NETCONF Protocol

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 respository, 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.

Candidate Config Support

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

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 Candidate Datastore. This is a shared data store which enables the user to create, add, delete and make changes

to the device configuration without effecting the running configuration on the device. A commit operation pushes the configuration from “candidate” to the “running” configuration on the device. When “candidate” data store is enabled, the Running data store is NOT writable through NETCONF sessions, all configurations get committed ONLY through Candidate. In other words, the writable-running NETCONF capability is not enabled with Candidate.

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

NETCONF Operations on Candidate

The following operations can be performed on Candidate data store.

Note The information in this section has been referenced from section 8.3.4 of RFC6241. Please refer to the RFC for for more details and the exact RPCs.

Lock

A <lock> RPC is used to “Lock” the target data store. This prevents others users from modifying the configuration in the “locked” data store. We can lock both candidate and running data through the lock operation.

Note “Locking” candidate datastore does not affect Cisco IOS config lock or running config 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 “Running” or “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” config 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

Candidate configuration can be used as a target for “edit-config” operation to modify a configuration. This provides you the the ability to stage configuration changes in candidate without affecting the running configuration on the device.

Discard

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

If the contents of candidate datastore have already been modified by NETCONF session A, and session B tries to lock the Candidate, the lock will fail. NETCONF session B must perform a <discard> operation to remove any outstanding configurations on candidate 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 data store, by specifying "candidate" as target in the Unlock RPC. Candidate is now available for all operations in other sessions.

If failure occurs with outstanding changes to candidate datastore, it can be challenging to recover the configuration and creates problems for other sessions. To avoid any issues, outstanding changes are discarded in candidate when the lock is released—either implicitly on "NETCONF session failure" or explicitly on "Unlock" operation.

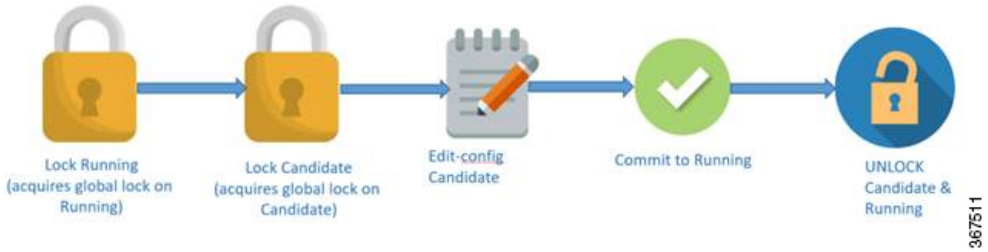
Get-config, Copy-config, Validate

Candidate 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 changes in candidate to the device but only to validate the configuration, you can do so by <validate> RPC followed by a "discard".

How to use Candidate?

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

Figure 1: Modifying Candidate Datastore Steps



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

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 will be displayed notifying the user that a restart of netconf-yang or restconf will occur in order for the change to take effect. When candidate is enabled, The running data store is *not* writable through Netconf sessions, all configurations get committed *only* through candidate. In other words, the writable-running Netconf capability is not enabled with candidate.

If the selection of candidate or running datastore, is specified in the configuration when a netconf-yang or restconf confd process starts, a warning 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 candidate or running is made after 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.

Note Candidate data store is a shared data store, that is, multiple Netconf sessions can modify the contents simultaneously. Therefore, it is important for a user to lock the data store before modifying its contents, to prevent conflicting commits which can eventually lead to losing any configuration changes; wherein another user overwrites the configuration by modifying the configuration and issuing a commit.

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
Step 1	**enable** **Example:** `Device# enable`	Enables privileged EXEC mode. Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**username** *name* **privilege** *level* **password** *password* **Example:** `Device(config)# username example-name privilege 15 password example_password`	Establishes a user name-based authentication system. Configure the following keywords: • **privilege** *level*: Sets the privilege level for the user. For the NETCONF protocol, it must be 15. • **password** *password*: Sets a password to access the CLI view.
Step 4	**aaa new-model** **Example:** `Device(config)# aaa new-model`	(Optional) Enables authorisation, authentication, and accounting (AAA). If the **aaa new-model** command is configured, AAA authentication and authorization is required.
Step 5	**aaa authentication login default local** **Example:** `Device(config)# aaa authentication login default local`	Sets the login authentication to use the local username database. **Note** Only the default AAA authentication login method is supported for the NETCONF protocol. • For a remote AAA server, replace *local* with your AAA server. The **default** keyword applies the local user database authentication to all ports.
Step 6	**aaa authorization exec default local** **Example:** `Device(config)# aaa authorization exec default local`	Configures user AAA authorization, check the local database, and allows the user to run an EXEC shell. **Note** Only the default AAA authorization method is supported for the NETCONF protocol. • For a remote AAA server, replace *local* with your AAA server. • The **default** keyword applies the local user database authentication to all ports.

	Command or Action	Purpose
Step 7	**end** **Example:** `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
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**netconf-yang** **Example:** `Device (config)# netconf-yang`	Enables the NETCONF interface on your network device. **Note** 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.
Step 4	**netconf-yang feature candidate-datastore** **Example:** `Device(config)# netconf-yang feature candidate-datastore`	Enables candidate datastore.
Step 5	**exit** **Example:** `Device (config)# exit`	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>
```

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 inteface (DMI) authentication daemon
vtyserverutild	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 *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.

Standards and RFCs

Standard/RFC	Title
RFC 6020	*YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
RFC 6241	*Network Configuration Protocol (NETCONF)*
RFC 6536	*Network Configuration Protocol (NETCONF) Access Control Model*
RFC 8040	*RESTCONF Protocol*

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. 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. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

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. 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	The NETCONF Protocol feature facilitates a programmatic and standards-based way of writing configurations and reading operational data from network devices. The following command was introduced: **netconf-yang**. This feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco Cloud Services Router 1000V Series
	Cisco IOS XE Everest 16.5.1a	This feature was implemented on the following platforms: • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches
	Cisco IOS XE Everest 16.6.2	This feature was implemented on the following platforms: • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches
	Cisco IOS XE Fuji 16.8.1a	

Feature Name	Release	Feature Information
		In Cisco IOS XE Fuji 16.8.1a, this feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco Catalyst 9500-High Performance Series Switches • Cisco CBR-8 Series Routers • Cisco Network Convergence System 4200 Series
	Cisco IOS XE Fuji 16.9.2	This feature was implemented on the following platforms: • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.10.1	In Cisco IOS XE Gibraltar 16.10.1, this feature was implemented on the following platforms: • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco Network Convergence System 520 Series
	Cisco IOS XE Gibraltar 16.11.1	In Cisco IOS XE Gibraltar 16.11.1, this feature was implemented on Cisco Catalyst 9600 Series Switches.

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: • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000V Series
	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	The NETCONF protocol supports a global lock, and the ability to kill non-responsive sessions. This feature is implemented on the following platforms: • Cisco 1100 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000v Series

Feature Name	Release	Feature Information
NETCONF: Candidate Configuration Support	Cisco IOS XE Fuji 16.9.1	The Candidate Config Support feature enables support for candidate capability by implementing RFC 6241 with a simple commit option. This feature was implemented on the following platforms: • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco CBR-8 Series Routers • Cisco Cloud Services Router 1000V Series The following command was introduced: **netconf-yang feature candidate-datastore**.