

NETCONF Protocol

- Information About the NETCONF Protocol, on page 1
- How to Configure the NETCONF Protocol, on page 25
- Verifying the NETCONF Protocol Configuration Through the CLI, on page 31
- Example: Named Method List, on page 34
- Displaying NETCONF-YANG Diagnostics Through RPCs, on page 34
- Additional References for NETCONF Protocol, on page 37
- Feature Information for the NETCONF Protocol, on page 39

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.

From Cisco IOS XE 17.15.1, use the **yang-interfaces feature deprecated disable** command or the Cisco-IOS-XE-Yang-interfaces-cfg.yang model to disable the deprecated NETCONF-YANG model elements. By default, deprecated model elements remain enabled.

From Cisco IOS XE 17.18.1, the deprecated NETCONF-YANG model elements are disabled by default. To enable these deprecated elements for backwards compatibility, use the **yang-interfaces feature deprecated enable** command or configure the Cisco-IOS-XE-yang-interfaces-cfg.yang model > general > disable-deprecated leaf.

This behavior changes from Cisco IOS XE 17.15.1 in which the deprecated elements were enabled by default.

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.

Restrictions for the NETCONF Protocol

- NETCONF is not supported on a device running dual IOSd configuration or software redundancy.
- If RP addresses from the NETCONF datastore are removed using the **no ip pim rp-address** command, there could be inconsistencies in the datastore, due to parser limitations. Use the RPC to remove RP address entries from the NETCONF datastore.
- On Cisco Catalyst 9300 Series Switches, if the IP address the NETCONF server IP address and NAT interface overload IP address is the same, BINOS services like NETCONF may not work. Other IOSd services will work normally.

YANG Model Version 1.1

YANG Version 1.1 is described by the *RFC 7950, The YANG 1.1 Data Modeling Language*. YANG Version 1.1 is a maintenance release of the YANG language that addresses ambiguities and defects in the YANG Version 1.0 specification.

The YANG module in YANG Version 1.1 is advertised through the *ietf-yang-library* instead of the NETCONF hello messages.

The following example shows the NETCONF <*get*> RPC that retrieves a list of all the YANG modules supported by a device:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>
    <filter>
      <modules-state xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library"/>
    </filter>
  </get>
</rpc>
```

The output of the RPC reply contains a list of all the YANG modules regardless of the YANG version each module uses.

Cisco IOS XE Cupertino 17.7.1 uses the YANG Version 1.0; however, you can still download the YANG Version 1.1 from GitHub at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe.

Alternatively, you can also download the YANG models from the device using the NETCONF *get-schema* operation, and migrate the downloaded models to this version using the *migrate_yang_version.py* script.

The following example shows how to migrate from YANG Version 1.0 to YANG Version 1.1 using the script:

```
migrate_yang_version.py [-h] [--out OUT] path
```

Use the **help** command to view the options available with the script:

```
python migrate_yang_version.py --help
usage: migrate_yang_version.py [-h] [--out OUT] path

positional arguments:
  path        Path to the YANG files

optional arguments:
  -h, --help  show this help message and exit
  --out OUT   Path to the output YANG file
```

The following example shows how to use the *out* argument to move a file from its original location to another folder:

```
python migrate_yang_version.py --out testdir/outdir testdir/indir
```

In the above example, *testdir/outdir* is the directory in which the YANG model Version 1.1 resides or where the output of the script is placed. This directory will be created, if it is not available.

The *testdir/indir* directory is where the YANG model Version 1.0 resides; the input for the script.

After the YANG model Version 1.1 is created, either by downloading it from GitHub or by using the *migrate_yang_version.py* script and compiled on the client application, end-to-end YANG model tests can be executed and validated against Cisco IOS XE devices.

Note The YANG models on the device is still YANG Version 1.0. However; there is no need to change the RPC payload of the client test cases.

For inquiries related to the *migrate_yang_version.py* script or the Cisco IOS XE YANG migration process, send an email to *xe-yang-migration@cisco.com*.

Cisco IOS XE Cupertino 17.8.1 uses YANG Version 1.1. The difference between YANG Version 1.1 and Version 1.0 is documented at https://tools.ietf.org/html/rfc7950#page-10.

YANG Version in Cisco IOS XE Dublin 17.10.1

Cisco-defined YANG models are in YANG Version 1.1 in Cisco IOS XE Dublin 17.10.1 and later releases. You can download this version from GitHub at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe.

In YANG Version 1.1, the critical change that impacts client applications that use NETCONF is in the <*hello*> message content. As per RFC 7950, a server advertises support for YANG 1.1 modules by using the *ietf-yang-library*, instead of listing them as capabilities in the <*hello*> message. We recommend that you use the *ietf-yang-library* to gather the list of supported YANG modules, instead of deriving this list from the <*hello*> message content.

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.

Converting IOS Commands to XML

In Cisco IOS XE Cupertino 17.7.1 and later releases, you can automatically translate IOS commands into relevant NETCONF-YANG XML or RESTCONF-JSON request messages. You can analyze the generated configuration messages and familiarize with the Xpaths used in these messages. The generated configuration in the structured format can be used to provision other devices in the network; however, this configuration cannot be modified.

Use the **show running-config** | **format netconf-xml** command or the **show running-config** | **format restconf-json** command to translate IOS commands.

If the **netconf-xml** keyword is selected, the IOS commands are translated into the NETCONF-YANG XML format, and if the **restconf-json** keyword is selected, the IOS commands are translated into the RESTCONF-JSON format.

The translation of IOS commands into a structured format is disabled by default. You must initially configure NETCONF-YANG, and once the data model interfaces (DMIs) are initialized, use the appropriate format option to translate the commands.

The following is sample output from the **show running-config** | **format netconf-xml** command:

```
Device# show running-config | format netconf-xml

<config xmlns="http://tail-f.com/ns/config/1.0">
  <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
    <version>17.8</version>
    <boot-start-marker/>
    <boot>
      <system>
        <flash>
          <flash-list-ordered-by-user>

<flash-leaf>bootflash:c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20211020_005209.SSA.bin</
              flash-leaf>
          </flash-list-ordered-by-user>
        </flash>
      </system>
    </boot>
    <boot-end-marker/>
    <memory>
      <free>
        <low-watermark>
          <processor>64219</processor>
        </low-watermark>
      </free>
    </memory>
    <call-home>
      <contact-email-addr xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
        sch-smart-licensing@cisco.com</contact-email-addr>
      <tac-profile xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-call-home">
        <profile>
          <CiscoTAC-1>
            <active>true</active>
            <destination>
              <transport-method>http</transport-method>
            </destination>
          </CiscoTAC-1>
        </profile>
      </tac-profile>
    </call-home>
    <service>
      <timestamps>
        <debug-config>
          <datetime>
            <msec/>
            <localtime/>
            <show-timezone/>
          </datetime>
        </debug-config>
        <log-config>
          <datetime>
            <msec/>
            <localtime/>
            <show-timezone/>
          </datetime>
        </log-config>
      </timestamps>
      <call-home/>
    </service>
```

```
<platform>
  <console xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <output>serial</output>
  </console>
  <qfp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <utilization>
      <monitor>
        <load>80</load>
      </monitor>
    </utilization>
  </qfp>
  <punt-keepalive xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-platform">
    <disable-kernel-core>true</disable-kernel-core>
  </punt-keepalive>
</platform>
<hostname>pi-prog-csr1</hostname>
<enable>
  <password>
    <secret>lab</secret>
  </password>
</enable>
<username>
  <name>admin</name>
  <privilege>15</privilege>
  <password>
    <encryption>0</encryption>
    <password>lab</password>
  </password>
</username>
<vrf>
  <definition>
    <name>Mgmt-intf</name>
    <address-family>
      <ipv4>
      </ipv4>
      <ipv6>
      </ipv6>
    </address-family>
  </definition>
</vrf>
<ip>
  <domain>
    <name>cisco</name>
  </domain>
  <forward-protocol>
    <protocol>nd</protocol>
  </forward-protocol>
  <route>
    <ip-route-interface-forwarding-list>
      <prefix>10.0.0.0</prefix>
      <mask>255.255.0.0</mask>
      <fwd-list>
        <fwd>10.45.0.1</fwd>
      </fwd-list>
    </ip-route-interface-forwarding-list>
    <vrf>
      <name>Mgmt-intf</name>
      <ip-route-interface-forwarding-list>
        <prefix>0.0.0.0</prefix>
        <mask>0.0.0.0</mask>
        <fwd-list>
          <fwd>10.104.54.129</fwd>
        </fwd-list>
      </ip-route-interface-forwarding-list>
```

```
    </vrf>
  </route>
  <ssh>
    <ssh-version>2</ssh-version>
  </ssh>
  <tftp>
    <source-interface>
      <GigabitEthernet>1</GigabitEthernet>
    </source-interface>
    <blocksize>8192</blocksize>
  </tftp>
  <http xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-http">
    <authentication>
      <local/>
    </authentication>
    <server>true</server>
    <secure-server>true</secure-server>
  </http>
</ip>
<ipv6>
  <unicast-routing/>
</ipv6>
<interface>
  <GigabitEthernet>
    <name>1</name>
    <vrf>
      <forwarding>Mgmt-intf</forwarding>
    </vrf>
    <ip>
      <address>
        <primary>
          <address>10.104.54.222</address>
          <mask>255.255.255.128</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>2</name>
    <ip>
      <address>
        <primary>
          <address>9.45.21.231</address>
          <mask>255.255.0.0</mask>
        </primary>
      </address>
    </ip>
    <mop>
      <enabled>false</enabled>
      <sysid>false</sysid>
    </mop>
    <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
      <auto>true</auto>
    </negotiation>
  </GigabitEthernet>
  <GigabitEthernet>
    <name>3</name>
```

```
      <mop>
        <enabled>false</enabled>
        <sysid>false</sysid>
      </mop>
      <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
        <auto>true</auto>
      </negotiation>
    </GigabitEthernet>
    <GigabitEthernet>
      <name>4</name>
      <mop>
        <enabled>false</enabled>
        <sysid>false</sysid>
      </mop>
      <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
        <auto>true</auto>
      </negotiation>
    </GigabitEthernet>
    <GigabitEthernet>
      <name>5</name>
      <mop>
        <enabled>false</enabled>
        <sysid>false</sysid>
      </mop>
      <negotiation xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ethernet">
        <auto>true</auto>
      </negotiation>
    </GigabitEthernet>
  </interface>
  <control-plane>
  </control-plane>
  <clock>
    <timezone>
      <zone>IST</zone>
      <hours>5</hours>
      <minutes>30</minutes>
    </timezone>
  </clock>
  <logging>
    <console-config>
      <console>false</console>
    </console-config>
  </logging>
  <aaa>
    <new-model xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa"/>
    <authentication xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
      <login>
        <name>default</name>
        <a1>
          <local/>
        </a1>
      </login>
    </authentication>
    <authorization xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
      <exec>
        <name>default</name>
        <a1>
          <local/>
        </a1>
      </exec>
    </authorization>
    <common-criteria xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">
      <policy>enable_secret_policy</policy>
      <char-changes>4</char-changes>
```

```
            <lower-case>1</lower-case>
            <max-length>127</max-length>
            <min-length>10</min-length>
            <numeric-count>1</numeric-count>
            <upper-case>1</upper-case>
          </common-criteria>
          <session-id xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-aaa">common</session-id>
        </aaa>
        <login>
          <on-success>
            <log>
            </log>
          </on-success>
        </login>
        <multilink>
          <bundle-name
    xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-ppp">authenticated</bundle-name>
        </multilink>
        <redundancy>
        </redundancy>
        <spanning-tree>
          <extend xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-spanning-tree">
            <system-id/>
          </extend>
        </spanning-tree>
        <subscriber>
          <templating/>
        </subscriber>
        <crypto>
          <pki xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-crypto">
            <certificate>
              <chain>
                <name>SLA-TrustPoint</name>
                <certificate>
                  <serial>01</serial>
                  <certtype>ca</certtype>
                </certificate>
              </chain>
              <chain>
                <name>TP-self-signed-2685563505</name>
                <certificate>
                  <serial>01</serial>
                  <certtype>self-signed</certtype>
                </certificate>
              </chain>
            </certificate>
            <trustpoint>
              <id>SLA-TrustPoint</id>
              <enrollment>
                <pkcs12/>
              </enrollment>
              <revocation-check>crl</revocation-check>
            </trustpoint>
            <trustpoint>
              <id>TP-self-signed-2685563505</id>
              <enrollment>
                <selfsigned/>
              </enrollment>
              <revocation-check>none</revocation-check>
              <rsakeypair>
                <key-label>TP-self-signed-2685563505</key-label>
              </rsakeypair>
              <subject-name>cn=IOS-Self-Signed-Certificate-2685563505</subject-name>
            </trustpoint>
```

```
  </pki>
</crypto>
<license>
  <udi>
    <pid>C8000V</pid>
    <sn>93SHKMJKOC6</sn>
  </udi>
  <boot>
    <level>
      <network-advantage>
        <addon>dna-advantage</addon>
      </network-advantage>
    </level>
  </boot>
</license>
<line>
  <aux>
    <first>0</first>
  </aux>
  <console>
    <first>0</first>
    <exec-timeout>
      <minutes>0</minutes>
      <seconds>0</seconds>
    </exec-timeout>
    <stopbits>1</stopbits>
  </console>
  <vty>
    <first>0</first>
    <last>4</last>
    <exec-timeout>
      <minutes>0</minutes>
      <seconds>0</seconds>
    </exec-timeout>
    <password>
      <secret>lab</secret>
    </password>
    <transport>
      <input>
        <all/>
      </input>
      <output>
        <all/>
      </output>
    </transport>
  </vty>
  <vty>
    <first>5</first>
    <last>31</last>
    <transport>
      <input>
        <all/>
      </input>
      <output>
        <all/>
      </output>
    </transport>
  </vty>
</line>
<diagnostic xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-diagnostics">
  <bootup>
    <level>minimal</level>
  </bootup>
</diagnostic>
```

```
  </native>
</config>
pi-prog-csr1#
pi-prog-csr1#
pi-prog-csr1#show running-config | format restconf-json
{
  "data": {
    "Cisco-IOS-XE-native:native": {
      "version": "17.8",
      "boot-start-marker": [null],
      "boot": {
        "system": {
          "flash": {
            "flash-list-ordered-by-user": [
              {
                "flash-leaf":
"bootflash:c8000v-universalk9.BLD_POLARIS_DEV_LATEST_20211020_005209.SSA.bin"
              }
            ]
          }
        }
      },
      "boot-end-marker": [null],
      "memory": {
        "free": {
          "low-watermark": {
            "processor": 64219
          }
        }
      },
      "call-home": {
        "Cisco-IOS-XE-call-home:contact-email-addr": "sch-smart-licensing@cisco.com",
        "Cisco-IOS-XE-call-home:tac-profile": {
          "profile": {
            "CiscoTAC-1": {
              "active": true,
              "destination": {
                "transport-method": "http"
              }
            }
          }
        }
      },
      "service": {
        "timestamps": {
          "debug-config": {
            "datetime": {
              "msec": [null],
              "localtime": [null],
              "show-timezone": [null]
            }
          },
          "log-config": {
            "datetime": {
              "msec": [null],
              "localtime": [null],
              "show-timezone": [null]
            }
          }
        },
        "call-home": [null]
      },
      "platform": {
        "Cisco-IOS-XE-platform:console": {
```

```
    "output": "serial"
  },
  "Cisco-IOS-XE-platform:qfp": {
    "utilization": {
      "monitor": {
        "load": 80
      }
    }
  },
  "Cisco-IOS-XE-platform:punt-keepalive": {
    "disable-kernel-core": true
  }
},
"hostname": "pi-prog-csr1",
"enable": {
  "password": {
    "secret": "lab"
  }
},
"username": [
  {
    "name": "admin",
    "privilege": 15,
    "password": {
      "encryption": "0",
      "password": "lab"
    }
  }
],
"vrf": {
  "definition": [
    {
      "name": "Mgmt-intf",
      "address-family": {
        "ipv4": {
        },
        "ipv6": {
        }
      }
    }
  ]
},
"ip": {
  "domain": {
    "name": "cisco"
  },
  "forward-protocol": {
    "protocol": "nd"
  },
  "route": {
    "ip-route-interface-forwarding-list": [
      {
        "prefix": "10].0.0.0",
        "mask": "255.255.0.0",
        "fwd-list": [
          {
            "fwd": "9.45.0.1"
          }
        ]
      }
    ],
    "vrf": [
      {
        "name": "Mgmt-intf",
```

```
        "ip-route-interface-forwarding-list": [
          {
            "prefix": "0.0.0.0",
            "mask": "0.0.0.0",
            "fwd-list": [
              {
                "fwd": "10.104.54.129"
              }
            ]
          }
        ]
      }
    ]
  },
  "ssh": {
    "ssh-version": "2"
  },
  "tftp": {
    "source-interface": {
      "GigabitEthernet": "1"
    },
    "blocksize": 8192
  },
  "Cisco-IOS-XE-http:http": {
    "authentication": {
      "local": [null]
    },
    "server": true,
    "secure-server": true
  }
},
"ipv6": {
  "unicast-routing": [null]
},
"interface": {
  "GigabitEthernet": [
    {
      "name": "1",
      "vrf": {
        "forwarding": "Mgmt-intf"
      },
      "ip": {
        "address": {
          "primary": {
            "address": "10.104.54.222",
            "mask": "255.255.255.128"
          }
        }
      },
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "2",
      "ip": {
        "address": {
          "primary": {
            "address": "10.45.21.231",
            "mask": "255.255.0.0"
```

```
          }
        }
      },
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "3",
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "4",
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    },
    {
      "name": "5",
      "mop": {
        "enabled": false,
        "sysid": false
      },
      "Cisco-IOS-XE-ethernet:negotiation": {
        "auto": true
      }
    }
  ]
},
"control-plane": {
},
"clock": {
  "timezone": {
    "zone": "IST",
    "hours": 5,
    "minutes": 30
  }
},
"logging": {
  "console-config": {
    "console": false
  }
},
"aaa": {
  "Cisco-IOS-XE-aaa:new-model": [null],
  "Cisco-IOS-XE-aaa:authentication": {
    "login": [
      {
        "name": "default",
        "a1": {
```

```
          "local": [null]
        }
      }
    ]
  },
  "Cisco-IOS-XE-aaa:authorization": {
    "exec": [
      {
        "name": "default",
        "a1": {
          "local": [null]
        }
      }
    ]
  },
  "Cisco-IOS-XE-aaa:common-criteria": [
    {
      "policy": "enable_secret_policy",
      "char-changes": 4,
      "lower-case": 1,
      "max-length": 127,
      "min-length": 10,
      "numeric-count": 1,
      "upper-case": 1
    }
  ],
  "Cisco-IOS-XE-aaa:session-id": "common"
},
"login": {
  "on-success": {
    "log": {
    }
  }
},
"multilink": {
  "Cisco-IOS-XE-ppp:bundle-name": "authenticated"
},
"redundancy": {
},
"spanning-tree": {
  "Cisco-IOS-XE-spanning-tree:extend": {
    "system-id": [null]
  }
},
"subscriber": {
  "templating": [null]
},
"crypto": {
  "Cisco-IOS-XE-crypto:pki": {
    "certificate": {
      "chain": [
        {
          "name": "SLA-TrustPoint",
          "certificate": [
            {
              "serial": "01",
              "certtype": "ca"
            }
          ]
        },
        {
          "name": "TP-self-signed-2685563505",
          "certificate": [
            {
```

```
                "serial": "01",
                "certtype": "self-signed"
              }
            ]
          }
        ]
      },
      "trustpoint": [
        {
          "id": "SLA-TrustPoint",
          "enrollment": {
            "pkcs12": [null]
          },
          "revocation-check": ["crl"]
        },
        {
          "id": "TP-self-signed-2685563505",
          "enrollment": {
            "selfsigned": [null]
          },
          "revocation-check": ["none"],
          "rsakeypair": {
            "key-label": "TP-self-signed-2685563505"
          },
          "subject-name": "cn=IOS-Self-Signed-Certificate-2685563505"
        }
      ]
    }
  },
  "license": {
    "udi": {
      "pid": "C8000V",
      "sn": "93SHKMJKOC6"
    },
    "boot": {
      "level": {
        "network-advantage": {
          "addon": "dna-advantage"
        }
      }
    }
  },
  "line": {
    "aux": [
      {
        "first": "0"
      }
    ],
    "console": [
      {
        "first": "0",
        "exec-timeout": {
          "minutes": 0,
          "seconds": 0
        },
        "stopbits": "1"
      }
    ],
    "vty": [
      {
        "first": 0,
        "last": 4,
        "exec-timeout": {
          "minutes": 0,
```

```
            "seconds": 0
          },
          "password": {
            "secret": "lab"
          },
          "transport": {
            "input": {
              "all": [null]
            },
            "output": {
              "all": [null]
            }
          }
        },
        {
          "first": 5,
          "last": 31,
          "transport": {
            "input": {
              "all": [null]
            },
            "output": {
              "all": [null]
            }
          }
        }
      ]
    },
    "Cisco-IOS-XE-diagnostics:diagnostic": {
      "bootup": {
        "level": "minimal"
      }
    }
  }
 }
}
```

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.

NETCONF SSH Algorithms

The NETCONF-SSH server configuration file contains the list of all supported algorithms. From Cisco IOS XE Dublin 17.12.1, you can enable or disable these algorithms at runtime by using commands or YANG models.

Use the **netconf-yang ssh server algorithm** {**encryption** | **kex** | **mac** | **hostkey**} command to enable the algorithms. Use the **no** form of this command to disable the algorithms. Also, the output of the **show netconf-yang status** command will display the list of configured algorithms.

Users can also enable or disable the NETCONF-SSH algorithms through YANG models. The following is a sample of the NETCONF request for the corresponding model:

```
<yang-interfaces-cfg-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-yang-interfaces-cfg">

      <ssh-server>
            <kex-algorithms>
                  <dh-group14-sha1>false</dh-group14-sha1>
            </kex-algorithms>
            <macs>
                  <hmac-sha1>false</hmac-sha1>
            </macs>
            <ciphers>
                  <aes128-cbc>true</aes128-cbc>
            </ciphers>
            <hostkey-algorithms>
                  <rsa-sha2-256>true </rsa-sha2-256>
            </hostkey-algorithms>
      </ssh-server>
</yang-interfaces-cfg-data>
```

Named Method List

gNMI, NETCONF, and RESTCONF uses the Cisco IOS authentication, authorization, and accounting (AAA) server to authenticate and authorize an user. Prior to the introduction of the named method-list, only the *default* method list was supported for authentication and authorization. This meant that the administrator could not use a custom method-list name for authentication and authorization.

With the introduction of the Named Method List feature, it is possible to use a custom method-list name for gNMI, NETCONF, and RESTCONF authentication and authorization, without changing the existing AAA configuration of a device. You can use the **yang-interfaces aaa {authentication | authorization} method-list** *named-method-list* command to create a custom method-list. Named method lists can provide multiple authentication and authorization options.

A method list is a named list that describes the authentication and authorization methods to be queried, such as, AAA, Lightweight Directory Access Protocol (LDAP), RADIUS, or TACACS+. Method lists define the method and the sequence in which authorization is performed. Method lists enable one or more security protocols for authentication and authorization, ensuring that a backup system is available in case of a failure.

The following is a sample NETCONF RPC that displays the named method list:

```
<edit-config>
<target><running/></target>
<config>
  <yang-interfaces-cfg-data xmlns=http://cisco.com/ns/yang/Cisco-IOS-XE-yang-interfaces-cfg>

    <aaa>
```

```
      <authn>
        <login-method-list>test</login-method-list>
      </authn>
      <authz>
        <exec-method-list>test</exec-method-list>
      </authz>
    </aaa>
  </yang-interfaces-cfg-data>
</config>
</edit-config>
```

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.

When a candidate datastore configuration changes, CONFd is re-initialized to adapt to the new configuration, and all management interfaces, gNMI, NETCONF, and RESTCONF experience a flap during the re-initialization.

We recommend that you configure the Candidate Configuration feature before enabling NETCONF. If the candidate configuration is enabled or disabled through NETCONF, the connection is lost immediately after making this change.

Note The candidate datastore is a shared data store and multiple NETCONF sessions can modify its contents simultaneously. Therefore, it is important to lock the datastore before modifying the 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 others 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 ca nuse 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:nc="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.

Side-Effect Synchronization of the Configuration Database

During configuration changes in the data model interface (DMI), a partial synchronization of the changes that are triggered when a command or RPC is configured happens. This is called the side-effect synchronization, and it reduces the synchronization time and NETCONF downtime. Prior to the side-effect synchronization, any configuration change used to trigger a time-consuming full synchronization of the configuration database.

The side-effect synchronization is enabled by the **netconf-yang feature side-effect-sync** command.

Some commands, when they are configured, triggers changes in some already configured commands. For example, the following is the configuration on a device before the NETCONF edit-config RPC is configured:

```
hostname device123
```

The NETCONF edit-config RPC:

```
<native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
 <hostname xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="delete"/>
</native>
```

The following is the configuration on the device after the NETCONF edit-config RPC is configured:

```
hostname Switch
```

Here, the side-effect of the NETCONF edit-config RPC is a change to the running configuration that is not directly intended by the RPC. The edit-config request is supposed to delete the host name, but instead the hostname is changed back to Switch. The side-effect synchronization does a synchronization of this configuration change to the NETCONF database without synchronizing the entire configuration, thereby improving performance.

The side-effect synchronization is based on the CLI-mode tree concept, where the commands are maintained with modes and submodes structure. This CLI-mode tree data structure consists of three main nodes:

- Same-Level Node: This node points to the list of CLI nodes that belongs to the same parent and on the same level.
- Parent Node: This node points to the CLI nodes parent, its mode, and submode node.
- Child Node: This node points to the child CLI; the CLI under the current mode or submode. If the node has multiple child nodes then those child nodes are linked as part of the same-level node pointers.

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

Procedure

	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.

	Command or Action	Purpose
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** Prior to Cisco IOS XE Cupertino 17.9.1, only the default AAA authentication login method is supported for the NETCONF protocol. From Cisco IOS XE Cupertino 17.9.1, named method-list is supported. • 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** Prior to Cisco IOS XE Cupertino 17.9.1, only the default AAA authentication login method is supported for the NETCONF protocol. From Cisco IOS XE Cupertino 17.9.1, named method-list is supported. • For a remote AAA server, replace *local* with your AAA server. • The **default** keyword applies the local user database authentication to all ports.
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

Procedure

	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

Procedure

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

Procedure

	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	**ip ssh pubkey-chain** **Example:** `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. • 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.
Step 4	**username** *username* **Example:** `Device(conf-ssh-pubkey)# username user1`	Configures the SSH username and enters public-key user configuration mode.
Step 5	**key-string** **Example:** `Device(conf-ssh-pubkey-user)# key-string`	Specifies the RSA public key of the remote peer and enters public-key data configuration mode. **Note**

	Command or Action	Purpose
		You can obtain the public key value from an open SSH client; that is, from the .ssh/id_rsa.pub file.
Step 6	**end** **Example:** `Device(conf-ssh-pubkey-data)# end`	Exits public-key data configuration mode and returns to privileged EXEC mode. • Use **no hostname** command to return to the default host.

Configuring a Named Method List

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **yang-interfaces aaa authentication method-list** *named-method-list*
4. **yang-interfaces aaa authorization method-list** *named-method-list*
5. **exit**

DETAILED STEPS

Procedure

	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	**yang-interfaces aaa authentication method-list** *named-method-list* **Example:** `Device(config)# yang-interfaces aaa authentication method-list authn-method`	Configures a named method-list for authentication.
Step 4	**yang-interfaces aaa authorization method-list** *named-method-list* **Example:** `Device(config)# yang-interfaces aaa authorization method-list authr-method`	Configures a named method-list for authorization.

	Command or Action	Purpose
Step 5	**exit** **Example:** `Device(config)# exit`	Exits global configuration mode and returns to privileged EXEC mode.

Verifying the NETCONF Protocol Configuration Through the CLI

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 diagnostics summary**
5. **show netconf-yang statistics**
6. **show platform software yang-management process**

DETAILED STEPS

Procedure

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 diagnostics summary**

Displays a summary of the NETCONF-YANG diagnostic information.

Example:

```
Device# show netconf-yang diagnostics summary

Diagnostic Debugging is ON
Diagnostic Debugging Level: Maximum
Total Log Size (bytes): 20097
Total Transactions: 1
message username session-id transaction-id start-time        end-time          log size
-------------------------------------------------------------------------------------
1       admin    35          53            03/12/21 14:31:03 03/12/21 14:31:04 20097
```

Step 5 **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 6 **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

Example: Named Method List

Along with the default method-list, you can enable multiple authentication or authorization options with the named method-list. Method lists enable one or more security protocols to be used for authorization. The method lists are processed serially by the Cisco IOS software. If the first configured method-list fails, the next one is processed. This process continues until a successful authentication or authorization, or until all configured methods are exhausted.

The following example shows how to configure named method-lists for NETCONF AAA:

```
Device> enable
Device# configure terminal
Device(config)# netconf-yang
Device(config)# yang-interfaces aaa authentication method-list netconf-authn
Device(config)# yang-interfaces aaa authorization method-list netconf-authr
Device(config)# end
```

Displaying NETCONF-YANG Diagnostics Through RPCs

You can either use the **show netconf-yang diagnostics** command or the following RPCs to view the diagnostics information.

The following is a sample RPC that enables NETCONF-YANG diagnostics, and the RPC response received from the host:

```
#308
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:b0f45ac0-3fe2-4e1d-a3a1-f57985965be6">
  <enable-netconf-diag xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc">
    <diag-level>diag-maximum</diag-level>
  </enable-netconf-diag>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:b0f45ac0-3fe2-4e1d-a3a1-f57985965be6"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

The following is a sample RPC that shows the current status and the RPC response received from the host:

```
#294
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:c6c986ac-fc44-45e2-9390-f8a5968dc8d4">
  <nc:get>
    <nc:filter>
```

```
      <netconf-diag-oper-data
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper"/>
    </nc:filter>
  </nc:get>
</nc:rpc>

#

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:c6c986ac-fc44-45e2-9390-f8a5968dc8d4"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <netconf-diag-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper">

      <diag-summary>
        <level>diag-maximum</level>
        <log-size>0</log-size>
        <trans-count>0</trans-count>
      </diag-summary>
    </netconf-diag-oper-data>
  </data>
</rpc-reply>
```

The following is a sample RPC to change the host name and the RPC response received from the host:

```
#
#364
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:f3005ee6-8a11-4146-b616-dd95a92b97d1">
  <nc:edit-config>
    <nc:target>
      <nc:running/>
    </nc:target>
    <nc:config>
      <native xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-native">
        <hostname>new-ott-c9300-35</hostname>
      </native>
    </nc:config>
  </nc:edit-config>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:f3005ee6-8a11-4146-b616-dd95a92b97d1"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

The following is a sample RPC to display the current status and the RPC response received from the host:

```
 #294
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:9bffb8d5-3866-48ef-b59d-0486e508fbc4">
  <nc:get>
    <nc:filter>
```

```
      <netconf-diag-oper-data
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper"/>
    </nc:filter>
  </nc:get>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:9bffb8d5-3866-48ef-b59d-0486e508fbc4"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
   <netconf-diag-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-oper">

      <diag-summary>
        <level>diag-maximum</level>
        <log-size>20775</log-size>
        <trans-count>1</trans-count>
      </diag-summary>
      <diag-trans>
        <message>1</message>
        <username>lab</username>
        <session-id>31</session-id>
        <trans-id>50</trans-id>
        <start-time>2021-03-12T14:08:26.830334+00:00</start-time>
        <end-time>2021-03-12T14:08:28.279414+00:00</end-time>
        <log-size>20775</log-size>
      </diag-trans>
    </netconf-diag-oper-data>
  </data>
</rpc-reply>
```

The following is a sample RPC to archive the collected system error messages, and the RPC response from the host:

```
#
#256
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:1dbc795c-f594-4194-a89b-fd4d88446b69">
  <archive-netconf-diag-logs xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc"/>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:1dbc795c-f594-4194-a89b-fd4d88446b69"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <log-file xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc">
    bootflash:netconf-yang-diag.20210312141009.tar.gz</log-file>


</rpc-reply>
```

The following is a sample RPC that disables NETCONF-YANG diagnostics, and the RPC response received from the host:

```
#309
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="urn:uuid:d253a313-4aec-42bc-80a2-672e9bb9ad56">
  <enable-netconf-diag xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-netconf-diag-rpc">
    <diag-level>diag-disabled</diag-level>
  </enable-netconf-diag>
</nc:rpc>

##

Received message from host
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:d253a313-4aec-42bc-80a2-672e9bb9ad56"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

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 7950	*The YANG 1.1 Data Modeling Language*
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 the NETCONF Protocol

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.
	Cisco IOS XE Gibraltar 16.12.1	In Cisco IOS XE Gibraltar 16.12.1, this feature was implemented on Cisco Catalyst 9800-L Wireless Controllers.
	Cisco IOS XE Amsterdam 17.3.1	

Feature Name	Release	Feature Information
		In Cisco IOS XE Amsterdam 17.3.1, this feature was implemented on the following platforms: • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 and 8500L Series Edge Platforms
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**.

Feature Name	Release	Feature Information
NETCONF: Candidate Configuration Commit Confirm	Cisco IOS XE Amsterdam 17.1.1	The candidate configuration supports the confirmed commit capability. This feature was implemented on the following platforms: • Cisco 1000 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 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
NETCONF-YANG SSH Server Support	Cisco IOS XE Gibraltar 16.12.1	This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series
NETCONF-SSH Algorithms	Cisco IOS XE Dublin 17.12.1	You can enable or disable the NETCONF-SSH algorithms during runtime by using Cisco IOS commands or U YANG models. This feature was implemented on the following platform: • Cisco Catalyst 9500 Series Switches

Feature Name	Release	Feature Information
Named Method List	Cisco IOS XE Cupertino 17.9.1	With the introduction of the Named Method List feature, it is possible to use a custom method-list name for authentication and authorization, without changing the existing AAA configuration of a device. Prior to this feature, only the default method-list was supported. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8200 Series Edge Platforms • Cisco Catalyst 8300 Series Edge Platforms • Cisco Catalyst 8500 Series and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800 Series Wireless Controllers • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
Side-Effect Synchronization of the Configuration Database	Cisco IOS XE Bengaluru 17.4.1	During configuration changes in the DMI, a partial synchronization of the changes that are triggered when a command or RPC is configured happens. This is called the side-effect synchronization, and it reduces the synchronization time and NETCONF downtime. This feature was implemented on the following platforms: • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 8500 and 8500L Series Edge Platforms • Cisco Catalyst 9200 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 Series Switches • Cisco Catalyst 9600 Series Switches

Feature Name	Release	Feature Information
YANG Model Version 1.1	Cisco IOS XE Cupertino 17.7.1	Cisco IOS XE Cupertino 17.7.1 uses the YANG Version 1.0; however, you can also use YANG Version 1.1. Download the YANG Version 1.1 from GitHub at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe folder. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 900 Aggregation Services Routers • Cisco ASR 920 Aggregation Services Routers • Cisco ASR 1000 Aggregation Services Routers • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco Catalyst 9600 Series Switches • Cisco Catalyst 9800-40 Wireless Controllers • Cisco Catalyst 9800-80 Wireless Controllers • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
	Cisco IOS XE Cupertino 17.8.1	Cisco IOS XE Cupertino 17.8.1 uses YANG Version 1.1. The difference between YANG Version 1.1 and Version 1.0 is documented at https://tools.ietf.org/html/rfc7950#page-10
	Cisco IOS XE Dublin 17.10.1	Cisco-defined YANG models are in YANG Version 1.1 in Cisco IOS XE Dublin 17.10.1 and later releases.
Converting IOS Commands to XML	Cisco IOS XE Cupertino 17.7.1	This feature helps to automatically translate IOS commands into relevant NETCONF-YANG XML or RESTCONF-JSON request messages. This feature is supported on all platforms that support NETCONF-YANG.