



# Implementing Secure Shell

Secure Shell (SSH) is an application and a protocol that provides a secure replacement to the Berkeley r-tools. The protocol secures sessions using standard cryptographic mechanisms, and the application can be used similarly to the Berkeley **rexec** and **rsh** tools.

Two versions of the SSH server are available: SSH Version 1 (SSHv1) and SSH Version 2 (SSHv2). SSHv1 uses Rivest, Shamir, and Adelman (RSA) keys and SSHv2 uses either Digital Signature Algorithm (DSA) keys or Rivest, Shamir, and Adelman (RSA) keys, or Elliptic Curve Digital Signature Algorithm (ECDSA) keys. Cisco IOS XR software supports both SSHv1 and SSHv2.



---

**Note** Cisco IOS XR does not support X11 forwarding through an SSH connection.

---

This module describes how to implement Secure Shell on the the Cisco ASR 9000 Series Router.



---

**Note** For a complete description of the Secure Shell commands used in this module , see the *Secure Shell Commands* module in *System Security Command Reference for Cisco ASR 9000 Series Routers*.

---

## Feature History for Implementing Secure Shell

Release	Modification
Release 3.7.2	This feature was introduced.
Release 3.9.0	Support was added for the following enhancements: <ul style="list-style-type: none"><li>• RSA based authentication on the SSH server</li><li>• SFTP client in interactive mode</li><li>• SFTP server implementation</li></ul>
Release 5.3.0	Support was added for Netconf Subsystem support on ssh server using a dedicated port. For more details see chapter <i>Implementing Network Configuration Protocol</i> in the <i>System Management Configuration Guide</i> .
Release 6.4.1	Support was added for ECDSA algorithm on IOS-XR SSHv2.

Release	Modification
Release 7.0.1	Support was added for SSH configuration option to restrict CIPHER public key and HMAC.
Release 7.0.1	Support was added for automatic host key generation for SSH algorithms.
Release 7.0.1	SSH and SFTP in baseline Cisco IOS XR Software image.
Release 7.0.1	Support was added for enabling CBC mode ciphers 3DES-CBC and AES-CBC for SSHv2 server and client connections.

- [Prerequisites for Implementing Secure Shell, on page 2](#)
- [SSH and SFTP in Baseline Cisco IOS XR Software Image, on page 3](#)
- [Restrictions for Implementing Secure Shell, on page 3](#)
- [Information About Implementing Secure Shell, on page 4](#)
- [How to Implement Secure Shell, on page 8](#)
- [Configuration Examples for Implementing Secure Shell, on page 18](#)
- [Multi-channeling in SSH, on page 18](#)
- [SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 20](#)
- [Additional References, on page 23](#)

## Prerequisites for Implementing Secure Shell

The following prerequisites are required to implement Secure Shell:

- You must be in a user group associated with a task group that includes the proper task IDs. The command reference guides include the task IDs required for each command. If you suspect user group assignment is preventing you from using a command, contact your AAA administrator for assistance.
- Download the required image on your router. The SSH server and SSH client require you to have a crypto package (data encryption standard [DES], 3DES and AES) from Cisco downloaded on your router.




---

**Note** From Cisco IOS XR Software Release 7.0.1 and later, the SSH and SFTP components are available in the baseline Cisco IOS XR software image itself. For details, see, [SSH and SFTP in Baseline Cisco IOS XR Software Image, on page 3](#).

---

- To run an SSHv2 server, you must have a VRF. This may be the default VRF or a specific VRF. VRF changes are applicable only to the SSH v2 server.
- Configure user authentication for local or remote access. You can configure authentication with or without authentication, authorization, and accounting (AAA). For more information, see the *Authentication, Authorization, and Accounting Commands on Cisco IOS XR Software* module in the *System Security Command Reference for Cisco ASR 9000 Series Routers* publication and *Configuring AAA Services on Cisco IOS XR Software* module in the *System Security Configuration Guide for Cisco ASR 9000 Series Routers* publication.
- AAA authentication and authorization must be configured correctly for Secure Shell File Transfer Protocol (SFTP) to work.

# SSH and SFTP in Baseline Cisco IOS XR Software Image

From Cisco IOS XR Software Release 7.0.1 and later, the management plane and control plane components that were part of the Cisco IOS XR security package (k9sec package) are moved to the base Cisco IOS XR software image. These include SSH, SCP, SFTP and IPsec control plane. However, *802.IX protocol (Port-Based Network Access Control)* and data plane components like MACsec and IPsec remain as a part of the security package as per the export compliance regulations. This segregation of package components makes the software more modular. It also gives you the flexibility of including or excluding the security package as per your requirements. The new segregation of package components is applicable for both 32 bit and 64 bit IOS XR images.

The base package and the security package allow FIPS, so that the control plane can negotiate FIPS-approved algorithms.

See [SSH and SFTP in Baseline Cisco IOS XR Software Image](#).

## Restrictions for Implementing Secure Shell

The following are some basic SSH restrictions and limitations of the SFTP feature:

- A VRF is not accepted as inband if that VRF is already set as an out-of-band VRF. SSH v1 continues to bind only to the default VRF.
- In order for an outside client to connect to the router, the router needs to have an RSA (for SSHv1 or SSHv2) or DSA (for SSHv2) or ECDSA (for SSHv2) key pair configured. ECDSA, DSA and RSA keys are not required if you are initiating an SSH client connection from the router to an outside routing device. The same is true for SFTP: ECDSA, DSA and RSA keys are not required because SFTP operates only in client mode.
- In order for SFTP to work properly, the remote SSH server must enable the SFTP server functionality. For example, the SSHv2 server is configured to handle the SFTP subsystem with a line such as **/etc/ssh2/sshd2\_config**:
- **subsystem-sftp /usr/local/sbin/sftp-server**
- The SFTP server is usually included as part of SSH packages from public domain and is turned on by default configuration.
- SFTP is compatible with sftp server version OpenSSH\_2.9.9p2 or higher.
- RSA-based user authentication is supported in the SSH and SFTP servers. The support however, is not extended to the SSH client.
- Execution shell and SFTP are the only applications supported.
- The AES encryption algorithm is supported on the SSHv2 server and client, but not on the SSHv1 server and client. Any requests for an AES cipher sent by an SSHv2 client to an SSHv1 server are ignored, with the server using 3DES instead.
- The SFTP client does not support remote filenames containing wildcards (\*, ?, []). The user must issue the **sftp** command multiple times or list all of the source files from the remote host to download them on to the router. For uploading, the router SFTP client can support multiple files specified using a wildcard provided that the issues mentioned in the first through third bullets in this section are resolved.

- The cipher preference for the SSH server follows the order AES128, AES192, AES256, and, finally, 3DES. The server rejects any requests by the client for an unsupported cipher, and the SSH session does not proceed.
- Use of a terminal type other than vt100 is unsupported, and the software generates a warning message in this case.
- Password messages of “none” are unsupported on the SSH client.
- Because the router infrastructure does not provide support for UNIX-like file permissions, files created on the local device lose the original permission information. For files created on the remote file system, the file permission adheres to the umask on the destination host and the modification and last access times are the time of the copy.

## Information About Implementing Secure Shell

To implement SSH, you should understand the following concepts:

### SSH Server

The SSH server feature enables an SSH client to make a secure, encrypted connection to a Cisco router. This connection provides functionality that is similar to that of an inbound Telnet connection. Before SSH, security was limited to Telnet security. SSH allows a strong encryption to be used with the Cisco IOS XR software authentication. The SSH server in Cisco IOS XR software works with publicly and commercially available SSH clients.

### SSH Client

The SSH client feature is an application running over the SSH protocol to provide device authentication and encryption. The SSH client enables a Cisco router to make a secure, encrypted connection to another Cisco router or to any other device running the SSH server. This connection provides functionality that is similar to that of an outbound Telnet connection except that the connection is encrypted. With authentication and encryption, the SSH client allows for a secure communication over an insecure network.

The SSH client in the Cisco IOS XR software worked with publicly and commercially available SSH servers. The SSH client supported the ciphers of AES, 3DES, message digest algorithm 5 (MD5), SHA1, and password authentication. User authentication was performed in the Telnet session to the router. The user authentication mechanisms supported for SSH were RADIUS, TACACS+, and the use of locally stored usernames and passwords.

The SSH client supports setting DSCP value in the outgoing packets.

```
ssh client dscp <value from 0 - 63>
```

If not configured, the default DSCP value set in packets is 16 (for both client and server).

The SSH client supports the following options:

- DSCP—DSCP value for SSH client sessions.

```
RP/0/5/CPU0:router#configure
RP/0/5/CPU0:router(config)#ssh ?
  client  Provide SSH client service
  server  Provide SSH server service
```

```

timeout Set timeout value for SSH
RP/0/5/CPU0:router(config)#ssh client ?

```

- **Knownhost**—Enable the host pubkey check by local database.
- **Source-interface**—Source interface for SSH client sessions.

```

RP/0/5/CPU0:router(config)#ssh client source-interface ?
ATM ATM Network Interface(s)
BVI Bridge-Group Virtual Interface
Bundle-Ether Aggregated Ethernet interface(s)
Bundle-POS Aggregated POS interface(s)
CEM Circuit Emulation interface(s)
GigabitEthernet GigabitEthernet/IEEE 802.3 interface(s)
IMA ATM Network Interface(s)
IMtestmain IM Test Interface
Loopback Loopback interface(s)
MgmtEth Ethernet/IEEE 802.3 interface(s)
Multilink Multilink network interface(s)
Null Null interface
PFItestmain PFI Test Interface
PFItestnothw PFI Test Not-HW Interface
POS Packet over SONET/SDH network interface(s)
PW-Ether PWHE Ethernet Interface
PW-IW PWHE VC11 IP Interworking Interface
Serial Serial network interface(s)
VASIleft VASI Left interface(s)
VASIRight VASI Right interface(s)
test-bundle-channel Aggregated Test Bundle interface(s)
tunnel-ipsec IPsec Tunnel interface(s)
tunnel-mte MPLS Traffic Engineering P2MP Tunnel interface(s)
tunnel-te MPLS Traffic Engineering Tunnel interface(s)
tunnel-tp MPLS Transport Protocol Tunnel interface
RP/0/5/CPU0:router(config)#ssh client source-interface
RP/0/5/CPU0:router(config)#

```

- **VRF**—Source interface VRF for SSH client sessions:

```

RP/0/5/CPU0:router(config)#ssh client vrf ?
WORD VRF name (max:32 chars)
RP/0/5/CPU0:router(config)#ssh client vrf shan ?
<cr>
RP/0/5/CPU0:router(config)#ssh client vrf shan

```

SSH also supports remote command execution as follows:

```

RP/0/5/CPU0:router#ssh ?
A.B.C.D IPv4 (A.B.C.D) address
WORD Hostname of the remote node
X:X::X IPv6 (A:B:C:D...:D) address
vrf vrf table for the route lookup
RP/0/5/CPU0:router#ssh 10.1.1.1 ?
cipher Accept cipher type
command Specify remote command (non-interactive)
source-interface Specify source interface
username Accept userid for authentication
<cr>
RP/0/5/CPU0:router#ssh 192.68.46.6 username admin command "show redundancy sum"
Password:

```

```

Wed Jan 9 07:05:27.997 PST
Active Node Standby Node
-----
0/4/CPU0 0/5/CPU0 (Node Ready, NSR: Not Configured)

```

```
RP/0/5/CPU0:router#
```

## SFTP Feature Overview

SSH includes support for standard file transfer protocol (SFTP), a new standard file transfer protocol introduced in SSHv2. This feature provides a secure and authenticated method for copying router configuration or router image files.

The SFTP client functionality is provided as part of the SSH component and is always enabled on the router. Therefore, a user with the appropriate level can copy files to and from the router. Like the **copy** command, the **sftp** command can be used only in EXEC mode.

The SFTP client is VRF-aware, and you may configure the secure FTP client to use the VRF associated with a particular source interface during connections attempts. The SFTP client also supports interactive mode, where the user can log on to the server to perform specific tasks via the Unix server.

The SFTP Server is a sub-system of the SSH server. In other words, when an SSH server receives an SFTP server request, the SFTP API creates the SFTP server as a child process to the SSH server. A new SFTP server instance is created with each new request.

The SFTP requests for a new SFTP server in the following steps:

- The user runs the **sftp** command with the required arguments
- The SFTP API internally creates a child session that interacts with the SSH server
- The SSH server creates the SFTP server child process
- The SFTP server and client interact with each other in an encrypted format
- The SFTP transfer is subject to LPTS policer "SSH-Known". Low policer values will affect SFTP transfer speeds



---

**Note** In IOS-XR SW release 4.3.1 onwards the default policer value for SSH-Known has been reset from 2500pps to 300pps. Slower transfers are expected due to this change. You can adjust the lpts policer value for this punt cause to higher values that will allow faster transfers

---

When the SSH server establishes a new connection with the SSH client, the server daemon creates a new SSH server child process. The child server process builds a secure communications channel between the SSH client and server via key exchange and user authentication processes. If the SSH server receives a request for the sub-system to be an SFTP server, the SSH server daemon creates the SFTP server child process. For each incoming SFTP server subsystem request, a new SSH server child and a SFTP server instance is created. The SFTP server authenticates the user session and initiates a connection. It sets the environment for the client and the default directory for the user.

Once the initialization occurs, the SFTP server waits for the SSH\_FXP\_INIT message from the client, which is essential to start the file communication session. This message may then be followed by any message based on the client request. Here, the protocol adopts a 'request-response' model, where the client sends a request to the server; the server processes this request and sends a response.

The SFTP server displays the following responses:

- Status Response

- Handle Response
- Data Response
- Name Response



---

**Note** The server must be running in order to accept incoming SFTP connections.

---

## RSA Based Host Authentication

Verifying the authenticity of a server is the first step to a secure SSH connection. This process is called the host authentication, and is conducted to ensure that a client connects to a valid server.

The host authentication is performed using the public key of a server. The server, during the key-exchange phase, provides its public key to the client. The client checks its database for known hosts of this server and the corresponding public-key. If the client fails to find the server's IP address, it displays a warning message to the user, offering an option to either save the public key or discard it. If the server's IP address is found, but the public-key does not match, the client closes the connection. If the public key is valid, the server is verified and a secure SSH connection is established.

The IOS XR SSH server and client had support for DSA based host authentication. But for compatibility with other products, like IOS, RSA based host authentication support is also added.

## RSA Based User Authentication

One of the method for authenticating the user in SSH protocol is RSA public-key based user authentication. The possession of a private key serves as the authentication of the user. This method works by sending a signature created with a private key of the user. Each user has a RSA keypair on the client machine. The private key of the RSA keypair remains on the client machine.

The user generates an RSA public-private key pair on a unix client using a standard key generation mechanism such as ssh-keygen. The max length of the keys supported is 4096 bits, and the minimum length is 512 bits. The following example displays a typical key generation activity:

```
bash-2.05b$ ssh-keygen -b 1024 -t rsa
Generating RSA private key, 1024 bit long modulus
```

The public key must be in base64 encoded (binary) format for it to be imported correctly into the box. You can use third party tools available on the Internet to convert the key to the binary format.

Once the public key is imported to the router, the SSH client can choose to use the public key authentication method by specifying the request using the “-o” option in the SSH client. For example:

```
client$ ssh -o PreferredAuthentications=publickey 1.2.3.4
```

If a public key is not imported to a router using the RSA method, the SSH server initiates the password based authentication. If a public key is imported, the server proposes the use of both the methods. The SSH client then chooses to use either method to establish the connection. The system allows only 10 outgoing SSH client connections.

Currently, only SSH version 2 and SFTP server support the RSA based authentication. For more information on how to import the public key to the router, see the *Implementing Certification Authority Interoperability on the Cisco ASR 9000 Series Router* chapter in this guide.




---

**Note** The preferred method of authentication would be as stated in the SSH RFC. The RSA based authentication support is only for local authentication, and not for TACACS/RADIUS servers.

---

Authentication, Authorization, and Accounting (AAA) is a suite of network security services that provide the primary framework through which access control can be set up on your Cisco router or access server. For more information on AAA, see the *Authentication, Authorization, and Accounting Commands on the Cisco ASR 9000 Series Router* Software module in the *System Security Command Reference for Cisco ASR 9000 Series Routers* publication and the *Configuring AAA Services on the Cisco ASR 9000 Series Router* module in the *System Security Configuration Guide for Cisco ASR 9000 Series Routers* publication.

## SSHv2 Client Keyboard-Interactive Authentication

An authentication method in which the authentication information is entered using a keyboard is known as keyboard-interactive authentication. This method is an interactive authentication method in the SSH protocol. This type of authentication allows the SSH client to support different methods of authentication without having to be aware of their underlying mechanisms.

Currently, the SSHv2 client supports the keyboard-interactive authentication. This type of authentication works only for interactive applications.




---

**Note** The password authentication is the default authentication method. The keyboard-interactive authentication method is selected if the server is configured to support only the keyboard-interactive authentication.

---

## How to Implement Secure Shell

To configure SSH, perform the tasks described in the following sections:

### Configuring SSH




---

**Note** For SSHv1 configuration, Step 1 to Step 4 are required. For SSHv2 configuration, Step 1 to Step 4 are optional.

---




---

**Note** From Cisco IOS XR Software Release 7.0.1 and later, the SSH host-key pairs are auto-generated at the time of router boot up. Hence you need not perform steps 5 to 7 to generate the host keys explicitly. See, [Automatic Generation of SSH Host-Key Pairs, on page 12](#) for details.

---

SSH server supports setting DSCP value in the outgoing packets.



```
ssh server dscp <value from 0 - 63>
```

If not configured, the default DSCP value set in packets is 16 (for both client and server).

This is the syntax for setting DSCP value:

```
RP/0/5/CPU0:router(config)#ssh server dscp ?
    <0-63>  DSCP value range

RP/0/5/CPU0:router(config)#ssh server dscp 63 ?
    <cr>
RP/0/5/CPU0:router(config)#ssh server dscp 63
RP/0/5/CPU0:router(config)#

RP/0/5/CPU0:router(config)#ssh client dscp ?
    <0-63>  DSCP value range

RP/0/5/CPU0:router(config)#ssh client dscp 0 ?
    <cr>
RP/0/5/CPU0:router(config)#ssh client dscp 0
RP/0/5/CPU0:router(config)#
```

Perform this task to configure SSH.

## SUMMARY STEPS

1. **configure**
2. **hostname** *hostname*
3. **domain name** *domain-name*
4. Use the **commit** or **end** command.
5. **crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*]
6. **crypto key generate dsa**
7. **crypto key generate ecdsa** [**nistp256** | **nistp384** | **nistp521**]
8. **configure**
9. **ssh timeout** *seconds*
10. Do one of the following:
  - **ssh server** [**vrf** *vrf-name* [**ipv4 access-list** IPv4 access-list name] [**ipv6 access-list** IPv6 access-list name]]
  - **ssh server v2**
11. Use the **commit** or **end** command.
12. **show ssh**
13. **show ssh session details**
14. **show ssh history**
15. **show ssh history details**
16. **show tech-support ssh**

## DETAILED STEPS

	Command or Action	Purpose
Step 1	<b>configure</b>  <b>Example:</b>	Enters global configuration mode.

	Command or Action	Purpose
	RP/0/RSP0/CPU0:router# configure	
<b>Step 2</b>	<b>hostname</b> <i>hostname</i> <b>Example:</b> RP/0/RSP0/CPU0:router(config)# hostname router1	Configures a hostname for your router.
<b>Step 3</b>	<b>domain name</b> <i>domain-name</i> <b>Example:</b> RP/0/RSP0/CPU0:router(config)# domain name cisco.com	Defines a default domain name that the software uses to complete unqualified host names.
<b>Step 4</b>	Use the <b>commit</b> or <b>end</b> command.	<b>commit</b> —Saves the configuration changes and remains within the configuration session. <b>end</b> —Prompts user to take one of these actions: <ul style="list-style-type: none"> <li>• <b>Yes</b> — Saves configuration changes and exits the configuration session.</li> <li>• <b>No</b> —Exits the configuration session without committing the configuration changes.</li> <li>• <b>Cancel</b> —Remains in the configuration session, without committing the configuration changes.</li> </ul>
<b>Step 5</b>	<b>crypto key generate rsa</b> [ <i>usage keys</i>   <i>general-keys</i> ] [ <i>keypair-label</i> ] <b>Example:</b> RP/0/RSP0/CPU0:router# crypto key generate rsa general-keys	Generates an RSA key pair. The RSA key modulus can be in the range of 512 to 4096 bits. <ul style="list-style-type: none"> <li>• To delete the RSA key pair, use the <b>crypto key zeroize rsa</b> command.</li> <li>• This command is used for SSHv1 only.</li> </ul>
<b>Step 6</b>	<b>crypto key generate dsa</b> <b>Example:</b> RP/0/RSP0/CPU0:router# crypto key generate dsa	Enables the SSH server for local and remote authentication on the router. The supported key sizes are: 512, 768 and 1024 bits. <ul style="list-style-type: none"> <li>• The recommended minimum modulus size is 1024 bits.</li> <li>• Generates a DSA key pair. To delete the DSA key pair, use the <b>crypto key zeroize dsa</b> command.</li> <li>• This command is used only for SSHv2.</li> </ul>
<b>Step 7</b>	<b>crypto key generate ecdsa</b> [ <i>nistp256</i>   <i>nistp384</i>   <i>nistp521</i> ] <b>Example:</b>	Generates an ECDSA key pair. The supported ECDSA curve types are: Nistp256, Nistp384 and Nistp521.

	Command or Action	Purpose
	<pre>RP/0/RSP0/CPU0:router# crypto key generate ecdsa nistp256</pre>	<ul style="list-style-type: none"> <li>To delete the ECDSA key pair, use the <b>crypto key zeroize ecdsa [ nistp256   nistp384   nistp521]</b> command.</li> <li>This command is used for SSHv2 only.</li> </ul>
<b>Step 8</b>	<p><b>configure</b></p> <p><b>Example:</b></p> <pre>RP/0/RSP0/CPU0:router# configure</pre>	Enters global configuration mode.
<b>Step 9</b>	<p><b>ssh timeout <i>seconds</i></b></p> <p><b>Example:</b></p> <pre>RP/0/RSP0/CPU0:router(config)# ssh timeout 60</pre>	<p>(Optional) Configures the timeout value for user authentication to AAA.</p> <ul style="list-style-type: none"> <li>If the user fails to authenticate itself to AAA within the configured time, the connection is terminated.</li> <li>If no value is configured, the default value of 30 seconds is used. The range is from 5 to 120.</li> </ul>
<b>Step 10</b>	<p>Do one of the following:</p> <ul style="list-style-type: none"> <li><b>ssh server [vrf <i>vrf-name</i> [ipv4 access-list IPv4 access-list name] [ipv6 access-list IPv6 access-list name]]</b></li> <li><b>ssh server v2</b></li> </ul> <p><b>Example:</b></p> <pre>RP/0/RSP0/CPU0:router(config)# ssh</pre> <p>or</p> <pre>RP/0/RSP0/CPU0:router(config)# ssh server v2</pre>	<ul style="list-style-type: none"> <li>(Optional) Brings up an SSH server using a specified VRF of up to 32 characters. If no VRF is specified, the default VRF is used. To stop the SSH server from receiving any further connections for the specified VRF, use the no form of this command. If no VRF is specified, the default is assumed. Optionally ACLs for IPv4 and IPv6 can be used to restrict access to the server before the port is opened. To stop the SSH server from receiving any further connections for the specified VRF, use the <b>no</b> form of this command. If no VRF is specified, the default is assumed.</li> </ul> <p><b>Note</b> The SSH server can be configured for multiple VRF usage.</p> <ul style="list-style-type: none"> <li>(Optional) Forces the SSH server to accept only SSHv2 clients if you configure the SSHv2 option by using the <b>ssh server v2</b> command. If you choose the <b>ssh server v2</b> command, only the SSH v2 client connections are accepted.</li> </ul>
<b>Step 11</b>	Use the <b>commit</b> or <b>end</b> command.	<p><b>commit</b> —Saves the configuration changes and remains within the configuration session.</p> <p><b>end</b> —Prompts user to take one of these actions:</p> <ul style="list-style-type: none"> <li><b>Yes</b> — Saves configuration changes and exits the configuration session.</li> <li><b>No</b> —Exits the configuration session without committing the configuration changes.</li> </ul>

	Command or Action	Purpose
		<ul style="list-style-type: none"> <li>• <b>Cancel</b> —Remains in the configuration session, without committing the configuration changes.</li> </ul>
<b>Step 12</b>	<b>show ssh</b> <b>Example:</b> <pre>RP/0/RSP0/CPU0:router# show ssh</pre>	(Optional) Displays all of the incoming and outgoing SSHv1 and SSHv2 connections to the router.
<b>Step 13</b>	<b>show ssh session details</b> <b>Example:</b> <pre>RP/0/RSP0/CPU0:router# show ssh session details</pre>	(Optional) Displays a detailed report of the SSHv2 connections to and from the router.
<b>Step 14</b>	<b>show ssh history</b> <b>Example:</b> <pre>RP/0/RSP0/CPU0:router# show ssh history</pre>	(Optional) Displays the last hundred SSH connections that were terminated.
<b>Step 15</b>	<b>show ssh history details</b> <b>Example:</b> <pre>RP/0/RSP0/CPU0:router# show ssh history details</pre>	(Optional) Displays the last hundred SSH connections that were terminated with additional details. This command is similar to <b>show ssh session details</b> command but also mentions the start and end time of the session.
<b>Step 16</b>	<b>show tech-support ssh</b> <b>Example:</b> <pre>RP/0/RSP0/CPU0:router# show tech-support ssh</pre>	(Optional) Automatically runs the <code>show</code> commands that display system information.



**Note** The order of priority while doing negotiation for a SSH connection is as follows:

1. ecdsa-nistp-521
2. ecdsa-nistp-384
3. ecdsa-nistp-256
4. rsa
5. dsa

## Automatic Generation of SSH Host-Key Pairs

This feature brings in the functionality of automatically generating the SSH host-key pairs for the DSA, ECDSA (such as **ecdsa-nistp256**, **ecdsa-nistp384**, and **ecdsa-nistp521**) and RSA algorithms. This in turn eliminates the need for explicitly generating each SSH host-key pair after the router boots up. Because the keys are already present in the system, the SSH client can establish connection with the SSH server soon after

the router boots up with the basic SSH configuration. This is useful especially during zero touch provisioning (ZTP) and Golden ISO boot up scenarios.

Before the introduction of this feature, you had to execute the **crypto key generate** command in EXEC mode to generate the required SSH host-key pairs.

Although the host-key pairs are auto-generated with the introduction of this feature, you still have the flexibility to select only the required algorithms on the SSH server. You can use the **ssh server algorithms host-key** command in Global Configuration mode to achieve the same. Alternatively, you can also use the existing **crypto key zeroize** command in EXEC mode to remove the algorithms that are not required.



---

**Note** In a system upgrade scenario from version 1 to version 2, the system does not generate the SSH host-key pairs automatically if they were already generated in version 1. The host-key pairs are generated automatically only if they were not generated in version 1.

---

## Configure the Allowed SSH Host-Key Pair Algorithms

When the SSH client attempts a connection with the SSH server, it sends a list of SSH host-key pair algorithms (in the order of preference) internally in the connection request. The SSH server, in turn, picks the first matching algorithm from this request list. The server establishes a connection only if that host-key pair is already generated in the system, and if it is configured (using the **ssh server algorithms host-key** command) as the allowed algorithm.



---

**Note** If this configuration of allowed host-key pairs is not present in the SSH server, then you can consider that the SSH server allows all host-key pairs. In that case, the SSH client can connect with any one of the host-key pairs. Not having this configuration also ensures backward compatibility in system upgrade scenarios.

---

### Configuration Example

You may perform this (optional) task to specify the allowed SSH host-key pair algorithm (in this example, **ecdsa**) from the list of auto-generated host-key pairs on the SSH server:

```
/* Example to select the ecdsa algorithm */  
Router(config)#ssh server algorithms host-key ecdsa-nistp521
```

Similarly, you may configure other algorithms.

### Running Configuration

```
ssh server algorithms host-key ecdsa-nistp521  
!
```

## Verify the SSH Host-Key Pair Algorithms



**Note** With the introduction of the automatic generation of SSH host-key pairs, the output of the **show crypto key mypubkey** command displays key information of all the keys that are auto-generated. Before its introduction, the output of this show command displayed key information of only those keys that you explicitly generated using the **crypto key generate** command.

```
Router#show crypto key mypubkey ecdsa
Mon Nov 19 12:22:51.762 UTC
Key label: the_default
Type      : ECDSA General Curve Nistp256
Degree   : 256
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04AC7533 3ABE7874 43F024C1 9C24CC66 490E83BE 76CEF4E2 51BBEF11 170CDB26
14289D03 6625FC4F 3E7F8F45 0DA730C3 31E960FE CF511A05 2B0AA63E 9C022482
6E

Key label: the_default
Type      : ECDSA General Curve Nistp384
Degree   : 384
Created  : 10:59:08 UTC Mon Nov 19 2018
Data     :
04B70BAF C096E2CA D848EE72 6562F3CC 9F12FA40 BE09BFE6 AF0CA179 F29F6407
FEE24A43 84C5A5DE D7912208 CB67EE41 58CB9640 05E9421F 2DCDC41C EED31288
6CACC8DD 861DC887 98E535C4 893CB19F 5ED3F6BC 2C90C39B 10EAED57 87E96F78
B6

Key label: the_default
Type      : ECDSA General Curve Nistp521
Degree   : 521
Created  : 10:59:09 UTC Mon Nov 19 2018
Data     :
0400BA39 E3B35E13 810D8AE5 260B8047 84E8087B 5137319A C2865629 8455928F
D3D9CE39 00E097FF 6CA369C3 EE63BA57 A4C49C02 B408F682 C2153B7F AAE53EF8
A2926001 EF113896 5F1DA056 2D62F292 B860FDFB 0314CE72 F87AA2C9 D5DD29F4
DA85AE4D 1CA453AC 412E911A 419E9B43 0A13DAD3 7B7E88E4 7D96794B 369D6247
E3DA7B8A 5E
```

### Related Topics

[Automatic Generation of SSH Host-Key Pairs, on page 12](#)

### Associated Commands

- ssh server algorithms host-key
- show crypto key mypubkey

## Configuring the SSH Client

Perform this task to configure an SSH client.

**SUMMARY STEPS**

1. **configure**
2. **ssh client knownhost** *device* : /filename
3. Use the **commit** or **end** command.
4. **ssh** {*ipv4-address* | *hostname*} [ **username** *user-id* | **cipher** **des** | **source-interface** *type instance*]

**DETAILED STEPS**

	Command or Action	Purpose
<b>Step 1</b>	<p><b>configure</b></p> <p><b>Example:</b></p> <pre>RP/0/RSP0/CPU0:router# configure</pre>	Enters global configuration mode.
<b>Step 2</b>	<p><b>ssh client knownhost</b> <i>device</i> : /filename</p> <p><b>Example:</b></p> <pre>RP/0/RSP0/CPU0:router(config)# ssh client knownhost slot1:/server_pubkey</pre>	<p>(Optional) Enables the feature to authenticate and check the server public key (pubkey) at the client end.</p> <p><b>Note</b> The complete path of the filename is required. The colon (:) and slash mark (/) are also required.</p>
<b>Step 3</b>	Use the <b>commit</b> or <b>end</b> command.	<p><b>commit</b> —Saves the configuration changes and remains within the configuration session.</p> <p><b>end</b> —Prompts user to take one of these actions:</p> <ul style="list-style-type: none"> <li>• <b>Yes</b> — Saves configuration changes and exits the configuration session.</li> <li>• <b>No</b> —Exits the configuration session without committing the configuration changes.</li> <li>• <b>Cancel</b> —Remains in the configuration session, without committing the configuration changes.</li> </ul>
<b>Step 4</b>	<p><b>ssh</b> {<i>ipv4-address</i>   <i>hostname</i>} [ <b>username</b> <i>user-id</i>   <b>cipher</b> <b>des</b>   <b>source-interface</b> <i>type instance</i>]</p> <p><b>Example:</b></p> <pre>RP/0/RSP0/CPU0:router# ssh remotehost username user1234</pre>	<p>Enables an outbound SSH connection.</p> <ul style="list-style-type: none"> <li>• To run an SSHv2 server, you must have a VRF. This may be the default or a specific VRF. VRF changes are applicable only to the SSH v2 server.</li> <li>• The SSH client tries to make an SSHv2 connection to the remote peer. If the remote peer supports only the SSHv1 server, the peer internally spawns an SSHv1 connection to the remote server.</li> <li>• The <b>cipher des</b> option can be used only with an SSHv1 client.</li> <li>• The SSHv1 client supports only the 3DES encryption algorithm option, which is still available by default for those SSH clients only.</li> </ul>

	Command or Action	Purpose
		<ul style="list-style-type: none"> <li>If the <i>hostname</i> argument is used and the host has both IPv4 and IPv6 addresses, the IPv6 address is used.</li> </ul>

- If you are using SSHv1 and your SSH connection is being rejected, the reason could be that the RSA key pair might have been zeroed out. Another reason could be that the SSH server to which the user is connecting to using SSHv1 client does not accept SSHv1 connections. Make sure that you have specified a hostname and domain. Then use the **crypto key generate rsa** command to generate an RSA host-key pair, and then enable the SSH server.
- If you are using SSHv2 and your SSH connection is being rejected, the reason could be that the DSA, RSA or ECDSA host-key pair might have been zeroed out. Make sure you follow similar steps as mentioned above to generate the required host-key pairs, and then enable the SSH server.
- When configuring the ECDSA, RSA or DSA key pair, you might encounter the following error messages:
  - No hostname specified

You must configure a hostname for the router using the **hostname** command.

- No domain specified

You must configure a host domain for the router using the **domain-name** command.

- The number of allowable SSH connections is limited to the maximum number of virtual terminal lines configured for the router. Each SSH connection uses a vty resource.
- From Cisco IOS XR Release 6.3.1 onwards, the **ssh client enable cipher** command is added for backward compatibility with the older Cisco IOS XR versions.
 

For FIPS compliance, in Cisco IOS XR Releases later than 6.2.1, support for weaker ciphers like 3DES and AES CBC was removed and only AES-CTR cipher is supported.
- SSH uses either local security or the security protocol that is configured through AAA on your router for user authentication. When configuring AAA, you must ensure that the console is not running under AAA by applying a keyword in the global configuration mode to disable AAA on the console.



**Note** If you are using Putty version 0.63 or higher to connect to the SSH client, set the 'Chokes on PuTTYs SSH2 winadj request' option under SSH > Bugs in your Putty configuration to 'On.' This helps avoid a possible breakdown of the session whenever some long output is sent from IOS XR to the Putty client.

## Configuring CBC Mode Ciphers

In release 7.0(1), you can enable CBC mode ciphers 3DES-CBC and AES-CBC for SSHv2 server and client connections. The ciphers are disabled by default.

### Step 1 configure



**Example:**

```
RP/0/RSP0/CPU0:router# configure
```

Enters global configuration mode.

**Step 2 ssh server enable cipher aes-cbc 3des-cbc****Example:**

```
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
```

**Step 3 ssh client enable cipher aes-cbc 3des-cbc****Example:**

```
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
```

**Step 4** Use the **commit** or **end** command.

**commit**—Saves the configuration changes and remains within the configuration session.

**end**—Prompts user to take one of these actions:

- **Yes**—Saves configuration changes and exits the configuration session.
- **No**—Exits the configuration session without committing the configuration changes.
- **Cancel**—Remains in the configuration session, without committing the configuration changes.

**Step 5 show ssh session details****Example:**

```
Router# show ssh session details
```

**Configuring CBC Mode Ciphers**

```
/*Enable CBC mode ciphers 3DES-CBC and AES-CBC */
Router# configure
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
Router(config)# commit
```

**Verify CBC Mode Cipher Configuration.**

```
Router# show ssh session details
```

```
Thu Sep  6 10:16:26.346 UTC
SSH version : Cisco-2.0
```

id	key-exchange	pubkey	incipher	outcipher	inmac	outmac
-----						
Incoming Session						
2	ecdh-sha2-nistp256	ssh-rsa	aes128-cbc	aes128-cbc	hmac-sha2-256	hmac-sha2-256

# Configuration Examples for Implementing Secure Shell

This section provides the following configuration example:

## Configuring Secure Shell: Example

This example shows how to configure SSHv2 by creating a hostname, defining a domain name, enabling the SSH server for local and remote authentication on the router by generating a DSA key pair, bringing up the SSH server, and saving the configuration commands to the running configuration file.

From Cisco IOS XR Software Release 7.0.1 and later, the crypto keys are auto-generated at the time of router boot up. Hence, you need to explicitly generate the host-key pair only if it is not present in the router under some scenarios.

After SSH has been configured, the SFTP feature is available on the router.

```
configure
hostname router1
domain name cisco.com
exit
crypto key generate dsa
configure
ssh server
end
```

## Multi-channeling in SSH

The multi-channeling (also called multiplexing) feature on the Cisco IOS XR software server allows you to establish multiple channels over the same TCP connection. Thus, rather than opening a new TCP socket for each SSH connection, all the SSH connections are multiplexed into one TCP connection. For example, with multiplexing support on your XR software server, on a single SSH connection you can simultaneously open a pseudo terminal, remotely execute a command and transfer a file using any file transfer protocol. Multiplexing offers the following benefits:

- You are required to authenticate only once at the time of creating the session. After that, all the SSH clients associated with a particular session use the same TCP socket to communicate to the server.
- Saves time consumed otherwise wasted in creating a new connection each time.

Multiplexing is enabled by default in the Cisco IOS XR software server. If your client supports multiplexing, you must explicitly set up multiplexing on the client for it to be able to send multi-channel requests to the server. You can use OpenSSH, Putty, Perl, WinSCP, Putty, FileZilla, TTSSH, Cygwin or any other SSH-based tool to set up multiplexing on the client. [Configure Client for Multiplexing, on page 19](#) provides an example of how you can configure the client for multiplexing using OpenSSH.

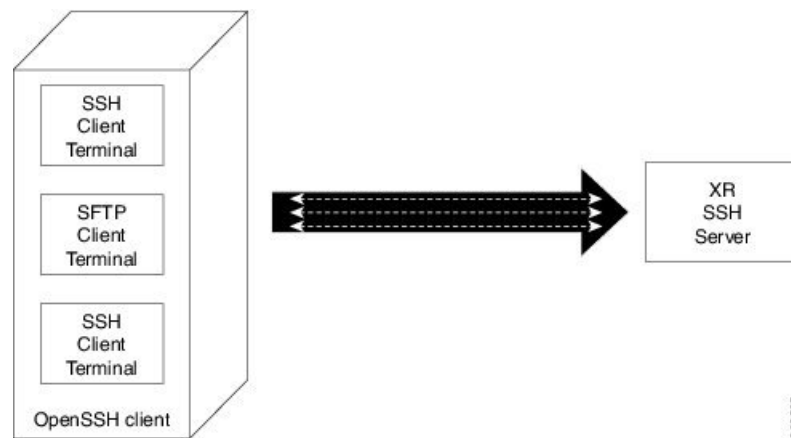
For more information on Multichannel feature, see the Cisco ASR 9000 Series Aggregation Services Router System Security Configuration Guide, Release 5.1.1.

## Restrictions for Multi-channeling Over SSH

- Do not use client multiplexing for heavy transfer of data as the data transfer speed is limited by the TCP speed limit. Hence, for a heavy data transfer it is advised that you run multiple SSH sessions, as the TCP speed limit is per connection.
- Client multiplexing must not be used for more than 15 concurrent channels per session simultaneously.
- You must ensure that the first channel created at the time of establishing the session is always kept alive in order for other channels to remain open.
- The **line template default session-limit** command is not supported for SSH.

## Client and Server Interaction Over Multichannel Connection

The figure below provides an illustration of a client-server interaction over a SSH multichannel connection.



As depicted in the illustration,

- The client multiplexes the collection of channels into a single connection. This allows different operations to be performed on different channels simultaneously. The dotted lines indicate the different channels that are open for a single session.
- After receiving a request from the client to open up a channel, the server processes the request. Each request to open up a channel represents the processing of a single service.



**Note** The Cisco IOX software supports server-side multiplexing only.

## Configure Client for Multiplexing

The SSH client opens up one TCP socket for all the connections. In order to do so, the client multiplexes all the connections into one TCP connection. Authentication happens only once at the time of creating the session. After that, all the SSH clients associated with the particular session uses the same TCP socket to communicate to the server. Use the following steps to configure client multiplexing using OpenSSH:

## SUMMARY STEPS

1. Edit the `ssh_config` file.
2. Add entries **ControlMaster auto** and **ControlPath**
3. Create a temporary folder.

## DETAILED STEPS

	Command or Action	Purpose
<b>Step 1</b>	Edit the <code>ssh_config</code> file.	Open the <code>ssh_config</code> file with your favorite text editor to configure values for session multiplexing. The system-wide SSH configuration file is located under <code>/etc/ssh/ssh_config</code> . The user configuration file is located under <code>~/.ssh/config</code> or <code>\$HOME/.ssh/config</code> .
<b>Step 2</b>	Add entries <b>ControlMaster auto</b> and <b>ControlPath</b>  <b>Example:</b> <pre>Host * ControlMaster auto ControlPath ~/.ssh/tmp/%r@%h:%p</pre>	Add the entry <code>ControlMaster auto</code> and <code>ControlPath</code> to the <code>ssh_config</code> file, save it and exit. <ul style="list-style-type: none"> <li>• <code>ControlMaster</code> determines whether SSH will listen for control connections and what to do about them. Setting the <code>ControlMaster</code> to 'auto' creates a primary session automatically but if there is a primary session already available, subsequent sessions are automatically multiplexed.</li> <li>• <code>ControlPath</code> is the location for the control socket used by the multiplexed sessions. Specifying the <code>ControlPath</code> ensures that any time a connection to a particular server uses the same specified primary connection.</li> </ul>
<b>Step 3</b>	Create a temporary folder.	Create a temporary directory inside the <code>/.ssh</code> folder for storing the control sockets.

## SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm

The Cisco IOS XR software provides a new configuration option to control the key algorithms to be negotiated with the peer while establishing an SSH connection with the router. With this feature, you can enable the insecure SSH algorithms on the SSH server, which are otherwise disabled by default. A new configuration option is also available to restrict the SSH client from choosing the HMAC, or hash-based message authentication codes algorithm while connecting to the SSH server on the router. You can also configure a list of ciphers as the default cipher list, thereby having the flexibility to enable or disable any particular cipher.

Commands introduced:

- [ssh algorithms cipher](#)
- [ssh disable hmac](#)



**Caution** Use caution in enabling the insecure SSH algorithms to avoid any possible security attack.

To disable the HMAC algorithm, use the **ssh client disable hmac** command or **ssh server disable hmac** command in Global Configuration mode.

To enable the required cipher, use the **ssh server enable cipher** command in Global Configuration mode.

The supported encryption algorithms (in the order of preference) are:

1. aes128-ctr
2. aes192-ctr
3. aes256-ctr
4. aes128-gcm@openssh.com
5. aes256-gcm@openssh.com
6. aes128-cbc
7. aes192-cbc
8. aes256-cbc
9. 3des-cbc

In SSH, the CBC-based ciphers are disabled by default. To enable these, you can use the **ssh client enable cipher** command or **ssh server enable cipher** command with the respective CBC options (aes-cbc or 3des-cbc). All CTR-based and GCM-based ciphers are enabled by default.

## Disable HMAC Algorithm

### Configuration Example to Disable HMAC Algorithm

```
Router(config)# ssh server disable hmac hmac-sha1
Router(config)#commit
```

```
Router(config)# ssh client disable hmac hmac-sha1
Router(config)#commit
```

### Running Configuration

```
ssh server disable hmac hmac-sha1
!
```

```
ssh client disable hmac hmac-sha1
!
```

### Related Topics

[SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 20](#)

### Associated Commands

- `ssh client disable hmac`
- `ssh server disable hmac`

## Enable Cipher Public Key

### Configuration Example to Enable Cipher Public Key

To enable all ciphers on the client and the server:

Router 1:

```
Router(config)# ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc  
aes128-ctr aes128-cbc aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
```

Router 2:

```
Router(config)# ssh server algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc  
aes128-ctr aes128-cbc aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
```

To enable the CTR cipher on the client and the CBC cipher on the server:

Router 1:

```
Router(config)# ssh client algorithms cipher aes128-ctr aes192-ctr aes256-ctr
```

Router 2:

```
Router(config)# ssh server algorithms cipher aes128-cbc aes256-cbc aes192-cbc 3des-cbc
```

Without any cipher on the client and the server:

Router 1:

```
Router(config)# no ssh client algorithms cipher
```

Router 2:

```
Router(config)# no ssh server algorithms cipher
```

Enable only the deprecated algorithms on the client and the server:

Router 1:

```
Router(config)# ssh client algorithms cipher aes128-cbc aes192-cbc aes256-cbc 3des-cbc
```

Router 2:

```
Router(config)# ssh server algorithms cipher aes128-cbc aes192-cbc aes256-cbc 3des-cbc
```

Enable the deprecated algorithm (using **enable cipher** command) and enable the CTR cipher (using **algorithms cipher** command) on the client and the server:

Router 1:

```
Router(config)# ssh client enable cipher aes-cbc 3des-cbc
Router(config)# ssh client algorithms cipher aes128-ctr aes192-ctr aes256-ctr
```

Router 2:

```
Router(config)# ssh server enable cipher aes-cbc 3des-cbc
Router(config)# ssh server algorithms cipher aes128-ctr aes192-ctr aes256-ctr
```

### Running Configuration

All ciphers enabled on the client and the server:

Router 1:

```
ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc aes128-ctr aes128-cbc
aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
!
```

Router 2:

```
ssh client algorithms cipher aes256-cbc aes256-ctr aes192-ctr aes192-cbc aes128-ctr aes128-cbc
aes128-gcm@openssh.com aes256-gcm@openssh.com 3des-cbc
!
```

### Related Topics

[SSH Configuration Option to Restrict Cipher Public Key and HMAC Algorithm, on page 20](#)

### Associated Commands

- **ssh client enable cipher**
- **ssh server enable cipher**
- **ssh client algorithms cipher**
- **ssh server algorithms cipher**

## Additional References

The following sections provide references related to implementing secure shell.

**Related Documents**

Related Topic	Document Title
AAA commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Authentication, Authorization, and Accounting Commands on the Cisco ASR 9000 Series Router Software module in System Security Command Reference for Cisco ASR 9000 Series Routers.</i>
AAA configuration tasks	<i>Configuring AAA Services on the Cisco ASR 9000 Series Router Software module in System Security Configuration Guide for Cisco ASR 9000 Series Routers.</i>
Host services and applications commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Host Services and Applications Commands on the Cisco ASR 9000 Series Router module in IP Addresses and Services Command Reference for Cisco ASR 9000 Series Routers.</i>
IPSec commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples  <b>Note</b> IPSec is supported only for Open Shortest Path First version 3 (OSPFv3).	<i>IPSec Network Security Commands on the Cisco ASR 9000 Series Router Software module in System Security Command Reference for Cisco ASR 9000 Series Routers</i>
SSH commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	<i>Secure Shell Commands on the Cisco ASR 9000 Series Router Software module in System Security Command Reference for Cisco ASR 9000 Series Routers</i>

**Standards**

Standards	Title
Draft-ietf-secsh-userauth-17.txt	<i>SSH Authentication Protocol, July 2003</i>
Draft-ietf-secsh-connect-17.txt	<i>SSH Connection Protocol, July 2003</i>
Draft-ietf-secsh-architecture-14.txt	<i>SSH Protocol Architecture, July 2003</i>
Draft-ietf-secsh-transport-16.txt	<i>SSH Transport Layer Protocol, July 2003</i>

**MIBs**

MIBs	MIBs Link
—	To locate and download MIBs using Cisco IOS XR software, use the Cisco MIB Locator found at the following URL and choose a platform under the Cisco Access Products menu: <a href="http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml">http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml</a>



**RFCs**

<b>RFCs</b>	<b>Title</b>
RFC 6020	Netconf/ Yang

**Technical Assistance**

<b>Description</b>	<b>Link</b>
The Cisco Technical Support website contains thousands of pages of searchable technical content, including links to products, technologies, solutions, technical tips, and tools. Registered Cisco.com users can log in from this page to access even more content.	<a href="http://www.cisco.com/techsupport">http://www.cisco.com/techsupport</a>

