Network Configuration Protocol

Network Configuration Protocol (NETCONF) defines an XML-based interface between a network device and a network management system to provide a mechanism to manage, configure, and monitor a network device.

In Cisco IOS-XR, NMS applications use defined XML schemas to manage network devices from multiple vendors. These capabilities are supported from a Cisco IOS XR agent to a client:

- TTY NETCONF session—Logon through telnet and then enter the `netconf` command.
- SSH NETCONF session—Logon through SSH and then enter the `netconf` command.

This example shows a `<hello>` message that the agent sends to a client:

```xml
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
      urn:ietf:params:netconf:base:1.0
    </capability>
    <capability>
      urn:ietf:params:netconf:capability:candidate:1.0
    </capability>
  </capabilities>
  <session-id>4</session-id>
</hello>
```

These sections about NETCONF are covered:

- Starting a NETCONF Session, page 15-139
- Ending a NETCONF Agent Session, page 15-140
- Starting an SSH NETCONF Session, page 15-140
- Ending an SSH NETCONF Agent Session, page 15-141
- Configuring a NETCONF agent, page 15-141
- Limitations of NETCONF in Cisco IOS XR, page 15-142

**Starting a NETCONF Session**

To start a NETCONF session, enter the `netconf` command from the exec prompt (through telnet or SSH).

This example shows how to start a TTY NETCONF agent session:

```
client(/users/ore)> telnet 1.66.32.82
Trying 1.66.32.82...
Connected to 1.66.32.82.
```
Chapter 15      Network Configuration Protocol

Ending a NETCONF Agent Session

Unlike the XML agent, the client ends the session by sending a <close-session> request.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <close-session/>
</rpc>
```

The agent replies with an <ok> tag and then closes the session.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Starting an SSH NETCONF Session

This example shows how to start an SSH NETCONF agent session:

```
client(/users/ore)> ssh lab@1.66.32.82
lab@1.66.32.82's password:
RP/0/1/CPU0:gsrb#netconf echo format
<?xml version="1.0" encoding="UTF-8" ?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
</hello>
```

When a new session is created, the NETCONF agent immediately sends out a <hello> message with capabilities. At the end of each message transmission, the NETCONF agent sends the EOD marker ‘>[]]>]>]’.

The NETCONF agent does not display a prompt like the XML agent does (XML>).

The NETCONF TTY agent does not echo back the received messages and does not format returning messages by default. These capabilities can be added by using the ‘echo’ and ‘format’ options.

The client is also required to send a <hello> message with capabilities.

Ending a NETCONF Agent Session

Unlike the XML agent, the client ends the session by sending a <close-session> request.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <close-session/>
</rpc>
```

The agent replies with an <ok> tag and then closes the session.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Starting an SSH NETCONF Session

This example shows how to start an SSH NETCONF agent session:

```
client(/users/ore)> ssh lab@1.66.32.82
lab@1.66.32.82's password:
RP/0/1/CPU0:gsrb#netconf echo format
<?xml version="1.0" encoding="UTF-8" ?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
```
The client can also directly start a NETCONF session by specifying the `netconf` command on the ssh command line:

```bash
client(/users/ore)> ssh lab@1.66.32.82 netconf echo format
lab@1.66.32.82's password:
<?xml version="1.0" encoding="UTF-8" ?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
 <capability>
  urn:ietf:params:netconf:base:1.0
 </capability>
 <capability>
  urn:ietf:params:netconf:capability:candidate:1.0
 </capability>
</capabilities>
<session-id>4</session-id>
</hello>
```

### Ending an SSH NETCONF Agent Session

This example shows how to end an SSH NETCONF agent session:

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session/>
</rpc>
```

The agent replies with an `<ok>` tag and then closes the session.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
```

### Configuring a NETCONF agent

To configure a NETCONF TTY agent, use the `netconf agent tty` command.

Use the `throttle` and `session timeout` parameters as you would with the XML TTY agent.

```bash
netconf agent tty
  throttle (memory | process-rate)
  session timeout
```

To enable the NETCONF SSH agent, use this command:

```bash
ssh server v2
netconf agent tty
```
Limitations of NETCONF in Cisco IOS XR

This sections identifies the limitations of NETCONF in Cisco IOS XR Software.

Configuration Datastores

Cisco IOS XR supports these configuration datastores:

- <running>
- <candidate>

Cisco IOS XR does not support the <startup> configuration datastore.

Configuration Capabilities

Cisco IOS XR supports these configuration capabilities:

- Candidate Configuration Capability
  urn:ietf:params:netconf:capability:candidate:1.0

Cisco IOS XR does not support these configuration capabilities:

- Writable-Running Capability
  urn:ietf:params:netconf:capability:writable-running:1.0
- Confirmed Commit Capability
  urn:ietf:params:netconf:capability:confirmed-commit:1.0

Transport (RFC4741 and RFC4742)

These transport operations are supported:

- Connection-oriented operation
- Authentication
- SSH Transport—Shell based SSH. IANA-assigned TCP port <830> for NETCONF SSH is not supported.
- Other transport

Subtree Filtering (RFC4741)

NETCONF has these subtree filtering limitations in Cisco IOS XR:

- Namespace Selection—Filtering based on specified namespace. This is not supported because Cisco IOS XR does not publish schema name spaces.
- Attribute Match Expressions—Filtering is done by matching a specified attribute value. This filtering with the “Match” attribute can be specified only in Table classes. See this example:

```xml
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      ...
    </source>
  </get-config>
</rpc>
```
• Containment Nodes—Filtering is done by specifying nodes (classes) that have child nodes (classes). This filtering is by specifying container classes. See this example:

```xml
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
        <filter>
            <Configuration>
                <InterfaceConfigurationTable>
                    <InterfaceConfiguration>
                        <Naming>
                            <Active>act</Active>
                            <InterfaceName Match="GigabitEthernet.*"/>
                        </Naming>
                    </InterfaceConfiguration>
                </InterfaceConfigurationTable>
            </Configuration>
        </filter>
    </get-config>
</rpc>
```

• Selection Nodes—Filtering is done by specifying leaf nodes. This filtering specifies leaf classes. See this example:

```xml
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
        <filter>
            <Configuration>
                <InterfaceConfigurationTable/>
            </Configuration>
        </filter>
    </get-config>
</rpc>
```

• Content Match Nodes—Filtering is done by exactly matching the content of a leaf node. This filtering is done by specifying naming the class value for table classes. See this example:

```xml
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get-config>
        <source>
            <running/>
        </source>
    </get-config>
</rpc>
```
According to the RFC, a request using an empty content match node should return all <Naming> elements of all entries of the table.

For example, for this request, the response should return <Naming> elements of all the entries of <InterfaceConfigurationTable>:

```xml
<rpc message-id="106" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <Configuration>
        <InterfaceConfigurationTable>
          <InterfaceConfiguration>
            <Naming/>
          </InterfaceConfiguration>
        </InterfaceConfigurationTable>
      </Configuration>
    </filter>
  </get-config>
</rpc>
```

In Cisco IOS XR, this request is not supported and is errored out.

**Protocol Operations (RFC4741)**

These protocol operations are supported in Cisco IOS XR:

- get—Root level query that returns both the entire configuration and state data is not supported
- get-config
- edit-config
- lock
- unlock
- close-session
- commit (by the Candidate Configuration Capability)
- discard-change (by the Candidate Configuration Capability)
Event Notifications (RFC5277)

Event notifications are not supported in Cisco IOS XR.
Limitations of NETCONF in Cisco IOS XR