Implementing Secure Socket Layer

This module describes how to implement SSL.

The Secure Socket Layer (SSL) protocol and Transport Layer Security (TLS) are application-level protocols that provide for secure communication between a client and server by allowing mutual authentication, the use of hash for integrity, and encryption for privacy. SSL and TLS rely on certificates, public keys, and private keys.

Certificates are similar to digital ID cards. They prove the identity of the server to clients. Certificates are issued by certification authorities (CAs), such as VeriSign or Thawte. Each certificate includes the name of the authority that issued it, the name of the entity to which the certificate was issued, the entity's public key, and time stamps that indicate the certificate's expiration date.

Public and private keys are the ciphers used to encrypt and decrypt information. Although the public key is shared quite freely, the private key is never given out. Each public-private key pair works together: Data encrypted with the public key can be decrypted only with the private key.

✎

Note   For a complete description of the Public Key Infrastructure (PKI) commands used here, see the *Public Key Infrastructure Commands* module in *System Security Command Reference for Cisco NCS 6000 Series Routers*. .

Feature History for Implementing Secure Socket Layer

Release	Modification
Release 5.0.0	This feature was introduced.

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 a crypto package (data encryption standard [DES], 3DES and AES) from Cisco downloaded on your router.

• 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 NCS 6000 Series Routers* publication and *Configuring AAA Services on* Cisco IOS XR Software module in the *System Security Configuration Guide for Cisco NCS 6000 Series Routers* publication.

• AAA authentication and authorization must be configured correctly for Secure Shell File Transfer Protocol (SFTP) to work.

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 1.1.1.1 ?
  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 12.28.46.6 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 XR 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* 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* module in the *System Security Command Reference for Cisco NCS 6000 Series Routers* publication and the *Configuring AAA Services on* module in the *System Security Configuration Guide for Cisco NCS 6000 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.

The Network Configuration Protocol

The Network Configuration Protocol (Netconf) provides mechanisms to install, manipulate, and delete the configuration of network devices. It uses an Extensible Markup Language (XML)-based data encoding for the configuration data as well as the protocol messages. Yang is a data modeling language used with Netconf.

Netconf uses a simple RPC-based (Remote Procedure Call) mechanism to facilitate communication between a client and a server. The client can be a script or application typically running as part of a network manager. The server is typically a network device.

The configuration of features need not be done the traditional way (using CLIs), the client application (controller) reads the Yang model and communicates with the Netconf server (IOS XR) accordingly.

✎

Note Following are the deviations from IETF-NACM YANG, where the system does not support:

- The *ordered-by-user* functionality for rule-lists and rules. rule-lists & rules are sorted based on name.

- The *enable-nacm* leaf.

- The *notification* related leafs (notification-name & denied-notifications.)

Netconf Sessions and Operations

A Netconf session is the logical connection between a network configuration application and a network device. A device should be capable of supporting multiple sessions and atleast one Netconf session.

Characteristics of a netconf session:

- Netconf is connection-oriented - SSH is the underlying transport.

- The netconf client establishes session with the server.

- Netconf sessions are established with the *hello* message. Features and capabilities are announced.

- Sessions can be terminated using the *close* or *kill* messages.

Basic Netconf operations:

- Get configuration <get-config>

- Get all information <get>

- Edit configuration <edit-config>

- Copy configuration <copy-config>

  ✎

  Note <copy-config> does not support source attribute with "data store" at present.

- <lock>, <unlock>

- <kill-session>

- <close-session>

- Commit configuration <commit>

The Yang data model

Each feature has a defined Yang Model which is synthesized from the schemas. A model is published in a tree format and includes:

- Top level nodes and their subtrees

- Subtrees that augment nodes in other yang models

```
Example: The aaa Yang model
module: Cisco-IOS-XR-aaa-lib-cfg
   +--rw aaa
      +--rw accountings
      |  +--rw accounting* [type listname]
      |     +--rw type                xr:Cisco-ios-xr-string
      |     +--rw listname            xr:Cisco-ios-xr-string
      |     +--rw rp-failover?        Aaa-accounting-rp-failover
      |     +--rw broadcast?          Aaa-accounting-broadcast
      |     +--rw type-xr?            Aaa-accounting
      |     +--rw method*             Aaa-method
      |     +--rw server-group-name*  string
      +--rw authorizations
      |  +--rw authorization* [type listname]
      |     +--rw type                xr:Cisco-ios-xr-string
      |     +--rw listname            xr:Cisco-ios-xr-string
      |     +--rw method*             Aaa-method
      |     +--rw server-group-name*  string
      +--rw accounting-update!
      |  +--rw type                Aaa-accounting-update
      |  +--rw periodic-interval?  uint32
      +--rw authentications
         +--rw authentication* [type listname]
            +--rw type                xr:Cisco-ios-xr-string
            +--rw listname            xr:Cisco-ios-xr-string
            +--rw method*             Aaa-method
            +--rw server-group-name*  string
```

Advantages of using the Yang model are:

- Yang supports programmatic interfaces.

- Yang supports simplified network management applications.

- Yang supports interoperability that provides a standard way to model management data.

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 to Step 4 are optional.

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	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**hostname** *hostname* **Example:** `RP/0/RP0/CPU0:router(config)# hostname router1`	Configures a hostname for your router.

	Command or Action	Purpose
Step 3	**domain name** *domain-name* **Example:** RP/0/RP0/CPU0:router(config)# domain name cisco.com	Defines a default domain name that the software uses to complete unqualified host names.
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	**crypto key generate rsa** [**usage keys** | **general-keys**] [*keypair-label*] **Example:** RP/0/RP0/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. • To delete the RSA key pair, use the **crypto key zeroize rsa** command. • This command is used for SSHv1 only.
Step 6	**crypto key generate dsa** **Example:** RP/0/RP0/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. • The recommended minimum modulus size is 1024 bits. • Generates a DSA key pair. To delete the DSA key pair, use the **crypto key zeroize dsa** command. • This command is used only for SSHv2.
Step 7	**crypto key generate ecdsa** [**nistp256** | **nistp384** | **nistp521**] **Example:** RP/0/RP0/CPU0:router# crypto key generate ecdsa nistp256	Generates an ECDSA key pair. The supported ECDSA curve types are: Nistp256, Nistp384 and Nistp521. • To delete the ECDSA key pair, use the **crypto key zeroize ecdsa [nistp256 | nistp384 | nistp521]** command. • This command is used for SSHv2 only.
Step 8	**configure** **Example:**	Enters XR Config mode.

	Command or Action	**Purpose**
	`RP/0/RP0/CPU0:router# configure`	
Step 9	**ssh timeout** *seconds* **Example:** `RP/0/RP0/CPU0:router(config)# ssh timeout 60`	(Optional) Configures the timeout value for user authentication to AAA. • If the user fails to authenticate itself to AAA within the configured time, the connection is terminated. • If no value is configured, the default value of 30 seconds is used. The range is from 5 to 120.

Step 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**
	
	Example:
	
	`RP/0/RP0/CPU0:router(config)# ssh`
	
	or
	
	`RP/0/RP0/CPU0:router(config)# ssh server v2`
		• (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 **no** form of this command. If no VRF is specified, the default is assumed.
		
		Note The SSH server can be configured for multiple VRF usage.
		
		• (Optional) Forces the SSH server to accept only SSHv2 clients if you configure the SSHv2 option by using the **ssh server v2** command. If you choose the **ssh server v2** command, only the SSH v2 client connections are accepted.

Step 11	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 12	**show ssh** **Example:** `RP/0/RP0/CPU0:router# show ssh`	(Optional) Displays all of the incoming and outgoing SSHv1 and SSHv2 connections to the router.

	Command or Action	**Purpose**
Step 13	**show ssh session details** **Example:** `RP/0/RP0/CPU0:router# show ssh session details`	(Optional) Displays a detailed report of the SSHv2 connections to and from the router.
Step 14	**show ssh history** **Example:** `RP/0/RP0/CPU0:router# show ssh history`	(Optional) Displays the last hundred SSH connections that were terminated.
Step 15	**show ssh history details** **Example:** `RP/0/RP0/CPU0:router# show ssh history details`	(Optional) Displays the last hundred SSH connections that were terminated with additional details. This command is similar to **show ssh session details** command but also mentions the start and end time of the session.
Step 16	**show tech-support ssh** **Example:** `RP/0/RP0/CPU0:router# show tech-support ssh`	(Optional) Automatically runs the `show` 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

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* | *ipv6-address* | *hostname*} [**username** *user-* **cipher** | **source-interface** *type instance*]

DETAILED STEPS

	Command or Action	**Purpose**
Step 1	**configure** **Example:** RP/0/RP0/CPU0:router# configure	Enters XR Config mode.
Step 2	**ssh client knownhost** *device* **:** /*filename* **Example:** RP/0/RP0/CPU0:router(config)# ssh client knownhost slot1:/server_pubkey	(Optional) Enables the feature to authenticate and check the server public key (pubkey) at the client end. **Note** The complete path of the filename is required. The colon (:) and slash mark (/) are also required.
Step 3	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 4	**ssh** {*ipv4-address* | *ipv6-address* | *hostname*} [**username** *user-* **cipher** | **source-interface** *type instance*]	Enables an outbound SSH connection. • 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. • 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. • The SSHv1 client supports only the 3DES encryption algorithm option, which is still available by default for those SSH clients only. • If the *hostname* argument is used and the host has both IPv4 and IPv6 addresses, the IPv6 address is used.

• 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 or that you have not successfully generated an RSA key pair for your router. 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.

- 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.

Enabling NETCONF over SSH

This task enables NETCONF over SSH. SSH is currently the only supported transport method .

If the client supports, Netconf over ssh can utilize the multi-channeling capabilities of IOS XR ssh server. For additional details about Multi-channeling in SSH, see *Implementing Secure Shell* in *System Security Configuration Guide*.

Prerequisites:

- k9sec pie must be installed, otherwise the port configuration for the netconf ssh server cannot be completed. (The Netconf subsystem for SSH, as well as, SSH cannot be configured without the k9sec pie.)

- Crypto keys must be generated prior to this configuration.

- The Netconf-YANG feature is packaged in the mgbl pie, which must be installed before enabling the Netconf-YANG agent.

SUMMARY STEPS

1. **configure**
2. **netconf-yang agent ssh**
3. **ssh server netconf** [**vrf** *vrf-name* [**ipv4 access-list***ipv4 access list name*] [**ipv6 access-list** *ipv6 access list name*]]
4. **ssh server netconf port** *port-number*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure** **Example:** `RP/0/RP0/CPU0:router# configure`	Enters XR Config mode.
Step 2	**netconf-yang agent ssh** **Example:** `RP/0/RP0/CPU0:router (config) # netconf agent ssh`	Enables NETCONF agent over SSH connection. After NETCONF is enabled, the Yang model in the controllcker, can configure the relevant models. **Note** The Yang models can be retrieved from the router via NETCONF <get-schema> operation.
Step 3	**ssh server netconf** [**vrf** *vrf-name* [**ipv4 access-list** *ipv4 access list name*] [**ipv6 access-list** *ipv6 access list name*]] **Example:** `RP/0/RP0/CPU0:router (config) # ssh server netconf vrf netconfvrf ipv4 access-list InternetFilter`	Brings up the netconf subsytem support with 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. Optionally ACLs for IPv4 and IPv6 can be used to restrict access to the netconf subsystem of the ssh server before the port is opened. **Note** The netconf subsystem support with SSH server can be configured for use with multiple VRFs .
Step 4	**ssh server netconf port** *port-number* **Example:** `RP/0/RP0/CPU0:router (config) # ssh server netconf port 830`	Configures a port for the netconf ssh server. This command is optional. If no port is specified, port 830 is uses by default. **Note** 830 is the IANA-assigned TCP port for NETCONF over SSH, but it can be changed using this command.

What to do next

The **show netconf-yang statistics** command and **show netconf-yang clients** command can be used to verify the configuration details of the netconf agent.

The **clear netconf-yang agent session** command clears the specified Netconf session (on the Netconf server side).

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.

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
```

Examples: Netconf over SSH

This section illustrates some examples relevant to Netconf:

Enabling netconf-yang for ssh transport and netconf subsystem for default vrf with default port (830)

```
config
netconf-yang agent ssh
ssh server netconf vrf default
  !
!
```

Enabling netconf-yang for ssh transport and netconf subsystem for vrf *green* and vrf *red* with netconf port (831)

```
config
netconf-yang agent ssh
!
ssh server netconf vrf green
ssh server netconf vrf red
ssh server netconf port 831
  !
!
```

Show command outputs

```
show netconf-yang statistics
Summary statistics          requests|            total time|   min time per request|    max
time per request|   avg time per request|
other                             0|      0h  0m  0s   0ms|      0h  0m  0s   0ms|
 0h  0m  0s   0ms|      0h  0m  0s   0ms|
close-session                     4|      0h  0m  0s   3ms|      0h  0m  0s   0ms|
 0h  0m  0s   1ms|      0h  0m  0s   0ms|
kill-session                      0|      0h  0m  0s   0ms|      0h  0m  0s   0ms|
 0h  0m  0s   0ms|      0h  0m  0s   0ms|
get-schema                        0|      0h  0m  0s   0ms|      0h  0m  0s   0ms|
 0h  0m  0s   0ms|      0h  0m  0s   0ms|
get                               0|      0h  0m  0s   0ms|      0h  0m  0s   0ms|
 0h  0m  0s   0ms|      0h  0m  0s
get-config                        1|      0h  0m  0s   1ms|      0h  0m  0s   1ms|
 0h  0m  0s   1ms|      0h  0m  0s   1ms|
edit-config                       3|      0h  0m  0s   2ms|      0h  0m  0s   0ms|
```

```
       0h  0m  0s   1ms|        0h  0m  0s   0ms|
commit                            0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
       0h  0m  0s   0ms|        0h  0m  0s   0ms|
cancel-commit                     0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
       0h  0m  0s   0ms|        0h  0m  0s   0ms|
lock                              0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
       0h  0m  0s   0ms|        0h  0m  0s   0ms|
unlock                            0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
       0h  0m  0s   0ms|        0h  0m  0s   0ms|
discard-changes                   0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
       0h  0m  0s   0ms|        0h  0m  0s   0ms|
validate                          0|       0h  0m  0s   0ms|       0h  0m  0s   0ms|
       0h  0m  0s   0ms|        0h  0m  0s   0ms|

show netconf-yang clients
client session ID|   NC version|    client connect time|        last OP time|        last
OP type|     <lock>|
22969|                  1.1|        0d  0h  0m  2s|            11:11:24|
close-session|       No|
15389|                  1.1|        0d  0h  0m  1s|            11:11:25|        get-config|
            No|
```

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	*Authentication, Authorization, and Accounting Commands on* module in *System Security Command Reference for Cisco NCS 6000 Series Routers*.
AAA configuration tasks	*Configuring AAA Services on* module in *System Security Configuration Guide for Cisco NCS 6000 Series Routers*.
Host services and applications commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Host Services and Applications Commands on* module in *IP Addresses and Services Command Reference for Cisco NCS 6000 Series Routers*.
IPSec commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*IPSec Network Security Commands on* module in *System Security Command Reference for Cisco NCS 6000 Series Routers*
SSH commands: complete command syntax, command modes, command history, defaults, usage guidelines, and examples	*Secure Shell Commands on* module in *System Security Command Reference for Cisco NCS 6000 Series Routers*

Standards

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

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: http://cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

RFCs

RFCs	Title
RFC 6020	Netconf/ Yang

Technical Assistance

Description	Link
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.	http://www.cisco.com/ techsupport