0..6]
   [SCP={ENAbled|DISabled}]
```

**Description**

This command modifies the configuration of the Secure Shell server, including whether Secure Copy connections are allowed. When the router is in security mode, this command requires a user with Security Officer privilege. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

| Parameter | Description | |
|---|---|---|
| HOSTKey | The key used for the router host. The specified key must exist. The *key-id* is a decimal number.<br>Default: no default | |
| SERVERKey | The key used for the Secure Shell server. The specified key must exist. The *key-id* is a decimal number.<br>Default: no default | |
| EXPirytime | The time in hours after which the Secure Shell server key expires. If 0 is specified, the key does not expire.<br>Default: **0** | |
| IDLEtimeout | A period of time, in seconds, for the SSH server's idle timer. If the specified time period lapses since the last time an SSH session received data from the remote client, the session is terminated. This applies from the moment that the SSH session becomes established, regardless of whether the user has logged in or not. If the SSH server idle timeout period is modified while there are established SSH sessions, the idle timers for those sessions are reset so that they use the new timeout value. Any idle time accumulated by those sessions prior to the set command is lost.<br>Default: **0** | |
| | 0 | If specified, the idle timer remains off, and the session must be explicitly terminated. |
| | 1..4294967295 | If specified, the idle timer is active, and the session terminates when the idletimeout limit is reached. |
| LOGintimeout | The time in seconds the server waits before disconnecting an un-authenticated client. This timeout cannot be disabled.<br>Default: **60** | |
| MAXSessions | The number of concurrent SSH sessions that are supported by the router. Once the limit is reached, any subsequent session requests are rejected. The session limit cannot be set below the number of currently established SSH sessions.<br>Default: **6** | |

| Parameter | Description (cont) | |
|---|---|---|
| SCP | Whether the SSH server supports SCP connections.<br>Default: **enabled** | |
| | ENAbled | Allows SCP connections. |
| | DISabled | Does not allow SCP connections. |

**Example** To set the Secure Shell server key expiry time to 1 hour, use the command:

```
set ssh ser exp=1
```

**Related Commands**

**disable ssh server**
**enable ssh server**
**show ssh**

# set ssh user

**Syntax**

```
SET SSH USER=username {PASSword=password|KEYid=key-id}
   [IPaddress={ipadd|ipv6add}] [MASk=mask]
```

**Description**

This command modifies an entry in a list of existing users permitted to connect to the router via Secure Shell. This command requires a user with Security Officer privilege when the router is in security mode. If security mode is disabled, sensitive data, for example encryption keys created on a router that is not in security mode, is destroyed when the router is restarted. See the **add user** command on page 41-33 of Chapter 41, User Authentication and **enable system security_mode** command on page 41-46 of Chapter 41, User Authentication.

| Parameter | Description | |
|---|---|---|
| USER | The name that the user must supply when connecting to the router using a Secure Shell client. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and decimal digits (0–9). The string cannot contain spaces.<br>Default: no default | |
| PASSword | The password that the user must supply when connecting to the router using a Secure Shell client. The *password* is a string 1 to 31 characters long. Valid characters are any printable character. If the string contains spaces, put double quotes around it.<br>Default: no default | |
| KEYid | The RSA key being used to authenticate the user. This parameter enables Secure Shell RSA authentication. The specified key must exist. The *key-id* is a decimal number.<br>Default: no default | |
| IPaddress | The range of IP addresses from which the user may connect.<br>Default: no default (all connections are accepted from any IP address). | |
| | *ipadd* | An IPv4 address in dotted decimal notation. |
| | *ipv6add* | A valid IPv6 address in hexadecimal notation. |
| MASk | The range of IP addresses from which the user may connect. Valid only when **ipaddress** is an IPv4 address.<br>Default: the natural mask for the network, based on whether it is a class A, B, or C network | |

**Example**

To change a Secure Shell user to RSA authentication or to change their key, use the command:

```
set ssh user=john key=5
```

**Related Commands**

**add ssh user**
**delete ssh user**
**show ssh user**

# show ssh

**Syntax** `SHow SSH`

**Description** This command displays the current configuration of the Secure Shell client and server (Figure 44-1, Table 44-1).

Figure 44-1: Example output from the **show ssh** command

```
Secure Shell Server Configuration
---------------------------------------------------
Version.............................. 1.5
SSH Server........................... Enabled
SCP Service.......................... Enabled
Maximum Sessions .................... 6
Current Sessions .................... 1
Port................................. 22
Host Key ID.......................... 0
Host Key Bits........................ 1024
Server Key ID........................ 1
Server Key Bits...................... 768
Server Key Expiry(hours)............. 0
Login Timeout (secs)................. 60
Idle Timeout(secs) .................. Off
Authentication Available............. Password,RSA
Ciphers Available.................... DES,3DES
Services Available................... Shell,Cmd,SCP
Debug................................ ALL

Secure Shell Client Configuration
---------------------------------------------------
Version.............................. 1.5
Login Timeout (secs)................. 30
Idle Timeout (secs).................. Off
Preserve File Modification Time (SCP).... Enabled
```

Table 44-1: Parameters in output of the **show ssh** command

| Parameter | Meaning |
|---|---|
| **Secure Shell Server Configuration** | |
| Version | Compatible version of the Secure Shell protocol. |
| SSH Server | Whether the Secure Shell server is enabled or disabled. |
| SCP Service | Whether Secure Copy is enabled or disabled. |
| Maximum Sessions | Maximum number of concurrent sessions supported by the Secure Shell server. |
| Current Sessions | Number of currently established Secure Shell sessions. |
| Port | TCP port where the Secure Shell server listens for connections. The default is port 22. |
| Host Key ID | ENCO key ID for the router's host RSA key. The Secure Shell server cannot operate without this key. |
| Host Key Bits | Length of the router host key, which must be from 768 to 2048 bits. |

Table 44-1: Parameters in output of the **show ssh** command (cont)

| Parameter | Meaning |
|---|---|
| Server Key ID | ENCO key ID for the Secure Shell server RSA key. The Secure Shell server cannot operate without this key. |
| Server Key Bits | Length of the Secure Shell server key. This must be at least 128 bits less than the host key (minimum 512 bits). |
| Server Key Expiry (hours) | Time in hours before the Secure Shell server key expires and is regenerated. |
| Login Timeout (secs) | Time in seconds when a client must authenticate itself before it is disconnected. |
| Idle Timeout (secs) | Time in seconds without data being received from a Secure Shell client before the corresponding session is terminated, or "Off" if the idle timeout is set to 0 (no timeout). |
| Authentication Available | List of available authentication methods; either Password or RSA. |
| Ciphers Available | List of the ciphers available; one or more of DES or 3DES. |
| Services Available | List of the available Secure Shell services; one or more of Shell, Cmd or SCP. |
| Debug | Whether debugging is active on the server. This can be set to debug SSH, SCP, ALL or NONE. |
| **Secure Shell Client Configuration** | |
| Version | Compatible version of the Secure Shell protocol. |
| Login Timeout (secs) | Time in seconds that the SSH client will wait to be authenticated. |
| Idle Timeout (secs) | Time in seconds that the SSH client will wait to receive data from an SSH server. The client disconnects if this timer limit is reached. If the timeout shows Off, timeout is set to 0 and never times out, so users must manually disconnect. |
| Preserve File Modification Time | Whether a copied file keep the source file's modification time (Enabled), or the modification time is set to the current time of copying (Disabled). |

**Example** To display the current configuration of the Secure Shell client and server, use the command:

```
sh ssh
```

**Related Commands**

**disable ssh server**
**enable ssh server**
**set ssh client**
**set ssh server**
**show ssh counter**
**show ssh session**

# show ssh counter

**Syntax** `SHow SSH COUnter[={ALL|SSH|SCP}]`

**Description** This command displays client and server counters for Secure Shell and Secure Copy.

| Parameter | Description | |
|---|---|---|
| COUnter | Which protocol to display counters for.<br>Default: **all** | |
| | ALL | Counters for both Secure Shell and Secure Copy are displayed (Figure 44-2, Table 44-2, Figure 44-3 on page 44-24, Table 44-3 on page 44-25). |
| | SCP | Counters for Secure Copy are displayed (Figure 44-2, Table 44-2). |
| | SSH | Counters for Secure Shell are displayed (Figure 44-3 on page 44-24, Table 44-3 on page 44-25). |

Figure 44-2: Example output from the **show ssh counter=scp** command

```
SCP Counters:

  uploadTotal ..................... 5      downloadTotal .................. 12
  uploadSuccess ................... 1      downloadSuccess ................. 5
  uploadFailed .................... 1      downloadFailed .................. 7
  uploadCancelled ................. 2      downloadCancelled ............... 0

  readFileRequest ............. 3570      writeFileRequest ............. 733
  readFileSuccess ............. 3570      writeFileSuccess ............. 733
  readFileFailed .................. 0      writeFileFailed .................. 0
```

Table 44-2: Parameters in output of the show **ssh counter=scp** command

| Parameter | Meaning |
|---|---|
| uploadTotal | Total number of upload requests received by the router. |
| downloadTotal | Total number of load requests received by the router. |
| uploadSuccess | Number of successful upload requests. |
| downloadSuccess | Number of successful load requests. |
| uploadFailed | Number of failed upload requests. All uncompleted requests are counted as failed, except those cancelled by using the **reset loader** command. Example reasons for failure include a request from an unauthorised user, or a missing file. |
| downloadFailed | Number of failed load requests. All uncompleted requests are counted as failed, except those cancelled by using the **reset loader** command. Example reasons for failure include a request from an unauthorised user, or an attempt to copy over an existing file. |
| uploadCancelled | Number of upload requests cancelled by using the **reset loader** command. |
| downloadCancelled | Number of load requests cancelled by using the **reset loader** command. |
| readFileRequest | Total number of read operations on local files. |
| writeFileRequests | Total number of write operations on local files. |
| readFileSuccess | Number of read successes. |

Table 44-2: Parameters in output of the show **ssh counter=scp** command (cont)

| | |
|---|---|
| writeFileSuccess | Number of write successes. |
| readFileFailed | Number of read failures. A read failure results in an upload failure. |
| writeFileFailed | Number of write failures. A write failure results in a load failure. |

Figure 44-3: Example output from the **show ssh counter=ssh** command

```
Secure Shell Counters:

  inOctets ................. 1057328      outOctets ................. 5745494
  rxPkt ..................... 821         txPkt ...................... 4169
  rxVersionID ................ 22         txVersionID ................ 22
  rxPktCheckFail ............. 0          txPktFail .................. 0
  rxPktDataLengthInvalid ..... 0
  rxPktDataOverflow .......... 0

  rxMSGDisconnect ............ 0          txMSGDisconnect ............ 22
  rxSMSGPublicKey ............ 17         txSMSGPublicKey ............ 4
  rxCMSGSessionKey ........... 3          txCMSGSessionKey ........... 17
  rxCMSGUser ................. 3          txCMSGUser ................. 17
  rxCMSGAuthRhosts ........... 0          txCMSGAuthRhosts ........... 0
  rxCMSGAuthRSA .............. 3          txCMSGAuthRSA .............. 0
  rxSMSGAuthRSAChallenge ..... 0          txSMSGAuthRSAChallenge ..... 2
  rxCMSGAuthRSAResponse ...... 2          txCMSGAuthRSAResponse ...... 0
  rxCMSGAuthPassword ......... 1          txCMSGAuthPassword ......... 17
  rxCMSGAuthRhostsRSA ........ 0          txCMSGAuthRhostsRSA ........ 0
  rxSMSGSuccess .............. 46         txSMSGSuccess .............. 7
  rxSMSGFailure .............. 18         txSMSGFailure .............. 4
  rxCMSGReqCompression ....... 0          txCMSGReqCompression ....... 0
  rxCMSGReqX11Forwarding ..... 0          txCMSGReqX11Forwarding ..... 0
  rxCMSGReqPortForwarding .... 0          txCMSGReqPortForwarding .... 0
  rxCMSGReqAgentForwarding ... 0          txCMSGReqAgentForwarding ... 0
  rxCMSGReqPty ............... 1          txCMSGReqPty ............... 1
  rxCMSGWindowSize ........... 0          txCMSGWindowSize ........... 0
  rxCMSGExecShell ............ 1          txCMSGExecShell ............ 1
  rxCMSGExecCmd .............. 2          txCMSGExecCmd .............. 0
  rxCMSGStdInData ............ 202        txCMSGStdInData ............ 119
  rxSMSGStdOutData ........... 499        txSMSGStdOutData ........... 303
  rxSMSGStdErrData ........... 0          txSMSGStdErrData ........... 0
  rxCMSGEOF .................. 0          txCMSGEOF .................. 0
  rxSMSGExitStatus ........... 0          txSMSGExitStatus ........... 0
  rxCMSGExitConfirmation ..... 0          txCMSGExitConfirmation ..... 0
  rxUnSupportedMsg ........... 0
  rxUnknownMsg ............... 0

  encodeSKSuccess ............ 34         decodeSKSuccess ............ 3
  encodeSKFail ............... 0          decodeSKFail ............... 0
  getHostKeyFail ............. 0          getServerKeyFail ........... 0
  serverKeyReGenerated ....... 0

  encodeRSAChallengeGood ..... 2          decodeRSAChallengeGood ..... 0
  encodeRSAChallengeFail ..... 0          decodeRSAChallengeFail ..... 0
  getUserKeyFail ............. 0

  encoConfigured  ............ 20         encoConfigureFail ......... 0
  encoDetached ............... 0          encoDead ................... 0
  encoEncodeStart ............ 4133       encoDecodeStart ............ 788
  encoEncoded ................ 4133       encoDecoded ................ 788
  encoEncodeFail ............. 0          encoDecodeFail ............. 0
  encoEncodeResetDone ........ 0          encoDecodeResetDone ........ 0
  encoEncodeResetFail ........ 0          encoDecodeResetFail ........ 0
  encoEncodeDiscard .......... 0          encoDecodeDiscard .......... 0
```

Table 44-3: Parameters in output of the **show ssh counter=ssh** command

| Parameter | Meaning |
|---|---|
| inOctets | Number of octets the router received. |
| outOctets | Number of octets the router transmitted. |
| rxPkt | Number of packets the router received. |
| txPkt | Number of packets the router transmitted. |
| rxVersionID | Number of version identification messages the router received. |
| txVersionID | Number of version identification messages the router transmitted. |
| txPktCheckFail | Number of packets the router received that contained an invalid checksum |
| rxPktFail | Number of packets the router does not transmit due to an error. |
| rxPktDataLengthInvalid | Number of packets the router received that contained an invalid data length |
| rxPktDataOverflow | Number of packets the router received that contained smaller data than the specified data size. |
| rxMSGDisconnect | Number of SSH_MSG_DISCONNECT messages to terminate a session that the router received. |
| txMSGDisconnect | Number of SSH_MSG_DISCONNECT messages to terminate a session that the router transmitted. |
| rxSMSGPublicKey | Number of SSH_SMSG_PUBLIC_KEY messages the client received containing the remote server's host key, server public key, supported ciphers and supported authentication methods. |
| txSMSGPublicKey | Number of SSH_SMSG_PUBLIC_KEY messages the server transmitted containing the server's host key, server public key, supported ciphers and supported authentication methods. |
| rxCMSGSessionKey | Number of SSH_CMSG_SESSION_KEY messages the server received containing the remote client's selected cipher, a copy of the 64-bit cookie sent by the server, the client's protocol flags and a session key encrypted with the server's host key and server key. |
| txCMSGSessionKey | Number of SSH_CMSG_SESSION_KEY messages the client transmitted, containing the client's selected cipher, a copy of the 64-bit cookie sent by the remote server, the client's protocol flags and a session key encrypted with the remote server's host key and server key. |
| rxCMSGUser | Number of SSH_CMSG_USER messages the server received containing a user name to login. |
| txCMSGUser | Number of SSH_CMSG_USER messages the client transmitted containing a user name to login. |
| rxCMSGAuthRhosts | Number of SSH_CMSG_AUTH_RHOSTS messages the server received containing the remote client's user name for .rhosts authentication. |
| txCMSGAuthRhosts | Number of SSH_CMSG_AUTH_RHOSTS messages the client tranmitted containing the client's user name for .rhosts authentication. |

Table 44-3: Parameters in output of the **show ssh counter=ssh** command (cont)

| Parameter | Meaning |
|---|---|
| rxCMSGAuthRSA | Number of SSH_CMSG_AUTH_RSA messages the server received, containing a remote client's public key for RSA authentication. |
| txCMSGAuthRSA | Number of SSH_CMSG_AUTH_RSA messages the client transmitted containing the public key for RSA authentication. |
| rxSMSGAuthRSAChallenge | Number of SSH_SMSG_AUTH_RSA_CHALLENGE messages the client received containing an encrypted challenge from the remote server. |
| txSMSGAuthRSAChallenge | Number of SSH_SMSG_AUTH_RSA_CHALLENGE messages the server transmitted containing an encrypted challenge for the remote client. |
| rxCMSGAuthRSAResponse | Number of SSH_CMSG_AUTH_RSA_RESPONSE messages the server received from a remote client containing the response to a challenge. |
| txCMSGAuthRSAResponse | Number of SSH_CMSG_AUTH_RSA_RESPONSE messages the client transmitted to a remote server, containing the response to a challenge. |
| rxCMSGAuthPassword | Number of SSH_CMSG_AUTH_PASSWORD messages the server received containing the remote client's plaintext password. |
| txCMSGAuthPassword | Number of SSH_CMSG_AUTH_PASSWORD messages the client transmitted containing the client's plaintext password. |
| rxCMSGAuthRhostsRSA | Number of SSH_CMSG_AUTH_RHOSTS_RSA messages the server received, containing the remote client's user name and public host key for .rhosts and RSA authentication. |
| txCMSGAuthRhostsRSA | Number of SSH_CMSG_AUTH_RHOSTS_RSA messages the client transmitted containing the client's user name and public host key for .rhosts and RSA authentication. |
| rxSMSGSuccess | Number of SSH_SMSG_SUCCESS messages the client received indicating a successful request. |
| txSMSGSuccess | Number of SSH_SMSG_SUCCESS messages the server transmitted indicating a successful request. |
| rxSMSGFailure | Number of SSH_SMSG_FAILURE messages the client received indicating a failed request. |
| txSMSGFailure | Number of SSH_SMSG_FAILURE messages the server transmitted indicating a failed request. |
| rxCMSGReqCompression | Number of SSH_CMSG_REQUEST_COMPRESSION messages the server received, requesting compression for the connection. |
| txCMSGReqCompression | Number of SSH_CMSG_REQUEST_COMPRESSION messages the client transmitted, requesting compression for the connection. |
| rxCMSGReqX11Forwarding | Number of SSH_CMSG_X11_REQUEST_FORWARDING messages the server received requesting forwarding of X11 connections. |
| txCMSGReqX11Forwarding | Number of SSH_CMSG_X11_REQUEST_FORWARDING messages the client transmitted requesting forwarding of X11 connections. |

Table 44-3: Parameters in output of the **show ssh counter=ssh** command (cont)

| Parameter | Meaning |
|---|---|
| rxCMSGReqPortForwarding | Number of SSH_CMSG_PORT_FORWARD_REQUEST messages the server received requesting forwarding of a TCP/IP port. |
| txCMSGReqPortForwarding | Number of SSH_CMSG_PORT_FORWARD_REQUEST messages the client transmitted requesting forwarding of a TCP/IP port. |
| rxCMSGReqAgentForwarding | Number of SSH_CMSG_AGENT_REQUEST_FORWARDING messages the server received requesting forwarding of the connection to the authentication agent. |
| txCMSGReqAgentForwarding | Number of SSH_CMSG_AGENT_REQUEST_FORWARDING messages the client transmitted requesting forwarding of the connection to the authentication agent. |
| rxCMSGReqPty | Number of SSH_CMSG_REQUEST_PTY messages the server received requesting a pseudo terminal device be allocated for this session. |
| txCMSGReqPty | Number of SSH_CMSG_REQUEST_PTY messages the client transmitted requesting a pseudo terminal device be allocated for this session. |
| rxCMSGWindowSize | Number of SSH_CMSG_WINDOW_SIZE messages the client transmitted specifying a new size for the client's window. |
| txCMSGWindowSize | Number of SSH_CMSG_WINDOW_SIZE messages the server received specifying a new size for the remote client's window. |
| rxCMSGExecShell | Number of SSH_CMSG_EXEC_SHELL messages the server received to create an interactive terminal session. |
| txCMSGExecShell | Number of SSH_CMSG_EXEC_SHELL messages the client transmitted to create an interactive terminal session. |
| rxCMSGExecCmd | Number of SSH_CMSG_EXEC_CMD messages the server received containing a command to execute. |
| txCMSGExecCmd | Number of SSH_CMSG_EXEC_CMD messages the client transmitted containing a command to execute. |
| rxCMSGStdInData | Number of SSH_CMSG_STDIN_DATA messages the client received containing data written to stdout by applications on the remote server. |
| txCMSGStdInData | Number of SSH_SMSG_STDOUT_DATA messages the client transmitted containing data from the client to be written to stdin on the remote server. |
| rxSMSGStdOutData | Number of SSH_SMSG_STDOUT_DATA messages the server received containing data from the remote client to be written to stdin on the server. |
| txSMSGStdOutData | Number of SSH_SMSG_STDOUT_DATA messages the server transmitted containing data written to stdout by applications on the server. |
| rxSMSGStdErrData | Number of SSH_SMSG_STDERR_DATA messages the client received containing data written to stderr by applications on the remote server. |
| txSMSGStdErrData | Number of SSH_SMSG_STDERR_DATA messages the server transmitted containing data written to stderr by applications on the server. |

Table 44-3: Parameters in output of the **show ssh counter=ssh** command (cont)

| Parameter | Meaning |
|---|---|
| rxCMSGEOF | Number of SSH_CMSG_EOF messages the server received from the remote client, indicating end of input from the remote client |
| txCMSGEOF | Number of SSH_CMSG_EOF messages the client transmitted to the remote server, indicating end of input from the client |
| rxSMSGExitStatus | Number of SSH_SMSG_EXITSTATUS messages the client received indicating that the shell or command has exited. |
| txSMSGExitStatus | Number of SSH_SMSG_EXITSTATUS messages the server transmitted indicating that the shell or command has exited. |
| rxCMSGExitConfirmation | Number of SSH_CMSG_EXIT_CONFIRMATION messages the server received in response to an SSH_SMSG_EXITSTATUS message. |
| txCMSGExitConfirmation | Number of SSH_CMSG_EXIT_CONFIRMATION messages the client transmitted in response to an SSH_SMSG_EXITSTATUS message. |
| rxUnSupportedMsg | Number of requests for unsupported options the server received. |
| rxUnknownMsg | Number of requests for unrecognised options the server received. |
| encodeSKSuccess | Number of double RSA encryptions of a session key. |
| decodeSKSuccess | Number of double RSA decryptions of a session key. |
| encodeSKFail | Number of failed attempts to encode a session key. |
| decodeSKFail | Number of failed attempts to decode a session key. |
| getHostKeyFail | Number of failed attempts to get the host key from the ENCO module. |
| getServerKeyFail | Number of failed attempts to get the server key from the ENCO module. |
| serverKeyReGenerated | Number of times a new random server key was created. |
| encodeRSAChallengeGood | Number of times the server encrypted a random challenge for RSA authentication. |
| decodeRSAChallengeGood | Number of times the server decrypted a random challenge for RSA authentication. |
| encodeRSAChallengeFail | Number of times the server failed to encrypt an RSA challenge. |
| decodeRSAChallengeFail | Number of times the server failed to decrypt an RSA challenge. |
| getUserKeyFail | Number of times the server failed to get the user RSA key from the ENCO module. |
| encoConfigured | Number of times an encryption channel was configured for a session. |
| encoConfigureFail | Number of times the server failed to configure an encryption channel. |
| encoDetached | Number of times the encryption channel of a session was destroyed. |
| encoDead | Number of times the encryption engine has failed. |

Table 44-3: Parameters in output of the **show ssh counter=ssh** command (cont)

| Parameter | Meaning |
|---|---|
| encoEncodeStart | Number of times an encode job was started on an ENCO channel. |
| encoDecodeStart | Number of times a decode job was started on an ENCO channel. |
| encoEncoded | Number of times an encode job completed. |
| encoDecoded | Number of times a decode job completed. |
| encoEncodeFail | Number of times an encode job failed to complete. |
| encoDecodeFail | Number of times a decode job failed to complete. |
| encoEncodeResetDone | Number of times an encryption channel has been reset. |
| encoDecodeResetDone | Number of times a decryption channel has been reset. |
| encoEncodeResetFail | Number of times the server failed to reset an encryption channel. |
| encoDecodeResetFail | Number of times the server failed to reset an decryption channel. |
| encoEncodeDiscard | Number of times an encode job has been discarded by the ENCO module. |
| encoDecodeDiscard | Number of times a decode job has been discarded by the ENCO module. |

**Example** To display the counters for both Secure Shell and Secure Copy, use the command:

```
sh ssh cou=all
```

**Related Commands**

**disable ssh server**
**enable ssh server**
**set ssh server**
**show ssh session**

# show ssh session

**Syntax** `SHow SSH SEssion[={ALL|SSH|SCP}]`

**Description** This command displays the status of Secure Shell and Secure Copy sessions currently active on the router, including both outbound sessions to another host and inbound sessions into the router.

| Parameter | Description | |
|---|---|---|
| SEssion | Whether session details for Secure Shell, Secure Copy, or both are displayed.<br>Default: **all** | |
| | ALL | The status of both Secure Shell and Secure Copy sessions are displayed (Figure 44-4, Table 44-4 Figure 44-5 on page 44-31, Table 44-5 on page 44-31). |
| | SCP | The status of Secure Copy sessions are displayed (Figure 44-4, Table 44-4). |
| | SSH | The status of both Secure Shell and Secure Copy sessions are displayed (Figure 44-5 on page 44-31, Table 44-5 on page 44-31). |

Figure 44-4: Example output from the **show ssh session=scp** command

```
SCP Sessions:

ID  Type   Operation  Filename         Filesize   State
-------------------------------------------------------------
 5  Server Download   86s-276.rez       4282204   RxData  8%
 6  Client Upload     test1.cfg          210372   TxData 34%
```

Table 44-4: Parameters in output of the **show ssh session=scp** command

| Parameter | Meaning | |
|---|---|---|
| ID | A unique identifier for each Secure Shell session. | |
| Type | The type of Secure Copy connection, either: | |
| | Server | The router is operating as an SCP server. |
| | Client | The router is operating as an SCP client. |
| Operation | The current type of file copying, either: | |
| | Download | The file is copying to the router. |
| | Upload | The file is copying to a remote machine. |
| Filename | The name of the file being copied. | |
| Filesize | The size of the file being copied. | |

Table 44-4: Parameters in output of the **show ssh session=scp** command (cont)

| Parameter | Meaning | |
|---|---|---|
| State | The current state of the SCP session, either: | |
| | Init | Session is initiated. |
| | Open | Server or client session started. |
| | Control | Awaiting a control message or a response to a control message. |
| | Ready | Ready to send or receive data. |
| | TxData | Transmitting data. This state will also show the progress of the file transfer as a percentage. |
| | RxData | Receiving data. This state will also show the progress of the file transfer as a percentage. |
| | WaitClosed | Awaiting a final message. |

Figure 44-5: Example output from the **show ssh session=ssh** command

```
Secure Shell Sessions:

ID Type    Dir Peer Address            User         State
-------------------------------------------------------------
 0 Listen  In  0.0.0.0                               Initial
 1 Listen  In  ::                                    Initial
 2 Shell   In  192.168.2.5             manager       Open
 3 Shell   Out 192.168.100.264         john          Open
 4 SCP     In  172.17.1.1              root          Authen
 5 SCP     Out 172.17.1.1              john          Request
```

Table 44-5: Parameters in output of the **show ssh session=ssh** command

| Parameter | Meaning | |
|---|---|---|
| ID | A unique identifier for each Secure Shell session. | |
| Type | The type of Secure Shell connection. | |
| | Listen | Listen session for incoming connections. |
| | Shell | Interactive shell connection. |
| | Cmd | Non-interactive remote command execution. |
| | SCP | Secure Copy connection. |
| Dir | The direction of the connection. | |
| | In | Connection made from a client to the Secure Shell server. |
| | Out | Connection made from the router to a remote Secure Shell server. |
| Peer Address | The IP address of the remote end of the Secure Shell connection. | |
| User | The username under which the connection is made, or "-" if user authentication has not yet taken place. For inbound sessions, the specified username identifies a user account on the router. For outbound sessions, it identifies a user on the remote host. | |

Table 44-5: Parameters in output of the **show ssh session=ssh** command (cont)

| Parameter | Meaning | |
|---|---|---|
| State | The current state of the Secure Shell connection: | |
| | Initial | Connection being initiated. |
| | Starting | Host-to-host authentication underway. |
| | Authen | User authentication in progress. |
| | Request | Request of session type in progress. |
| | Open | Session in progress. |

**Example** To display current Secure Copy sessions, use the command:

```
sh ssh se=scp
```

**Related Commands**

**delete ssh session**
**disable ssh server**
**enable ssh server**
**set ssh client**
**set ssh server**
**show ssh**

# show ssh user

**Syntax** `SHow SSH USER[=username]`

**Description** This command displays information about the users allowed to make connections to the Secure Shell server.

The **user** parameter specifies the user name being displayed. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and digits (0–9). The string cannot contain spaces.

If a user is not specified, summary information about all users is displayed (Figure 44-6, Table 44-6).

If a user is specified, details are displayed about that user (Figure 44-7, Table 44-7 on page 44-34).

Figure 44-6: Example output from the **show ssh user** command

```
Secure Shell User List

User     IpAddr                          Auth  KeyId  Status
-------------------------------------------------------------
test4    fe80:230:84ff:fe0e:263e         Pass      0  enabled
test2    fe80:230:84ff:fe0e:263d         Pass      0  enabled
secoff   0.0.0.0                         RSA       5  enabled
800      0.0.0.0                         RSA       4  enabled
admin    0.0.0.0                         RSA       7  disabled
john     192.168.2.1                     Pass      0  enabled
```

Table 44-6: Parameters in output of the **show ssh user** command

| Parameter | Meaning |
|---|---|
| User | The username for the user. |
| IpAddr | The IP address from which the user may login. |
| Auth | The type of authentication required for the user; either Pass or RSA. |
| KeyId | The key ID of the ENCO key used for RSA authentication. |
| Status | Whether the status of the user is enabled or disabled. |

Figure 44-7: Example output from the **show ssh user** command for a specific user

```
User..................... john
Status................... Enabled
Authorisation method..... Password
RSA key ID............... 0
Shell.................... Yes
IpAddress................ 192.168.2.1
Mask..................... 255.255.255.255
Failed Logins............ 0
```

Table 44-7: Parameters in output of the **show ssh user** command for a specific user

| Parameter | Meaning |
|---|---|
| User | The username for the user. |
| Status | Whether the status of the user is enabled or disabled. |
| Authorisation method | The type of authentication required for the user; either Password or RSA. |
| RSA key ID | The key ID of the ENCO key used for RSA authentication. |
| Shell | Whether the user has Shell access. |
| IpAddress | The IP address from which the user may login. |
| Mask | The network mask for the IP address. |
| Failed Logins | The total number of failed login attempts for the user. |

**Example** To display the Secure Shell user list, use the command:

```
sh ssh user
```

To display specific information about the Secure Shell user named "john", use the command:

```
sh ssh user=john
```

**Related Commands**

**add ssh user**
**delete ssh user**
**disable ssh user**
**enable ssh user**
**set ssh user**

# ssh

**Syntax**

```
SSH {ipadd|ipv6add[%interface]|host} USER=username
   {PASSword=password|KEYid=key-id} [COMmand=string]
```

**Description**

This command initiates a Secure Shell connection to the specified host.

A Secure Shell request to an IPv6 link-local address requires interface information as well as the address, because a single link-local address can belong to several interfaces. To open a secure shell session to a link-local address, specify the interface out which the router is to send the secure shell request, as well as the address to which the router is to send the request (Figure 31-2 on page 31-19 in Chapter 31, Internet Protocol version 6 (IPv6)). For example:

```
ssh fe80:7c27%vlan1 user=Admin password=18Again
```

| Parameter | Description |
|---|---|
| *ipadd* | an IP address in dotted decimal notation.<br>Default: no default |
| *ipv6add* | A valid IPv6 address in hexadecimal notation.<br>Default: no default |
| *interface* | The interface the SSH request is sent out, for a request to SSH to an IPv6 link-local address e.g. eth0. Separate the address and interface with a % sign.<br>Default: no default |
| *host* | A full-domain name of a host, a host nickname created with the **add ip host** command on page 22-79 of Chapter 22, Internet Protocol (IP), or a host name in the same domain.<br>Default: no default |
| USER | The name of a user on the remote host. The *username* is a string 1 to 15 characters long. Valid characters are uppercase and lowercase letters, and digits (0-9). The string cannot contain spaces.<br>Default: no default |
| PASSword | The password for the Secure Shell password authentication. The *password* is a string 1 to 31 characters long. Valid characters are any printable character. If the string contains spaces, put double quotes around it.<br>Default: no default |
| KEYid | The RSA key being used to authenticate the user. This parameter enables Secure Shell RSA authentication. The specified key must exist on both the client (RSA Private Key) and the server (RSA Public Key). The *key-id* is a decimal number.<br>Default: no default |
| COMmand | A command to be executed on the remote host or router. If this parameter is present, the command is executed on the remote system and the connection is closed. If **command** is not specified, an interactive terminal session is created. The *string* can be 1 to 80 characters long. Valid characters are any printable characters. If the string contains spaces, put double quotes around it.<br>Default: no default |

**Examples** To connect to the remote host with IP address 172.16.1.5 as user "Admin" with password "l8Again", use the command:

```
ssh 172.16.1.5 user=Admin pass=l8Again
```

**Related Commands** **set ssh client**
**show ssh session**