Secure Shell Configuration Guide, Cisco IOS XE 17 (Cisco NCS 520 Series)

First Published: 2019-11-26

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

© 2020 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **Secure Copy** 1

Finding Feature Information 1

Prerequisites for Secure Copy 1

Information About Secure Copy 2

How Secure Copy Works 2

How to Configure Secure Copy 2

Configuring Secure Copy 2

Configuration Examples for Secure Copy 4

Example: Secure Copy Configuration Using Local Authentication 4

Example SCP Server-Side Configuration Using Network-Based Authentication 4

Additional References 4

Glossary 5

CHAPTER 2 **Secure Shell Version 2 Support** 7

Finding Feature Information 7

Prerequisites for Secure Shell Version 2 Support 7

Restrictions for Secure Shell Version 2 Support 8

Information About Secure Shell Version 2 Support 8

Secure Shell Version 2 8

Secure Shell Version 2 Enhancements 9

Secure Shell Version 2 Enhancements for RSA Keys 9

SNMP Trap Generation 10

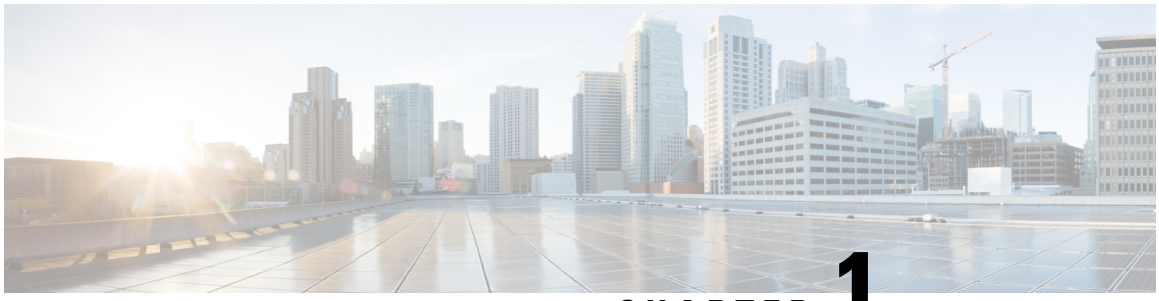
SSH Keyboard Interactive Authentication 10

How to Configure Secure Shell Version 2 Support 11

Configuring a Device for SSH Version 2 Using a Hostname and Domain Name 11

Configuring a Device for SSH Version 2 Using RSA Key Pairs 12

Configuring the Cisco SSH Server to Perform RSA-Based User Authentication 13
Configuring the Cisco IOS SSH Client to Perform RSA-Based Server Authentication 15
Starting an Encrypted Session with a Remote Device 17
Troubleshooting Tips 17
Enabling Secure Copy Protocol on the SSH Server 17
Troubleshooting Tips 19
Verifying the Status of the Secure Shell Connection 19
Verifying the Secure Shell Status 21
Monitoring and Maintaining Secure Shell Version 2 22
Configuration Examples for Secure Shell Version 2 Support 25
Example: Configuring Secure Shell Version 2 25
Example: Configuring Secure Shell Versions 1 and 2 25
Example: Starting an Encrypted Session with a Remote Device 25
Example: Configuring Server-Side SCP 25
Example: Setting an SNMP Trap 26
Examples: SSH Keyboard Interactive Authentication 26
Example: Enabling Client-Side Debugs 26
Example: Enabling ChPass with a Blank Password Change 27
Example: Enabling ChPass and Changing the Password on First Login 27
Example: Enabling ChPass and Expiring the Password After Three Logins 27
Example: SNMP Debugging 28
Examples: SSH Debugging Enhancements 28
Where to Go Next 30
Additional References for Secure Shell Version 2 Support 30



CHAPTER 1

Secure Copy

The Secure Copy (SCP) feature provides a secure and authenticated method for copying device configurations or device image files. SCP relies on Secure Shell (SSH), an application and protocol that provide a secure replacement for the Berkeley r-tools suite (Berkeley university's own set of networking applications). This document provides the procedure to configure a Cisco device for SCP server-side functionality.

- Finding Feature Information, on page 1
- Prerequisites for Secure Copy, on page 1
- Information About Secure Copy, on page 2
- How to Configure Secure Copy, on page 2
- Configuration Examples for Secure Copy, on page 4
- Additional References, on page 4
- Glossary, on page 5

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Secure Copy

- Before enabling Secure Copy (SCP), you must correctly configure Secure Shell (SSH), authentication, and authorization on the device.
- Because SCP relies on SSH for its secure transport, the device must have a Rivest, Shamir, and Adelman (RSA) key pair.

Information About Secure Copy

How Secure Copy Works

The behavior of Secure Copy (SCP) is similar to that of remote copy (RCP), which comes from the Berkeley r-tools suite (Berkeley university's own set of networking applications), except that SCP relies on Secure Shell (SSH) for security. In addition, SCP requires that authentication, authorization, and accounting (AAA) authorization be configured so that the device can determine whether the user has the correct privilege level.

SCP allows a user only with a privilege level of 15 to copy any file that exists in the Cisco IOS File System (IFS) to and from a device by using the **copy** command. An authorized administrator may also perform this action from a workstation.

Note Enable the SCP option while using the pscp.exe file with the Cisco software.

How to Configure Secure Copy

Configuring Secure Copy

To configure a Cisco device for Secure Copy (SCP) server-side functionality, perform the following steps.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **aaa new-model**

Example:

```
Device(config)# aaa new-model
```

Sets AAA authentication at login.

Step 4 **aaa authentication login** {**default** | *list-name*} *method1* [*method2...*]

Example:

```
Device(config)# aaa authentication login default group tacacs+
```

Enables the AAA access control system.

Step 5 **username** *name* [**privilege** *level*] **password** *encryption-type encrypted-password*

Example:

```
Device(config)# username superuser privilege 2 password 0 superpassword
```

Establishes a username-based authentication system.

Note You may omit this step if a network-based authentication mechanism, such as TACACS+ or RADIUS, has been configured.

Step 6 **ip scp server enable**

Example:

```
Device(config)# ip scp server enable
```

Enables SCP server-side functionality.

Step 7 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Step 8 **show running-config**

Example:

```
Device# show running-config
```

(Optional) Displays the SCP server-side functionality.

Step 9 **debug ip scp**

Example:

```
Device# debug ip scp
```

(Optional) Troubleshoots SCP authentication problems.

Configuration Examples for Secure Copy

Example: Secure Copy Configuration Using Local Authentication

The following example shows how to configure the server-side functionality of Secure Copy (SCP). This example uses a locally defined username and password.

```
! AAA authentication and authorization must be configured properly in order for SCP to work.
aaa new-model
aaa authentication login default local
aaa authorization exec default local
username user1 privilege 15 password 0 lab
! SSH must be configured and functioning properly.
ip scp server enable
```

Example SCP Server-Side Configuration Using Network-Based Authentication

The following example shows how to configure the server-side functionality of SCP using a network-based authentication mechanism:

```
! AAA authentication and authorization must be configured properly for SCP to work.
aaa new-model
aaa authentication login default group tacacs+
aaa authorization exec default group tacacs+
! SSH must be configured and functioning properly.
ip ssh time-out 120
ip ssh authentication-retries 3
ip scp server enable
```

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
Secure Shell Version 1 and 2 support	*Secure Shell Configuration Guide*
Authentication and authorization commands	Cisco IOS Security Command Reference: Commands A to C
Configuring authentication and authorization	*Authentication, Authorization, and Accounting Configuration Guide*

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

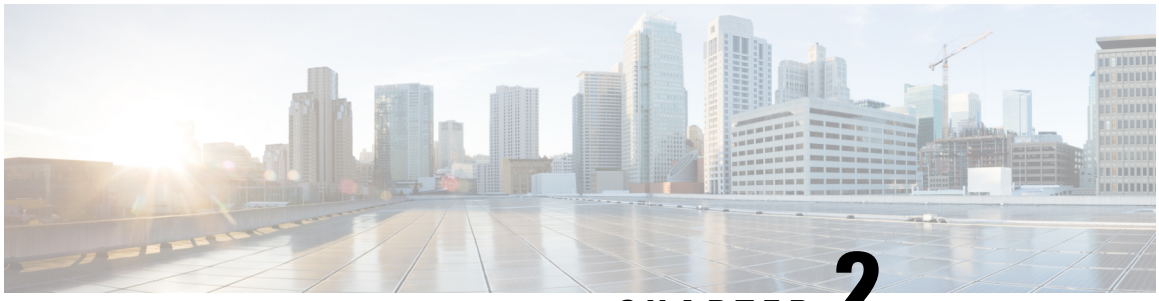
Glossary

AAA—authentication, authorization, and accounting. A framework of security services that provide the method for identifying users (authentication), for remote access control (authorization), and for collecting and sending security server information used for billing, auditing, and reporting (accounting).

RCP—remote copy. Relies on Remote Shell (Berkeley r-tools suite) for security; RCP copies files such as device images and startup configurations to and from devices.

SCP—secure copy. Relies on SSH for security; SCP support allows secure and authenticated copying of anything that exists in the Cisco IOS File System (IFS). SCP is derived from RCP.

SSH—Secure Shell. An application and protocol that provide a secure replacement for the Berkeley r-tools suite. The protocol secures the sessions using standard cryptographic mechanisms, and the application can be used similar to the Berkeley rexec and rsh tools. SSH Version 1 is implemented in the Cisco software.



CHAPTER 2

Secure Shell Version 2 Support

The Secure Shell Version 2 Support feature allows you to configure Secure Shell (SSH) Version 2. (SSH Version 1 support was implemented in an earlier Cisco software release.) SSH runs on top of a reliable transport layer and provides strong authentication and encryption capabilities. The only reliable transport that is defined for SSH is TCP. SSH provides a means to securely access and securely execute commands on another computer over a network. The Secure Copy Protocol (SCP) feature that is provided with SSH allows for the secure transfer of files.

- Finding Feature Information, on page 7
- Prerequisites for Secure Shell Version 2 Support, on page 7
- Restrictions for Secure Shell Version 2 Support, on page 8
- Information About Secure Shell Version 2 Support, on page 8
- How to Configure Secure Shell Version 2 Support, on page 11
- Configuration Examples for Secure Shell Version 2 Support, on page 25
- Where to Go Next, on page 30
- Additional References for Secure Shell Version 2 Support, on page 30

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the feature information table.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Prerequisites for Secure Shell Version 2 Support

- Before configuring SSH, ensure that the required image is loaded on your device. The SSH server requires you to have a k9 (Triple Data Encryption Standard [3DES]) software image depending on your release.
- You have to use a SSH remote device that supports SSH Version 2 and connect to a Cisco device.
- SCP relies on authentication, authorization, and accounting (AAA) to function correctly. Therefore, AAA must be configured on the device to enable the secure copy protocol on the SSH Server.

Note The SSH Version 2 server and the SSH Version 2 client are supported on your Cisco software, depending on your release. (The SSH client runs both the SSH Version 1 protocol and the SSH Version 2 protocol. The SSH client is supported in k9 images depending on your release.)

For more information about downloading a software image, refer to the *Configuration Fundamentals Configuration Guide*.

Restrictions for Secure Shell Version 2 Support

- Secure Shell (SSH) servers and SSH clients are supported in Triple Data Encryption Standard (3DES) software images.
- Execution Shell, remote command execution, and Secure Copy Protocol (SCP) are the only applications supported.
- Rivest, Shamir, and Adleman (RSA) key generation is an SSH server-side requirement. Devices that act as SSH clients need not generate RSA keys.
- The RSA key pair size must be greater than or equal to 768 bits.
- The following features are not supported:
  - Port forwarding
  - Compression

Information About Secure Shell Version 2 Support

Secure Shell Version 2

The Secure Shell Version 2 Support feature allows you to configure SSH Version 2.

The configuration for the SSH Version 2 server is similar to the configuration for SSH Version 1. The **ip ssh version** command defines the SSH version to be configured. If you do not configure this command, SSH by default runs in compatibility mode; that is, both SSH Version 1 and SSH Version 2 connections are honored.

Note SSH Version 1 is a protocol that has never been defined in a standard. If you do not want your device to fall back to the undefined protocol (Version 1), you should use the **ip ssh version** command and specify Version 2.

The **ip ssh rsa keypair-name** command enables an SSH connection using the Rivest, Shamir, and Adleman (RSA) keys that you have configured. Previously, SSH was linked to the first RSA keys that were generated (that is, SSH was enabled when the first RSA key pair was generated). This behavior still exists, but by using the **ip ssh rsa keypair-name** command, you can overcome this behavior. If you configure the **ip ssh rsa keypair-name** command with a key pair name, SSH is enabled if the key pair exists or SSH will be enabled

if the key pair is generated later. If you use this command to enable SSH, you are not forced to configure a hostname and a domain name, which was required in SSH Version 1 of the Cisco software.

Note The login banner is supported in SSH Version 2, but it is not supported in Secure Shell Version 1.

Secure Shell Version 2 Enhancements

The SSH Version 2 Enhancements feature includes a number of additional capabilities such as supporting Virtual Routing and Forwarding (VRF)-Aware SSH, SSH debug enhancements, and Diffie-Hellman (DH) group exchange support.

Note The VRF-Aware SSH feature is supported depending on your release.

The Cisco SSH implementation has traditionally used 768-bit modulus, but with an increasing need for higher key sizes to accommodate DH Group 14 (2048 bits) and Group 16 (4096 bits) cryptographic applications, a message exchange between the client and the server to establish the favored DH group becomes necessary. The **ip ssh dh min size** command configures the modulus size on the SSH server. In addition to this, the **ssh** command was extended to add VRF awareness to the SSH client-side functionality through which the VRF instance name in the client is provided with the IP address to look up the correct routing table and establish a connection.

Debugging was enhanced by modifying SSH debug commands. The **debug ip ssh** command was extended to simplify the debugging process. Before the simplification of the debugging process, this command printed all debug messages related to SSH regardless of what was specifically required. The behavior still exists, but if you configure the **debug ip ssh** command with a keyword, messages are limited to information specified by the keyword.

Secure Shell Version 2 Enhancements for RSA Keys

Cisco SSH Version 2 supports keyboard-interactive and password-based authentication methods. The SSH Version 2 Enhancements for RSA Keys feature also supports RSA-based public key authentication for the client and the server.

User authentication—RSA-based user authentication uses a private/public key pair associated with each user for authentication. The user must generate a private/public key pair on the client and configure a public key on the Cisco SSH server to complete the authentication.

An SSH user trying to establish credentials provides an encrypted signature using the private key. The signature and the user's public key are sent to the SSH server for authentication. The SSH server computes a hash over the public key provided by the user. The hash is used to determine if the server has a matching entry. If a match is found, an RSA-based message verification is performed using the public key. Hence, the user is authenticated or denied access based on the encrypted signature.

Server authentication—While establishing an SSH session, the Cisco SSH client authenticates the SSH server by using the server host keys available during the key exchange phase. SSH server keys are used to identify the SSH server. These keys are created at the time of enabling SSH and must be configured on the client.

For server authentication, the Cisco SSH client must assign a host key for each server. When the client tries to establish an SSH session with a server, the client receives the signature of the server as part of the key exchange message. If the strict host key checking flag is enabled on the client, the client checks if it has the host key entry corresponding to the server. If a match is found, the client tries to validate the signature by using the server host key. If the server is successfully authenticated, the session establishment continues; otherwise, it is terminated and displays a "Server Authentication Failed" message.

Note Storing public keys on a server uses memory; therefore, the number of public keys configurable on an SSH server is restricted to ten users, with a maximum of two public keys per user.

Note RSA-based user authentication is supported by the Cisco server, but Cisco clients cannot propose public key as an authentication method. If the Cisco server receives a request from an open SSH client for RSA-based authentication, the server accepts the authentication request.

Note For server authentication, configure the RSA public key of the server manually and configure the **ip ssh stricthostkeycheck** command on the Cisco SSH client.

SNMP Trap Generation

Depending on your release, Simple Network Management Protocol (SNMP) traps are generated automatically when an SSH session terminates if the traps have been enabled and SNMP debugging has been enabled. For information about enabling SNMP traps, see the "Configuring SNMP Support" module in the *SNMP Configuration Guide*.

Note When you configure the **snmp-server host** command, the IP address must be the address of the PC that has the SSH (telnet) client and that has IP connectivity to the SSH server. For an example of an SNMP trap generation configuration, see the *Example: Setting an SNMP Trap* section.

You must also enable SNMP debugging using the **debug snmp packet** command to display the traps. The trap information includes information such as the number of bytes sent and the protocol that was used for the SSH session. For an example of SNMP debugging, see the *Example: Setting an SNMP Trap* section.

SSH Keyboard Interactive Authentication

The SSH Keyboard Interactive Authentication feature, also known as Generic Message Authentication for SSH, is a method that can be used to implement different types of authentication mechanisms. Basically, any currently supported authentication method that requires only user input can be performed with this feature. The feature is automatically enabled.

The following methods are supported:

- Password

- SecurID and hardware tokens printing a number or a string in response to a challenge sent by the server
- Pluggable Authentication Module (PAM)
- S/KEY (and other One-Time-Pads)

For examples of various scenarios in which the SSH Keyboard Interactive Authentication feature has been automatically enabled, see the "Examples: SSH Keyboard Interactive Authentication, on page 26" section.

How to Configure Secure Shell Version 2 Support

Configuring a Device for SSH Version 2 Using a Hostname and Domain Name

- **enable**

  Enables privileged EXEC mode.

  - Enter your password if prompted.

  ```
  Device> enable
  ```

- **configure terminal**

  Enters global configuration mode.

  ```
  Device# configure terminal
  ```

- **hostname** *name*

  Configures a hostname for your device.

  ```
  Device(config)# hostname cisco520
  ```

- **ip domain-name** *name*

  Configures a domain name for your device.

  ```
  cisco520(config)# ip domain-name example.com
  ```

- **crypto key generate rsa**

  Enables the SSH server for local and remote authentication.

  ```
  cisco520(config)# crypto key generate rsa
  ```

- **ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*]

  (Optional) Configures SSH control variables on your device.

  ```
  cisco520(config)# ip ssh time-out 120
  ```

- **ip ssh version** [**2**]

  (Optional) Specifies the version of SSH to be run on your device.

  ```
  cisco520(config)# ip ssh version 2
  ```

- **exit**

  Exits global configuration mode and enters privileged EXEC mode.

- Use **no hostname** command to return to the default host.

```
cisco520(config)# exit
```

Configuring a Device for SSH Version 2 Using RSA Key Pairs

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **ip ssh rsa keypair-name** *keypair-name*

Example:

```
Device(config)# ip ssh rsa keypair-name sshkeys
```

Specifies the RSA key pair to be used for SSH.

Note A Cisco device can have many RSA key pairs.

Step 4 **crypto key generate rsa usage-keys label** *key-label* **modulus** *modulus-size*

Example:

```
Device(config)# crypto key generate rsa usage-keys label sshkeys modulus 768
```

Enables the SSH server for local and remote authentication on the device.

- For SSH Version 2, the modulus size must be at least 768 bits.

Note To delete the RSA key pair, use the **crypto key zeroize rsa** command. When you delete the RSA key pair, you automatically disable the SSH server.

Step 5 **ip ssh** [**time-out** *seconds* | **authentication-retries** *integer*]

Example:

```
Device(config)# ip ssh time-out 12
```

Configures SSH control variables on your device.

Step 6 **ip ssh version 2**

Example:

```
Device(config)# ip ssh version 2
```

Specifies the version of SSH to be run on the device.

Step 7 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and enters privileged EXEC mode.

Configuring the Cisco SSH Server to Perform RSA-Based User Authentication

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **hostname** *name*

Example:

```
Device(config)# hostname host1
```

Specifies the hostname.

Step 4 **ip domain-name** *name*

Example:

```
host1(config)# ip domain-name name1
```

Defines a default domain name that the Cisco software uses to complete unqualified hostnames.

Step 5 **crypto key generate rsa**

Example:

```
host1(config)# crypto key generate rsa
```

Generates RSA key pairs.

Step 6 **ip ssh pubkey-chain**

Example:

```
host1(config)# ip ssh pubkey-chain
```

Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.

- The user authentication is successful if the RSA public key stored on the server is verified with the public or the private key pair stored on the client.

Step 7 **username** *username*

Example:

```
host1(conf-ssh-pubkey)# username user1
```

Configures the SSH username and enters public-key user configuration mode.

Step 8 **key-string**

Example:

```
host1(conf-ssh-pubkey-user)# key-string
```

Specifies the RSA public key of the remote peer and enters public-key data configuration mode.

Note You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.

Step 9 **key-hash** *key-type key-name*

Example:

```
host1(conf-ssh-pubkey-data)# key-hash ssh-rsa key1
```

(Optional) Specifies the SSH key type and version.

- The key type must be ssh-rsa for the configuration of private public key pairs.
- This step is optional only if the **key-string** command is configured.
- You must configure either the **key-string** command or the **key-hash** command.

Note You can use a hashing software to compute the hash of the public key string, or you can also copy the hash value from another Cisco device. Entering the public key data using the **key-string** command is the preferred way to enter the public key data for the first time.

Step 10 **end**

Example:

```
host1(conf-ssh-pubkey-data)# end
```

Exits public-key data configuration mode and returns to privileged EXEC mode.

- Use **no hostname** command to return to the default host.

Configuring the Cisco IOS SSH Client to Perform RSA-Based Server Authentication

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **hostname** *name*

Example:

```
Device(config)# hostname host1
```

Specifies the hostname.

Step 4 **ip domain-name** *name*

Example:

```
host1(config)# ip domain-name name1
```

Defines a default domain name that the Cisco software uses to complete unqualified hostnames.

Step 5 **crypto key generate rsa**

Example:

```
host1(config)# crypto key generate rsa
```

Generates RSA key pairs.

Step 6 **ip ssh pubkey-chain**

Example:

```
host1(config)# ip ssh pubkey-chain
```

Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode.

Step 7 **server** *server-name*

Example:

```
host1(conf-ssh-pubkey)# server server1
```

Enables the SSH server for public-key authentication on the device and enters public-key server configuration mode.

Step 8 **key-string**

Example:

```
host1(conf-ssh-pubkey-server)# key-string
```

Specifies the RSA public-key of the remote peer and enters public key data configuration mode.

Note You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.

Step 9 **exit**

Example:

```
host1(conf-ssh-pubkey-data)# exit
```

Exits public-key data configuration mode and enters public-key server configuration mode.

Step 10 **key-hash** *key-type key-name*

Example:

```
host1(conf-ssh-pubkey-server)# key-hash ssh-rsa key1
```

(Optional) Specifies the SSH key type and version.

- The key type must be ssh-rsa for the configuration of private/public key pairs.
- This step is optional only if the **key-string** command is configured.
- You must configure either the **key-string** command or the **key-hash** command.

Note You can use a hashing software to compute the hash of the public key string, or you can copy the hash value from another Cisco device. Entering the public key data using the **key-string** command is the preferred way to enter the public key data for the first time.

Step 11 **end**

Example:

```
host1(conf-ssh-pubkey-server)# end
```

Exits public-key server configuration mode and returns to privileged EXEC mode.

Step 12 **configure terminal**

Example:

```
host1# configure terminal
```

Enters global configuration mode.

Step 13 **ip ssh stricthostkeycheck**

Example:

```
host1(config)# ip ssh stricthostkeycheck
```

Ensures that server authentication takes place.

- The connection is terminated in case of a failure.
- Use **no hostname** command to return to the default host.

Starting an Encrypted Session with a Remote Device

Note The device with which you want to connect must support a Secure Shell (SSH) server that has an encryption algorithm that is supported in Cisco software. Also, you need not enable your device. SSH can be run in disabled mode.

ssh [**-v** {**1** | **2**} | **-c** {**aes128-ctr** | **aes192-ctr** | **aes256-ctr** | **aes128-cbc** | **3des** | **aes192-cbc** | **aes256-cbc**} | **-l** *user-id* | **-l** *user-id***:***vrf-name number ip-address ip-address* | **-l** *user-id***:***rotary number ip-address* | **-m** {**hmac-md5-128** | **hmac-md5-96** | **hmac-sha1-160** | **hmac-sha1-96**} | **-o numberofpasswordprompts** *n* | **-p** *port-num*] {*ip-addr* | *hostname*} [**command** | **-vrf**]

Example:

```
Device# ssh -v 2 -c aes256-ctr -m hmac-sha1-96 -l user2 10.76.82.24
```

Starts an encrypted session with a remote networking device.

Troubleshooting Tips

The **ip ssh version** command can be used for troubleshooting your SSH configuration. By changing versions, you can determine the SSH version that has a problem.

Enabling Secure Copy Protocol on the SSH Server

Note The following task configures the server-side functionality for SCP. This task shows a typical configuration that allows the device to securely copy files from a remote workstation.

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **configure terminal**

Example:

```
Device# configure terminal
```

Enters global configuration mode.

Step 3 **aaa new-model**

Example:

```
Device(config)# aaa new-model
```

Enables the AAA access control model.

Step 4 **aaa authentication login default local**

Example:

```
Device(config)# aaa authentication login default local
```

Sets AAA authentication at login to use the local username database for authentication.

Step 5 **aaa authorization exec defaultlocal**

Example:

```
Device(config)# aaa authorization exec default local
```

Sets the parameters that restrict user access to a network, runs the authorization to determine if the user ID is allowed to run an EXEC shell, and specifies that the system must use the local database for authorization.

Step 6 **username***name* **privilege** *privilege-level* **password** *password*

Example:

```
Device(config)# username samplename privilege 15 password password1
```

Establishes a username-based authentication system, and specifies the username, privilege level, and an unencrypted password.

Note The minimum value for the *privilege-level* argument is 15. A privilege level of less than 15 results in the connection closing.

Step 7 **ip ssh time-out***seconds*

Example:

```
Device(config)# ip ssh time-out 120
```

Sets the time interval (in seconds) that the device waits for the SSH client to respond.

Step 8 **ip ssh authentication-retries** *integer*

Example:

```
Device(config)# ip ssh authentication-retries 3
```

Sets the number of authentication attempts after which the interface is reset.

Step 9 **ip scpserverenable**

Example:

```
Device(config)# ip scp server enable
```

Enables the device to securely copy files from a remote workstation.

Step 10 **exit**

Example:

```
Device(config)# exit
```

Exits global configuration mode and returns to privileged EXEC mode.

Step 11 **debug ip scp**

Example:

```
Device# debug ip scp
```

(Optional) Provides diagnostic information about SCP authentication problems.

Troubleshooting Tips

To troubleshoot SCP authentication problems, use the **debug ip scp**command.

Verifying the Status of the Secure Shell Connection

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ssh**

Example:

```
Device# show ssh
```

Displays the status of SSH server connections.

Step 3 **exit**

Example:

```
Device# exit
```

Exits privileged EXEC mode and returns to user EXEC mode.

Examples

The following sample output from the **show ssh** command displays status of various SSH Version 1 and Version 2 connections for Version 1 and Version 2 connections:

```
------------------------------------------------------------------------
Device# show ssh

Connection      Version Encryption      State                   Username
 0              1.5     3DES            Session started         lab
Connection Version Mode Encryption  Hmac                State
Username
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
------------------------------------------------------------------------
```

The following sample output from the **show ssh** command displays status of various SSH Version 1 and Version 2 connections for a Version 2 connection with no Version 1 connection:

```
------------------------------------------------------------------------
Device# show ssh

Connection Version Mode Encryption  Hmac                State
Username
1          2.0     IN   aes128-cbc  hmac-md5     Session started       lab
1          2.0     OUT  aes128-cbc  hmac-md5     Session started       lab
%No SSHv1 server connections running.
------------------------------------------------------------------------
```

The following sample output from the **show ssh** command displays status of various SSH Version 1 and Version 2 connections for a Version 1 connection with no Version 2 connection:

```
------------------------------------------------------------------------
Device# show ssh

Connection      Version Encryption      State                   Username
 0              1.5     3DES            Session started         lab
%No SSHv2 server connections running.
------------------------------------------------------------------------
```

Verifying the Secure Shell Status

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **show ip ssh**

Example:

```
Device# show ip ssh
```

Displays the version and configuration data for SSH.

Step 3 **exit**

Example:

```
Device# exit
```

Exits privileged EXEC mode and returns to user EXEC mode.

Examples

The following sample output from the **show ip ssh** command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries for Version 1 and Version 2 connections:

```
------------------------------------------------------------------------
Device# show ip ssh

SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
------------------------------------------------------------------------
```

The following sample output from the **show ip ssh** command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries for a Version 2 connection with no Version 1 connection:

```
------------------------------------------------------------------------
Device# show ip ssh

SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3
------------------------------------------------------------------------
```

The following sample output from the **show ip ssh** command displays the version of SSH that is enabled, the authentication timeout values, and the number of authentication retries for a Version 1 connection with no Version 2 connection:

```
------------------------------------------------------------------------
Device# show ip ssh

3d06h: %SYS-5-CONFIG_I: Configured from console by console
SSH Enabled - version 1.5
Authentication timeout: 120 secs; Authentication retries: 3
------------------------------------------------------------------------
```

Monitoring and Maintaining Secure Shell Version 2

Step 1 **enable**

Example:

```
Device> enable
```

Enables privileged EXEC mode.

- Enter your password if prompted.

Step 2 **debug ip ssh**

Example:

```
Device# debug ip ssh
```

Enables debugging of SSH.

Step 3 **debug snmp packet**

Example:

```
Device# debug snmp packet
```

Enables debugging of every SNMP packet sent or received by the device.

Example

The following sample output from the **debug ip ssh** command shows the connection is an SSH Version 2 connection:

```
Device# debug ip ssh

00:33:55: SSH1: starting SSH control process
00:33:55: SSH1: sent protocol version id SSH-1.99-Cisco-1.25
00:33:55: SSH1: protocol version id is - SSH-2.0-OpenSSH_2.5.2p2
00:33:55: SSH2 1: send: len 280 (includes padlen 4)
00:33:55: SSH2 1: SSH2_MSG_KEXINIT sent
```

```
00:33:55: SSH2 1: ssh_receive: 536 bytes received
00:33:55: SSH2 1: input: packet len 632
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: ssh_receive: 96 bytes received
00:33:55: SSH2 1: partial packet 8, need 624, maclen 0
00:33:55: SSH2 1: input: padlen 11
00:33:55: SSH2 1: received packet type 20
00:33:55: SSH2 1: SSH2_MSG_KEXINIT received
00:33:55: SSH2: kex: client->server aes128-cbc hmac-md5 none
00:33:55: SSH2: kex: server->client aes128-cbc hmac-md5 none
00:33:55: SSH2 1: expecting SSH2_MSG_KEXDH_INIT
00:33:55: SSH2 1: ssh_receive: 144 bytes received
00:33:55: SSH2 1: input: packet len 144
00:33:55: SSH2 1: partial packet 8, need 136, maclen 0
00:33:55: SSH2 1: input: padlen 5
00:33:55: SSH2 1: received packet type 30
00:33:55: SSH2 1: SSH2_MSG_KEXDH_INIT received
00:33:55: SSH2 1: signature length 111
00:33:55: SSH2 1: send: len 384 (includes padlen 7)
00:33:55: SSH2: kex_derive_keys complete
00:33:55: SSH2 1: send: len 16 (includes padlen 10)
00:33:55: SSH2 1: newkeys: mode 1
00:33:55: SSH2 1: SSH2_MSG_NEWKEYS sent
00:33:55: SSH2 1: waiting for SSH2_MSG_NEWKEYS
00:33:55: SSH2 1: ssh_receive: 16 bytes received
00:33:55: SSH2 1: input: packet len 16
00:33:55: SSH2 1: partial packet 8, need 8, maclen 0
00:33:55: SSH2 1: input: padlen 10
00:33:55: SSH2 1: newkeys: mode 0
00:33:55: SSH2 1: received packet type 2100:33:55: SSH2 1: SSH2_MSG_NEWKEYS received
00:33:56: SSH2 1: ssh_receive: 48 bytes received
00:33:56: SSH2 1: input: packet len 32
00:33:56: SSH2 1: partial packet 16, need 16, maclen 16
00:33:56: SSH2 1: MAC #3 ok
00:33:56: SSH2 1: input: padlen 10
00:33:56: SSH2 1: received packet type 5
00:33:56: SSH2 1: send: len 32 (includes padlen 10)
00:33:56: SSH2 1: done calc MAC out #3
00:33:56: SSH2 1: ssh_receive: 64 bytes received
00:33:56: SSH2 1: input: packet len 48
00:33:56: SSH2 1: partial packet 16, need 32, maclen 16
00:33:56: SSH2 1: MAC #4 ok
00:33:56: SSH2 1: input: padlen 9
00:33:56: SSH2 1: received packet type 50
00:33:56: SSH2 1: send: len 32 (includes padlen 13)
00:33:56: SSH2 1: done calc MAC out #4
00:34:04: SSH2 1: ssh_receive: 160 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #5 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 50
00:34:04: SSH2 1: send: len 16 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #5
00:34:04: SSH2 1: authentication successful for lab
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #6 ok
00:34:04: SSH2 1: input: padlen 6
00:34:04: SSH2 1: received packet type 2
00:34:04: SSH2 1: ssh_receive: 64 bytes received
00:34:04: SSH2 1: input: packet len 48
00:34:04: SSH2 1: partial packet 16, need 32, maclen 16
00:34:04: SSH2 1: MAC #7 ok
```

```
00:34:04: SSH2 1: input: padlen 19
00:34:04: SSH2 1: received packet type 90
00:34:04: SSH2 1: channel open request
00:34:04: SSH2 1: send: len 32 (includes padlen 10)
00:34:04: SSH2 1: done calc MAC out #6
00:34:04: SSH2 1: ssh_receive: 192 bytes received
00:34:04: SSH2 1: input: packet len 64
00:34:04: SSH2 1: partial packet 16, need 48, maclen 16
00:34:04: SSH2 1: MAC #8 ok
00:34:04: SSH2 1: input: padlen 13
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: pty-req request
00:34:04: SSH2 1: setting TTY - requested: height 24, width 80; set: height 24,
width 80
00:34:04: SSH2 1: input: packet len 96
00:34:04: SSH2 1: partial packet 16, need 80, maclen 16
00:34:04: SSH2 1: MAC #9 ok
00:34:04: SSH2 1: input: padlen 11
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: x11-req request
00:34:04: SSH2 1: ssh_receive: 48 bytes received
00:34:04: SSH2 1: input: packet len 32
00:34:04: SSH2 1: partial packet 16, need 16, maclen 16
00:34:04: SSH2 1: MAC #10 ok
00:34:04: SSH2 1: input: padlen 12
00:34:04: SSH2 1: received packet type 98
00:34:04: SSH2 1: shell request
00:34:04: SSH2 1: shell message received
00:34:04: SSH2 1: starting shell for vty
00:34:04: SSH2 1: send: len 48 (includes padlen 18)
00:34:04: SSH2 1: done calc MAC out #7
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #11 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #8
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #12 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #9
00:34:07: SSH2 1: ssh_receive: 48 bytes received
00:34:07: SSH2 1: input: packet len 32
00:34:07: SSH2 1: partial packet 16, need 16, maclen 16
00:34:07: SSH2 1: MAC #13 ok
00:34:07: SSH2 1: input: padlen 17
00:34:07: SSH2 1: received packet type 94
00:34:07: SSH2 1: send: len 32 (includes padlen 17)
00:34:07: SSH2 1: done calc MAC out #10
00:34:08: SSH2 1: ssh_receive: 48 bytes received
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #14 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 17)
00:34:08: SSH2 1: done calc MAC out #11
00:34:08: SSH2 1: ssh_receive: 48 bytes received
```

```
00:34:08: SSH2 1: input: packet len 32
00:34:08: SSH2 1: partial packet 16, need 16, maclen 16
00:34:08: SSH2 1: MAC #15 ok
00:34:08: SSH2 1: input: padlen 17
00:34:08: SSH2 1: received packet type 94
00:34:08: SSH2 1: send: len 32 (includes padlen 16)
00:34:08: SSH2 1: done calc MAC out #12
00:34:08: SSH2 1: send: len 48 (includes padlen 18)
00:34:08: SSH2 1: done calc MAC out #13
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #14
00:34:08: SSH2 1: send: len 16 (includes padlen 6)
00:34:08: SSH2 1: done calc MAC out #15
00:34:08: SSH1: Session terminated normally
```

Configuration Examples for Secure Shell Version 2 Support

Example: Configuring Secure Shell Version 2

```
Device# configure terminal
Device(config)# ip ssh version 2
```

Example: Configuring Secure Shell Versions 1 and 2

```
Device# configure terminal
Device(config)# no ip ssh version
```

Example: Starting an Encrypted Session with a Remote Device

```
Device# ssh -v 2 -c aes256-cbc -m hmac-sha1-160 -l shaship 10.76.82.24
```

Example: Configuring Server-Side SCP

The following example shows how to configure the server-side functionality for SCP. This example also configures AAA authentication and authorization on the device. This example uses a locally defined username and password.

```
Device# configure terminal
Device(config)# aaa new-model
Device(config)# aaa authentication login default local

Device(config)# username samplename privilege 15 password password1
Device(config)# ip ssh time-out 120
Device(config)# ip ssh authentication-retries 3
Device(config)# ip scp server enable
```

Example: Setting an SNMP Trap

The following example shows that an SNMP trap is set. The trap notification is generated automatically when the SSH session terminates. In the example, a.b.c.d is the IP address of the SSH client. For an example of SNMP trap debug output, see the "Example: SNMP Debugging, on page 28" section.

```
snmp-server
snmp-server host a.b.c.d public tty
```

Examples: SSH Keyboard Interactive Authentication

Example: Enabling Client-Side Debugs

The following example shows that the client-side debugs are turned on, and the maximum number of prompts is six (three for the SSH keyboard interactive authentication method and three for the password authentication method).

```
Password:
Password:
Password:
Password:
Password:
Password: cisco123
Last login: Tue Dec 6 13:15:21 2005 from 10.76.248.213
user1@courier:~> exit
logout
[Connection to 10.76.248.200 closed by foreign host]
Device1# debug ip ssh client

SSH Client debugging is on

Device1# ssh -l lab 10.1.1.3

Password:
*Nov 17 12:50:53.199: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: sent protocol version id SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.199: SSH CLIENT0: protocol version exchange successful
*Nov 17 12:50:53.203: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
*Nov 17 12:50:53.335: SSH CLIENT0: key exchange successful and encryption on
*Nov 17 12:50:53.335: SSH2 CLIENT 0: using method keyboard-interactive
Password:
Password:
Password:
*Nov 17 12:51:01.887: SSH2 CLIENT 0: using method password authentication
Password:
Password: lab
Device2>

*Nov 17 12:51:11.407: SSH2 CLIENT 0: SSH2_MSG_USERAUTH_SUCCESS message received
*Nov 17 12:51:11.407: SSH CLIENT0: user authenticated
*Nov 17 12:51:11.407: SSH2 CLIENT 0: pty-req request sent
*Nov 17 12:51:11.411: SSH2 CLIENT 0: shell request sent
*Nov 17 12:51:11.411: SSH CLIENT0: session open
```

Example: Enabling ChPass with a Blank Password Change

In the following example, the ChPass feature is enabled, and a blank password change is accomplished using the SSH Keyboard Interactive Authentication method. A TACACS+ access control server (ACS) is used as the back-end AAA server.

```
Device1# ssh -l cisco 10.1.1.3

Password:
Old Password: cisco
New Password: cisco123
Re-enter New password: cisco123

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]
```

Example: Enabling ChPass and Changing the Password on First Login

In the following example, the ChPass feature is enabled and TACACS+ ACS is used as the back-end server. The password is changed on the first login using the SSH keyboard interactive authentication method.

```
Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password:cisco1
Your password has expired.
Enter a new one now.
New Password: cisco
Re-enter New password: cisco12
The New and Re-entered passwords have to be the same.
Try again.
New Password: cisco
Re-enter New password: cisco

Device2>
```

Example: Enabling ChPass and Expiring the Password After Three Logins

In the following example, the ChPass feature is enabled and TACACS+ ACS is used as the back-end AAA server. The password expires after three logins using the SSH keyboard interactive authentication method.

```
Device# ssh -l cisco. 10.1.1.3

Password: cisco

Device2> exit
```

```
[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password: cisco

Device2> exit

Device1# ssh -l cisco 10.1.1.3

Password: cisco

Device2> exit

[Connection to 10.1.1.3 closed by foreign host]

Device1# ssh -l cisco 10.1.1.3

Password: cisco
Your password has expired.
Enter a new one now.
New Password: cisco123
Re-enter New password: cisco123

Device2>
```

Example: SNMP Debugging

The following is sample output from the **debug snmp packet** command. The output provides SNMP trap information for an SSH session.

```
Device1# debug snmp packet

SNMP packet debugging is on
Device1# ssh -l lab 10.0.0.2
Password:

Device2# exit

[Connection to 10.0.0.2 closed by foreign host]
Device1#
*Jul 18 10:18:42.619: SNMP: Queuing packet to 10.0.0.2
*Jul 18 10:18:42.619: SNMP: V1 Trap, ent cisco, addr 10.0.0.1, gentrap 6, spectrap 1
local.9.3.1.1.2.1 = 6
tcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 4
ltcpConnEntry.5.10.0.0.1.22.10.0.0.2.55246 = 1015
ltcpConnEntry.1.10.0.0.1.22.10.0.0.2.55246 = 1056
ltcpConnEntry.2.10.0.0.1.22.10.0.0.2.55246 = 1392
local.9.2.1.18.2 = lab
*Jul 18 10:18:42.879: SNMP: Packet sent via UDP to 10.0.0.2

Device1#
```

Examples: SSH Debugging Enhancements

The following is sample output from the **debug ip ssh detail** command. The output provides debugging information about the SSH protocol and channel requests.

```
Device# debug ip ssh detail
```

```
00:04:22: SSH0: starting SSH control process
00:04:22: SSH0: sent protocol version id SSH-1.99-Cisco-1.25
00:04:22: SSH0: protocol version id is - SSH-1.99-Cisco-1.25
00:04:22: SSH2 0: SSH2_MSG_KEXINIT sent
00:04:22: SSH2 0: SSH2_MSG_KEXINIT received
00:04:22: SSH2:kex: client->server enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2:kex: server->client enc:aes128-cbc mac:hmac-sha1
00:04:22: SSH2 0: expecting SSH2_MSG_KEXDH_INIT
00:04:22: SSH2 0: SSH2_MSG_KEXDH_INIT received
00:04:22: SSH2: kex_derive_keys complete
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS sent
00:04:22: SSH2 0: waiting for SSH2_MSG_NEWKEYS
00:04:22: SSH2 0: SSH2_MSG_NEWKEYS received
00:04:24: SSH2 0: authentication successful for lab
00:04:24: SSH2 0: channel open request
00:04:24: SSH2 0: pty-req request
00:04:24: SSH2 0: setting TTY - requested: height 24, width 80; set: height 24, width 80
00:04:24: SSH2 0: shell request
00:04:24: SSH2 0: shell message received
00:04:24: SSH2 0: starting shell for vty
00:04:38: SSH0: Session terminated normally
```

The following is sample output from the **debug ip ssh packet** command. The output provides debugging information about the SSH packet.

```
Device# debug ip ssh packet

00:05:43: SSH2 0: send:packet of  length 280 (length also includes padlen of 4)
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 280 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 24 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 272 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 4 bytes
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: input: total packet length of 144 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 64 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 136 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 6 bytes
00:05:43: SSH2 0: signature length 143
00:05:43: SSH2 0: send:packet of  length 448 (length also includes padlen of 7)
00:05:43: SSH2 0: send:packet of  length 16 (length also includes padlen of 10)
00:05:43: SSH2 0: newkeys: mode 1
00:05:43: SSH2 0: ssh_receive: 16 bytes received
00:05:43: SSH2 0: input: total packet length of 16 bytes
00:05:43: SSH2 0: partial packet length(block size)8 bytes,needed 8 bytes, maclen 0
00:05:43: SSH2 0: input: padlength 10 bytes
00:05:43: SSH2 0: newkeys: mode 0
00:05:43: SSH2 0: ssh_receive: 52 bytes received
00:05:43: SSH2 0: input: total packet length of 32 bytes
00:05:43: SSH2 0: partial packet length(block size)16 bytes,needed 16 bytes, maclen 20
00:05:43: SSH2 0: MAC compared for #3 :ok
```

Where to Go Next

You have to use a SSH remote device that supports SSH Version 2, and you have to connect to a Cisco device.

Additional References for Secure Shell Version 2 Support

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Command List, All Releases
AAA Hostname and host domain configuration tasks Secure shell configuration tasks	*Security Configuration Guide: Securing User Services*
Downloading a software image Configuration fundamentals	*Configuration Fundamentals Configuration Guide*
IPsec configuration tasks	*Security Configuration Guide: Secure Connectivity*
SNMP traps configuration tasks	*SNMP Configuration Guide*

Standards

Standards	Title
IETF Secure Shell Version 2 Draft Standards	Internet Engineering Task Force website

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html