



# NETCONF Protocol

---

- [Information About the NETCONF Protocol, on page 1](#)
- [How to Configure the NETCONF Protocol, on page 8](#)
- [Verifying the NETCONF Protocol Configuration, on page 13](#)
- [Additional References for NETCONF Protocol, on page 15](#)
- [Feature Information for NETCONF Protocol, on page 16](#)

## Information About the NETCONF Protocol

### Introduction to Data Models - Programmatic and Standards-Based Configuration

The traditional way of managing network devices is by using Command Line Interfaces (CLIs) for configurational (configuration commands) and operational data (show commands). For network management, Simple Network Management Protocol (SNMP) is widely used, especially for exchanging management information between various network devices. Although CLIs and SNMP are heavily used, they have several restrictions. CLIs are highly proprietary, and human intervention is required to understand and interpret their text-based specification. SNMP does not distinguish between configurational and operational data.

The solution lies in adopting a programmatic and standards-based way of writing configurations to any network device, replacing the process of manual configuration. Network devices running on Cisco IOS XE support the automation of configuration for multiple devices across the network using data models. Data models are developed in a standard, industry-defined language, that can define configuration and state information of a network.

Cisco IOS XE supports the Yet Another Next Generation (YANG) data modeling language. YANG can be used with the Network Configuration Protocol (NETCONF) to provide the desired solution of automated and programmable network operations. NETCONF (RFC 6241) is an XML-based protocol that client applications use to request information from and make configuration changes to the device. YANG is primarily used to model the configuration and state data used by NETCONF operations.

In Cisco IOS XE, model-based interfaces interoperate with existing device CLI, Syslog, and SNMP interfaces. These interfaces are optionally exposed northbound from network devices. YANG is used to model each protocol based on RFC 6020.



---

**Note** To access Cisco YANG models in a developer-friendly way, clone the [GitHub repository](#), and navigate to the [vendor/cisco](#) subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

---

## NETCONF

NETCONF provides a mechanism 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.

NETCONF uses a simple Remote Procedure Call (RPC) based mechanism to facilitate communication between a client and a server. The client can be a script or application running as part of a network manager. The server is typically a network device (switch or router). It uses Secure Shell (SSH) as the transport layer across network devices. It uses SSH port number 830 as the default port. The port number is a configurable option.

NETCONF also supports capability discovery and model downloads. Supported models are discovered using the *ietf-netconf-monitoring* model. Revision dates for each model are shown in the capabilities response. Data models are available for optional download from a device using the *get-schema* RPC. You can use these YANG models to understand or export the data model. For more details on NETCONF, see *RFC 6241*.

In releases prior to Cisco IOS XE Fuji 16.8.1, an operational data manager (based on polling) was enabled separately. In Cisco IOS XE Fuji 16.8.1 and later releases, operational data works on platforms running NETCONF (similar to how configuration data works), and is enabled by default. For more information on the components that are enabled for operational data queries or streaming, see the [GitHub](#) repository, to view *\*-oper* in the naming convention.

## NETCONF RESTCONF IPv6 Support

Data model interfaces (DMIs) support the use of IPv6 protocol. DMI IPv6 support helps client applications to communicate with services that use IPv6 addresses. External facing interfaces will provide dual-stack support; both IPv4 and IPv6.

DMIs are a set of services that facilitate the management of network elements. Application layer protocols such as, NETCONF and RESTCONF access these DMIs over a network.

If IPv6 addresses are not configured, external-facing applications will continue to listen on IPv6 sockets; but these sockets will be unreachable.

## NETCONF Global Session Lock

The NETCONF protocol provides a set of operations to manage device configurations and retrieve device state information. NETCONF supports a global lock, and the ability to kill non-responsive sessions are introduced in NETCONF.

To ensure consistency and prevent conflicting configurations through multiple simultaneous sessions, the owner of the session can lock the NETCONF session. The NETCONF lock RPC locks the configuration parser and the running configuration database. All other NETCONF sessions (that do not own the lock) cannot perform edit operations; but can perform read operations. These locks are intended to be short-lived and allow

the owner to make changes without interaction with other NETCONF clients, non-NETCONF clients (such as, SNMP and CLI scripts), and human users.

A global lock held by an active session is revoked when the associated session is killed. The lock gives the session holding the lock exclusive write access to the configuration. When a configuration change is denied due to a global lock, the error message will specify that a NETCONF global lock is the reason the configuration change has been denied.

The `<lock>` operation takes a mandatory parameter, `<target>` that is the name of the configuration datastore that is to be locked. When a lock is active, the `<edit-config>` and `<copy-config>` operations are not allowed.

If the **clear configuration lock** command is specified while a NETCONF global lock is being held, a full synchronization of the configuration is scheduled and a warning syslog message is produced. This command clears only the parser configuration lock.

The following is a sample RPC that shows the `<lock>` operation:

```
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <lock>
    <target>
      <running/>
    </target>
  </lock>
</rpc>
```

## NETCONF Kill Session

During a session conflict or client misuse of the global lock, NETCONF sessions can be monitored via the **show netconf-yang sessions** command, and non-responsive sessions can be cleared using the **clear netconf-yang session** command. The **clear netconf-yang session** command clears both the NETCONF lock and the configuration lock.

A `<kill-session>` request will force a NETCONF session to terminate. When a NETCONF entity receives a `<kill-session>` request for an open session, it stops all operations in process, releases all locks and resources associated with the session, and closes any associated connections.

A `<kill-session>` request requires the session-ID of the NETCONF session that is to be terminated. If the value of the session-ID is equal to the current session ID, an invalid-value error is returned. If a NETCONF session is terminated while its transaction is still in progress, the data model infrastructure will request a rollback, apply it to the network element, and trigger a synchronization of all YANG models.

If a session kill fails, and a global lock is held, enter the **clear configuration lock** command via the console or vty. At this point, the data models can be stopped and restarted.

## NETCONF-YANG SSH Server Support

NETCONF-YANG uses the IOS Secure Shell (SSH) Rivest, Shamir, and Adleman (RSA) public keys to authenticate users as an alternative to password-based authentication.

For public-key authentication to work on NETCONF-YANG, the IOS SSH server must be configured. To authenticate users to the SSH server, use one of the RSA keys configured by using the **ip ssh pubkey-chain** and **user** commands.

NACM is a group-based access control mechanism. When users are authenticated, they are automatically placed in an NACM privilege group based on their configured privilege level. Users can also be manually placed in other user-defined groups. The default privilege level is 1. There are 16 privilege levels, PRIV00 to PRIV15.

If a user authenticates via the public-key; but does not have a corresponding Authentication, Authorization, and Accounting (AAA) configuration, this user is rejected. If a user authenticates via a public-key; but the AAA configuration for NETCONF is using a AAA source other than the local, this user is also rejected. Local and TACACS+ AAA authorization are supported.

Token-based RESTCONF authentication is not supported. SSH user certificates are not supported.

## Candidate Configuration Support

The Candidate Configuration feature enables support for candidate capability by implementing RFC 6241 with a simple commit option.

The candidate datastore provides a temporary work space in which a copy of the device's running configuration is stored. You can create and modify the running configuration before committing the running configuration to the device. Candidate capability is indicated by the following NETCONF capability: urn:ietf:params:netconf:capability:candidate:1.0. This NETCONF capability indicates that the device supports the candidate datastore.

This is a shared data store which enables the user to create, add, delete and make changes to the device configuration without affecting the running configuration on the device. A commit operation pushes the configuration from the candidate to the running configuration on the device. When the candidate data store is enabled, the running data store is *not* writable through NETCONF sessions, and all configurations get committed only through the candidate. In other words, the writable-running NETCONF capability is not enabled with the candidate configuration.




---

**Note** It must be kept in mind that candidate datastore is a shared data store. Multiple NETCONF sessions can modify its contents simultaneously. Therefore, it is important to lock the datastore before modifying its contents, to prevent conflicting commits that can eventually lead to the loss of any configuration changes.

---

## NETCONF Operations on Candidate

The following operations can be performed on the candidate data store.




---

**Note** The information in this section has been referenced from section 8.3.4 of RFC 6241. Please refer to the RFC for more details and the exact RPCs.

---

### Lock

A <lock> RPC is used to lock the target data store. This prevents other users from modifying the configuration in the locked data store. Both candidate and running data can be locked through the lock operation.



---

**Note** Locking the candidate datastore does not affect the Cisco IOS config lock or the running configuration lock and vice versa.

---

### Commit

A <commit> RPC, copies the candidate configuration to the device's running configuration. A *commit* operation must be performed after you have updated the candidate configuration to push the configuration to the device.

If either the running or the candidate datastore is locked by another NETCONF session, the <commit> RPC will fail with an RPC error reply. The <error-tag> should be <in-use> and <error-info> should have the session ID of the NETCONF session holding the lock. You can also lock the running configuration by using the global lock by entering the `conf t lock` mode, but, the commit operation will fail with an RPC error reply, with error-tag value <in-use> and the session-id will be "0".

### Edit-config

The candidate configuration can be used as a target for the edit-config operation to modify a configuration. You can change the candidate configuration without affecting the running configuration on the device.

### Discard

To remove the changes made to the candidate configuration, perform a discard operation to revert the candidate configuration to running configuration.

If contents of the candidate datastore are modified by NETCONF session A, and session B tries to lock the candidate datastore, the lock fails. NETCONF session B must perform a <discard> operation to remove any outstanding configuration changes on the candidate datastore from other NETCONF sessions before locking a candidate.

### Unlock

After working on candidate configuration, such as, lock, edit-config, or commit operations, you can unlock the datastore, by specifying candidate as target in the unlock RPC. The candidate datastore is now available for all operations in other sessions.

If a failure occurs with outstanding changes to the candidate datastore, it can be challenging to recover the configuration, and may create problems for other sessions. To avoid any issues, outstanding changes must be discarded when the lock is released—either implicitly on "NETCONF session failure" or explicitly by using the unlock operation.

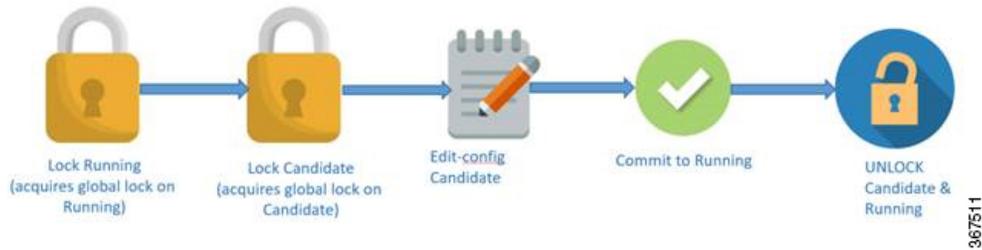
### Get-config, Copy-config, Validate

The candidate datastore can be used as a source or target for any of the get-config, copy-config or validate config operations. If you do not want to commit the changes in the candidate datastore to the device, but only to validate the configuration, you can use the <validate> RPC followed by a discard operation.

### Modifying the Candidate Datastore

The following diagram explains the recommended best practice when modifying the device configuration through candidate datastore:

Figure 1: Modifying Candidate Datastore Steps



1. Lock the running datastore.
2. Lock the candidate datastore.
3. Make modifications to the candidate configuration through edit-config RPCs with the target candidate.
4. Commit the candidate configuration to the running configuration.
5. Unlock the candidate and running datastores.

## Confirmed Candidate Configuration Commit

The candidate configuration supports the confirmed commit capability. This implementation is as specified in RFC 6241 for the confirmed commit capability which, when issued, sets the running configuration to the current contents of the candidate configuration and starts a confirmed commit timer. The confirmed commit operation will be rolled back if the commit is not issued within the timeout period. The default timeout period is 600 seconds or 10 minutes.

When you commit the candidate configuration, you can require an explicit confirmation for the commit to become permanent. The confirmed commit operation is useful for verifying that a configuration change works correctly and does not prevent management access to the device. If the change prevents access or causes other errors, the automatic rollback to the previous configuration restores access after the rollback deadline passes. If the commit is not confirmed within the specified amount of time, by default, the device automatically retrieves and commits (rolls back to) the previously committed configuration.



**Note** RESTCONF does not support confirmed commit.

In a NETCONF session, to commit the candidate configuration and to explicitly confirm the commit to become permanent, a client application encloses the empty `<confirmed/>` tag in the `<commit>` and `<rpc>` tag elements:

```
<rpc>
  <commit>
    <confirmed/>
  </commit>
</rpc>
```

The following sample RPC shows how to change the default rollback timer:

```
<rpc>
  <commit>
    <confirmed/>
    <confirm-timeout>nnn</confirm-timeout> !nnn is the rollback-delay in seconds.
```

```

    </commit>
</rpc>

```

The following sample RPC shows that the NETCONF server confirms that the candidate configuration is committed temporarily:

```

<rpc-reply xmlns="URN" xmlns:junos="URL">
  <ok/>
</rpc-reply>

```

If the NETCONF server cannot commit the candidate configuration, the `<rpc-reply>` element will enclose an `<rpc-error>` element explaining the reason for the failure. The most common causes are semantic or syntactic errors in the candidate configuration.

To delay the rollback to a time later than the current rollback timer, the client application sends a `<confirmed/>` tag inside a `<commit>` tag element again before the deadline passes. Optionally, it includes the `<confirm-timeout>` element to specify how long to delay the next rollback. The client application can delay the rollback indefinitely by sending the `<confirmed/>` tag repeatedly.

To commit the configuration permanently, the client application sends the `<commit/>` tag enclosed in an `<rpc>` tag element before the rollback deadline passes. The rollback is canceled and the candidate configuration is committed immediately. If the candidate configuration is the same as the temporarily committed configuration, the temporarily committed configuration is recommitted.

If another application uses the `<kill-session/>` tag element to terminate this application's session while a confirmed commit is pending (this application has committed changes but not yet confirmed them), the NETCONF server that is using this session restores the configuration to its state before the confirmed commit instruction was issued.

The candidate datastore is disabled by using the **no netconf-yang feature candidate-datastore** command. Because the candidate datastore confirmed commit is enabled when the candidate datastore is enabled, the confirmed commit is disabled when the candidate datastore is disabled. All sessions in progress are terminated, and the confd program is restarted.

## Candidate Support Configuration

The candidate datastore functionality can be enabled by using the **netconf-yang feature candidate-datastore** command. When the datastore state changes from running to candidate or back, a warning message is displayed, notifying the user that a restart of NETCONF or RESTCONF will occur in order for the change to take effect.

If the selection of the candidate or running datastore is specified in the configuration when a NETCONF-YANG or RESTCONF confd process starts, a warning message appears as shown below:

```

Device(config)# netconf-yang feature candidate-datastore

netconf-yang initialization in progress - datastore transition not allowed, please try again
after 30 seconds

```

If the selection of the candidate or running datastore is made after the NETCONF-YANG or RESTCONF confd process starts, the following apply:

- If the **netconf-yang feature candidate-datastore** command is configured, the command enables the candidate datastore and prints the following warning:

```

"netconf-yang and/or restconf is transitioning from running to candidate netconf-yang
and/or restconf will now be restarted,
and any sessions in progress will be terminated".

```

- If the **netconf-yang feature candidate-datastore** command is removed, the command disables the candidate datastore, enables the running datastore and prints the following warning:

```
netconf-yang and/or restconf is transitioning from candidate to running netconf-yang
and/or restconf will now be restarted,
and any sessions in progress will be terminated".
```

- When NETCONF-YANG or RESTCONF are restarted, sessions in progress will be lost.

## How to Configure the NETCONF Protocol

NETCONF-YANG uses the primary trustpoint of a device. If a trustpoint does not exist, when NETCONF-YANG is configured, it creates a self-signed trustpoint. For more information, see the [Public Key Infrastructure Configuration Guide, Cisco IOS XE Gibraltar 16.10.x](#).

### Providing Privilege Access to Use NETCONF

To start working with NETCONF APIs, you must be a user with privilege level 15.

#### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **username name privilege level password password**
4. **aaa new-model**
5. **aaa authentication login default local**
6. **aaa authorization exec default local**
7. **end**

#### DETAILED STEPS

	Command or Action	Purpose
<b>Step 1</b>	<b>enable</b> <b>Example:</b> Device# enable	Enables privileged EXEC mode. Enter your password if prompted.
<b>Step 2</b>	<b>configure terminal</b> <b>Example:</b> Device# configure terminal	Enters global configuration mode.
<b>Step 3</b>	<b>username name privilege level password password</b> <b>Example:</b> Device(config)# username example-name privilege 15 password example_password	Establishes a user name-based authentication system. Configure the following keywords: <ul style="list-style-type: none"> <li>• <b>privilege level</b>: Sets the privilege level for the user. For the NETCONF protocol, it must be 15.</li> <li>• <b>password password</b>: Sets a password to access the CLI view.</li> </ul>

	Command or Action	Purpose
Step 4	<b>aaa new-model</b> <b>Example:</b> Device(config)# aaa new-model	(Optional) Enables authorisation, authentication, and accounting (AAA).  If the <b>aaa new-model</b> command is configured, AAA authentication and authorization is required.
Step 5	<b>aaa authentication login default local</b> <b>Example:</b> Device(config)# aaa authentication login default local	Sets the login authentication to use the local username database.  <b>Note</b> Only the default AAA authentication login method is supported for the NETCONF protocol.  <ul style="list-style-type: none"> <li>• For a remote AAA server, replace <i>local</i> with your AAA server.</li> </ul> The <b>default</b> keyword applies the local user database authentication to all ports.
Step 6	<b>aaa authorization exec default local</b> <b>Example:</b> Device(config)# aaa authorization exec default local	Configures user AAA authorization, check the local database, and allows the user to run an EXEC shell.  <b>Note</b> Only the default AAA authorization method is supported for the NETCONF protocol.  <ul style="list-style-type: none"> <li>• For a remote AAA server, replace <i>local</i> with your AAA server.</li> <li>• The <b>default</b> keyword applies the local user database authentication to all ports.</li> </ul>
Step 7	<b>end</b> <b>Example:</b> Device(config)# end	Exits global configuration mode and returns to privileged EXEC mode.

## Configuring NETCONF-YANG

If the legacy NETCONF protocol is enabled on your device, the RFC-compliant NETCONF protocol does not work. Disable the legacy NETCONF protocol by using the **no netconf legacy** command.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **netconf-yang**
4. **netconf-yang feature candidate-datastore**
5. **exit**

## DETAILED STEPS

	Command or Action	Purpose
<b>Step 1</b>	<b>enable</b> <b>Example:</b> Device> <b>enable</b>	Enables privileged EXEC mode. <ul style="list-style-type: none"><li>• Enter your password if prompted.</li></ul>
<b>Step 2</b>	<b>configure terminal</b> <b>Example:</b> Device# <b>configure terminal</b>	Enters global configuration mode.
<b>Step 3</b>	<b>netconf-yang</b> <b>Example:</b> Device (config)# <b>netconf-yang</b>	Enables the NETCONF interface on your network device. <b>Note</b> After the initial enablement through the CLI, network devices can be managed subsequently through a model based interface. The complete activation of model-based interface processes may require up to 90 seconds.
<b>Step 4</b>	<b>netconf-yang feature candidate-datastore</b> <b>Example:</b> Device(config)# <b>netconf-yang feature candidate-datastore</b>	Enables candidate datastore.
<b>Step 5</b>	<b>exit</b> <b>Example:</b> Device (config)# <b>exit</b>	Exits global configuration mode.

## Configuring NETCONF Options

### Configuring SNMP

Enable the SNMP Server in IOS to enable NETCONF to access SNMP MIB data using YANG models generated from supported MIBs, and to enable supported SNMP traps in IOS to receive NETCONF notifications from the supported traps.

Perform the following steps:

#### SUMMARY STEPS

1. Enable SNMP features in IOS.
2. After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.
3. Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

## DETAILED STEPS

**Step 1** Enable SNMP features in IOS.

**Example:**

```
configure terminal
logging history debugging
logging snmp-trap emergencies
logging snmp-trap alerts
logging snmp-trap critical
logging snmp-trap errors
logging snmp-trap warnings
logging snmp-trap notifications
logging snmp-trap informational
logging snmp-trap debugging
!
snmp-server community public RW
snmp-server trap link ietf
snmp-server enable traps snmp authentication linkdown linkup
snmp-server enable traps syslog
snmp-server manager
exit
```

**Step 2** After NETCONF-YANG starts, enable SNMP Trap support by sending the following RPC <edit-config> message to the NETCONF-YANG port.

**Example:**

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <netconf-yang xmlns="http://cisco.com/yang/cisco-self-mgmt">
        <cisco-ia xmlns="http://cisco.com/yang/cisco-ia">
          <snmp-trap-control>
            <trap-list>
              <trap-oid>1.3.6.1.4.1.9.9.41.2.0.1</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.3</trap-oid>
            </trap-list>
            <trap-list>
              <trap-oid>1.3.6.1.6.3.1.1.5.4</trap-oid>
            </trap-list>
          </snmp-trap-control>
        </cisco-ia>
      </netconf-yang>
    </config>
  </edit-config>
</rpc>
```

**Step 3** Send the following RPC message to the NETCONF-YANG port to save the running configuration to the startup configuration.

**Example:**

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="">
```

```
<cisco-ia:save-config xmlns:cisco-ia="http://cisco.com/yang/cisco-ia"/>
</rpc>
```

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

Perform this task to configure the SSH public key for NETCONF-YANG to authenticate users.

### SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip ssh pubkey-chain**
4. **username *username***
5. **key-string**
6. **end**

### DETAILED STEPS

	Command or Action	Purpose
<b>Step 1</b>	<b>enable</b> <b>Example:</b> Device> enable	Enables privileged EXEC mode. <ul style="list-style-type: none"> <li>• Enter your password if prompted.</li> </ul>
<b>Step 2</b>	<b>configure terminal</b> <b>Example:</b> Device# configure terminal	Enters global configuration mode.
<b>Step 3</b>	<b>ip ssh pubkey-chain</b> <b>Example:</b> Device(config)# ip ssh pubkey-chain	Configures SSH-RSA keys for user and server authentication on the SSH server and enters public-key configuration mode. <ul style="list-style-type: none"> <li>• The user authentication is successful if the RSA public key stored on the server is verified with the public or the private key pair stored on the client.</li> </ul>
<b>Step 4</b>	<b>username <i>username</i></b> <b>Example:</b> Device(conf-ssh-pubkey)# username user1	Configures the SSH username and enters public-key user configuration mode.
<b>Step 5</b>	<b>key-string</b> <b>Example:</b> Device(conf-ssh-pubkey-user)# key-string	Specifies the RSA public key of the remote peer and enters public-key data configuration mode.  <b>Note</b> You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.

	Command or Action	Purpose
Step 6	<b>end</b> <b>Example:</b> Device(conf-ssh-pubkey-data)# end	Exits public-key data configuration mode and returns to privileged EXEC mode. <ul style="list-style-type: none"> <li>• Use <b>no hostname</b> command to return to the default host.</li> </ul>

## Verifying the NETCONF Protocol Configuration

Use the following commands to verify your NETCONF configuration.

### SUMMARY STEPS

1. **show netconf-yang datastores**
2. **show netconf-yang sessions**
3. **show netconf-yang sessions detail**
4. **show netconf-yang statistics**
5. **show platform software yang-management process**

### DETAILED STEPS

#### Step 1 **show netconf-yang datastores**

Displays information about NETCONF-YANG datastores.

**Example:**

```
Device# show netconf-yang datastores

Device# show netconf-yang datastores
Datastore Name : running
Globally Locked By Session : 42
Globally Locked Time : 2018-01-15T14:25:14-05:00
```

#### Step 2 **show netconf-yang sessions**

Displays information about NETCONF-YANG sessions.

**Example:**

```
Device# show netconf-yang sessions

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore
Number of sessions : 10
session-id transport username source-host global-lock
-----
40 netconf-ssh admin 10.85.70.224 None
42 netconf-ssh admin 10.85.70.224 None
44 netconf-ssh admin 10.85.70.224 None
46 netconf-ssh admin 10.85.70.224 None
48 netconf-ssh admin 10.85.70.224 None
50 netconf-ssh admin 10.85.70.224 None
```

```

52 netconf-ssh admin 10.85.70.224 None
54 netconf-ssh admin 10.85.70.224 None
56 netconf-ssh admin 10.85.70.224 None
58 netconf-ssh admin 10.85.70.224 None

```

### Step 3 show netconf-yang sessions detail

Displays detailed information about NETCONF-YANG sessions.

#### Example:

```

Device# show netconf-yang sessions detail

R: Global-lock on running datastore
C: Global-lock on candidate datastore
S: Global-lock on startup datastore

Number of sessions      : 1

session-id              : 19
transport               : netconf-ssh
username                : admin
source-host             : 2001:db8::1
login-time              : 2018-10-26T12:37:22+00:00
in-rpcs                 : 0
in-bad-rpcs             : 0
out-rpc-errors          : 0
out-notifications      : 0
global-lock             : None

```

### Step 4 show netconf-yang statistics

Displays information about NETCONF-YANG statistics.

#### Example:

```

Device# show netconf-yang statistics

netconf-start-time : 2018-01-15T12:51:14-05:00
in-rpcs : 0
in-bad-rpcs : 0
out-rpc-errors : 0
out-notifications : 0
in-sessions : 10
dropped-sessions : 0
in-bad-hellos : 0

```

### Step 5 show platform software yang-management process

Displays the status of the software processes required to support NETCONF-YANG.

#### Example:

```

Device# show platform software yang-management process

confd          : Running
nesd           : Running
syncfd        : Running
ncsshd        : Running
dmiauthd      : Running
vtyserverutild : Running
opdatamgrd    : Running

```

```
nginx          : Running
ndbmand       : Running
```

**Note** The process *nginx* runs if **ip http secure-server** or **ip http server** is configured on the device. This process is not required to be in the *running* state for NETCONF to function properly. However, the *nginx* process is required for RESTCONF.

**Table 1: show platform software yang-management process Field Descriptions**

Field	Description
confd	Configuration daemon
nesd	Network element synchronizer daemon
syncfd	Sync from daemon
ncsshd	NETCONF Secure Shell (SSH) daemon
dmiauthd	Device management interface (DMI) authentication daemon
vtyservertild	VTY server util daemon
opdatamgrd	Operational Data Manager daemon
nginx	NGINX web server
ndbmand	NETCONF database manager

## Additional References for NETCONF Protocol

### Related Documents

Related Topic	Document Title
YANG data models for various release of IOS-XE, IOS-XR, and NX-OS platforms	To access Cisco YANG models in a developer-friendly way, please clone the <a href="#">GitHub repository</a> , and navigate to the <a href="#">vendor/cisco</a> subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

### Standards and RFCs

Standard/RFC	Title
RFC 6020	<i>YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)</i>
RFC 6241	<i>Network Configuration Protocol (NETCONF)</i>

Standard/RFC	Title
RFC 6536	<i>Network Configuration Protocol (NETCONF) Access Control Model</i>
RFC 8040	<i>RESTCONF Protocol</i>

#### Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	<a href="http://www.cisco.com/support">http://www.cisco.com/support</a>

## Feature Information for NETCONF Protocol

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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

Table 2: Feature Information for NETCONF Protocol

Feature Name	Release	Feature Information
NETCONF Protocol	Cisco IOS XE Denali 16.3.1	<p>The NETCONF Protocol feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices.</p> <p>The following command was introduced: <b>netconf-yang</b>.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco 4000 Series Integrated Services Routers</li> <li>• Cisco ASR 1000 Series Aggregation Services Routers</li> <li>• Cisco Cloud Services Router 1000V Series</li> </ul>
	Cisco IOS XE Everest 16.5.1a	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco Catalyst 3650 Series Switches</li> <li>• Cisco Catalyst 3850 Series Switches</li> </ul>
	Cisco IOS XE Everest 16.6.2	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco Catalyst 9300 Series Switches</li> <li>• Cisco Catalyst 9400 Series Switches</li> <li>• Cisco Catalyst 9500 Series Switches</li> </ul>
	Cisco IOS XE Fuji 16.8.1a	

Feature Name	Release	Feature Information
		<p>In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco 1000 Series Integrated Services Routers</li> <li>• Cisco ASR 900 Series Aggregation Services Routers</li> <li>• Cisco ASR 920 Series Aggregation Services Routers</li> <li>• Cisco Catalyst 9500-High Performance Series Switches</li> <li>• Cisco CBR-8 Series Routers</li> <li>• Cisco Network Convergence System 4200 Series</li> </ul>
	Cisco IOS XE Fuji 16.9.2	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco Catalyst 9200 and 9200L Series Switches</li> <li>• Cisco Catalyst 9300L SKUs</li> </ul>
	Cisco IOS XE Gibraltar 16.10.1	<p>In Cisco IOS XE Gibraltar 16.10.1, this feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco Catalyst 9800-40 Wireless Controllers</li> <li>• Cisco Catalyst 9800-80 Wireless Controllers</li> <li>• Cisco Network Convergence System 520 Series</li> </ul>
	Cisco IOS XE Gibraltar 16.11.1	<p>In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9600 Series Switches.</p>
	Cisco IOS XE Gibraltar 16.12.1	<p>In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9800-L Wireless Controllers.</p>

Feature Name	Release	Feature Information
NETCONF and RESTCONF IPv6 Support	Cisco IOS XE Fuji 16.8.1a	IPv6 support for the NETCONF and RESTCONF protocols. This feature was implemented on the following platforms: <ul style="list-style-type: none"> <li>• Cisco 4000 Series Integrated Services Routers</li> <li>• Cisco ASR 1000 Series Aggregation Services Routers</li> <li>• Cisco ASR 900 Series Aggregation Services Routers</li> <li>• Cisco Catalyst 3650 Series Switches</li> <li>• Cisco Catalyst 3850 Series Switches</li> <li>• Cisco Catalyst 9300 Series Switches</li> <li>• Cisco Catalyst 9400 Series Switches</li> <li>• Cisco Catalyst 9500 Series Switches</li> <li>• Cisco CBR-8 Series Routers</li> <li>• Cisco Cloud Services Router 1000V Series</li> </ul>
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9500-High Performance Series Switches.

Feature Name	Release	Feature Information
NETCONF Global Lock and Kill Session	Cisco IOS XE Fuji 16.8.1a	<p>The NETCONF protocol supports a global lock, and the ability to kill non-responsive sessions. This feature is implemented on the following platforms:</p> <ul style="list-style-type: none"><li>• Cisco 1100 Series Integrated Services Routers</li><li>• Cisco 4000 Series Integrated Services Routers</li><li>• Cisco ASR 1000 Series Aggregation Services Routers</li><li>• Cisco ASR 900 Series Aggregation Services Routers</li><li>• Cisco Catalyst 3650 Series Switches</li><li>• Cisco Catalyst 3850 Series Switches</li><li>• Cisco Catalyst 9300 Series Switches</li><li>• Cisco Catalyst 9400 Series Switches</li><li>• Cisco Catalyst 9500 Series Switches</li><li>• Cisco CBR-8 Series Routers</li><li>• Cisco Cloud Services Router 1000v Series</li></ul>

Feature Name	Release	Feature Information
NETCONF: Candidate Configuration Support	Cisco IOS XE Fuji 16.9.1	<p>The Candidate Config Support feature enables support for candidate capability by implementing RFC 6241 with a simple commit option.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"><li>• Cisco 4000 Series Integrated Services Routers</li><li>• Cisco ASR 1000 Series Aggregation Services Routers</li><li>• Cisco ASR 900 Series Aggregation Services Routers</li><li>• Cisco Catalyst 3650 Series Switches</li><li>• Cisco Catalyst 3850 Series Switches</li><li>• Cisco Catalyst 9300 Series Switches</li><li>• Cisco Catalyst 9400 Series Switches</li><li>• Cisco Catalyst 9500 Series Switches</li><li>• Cisco CBR-8 Series Routers</li><li>• Cisco Cloud Services Router 1000V Series</li></ul> <p>The following command was introduced: <b>netconf-yang feature candidate-datastore.</b></p>

Feature Name	Release	Feature Information
NETCONF: Candidate Configuration Commit Confirm	Cisco IOS XE Amsterdam 17.1.1	<p>The candidate configuration supports the confirmed commit capability.</p> <p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco 1000 Series Integrated Services Routers</li> <li>• Cisco 4000 Series Integrated Services Routers</li> <li>• Cisco ASR 1000 Series Aggregation Services Routers</li> <li>• Cisco ASR 900 Series Aggregation Services Routers</li> <li>• Cisco Catalyst 3650 Series Switches</li> <li>• Cisco Catalyst 3850 Series Switches</li> <li>• Cisco Catalyst 9200 Series Switches</li> <li>• Cisco Catalyst 9300 Series Switches</li> <li>• Cisco Catalyst 9400 Series Switches</li> <li>• Cisco Catalyst 9500 Series Switches</li> <li>• Cisco cBR-8 Converged Broadband Router</li> <li>• Cisco Cloud Services Router 1000V Series</li> <li>• Cisco Network Convergence System 520 Series</li> <li>• Cisco Network Convergence System 4200 Series</li> </ul>

Feature Name	Release	Feature Information
NETCONF-YANG SSH Server Support	Cisco IOS XE Gibraltar 16.12.1	<p>This feature was implemented on the following platforms:</p> <ul style="list-style-type: none"> <li>• Cisco 1000 Series Integrated Services Routers</li> <li>• Cisco 4000 Series Integrated Services Routers</li> <li>• Cisco ASR 900 Series Aggregation Services Routers</li> <li>• Cisco ASR 920 Series Aggregation Services Routers</li> <li>• Cisco ASR 1000 Aggregation Services Routers</li> <li>• Cisco Catalyst 3650 Series Switches</li> <li>• Cisco Catalyst 3850 Series Switches</li> <li>• Cisco Catalyst 9200 Series Switches</li> <li>• Cisco Catalyst 9300 Series Switches</li> <li>• Cisco Catalyst 9400 Series Switches</li> <li>• Cisco Catalyst 9500 Series Switches</li> <li>• Cisco Catalyst 9600 Series Switches</li> <li>• Cisco Catalyst 9800 Series Wireless Controllers</li> <li>• Cisco cBR-8 Converged Broadband Router</li> <li>• Cisco Cloud Services Router 1000V Series</li> <li>• Cisco Network Convergence System 520 Series</li> <li>• Cisco Network Convergence System 4200 Series</li> </ul>

