Configuring SSH and Telnet

This chapter describes how to configure Secure Shell Protocol (SSH) and Telnet on Cisco NX-OS devices.

This chapter includes the following sections:

About SSH and Telnet

This section includes information about SSH and Telnet.

SSH Server

You can use the SSH server to enable an SSH client to make a secure, encrypted connection to a Cisco NX-OS device. SSH uses strong encryption for authentication. The SSH server in the Cisco NX-OS software can interoperate with publicly and commercially available SSH clients.

The user authentication mechanisms supported for SSH are RADIUS, TACACS+, LDAP, and the use of locally stored usernames and passwords.

SSH Client

The SSH client feature is an application that runs over the SSH protocol to provide device authentication and encryption. The SSH client enables a Cisco NX-OS device to make a secure, encrypted connection to another Cisco NX-OS device or to any other device that runs the SSH server. This connection provides an outbound

connection that is encrypted. With authentication and encryption, the SSH client allows for a secure communication over an insecure network.

The SSH client in the Cisco NX-OS software works with publicly and commercially available SSH servers.

SSH Server Keys

SSH requires server keys for secure communications to the Cisco NX-OS device. You can use SSH server keys for the following SSH options:

- SSH version 2 using Rivest, Shamir, and Adelman (RSA) public-key cryptography
- SSH version 2 using the Digital System Algrorithm (DSA)
- SSH version 2 using the Elliptic Curve Digital Signature Algorithm (ECDSA)

Be sure to have an SSH server key-pair with the appropriate version before enabling the SSH service. You can generate the SSH server key-pair according to the SSH client version used. The SSH service accepts the following types of key-pairs for use by SSH version 2:

- The **dsa** option generates the DSA key-pair for the SSH version 2 protocol.
- The **rsa** option generates the RSA key-pair for the SSH version 2 protocol.
- The **ecdsa** option generates the ECDSA key-pair for the SSH version 2 protocol.

By default, the Cisco NX-OS software generates an RSA key using 1024 bits.

SSH supports the following public key formats:

- OpenSSH
- IETF Secure Shell (SECSH)
- Public Key Certificate in Privacy-Enhanced Mail (PEM)

Caution If you delete all of the SSH keys, you cannot start the SSH services.

SSH Authentication Using Digital Certificates

SSH authentication on Cisco NX-OS devices provide X.509 digital certificate support for host authentication. An X.509 digital certificate is a data item that ensures the origin and integrity of a message. It contains encryption keys for secured communications and is signed by a trusted certification authority (CA) to verify the identity of the presenter. The X.509 digital certificate support provides either DSA or RSA algorithms for authentication.

The certificate infrastructure uses the first certificate that supports the Secure Socket Layer (SSL) and is returned by the security infrastructure, either through a query or a notification. Verification of certificates is successful if the certificates are from any of the trusted CAs configured and if not revoked or expired.

You can configure your device for SSH authentication using an X.509 certificate. If the authentication fails, you are prompted for a password.

You can configure SSH authentication using X.509v3 certificates (RFC 6187). X.509v3 certificate-based SSH authentication uses certificates combined with a smartcard to enable two-factor authentication for Cisco device access. The SSH client is provided by Cisco partner Pragma Systems.

SSH Authentication Using Host Identity Based Authorization (HIBA)

Host-Based Authentication is an SSH authentication method that authenticates the client's host to the server (Cisco Nexus 9000 switch) by verifying the client's host public key in the server's `known_hosts` file. This is distinct from SSH Certificate-Based Authentication, which uses digital certificates signed by a Certificate Authority (CA) to authenticate users or hosts.

Host Identity Based Authorization (HIBA) is a method that centralizes SSH authorization management by embedding host authorization information within certificates.

- Host authorization information is embedded in the host certificate.
- User certificates contain "grants" specifying permitted access.
- Authorization is managed centrally by a Certificate Authority (CA).

HIBA simplifies SSH access control, reduces administrative overhead, and eliminates dependencies on external AAA servers for authorization.

Benefits of HIBA

HIBA offers several advantages over traditional SSH key management:

Key benefits of HIBA include:

- **Simplified Management:** Centralized authorization through certificate-based identity simplifies management.
- **Scalability:** Easier management of SSH access in large, complex environments.
- **Reduced Dependencies:** Eliminates the dependency on external AAA servers for authorization, making it suitable for last-resort access.
- **Enhanced Security:** Improves control over temporary and privileged access with short-lived certificates.

How SSH Authentication with HIBA Works

This process describes how SSH authentication occurs when HIBA is configured.

Summary

The SSH server invokes the HIBA authorization module to process user certificates during authentication. Access is granted if the HIBA module successfully validates the user's certificate against the configured host identity and grants. If HIBA validation fails, the SSH server may fall back to other authentication methods, depending on the configuration.

Workflow

These stages describe the SSH authentication process with HIBA:

1. **SSH Connection Attempt** - A user attempts to connect to the switch via SSH.

2. **Certificate Presentation** - The SSH client presents the user's certificate to the SSH server on the switch.
3. **HIBA Module Invocation** - The SSH server, based on its configuration (AuthorizedPrincipalsCommand), invokes the HIBA authorization module.
4. **Certificate Validation** - The HIBA module performs the following validations:
   - Verifies the user certificate's signature against the configured HIBA CA.
   - Extracts the host identity from the host certificate.
   - Checks for a valid "grant" in the user certificate that matches the host identity.
5. **Access Decision** - Based on the HIBA module's validation, one of the following occurs:

When...	And...	Then...	And...
The user certificate is successfully validated by the HIBA module	A valid grant for the target host is found in the user certificate	Access is granted to the user.	The SSH session proceeds.
The user certificate is invalid or cannot be validated.	No valid grant is found in the user certificate.	Access is denied by the HIBA module.	The SSH server may fall back to other authentication methods (if configured).

Configuring HIBA for SSH Authentication

This steps guides you through the configuration of SSH Host Identity Based Authorization (HIBA).

This configuration involves generating SSH server keys, configuring a trustpoint for the HIBA CA, enrolling the SSH host certificate, and configuring the SSH server to use HIBA for authentication.

Note To configure HIBA for the first time, you can log in to the switch using traditional SSH authentication methods, such as local user accounts or other configured AAA servers. Enabling HIBA does not remove or block existing local SSH users unless you explicitly delete those accounts.

Before you begin

Before configuring HIBA, ensure that you have:

- A functional PKI infrastructure, including a Certificate Authority (CA).
- Connectivity to the CA server.

Procedure

Step 1 **configure terminal**

Example:

```
switch# configure terminal
```

Enter global configuration mode.

Step 2 **ssh key ecdsa** *bits*

Example:

```
switch(config)# ssh key ecdsa 384
```

Generate ECDSA keypair for the switch. This example uses a 384-bit ECDSA key. Use a key size supported by your security policy and platform.

Step 3 **ssh key export** *bootflash:file_name* **ecdsa**

Example:

```
switch(config)# ssh key export bootflash:host_key ecdsa
            Enter Passphrase:
```

Export the SSH host ECDSA key to bootflash. Replace `file_name` as needed.

After export, transfer `host_key` and `host_key.pub` files to your CA machine using SFTP:

```
switch(config)# feature sftp-server
# On CA machine:
sftp admin@<switch_ip>
sftp> get host_key .
sftp> get host_key.pub .
```

Step 4 **crypto ca trustpoint** *openssh-ca* **type ssh**

Example:

```
switch(config)# crypto ca trustpoint openssh-ca type ssh
```

Create a trustpoint for the HIBA CA. Use the name **openssh-ca** for consistency.

Step 5 **crypto ca authenticate** *openssh-ca* **type ssh** *ecdsa-sha2-nistp384 public_key*

Example:

```
switch(config-trustpoint)# crypto ca authenticate openssh-ca type ssh ecdsa-sha2-nistp384
AAAAE2VjZHNhLXNoYTItbmlzdHAzODQAAAAIbmlzdHAzODQAAABhBBPiMs3fwftVUoMT... /home/admin/.hiba-ca
 CA
```

Authenticate the HIBA CA by importing the CA public key. Replace the key string with your actual CA public key.

Step 6 **crypto ca enroll** *openssh-ca* **type ssh host-certificate** *ecdsa-sha2-nistp384-cert-v01@openssh.com certificate_content*

Example:

```
switch(config)# crypto ca enroll openssh-ca type ssh host-certificate
ecdsa-sha2-nistp384-cert-v01@openssh.com
[redacted]
 root@switch
```

Enroll the SSH host certificate signed by your CA. Use the certificate content generated as per the Google HIBA CA wiki instructions.

Example Configuration: HIBA SSH Client on Linux

Important The following steps are provided as an **example** for configuring a HIBA SSH client on a Linux system. The exact procedure and output may vary depending on your client operating system and SSH version. Consult your system's official SSH documentation for authoritative instructions.

This steps guides you through the client-side configuration for using Host Identity Based Authorization (HIBA) with SSH.

Note The term "HIBA server" refers to the SSH server running on the Cisco Nexus 9000 switch, configured to use HIBA.

Before you begin

Before configuring the HIBA SSH client, ensure that you have:

- A valid installation of `openssh-client` on your host.
- The CA public key (`ca.pub`).
- Your user private key and matching certificate with a valid HIBA extension.
- Your user public key (`key_rsa.pub` or equivalent).

Procedure

Step 1 **$ cat /etc/ssh/ssh_config**

Example:

```
$ cat /etc/ssh/ssh_config
# Enable host key checking
StrictHostKeyChecking yes
# Declare our trusted CA
GlobalKnownHostsFile /etc/ssh/known_hosts
```

Configure SSH client settings

Edit `/etc/ssh/ssh_config` to enable strict host key checking and specify a `GlobalKnownHostsFile` that will contain your CA public key for SSH certificate validation.

Step 2 **$ echo "@cert-authority * $(cat /etc/ssh/ca.pub)" > /etc/ssh/known_hosts**

Example:

```
$ echo "@cert-authority * $(cat /etc/ssh/ca.pub)" > /etc/ssh/known_hosts
```

Populate **known_hosts** with CA public key

Add the CA public key to the `known_hosts` file using the `@cert-authority` directive. This step ensures the SSH client trusts any host certificate signed by this CA.

Step 3 **$ cat ~/.ssh/key_rsa.pub**

Example:

```
$ cat ~/.ssh/key_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQE... user@host
```

View your user public key

Display the contents of your user public key file. This key is required for certificate-based authentication and should correspond to your private key and certificate.

Note
If your key has a different name or location, adjust the path accordingly.

Step 4 **$ ssh -i <path_to_private_key> <user>@<hiba_server_ip>**

Example:

```
$ ssh -i <path_to_private_key> <user>@<hiba_server_ip>
```

Connect to the HIBA-enabled SSH server

Use your private key (and its matching certificate, if required) to connect to the SSH server.

Note
The `-i` option specifies the user's private key (identity file).

If configured correctly, the SSH connection should be established using HIBA certificate-based authentication, and host validation will succeed against the CA public key. Passwordless login will be possible if the public key is present in `authorized_keys` on the server.

Verifying HIBA Configuration

Procedure

Step 1 **show crypto ca certificates type ssh**

Example:

```
switch(config)# show crypto ca certificates type ssh
            trustpoint: openssh-ca
            CA Public Key:
            ecdsa-sha2-nistp384
[illegible]
 /home/admin/.hiba-ca CA
            Finger Print:
            384 SHA256:ZcJws/mPrts6twB29OoZU/c3AMAL0x3mUp00YxwSRmk /home/admin/.hiba-ca CA
 (ECDSA)

            Host Certificate:
            Type: ecdsa-sha2-nistp384-cert-v01@openssh.com host certificate
            Public key: ECDSA-CERT SHA256:bZkNWnvyxUK1DHRwqayWivobGUwA25GRGkUMNEd/Ujw
            Signing CA: ECDSA SHA256:ZcJws/mPrts6twB29OoZU/c3AMAL0x3mUp00YxwSRmk (using
ecdsa-sha2-nistp384)
            Key ID: "cisco_nexus_9000"
            Serial: 1
```

```
Valid: from 2025-06-05T04:34:00 to 2025-08-28T04:35:39
Principals:
cisco_nexus_9000
Critical Options: (none)
Extensions:
identity@hibassh.dev

HIBA Info:
certificate 'cisco_nexus_9000' (1 principal) contains 1 HIBA grant
principal: 'cisco_nexus_9000'
identity@hibassh.dev (v2):
[0] domain = 'google.com'
```

Display the SSH certificates and verify that the host certificate is enrolled and associated with the correct trustpoint (`openssh-ca`).

Expected Output: The output should display the SSH host certificate details, including the "HIBA Info" section, which shows the HIBA grants.

If the host certificate and HIBA information are displayed correctly, the certificate enrollment is successful.

Step 2 **show crypto ca trustpoints type ssh**

Example:

```
switch(config)# show crypto ca trustpoints type ssh
            trustpoint: openssh-ca
```

Display the SSH trustpoints and verify that the HIBA CA trustpoint (`openssh-ca`) is present.

Expected Output: The output should list the trustpoint names of type `ssh`.

If the HIBA CA trustpoint appears in the output, the trustpoint has been configured successfully.

Step 3 **ssh -i path_to_private_key> <user>@<switch_ip**

Example:

```
ssh -i /home/admin/.hiba-ca/users/google-user admin@10.126.67.44
```

Attempt to SSH to the switch using a user with a HIBA-enabled certificate signed by the CA.

Note: The **-i** option specifies the path to the user's `private key` (identity file). The HIBA extension must be included in the certificate that pairs with this private key, and the CA public key must be trusted by the switch. Ensure the private key file is kept secure.

The SSH connection should be established successfully without prompting for a password (if password authentication is disabled).

Telnet Server

The Telnet protocol enables TCP/IP connections to a host. Telnet allows a user at one site to establish a TCP connection to a login server at another site and then passes the keystrokes from one device to the other. Telnet can accept either an IP address or a domain name as the remote device address.

The Telnet server is disabled by default on the Cisco NX-OS device.

Prerequisites for SSH and Telnet

Make sure that you have configured IP on a Layer 3 interface, out-of-band on the mgmt 0 interface, or inband on an Ethernet interface.

Guidelines and Limitations for SSH and Telnet

SSH and Telnet have the following configuration guidelines and limitations:

- The Cisco NX-OS software supports only SSH version 2 (SSHv2).
- Beginning with Cisco NX-OS Release 10.4(3)F, the Cisco Nexus 9000 Series switches support SSH authorization using X.509 certificates through a TACACS+ server. This feature is not supported with RADIUS.
- When you use the **no feature ssh feature** command, port 22 is not disabled . Port 22 is always open and a deny rule is pushed to deny all incoming external connections.
- Due to a Poodle vulnerability, SSLv3 is no longer supported.
- IPSG is not supported on the following:
  - The last six 40-Gb physical ports on the Cisco Nexus 9372PX, 9372TX, and 9332PQ switches
  - All 40G physical ports on the Cisco Nexus 9396PX, 9396TX, and 93128TX switches
- You can configure your device for SSH authentication using an X.509 certificate. If the authentication fails, you are prompted for a password.
- The SFTP server feature does not support the regular SFTP **chown** and **chgrp** commands.
- When the SFTP server is enabled, only the admin user can use SFTP to access the device.
- SSH public and private keys imported into user accounts that are remotely authenticated through a AAA protocol (such as RADIUS or TACACS+) for the purpose of SSH Passwordless File Copy will not persist when the Nexus device is reloaded unless a local user account with the same name as the remote user account is configured on the device before the SSH keys are imported.
- SSH timeout period must be longer than the time of the tac-pac generation time. Otherwise, the VSH log might show %VSHD-2-VSHD_SYSLOG_EOL_ERR error. Ideally, set to 0 (infinity) before collecting tac-pac or showtech.
- Beginning with Cisco NX-OS Release 10.4(3)F, the **show running-config all** command does not display the details of the following commands:
  - no feature telnet
  - no feature nxdb
  - no feature scp-server
  - no feature sftp-server

Note If you are familiar with the Cisco IOS CLI, be aware that the Cisco NX-OS commands for this feature might differ from the Cisco IOS commands that you would use.

- Beginning with Cisco NX-OS Release 10.2(2)F, a new desynchronization CLI is introduced to provide you an option to disable the user synchronization between the SNMP and the security components. For more information, refer to the *Configuring SNMP* chapter in the *System Management Configuration Guide*.

  For more information about the Cisco Nexus 9000 switches that support various features spanning from release 7.0(3)I7(1) to the current release, refer to Nexus Switch Platform Support Matrix.

- When the desynchronization CLI is enabled, remote users will not be synced to SNMP database.

- The security users created using DCNM (also called as Nexus Dashboard Fabric Controller from Release 12.0.1a) will not have a corresponding SNMPv3 profile when the desynchronization CLI is enabled. When the synchronization is disabled, the users created on the security component can log in to the switch, but the switches will not be discovered by the controller, as the controller uses the SNMP configuration created for the security user to discover the switch. Furthermore, the SNMP does not recognize the security users created due to the desynchronized state of the userDB, resulting in failure to discover the switch. Therefore, to have the switches discovered by the controller, the SNMP user must be explicitly created. It is not recommended to use the desynchronization CLI along with DCNM functionality. For more information, refer to the *Cisco Nexus 9000 NX-OS Security Configuration Guide*.

- Beginning with Cisco NX-OS Release 10.6(1)F, the following DSA algorithm and all DSA-related SSH CLI commands are deprecated:

  - **show ssh key dsa**
  - **[no] ssh key dsa**
  - **[no] username username keypair generate {dsa [force]}**
  - **username username keypair export {bootflash:filename | volatile:filename} {dsa} [force]**
  - **username username keypair import {bootflash:filename | volatile:filename} {dsa} [force]**
  - **username user-id ssh-cert-dn dn-name {dsa}**
  - **[no] ssh cipher-mode weak**
  - **ssh ciphers aes256-gcm** - Upon executing this command, the following warning will be shown: Undefined algorithm name

Note For deprecated DSA and cipher-mode CLI, depreciation warning will not be shown during CLI Config replace, Dual-Stage commit and netconf operation. For enhanced security, generate and use RSA or ECDSA keys for SSH authentication and management.

Default Settings for SSH and Telnet

This table lists the default settings for SSH and Telnet parameters.

Table 1: Default SSH and Telnet Parameters

Parameters	Default
SSH server	Enabled
SSH server key	RSA key generated with 1024 bits
RSA key bits for generation	1024
Telnet server	Disabled
Telnet port number	23
Maximum number of SSH login attempts	3
SCP server	Disabled
SFTP server	Disabled

Configuring SSH

This section describes how to configure SSH.

Generating SSH Server Keys

You can generate an SSH server key based on your security requirements. The default SSH server key is an RSA key that is generated using 1024 bits.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**no feature ssh** **Example:** `switch(config)# no feature ssh`	Disables SSH.
Step 3	**ssh key export** *export-host-keypath* {**rsa** | **ecdsa**} [**force**]	Exports the SSH server key.

	Command or Action	Purpose

	Example:
	`switch(config)# ssh key rsa export bootflash:host_key rsa`
	`Enter Passphrase:`
		If you want to export SSH server key to an existing file path, use the **force** keyword.

Step 4	**ssh rekey max-data** *max-data* **max-time** *max-time*i **Example:** `switch(config)# ssh rekey max-data 1K max-time 1M`	Configures the rekey parameters.
Step 5	**feature ssh** **Example:** `switch(config)# feature ssh`	Enables SSH.

Step 6	**exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 7	(Optional) **show ssh key** [**dsa** | **rsa** | **ecdsa**] [**md5**] **Example:** `switch# show ssh key`	Displays the SSH server keys. This command displays the fingerprint in SHA256 format by default. SHA256 is more secure than the old default format of MD5. However, the **md5** option has been added, if you want to see the fingerprint in MD5 format for backward compatibility. **Note** Begining with Cisco NX-OS Release 10.6(1)F, dsa key word is hidden. If you enter the full command, NX-OS will accept it with a deprecation warning.
Step 8	**show run security all**	
Step 9	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Specifying the SSH Public Keys for User Accounts

You can configure an SSH public key to log in using an SSH client without being prompted for a password. You can specify the SSH public key in one of these formats:

- OpenSSH format
- IETF SECSH format

Specifying the SSH Public Keys in IETF SECSH Format

You can specify the SSH public keys in IETF SECSH format for user accounts.

Before you begin

Generate an SSH public key in IETF SCHSH format.

Procedure

	Command or Action	Purpose
Step 1	**copy** *server-file* **bootflash:***filename* **Example:** `switch# copy tftp://10.10.1.1/secsh_file.pub bootflash:secsh_file.pub`	Downloads the file containing the SSH key in IETF SECSH format from a server. The server can be FTP, secure copy (SCP), secure FTP (SFTP), or TFTP.
Step 2	**configure terminal** **Example:** `switch# configure terminal switch(config)#`	Enters global configuration mode.
Step 3	**username** *username* **sshkey file bootflash:***filename* **Example:** `switch(config)# username User1 sshkey file bootflash:secsh_file.pub`	Configures the SSH public key in IETF SECSH format.
Step 4	**exit** **Example:** `switch(config)# exit switch#`	Exits global configuration mode.
Step 5	(Optional) **show user-account** **Example:** `switch# show user-account`	Displays the user account configuration.
Step 6	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Specifying the SSH Public Keys in OpenSSH Format

You can specify the SSH public keys in OpenSSH format for user accounts.

Before you begin

Generate an SSH public key in OpenSSH format.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**username** *username* **sshkey** *ssh-key*
	Example:
	`switch(config)# username User1 sshkey ssh-rsa`
	`AAAAB3NzaC1yc2EAAAABIwAAAIEAyi9cF6QsZl9G+3flXsuK3OiWH7YjUuA50rv7gsEPJ`
	`hOBmsiGFAVkilnIf/DQhum+LNqP/eLowb7ioO+lVRRFY/G+LNIQW8g9igG30c6k6+`
	`XVn+NjnIlB7ihvpVh7dLddMOXwOnXHYshXmSiH3UD/vKyziEh5S4Tplx8=`
		Configures the SSH public key in OpenSSH format.

Step 3	**exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 4	(Optional) **show user-account** **Example:** `switch# show user-account`	Displays the user account configuration.
Step 5	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring a Maximum Number of SSH Login Attempts

You can configure the maximum number of SSH login attempts. If the user exceeds the maximum number of permitted attempts, the session disconnects.

Note The total number of login attempts includes attempts through public-key authentication, certificate-based authentication, and password-based authentication. If public-key authentication is enabled, it takes priority. If only certificate-based and password-based authentication are enabled, certificate-based authentication takes priority. If you exceed the configured number of login attempts through all of these methods, a message appears indicating that too many authentication failures have occurred.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**ssh login-attempts** *number* **Example:** `switch(config)# ssh login-attempts 5`	Configures the maximum number of times that a user can attempt to log into an SSH session. The default maximum number of login attempts is 3. The range is from 1 to 10. **Note** The **no** form of this command removes the previous login attempts value and sets the maximum number of login attempts to the default value of 3.
Step 3	(Optional) **show running-config security all** **Example:** `switch(config)# show running-config security all`	Displays the configured maximum number of SSH login attempts.
Step 4	(Optional) **copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	(Optional) Copies the running configuration to the startup configuration.

Starting SSH Sessions

You can start SSH sessions using IPv4 or IPv6 to connect to remote devices from the Cisco NX-OS device.

Before you begin

Obtain the hostname for the remote device and, if needed, the username on the remote device.

Enable the SSH server on the remote device.

Procedure

	Command or Action	Purpose
Step 1	**ssh** [*username@*]{*ipv4-address* | *hostname*} [**vrf** *vrf-name*] **Example:** `switch# ssh 10.10.1.1`	Creates an SSH IPv4 session to a remote device using IPv4. The default VRF is the default VRF.

	Command or Action	Purpose
Step 2	**ssh6** [*username@*]{*ipv6-address* | *hostname*} [**vrf** *vrf-name*] **Example:** `switch# ssh6 HostA`	Creates an SSH IPv6 session to a remote device using IPv6.

Starting SSH Sessions from Boot Mode

You can start SSH sessions from the boot mode of the Cisco NX-OS device to connect to remote devices.

Before you begin

Obtain the hostname for the remote device and, if needed, the username on the remote device.

Enable the SSH server on the remote device.

Procedure

	Command or Action	Purpose
Step 1	**ssh** [*username@*]*hostname* **Example:** `switch(boot)# ssh user1@10.10.1.1`	Creates an SSH session to a remote device from the boot mode of the Cisco NX-OS device. The default VRF is always used.
Step 2	**exit** **Example:** `switch(boot)# exit`	Exits boot mode.
Step 3	**copy scp:**//[*username@*]*hostname*/*filepath directory* **Example:** `switch# copy scp://user1@10.10.1.1/users abc`	Copies a file from the Cisco NX-OS device to a remote device using the Secure Copy Protocol (SCP). The default VRF is always used.

Configuring SSH Passwordless File Copy

You can copy files from a Cisco NX-OS device to a secure copy (SCP) or secure FTP (SFTP) server without a password. To do so, you must create an RSA or DSA identity that consists of public and private keys for authentication with SSH.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enters global configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	
Step 2	[**no**] **username** *username* **keypair generate** {**rsa** [*bits* [**force**]] | **dsa** [**force**]} **Example:** `switch(config)# username user1 keypair generate rsa 2048 force`	Generates the SSH public and private keys and stores them in the home directory ($HOME/.ssh) of the Cisco NX-OS device for the specified user. The Cisco NX-OS device uses the keys to communicate with the SSH server on the remote machine. The *bits* argument is the number of bits used to generate the key. The range is from 768 to 4096. The default value is 1024. Use the **force** keyword to replace an existing key. The SSH keys are not generated if the **force** keyword is omitted and SSH keys are already present. **Note** Begining with Cisco NX-OS Release 10.6(1)F, dsa key word is hidden. If you enter the full command, NX-OS will accept it with a deprecation warning.
Step 3	(Optional) **show username** *username* **keypair** **Example:** `switch(config)# show username user1 keypair`	Displays the public key for the specified user. **Note** For security reasons, this command does not show the private key.
Step 4	Required: **username** *username* **keypair export** {**bootflash:***filename* | **volatile:***filename*} {**rsa** | **dsa**} [**force**] **Example:** `switch(config)# username user1 keypair export bootflash:key_rsa rsa`	Exports the public and private keys from the home directory of the Cisco NX-OS device to the specified bootflash or volatile directory. Use the **force** keyword to replace an existing key. The SSH keys are not exported if the **force** keyword is omitted and SSH keys are already present. To export the generated key pair, you are prompted to enter a passphrase that encrypts the private key. The private key is exported as the file that you specify, and the public key is exported with the same filename followed by a .pub extension. You can now copy this key pair to any Cisco NX-OS device and use SCP or SFTP to copy the public key file (*.pub) to the home directory of the server. **Note** For security reasons, this command can be executed only from global configuration mode.

	Command or Action	Purpose
		Note Begining with Cisco NX-OS Release 10.6(1)F, dsa key word is hidden. If you enter the full command, NX-OS will accept it with a deprecation warning.
Step 5	Required: **username** *username* **keypair import** {**bootflash:***filename* | **volatile:***filename*} {**rsa** | **dsa**} [**force**] **Example:** `switch(config)# username user1 keypair import bootflash:key_rsa rsa`	Imports the exported public and private keys from the specified bootflash or volatile directory to the home directory of the Cisco NX-OS device. **Note** Begining with Cisco NX-OS Release 10.6(1)F, dsa key word is hidden. If you enter the full command, NX-OS will accept it with a deprecation warning. Use the **force** keyword to replace an existing key. The SSH keys are not imported if the **force** keyword is omitted and SSH keys are already present. To import the generated key pair, you are prompted to enter a passphrase that decrypts the private key. The private key is imported as the file that you specify, and the public key is imported with the same filename followed by a .pub extension. **Note** For security reasons, this command can be executed only from global configuration mode. **Note** Only the users whose keys are configured on the server are able to access the server without a password.

What to do next

On the SCP or SFTP server, use the following command to append the public key stored in the *.pub file (for example, key_rsa.pub) to the authorized_keys file:

$ cat key_rsa.pub >> $HOME/.ssh/ authorized_keys

You can now copy files from the Cisco NX-OS device to the server without a password using standard SSH and SCP commands.

Configuring SCP and SFTP Servers

You can configure an SCP or SFTP server on the Cisco NX-OS device in order to copy files to and from a remote device. After you enable the SCP or SFTP server, you can execute an SCP or SFTP command on the remote device to copy the files to or from the Cisco NX-OS device.

Note The arcfour and blowfish cipher options are not supported for the SCP server.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	[**no**] **feature scp-server** **Example:** `switch(config)# feature scp-server`	Enables or disables the SCP server on the Cisco NX-OS device.
Step 3	Required: [**no**] **feature sftp-server** **Example:** `switch(config)# feature sftp-server`	Enables or disables the SFTP server on the Cisco NX-OS device.

Step 4	Required: **exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 5	(Optional) **show running-config security** **Example:** `switch# show running-config security`	Displays the configuration status of the SCP and SFTP servers.
Step 6	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring X.509v3 Certificate-Based SSH Authentication

You can configure SSH authentication using X.509v3 certificates.

Before you begin

Enable the SSH server on the remote device.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**username** *user-id* [**password** [**0** | **5**] *password*] **Example:** `switch(config)# username jsmith password 4Ty18Rnt`	Configures a user account. The *user-id* argument is a case-sensitive, alphanumeric character string with a maximum length of 28 characters for both local and remote users. Valid characters are uppercase letters A through Z, lowercase letters a through z, numbers 0 through 9, hyphen (-), period (.), underscore (_), plus sign (+), and equal sign (=). The at symbol (@) is supported in remote usernames but not in local usernames. Usernames must begin with an alphanumeric character. The default password is undefined. The **0** option indicates that the password is clear text, and the **5** option indicates that the password is encrypted. The default is **0** (clear text). **Note** If you do not specify a password, the user might not be able to log in to the Cisco NX-OS device. **Note** If you create a user account with the encrypted password option, the corresponding SNMP user will not be created. **Note** When the desynchronization CLI is enabled, if you create a user account, the corresponding SNMP user will not be created.
Step 3	**username** *user-id* **ssh-cert-dn** *dn-name* {**dsa** | **rsa**} **Example:** `switch(config)# username jsmith ssh-cert-dn "/O = ABCcompany, OU = ABC1, emailAddress = jsmith@ABCcompany.com, L = Metropolis, ST = New York, C = US, CN = jsmith" rsa`	Specifies an SSH X.509 certificate distinguished name and DSA or RSA algorithm to use for authentication for an existing user account. The distinguished name can be up to 512 characters and must follow the format shown in the examples. Make sure the email address and state are configured as emailAddress and ST, respectively.
Step 4	[**no**] **crypto ca trustpoint** *trustpoint*	Configures a trustpoint.

	Command or Action	Purpose

	Example:
	`switch(config)# crypto ca trustpoint winca`
	`switch(config-trustpoint)#`
		Note
		Before you delete a trustpoint using the **no** form of this command, you must first delete the CRL and CA certificate, using the **delete crl** and **delete ca-certificate** commands.

Step 5	**crypto ca authenticate** *trustpoint* **Example:** `switch(config-trustpoint)# crypto ca authenticate winca`	Configures a CA certificate for the trustpoint. **Note** To delete a CA certificate, enter the **delete ca-certificate** command in the trustpoint configuration mode.
Step 6	(Optional) **crypto ca crl request** *trustpoint* **bootflash:***static-crl***.crl** **Example:** `switch(config-trustpoint)# crypto ca crl request winca bootflash:crllist.crl`	This command is optional but highly recommended. Configures the certificate revocation list (CRL) for the trustpoint. The CRL file is a snapshot of the list of revoked certificates by the trustpoint. This static CRL list is manually copied to the device from the Certification Authority (CA). **Note** Static CRL is the only supported revocation check method. **Note** To delete the CRL, enter the **delete crl** command.
Step 7	(Optional) **show crypto ca certificates** **Example:** `switch(config-trustpoint)# show crypto ca certificates`	Displays the configured certificate chain and associated trustpoint.
Step 8	(Optional) **show crypto ca crl** *trustpoint* **Example:** `switch(config-trustpoint)# show crypto ca crl winca`	Displays the contents of the CRL list of the specified trustpoint.
Step 9	(Optional) **show user-account** **Example:** `switch(config-trustpoint)# show user-account`	Displays configured user account details.
Step 10	(Optional) **show users** **Example:** `switch(config-trustpoint)# show users`	Displays the users logged into the device.

	Command or Action	Purpose
Step 11	(Optional) **copy running-config startup-config** **Example:** `switch(config-trustpoint)# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Configuring SSH-Cert-Authorization on TACACS Servers

Beginning with Cisco NX-OS Release 10.4(3)F, the Cisco Nexus 9000 Series switches support SSH authorization using X.509 certificates through a TACACS+ server. This feature is not supported with RADIUS. This feature can be enabled using **aaa authorization ssh-certificate default group** *tac-group-name* command. For more information on configuration, see Configuring AAA SSH-Cert-Authorization on TACACS Servers.

Customizing SSH Cryptographic Algorithms

Cisco Nexus 9000 switches support strong algorithms by default. You can choose to remain with the default mode that enables only strong algorithms as defined by Cisco Product Security Baseline (PSB) or allow all supported algorithms. Note that these algorithms are applicable to the incoming server connections. You can also configure support for SSH key exchange algorithms, message authentication codes (MACs), key types, and ciphers.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters the global configuration mode.
Step 2	(Optional) **ssh kexalgos** [**all** | **key-exchangealgorithm-name**] **Example:** `switch(config)# ssh kexalgos ecdhsha2-nistp384`	Use the **all** keyword to enable all supported KexAlgorithms which are the key exchange methods that are used to generate per-connection keys. Supported KexAlgorithmns are: • curve25519-sha256 • curve25519-sha256@libssh.org • diffie-hellman-group14-sha1 • diffie-hellman-group14-sha256 • diffie-hellman-group16-sha512 • ecdh-sha2-nistp256 • ecdh-sha2-nistp521

	Command or Action	Purpose
		Unsupported KexAlgorithmns are: • diffie-hellman-group1-sha1 • diffie-hellman-group-exchange-sha256 To enable only the **ecdh-sha2-nistp384** KexAlgorithm, use the **ecdh-sha2-nistp384** keyword. **Note** Starting with Cisco NX-OS Release 10.4(2)F, you can configure any of the supported KexAlgorithmns. From this release, keyword **ecdh-sha2-nistp384** is deprecated.
Step 3	(Optional) **ssh macs [all | mac-name]** **Example:** `switch(config)# ssh macs hmacsha2-256-etm@openssh.com`	Enables all supported MACs which are the message authentication codes used to detect traffic modification. Supported MACs are: • hmac-sha1 • hmac-sha2-256 • hmac-sha2-512 • hmac-sha2-256-etm@openssh.com • hmac-sha2-512-etm@openssh.com • hmac-sha1-etm@openssh.com **Note** Starting with Cisco NX-OS Release 10.4(2)F, you can configure any of the supported MACs.
Step 4	(Optional) **ssh ciphers** [**all** | **cipher-name**] **Example:** `switch(config)# ssh ciphers aes192-ctr`	Use the **all** keyword to enable all supported ciphers to encrypt the connection. Supported ciphers are: • aes128-cbc • aes192-cbc • aes256-cbc • aes128-ctr • aes192-ctr • aes256-ctr • aes128-gcm@openssh.com

	Command or Action	Purpose
		• chacha20-poly1305@openssh.com **Note** Starting with Cisco NX-OS Release 10.4(2)F, you can configure any of the supported ciphers. From this release, keyword **aes256-gcm** is deprecated. **Note** Starting with Cisco NX-OS Release 10.6(1), the command `ssh ciphers aes256-gcm` is deprecated and no longer supported. If this command exists in your configuration, configuration replace (CR) and ISSU will fail. Remove the command from your configuration before upgrading.
Step 5	(Optional) **ssh keytypes [all | keytype-string]** **Example:** `switch(config)# ssh keytypes ecdsa-sha2-nistp256`	Enables all supported PubkeyAcceptedKeyTypes which are the public key algorithms that the server can use to authenticate itself to the client. Supported key types are: • ecdsa-sha2-nistp256 • ecdsa-sha2-nistp384 • ecdsa-sha2-nistp521 • ecdsa-sha2-nistp256-cert-v01@openssh.com • ecdsa-sha2-nistp384-cert-v01@openssh.com • ecdsa-sha2-nistp521-cert-v01@openssh.com • ssh-dss • rsa-sha2-256 • ssh-rsa-cert-v01@openssh.com • ssh-rsa **Note** Starting with Cisco NX-OS Release 10.4(2)F, you can configure any of the supported keytypes. **Note** To enable rsa, dsa, and ecdsa key types, you need to generate corresponding SSH host keys. **Example configuration:**

	Command or Action	Purpose
		switch(config)# ssh key rsa 2048 switch(config)# ssh key dsa switch(config)# ssh key ecdsa 256

Example

Users can check the supported algorithms using **show ssh [ciphers | macs | keytypes | kexalogs | version]** commands.

```
show ssh ciphers
Cipher                                          Status    FIPS
-------------------------------------------------------------
aes128-ctr                                   permitted     yes
aes192-ctr                                      denied     yes
aes256-ctr                                   permitted     yes
aes128-cbc                                      denied     yes
aes192-cbc                                      denied     yes
aes256-cbc                                      denied     yes
aes256-gcm@openssh.com                    permitted     yes
aes128-gcm@openssh.com                     permitted     yes
chacha20-poly1305@openssh.com              permitted      no
```

```
show ssh macMAC                                              Status    FIPS
-------------------------------------------------------------
hmac-sha2-256-etm@openssh.com              permitted      no
hmac-sha2-512-etm@openssh.com              permitted      no
hmac-sha1-etm@openssh.com                  permitted      no
hmac-sha2-256                              permitted     yes
hmac-sha2-512                              permitted     yes
hmac-sha1                                  permitted     yes
hmac-sha1-96                             unsupported      no
hmac-md5                                 unsupported      no
hmac-md5-96                              unsupported      no
umac-64@openssh.com                      unsupported      no
umac-128@openssh.com                     unsupported      no
hmac-sha1-96-etm@openssh.com             unsupported      no
hmac-md5-etm@openssh.com                 unsupported      no
umac-64-etm@openssh.com                  unsupported      no
umac-128-etm@openssh.com                 unsupported      no
```

```
show ssh keytypes Keytype                                       Status    FIPS
-------------------------------------------------------------
ecdsa-sha2-nistp256-cert-v01@openssh.com   permitted      no <<Currently not suppported>>
ecdsa-sha2-nistp384-cert-v01@openssh.com   permitted      no <<Currently not suppported>>
ecdsa-sha2-nistp521-cert-v01@openssh.com   permitted      no <<Currently not suppported>>
ssh-rsa-cert-v01@openssh.com               permitted      no
ecdsa-sha2-nistp256                        permitted     yes
ecdsa-sha2-nistp384                        permitted     yes
ecdsa-sha2-nistp521                        permitted      no
rsa-sha2-256                               permitted      no
ssh-rsa                                    permitted     yes
ssh-dss                                       denied      no
ssh-ed25519                              unsupported      no
ssh-ed25519-cert-v01@openssh.com         unsupported      no
```

```
ssh-dss-cert-v01@openssh.com             unsupported      no
```

```
show ssh kexalgos
KexAlgorithm                                      Status    FIPS
-------------------------------------------------------------
curve25519-sha256                               permitted      no
curve25519-sha256@libssh.org                    permitted      no
ecdh-sha2-nistp256                              permitted     yes
ecdh-sha2-nistp384                              permitted     yes
ecdh-sha2-nistp521                              permitted     yes
diffie-hellman-group16-sha512                   permitted     yes
diffie-hellman-group14-sha1                     permitted     yes
diffie-hellman-group14-sha256                   permitted      no
```

```
show ssh version
CiscoSSH 1.9.29, OpenSSH_8.3p1, CiscoSSL 1.1.1t.7.2.500
```

Algorithms Supported - FIPs Mode Enabled

The list of algorithms supported when the FIPs mode is enabled are as follows:

Table 2: Algorithms Supported - FIPs Mode Enabled

Algorithms	Supported	Unsupported
ciphers	• aes128-ctr • aes256-ctr • aes256-gcm@openssh.com • aes128-gcm@openssh.com • chacha20-poly1305@openssh.com	• aes192-ctr • aes128-cbc • aes192-cbc • aes256-cbc
hmac	• hmac-sha2-256 • hmac-sha2-512 • hmac-sha1 • hmac-sha2-256-etm@openssh.com • hmac-sha2-512-etm@openssh.com • hmac-sha1-etm@openssh.com	-

Algorithms	Supported	Unsupported
kexalgo	• ecdh-sha2-nistp256 • ecdh-sha2-nistp384 • ecdh-sha2-nistp521 • diffie-hellman-group16-sha512 • diffie-hellman-group14-sha1 • diffie-hellman-group14-sha256 • curve25519-sha256@libssh.org • curve25519-sha256	-
keytypes	• ecdsa-sha2-nistp256-cert-v01@openssh.com • ecdsa-sha2-nistp384-cert-v01@openssh.com • ecdsa-sha2-nistp521-cert-v01@openssh.com • ssh-rsa-cert-v01@openssh.com • rsa-sha2-256 • ssh-rsa • ecdsa-sha2-nistp256 • ecdsa-sha2-nistp384 • ecdsa-sha2-nistp521	• ssh-dss

Changing the Default SSH Server Port

Beginning with Cisco NX-OS Cisco Release 9.2(1), you can change the SSHv2 port number from the default port number 22. Encryptions used while changing the default SSH port provides you with connections that support stronger privacy and session integrity

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**no feature ssh** **Example:** `switch(config)# no feature ssh`	Disables SSH.

	Command or Action	Purpose

Step 3	**show sockets** *local-port-range*
	Example:
	`switch(config)# show sockets local port range (15001 - 58000)`
	`switch(config)# local port range (58001 - 63535) and nat port range (63536 - 65535)`
	`switch# show sockets local-port-range`
	`Kstack local port range (15001 - 22002)`
	`Netstack local port range (22003 – 65535)`
		Displays the available port range.

Step 4	**ssh port** *local-port* **Example:** `switch(config)# ssh port 58003`	Configures the port. **Note** When you upgrade from an earlier release to Release 9.3(1) or later releases, ensure that features with user-defined SSH port, are within the following range: • For Release 9.3(1) and Release 9.3(2): Kstack local port range is from 15001 to 58000, netstack local port range is from 58001 - 63535, and nat port range is from 63536 to 65535 • From Release 9.3(3): Kstack local port range is from 15001 to 58000, netstack local port range is from 58001 to 60535, and nat port range is from 60536 to 65535
Step 5	**feature ssh** **Example:** `switch(config)# feature ssh`	Enables SSH.

Step 6	**exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 7	(Optional) **show running-config security all** **Example:** `switch# ssh port 58003`	Displays the security configuration.
Step 8	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Clearing SSH Hosts

When you download a file from a server using SCP or SFTP, or when you start an SSH session from this device to a remote host, you establish a trusted SSH relationship with that server. You can clear the list of trusted SSH servers for your user account.

Procedure

	Command or Action	Purpose
Step 1	**clear ssh hosts** **Example:** `switch# clear ssh hosts`	Clears the SSH host sessions and the known host file.

Disabling the SSH Server

By default, the SSH server is enabled on the Cisco NX-OS device. You can disable the SSH server to prevent SSH access to the switch.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**no feature ssh** **Example:** `switch(config)# no feature ssh`	Disables SSH.

Step 3	**exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 4	(Optional) **show ssh server** **Example:** `switch# show ssh server`	Displays the SSH server configuration.
Step 5	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Deleting SSH Server Keys

You can delete SSH server keys on the Cisco NX-OS device after you disable the SSH server.

Note To reenable SSH, you must first generate an SSH server key.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**no feature ssh** **Example:** `switch(config)# no feature ssh`	Disables SSH.
Step 3	**no ssh key** [**dsa** | **rsa** | **ecdsa**] **Example:** `switch(config)# no ssh key rsa`	Deletes the SSH server key. The default is to delete all the SSH keys.

Step 4	**exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 5	(Optional) **show ssh key** **Example:** `switch# show ssh key`	Displays the SSH server key configuration.
Step 6	(Optional) **copy running-config startup-config** **Example:** `switch# copy running-config startup-config`	Copies the running configuration to the startup configuration.

Related Topics

Generating SSH Server Keys, on page 11

Clearing SSH Sessions

You can clear SSH sessions from the Cisco NX-OS device.

Procedure

	Command or Action	Purpose
Step 1	**show users** **Example:** `switch# show users`	Displays user session information.
Step 2	**clear line** *vty-line* **Example:** `switch(config)# clear line pts/12`	Clears a user SSH session.

Configuring Telnet

This section describes how to configure Telnet on the Cisco NX-OS device.

Enabling the Telnet Server

You can enable the Telnet server on the Cisco NX-OS device. By default, the Telnet server is disabled.

Procedure

	Command or Action	Purpose

Step 1	**configure terminal**
	Example:
	`switch# configure terminal`
	`switch(config)#`
		Enters global configuration mode.

Step 2	**feature telnet** **Example:** `switch(config)# feature telnet`	Enables the Telnet server. The default is disabled.

Step 3	**exit**
	Example:
	`switch(config)# exit`
	`switch#`
		Exits global configuration mode.

Step 4	(Optional) **show telnet server** **Example:** `switch# show telnet server`	Displays the Telnet server configuration.
Step 5	(Optional) **copy running-config startup-config** **Example:**	Copies the running configuration to the startup configuration.

	Command or Action	Purpose
	`switch# copy running-config startup-config`	

Starting Telnet Sessions to Remote Devices

You can start Telnet sessions to connect to remote devices from the Cisco NX-OS device. You can start Telnet sessions using either IPv4 or IPv6.

Before you begin

Obtain the hostname or IP address for the remote device and, if needed, the username on the remote device.

Enable the Telnet server on the Cisco NX-OS device.

Enable the Telnet server on the remote device.

Procedure

	Command or Action	Purpose
Step 1	**telnet** {*ipv4-address* | *host-name*} [*port-number*] [**vrf** *vrf-name*] **Example:** `switch# telnet 10.10.1.1`	Starts a Telnet session to a remote device using IPv4. The default port number is 23. The range is from 1 to 65535. The default VRF is the default VRF.
Step 2	**telnet6** {*ipv6-address* | *host-name*} [*port-number*] [**vrf** *vrf-name*] **Example:** `switch# telnet6 2001:0DB8::ABCD:1 vrf management`	Starts a Telnet session to a remote device using IPv6. The default port number is 23. The range is from 1 to 65535. The default VRF is the default VRF.

Related Topics

Enabling the Telnet Server, on page 31

Clearing Telnet Sessions

You can clear Telnet sessions from the Cisco NX-OS device.

Before you begin

Enable the Telnet server on the Cisco NX-OS device.

Procedure

	Command or Action	Purpose
Step 1	**show users** **Example:**	Displays user session information.

	Command or Action	Purpose
	`switch# show users`	
Step 2	**clear line** *vty-line* **Example:** `switch(config)# clear line pts/12`	Clears a user Telnet session.

Verifying the SSH and Telnet Configuration

To display the SSH and Telnet configuration information, perform one of the following tasks:

Command	Purpose
show ssh key [**dsa** | **rsa**] [**md5**]	Displays the SSH server keys. For Cisco NX-OS Release 7.0(3)I4(6) and 7.0(3)I6(1) and any later releases, this command displays the fingerprint in SHA256 format by default. SHA256 is more secure than the old default format of MD5. However, the **md5** option has been added, if you want to see the fingerprint in MD5 format for backward compatibility.
show running-config security [**all**]	Displays the SSH and user account configuration in the running configuration. The **all** keyword displays the default values for the SSH and user accounts.
show ssh server	Displays the SSH server configuration.
show telnet server	Displays the Telnet server configuration.
show username *username* **keypair**	Displays the public key for the specified user.
show user-account	Displays configured user account details.
show users	Displays the users logged into the device.
show crypto ca certificates	Displays the configured CA certificate and associated trustpoint for X.509v3 certificate-based SSH authentication.
show crypto ca crl *trustpoint*	Displays the contents of the CRL list of the specified trustpoint for X.509v3 certificate-based SSH authentication.

Configuration Example for SSH

The following example shows how to configure SSH with an OpenSSH key:

Procedure

Step 1 Disable the SSH server.

Example:

```
switch# configure terminal
switch(config)# no feature ssh
```

Step 2 Generate an SSH server key.

Example:

```
switch(config)# ssh key rsa
generating rsa key(1024 bits)......
generated rsa key
```

Step 3 Enable the SSH server.

Example:

```
switch(config)# feature ssh
```

Step 4 Display the SSH server key.

Example:

```
switch(config)# show ssh key
could not retrieve dsa key information
************************************
rsa Keys generated:Tue Mar 14 13:13:47 2017

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQDh4+DZboQJbJt10nJhgKBYL5lOlhsFM2oZRi9+JqEU
GA44I9ej+E5NIRZ1x8ohIt6Vx9Et5csO7Pw72rjUwR3UPmuAm79k7I/SyLGEP3WUL7sqbLvNF5GqKXph
oqMT075WUdbGWphorA2g0tTObRrFIQBJVQ0SSBh3oEaaALqYUQ==

bitcount:1024
fingerprint:
SHA256:V6KAeLAiKRRUPBZm1Yq3rl6JW7Eo7vhLi6CXYxnD/+Y
************************************
************************************


switch(config)# show ssh key
rsa Keys generated:Sat Sep 29 00:10:39 2013

ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAvWhEBsF55oaPHNDBnpXOTw6+/OdHoLJZKr
+MZm99n2U0ChzZG4svRWmHuJY4PeDWl0e5yE3g3EO3pjDDmt923siNiv5aSga60K36lr39
HmXL6VgpRVn1XQFiBwn4na+H1d3Q0hDt+uWEA0tka2uOtXlDhliEmn4HVXOjGhFhoNE=

bitcount:1024
fingerprint:
51:6d:de:1c:c3:29:50:88:df:cc:95:f0:15:5d:9a:df
************************************
could not retrieve dsa key information
************************************
```

Step 5 Specify the SSH public key in OpenSSH format.

Example:

```
switch(config)# username User1 sshkey ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAy19oF6QaZ19G+3f1XswK3OiW4H7YyUyuA50r
v7gsEPjhOBYmsi6PAVKui1nIf/DQhum+lJNqJP/eLowb7ubO+lVKRXFY/G+lJNIQ
W3g9igG30c6k6+XVn+NjnI1B7ihvpVh7dLddMOXwOnXHYshXmSiH3UD/vKyziEh5
4Tplx8=
```

Step 6 Save the configuration.

Example:

```
switch(config)# copy running-config startup-config
```

Configuration Example for SSH Passwordless File Copy

The following example shows how to copy files from a Cisco NX-OS device to a secure copy (SCP) or secure FTP (SFTP) server without a password:

Procedure

Step 1 Generate the SSH public and private keys and store them in the home directory of the Cisco NX-OS device for the specified user.

Example:

```
switch# configure terminal
switch(config)# username admin keypair generate rsa
generating rsa key(1024 bits)......
generated rsa key
```

Step 2 Display the public key for the specified user.

Example:

```
switch(config)# show username admin keypair

**************************************

rsa Keys generated: Thu Jul  9 11:10:29 2013

ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAxWmjJT+oQhIcvnrMbx2BmD0P8boZElTfJ
Fx9fexWp6rOiztlwODtehnjadWc6A+DE2DvYNvqsrU9TBypYDPQkR/+Y6cKubyFW
VxSBG/NHztQc3+QC1zdkIxGNJbEHyFoajzNEO8LLOVFIMCZ2Td7gxUGRZc+fbq
S33GZsCAX6v0=

bitcount:262144
fingerprint:
8d:44:ee:6c:ca:0b:44:95:36:d0:7d:f2:b5:78:74:7d
**************************************

could not retrieve dsa key information
**************************************
```

Step 3 Export the public and private keys from the home directory of the Cisco NX-OS device to the specified bootflash directory.

Example:

```
switch(config)# username admin keypair export bootflash:key_rsa rsa
Enter Passphrase:
switch(config)# dir
.
.
.
        951     Jul 09 11:13:59 2013  key_rsa
        221     Jul 09 11:14:00 2013  key_rsa.pub
.
.
```

Step 4 After copying these two files to another Cisco NX-OS device using the **copy scp** or **copy sftp** command, import them to the home directory of the Cisco NX-OS device.

Example:

```
switch(config)# username admin keypair import bootflash:key_rsa rsa
Enter Passphrase:
switch(config)# show username admin keypair
**************************************

rsa Keys generated: Thu Jul  9 11:10:29 2013

ssh-rsa
AAAAB3NzaC1yc2EAAAABIwAAAIEAxWmjJT+oQhIcvnrMbx2BmD0P8boZElTfJ
Fx9fexWp6rOiztlwODtehnjadWc6A+DE2DvYNvqsrU9TBypYDPQkR/+Y6cKubyFW
VxSBG/NHztQc3+QC1zdkIxGNJbEHyFoajzNEO8LLOVFIMCZ2Td7gxUGRZc+fbq
S33GZsCAX6v0=

bitcount:262144
fingerprint:
8d:44:ee:6c:ca:0b:44:95:36:d0:7d:f2:b5:78:74:7d
**************************************

could not retrieve dsa key information
**************************************
switch(config)#
```

Step 5 On the SCP or SFTP server, append the public key stored in key_rsa.pub to the authorized_keys file.

Example:

```
$ cat key_rsa.pub >> $HOME/.ssh/ authorized_keys
```

You can now copy files from the Cisco NX-OS device to the server without a password using standard SSH and SCP commands.

Step 6 (Optional) Repeat this procedure for the DSA keys.

Configuration Example for X.509v3 Certificate-Based SSH Authentication

The following example shows how to configure SSH authentication using X.509v3 certificates:

Note Beginning with Cisco NX-OS Release 10.4(3)F, the Cisco Nexus 9000 Series switches support SSH authorization using X.509 certificates through a TACACS+ server. This feature is not supported with RADIUS.

```
configure terminal
username jsmith password 4Ty18Rnt
username jsmith ssh-cert-dn "/O = ABCcompany, OU = ABC1,
emailAddress = jsmith@ABCcompany.com, L = Metropolis, ST = New York, C = US, CN = jsmith"
rsa
crypto ca trustpoint tp1
crypto ca authenticate tp1
crypto ca crl request tp1 bootflash:crl1.crl

show crypto ca certificates
Trustpoint: tp1
CA certificate 0:
subject= /CN=SecDevCA
issuer= /CN=SecDevCA
serial=01AB02CD03EF04GH05IJ06KL07MN
notBefore=Jun 29 12:36:26 2016 GMT
notAfter=Jun 29 12:46:23 2021 GMT
SHA1 Fingerprint=47:29:E3:00:C1:C1:47:F2:56:8B:AC:B2:1C:64:48:FC:F4:8D:53:AF
purposes: sslserver sslclient

show crypto ca crl tp1
Trustpoint: tp1 CRL: Certificate Revocation List (CRL):
    Version 2 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
    Issuer: /CN=SecDevCA
    Last Update: Aug 8 20:03:15 2016 GMT
    Next Update: Aug 16 08:23:15 2016 GMT
    CRL extensions:
        X509v3 Authority Key Identifier:
            keyid:30:43:AA:80:10:FE:72:00:DE:2F:A2:17:E4:61:61:44:CE:78:FF:2A

show user-account
user:user1
        this user account has no expiry date
        roles:network-operator
        ssh cert DN : /C = US, ST = New York, L = Metropolis, O = cisco , OU = csg, CN =
user1; Algo: x509v3-sign-rsa

show users
NAME      LINE        TIME          IDLE      PID         COMMENT
user1     pts/1       Jul 27 18:43  00:03     18796      (10.10.10.1)   session=ssh
```

Additional References for SSH and Telnet

This section describes additional information related to implementing SSH and Telnet.

Related Documents

Related Topic	Document Title
Cisco NX-OS licensing	*Cisco NX-OS Licensing Guide*
VRF configuration	*Cisco Nexus 9000 Series NX-OS Unicast Routing Configuration Guide*

RFCs

RFCs	Title
RFC 6187	*X.509v3 Certificates for Secure Shell Authentication*

MIBs

MIBs	MIBs Link
MIBs related to SSH and Telnet	To locate and download supported MIBs, go to the following URL: https://cisco.github.io/cisco-mibs/supportlists/nexus9000/Nexus9000MIBSupportList.html