Contents

Configuring NETCONF

Overview

Network Configuration Protocol (NETCONF) is an XML-based network management protocol with filtering capabilities. It provides programmable mechanisms to manage and configure network devices. Through NETCONF, you can configure device parameters, retrieve parameter values, and get statistics information.

In NETCONF messages, each data item is contained in a fixed element. This enables different devices of the same vendor to provide the same access method and the same result presentation method. For the devices of different vendors, XML mapping in NETCONF messages can achieve the same effect. For a network environment containing different devices regardless of vendors, you can develop a NETCONF-based NMS system to configure and manage devices in a simple and effective way.

NETCONF structure

NETCONF has four layers: content layer, operations layer, RPC layer, and transport protocol layer.

Table 1 NETCONF layers and XML layers

NETCONF layer	XML layer	Description
Content	Configuration data, status data, and statistics information	The content layer contains a set of managed objects, which can be configuration data, status data, and statistics information. For information about the operable data, see the NETCONF XML API reference for the device.
Operations	<get>,<get-config>, <edit-config>…	The operations layer defines a set of base operations invoked as RPC methods with XML-encoded parameters. NETCONF base operations include data retrieval operations, configuration operations, lock operations, and session operations. For the device supported operations, see "Appendix A Supported NETCONF operations."
RPC	<rpc>,<rpc-reply>	The RPC layer provides a simple, transport-independent framing mechanism for encoding RPCs. The <rpc> and <rpc-reply> elements are used to enclose NETCONF requests and responses (data at the operations layer and the content layer).
Transport Protocol	• In non-FIPS mode: Console/Telnet/SSH/HTTP/HTTPS/TLS • In FIPS mode: Console/SSH/HTTPS/TLS	The transport protocol layer provides reliable, connection-oriented, serial data links. In non-FIPS mode, the following transport layer sessions are available: • **CLI sessions**—NETCONF over Telnet, NETCONF over SSH, and NETCONF over console. • **NETCONF over SOAP sessions**—NETCONF over SOAP over HTTP and NETCONF over SOAP over HTTPS. In FIPS mode, the following transport layer sessions are available: • **CLI sessions**—NETCONF over SSH and NETCONF over console. • **NETCONF over SOAP sessions**—NETCONF over SOAP over HTTPS.

NETCONF message format

NETCONF

IMPORTANT:

When configuring NETCONF in XML view, you must add end mark **]]>]]>** at the end of an XML message. Otherwise, the device cannot identify the message. Examples in this chapter do not have this end mark. Do add the end mark in actual operations.

All NETCONF messages are XML-based and comply with RFC 4741. Any incoming NETCONF messages must pass XML Schema check before it can be processed. If a NETCONF message fails XML Schema check, the device sends an error message to the client.

For information about the NETCONF operations supported by the device and the operable data, see the NETCONF XML API reference for the device.

The following example shows a NETCONF message for getting all parameters of all interfaces on the device:

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-bulk>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
        <Ifmgr>
          <Interfaces>
                 <Interface/>
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
  </get-bulk>
</rpc>
```

NETCONF over SOAP

All NETCONF over SOAP messages are XML-based and comply with RFC 4741. NETCONF messages are contained in the <Body> element of SOAP messages. NETCONF over SOAP messages also comply with the following rules:

- SOAP messages must use the SOAP Envelope namespaces.
- SOAP messages must use the SOAP Encoding namespaces.
- SOAP messages cannot contain the following information:
  - DTD reference.
  - XML processing instructions.

The following example shows a NETCONF over SOAP message for getting all parameters of all interfaces on the device:

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <auth:Authentication env:mustUnderstand="1"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
      <auth:AuthInfo>800207F0120020C</auth:AuthInfo>
    </auth:Authentication>
  </env:Header>
```

```
  <env:Body>
    <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <get-bulk>
        <filter type="subtree">
          <top xmlns="http://www.h3c.com/netconf/data:1.0">
            <Ifmgr>
              <Interfaces>
                <Interface/>
              </Interfaces>
            </Ifmgr>
          </top>
        </filter>
      </get-bulk>
    </rpc>
  </env:Body>
</env:Envelope>
```

How to use NETCONF

You can use NETCONF to manage and configure the device by using the methods in Table 2.

Table 2 NETCONF methods for configuring the device

Configuration tool	Login method	Remarks
CLI	• Console port • SSH • Telnet	To implement NETCONF operations, copy valid NETCONF messages to the CLI in XML view.
Custom interface	N/A	To use this method, you must enable NETCONF over SOAP to encode the NETCONF messages sent from a custom interface in SOAP.

Protocols and standards

- RFC 3339, *Date and Time on the Internet: Timestamps*
- RFC 4741, *NETCONF Configuration Protocol*
- RFC 4742, *Using the NETCONF Configuration Protocol over Secure SHell (SSH)*
- RFC 4743, *Using NETCONF over the Simple Object Access Protocol (SOAP)*
- RFC 5277, *NETCONF Event Notifications*
- RFC 5381, *Experience of Implementing NETCONF over SOAP*
- RFC 5539, *NETCONF over Transport Layer Security (TLS)*
- RFC 6241, *Network Configuration Protocol*

FIPS compliance

The device supports the FIPS mode that complies with NIST FIPS 140-2 requirements. Support for features, commands, and parameters might differ in FIPS mode (see *Security Configuration Guide*) and non-FIPS mode.

NETCONF configuration task list

Tasks at a glance
(Optional.) Configuring NETCONF over SOAP
(Optional.) Enabling NETCONF over SSH
(Optional.) Enabling NETCONF logging
(Optional.) Configuring NETCONF to use module-specific namespaces
(Required.) Establishing a NETCONF session
(Optional.) Subscribing to event notifications
(Optional.) Locking/unlocking the configuration
(Optional.) Performing the <get>/<get-bulk> operation
(Optional.) Performing the <get-config>/<get-bulk-config> operation
(Optional.) Performing the <edit-config> operation
(Optional.) Saving, rolling back, and loading the configuration
(Optional.) Enabling preprovisioning
(Optional.) Filtering data
(Optional.) Performing CLI operations through NETCONF
(Optional.) Retrieving NETCONF information
(Optional.) Retrieving YANG file content
(Optional.) Retrieving NETCONF session information
(Optional.) Terminating another NETCONF session
(Optional.) Returning to the CLI

Configuring NETCONF over SOAP

NETCONF over SOAP encapsulates NETCONF messages into SOAP messages and transmits the SOAP messages over HTTP or HTTPS. You can use a custom user interface to establish a NETCONF over SOAP session to the device and perform NETCONF operations.

To configure NETCONF over SOAP:

Step	Command	Remark
1. Enter system view.	**system-view**	N/A
2. Enable NETCONF over SOAP.	• Enable NETCONF over SOAP over HTTP (not available in FIPS mode): **netconf soap http enable** • Enable NETCONF over SOAP over HTTPS: **netconf soap https enable**	By default, the NETCONF over SOAP feature is disabled.
3. Set the DSCP value for NETCONF over SOAP packets.	• Set the DSCP value for NETCONF over SOAP over HTTP packets:	By default, the DSCP value is 0 for NETCONF over SOAP packets.

Step	Command	Remark
	netconf soap http dscp *dscp-value* • Set the DSCP value for NETCONF over SOAP over HTTPs packets: **netconf soap https dscp** *dscp-value*	
4. Apply an ACL to control NETCONF over SOAP access.	• Apply an ACL to control NETCONF over SOAP over HTTP access (not available in FIPS mode): **netconf soap http acl** { *acl-number* | **name** *acl-name* } • Apply an ACL to control NETCONF over SOAP over HTTPS access: **netconf soap https acl** { *acl-number* | **name** *acl-name* }	By default, no ACL is applied to control NETCONF over SOAP access.
5. Specify a mandatory authentication domain for NETCONF users.	**netconf soap domain** *domain-name*	By default, no mandatory authentication domain is specified for NETCONF users. For information about authentication domains, see *Security Configuration Guide*.

Enabling NETCONF over SSH

This feature allows users to use a client to perform NETCONF operations on the device through a NETCONF over SSH connection.

To enable NETCONF over SSH:

Step	Command	Remark
1. Enter system view.	**system-view**	N/A
2. Enable NETCONF over SSH.	**netconf ssh server enable**	By default, NETCONF over SSH is disabled.
3. Specify a port to listen for NETCONF over SSH connections.	**netconf ssh server port** *port-number*	By default, port 830 listens for NETCONF over SSH connections.

Enabling NETCONF logging

NETCONF logging generates logs for different NETCONF operation sources and NETCONF operations.

To enable NETCONF logging:

Step	Command	Remarks
1. Enter system view.	**system-view**	N/A
2. Enable NETCONF logging.	**netconf log source** { **all** | { **agent** | **soap** } * } { **protocol-operation**	By default, NETCONF logging is disabled.

Step	Command	Remarks
	{ **all** | { **action** | **config** | **get** | **set** | **session** | **syntax** | **others** } * } **|** **row-operation** | **verbose** }	

Configuring NETCONF to use module-specific namespaces

Overview

NETCONF supports the following types of namespaces:

- **Common namespace**—The common namespace is shared by all modules. In a packet that uses the common namespace, the namespace is indicated in the <top> element, and the modules are listed under the <top> element.

  Example:

```
<rpc message-id="100" xmlns="urn:ietf:Params:xml:ns:netconf:base:1.0">
   <get-bulk>
      <filter type="subtree">
         <top xmlns="http://www.h3c.com/netconf/data:1.0">
            <Ifmgr>
               <Interfaces>
               </Interfaces>
            </Ifmgr>
         </top>
      </filter>
   </get-bulk>
</rpc>
```

- **Module-specific namespace**—Each module has its own namespace. A packet that uses a module-specific namespace does not have the <top> element. The namespace follows the module name.

  Example:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-bulk>
      <filter type="subtree">
            <Ifmgr xmlns="http://www.h3c.com/netconf/data:1.0-Ifmgr">
               <Interfaces>
               </Interfaces>
            </Ifmgr>
      </filter>
   </get-bulk>
</rpc>
```

The common namespace is incompatible with module-specific namespaces. To set up a NETCONF session, the device and the client must use the same type of namespaces. By default, the common namespace is used. If the client does not support the common namespace, use this feature to configure the device to use module-specific namespaces.

Configuration restrictions and guidelines

For this feature to take effect, you must re-establish the NETCONF session.

Configuration procedure

To configure NETCONF to use module-specific namespaces:

Step	Command	Remark
1. Enter system view.	**system-view**	N/A
2. Configure NETCONF to use module-specific namespaces.	**netconf capability specific-namespace**	By default, the common namespace is used.

Establishing a NETCONF session

After a NETCONF session is established, the device automatically sends its capabilities to the client. You must send the capabilities of the client to the device before you can perform any other NETCONF operations.

You can use the **aaa session-limit** command to set the maximum number of NETCONF sessions that the device can support. If the upper limit is reached, new NETCONF users cannot access the device. For information about this command, see *Security Configuration Guide*.

Before performing a NETCONF operation, make sure no other users are configuring or managing the device. If multiple users simultaneously configure or manage the device, the result might be different from what you expect.

Setting the NETCONF session idle timeout time

If no NETCONF packets are exchanged on a NETCONF session within the NETCONF session idle timeout time, the device tears down the session.

To set the NETCONF session idle timeout time:

Task	Command	Remarks
1. Enter system view.	**system-view**	N/A
2. Set the NETCONF session idle timeout time.	**netconf** { **soap** | **agent** } **idle-timeout** *minute*	By default, the NETCONF session idle timeout time is as follows: • 10 minutes for NETCONF over SOAP over HTTP sessions and NETCONF over SOAP over HTTPS sessions. • 0 minutes for NETCONF over SSH sessions, NETCONF over Telnet sessions, and NETCONF over console sessions. The sessions never time out.

Entering XML view

Task	Command	Remarks
Enter XML view.	**xml**	Available in user view.

Exchanging capabilities

After you enter XML view, the client and the device exchange their capabilities before you can perform subsequent operations. The device automatically advertises its NETCONF capabilities to the client in a hello message as follows:

```
<?xml version="1.0" encoding="UTF-8"?><hello
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><capabilities><capability>urn:ietf:pa
rams:netconf:base:1.1</capability><capability>urn:ietf:params:netconf:writable-runnin
g</capability><capability>urn:ietf:params:netconf:capability:notification:1.0</capabi
lity><capability>urn:ietf:params:netconf:capability:validate:1.1</capability><capabil
ity>urn:ietf:params:netconf:capability:interleave:1.0</capability><capability>urn:h3c
:params:netconf:capability:h3c-netconf-ext:1.0</capability></capabilities><session-id
>1</session-id></hello>]]>]]>
```

The <capabilities> parameter represents the capabilities supported by the device. The supported capabilities vary by device model.

The <session-id> parameter represents the unique ID assigned to the current session.

After receiving the hello message from the device, copy the following message to notify the device of the capabilities (user-configurable) supported by the client:

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

Use a pair of <capability> and </capability> tags to enclose each capability set.

Subscribing to event notifications

Overview

After you subscribe to event notifications, the device sends event notifications to the NETCONF client when a subscribed event takes place on the device.

A subscription takes effect only on the current session. If the session is terminated, the subscription is automatically canceled.

If you do not configure the event stream, the device sends syslog event notifications to the NETCONF client.

Subscribing to syslog events

Copy the following message to the client to complete the subscription:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription  xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <stream>NETCONF</stream>
      <filter>
        <event xmlns="http://www.h3c.com/netconf/event:1.0">
          <Code>code</Code>
            <Group>group</Group>
              <Severity>severity</Severity>
        </event>
      </filter>
      <startTime>start-time</startTime>
      <stopTime>stop-time</stopTime>
   </create-subscription>
</rpc>
```

The <stream> parameter represents the subscribed event stream. The name for the syslog event stream is **NETCONF**.

The <event> parameter represents a log event to which you subscribe. For information about which event notifications you can subscribe to, see the system log messages reference for the device.

The <code> parameter represents a mnemonic symbol.

The <group> parameter represents the module name.

The <severity> parameter represents the severity level of the event.

The <start-time> parameter represents the start time of the subscription.

The <stop-time> parameter represents the end time of the subscription.

After receiving the subscription request from the client, the device returns a response in the following format if the subscription is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
```

If the subscription fails, the device returns an error message in the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<rpc-error>
   <error-type>error-type</error-type>
   <error-tag>error-tag</error-tag>
   <error-severity>error-severity</error-severity>
   <error-message xml:lang="en">error-message</error-message>
</rpc-error>
</rpc-reply>
```

For more information about error messages, see RFC 4741.

Subscribing to events monitored by NETCONF

Copy the following message to the client to complete the subscription:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
<create-subscription xmlns ='urn:ietf:params:xml:ns:netconf:notification:1.0'>
  <stream>NETCONF_MONITOR_EXTENSION</stream>
  <filter>
    <NetconfMonitor xmlns='http://www.h3c.com/netconf/monitor:1.0'>
      <XPath>XPath</XPath>
      <Interval>interval</Interval>
      <ColumnConditions>
        <ColumnCondition>
          <ColumnName>ColumnName</ColumnName>
          <ColumnValue>ColumnValue</ColumnValue>
          <ColumnCondition>ColumnCondition</ColumnCondition>
        </ColumnCondition>
      </ColumnConditions>
      <MustIncludeResultColumns>
        <ColumnName>columnName</ColumnName>
      </MustIncludeResultColumns>
    </NetconfMonitor>
  </filter>
<startTime>start-time</startTime>
<stopTime>stop-time</stopTime>
</create-subscription>
</rpc>
```

The <stream> parameter represents the event stream. The name for the monitoring event stream is **NETCONF_MONITOR_EXTENSION**.

NetconfMonitor represents the filtering information for the monitoring event.

The <XPath> parameter represents the path of a monitoring event in the format of *ModuleName*[/*SubmoduleName*]/*TableName*.

The <interval> parameter represents the interval for the device to obtain information that matches the subscription condition. The value range is 1 to 4294967 seconds. The default value is 300 seconds.

The <ColumnName> parameter represents the name of the monitoring column in the format of [*GroupName*.]*ColumnName*.

The <ColumnValue> parameter represents the baseline value.

The <ColumnCondition> parameter represents the filter condition. Table 3 displays the options. Choose the filter condition according to the type of the baseline value.

Table 3 Filter condition options

Operation	Remarks
more	More than the specified filter value.
less	Less than the specified baseline value.
notLess	Not less than the specified baseline value.
notMore	Not more than the specified baseline value.
equal	Equal to the specified baseline value.
notEqual	Not equal to the specified baseline value.
include	Including the specified baseline value.

Operation	Remarks
exclude	Excluding the specified baseline value.
startWith	Starting with the specified baseline value.
endWith	Ending with the specified baseline value.

The <start-time> parameter represents the start time of the subscription.

The <stop-time> parameter represents the end time of the subscription.

After receiving the subscription request from the client, the device returns a response in the following format if the subscription is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
```

Subscribing to events reported by modules

Copy the following message to the client to complete the subscription:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xs="http://www.h3c.com/netconf/base:1.0">
<create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
<stream>XXX_STREAM</stream>
   <filter type="subtree">
<event xmlns="http://www.h3c.com/netconf/event:1.0/xxx-features-list-name:1.0">
       <ColumnName xs:condition="Condition">value</ColumnName>
</event>
</filter>
<startTime>start-time</startTime>
<stopTime>stop-time</stopTime>
</create-subscription>
</rpc>
```

The <stream> parameter represents the event stream. The name for the module notification event stream varies by actual condition.

The event parameter represents the event name. An event stream includes multiple events. The namespace of the event is the namespace of the event stream.

The ColumnName parameter represents the name of the column to be filtered in the current event.

The Condition parameter represents the filter condition. The options are described in Table 3. Choose the filter condition according to the type of the baseline value.

The value parameter represents the baseline value for the column.

The <start-time> parameter represents the start time of the subscription.

The <stop-time> parameter represents the end time of the subscription.

After receiving the subscription request from the client, the device returns a response in the following format if the subscription is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
```

```
</rpc-reply>
```

Example for subscribing to event notifications

Network requirements

Configure a client to subscribe to all events with no time limitation. After the subscription is successful, all events on the device are sent to the client before the session between the device and client is terminated.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
            urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Subscribe to all events with no time limitation.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <create-subscription xmlns ="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <stream>NETCONF</stream>
  </create-subscription>
</rpc>
```

Verifying the configuration

If the client receives the following response, the subscription is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
    <ok/>
</rpc-reply>
```

When another client (192.168.100.130) logs in to the device, the device sends a notification to the client that has subscribed to all events:

```
<?xml version="1.0" encoding="UTF-8"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2011-01-04T12:30:52</eventTime>
    <event xmlns="http://www.h3c.com/netconf/event:1.0">
        <Group>SHELL</Group>
        <Code>SHELL_LOGIN</Code>
        <Slot>6</Slot>
        <Severity>Notification</Severity>
        <context>VTY logged in from 192.168.100.130.</context>
    </event>
</notification>
```

Locking/unlocking the configuration

You can use multiple methods such as NETCONF, CLI, and SNMP to configure the device. During device configuration and maintenance or network troubleshooting, a user can lock the configuration to prevent other users from changing it. After that, only the user holding the lock can change the configuration, and other users can only read the configuration.

The <lock> operation locks only configuration data that can be changed by <edit-config> operations. Other configuration data are not limited by the <lock> operation.

In addition, only the user holding the lock can release the lock. After the lock is released, other users can change the current configuration or lock the configuration. If the session of the user that holds the lock is terminated, the system automatically releases the lock.

Locking the configuration

Copy the following text to the client to lock the configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <lock>
      <target>
        <running/>
      </target>
    </lock>
  </rpc>
```

After receiving the lock request, the device returns a response in the following format if the <lock> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ok/>
</rpc-reply>
```

Unlocking the configuration

Copy the following text to the client to unlock the configuration:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <unlock>
      <target>
        <running/>
      </target>
    </unlock>
  </rpc>
```

After receiving the unlock request, the device returns a response in the following format if the <unlock> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

Example for locking the configuration

Network requirements

Lock the device configuration so that other users cannot change the device configuration.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>
            urn:ietf:params:netconf:base:1.0
        </capability>
    </capabilities>
</hello>
```

Lock the configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <lock>
      <target>
        <running/>
      </target>
    </lock>
  </rpc>
```

Verifying the configuration

If the client receives the following response, the <lock> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

If another client sends a lock request, the device returns the following response:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<rpc-error>
  <error-type>protocol</error-type>
  <error-tag>lock-denied</error-tag>
  <error-severity>error</error-severity>
  <error-message xml:lang="en"> Lock failed because the NETCONF lock is held by another
session.</error-message>
  <error-info>
    <session-id>1</session-id>
  </error-info>
  </rpc-error>
</rpc-reply>
```

The output shows that the <lock> operation failed because the client with session ID 1 held the lock, and only the client holding the lock can release the lock.

Performing service operations

You can use NETCONF to perform service operations on the device, such as retrieving and modifying the specified information. The basic operations include <get>, <get-bulk>, <get-config>, <get-bulk-config>, and <edit-config>, which are used to retrieve all data, retrieve configuration data, and edit the data of the specified module. For more information, see the NETCONF XML API reference for the device.

NOTE:

During a <get>, <get-bulk>, <get-config>, or <get-bulk-config> operation, NETCONF replaces unidentifiable characters in the retrieved data with question marks (?) before sending the data to the client.

Performing the <get>/<get-bulk> operation

The <get> operation is used to retrieve device configuration and state information that match the conditions. In some cases, this operation leads to inefficiency.

The <get-bulk> operation is used to retrieve a number of data entries starting from the data entry next to the one with the specified index. One data entry contains a device configuration entry and a state information entry. The data entry quantity is defined by the **count** attribute, and the index is specified by the **index** attribute. The returned output does not include the index information. If you do not specify the **index** attribute, the index value starts with 1 by default.

The <get-bulk> operation retrieves all the rest data entries starting from the data entry next to the one with the specified index if either of the following conditions occurs:

- You do not specify the **count** attribute.
- The number of matching data entries is less than the value of the **count** attribute.

Copy the following text to the client to perform the <get> operation:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <getoperation>
    <filter>
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
          Specify the module, submodule, table name, and column name
      </top>
    </filter>
  </getoperation>
</rpc>
```

The <getoperation> parameter can be <get> or <get-bulk>. The <filter> element is used to filter data, and it can contain module name, submodule name, table name, and column name.

- If the module name and the submodule name are not provided, the operation retrieves the data for all modules and submodules. If a module name or a submodule name is provided, the operation retrieves the data for the specified module or submodule.
- If the table name is not provided, the operation retrieves the data for all tables. If a table name is provided, the operation retrieves the data for the specified table.
- If only the index column is provided, the operation retrieves the data for all columns. If the index column and other columns are provided, the operation retrieves the data for the index column and the specified columns.

The <get> and <get-bulk> messages are similar. A <get-bulk> message carries the **count** and **index** attributes. The following is a <get-bulk> message example:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="http://www.h3c.com/netconf/base:1.0">
  <get-bulk>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0"
xmlns:base="http://www.h3c.com/netconf/base:1.0">
        <Syslog>
          <Logs xc:count="5">
            <Log>
              <Index>10</Index>
           </Log>
             </Logs>
        </Syslog>
      </top>
    </filter>
  </get-bulk>
</rpc>
```

The **count** attribute complies with the following rules:

- The **count** attribute can be placed in the module node and table node. In other nodes, it cannot be resolved.
- When the **count** attribute is placed in the module node, a descendant node inherits this **count** attribute if the descendant node does not contain the **count** attribute.

When retrieving interface information, the device cannot identify whether an integer value for the IfIndex element represents an interface name or index. When retrieving VRF information, the device cannot identify whether an integer value for the vrfindex element represents a VPN name or index. To resolve the issue, you can use the **valuetype** attribute to specify the value type.

The **valuetype** attribute has the following values:

- **name**—The IfIndex or vrfindex element is carrying a name.
- **index**—The IfIndex or vrfindex element is carrying an index.
- **auto**—Default value. The device uses the value of the IfIndex or vrfindex element as a name for interface or VRF matching. If no match is found, the device uses the value as an index for interface or VRF matching.

The following example specifies an index-type value for the IfIndex element:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <getoperation>
    <filter>
      <top xmlns="http://www.h3c.com/netconf/config:1.0"
xmlns:base=”http://www.h3c.com/netconf/base:1.0”>
       <VLAN>
        <TrunkInterfaces>
          <Interface>
            <IfIndex base:valuetype="index">1</IfIndex>
          </Interface>
        </TrunkInterfaces>
       </VLAN>
      </top>
    </filter >
```

```
  </getoperation>
</rpc>
```

Verifying the configuration

After receiving the get-bulk request, the device returns a response in the following format if the operation is successful:

```
<?xml version="1.0"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
     Device state and configuration data
  </data>
</rpc-reply>
```

Performing the <get-config>/<get-bulk-config> operation

The <get-config> and <get-bulk-config> operations are used to retrieve all non-default settings, which are configured by using the CLI or MIB. The <get-config> and <get-bulk-config> messages can contain the <filter> element for filtering data.

The <get-config> and <get-bulk-config> messages are similar. The following is a <get-config> message example:

```
<?xml version="1.0"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter>
      <top xmlns="http://www.h3c.com/netconf/config:1.0">
          Specify the module name, submodule name, table name, and column name
      </top>
    </filter>
  </get-config>
</rpc>
```

Verifying the configuration

After receiving the get-config request, the device returns a response in the following format if the operation is successful:

```
<?xml version="1.0"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    All data matching the specified filter
  </data>
</rpc-reply>
```

Performing the <edit-config> operation

The <edit-config> operation supports the following operation attributes: **merge**, **create**, **replace**, **remove**, **delete**, **default-operation**, **error-option**, **test-option**, and **incremental**. For more information about these attributes, see "Appendix A Supported NETCONF operations."

Copy the following text to perform the <edit-config> operation:

```
<?xml version="1.0"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target><running></running></target>
    <error-option>
       error-option
    </error-option>
    <config>
      <top xmlns="http://www.h3c.com/netconf/config:1.0">
        Specify the module name, submodule name, table name, and column name
      </top>
    </config>
  </edit-config>
</rpc>
```

The <error-option> element indicates the action to be taken in response to an error that occurs during the operation. It has the following values:

- **stop-on-error**—Stops the operation.
- **continue-on-error**—Continues the operation.
- **rollback-on-error**—Rolls back the configuration to the configuration before the <edit-config> operation was performed.

  By default, an <edit-config> operation cannot be performed while the device is rolling back the configuration. If the rollback time exceeds the maximum time that the client can wait, the client determines that the <edit-config> operation has failed and performs the operation again. Because the previous rollback is not completed, the operation triggers another rollback. If this process repeats itself, CPU and memory resources will be exhausted and the device will reboot.

  To allow an <edit-config> operation to be performed while the device is rolling back the configuration, perform an <action> operation to change the value of the **DisableEditConfigWhenRollback** attribute to **false**.

After receiving the edit-config request, the device returns a response in the following format if the operation is successful:

```
<?xml version="1.0">
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
```

Perform the <get> operation to verify that the current value of the parameter is the same as the value specified through the <edit-config> operation. (Details not shown.)

All-module configuration data retrieval example

Network requirements

Retrieve configuration data for all modules.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
```

```
    <capability>
            urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Retrieve configuration data for all modules.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
  </get-config>
</rpc>
```

Verifying the configuration

If the client receives the following text, the <get-config> operation is successful:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:web="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
    <data>
        <top xmlns="http://www.h3c.com/netconf/config:1.0">
            <Ifmgr>
                <Interfaces>
                    <Interface>
                        <IfIndex>1307</IfIndex>
                        <Shutdown>1</Shutdown>
                    </Interface>
                    <Interface>
                        <IfIndex>1308</IfIndex>
                        <Shutdown>1</Shutdown>
                    </Interface>
                    <Interface>
                        <IfIndex>1309</IfIndex>
                        <Shutdown>1</Shutdown>
                    </Interface>
                    <Interface>
                        <IfIndex>1311</IfIndex>

                            <VlanType>2</VlanType>

                    </Interface>
                    <Interface>
                        <IfIndex>1313</IfIndex>

                            <VlanType>2</VlanType>

                    </Interface>
                </Interfaces>
            </Ifmgr>
            <Syslog>
```

```
                <LogBuffer>
                    <BufferSize>120</BufferSize>
                </LogBuffer>
            </Syslog>
            <System>
                <Device>
                    <SysName>H3C</SysName>
                    <TimeZone>
                        <Zone>+11:44</Zone>
                        <ZoneName>beijing</ZoneName>
                    </TimeZone>
                </Device>
            </System>
        </top>
    </data>
</rpc-reply>
```

Syslog configuration data retrieval example

Network requirements

Retrieve configuration data for the Syslog module.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
            urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Retrieve configuration data for the Syslog module.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/config:1.0">
        <Syslog/>
      </top>
    </filter>
  </get-config>
</rpc>
```

Verifying the configuration

If the client receives the following text, the <get-config> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
    <data>
        <top xmlns="http://www.h3c.com/netconf/config:1.0">
            <Syslog>
                <LogBuffer>
                    <BufferSize>120</BufferSize>
                </LogBuffer>
            </Syslog>
        </top>
    </data>
</rpc-reply>
```

Example for retrieving a data entry for the interface table

Network requirements

Retrieve a data entry for the interface table.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
```

Retrieve a data entry for the interface table.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:web="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-bulk>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0"
xmlns:web="http://www.h3c.com/netconf/base:1.0">
        <Ifmgr>
          <Interfaces web:count="1">
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
  </get-bulk>
</rpc>
```

Verifying the configuration

If the client receives the following text, the <get-bulk> operation is successful:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:web="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
```

```
  <data>
    <top xmlns="http://www.h3c.com/netconf/data:1.0">
      <Ifmgr>
        <Interfaces>
          <Interface>
            <IfIndex>3</IfIndex>
            <Name>Ten-GigabitEthernet1/0/2</Name>
            <AbbreviatedName>XGE1/0/2</AbbreviatedName>
            <PortIndex>3</PortIndex>
            <ifTypeExt>22</ifTypeExt>
            <ifType>6</ifType>
            <Description>Ten-GigabitEthernet1/0/2 Interface</Description>
            <AdminStatus>2</AdminStatus>
            <OperStatus>2</OperStatus>
            <ConfigSpeed>0</ConfigSpeed>
            <ActualSpeed>100000</ActualSpeed>
            <ConfigDuplex>3</ConfigDuplex>
            <ActualDuplex>1</ActualDuplex>
          </Interface>
        </Interfaces>
      </Ifmgr>
    </top>
  </data>
</rpc-reply>
```

Example for changing the value of a parameter

Network requirements

Change the log buffer size for the Syslog module to 512.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>
```

Change the log buffer size for the Syslog module to 512.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:web="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <top xmlns="http://www.h3c.com/netconf/config:1.0" web:operation="merge">
```

```
        <Syslog>
          <LogBuffer>
            <BufferSize>512</BufferSize>
          </LogBuffer>
        </Syslog>
      </top>
    </config>
  </edit-config>
</rpc>
```

Verifying the configuration

If the client receives the following text, the <edit-config> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
```

Saving, rolling back, and loading the configuration

Use NETCONF to save, roll back, or load the configuration.

Performing the <save>, <rollback>, or <load> operation consumes a lot of system resources. Do not perform these operations when the system resources are heavily occupied.

By default, an <edit-config> operation cannot be performed while the device is rolling back the configuration. To allow an <edit-config> operation to be performed while the device is rolling back the configuration, perform an <action> operation to change the value of the **DisableEditConfigWhenRollback** attribute to **false**.

Saving the configuration

Copy the following text to the client to save the device configuration to the specified file:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save OverWrite="false" Binary-only="false">
    <file>Configuration file name</file>
  </save>
</rpc>
```

Item	Description
file	Specifies a .cfg configuration file by its name. The name must start with the storage medium name. If you specify the file column, a file name is required. If the **Binary-only** attribute is **false**, the device saves the running configuration to both the text and binary configuration files. • If the specified .cfg file does not exist, the device creates the binary and text configuration files to save the running configuration. • If you do not specify the file column, the device saves the running configuration to the text and binary next-startup configuration files.
OverWrite	Determines whether to overwrite the specified file if the file already exists. The following values are available: • **true**—Overwrite the file.

Item	Description
	• **false**—Do not overwrite the file. The running configuration cannot be saved, and the system displays an error message. The default value is **true**.

Binary-only	Determines whether to save the running configuration only to the binary configuration file. The following values are available:
	• **true**—Save the running configuration only to the binary configuration file.
	  ◦ If **file** specifies a nonexistent file, the <save> operation fails.
	  ◦ If you do not specify the file column, the device identifies whether the main next-startup configuration file is specified. If yes, the device saves the running configuration to the corresponding binary file. If not, the <save> operation fails.
	• **false**—Save the running configuration to both the text and binary configuration files. For more information, see the description for the file column in this table.
	Saving the running configuration to both the text and binary configuration files requires more time.
	The default value is **false**.

After receiving the save request, the device returns a response in the following format if the <save> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

Rolling back the configuration based on a configuration file

Copy the following text to the client to roll back the configuration based on a configuration file:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <rollback>
    <file>Specify the configuration file name</file>
  </rollback>
</rpc>
```

After receiving the rollback request, the device returns a response in the following format if the <rollback> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

Rolling back the configuration based on a rollback point

You can roll back the running configuration based on a rollback point when one of the following situations occurs:

- A NETCONF client sends a rollback request.
- The NETCONF session idle time is longer than the rollback idle timeout time.
- A NETCONF client is unexpectedly disconnected from the device.

To roll back the configuration based on a rollback point, perform the following tasks:

1. Lock the system.

   Multiple users might simultaneously use NETCONF to configure the device. As a best practice, lock the system before rolling back the configuration to prevent other users from modifying the running configuration.
2. Mark the beginning of a <rollback> operation. For more information, see "Performing the <save-point/begin> operation."
3. Edit the device configuration. For more information, see "Performing the <edit-config> operation."
4. Configure the rollback point. For more information, see "Performing the <save-point/commit> operation."

   You can repeat this step to configure multiple rollback points.
5. Roll back the configuration based on the rollback point. For more information, see "Performing the <save-point/rollback> operation."

   The configuration can also be automatically rolled back based on the most recently configured rollback point when the NETCONF session idle time is longer than the rollback idle timeout time.
6. End the rollback configuration. For more information, see "Performing the <save-point/end> operation."
7. Release the lock.

Performing the <save-point/begin> operation

Copy the following text to the client to mark the beginning of a <rollback> operation based on a rollback point:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <save-point>
        <begin>
          <confirm-timeout>100</confirm-timeout>
       </begin>
      </save-point>
</rpc>
```

The <confirm-timeout> parameter specifies the rollback idle timeout time in the range of 1 to 65535 seconds (the default is 600 seconds). This parameter is optional.

After receiving the begin request, the device returns a response in the following format if the <begin> operation is successful:

```
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <save-point>
       <commit>
          <commit-id>1</commit-id>
       </commit>
    </save-point>
  </data>
</rpc-reply>
```

Performing the <save-point/commit> operation

The system supports a maximum of 50 rollback points. When the limit is reached, you must specify the **force** attribute to overwrite the earliest rollback point.

Copy the following text to the client to configure the rollback point:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
```

```
    <commit>
      <label>SUPPORT VLAN<label>
      <comment>vlan 1 to 100 and interfaces.</comment>
     </commit>
  </save-point>
</rpc>
```

The <label> and <comment> parameters are optional.

After receiving the commit request, the device returns a response in the following format if the <commit> operation is successful:

```
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <save-point>
       <commit>
          <commit-id>2</commit-id>
       </commit>
    </save-point>
  </data>
</rpc-reply>
```

Performing the <save-point/rollback> operation

Copy the following text to the client to roll back the configuration:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
    <rollback>
      <commit-id/>
      <commit-index/>
      <commit-label/>
    </rollback>
  </save-point>
</rpc>
```

The <commit-id> parameter uniquely identifies a rollback point.

The <commit-index> parameter specifies 50 most recently configured rollback points. The value of 0 indicates the most recently configured one and 49 indicates the earliest configured one.

The <commit-label> parameter exclusively specifies a label for a rollback point. The label is not required for a rollback point.

Specify one of these parameters to roll back the specified configuration. If no parameter is specified, this operation rolls back configuration based on the most recently configured rollback point.

After receiving the rollback request, the device returns a response in the following format if the <rollback> operation is successful:

```
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok></ok>
</rpc-reply>
```

Performing the <save-point/end> operation

Copy the following text to the client to end the rollback configuration:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
    <end/>
  </save-point>
</rpc>
```

After receiving the end request, the device returns a response in the following format if the <end> operation is successful:

```
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

Performing the <save-point/get-commits> operation

Copy the following text to the client to get the rollback point configuration records:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
    <get-commits>
      <commit-id/>
      <commit-index/>
      <commit-label/>
    </get-commits>
  </save-point>
</rpc>
```

Specify one of the <commit-id>, <commit-index>, and <commit-label> parameters to get the specified rollback point configuration records. If no parameter is specified, this operation gets records for all rollback point settings. The following text is a <save-point>/<get-commits> request example:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
    <get-commits>
      <commit-label>SUPPORT VLAN</commit-label>
    </get-commits>
  </save-point>
</rpc>
```

After receiving the get commits request, the device returns a response in the following format if the <get commits> operation is successful:

```
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <save-point>
         <commit-information>
           <CommitID>2</CommitID>
           <TimeStamp>Thu Oct 30 11:30:28 1980</TimeStamp>
           <UserName>test</UserName>
           <Label>SUPPORT VLAN</Label>
         </commit-information>
    </save-point>
  </data>
</rpc-reply>
```

Performing the <save-point/get-commit-information> operation

Copy the following text to the client to get the system configuration data corresponding to a rollback point:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
    <get-commit-information>
      <commit-information>
        <commit-id/>
        <commit-index/>
```

```
        <commit-label/>
      </commit-information>
      <compare-information>
        <commit-id/>
        <commit-index/>
        <commit-label/>
      </compare-information>
    </get-commit-information>
  </save-point>
</rpc>
```

Specify one of the <commit-id>, <commit-index>, and <commit-label> parameters to get the configuration data corresponding to the specified rollback point. The <compare-information> parameter is optional. If no parameter is specified, this operation gets the configuration data corresponding to the most recently configured rollback point. The following text is a <save-point>/<get-commit-information> request example:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save-point>
    <get-commit-information>
              <commit-information>
                 <commit-label>SUPPORT VLAN</commit-label>
              </commit-information>
    </get-commit-information>
  </save-point>
</rpc>
```

After receiving the get-commit-information request, the device returns a response in the following format if the <get-commit-information> operation is successful:

```
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
     <save-point>
        <commit-information>
           <content>
              …
              interface vlan 1
              …
           </content>
        </commit-information>
     </save-point>
   </data>
</rpc-reply>
```

Loading the configuration

After you perform the <load> operation, the loaded settings are merged into the current configuration as follows:

- New settings are directly loaded.
- Settings that already exist in the current configuration are replaced by those loaded from the configuration file.

Some settings in a configuration file might conflict with the existing settings. For the settings in the file to take effect, delete the existing conflicting settings, and then load the configuration file.

Copy the following text to the client to load a configuration file for the device:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <load>
   <file>Specify the configuration file name</file>
  </load>
</rpc>
```

The name of the specified configuration file must start with the storage media name and end with the **.cfg** extension.

After receiving the load request, the device returns a response in the following format if the <load> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

Example for saving the configuration

Network requirements

Save the current configuration to configuration file **my_config.cfg**.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
           urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Save the configuration of the device to configuration file **my_config.cfg**.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <save>
    <file>my_config.cfg</file>
  </save>
</rpc>
```

Verifying the configuration

If the client receives the following response, the <save> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

Enabling preprovisioning

The <config-provisioned> operation enables preprovisioning.

- With preprovisioning disabled, the configuration for a member device or subcard is lost if the following sequence of events occurs:

  a. The member device leaves the IRF fabric or the subcard goes offline.

  b. You save the running configuration and reboot the IRF fabric.

  If the member device joins the IRF fabric or the subcard comes online again, you must reconfigure the member device or subcard.

- With preprovisioning enabled, you can view and modify the configuration for a member device or subcard after the member device leaves the IRF fabric or the subcard goes offline. If you save the running configuration and reboot the IRF fabric, the configuration for the member device or subcard is still retained. If the member device joins the IRF fabric or the subcard comes online again, the system applies the retained configuration to the member device or subcard. You do not need to reconfigure the member device or subcard.

To view or modify the configuration for an offline member device or subcard, you can use only CLI commands.

Only the following commands support preprovisioning:

- Commands in the interface view of a member device or subcard.
- Commands in slot view.
- Command **qos traffic-counter**.

Only member devices and subcards in Normal state support preprovisioning.

Copy the following text to the client to enable preprovisioning:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <config-provisioned>
  </config-provisioned>
</rpc>
```

The device returns a response in the following format if preprovisioning is successfully enabled:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
```

Filtering data

You can define a filter to filter information when you perform a <get>, <get-bulk>, <get-config>, or <get-bulk-config> operation. Data filtering includes the following types:

- **Table-based filtering**—Filters table information.
- **Column-based filtering**—Filters information for a single column.

For table-based filtering to take effect, you must configure table-based filtering before column-based filtering.

Table-based filtering

You can specify a match criterion for the row attribute **filter** to implement table-based filtering, for example, IP address filtering. The namespace is **http://www.h3c.com/netconf/base:1.0**. For information about the support for table-based match, see NETCONF XML API documents.

Copy the following text to the client to retrieve the longest data with IP address **1.1.1.0** and mask length **24** from the IPv4 routing table:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
        <Route>
         <Ipv4Routes>
           <RouteEntry h3c:filter="IP 1.1.1.0 MaskLen 24 longer"/>
         </Ipv4Routes>
        </Route>
      </top>
    </filter>
  </get>
</rpc>
```

Column-based filtering

Column-based filtering includes full match filtering, regular expression match filtering, and conditional match filtering. Full match filtering has the highest priority and conditional match filtering has the lowest priority. When more than one filtering criterion is specified, the one with the highest priority takes effect.

Full match filtering

You can specify an element value in an XML message to implement full match filtering. If multiple element values are provided, the system returns the data that matches all the specified values.

Copy the following text to the client to retrieve configuration data of all interfaces in UP state:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
        <Ifmgr>
          <Interfaces>
            <Interface>
              <AdminStatus>1</AdminStatus>
            </Interface>
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
  </get>
</rpc>
```

You can also specify an attribute name that is the same as a column name of the current table at the row to implement full match filtering. The system returns only configuration data that matches this attribute name. The XML message equivalent to the above element-value-based full match filtering is as follows:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type="subtree">
      <top
xmlns="http://www.h3c.com/netconf/data:1.0"xmlns:data="http://www.h3c.com/netconf/dat
a:1.0">
        <Ifmgr>
          <Interfaces>
            <Interface data:AdminStatus="1"/>
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
  </get>
</rpc>
```

The above examples show that both element-value-based full match filtering and attribute-name-based full match filtering can retrieve the same index and column information for all UP interfaces.

Regular expression match filtering

To implement a complex data filtering with characters, you can add a **regExp** attribute for a specific element.

The supported data types include integer, date and time, character string, IPv4 address, IPv4 mask, IPv6 address, MAC address, OID, and time zone.

Copy the following text to the client to retrieve the descriptions of interfaces, of which all the characters must be upper-case letters from A to Z:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/config:1.0">
        <Ifmgr>
          <Interfaces>
            <Interface>
              <Description h3c:regExp="^[A-Z]*$"/>
            </Interface>
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
  </get-config>
</rpc>
```

Conditional match filtering

To implement a complex data filtering with digits and character strings, you can add a **match** attribute for a specific element. Table 4 lists the conditional match operators.

Table 4 Conditional match operators

Operation	Operator	Remarks
More than	match="more:*value*"	More than the specified value. The supported data types include date, digit, and character string.
Less than	match="less:*value*"	Less than the specified value. The supported data types include date, digit, and character string.
Not less than	match="notLess:*value*"	Not less than the specified value. The supported data types include date, digit, and character string.
Not more than	match="notMore:*value*"	Not more than the specified value. The supported data types include date, digit, and character string.
Equal	match="equal:*value*"	Equal to the specified value. The supported data types include date, digit, character string, OID, and BOOL.
Not equal	match="notEqual:*value*"	Not equal to the specified value. The supported data types include date, digit, character string, OID, and BOOL.
Include	match="include:*string*"	Includes the specified string. The supported data types include only character string.
Not include	match="exclude:*string*"	Excludes the specified string. The supported data types include only character string.
Start with	match="startWith:*string*"	Starts with the specified string. The supported data types include character string and OID.
End with	match="endWith:*string*"	Ends with the specified string. The supported data types include only character string.

Copy the following text to the client to retrieve extension information about the entity whose CPU usage is more than 50%:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
        <Device>
          <ExtPhysicalEntities>
            <Entity>
              <CpuUsage h3c:match="more:50"></CpuUsage>
            </Entity>
          </ExtPhysicalEntities>
        </Device>
      </top>
    </filter>
  </get>
</rpc>
```

Example for filtering data with regular expression match

Network requirements

Retrieve all data including **Gigabit** in the **Description** column of the Interfaces table under the Ifmgr module.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
           urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Retrieve all data including **Gigabit** in the **Description** column of the Interfaces table under the Ifmgr module.

```
<?xml version="1.0"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
        <Ifmgr>
          <Interfaces>
            <Interface>
              <Description h3c:regExp="(Gigabit)+"/>
            </Interface>
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
  </get>
</rpc>
```

Verifying the configuration

If the client receives the following text, the operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0" message-id="100">
    <data>
        <top xmlns="http://www.h3c.com/netconf/data:1.0">
            <Ifmgr>
                <Interfaces>
                    <Interface>
                        <IfIndex>2681</IfIndex>
                        <Description>Ten-GigabitEthernet1/0/1 Interface</Description>
```

```
                    </Interface>
                    <Interface>
                        <IfIndex>2685</IfIndex>
                        <Description>Ten-GigabitEthernet1/0/2 Interface</Description>
                    </Interface>
                    <Interface>
                        <IfIndex>2689</IfIndex>
                        <Description>Ten-GigabitEthernet1/0/3 Interface</Description>
                    </Interface>
                <Interface>
            </Ifmgr>
        </top>
    </data>
</rpc-reply>
```

Example for filtering data by conditional match

Network requirements

Retrieve data in the **Name** column with the ifindex value not less than 5000 in the Interfaces table under the Ifmgr module.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
           urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Retrieve data in the **Name** column with the ifindex value not less than 5000 in the Interfaces table under the Ifmgr module.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0">
  <get>
    <filter type="subtree">
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
        <Ifmgr>
          <Interfaces>
            <Interface>
              <IfIndex h3c:match="notLess:5000"/>
              <Name/>
            </Interface>
          </Interfaces>
        </Ifmgr>
      </top>
    </filter>
```

```
  </get>
</rpc>
```

Verifying the configuration

If the client receives the following text, the operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:h3c="http://www.h3c.com/netconf/base:1.0" message-id="100">
    <data>
        <top xmlns="http://www.h3c.com/netconf/data:1.0">
            <Ifmgr>
                <Interfaces>
                    <Interface>
                        <IfIndex>7241</IfIndex>
                        <Name>NULL0</Name>
                    </Interface>
                </Interfaces>
            </Ifmgr>
        </top>
    </data>
</rpc-reply>
```

Performing CLI operations through NETCONF

You can enclose command lines in XML messages to configure the device.

Performing CLI operations through NETCONF is resource intensive. As a best practice, do not perform the following tasks:

- Enclose multiple command lines in one XML message.
- Use NETCONF to perform a CLI operation when other users are performing NETCONF CLI operations.

Configuration procedure

Copy the following text to the client to execute the commands:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <CLI>
    <Execution>
      Commands
    </Execution>
  </CLI>
</rpc>
```

The <Execution> element can contain multiple commands, with one command on one line.

After receiving the CLI operation request, the device returns a response in the following format if the CLI operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <CLI>
```

```
    <Execution>
      <![CDATA[Responses to the commands]]>
    </Execution>
  </CLI>
</rpc-reply>
```

CLI operation example

Configuration requirements

Send the **display vlan** command to the device.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
           urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Copy the following text to the client to execute the **display vlan** command:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <CLI>
    <Execution>
         display vlan
    </Execution>
  </CLI>
</rpc>
```

Verifying the configuration

If the client receives the following text, the operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <CLI>
    <Execution><![CDATA[
<Sysname>display vlan
 Total VLANs: 1
 The VLANs include:
 1(default)
   ]]>
   </Execution>
  </CLI>
</rpc-reply>
```

Retrieving NETCONF information

Copy the following text to the client to retrieve NETCONF information:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="m-641" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter type='subtree'>
      <netconf-state xmlns='urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring'>
        <getType/>
      </netconf-state>
    </filter>
  </get>
</rpc>
```

The value for the *<getType>* parameter can be one of the following operations:

- **capabilities**—Retrieves device capabilities.
- **datastores**—Retrieves databases from the device.
- **schemas**—Retrieves the list of the YANG file names from the device.
- **sessions**—Retrieves session information from the device.
- **statistics**—Retrieves NETCONF statistics.

If you do not specify a value for the *<getType>* parameter, the retrieval operation retrieves all NETCONF information.

The retrieval operation does not support data filtering.

After receiving the NETCONF information retrieval request, the device returns a response in the following format if the operation is successful:

```
<?xml version="1.0"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
     ALL NETCONF information
  </data>
</rpc-reply>
```

Retrieving YANG file content

YANG files save the NETCONF operations supported by the device. A user can know the supported operations by retrieving and analyzing the content of YANG files.

YANG files are integrated in the device software and are named in the format of *yang_identifier@yang_version*.yang. You cannot view the YANG file names by executing the **dir** command. For information about how to retrieve the YANG file names, see "Retrieving NETCONF information."

Copy the following text to the client to retrieve the YANG file named **syslog-data@2015-05-07.yang**:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-schema xmlns='urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring'>
    <identifier>syslog-data</identifier>
    <version>2015-05-07</version>
    <format>yang</format>
```

```
  </get-schema>
</rpc>
```

After receiving the YANG file retrieve request, the device returns a response in the following format if the operation is successful:

```
<?xml version="1.0"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
     Content of the specified YANG file
  </data>
</rpc-reply>
```

Retrieving NETCONF session information

You can use the <get-sessions> operation to retrieve NETCONF session information of the device.

Copy the following message to the client to retrieve NETCONF session information from the device:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-sessions/>
</rpc>
```

After receiving the get-sessions request, the device returns a response in the following format if the <get-sessions> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-sessions>
    <Session>
      <SessionID>Configuration session ID</SessionID>
      <Line>Line information</Line>
      <UserName>Name of the user creating the session</UserName>
      <Since>Time when the session was created</Since>
      <LockHeld>Whether the session holds a lock</LockHeld>
    </Session>
  </get-sessions>
</rpc-reply>
```

For example, to get NETCONF session information:

Enter XML view.

```
<Sysname> xml
```

Copy the following message to the client to exchange capabilities with the device:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
            urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Copy the following message to the client to get the current NETCONF session information on the device:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-sessions/>
</rpc>
```

If the client receives a message as follows, the operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="100">
    <get-sessions>
        <Session>
            <SessionID>1</SessionID>
            <Line>vty0</Line>
            <UserName></UserName>
            <Since>2011-01-05T00:24:57</Since>
            <LockHeld>false</LockHeld>
        </Session>
    </get-sessions>
</rpc-reply>
```

The output shows the following information:

- The session ID of an existing NETCONF session is 1.
- The login user type is vty0.
- The login time is 2011-01-05T00:24:57.
- The user does not hold the lock of the configuration.

Terminating another NETCONF session

NETCONF allows one client to terminate the NETCONF session of another client. The client whose session is terminated returns to user view.

Configuration procedure

Copy the following message to the client to terminate the specified NETCONF session:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>
      Specified session-ID
    </session-id>
  </kill-session>
</rpc>
```

After receiving the kill-session request, the device returns a response in the following format if the <kill-session> operation is successful:

```
<?xml version="1.0" encoding="UTF-8"?>
    <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <ok/>
</rpc-reply>
```

Configuration example

Configuration requirement

The user whose session's ID is 1 terminates the session with session ID 2.

Configuration procedure

Enter XML view.

```
<Sysname> xml
```

Notify the device of the NETCONF capabilities supported on the client.

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>
            urn:ietf:params:netconf:base:1.0
    </capability>
  </capabilities>
</hello>
```

Terminate the session with session ID 2.

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>2</session-id>
  </kill-session>
</rpc>
```

Verifying the configuration

If the client receives the following text, the NETCONF session with session ID 2 has been terminated, and the client with session ID 2 has returned from XML view to user view:

```
<?xml version="1.0" encoding="UTF-8"?>
  <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
```

Returning to the CLI

To return from XML view to the CLI, send the following close-session request:

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <close-session/>
</rpc>
```

When the device receives the close-session request, it sends the following response and returns to CLI's user view:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

Appendix

Appendix A Supported NETCONF operations

Table 5 lists the NETCONF operations available with Comware 7.

Table 5 NETCONF operations

Operation	Description	XML example

get	Retrieves device configuration and state information.	To retrieve device configuration and state information for the Syslog module:
		`<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:xc="http://www.h3c.com/netconf/base:1.0">`
		`<get>`
		`<filter type="subtree">`
		`<top xmlns="http://www.h3c.com/netconf/data:1.0">`
		`<Syslog>`
		`</Syslog>`
		`</top>`
		`</filter>`
		`</get>`
		`</rpc>`

get-config	Retrieves the non-default configuration data. If non-default configuration data does not exist, the device returns a response with empty data.	To retrieve non-default configuration data for the interface table:
		`<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:xc="http://www.h3c.com/netconf/base:1.0">`
		`<get-config>`
		`<source>`
		`<running/>`
		`</source>`
		`<filter type="subtree">`
		`<top xmlns="http://www.h3c.com/netconf/config:1.0">`
		`<Ifmgr>`
		`<Interfaces>`
		`<Interface/>`
		`</Interfaces>`
		`</Ifmgr>`
		`</top>`
		`</filter>`
		`</get-config>`
		`</rpc>`

Operation	Description	XML example
get-bulk	Retrieves a number of data entries (including device configuration and state information) starting from the data entry next to the one with the specified index.	To retrieve device configuration and state information for all interface: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get-bulk> <filter type="subtree"> <top xmlns="http://www.h3c.com/netconf/data:1.0"> <Ifmgr> <Interfaces xc:count="5" xmlns:xc="http://www.h3c.com/netconf/base:1.0"> <Interface/> </Interfaces> </Ifmgr> </top> </filter> </get-bulk> </rpc>
get-bulk-config	Retrieves a number of non-default configuration data entries starting from the data entry next to the one with the specified index.	To retrieve non-default configuration for all interfaces: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get-bulk-config> <source> <running/> </source> <filter type="subtree"> <top xmlns="http://www.h3c.com/netconf/config:1.0"> <Ifmgr> </Ifmgr> </top> </filter> </get-bulk-config> </rpc>
edit-config*:* incremental	Adds configuration data to a column without affecting the original data. The **incremental** attribute applies to a list column such as the vlan permitlist column. You can use the **incremental** attribute for <edit-config> operations except for the replace operation. Support for the **incremental** attribute varies by module. For more information, see NETCONF XML API documents.	To add VLANs 1 through 10 to an untagged VLAN list that has untagged VLANs 12 through 15: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:h3c="http://www.h3c.com/netconf/base:1.0"> <edit-config> <target> <running/>

Operation	Description	XML example
		</target> <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"> <top xmlns="http://www.h3c.com/netconf/config:1.0"> <VLAN xc:operation="merge"> <HybridInterfaces> <Interface> <IfIndex>262</IfIndex> <UntaggedVlanList h3c:incremental="true">1-10</UntaggedVlanList> </Interface> </HybridInterfaces> </VLAN> </top> </config> </edit-config> </rpc>
edit-config: merge	Changes the running configuration. To use the **merge** attribute in an <edit-config> operation, you must specify the operation target (on a specified level): • If the specified target exists, the operation directly changes the configuration for the target. • If the specified target does not exist, the operation creates and configures the target. • If the specified target does not exist and it cannot be created, an error message is returned.	To change the buffer size to 120: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"> <edit-config> <target> <running/> </target> <config> <top xmlns="http://www.h3c.com/netconf/config:1.0"> <Syslog xmlns="http://www.h3c.com/netconf/config:1.0" xc:operation="merge"> <LogBuffer> <BufferSize>120</BufferSize> </LogBuffer> </Syslog> </top> </config> </edit-config> </rpc>
edit-config: create	Creates a specified target. To use the **create** attribute in an <edit-config> operation, you must specify the operation	The XML data format is the same as the edit-config message with the **merge** attribute. Change the operation attribute

Operation	Description	XML example
	target. • If the table supports target creation and the specified target does not exist, the operation creates and then configures the target. • If the specified target exists, a data-exist error message is returned.	from **merge** to **create**.
edit-config: replace	Replaces the specified target. • If the specified target exists, the operation replaces the configuration of the target with the configuration carried in the message. • If the specified target does not exist but is allowed to be created, create the target and then apply the configuration of the target. • If the specified target does not exist and is not allowed to be created, the operation is not conducted and an invalid-value error message is returned.	The syntax is the same as the edit-config message with the **merge** attribute. Change the operation attribute from **merge** to **replace**.
edit-config: remove	Removes the specified configuration. • If the specified target has only the table index, the operation removes all configuration data of the specified target, and the target itself. • If the specified target has the table index and configuration data, the operation removes the specified configuration data of this target. • If the specified target does not exist, or the XML message does not specify any targets, a success message is returned.	The syntax is the same as the edit-config message with the **merge** attribute. Change the operation attribute from **merge** to **remove**.
edit-config: delete	Deletes the specified configuration. • If the specified target has only the table index, the operation removes all configuration data of the specified target, and the target itself. • If the specified target has the table index and configuration data, the operation removes the specified configuration data of this target. • If the specified target does not exist, an error message is returned, showing that the target does not exist.	The syntax is the same as the edit-config message with the **merge** attribute. Change the operation attribute from **merge** to **delete**.

edit-config: default-operation	Modifies the current configuration of the device using the default operation method.
	If you do not specify an operation attribute for an <edit-config> message, NETCONF uses one of the following default operation attributes: merge, create, delete, and replace. Your setting of the value for the <default-operation> element takes effect only once. If you do not specify an operation attribute and the default operation method for an <edit-config> message, **merge** is
		To issue an empty operation for schema verification purposes:
		`<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
		`<edit-config>`
		`<target>`
		`<running/>`
		`</target>`

Operation	Description	XML example
	always applied. • **merge**—The default value for the <default-operation> element. • **replace**—Value used when the operation attribute is not specified and the default operation method is specified as **replace**. • **none**—Value used when the operation attribute is not specified and the default operation method is specified as **none**. If this value is specified, the <edit-config> operation is used only for schema verification rather than issuing a configuration. If the schema verification is passed, a successful message is returned. Otherwise, an error message is returned.	<default-operation>none</default-operation> <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"> <top xmlns="http://www.h3c.com/netconf/config:1.0"> <Ifmgr> <Interfaces> <Interface> <IfIndex>262</IfIndex> <Description>222222</Description> </Interface> </Interfaces> </Ifmgr> </top> </config> </edit-config> </rpc>
edit-config: error-option	Determines the action to take in case of a configuration error. The error-option element has one of the following values: • **stop-on-error**—Stops the operation on error and returns an error message. This is the default error-option value. • **continue-on-error**—Continues the operation on error and returns an error message. • **rollback-on-error**—Rolls back the configuration.	To issue the configuration for two interfaces with the error-option element value as continue-on-error: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <edit-config> <target> <running/> </target> <error-option>continue-on-error</error-option> <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"> <top xmlns="http://www.h3c.com/netconf/config:1.0"> <Ifmgr xc:operation="merge"> <Interfaces> <Interface> <IfIndex>262</IfIndex> <Description>222</Description> <ConfigSpeed>1024</ConfigSpeed> <ConfigDuplex>1</ConfigDuplex> </Interface> <Interface>

Operation	Description	XML example
		<IfIndex>263</IfIndex> <Description>333</Description> <ConfigSpeed>1024</ConfigSpeed> <ConfigDuplex>1</ConfigDuplex> </Interface> </Interfaces> </Ifmgr> </top> </config> </edit-config> </rpc>
edit-config: test-option	Determines whether to issue a configuration item in an <edit-config> operation. The test-option element has one of the following values: • **test-then-set**—Performs a validation test before attempting to set. If the validation test fails, the <edit-config> operation is not performed. This is the default test-option value. • **set**—Directly performs the set operation without the validation test. • **test-only**—Performs only a validation test without attempting to set. If the validation test succeeds, a successful message is returned. Otherwise, an error message is returned.	To issue the configuration for an interface for test purposes: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <edit-config> <target> <running/> </target> <test-option>test-only</test-option> <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0"> <top xmlns="http://www.h3c.com/netconf/config:1.0"> <Ifmgr xc:operation="merge"> <Interfaces> <Interface> <IfIndex>262</IfIndex> <Description>222</Description> <ConfigSpeed>1000000</ConfigSpeed> <ConfigDuplex>1</ConfigDuplex> </Interface> </Interfaces> </Ifmgr> </top> </config> </edit-config> </rpc>

Operation	Description	XML example
action	Issues actions that are not for configuring data, for example, reset action.	To clear statistics information for all interfaces: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <action> <top xmlns="http://www.h3c.com/netconf/action:1.0"> <Ifmgr> <ClearAllIfStatistics> <Clear> </Clear> </ClearAllIfStatistics> </Ifmgr> </top> </action> </rpc>
lock	Locks configuration data that can be changed by <edit-config> operations. Other configuration data are not limited by the lock operation. After a user locks the configuration, other users cannot use NETCONF or other configuration methods such as CLI and SNMP to configure the device.	To lock the configuration: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <lock> <target> <running/> </target> </lock> </rpc>
unlock	Unlocks the configuration, so NETCONF sessions can change device configuration. When a NETCONF session is terminated, the related locked configuration is also unlocked.	To unlock the configuration: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <unlock> <target> <running/> </target> </unlock> </rpc>
get-sessions	Retrieves information about all NETCONF sessions in the system.	To retrieve information about all NETCONF sessions in the system: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get-sessions/> </rpc>
close-session	Terminates the NETCONF session for the current user, to unlock the configuration and release the resources (for example, memory) of this session. This operation logs the current user off the XML view.	To terminate the NETCONF session for the current user: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">

Operation	Description	XML example
		<close-session/> </rpc>
kill-session	Terminates the NETCONF session for another user. This operation cannot terminate the NETCONF session for the current user.	To terminate the NETCONF session with session-id 1: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <kill-session> <session-id>1</session-id> </kill-session> </rpc>
CLI	Executes CLI operations. A request message encloses commands in the <CLI> element, and a response message encloses the command output in the <CLI> element. You can use the following elements to execute commands: •**Execution**—Executes commands in user view. •**Configuration**—Executes commands in system view. To execute commands in a lower-level view of the system view, use the <Configuration> element to enter the view first. To use this element, include the exec-use-channel attribute and specify a value for the attribute: ○ **false**—Executes commands without using a channel. ○ **true**—Executes commands by using a temporary channel. The channel is automatically closed after the execution. ○ **persist**—Executes commands by using the persistent channel for the session. To use the persistent channel, first perform an <Open-channel> operation to open the persistent channel. If you do not do so, the system will automatically open the persistent channel. After using the persistent channel, perform a <Close-channel> operation to close the channel and return to system view. If you do not perform an <Open-channel> operation, the system stays in the view and will execute subsequent commands in the view. A NETCONF session supports only one persistent channel and but supports multiple temporary channels. NETCONF does not support executing interactive commands. You cannot execute the **quit** command by	To execute the **display this** command in system view without using a channel: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <CLI> <Configuration exec-use-channel="false">display this</Configuration> </CLI> </rpc>

Operation	Description	XML example
	using a channel to exit user view.	
save	Saves the running configuration. You can use the <file> element to specify a file for saving the configuration. If the text does not include the file column, the running configuration is automatically saved to the main next-startup configuration file. The **OverWrite** attribute determines whether the current configuration overwrites the original configuration file when the specified file already exists. The **Binary-only** attribute determines whether to save the running configuration only to the binary configuration file.	To save the running configuration to file **test.cfg**: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <save OverWrite="false" Binary-only="true"> <file>test.cfg</file> </save> </rpc>
load	Loads the configuration. After the device finishes the <load> operation, the configuration in the specified file is merged into the current configuration of the device.	To merge the configuration in file **a1.cfg** to the current configuration of the device: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <load> <file>a1.cfg</file> </load> </rpc>
rollback	Rolls back the configuration. To do so, you must specify the configuration file in the <file> element. After the device finishes the <rollback> operation, the current device configuration is totally replaced with the configuration in the specified configuration file.	To roll back the current configuration to the configuration in file **1A.cfg**: <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <rollback> <file>1A.cfg</file> </rollback> </rpc>