使用 NETCONF 配置设备操作指导书

目 录

1 NETCONF简介

NETCONF（Network Configuration Protocol，网络配置协议）是一种基于 XML 的网络管理协议，它提供了一种可编程的、对网络设备进行配置和管理的方法。用户可以通过该协议设置参数、获取参数值、获取统计信息等。

NETCONF 报文使用 XML 格式，具有强大的过滤能力，而且每一个数据项都有一个固定的元素名称和位置，这使得同一厂商的不同设备具有相同的访问方式和结果呈现方式，不同厂商之间的设备也可以经过映射 XML 得到相同的效果，这使得它在第三方软件的开发上非常便利，很容易开发出在混合不同厂商、不同设备的环境下的特殊定制的网管软件。在这样的网管软件的协助下，使用 NETCONF 功能会使网络设备的配置管理工作，变得更简单更高效。

1.1 NETCONF协议结构

NETCONF 协议采用分层结构，分为内容层（Content）、操作层（Operations）、RPC（Remote Procedure Call，远程调用）层和通信协议层（Transport Protocol）。

表1 XML 分层与 NETCONF 分层模型对应关系

NETCONF 分层	XML 分层	说明
内容层	配置数据、状态数据、统计信息等	被管理对象的集合，可以是配置数据、状态数据、统计信息等 NETCONF协议具体可读写的数据请参见《NETCONF XML API 手册》
操作层	<get>,<get-config>,<edit-config>	在RPC中应用的基本的原语操作集，这些操作组成NETCONF的基本能力 NETCONF 全面地定义了对被管理设备的各种基础操作，如get、get-config、get-bulk、edit-config操作等 设备支持的操作请参见“7.4 Comware V7中支持的NETCONF操作类型”
RPC层	<rpc>,<rpc-reply>	为RPC模块的编码提供了简单的、传输协议无关的机制。通过使用<rpc>和<rpc-reply>元素分别对NETCONF请求和响应数据（即操作层和内容层的内容）进行封装
通信协议层	非FIPS模式下：Console/Telnet/SSH/HTTP/HTTPS/TLS FIPS模式下： Console/SSH/HTTPS/TLS	为NETCONF提供面向连接的、可靠的、顺序的数据链路。 非FIPS模式下： • NETCONF 支持 Telnet、SSH 和 Console 等 CLI 登录方式/协议，即 NETCONF over SSH、NETCONF over Telnet 和 NETCONF over Console • NETCONF 支持 HTTP 和 HTTPS 协议，即 NETCONF over HTTP 和 NETCONF over HTTPS • NETCONF 支持封装成 SOAP（Simple Object Access Protocol，简单对象访问协议）报文后通过 HTTP 或 HTTPS 协议传输，即 NETCONF over SOAP over HTTP 和 NETCONF over SOAP over HTTPS FIPS模式下： • NETCONF 支持 SSH 和 Console 等 CLI 方式/协议，即 NETCONF over SSH 和 NETCONF over Console • NETCONF 支持 HTTPS 登录协议，即 NETCONF over HTTPS • NETCONF 支持封装成 SOAP 报文后通过 HTTPS 协议传输，即 NETCONF over SOAP over HTTPS

1.2 NETCONF报文格式

通信协议层不进行 SOAP 封装时的报文格式我们称为 NETCONF 格式。通信协议层进行 SOAP 封装时的报文格式我们称为 NETCONF over SOAP 格式。

1.2.1 NETCONF

NETCONF 命令必须符合 XML 语言的基本格式。NETCONF 报文格式遵循 RFC 4741/RFC 6241。

1. 请求格式

请求分为协议定义、H3C 自有两个部分，其格式如下：

- 协议定义部分：即 RFC 4741/RFC 6241 中规定的部分。其中：
  - encoding 表示使用的 XML 编码格式。Comware V7 NETCONF 支持 GB2312、GB18030、UTF-8、UTF-16、UTF-16BE、UTF-16LE、UTF-32、UTF-32BE、UTF-32LE 编码格式。如果请求中没有携带声明部分（即<?xml version="1.0" encoding="utf-8"?>）指定 XML 编码格式，则默认使用 UTF-8 编码格式。
  - message-id 表示消息 ID。客户端使用单调递增的整数来表示消息 ID。服务器端在应答中会使用相同的消息 ID 以表示应答对应的请求。
  - 协议部分的命名空间必须为 urn:ietf:params:xml:ns:netconf:base:1.0。

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id =”101” xmlns=”urn:ietf:params:xml:ns:netconf:base:1.0”>
  <operation>
</rpc>
```

- H3C 自有部分：对于 get 系列操作，filter 元素下的内容为 H3C 自有部分；对于 edit-config 系列操作，config 元素下的内容为 H3C 自有部分。
  - H3C 自有部分需要使用 H3C 命名空间，H3C 命名空间又分为 base、config、data、action 命名空间。

    Base 命名空间：http://www.h3c.com/netconf/base:1.0

    Config 命名空间：http://www.h3c.com/netconf/config:1.0

    Data 命名空间：http://www.h3c.com/netconf/data:1.0

    Action 命名空间：http://www.h3c.com/netconf/action:1.0

    具体使用哪个命名空间与操作类型和内容有关，可参见“7.4 Comware V7 中支持的NETCONF操作类型”部分描述。
  - H3C 自有部分可以以 top 元素为起点，也可以以具体模块为起点。是否使用 top 元素由 **netconf capability specific-namespace** 命令进行配置。缺省使用 top 元素，命名空间位于 top 元素之后，各模块共用命名空间。配置 **netconf capability specific-namespace** 命令后不再需要使用 top 元素，命名空间位于模块名之后，各模块专用自己的命名空间。模块专用的命名空间定义方式为 *H3C 命名空间-模块名*，例如，接口管理模块的 data 命名空间为 http://www.h3c.com/netconf/data:1.0-Ifmgr。

- “模块信息”部分的内容由 NETCONF API 文档定义。

get-config 的例子（使用共有命名空间）：

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id=”101” 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”>
        模块信息
      </top>
    </filter>
  </get-config>
</rpc>
```

edit-config 的例子（使用共有命名空间）：

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id =”101” xmlns=”urn:ietf:params:xml:ns:netconf:base:1.0”>
  <edit-config>
    <target>
      <running/>
    </target>
    <config xmlns:xc=”urn:ietf:params:xml:ns:netconf:base:1.0”>
      <top xmlns=”http://www.h3c.com/netconf/config:1.0”>
        模块信息
      </top>
    </config>
  </edit-config>
</rpc>
```

2. 应答格式

统一为协议 RFC 4741 定义的<rpc-reply>，如：

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns=”urn:ietf:params:xml:ns:netconf:base:1.0” message-id=”101”>
  <ok/>
</rpc-reply>
```

3. NETCONF请求示例

如下为一个 NETCONF 报文示例，用于获取设备上所有接口的所有参数：

- 使用共有命名空间时：

```
<?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>
```

- 使用专用命名空间时：

```
<?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">
        <Ifmgr xmlns="http://www.h3c.com/netconf/data:1.0-Ifmgr">
          <Interfaces>
             <Interface/>
          </Interfaces>
        </Ifmgr>
    </filter>
  </get-bulk>
</rpc>
```

1.2.2 NETCONF over SOAP

1. SOAP语法

一条 SOAP 消息包含以下元素：

- Envelope 元素：必选，用来将 XML 文档标识为一条 SOAP 消息。
- Header 元素：可选，包含头部信息。
- Body 元素：必选，包含所有的调用和响应信息。
- Fault 元素：可选，提供在处理此消息时发生错误的信息。

NETCONF over SOAP 之后，NETCONF 报文会放在 SOAP 报文的 BODY 元素里，这些报文除了需要遵循纯 NETCONF 报文的规则外，还需要遵循以下规则：

- SOAP 消息必须用 XML 来编码。
- SOAP 消息必须使用 SOAP Envelope 命名空间，包括 http://schemas.xmlsoap.org/soap/envelope/和 http://www.w3.org/2003/05/soap-envelope 两种。不同设备支持的命名空间不同，请以设备的实际情况为准。
- SOAP 消息不能包含 DTD（Document Type Definition，文件类型定义）引用。
- SOAP 消息不能包含 XML 处理指令。

2. NETCONF over SOAP报文格式

Hello 报文格式为：

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
    <auth:Authentication env:mustUnderstand="1"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
      <auth:UserName>test</auth:UserName>
      <auth:Password>test</auth:Password>
```

```
      <auth:Language>zh-cn</auth:Language>
    </auth:Authentication>
  </env:Header>
  <env:Body>
    <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
      <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
      </capabilities>
    </hello>
  </env:Body>
</env:Envelope>
```

NETCONF over SOAP 请求的格式为：

```
<env:Envelope xmlns:env=”http://schemas.xmlsoap.org/soap/envelope/”>
  <env:Header>
    <auth:Authentication env:mustUnderstand=”1”
xmlns:auth=”http://www.h3c.com/netconf/base:1.0”>
      <auth:AuthInfo>10027c2abebdb482633f847102fbc890d22a</auth:AuthInfo>
      <auth:Language>en</auth:Language>
    </auth:Authentication>
  </env:Header>
  <env:Body>
    <rpc message-id=”101” 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>
  </env:Body>
</env:Envelope>
```

NETCONF over SOAP 报文格式中：

- Envelope 元素：固定格式，在该元素中需要指定 Envelope 命名空间。
- Header 元素：携带认证、使用的语言等信息。
  - 认证信息：发送 Hello 建立连接时，NETCONF 客户端在请求消息中携带 auth:UserName 和 auth:Password，将认证用户名和密码发送给设备。如果认证成功，则设备在应答消息中返回登录成功的凭据（auth:AuthInfo）。后续基于该连接的 NETCONF 请求必须携带该凭据。

    Authentication元素必须使用命名空间 http://www.h3c.com/netconf/base:1.0，必须具有 http://www.w3.org/2003/05/soap-envelope命名空间下的mustUnderstand属性，且属性值必须为 1 或者true。

- 使用的语言：在请求消息中通过 auth:Language 设置客户端期望的返回错误信息使用的语言。目前支持中文（zh-cn）和英文（en）两种语言。如果未指定，则表示使用英文。

  并不是所有的请求消息都支持返回中文错误提示信息，应答消息中通过 xml:lang="*en*"表明实际使用的语言。

  auth:Language 必须携带在 auth:UserName、auth:Password 和 auth:AuthInfo 之后。

- Body 元素：携带的内容为 NETCONF 报文。

说明

NETCONF over SOAP 报文格式与 NETCONF 报文格式类似。下文如无特殊情况，均以 NETCONF 报文为例，说明报文构造方法。将 NETCONF 报文封装到 NETCONF over SOAP 报文的 Body 元素中，即可构造出对应的 NETCONF over SOAP 报文。

3. SOAP请求示例

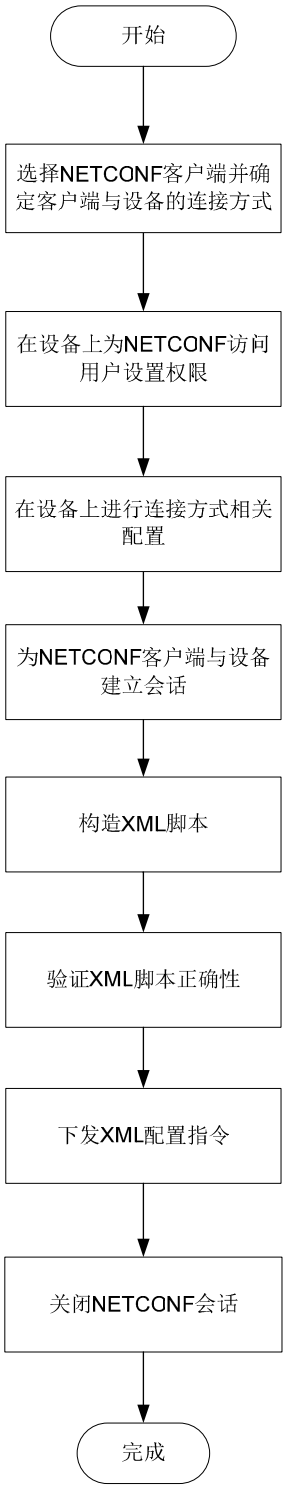
如下为一个 NETCONF over SOAP 报文示例，用于获取设备上所有接口的所有参数：

```
<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>
```

2 NETCONF配置任务简介

使用NETCONF客户端对接并配置设备的主要过程如 图 1 所示。

图1 使用 NETCONF 配置设备流程图



提示

除上述方法外，用户也可以在 Python 上安装 ncclient 库，利用已有或用户自行开发的 Python IDE（Integrated Development Environment，集成开发环境），通过 ncclient 库将较为直观地 Python 脚本语言转化为 Netconf 报文，对 Netconf 客户端（设备）进行配置和管理，以简化网络配置脚本语言的编写。

3 选择NETCONF客户端并确定客户端与设备的连接方式

3.1 NETCONF配置方式简介

用户可通过以下方式来使用 NETCONF 配置和管理设备：

- 通过Telnet、SSH或Console登录到设备并进入XML视图，将合法的NETCONF报文（1.2.1 NETCONF节所描述格式）直接拷贝、粘贴到命令行提示符处执行，即可实现对设备的配置和管理。

  说明

  在 XML 视图下进行 NETCONF 配置时，NETCONF 报文最后需要添加“**]]>]]>**”结束符，否则设备无法识别。

- 通过 HTTP 或 HTTPS 登录到设备的 Web 页面，系统会自动将 Web 页面的配置转换成 NETCONF 指令下发给设备来实现对设备的配置和管理。该方式通过简洁易用的图像化 Web 页面上完成配置，配置结果也会以简洁清晰的方式呈现给用户，整个 NETCONF 的交互过程对用户透明。
- 使用配置工具与设备建立连接并向设备下发 NETCONF 指令来实现对设备的配置和管理。配置工具可以分为以下两类：
  - SSH配置工具：该类配置工具与设备建立SSH连接后，可下发NETCONF格式（即 1.2.1 NETCONF节所描述格式）报文配置和管理设备。
  - SOAP配置工具：该类配置工具通过HTTP/HTTPS与设备建立连接，并将NETCONF指令用SOAP封装成通用格式的报文后发送给设备，因此需要使用 1.2.2 NETCONF over SOAP 节所描述格式。使用该方式前设备上必须开启NETCONF over SOAP功能。

  SOAP 配置工具使用短连接，SSH 配置工具使用长连接。因此，SSH 配置工具具有较高的效率，但对设备和客户端的资源消耗较高。

综上，实际应用中使用 NETCONF 配置设备时主要采用配置工具的方式。客户可以开发和定制自己专用的配置工具软件，也可以使用第三方开发的配置工具。第三方的配置工具有：

- SSH 配置工具：NETCONF Browser 等。
- SOAP 配置工具：SOAP UI 等。

关于各配置工具的使用方法，请参见配置工具的配置指导。

3.2 连接方式介绍

H3C 设备目前支持如下连接方式：

非 FIPS 模式下：

- NETCONF over SSH
- NETCONF over Telnet
- NETCONF over Console
- NETCONF over HTTP
- NETCONF over HTTPS
- NETCONF over SOAP over HTTP
- NETCONF over SOAP over HTTPS

FIPS 模式下：

- NETCONF over SSH
- NETCONF over Console
- NETCONF over HTTPS
- NETCONF over SOAP over HTTPS

不同配置方式使用的连接方式不同：

- 使用 Telnet 登录到设备并进入 XML 视图即建立 NETCONF over Telnet 连接。
- 使用 Console 登录到设备并进入 XML 视图即建立 NETCONF over Console 连接。建议尽量不使用 NETCONF over Console 方式，因 Console 口的速度限制，且 XML 视图下不输出提示、告警信息，比较容易出现错误。
- 使用 HTTP 或 HTTPS 登录到设备的 Web 页面进行配置时，Web 使用 NETCONF over HTTP 或 NETCONF over HTTPs 连接方式与设备交互配置信息。交互过程对用户透明，本文后续不再介绍。通过 Web 配置设备的支持情况与产品型号有关，请以设备的实际情况为准。
- 使用 SSH 配置工具或使用 SSH 登录到设备进入 XML 视图执行 NETCONF 配置时需要使用 NETCONF over SSH 连接方式。
- 使用 SOAP 配置工具下发 NETCONF 指令配置设备时，需要使用 NETCONF over SOAP over HTTP 或 NETCONF over SOAP over HTTPS 方式。

说明

本文重点介绍通过 NETCONF over SSH、NETCONF over SOAP over HTTP 和 NETCONF over SOAP over HTTPS 配置工具与设备建立连接，其他连接方式不再介绍。

4 在设备上为NETCONF用户设置权限

使用配置工具与设备建立 NETCONF 连接前，需要确保 NETCONF 用户具有对应的操作权限。

4.1 确定NETCONF用户需要的权限

在 Comware V7 系统中，用户的权限通过它所属的角色的权限来控制，角色的权限主要包括规则和资源策略两个方面。

在规则方面，NETCONF 用户角色需要如下权限：

- 执行NETCONF操作需要具有执行XML元素NETCONF RPC节点的权限，不同NETCONF操作需要的权限不同，具体请参见 表 2。
- 执行 NETCONF 操作内容的权限可以通过用户执行 XML 元素具体模块及其表的 Xpath 的权限控制，例如配置用户具有执行 XML 元素接口模块的权限，需要执行 **rule** *number* **permit read write execute xml-element ifmgr/**命令。

在资源策略方面，可以根据实际需要配置用户角色操作接口、VLAN、VPN、安全域的权限。缺省情况下，用户具有操作所有资源的权限。

缺省用户角色 network-admin、mdc-admin、context-admin 可操作对应设备/MDC/Context 的所有功能和资源（除安全日志文件管理相关命令 **display security-logfile summary**、**info-center security-logfile directory**、**security-logfile save** 之外）的权限，如果用户所属角色是这几种，则不需要进行权限配置，只需要指定用户角色为 network-admin、mdc-admin 或 context-admin 即可。

更多关于用户角色权限控制的内容，请参见“基础配置指导”中的“RBAC”。

表2 执行 NETCONF 操作需要配置的权限

执行的 NETCONF 操作	需要配置的权限节点	需要配置的权限
建立NETCONF会话	不涉及	执行xml命令的权限
从设备订阅事件	rpc/create-subscription	execute
给当前配置加锁	rpc/lock	execute
给当前配置解锁	rpc/unlock	execute
使用<get>获取信息	rpc/get	read
使用<get>获取系统支持的事件流	rpc/get/filter/netconf	read
使用<get>获取NETCON状态信息	rpc/get/filter/netconf-state	read
使用<get-bulk>获取信息	rpc/get-bulk	read
使用<get-config>获取配置信息	rpc/get-config	read
使用<get-bulk-config>获取配置信息	rpc/get-bulk-config	read
使用<get-schema>获取yang文件信息	rpc/get-schema	read
<edit-config>编辑指定模块数据	rpc/edit-config	write
执行一个<action>操作	rpc/action	execute

执行的 NETCONF 操作	需要配置的权限节点	需要配置的权限
配置保存	rpc/save	write
配置回滚	rpc/rollback	write
配置加载	rpc/load	write
命令行操作	rpc/CLI	write
获取会话信息	rpc/get-sessions	read
关闭另一个会话	rpc/kill-session	execute
执行语法验证	rpc/validate	read

4.2 定义NETCONF用户角色规则

(1) 进入系统视图。

system-view

(2) 创建用户角色，并进入用户角色视图。

role name *role-name*

(3) 请根据需要配置用户操作权限的规则。

- 配置用户具有执行 XML 元素 NETCONF RPC 节点的权限。

  rule *number* **permit** { **execute** | **read** | **write** } * **xml-element rpc/**

  通过 **rule** *number* **permit** { **execute** | **read** | **write** } * **xml-element rpc/?**可以查看具体操作的列表。不指定具体操作时，表示所有操作。

- 配置用户执行 XML 元素具体模块的权限。

  rule *number* { **deny** | **permit** } { **execute** | **read** | **write** } * **xml-element** [*module-xpath*]

  通过 **rule** *number* { **deny** | **permit** } { **execute** | **read** | **write** } * **xml-element ?** 可以查看具体模块列表，通过 **rule** *number* { **deny** | **permit** } { **execute** | **read** | **write** } * **xml-element** *module-name/?* 可以查看模块中具体表的列表。不指定具体模块和表时表示所有模块/所有表。

4.3 配置NETCONF用户属性

1. 配置限制和指导

本文采用新建本地用户介绍配置过程。使用远端认证用户的配置方式请参见具体认证方式的配置指导，只要使通过认证的 NETCONF 用户满足以下两个条件：

- 使用 NETCONF over SSH 连接方式时，服务类型为 SSH。使用 NETCONF over SOAP over HTTP 连接方式时，服务类型为 HTTP。使用 NETCONF over SOAP over HTTPS 连接方式时，服务类型为 HTTPS。
- 具有所需权限。

2. 配置步骤

(1) 进入系统视图。

system-view

(2) 添加设备管理类本地用户，并进入设备管理类本地用户视图。

local-user *user-name* **class manage**

(3) 设置本地用户的密码。

（非 FIPS 模式）

password [{ **hash** | **simple** } *string*]

（FIPS 模式）

password

在非 FIPS 模式下，可以不为本地用户设置密码；在 FIPS 模式下，必须且只能通过交互式方式设置明文密码，否则用户的本地认证不能成功。

(4) 设置本地用户可以使用的服务类型。请根据连接方式选择其中一项进行配置

- 使用 NETCONF over SSH 连接方式：

  service-type ssh

- 使用 NETCONF over SOAP over HTTP 或 NETCONF over SOAP over HTTPS 连接方式：

  （非 FIPS 模式）

  service-type { **http** | **https** } *

  （FIPS 模式）

  service-type https

缺省情况下，本地用户不能使用任何服务类型。

(5) 设置本地用户的角色。

authorization-attribute user-role *role-name*

5 在设备上进行连接方式相关配置

5.1 选用NETCONF over SSH连接方式

5.1.1 配置SSH登录

以下配置步骤只介绍采用 **password** 方式认证 SSH 客户端的配置方法，**publickey** 方式的配置方法及 SSH 的详细介绍，请参见“安全配置指导”中的“SSH”。

(1) 进入系统视图。

system-view

(2) 生成本地密钥对。

（非 FIPS 模式）

public-key local create { **dsa** | **ecdsa** [**secp192r1** | **secp256r1** | **secp384r1** | **secp521r1**] | **rsa** } [**name** *key-name*]

（FIPS 模式）

public-key local create { **dsa** | **ecdsa** [**secp256r1** | **secp384r1** | **secp521r1**] | **rsa** } [**name** *key-name*]

(3) （可选）创建 SSH 用户，并指定 SSH 用户的服务类型为 NETCONF，认证方式为 password。

ssh user *username* **service-type netconf authentication-type password**

(4) 进入 VTY 用户线或 VTY 用户线类视图。

- 进入 VTY 用户线视图。

  line vty *first-number* [*last-number*]

- 进入 VTY 用户线类视图。

  line class vty

(5) 配置 VTY 用户线的认证方式为 scheme 方式。

（非 FIPS 模式）

authentication-mode scheme

缺省情况下，VTY 用户线的认证方式为 **password** 方式。

（FIPS 模式）

authentication-mode scheme

缺省情况下，VTY 用户线的认证方式为 **scheme** 方式。

5.1.2 配置NETCONF over SSH

(1) 进入系统视图。

system-view

(2) 开启 NETCONF over SSH。

netconf ssh server enable

缺省情况下，NETCONF over SSH 处于关闭状态。

(3) 配置 NETCONF over SSH 的监听端口。

netconf ssh server port *port-number*

缺省情况下，NETCONF over SSH 的监听端口为 830。

(4) （可选）开启 NETCONF 日志功能。

netconf log source { **all** | { **agent** | **soap** | **web** } * } { **protocol-operation** { **all** | { **action** | **config** | **get** | **set** | **session** | **syntax** | **others** } * } | **row-operation** | **verbose** }

缺省情况下，NETCONF 日志功能处于关闭状态。

(5) 通过 SSH 配置工具与设备建立 NETCONF over SSH 会话。关于 SSH 配置工具的使用方法，具体参见 SSH 配置工具的配置指导。

5.2 选用NETCONF over SOAP over HTTP或NETCONF over SOAP over HTTPS连接方式

1. 功能简介

使用 NETCONF over SOAP over HTTP 或 NETCONF over SOAP over HTTPS 连接方式时，配置工具将配置指令封装成 SOAP 报文后通过 HTTP 或 HTTPS 协议传输到设备。

2. 配置限制和指导

本文仅介绍基本配置，有关 SOAP 报文的 DSCP 优先级、NETCONF 客户端访问控制、NETCONF 用户使用的认证域等相关内容请参见产品对应版本的 NETCONF 配置、NETCONF 命令手册。

3. 开启NETCONF over SOAP功能

(1) 进入系统视图。

system-view

(2) 开启 NETCONF over SOAP 功能。

（非 FIPS 模式）

netconf soap { http | https } enable

（FIPS 模式）

netconf soap https enable

缺省情况下，NETCONF over SOAP 处于关闭状态。

(3) （可选）开启 NETCONF 日志功能。

netconf log source { all | { agent | soap | web } * } { protocol-operation { all | { action | config | get | set | session | syntax | others } * } | row-operation | verbose }

缺省情况下，NETCONF 日志功能处于关闭状态。

(4) 通过配置工具与设备建立 NETCONF over SOAP 会话。关于配置工具的使用方法，具体参见配置工具的配置指导。

6 为NETCONF客户端与设备建立连接

6.1 配置限制和指导

建立 NETCONF 会话后，客户端需要先与设备进行能力集交互，完成能力集的交互后，设备才会处理客户端发送的其他请求。能力集的表示形式为 URI（Uniform Resource Identifier，统一资源标识符），例如“urn:ietf:params:netconf:base:1.0”。客户端支持的能力集以客户端实际情况为准。客户端发送 Hello 报文时，请根据实际情况指定能力集。本文以客户端指定能力集 urn:ietf:params:netconf:base:1.0 为例进行举例。

多个用户同时配置设备时，可能会导致用户配置与配置结果不一致，因此，请避免多个用户同时配置设备。

设备同一时间内允许建立的最大会话数可以通过 **aaa session-limit** 命令配置，关于该命令的详细描述，请参见“安全配置指导”中的“AAA”。用户数超过上限后，新登录的用户将登录失败。

6.2 NETCONF方式

适用于 NETCONF over SSH 配置工具的方式。

1. 客户端向设备发送Hello

客户端收到设备发送的能力集协商报文后，需要给设备发送如下格式的报文，告知设备客户端支持哪些 NETCONF 能力集。

Hello 协商报文格式如下：

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

说明

一个<capability>和</capability>选项对中只能填写一个能力集。可以使用多个<capability>和</capability>选项对，配置多个能力集。

2. 设备向客户端回复能力集

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
    更多能力集显示略，以设备的实际回复为准

<capability>http://www.h3c.com/netconf/action:1.0-Configuration?module=Configuration&amp
;revision=2015-05-07</capability>
  </capabilities>
  <session-id>1</session-id>
</hello>
```

<capabilities>和</capabilities>之间的内容表示设备支持的能力集，以具体设备实际情况为准。

<session-id>和</session-id>之间的内容表示为本次会话分配的会话 ID，用来唯一标识本次会话。

6.3 NETCONF over SOAP方式

适用于 NETCONF over SOAP 配置工具的方式。

1. 客户端向设备发送Hello

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  <env:Header>
    <auth:Authentication env:mustUnderstand="1"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
      <auth:UserName>test</auth:UserName>
      <auth:Password>test</auth:Password>
      <auth:Language>zh-cn</auth:Language>
    </auth:Authentication>
  </env:Header>
```

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

说明

一个<capability>和</capability>选项对中只能填写一个能力集。可以使用多个<capability>和</capability>选项对，配置多个能力集。

2. 设备向客户端回复能力集

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header>
      <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100002fbf58da6797490a3ef11142d879212</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
      <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <capabilities>
            <capability>urn:ietf:params:netconf:base:1.0</capability>
            更多能力集显示略，以设备的实际回复为准

<capability>http://www.h3c.com/netconf/action:1.0-Configuration?module=Configuration&amp
;revision=2015-05-07</capability>
         </capabilities>
         <session-id>2</session-id>
      </hello>
   </env:Body>
</env:Envelope>
```

<capabilities>和</capabilities>之间的内容表示设备支持的能力集，以具体设备实际情况为准。

<session-id>和</session-id>之间的内容表示为本次会话分配的会话 ID，用来唯一标识本次会话。

7 构造NETCONF请求

7.1 NETCONF API文档介绍

7.1.1 文档组织

NETCONF API 文档分属于三个命名空间：

- data 命名空间：提供系统的运行状态数据和配置数据，为只读，支持下发 get 和 get-bulk 操作。

- config 命名空间：提供系统的配置数据信息，可读写，支持下发 get-config、get-bulk-config 和 edit-config 操作。
- action 命名空间：通常提供系统非配置类的操作和运行状态数据（如 ping、reset 操作），可读写，支持下发 action 操作。

每一个支持 NETCONF 的模块都包括一个或者多个 NETCONF API 文档，分别描述其在 data 命名空间、config 命名空间和 action 命名空间支持的功能。文档命名遵循如下格式，其中 XXX 代表模块名：

- data 命名空间：Comware V7 XXX NETCONF XML API Data Reference.docx
- config 命名空间：Comware V7 XXX NETCONF XML API Configuration Reference.docx
- action 命名空间：Comware V7 XXX NETCONF XML API Action Reference.docx

7.1.2 文档内容说明

每一个 NETCONF API 文档对应一个功能模块，如 ARP、DHCP。每个模块由一个或者多个表组成。表以行和列的形式进行组织：

- 行：表示一个对象实例。
- 列：表示每个对象实例中包含的信息。

以 ARP 表为例，一条 ARP 表项为一行，ARP 表项中的 IP 地址、MAC 地址、所属 VLAN 等为列。

在 NETCONF API 文档中，每个表包含下面几个部分的信息：

- 表名称：提供了从模块开始的路径和最终的表名，如：ACL/Base
- 表的 XML 结构（XML structure）：列举了本表从模块开始的 XML 表示方式，但不包括值信息，例如：

XML structure

```
<VLAN>
  <BatchVlans>
    <CreateVlanList></CreateVlanList>
    <DestroyVlanList></DestroyVlanList>
  </BatchVlans>
</VLAN>
```

- 表描述（Table description）：提供模块名称、表名称、表类型、行名称和约束信息。其中，表类型取值包括：
  - Multi-instance table：多实例表格，表示该表可以包含多行。
  - Single-instance table：单实例表格，表示该表能包括一行。
- 列详细信息（Columns）：提供列名称、列描述、列数据类型和约束条件等信息。列数据类型取值包括：
  - Index：表示该列为表格的索引列。
  - N/A：表示该列不作为索引列。

- 应答的 XML 结构（Responsed XML structure）：列举了本表从模块开始应答消息的 XML 表示方式，但不包括值信息。客户端可以依据返回结果来判断操作执行的效果。仅 action 命名空间中可以返回操作结果的表提供此信息。例如：

Responsed XML structure

```
<VLAN>
  <BatchVlans>
    <CreateVlanList></CreateVlanList>
    <DestroyVlanList></DestroyVlanList>
  </BatchVlans>
</VLAN>
```

- 应答列详细信息（Responsed columns）：用来描述应答 XML 结构中的各列。

提示

Comware V7 NETCONF API 文档描述的是全集，需要结合设备的 XSD 文件来查看当前设备的支持情况。设备可能不支持某些功能、某些表格或某些列。

7.2 使用NETCONF API文档构造NETCONF请求

NETCONF 请求构造方法为：

(1) 构造 XML 请求的协议定义部分，即 top 外层的部分。

以 get 操作为例，top 以外的部分为协议定义部分，可以直接从现有请求或者本文档拷贝。

```
<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <top xmlns="http://www.h3c.com/netconf/data:1.0">
          指定模块，子模块，表名，列名
      </top>
    </filter>
  </get>
</rpc>
```

其他操作的协议定义部分格式，请参见“7.4 Comware V7 中支持的NETCONF操作类型”。

(2) 构造 top 元素。

top 元素的格式通常为：<top xmlns="http://www.h3c.com/netconf/data:1.0">。其中，http://www.h3c.com/netconf/data:1.0 为 H3C 自定义的命名空间，需要根据需要设置为 data、config 或 action 命名空间，即 data:1.0 也可根据需要设置为 config:1.0 或 action:1.0。

(3) 构造模块操作部分。

拷贝 NETCONF API 文档中表格 XML 结构到“指定模块，子模块，表名，列名”部分，然后依据实际情况删除不需要的列或者为已知列赋值。

7.3 构造NETCONF请求示例

config命名空间中ARP模块的ArpConfig表用来配置ARP表项，其XML结构如 图 2 所示。该表中包括ARP表项的IP地址、MAC地址、所属VPN实例、所属VLAN、出接口、表项类型信息。

图2 ARP/ArpConfig 表的 XML 结构

ARP/ArpConfig

This table contains ARP configuration information.

XML structure

```
<ARP>
  <ArpConfig>
    <Config>
      <VrfIndex></VrfIndex>
      <Ipv4Address></Ipv4Address>
      <MacAddress></MacAddress>
      <VLANID></VLANID>
      <PortIndex></PortIndex>
      <ArpType></ArpType>
    </Config>
  </ArpConfig>
</ARP>
```

通过 NETCONF 创建公网中 IP 地址为 1.1.1.1、MAC 地址为 1-1-1、所属 VLAN 为 VLAN 10、出接口索引为 20 的静态 ARP 表项时，需要采用 edit-config:merge 操作。该操作使用的 XML 请求为：

```
<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" xc:operation="merge">
        <ARP>
          <ArpConfig>
            <Config>
              <VrfIndex>0</VrfIndex>
              <Ipv4Address>1.1.1.1</Ipv4Address>
              <MacAddress>00-01-00-01-00-02</MacAddress>
              <VLANID>10</VLANID>
              <PortIndex>20</PortIndex>
              <ArpType>1</ArpType>
            </Config>
          </ArpConfig>
        </ARP>
      </top>
    </config>
  </edit-config>
</rpc>
```

7.4 Comware V7中支持的NETCONF操作类型

Comware V7 平台对NETCONF标准协议做了一些修订，删除了不常用的操作，增加部分新的操作，如 表 3 所示。

表3 NETCONF 协议支持的操作

操作	说明	XML 格式样例
get	获取数据，包括运行状态数据和配置数据	获取Syslog模块的全部数据的XML请求如下： <rpc message-id="101" 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"> <Syslog> </Syslog> </top> </filter> </get> </rpc>
get-config	获取配置数据，和get不同，它只返回非缺省的配置数据。如果未配置数据，则返回一个空的<data>	获取接口表内所有配置的XML请求如下： <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/> </Interfaces> </Ifmgr> </top> </filter> </get-config> </rpc>

操作	说明	XML 格式样例
get-bulk	从指定索引的下一条开始批量获取后续N条数据（索引行数据不返回），包括运行状态数据和配置数据。用户通过index属性指定索引，通过count属性指定N。如未指定索引，则以第一条为索引；如未指定N，或者数据表中符合条件的数据记录不足N条，则返回表中所有剩下的数据条目 get操作会返回所有符合条件的数据，这在某些情况下会导致效率问题。get-bulk允许用户从固定数据项开始，向后获取指定条目的数据记录	取全部接口的数据的xml请求如下： <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-bulk> <filter type="subtree"> <top xmlns="http://www.h3c.com/netconf/data:1.0"> <Ifmgr> <Interfaces h3c:count="5"> <Interface/> </Interfaces> </Ifmgr> </top> </filter> </get-bulk> </rpc>
get-bulk-config	从指定索引的下一条批量获取配置数据。和get-config类似，只返回非默认配置；其他约束类似get-bulk	获取全部接口配置信息的xml请求如下： <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>

操作	说明	XML 格式样例

edit-config *增量下发*	增量下发选项incremental放置在列上，对于类似vlan permitlist列表集合性质的列，可能支持增量下发，用户请求XML中有增量下发选项时，最终执行结果不影响本列原有的数据。
	增量下发只支持edit-config，但不支持edit-config中的replace。
	不是所有模块都支持增量下发，具体请参见模块对应的NETCONF XML API，查找其是否支持增量下发
		下发一个接口的VLAN配置，使用增量下发，262接口原有untagvlanlist为12～15，下发后为1～10，12～15。XML请求如下：
		`<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:h3c="http://www.h3c.com/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" xc:operation="merge">`
		`<VLAN>`
		`<HybridInterfaces>`
		`<Interface>`
		`<IfIndex>262</IfIndex>`
		`<UntaggedVlanList h3c:incremental="true">1-10</UntaggedVlanList>`
		`</Interface>`
		`</HybridInterfaces>`
		`</VLAN>`
		`</top>`
		`</config>`
		`</edit-config>`
		`</rpc>`

edit-config: merge	在当前运行配置的基础上直接运行指定配置
	merge操作必须指定具体的操作对象（行）：
	• 如果指定的对象存在，则直接配置该对象
	• 如果指定的对象不存在，但允许创建，则先创建再配置该对象
	• 如果指定的对象不存在且不允许创建，则返回 merge 失败
		将BufferSize设置为120的xml请求如下：
		`<rpc message-id="101" 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" xc:operation="merge">`
		`<Syslog>`
		`<LogBuffer>`
		`<BufferSize>120</BufferSize>`
		`</LogBuffer>`
		`</Syslog>`
		`</top>`
		`</config>`
		`</edit-config>`
		`</rpc>`

操作	说明	XML 格式样例
edit-config: create	创建指定对象。create操作必须指定配置对象。create操作的XML数据格式和merge类似，只是operation属性需要指定为“create” • 如果当前配置表支持创建对象，且当前对象不存在，则先创建配置对象，再创建指定的配置 • 如果配置对象下对应的配置项已经存在，则返回 data-exist 错误	同上，把merge修改为create即可
edit-config: replace	• 如果指定的对象存在，则替换指定对象的配置为当前配置 • 如果指定的对象不存在，但允许创建，则先创建再配置该对象为当前配置 • 如果指定对象不存在且不允许创建，则不进行 replace 操作，返回 invalid-value 错误，提示用户配置对象不存在	同上，把merge修改为replace即可
edit-config: remove	删除指定配置 • 当指定的删除对象中只有表索引时，则删除此配置指定对象的所有配置，同时删除指定对象 • 当指定的删除对象中不仅仅有表索引还存在配置项时，则删除此对象下面的指定配置 • 如果系统中指定对象不存在，或者 XML 消息未指定对象，则直接返回成功	同上，把merge修改为remove即可
edit-config: delete	删除指定配置 • 当指定的删除对象中只有表索引时，则删除此配置指定对象的所有配置，同时删除指定对象 • 当指定的删除对象中不仅仅有表索引还存在配置项时，则删除此对象下面的指定配置 • 如果系统中指定对象不存在，则直接返回不存在的错误消息	同上，把merge修改为delete即可

操作	说明	XML 格式样例

edit-config *默认操作选项*	edit-config操作用于修改当前系统配置。NETCONF定制了四种修改配置的方式：merge、create、delete和replace。当XML消息中未指定修改配置方式的时候，则使用默认操作做为当前指令的操作方式，不会修改默认操作的缺省值
	默认操作的缺省值是merge，在XML消息中可以通过<default-operation>节点来设置默认操作，取值为：
	• merge：当配置方式和默认操作方式均未指定时，使用该方式
	• replace：当配置方式未指定，默认操作指定为 replace 的时候，edit-config 操作会默认为 replace 操作
	• none：当配置方式未指定，默认操作指定为 none 的时候，edit-config 操作会默认为 none 操作。none 操作主要用来检查，下发为 none 操作的配置仅仅做 Schema 校验，不进行真正的配置下发。语法检查通过，就返回 ok 成功，否则失败
		下发一个空的操作，该操作仅仅验证格式，并不真正下发给系统，xml请求如下：
		`<rpc message-id="101" 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>`
		`<default-operation>none</default-operation>`
		`    <config>`
		`      <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>`

操作	说明	XML 格式样例
edit-config *默认错误处理选项*	edit-config将指定的配置设置到系统上，完成配置设置的操作。在执行edit-config的过程中，如果遇到一个实例配置出错，默认情况下会直接返回错误，但是为了使我们的应用更加灵活，edit-config为我们提供了错误选项，通过错误选项取值的不同，在发生错误的时候进行不同的处理操作 <error-option>节点用于设置一个实例配置出错后，后续实例配置的处理方式，缺省值为stop-on-error，全部取值为： • stop-on-error：停止处理，返回错误。此选项为默认选项 • continue-on-error：继续处理，但是报告错误 • rollback-on-error：停止并回滚配置	下发两个接口的配置，当下发第一个接口的配置发生错误时，继续进行下一个接口配置的下发，XML请求如下： <rpc message-id="101" 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> <error-option>continue-on-error</error-option> <config> <top xmlns="http://www.h3c.com/netconf/config:1.0" xc:operation="merge"> <Ifmgr> <Interfaces> <Interface> <IfIndex>262</IfIndex> <Description>222</Description> <ConfigSpeed>1024</ConfigSpeed> <ConfigDuplex>1</ConfigDuplex> </Interface> <Interface> <IfIndex>263</IfIndex> <Description>333</Description> <ConfigSpeed>1024</ConfigSpeed> <ConfigDuplex>1</ConfigDuplex> </Interface> </Interfaces> </Ifmgr> </top> </config> </edit-config> </rpc>

操作	说明	XML 格式样例
edit-config *测试处理选项*	在真正执行edit-config操作时，可指定一个测试选项，使用<test-option>节点来决定当前配置是否真正下发。该节点的缺省值为test-then-set，全部取值为： • test-then-set：如果没有错误则将配置设置到系统 • set：将配置设置到系统 • test-only：只测试，并不下发配置到系统。语法检查通过，就返回 ok 成功，否则失败	下发一个接口的配置，仅测试，XML请求如下： <rpc message-id="101" 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> <test-option>test-only</test-option> <config> <top xmlns="http://www.h3c.com/netconf/config:1.0" xc:operation="merge"> <Ifmgr> <Interfaces> <Interface> <IfIndex>262</IfIndex> <Description>222</Description> <ConfigSpeed>2</ConfigSpeed> <ConfigDuplex>1</ConfigDuplex> </Interface> </Interfaces> </Ifmgr> </top> </config> </edit-config> </rpc>
action	下发非配置数据的设置动作，比如，reset操作	清除全部接口的统计信息，XML请求如下： <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	锁保护的是配置数据，即edit-config中可以指定的那些模块的配置数据，其他操作不受锁的限制 NETCONF锁仅仅保护NETCONF 会话，不保护SNMP等其他请求下发的配置	禁止NETCONF会话修改设备的当前配置，XML请求如下： <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <lock> <target> <running/> </target> </lock> </rpc>

操作	说明	XML 格式样例
unlock	取消锁保护 当会话结束时锁也会被自动释放	取消锁保护，允许NETCONF会话修改设备的当前配置： <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <unlock> <target> <running/> </target> </unlock> </rpc>
get-sessions	获取当前系统中所有NETCONF会话的信息（不能指定sessions-ID）	获取当前系统中所有NETCONF会话的信息： <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get-sessions/> </rpc>
close-session	关闭当前NETCONF会话，并释放锁和这个session用到的内部资源（如内存等），退出XML视图	关闭当前NETCONF会话，并释放锁和这个session用到的内部资源（如内存等），退出XML视图 <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <close-session/> </rpc>
kill-session	关闭其他NETCONF会话，不支持关闭用户自己的NETCONF会话	关闭session-id为1的NETCONF会话： <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <kill-session> <session-id>1</session-id> </kill-session> </rpc>

操作	说明	XML 格式样例

CLI	执行命令行的命令。请求消息将命令行语句封装在<CLI>标签中，命令行输出信息被封装在<CLI>标签中返回
	CLI支持Configuration和Execution方式执行命令行：
	• Execution：在用户视图下执行命令行
	• Configuration：在系统视图下执行命令行。使用该方式时需要指定exec-use-channel的属性：
	o false：不使用channel方式执行命令行
	o true：使用临时channel执行命令行，执行完成后，自动关闭该channel
	o persist：使用保留channel执行命令行。使用该方式时，则需要执行Open-channel操作打开当前对话的channel；使用完成后，执行Close-channel关闭Channel。如果未执行Open-channel操作，则系统将自动打开channel；如果未执行关闭保留channel操作，再次使用保留channel时，系统在上次最后执行的命令行所在视图再次执行命令行
	• 对于其他视图下命令，则需要在Configuration下先指定进子视图的命令，再指定配置命令
	一个NETCONF会话只能打开一个保留channel，可以打开多个临时channel
	不支持执行交互式命令
	使用channel方式执行命令行时，不支持执行**quit**命令退出用户视图
		在系统视图下不使用channel方式执行**telnet server enable**命令：
		`<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
		`<CLI>`
		`<Configuration exec-use-channel="false">telnet server enable</Configuration>`
		`</CLI>`
		`</rpc>`

save	保存系统运行配置。save操作可以使用子元素<file>来指定保存的配置文件名称。当save操作中不存在子元素<file>列时，则设备会自动将当前运行配置保存到主用下次启动配置文件中。OverWrite属性用来判断当指定的配置文件名存在时，当前配置是否覆盖原配置文件并保存成功。Binary-only属性用来只保存当前使用的二进制配置文件	将设备当前配置保存到文件test.cfg中：
		`<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	配置加载。<load>操作执行后，指定文件中的配置会被合并到设备的当前配置中	将文件a1.cfg中的配置合并到设备的当前配置中：
		`<rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
		`<load>`
		`<file>a1.cfg</file>`
		`</load>`
		`</rpc>`

操作	说明	XML 格式样例
rollback	配置回滚。<rollback>操作必须使用子元素<file>来指定需要回滚的配置文件名称<rollback>操作执行后，当前系统运行配置会被完全替换为指定文件中所描述的配置	将设备当前配置回退到文件1A.cfg中配置的状态： <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <rollback> <file>1A.cfg</file> </rollback> </rpc>

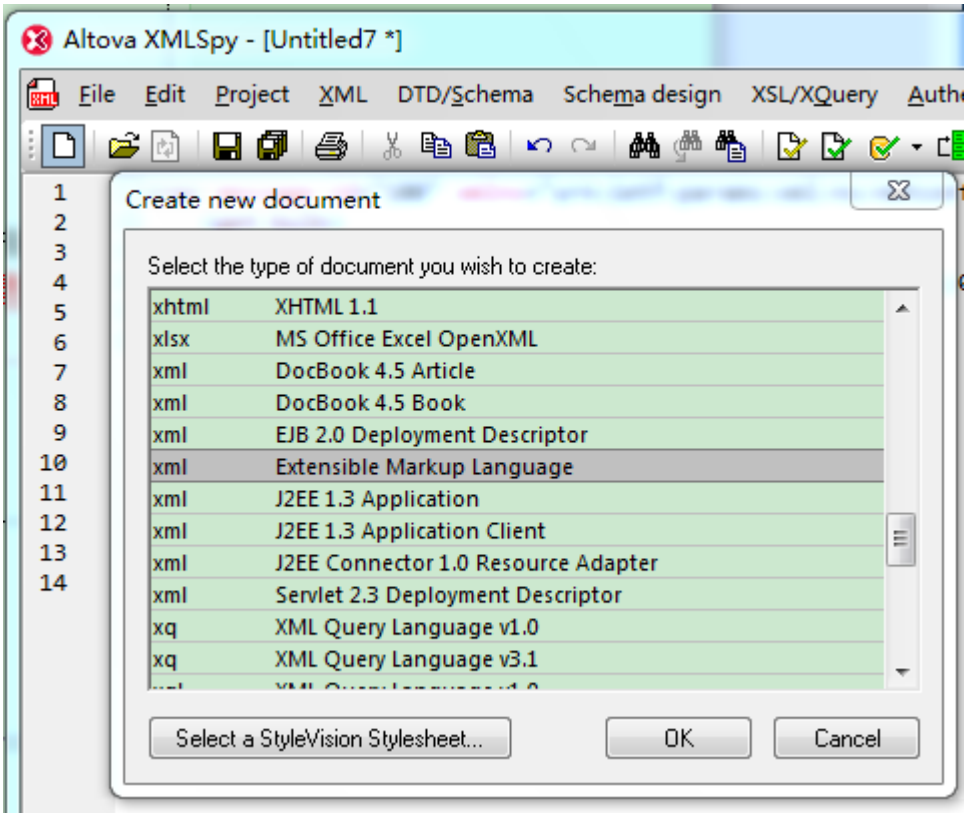
8 使用XML工具简单验证构造的XML正确性

在下发请求前，建议先利用 XSD 使用工具验证下请求的 XML 设备是否支持，是否存在格式问题。本文使用 Altova XMLSpy 来验证 XML 是否正确，也可以使用其他支持 XML 验证的工具。

(1) 把设备 XSD 文件拷贝出来，放在一个目录中。XSD 文件随软件版本发布。

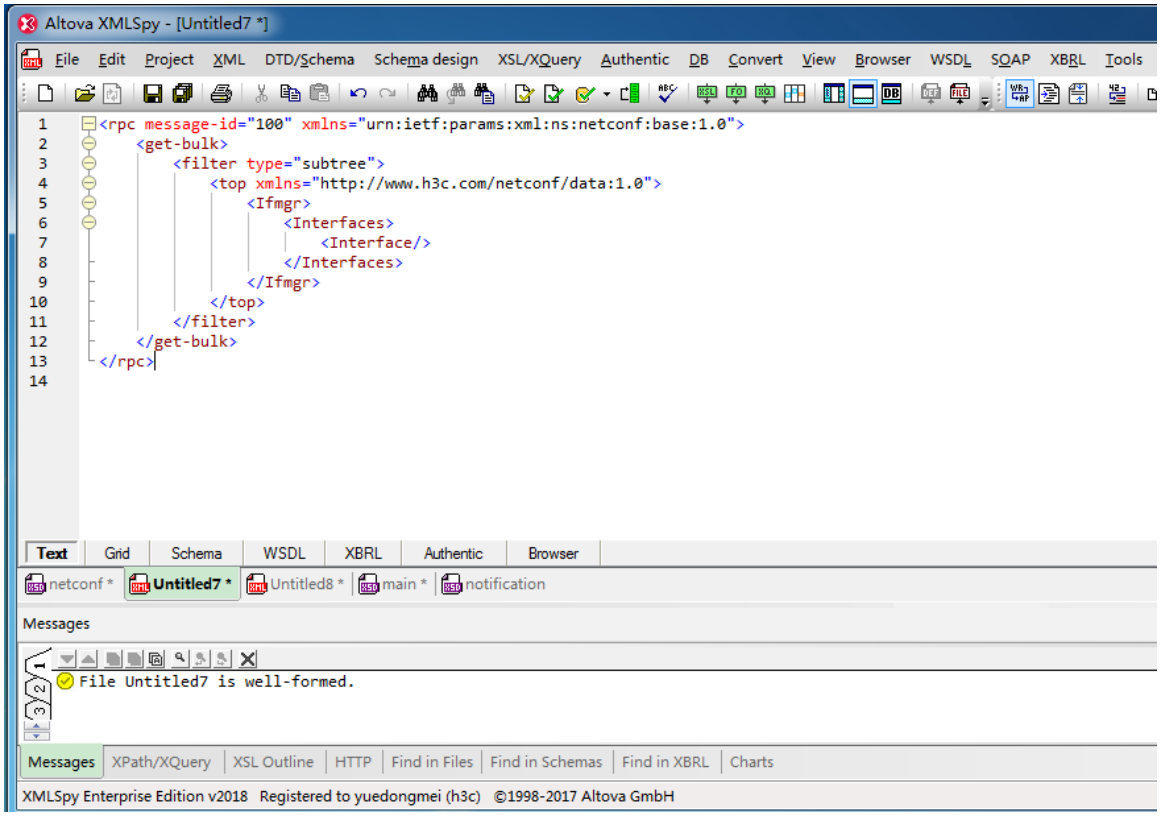
(2) 新建 XML 文件，将待验证 NETCONF 报文粘贴到该 XML 文件里。

图3 新建 XML 文件



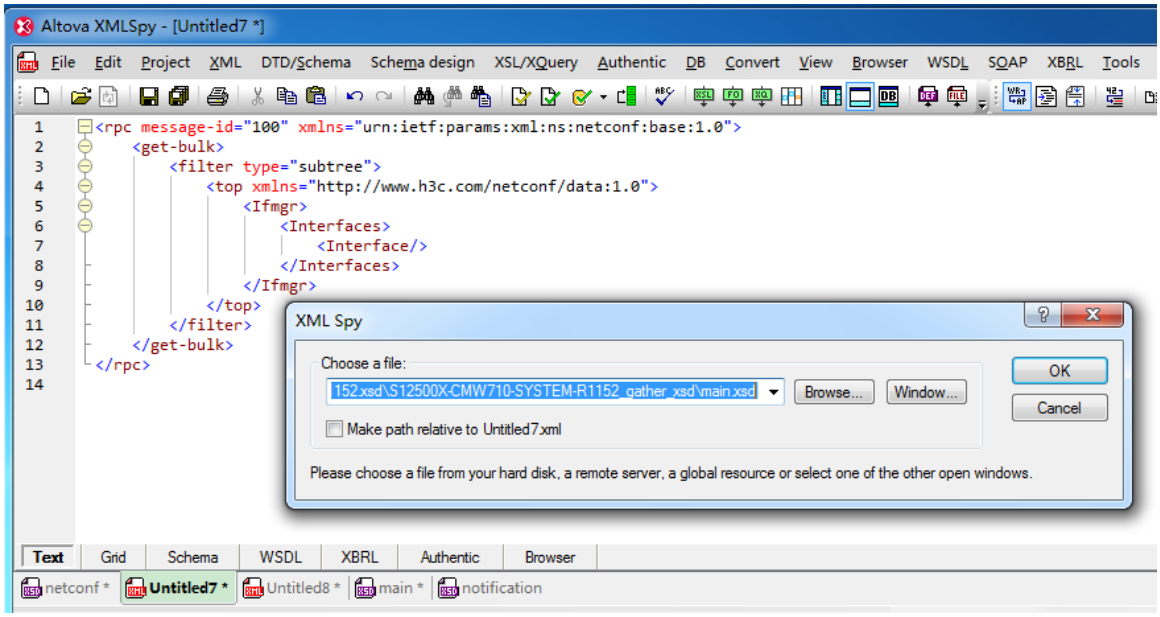
(3) 点击 XML—〉Check Formedness 菜单验证 XML 格式是否良好。如果有 XML 格式问题，请根据下发 Messages 处提示信息修改。

图4 验证 XML 格式是否良好



(4) 根据XSD文件验证报文是否合法。点击DTD/Schema —〉Assign Schema菜单，选择main.xsd。

图5 选择 XSD 文件



(5) 点击 XML—〉Validate XML 菜单验证 XML 是否合法。如果 Messages 处有错误提示，请根据提示进行修改。修改完成后再次验证，直到通过验证。

图6 验证 XML 是否合法（合法时）

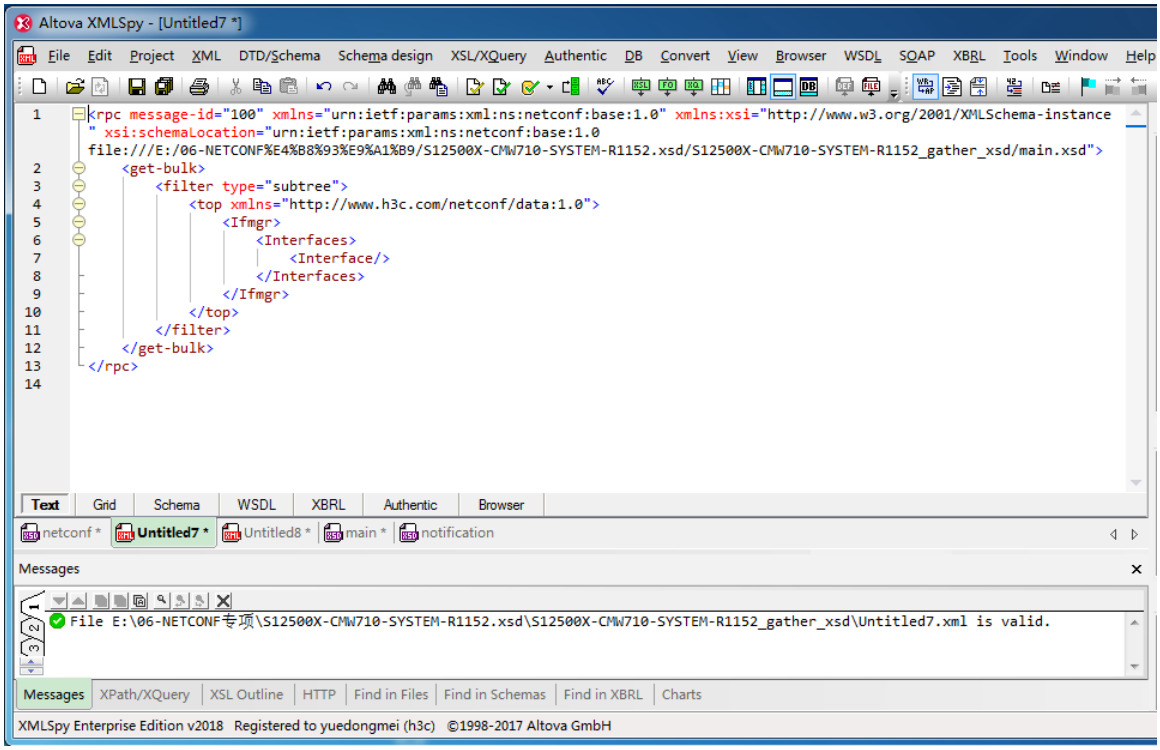
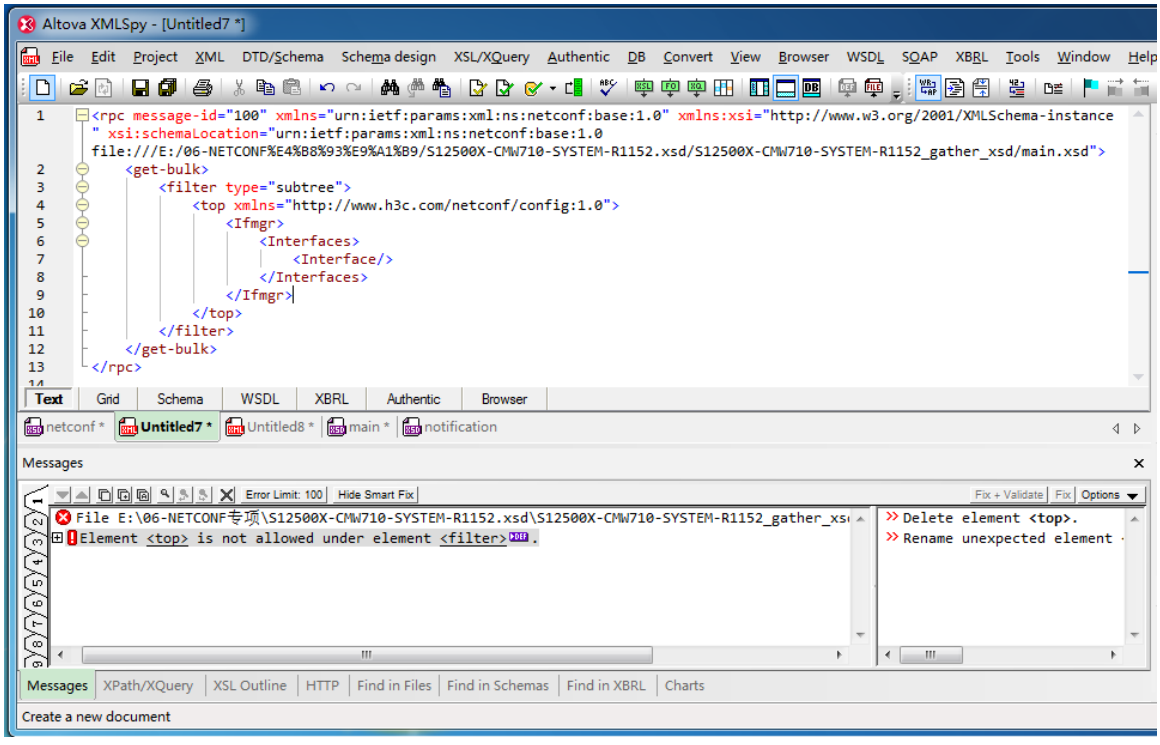


图7 验证 XML 是否合法（不合法时）



9 正式下发配置指令

使用 XSD 验证 XML 报文正确后，即可正式通过配置工具下发配置指令。

10 关闭NETCONF会话

1. 客户端发送报文

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

2. 结果验证

设备向客户端应答如下报文，但此时由于关闭了会话，客户端可能收不到此报文。

```
<?xml version="1.0" encoding="UTF-8" ?>
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <ok/>
</rpc-reply>
```

11 NETCONF典型配置举例

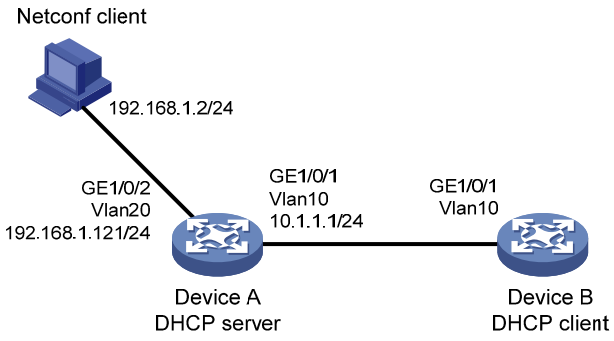
11.1 通过NETCONF配置DHCP服务器和DHCP客户端

11.1.1 组网需求

在主机上安装 NETCONF 客户端软件，通过 NETCONF 配置 Device A 作为 DHCP 服务器、Device B 作为 DHCP 客户端。DHCP 服务器可以为 DHCP 客户端分配以下网络参数：

- 网段 10.1.1.0/24 内的 IP 地址。
- IP 地址租约为 1 天。
- 网关地址为 10.1.1.1。
- DNS 服务器地址为 10.1.1.10。

图8 通过 NETCONF 配置 DHCP 服务器组网图



11.1.2 在Device A上配置NETCONF功能

(1) 开启 NETCONF over SOAP 功能。

开启 NETCONF over SOAP 功能。

```
<DeviceA> system-view
[DeviceA] netconf soap http enable
```

(2) 配置用户 admin 可以通过 NETCONF 操作 DHCP 和接口管理 Ifmgr 模块。

创建用户角色 dhcp-ifmgr-vlan，指定该用户角色可以读、写、执行 DHCP 和 Ifmgr 模块的 XML 元素。

```
[DeviceA] role name dhcp-ifmgr-vlan
[DeviceA-role-dhcp-ifmgr-vlan] rule 1 permit command xml
[DeviceA-role-dhcp-ifmgr-vlan] rule 2 permit read write execute xml-element rpc/
[DeviceA-role-dhcp-ifmgr-vlan] rule 3 permit read write execute xml-element dhcp/
[DeviceA-role-dhcp-ifmgr-vlan] rule 4 permit read write execute xml-element ifmgr/
[DeviceA-role-dhcp-ifmgr-vlan] rule 5 permit read write execute xml-element vlan/
[DeviceA-role-dhcp-ifmgr-vlan] quit
```

创建设备管理类本地用户 admin，设置其密码为 admin、服务类型为 HTTP。

```
[DeviceA] local-user admin
[DeviceA-luser-manage-admin] password simple admin
[DeviceA-luser-manage-admin] service-type http
```

配置为用户 admin 授权的用户角色为 dhcp-ifmgr-vlan。

```
[DeviceA-luser-manage-admin] authorization-attribute user-role dhcp-ifmgr-vlan
```

11.1.3 通过NETCONF客户端配置DHCP服务器

(1) 配置 NETCONF 客户端软件，配置方法请参见客户端软件相关指导，具体配置过程略。

(2) 向 Device A 发送 Hello

在 TestCase-DHCP 中添加步骤 Hello，XML 内容如下。其中，用户名为 admin，密码为 admin。所有步骤窗口中的 URL 地址均需要设置为：http://192.168.1.121/soap/netconf/。

```
<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:UserName>admin</auth:UserName>
         <auth:Password>admin</auth:Password>
         <auth:Language>en</auth:Language>
      </auth:Authentication>
   </env:Header>
   <env:Body>
      <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <capabilities>
            <capability>urn:ietf:params:netconf:base:1.0</capability>
         </capabilities>
      </hello>
   </env:Body>
</env:Envelope>
```

接收到的应答消息中，auth:AuthInfo 为 100001ac479ef0988d587b74e787a0917f69。后续的请求报文都需要携带该认证信息。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
      <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
      <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <capabilities>
            <capability>urn:ietf:params:netconf:base:1.0</capability>
            ...省略若干 capability...
         </capabilities>
         <session-id>1</session-id>
      </hello>
   </env:Body>
</env:Envelope>
```

(3) 获取接口 GigabitEthernet1/0/1 和 GigabitEthernet1/0/2 的接口索引值。

创建步骤 GetIfINdex，向 Device A 发送请求获取接口索引值。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
         <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
     <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>
                   <IfIndex></IfIndex>
                   <Name>GigabitEthernet1/0/1</Name>
                 </Interface>
                 <Interface>
                   <IfIndex></IfIndex>
                   <Name>GigabitEthernet1/0/2</Name>
                 </Interface>
               </Interfaces>
             </Ifmgr>
             </top>
          </filter>
       </get>
```

```
     </rpc>
   </env:Body>
</env:Envelope>
```

接收到的应答消息，获知接口 GigabitEthernet1/0/1 和 GigabitEthernet1/0/2 的索引值分别为 1 和 2。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
      <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
      <rpc-reply message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
         <data>
            <top xmlns="http://www.h3c.com/netconf/data:1.0">
               <Ifmgr>
                  <Interfaces>
                     <Interface>
                        <IfIndex>1</IfIndex>
                        <Name>GigabitEthernet1/0/1</Name>
                     </Interface>
                  </Interfaces>
               </Ifmgr>
            </top>
         </data>
      </rpc-reply>
   </env:Body>
</env:Envelope>
```

(4) 创建 VLAN，将接口加入 VLAN，创建 VLAN 接口，并配置 VLAN 接口的 IP 地址。

创建步骤 CreateVLAN，在 Device A 上创建 VLAN 10 和 20，将接口 GigabitEthernet1/0/1 和 GigabitEthernet1/0/2 分别加入 VLAN 10 和 20。创建 VLAN 接口 10 和 20，配置其 IP 地址分别为 10.1.1.1/24 和 192.168.1.1/24。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
         <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
<rpc message-id ="101" 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" xc:operation="merge">
        <VLAN>
          <VLANs>
            <VLANID>
              <ID>10</ID>
              <AccessPortList>1</AccessPortList>
              <Ipv4>
                <Ipv4Address>10.1.1.1</Ipv4Address>
                <Ipv4Mask>255.255.255.0</Ipv4Mask>
              </Ipv4>
            </VLANID>
              <VLANID>
              <ID>20</ID>
              <AccessPortList>2</AccessPortList>
              <Ipv4>
                <Ipv4Address>192.168.2.1</Ipv4Address>
                <Ipv4Mask>255.255.255.0</Ipv4Mask>
              </Ipv4>
            </VLANID>
          </VLANs>
        </VLAN>
      </top>
    </config>
  </edit-config>
</rpc>
```

接收到应答消息，表明配置成功。

```
   </env:Body>
     </env:Envelope>
       <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
        <env:Header>
           <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
              <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
           </auth:Authentication>
        </env:Header>
        <env:Body>
      <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ok/>
      </rpc-reply>
   </env:Body>
</env:Envelope>
```

(5)　创建 DHCP 地址池

创建步骤 CreatePool，在 Device A 上创建 DHCP 地址池 pool1，该地址池分配 10.1.1.0/24 网段地址、IP 地址租约为 1 天、网关地址为 10.1.1.1、DNS 服务器地址为 10.1.1.10。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
```

```
    <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
      <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
  </env:Header>
  <env:Body>
    <rpc message-id ="101" 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" xc:operation="create">
            <DHCP>
              <DHCPServerIpPool>
                <IpPool>
                  <PoolIndex>1</PoolIndex>
                  <PoolName>pool1</PoolName>
                  <NetworkIpv4Address>10.1.1.0</NetworkIpv4Address>
                  <NetworkIpv4Mask>255.255.255.0</NetworkIpv4Mask>
                  <LeaseDay>1</LeaseDay>
                  <LeaseHour>0</LeaseHour>
                  <LeaseMinute>0</LeaseMinute>
                  <LeaseSecond>0</LeaseSecond>
                  <GatewayIpv4Address>10.1.1.1</GatewayIpv4Address>
                  <DNSIpv4Address>10.1.1.10</DNSIpv4Address>
                </IpPool>
              </DHCPServerIpPool>
            </DHCP>
          </top>
        </config>
      </edit-config>
    </rpc>
  </env:Body>
</env:Envelope>
```

接收到应答消息，表明配置成功。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
      <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
      <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ok/>
      </rpc-reply>
```

```
   </env:Body>
</env:Envelope>
```

(6) 开启 DHCP 服务。

创建步骤 DHCP-Enable，开启 DHCP 服务。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
      <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
    </auth:Authentication>
  </env:Header>
  <env:Body>
    <rpc message-id ="101" 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" xc:operation="create">
            <DHCP>
              <DHCPConfig>
                <DHCPEnable>1</DHCPEnable>
              </DHCPConfig>
            </DHCP>
          </top>
        </config>
      </edit-config>
    </rpc>
  </env:Body>
</env:Envelope>
```

接收到应答消息，表明配置成功。

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
      <auth:Authentication env:mustUnderstand="true"
xmlns:auth="http://www.h3c.com/netconf/base:1.0">
         <auth:AuthInfo>100001ac479ef0988d587b74e787a0917f69</auth:AuthInfo>
      </auth:Authentication>
   </env:Header>
   <env:Body>
      <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
         <ok/>
      </rpc-reply>
   </env:Body>
</env:Envelope>
```

11.1.4 在Device B上通过CLI配置DHCP客户端功能

创建 VLAN 10，将接口 GigabitEthernet1/0/1 加入 VLAN 10。

```
<DeviceB> system-view
[DeviceB] vlan 10
[DeviceB-vlan10] port GigabitEthernet 1/0/1
[DeviceB-vlan10] quit
```

创建 VLAN 接口 10，配置该接口通过 DHCP 获取 IP 地址。

```
[DeviceB] interface vlan-interface 10
[DeviceB-Vlan-interface10] ip address dhcp-alloc
[DeviceB-Vlan-interface10] quit
```

11.1.5 验证配置

在 Device B 上查看接口 IP 地址，可以看到 VLAN 接口 10 获取到 IP 地址 10.1.1.2。

```
[DeviceB] display ip interface brief
*down: administratively down
 (s): spoofing  (l): loopback
Interface            Physical Protocol IP address      VPN instance Description
Vlan10               up       up       10.1.1.2        --           --
```

在 Device B 上查看 DHCP 客户端信息，可以看到 VLAN 接口 10 获取到的参数与 DHCP 服务器上的配置一致。

```
[DeviceB] display dhcp client verbose
Vlan-interface10 DHCP client information:
  Current state: BOUND
  Allocated IP: 10.1.1.2 255.255.255.0
  Allocated lease: 86400 seconds, T1: 38637 seconds, T2: 75600 seconds
  Lease from Jan  1 08:54:48 2013   to   Jan  2 08:54:48 2013
  DHCP server: 10.1.1.1
  Transaction ID: 0x81876da6
  Default router: 10.1.1.1
  DNS servers: 10.1.1.10
  Client ID type: ascii(type value=00)
  Client ID value: 00e0.fc00.511a-Vlan10
  Client ID (with type) hex: 0030-3065-302e-6663-
                             3030-2e35-3131-612d-
                             566c-616e-3130
  T1 will timeout in 0 days 10 hours 40 minutes 20 seconds
```

11.2 基于ncclient工具的NETCONF over SSH配置举例

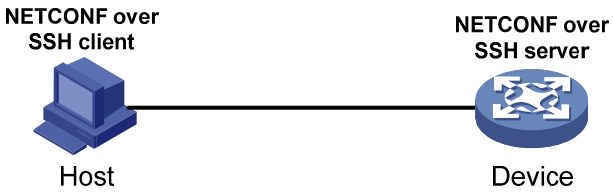
11.2.1 组网需求

在主机 Host 上安装开源工具 ncclient，通过主机 Host 与 Device 设备建立 NETCONF over SSH 会话，实现如下需求：

- 查询 Device 设备的 ARP 表项和接口信息。
- 配置 BGP。

11.2.2 组网图

图9 通过 NETCONF 查询和配置 Device 设备组网图



11.2.3 配置步骤

(1) 配置主机 Host 与 Device 设备的 IP 地址，确保主机 Host 与 Device 互通（略）。

(2) 在 Device 设备上开启 NETCONF over SSH server 功能。

生成 RSA 密钥对。

```
<Router> system-view
[Router] public-key local create rsa
The range of public key modulus is (512 ~ 4096).
If the key modulus is greater than 512, it will take a few minutes.
Press CTRL+C to abort.
Input the modulus length [default = 1024]:
Generating Keys...
.
Create the key pair successfully.
```

生成 DSA 密钥对。

```
[Router] public-key local create dsa
The range of public key modulus is (512 ~ 2048).
If the key modulus is greater than 512, it will take a few minutes.
Press CTRL+C to abort.
Input the modulus length [default = 1024]:
Generating Keys...
.
Create the key pair successfully.
```

生成 ECDSA 密钥对。

```
[Router] public-key local create ecdsa secp256r1
Generating Keys...
.
Create the key pair successfully.
```

开启 NETCONF over SSH 服务器功能。

```
<Device> system-view
[Device] netconf ssh server enable
```

创建用户 admin，并配置密码为 admin 和用户使用的服务类型为 SSH。

```
[Device] local-user admin class manage
[Device-luser-manage-admin] password simple admin
[Device-luser-manage-admin] service-type ssh
```

配置用户 admin 的授权角色为 network-admin 和 network-operator。

```
[Device-luser-manage-admin] authorization-attribute user-role network-admin
[Device-luser-manage-admin] authorization-attribute user-role network-operator
[Device-luser-manage-admin] quit
```

配置 admin 登录设备时，需要输入用户名和密码进行 AAA 认证。

```
[Device] line vty 0 63
[Device-line-vty0-63] authentication-mode scheme
[Device-line-vty0-63] user-role network-admin
[Device-line-vty0-63] user-role network-operator
```

(3) 在主机 Host 安装 ncclient。

ncclient 安装有如下两种方法：

- 使用 pip 工具安装。

在命令行执行 **`pip install ncclient`** 安装。

pip 安装的好处在于，在联网的前提下，就可以把 ncclient 依赖的软件包都下载和安装完整，省去了排错的问题。

- 使用源码安装。

在官网下载软件包，地址 https://pypi.org/project/ncclient/，解压文件，使用**`python setup.py install`**命令行安装。源码安装需要手工安装好多依赖包。

(4) 基于 ncclient 编写 Device 设备信息查询和功能配置的 Python 代码。

创建 query.py 文件，编写查询 Device 设备的 ARP 表项和接口信息代码。

```
#! /usr/bin/env python2.7
import sys, os, warnings
warnings.simplefilter("ignore", DeprecationWarning)
from ncclient import manager
import time
def my_unknown_host_cb(host, figerprint):
    return True

def demo(host, port, user, pwd):
    with manager.connect_ssh(host=host,
        port=port,
        username=user,
        password=pwd,
        unknown_host_cb=my_unknown_host_cb,

        device_params = {'name':'h3c'}) as m:
        for c in m.server_capabilities:
            print (c)

        get_xml = """
    <top xmlns="http://www.h3c.com/netconf/data:1.0">
  <ARP>
  <ArpTable>
    <ArpEntry>
      <IfIndex></IfIndex>
```

```
      <Ipv4Address></Ipv4Address>
      <MacAddress></MacAddress>
      <VLANID></VLANID>
      <PortIndex></PortIndex>
      <VrfIndex></VrfIndex>
      <ArpType></ArpType>
    </ArpEntry>
  </ArpTable>
  </ARP>
  <Ifmgr>
  <Interfaces>
    <Interface>
      <IfIndex></IfIndex>
      <Name></Name>
    </Interface>
  </Interfaces>
  </Ifmgr>
  </top>
    """
        print (m.get(('subtree', get_xml)))

if __name__ == '__main__':
    demo("17.1.1.124", 830, "admin", "admin")
    print ("closed")
    time.sleep(1)
```

保存该文件后，在主机 Host 执行 **python** *query.py* 命令执行该文件，即可看到 Device 设备的 ARP 表项和接口信息。

创建 deploy.py 文件，编写 Device 设备使能 BGP NSR 功能的 Python 代码。

```
#! /usr/bin/env python2.7
import sys, os, warnings
warnings.simplefilter("ignore", DeprecationWarning)
from ncclient import manager
import time
def my_unknown_host_cb(host, figerprint):
    return True
def demo(host, port, user, pwd):
    xml = """
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<top xmlns="http://www.h3c.com/netconf/config:1.0">
    <BGP>
    <Instances>
    <Instance>
      <Name></Name>
      <ASNumber>111</ASNumber>
      <NSR>1</NSR>
    </Instance>
    </Instances>
```

```
    </BGP>
    </top>
    </config>"""
    with manager.connect_ssh(host=host,
    port=port,
    username=user,
    password=pwd,
    unknown_host_cb=my_unknown_host_cb,
    device_params = {'name':'h3c'}) as m:
        for c in m.server_capabilities:
            print (c)
        print (xml)
        print (m.edit_config(target='running', config=xml))

if __name__ == '__main__':
    demo("17.1.1.124", 830, "admin", "admin")
```

保存该文件后，在主机 Host 上执行 **python** *deploy.py* 命令执行该文件，会收到 Device 设备返回的信息，如果“rpc-reply”元素内显示信息为“ok”，即表示执行成功。然后在 Device 设备上使用 **display current-configuration configuration bgp** 命令查看 BGP AS111 的配置信息。