S380 Multi-Service Gateway
V600R022C10

MD-CLI Configuration Reference

Issue	02
Date	2023-05-20

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://e.huawei.com

Contents

1 Basic Configuration

1.1 First Login to a Device Configuration

1.1.1 Overview of the First Login

Definition

First login to a device refers to logging in to a new device locally for the first time prior to configuring it.

Purpose

Before configuring services on a new device, you need to log in to the device locally through the management port.

After logging in locally, complete basic configurations such as the device name and management IP address to provide a basic environment for subsequent configurations.

1.1.2 Performing Basic Configurations After the First Login

1.1.2.1 Configuring User Login Prompt Information

Context

To provide some prompts or alarms to users, you can configure such information as titles on the device. When a user attempts to log in to the device, the configured titles will be displayed.

The default information is as follows: Authorized uses only. All activity may be monitored and reported.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the node view for setting the prompt to be displayed before login.

```
cli header
```

Step 3 Set the prompt to be displayed before login.

```
login-text text
```

Step 4 Commit the configuration.

```
commit
```

----End

1.1.2.2 Setting the Device Name

Context

To differentiate devices on the network, you can set a unique device name for each device.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the system setting node view.

```
system system-info
```

Step 3 Set the device name.

```
sys-name host-name
```

By default, the host name of a device is HUAWEI.

Step 4 Commit the configuration.

```
commit
```

----End

1.1.2.3 Configuring the Management Address and Gateway of the Device

Context

Each device on a network must have a globally unique management address, enabling O&M personnel to easily locate and log in to the device.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```


Step 2 Set an IP address and a mask for the interface.

```
ifm interfaces interface name interface-name
ipv4 addresses address ip ip-address
mask mask type { main | sub }
commit
quit 255
```

Step 3 Configure routes on the device.

```
network-instance instances instance name _public_
afs af type ipv4-unicast
routing routing-manage topologys topology name basic
quit 4
routing static-routing unicast-route2s unicast-route2 topology-name basic prefix mask mask-length mask-length
nexthop-interface-addresses nexthop-interface-address address ip-address interface-name interface-name
commit
```

----End

1.1.3 Verifying the Configuration

Verifying the Configuration

- Run the **display cli/header/login-text** command to check user login prompt information.

```
[(gl)device@HUAWEI]
MDCLI> display cli/header/login-text
"text"
```

- Run the **display system/system-info/sys-name** command to check the device name.

```
[(gl)device@HUAWEI]
MDCLI> display system/system-info/sys-name all
"HUAWEI"
```

- Run the **display ifm/interfaces/interface[name="GE0/0/0"]** command to check the management address of the device.

```
[(gl)device@HUAWEI]
MDCLI> display ifm/interfaces/interface[name="GE0/0/0"]
{
  "name": "GE0/0/0",
  "class": "main-interface",
  "type": "GigabitEthernet",
  "link-protocol": "ethernet",
  "huawei-ip:ipv4": {
    "addresses": {
      "address": [
        {
          "ip": "10.1.1.0",
          "mask": "255.255.255.0",
          "type": "main"
        }
      ]
    }
  }
}
```

- Run the **display network-instance/instances/instance[name="_public_"]** command to check the routes of the device.

```
[(gl)device@HUAWEI]
MDCLI> display network-instance/instances/instance[name="_public_"]
{
  "name": "_public_",
  "huawei-l3vpn:afs": {
    "af": [
```



```
    {
      "type": "ipv4-unicast",
      "huawei-routing:routing": {
        "routing-manage": {
          "topologys": {
            "topology": [
              {
                "name": "base"
              }
            ]
          }
        }
      }
    }
  ]
 }
}
```

1.1.4 Example for Performing Basic Configurations After the First Login

Networking Requirements

You need to use the console port for first login to the device and perform basic configurations.

Figure 1-1 Performing basic configurations after the first login through the console port

NOTE

In this example, interface1 represents GE0/0/1.

Procedure

Step 1 Log in to the device through the console port from PC1.

Step 2 Enter the edit-config view.

```
MDCLI> edit-config
```

Step 3 # Set the device name and IP address of the management interface.

```
MDCLI> ifm interfaces interface name GE0/0/1
[*(gl)admin123@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> ipv4 addresses address ip
10.8.1.1
[*(gl)admin123@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ipv4/addresses/
address[ip="10.8.1.1"]
MDCLI> mask 255.0.0.0 type main
[*(gl)admin123@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ipv4/addresses/
address[ip="10.8.1.1"]
MDCLI> commit
[(gl)admin123@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ipv4/addresses/
address[ip="10.8.1.1"]
MDCLI> quit 10
```

Configure a default route for the device with a gateway address of 10.8.1.1.


```
[(gl)admin123@HUAWEI]
MDCLI> network-instance instances instance name
_public_
[*(gl)admin123@HUAWEI]/network-instance/instances/instance[name="_public_"]
MDCLI> afs af type ipv4-
unicast
[*(gl)admin123@HUAWEI]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-
unicast"]
MDCLI> routing routing-manage topologys topology name
basic
[*(gl)admin123@HUAWEI]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-
unicast"]/routing/routing-manage/topologys/topology[name="basic"]
MDCLI> quit
4
[*(gl)admin123@HUAWEI]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-
unicast"]
MDCLI> routing static-routing unicast-route2s unicast-route2 topology-name basic prefix 10.1.1.0
mask-length 24
[*(gl)admin123@HUAWEI]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-
unicast"]/routing/static-routing/unicast-route2s/unicast-route2[topology-name="basic"][prefix="10.1.1.0"]
[mask-length="24"]
MDCLI> nexthop-interface-addresses nexthop-interface-address address 10.8.1.1 interface-name
GE0/0/1
[*(gl)admin123@HUAWEI]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-
unicast"]/routing/static-routing/unicast-route2s/unicast-route2[topology-name="basic"][prefix="10.1.1.0"]
[mask-length="24"]/nexthop-interface-addresses/nexthop-interface-address[interface-name="GE0/0/1"]
[address="10.8.1.1"]
MDCLI> commit
```

----End

Verifying the Configuration

Log in to the device using STelnet from PC2. The third-party software OpenSSH and Windows CLI are used in the following example.

- For details about how to install OpenSSH, see the OpenSSH installation guide.
- To use OpenSSH to connect to the device using STelnet, run the OpenSSH commands. For details about OpenSSH commands, see the OpenSSH help.
- The Windows CLI can identify OpenSSH commands only when OpenSSH is installed on the terminal.

Access the Windows CLI and run the OpenSSH command to log in to the device. The command format is **ssh** *user name***@***IP address.* (The following information is for reference only.)

```
C:\Users\******>ssh admin123@10.8.1.1
Authorized uses only. All activity may be monitored and reported.
admin123@10.8.1.1's password:

[admin123@HUAWEI]
MDCLI>
```


1.2 File System Management Configuration

Context

NOTE

When downloading files to a device or performing other file-related operations on a device, ensure that the power supply of the device is working properly. Otherwise, the downloaded files or the file system may be damaged, further damaging the storage medium or causing the device startup to fail.

1.2.1 Overview of the File System

File System

The file system manages files and directories in storage media. It allows users to create, delete, and modify files and directories, as well as view the contents of files. Files can be only uploaded and downloaded currently.

File Naming Rules

The value is a string of case-sensitive characters without spaces. The value length is as follows: FTP: 1 to 128 characters; SFTP: 1 to 256 characters; HTTP: 1 to 255 characters. The file name consists of letters, digits, and special characters including underscores (_), dots (.), hyphens (-), and @, and must start with a letter, digit, or underscore (_).

1.2.2 Configuration Precautions for File System Management

Licensing Requirements

File System Management is not under license control.

Hardware Requirements

Table 1-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

1.2.3 File System Management Modes Supported by the Device

During file management, a device may work as either a server or a client with the following functions:

- If the device works as a server, you can access the device using a terminal.
- If the device works as a client, you can use the device to access another device that works as a server.

Currently, the device can only function as a client.

There are advantages and disadvantages to each file system management mode, making them applicable to varying scenarios. Details are provided in **Table 1-2**. You can select an appropriate mode based on your specific requirements.

Table 1-2 File system management modes

File System Management Mode	Application Scenario	Advantage	Disadvantage
FTP	This mode applies to the file transfer scenario with low network security requirements, and is widely used in version upgrade.	• FTP is easy to configure and supports file transfer and file directory operations. • FTP supports file transfer between two file systems. • The authorization and authentication functions are provided.	Data is transmitted in plain text, resulting in potential security risks.
SFTP	This mode applies to scenarios demanding high network security, such as log download and configuration file backup scenarios.	• Encryption and integrity check are performed on data to ensure high security. • File transfer is supported.	The configuration is complex.
HTTPS	This mode applies to scenarios demanding high network security, such as log download and configuration file backup scenarios.	• Encryption and integrity check are performed on data to ensure high security. • File upload/download is efficient, requiring a single command that also sets up the client-server connection.	The configuration is complex.

FTP is easy to understand and configure, and is therefore not detailed here. The following only details the SFTP and HTTPS modes.

SFTP

As an extension of SSH, SFTP provides a secure channel through which remote users can log in to a device to manage and transfer files. In addition, the device can function as an SFTP client, from which users can securely log in to an SSH server for file transfer.

HTTPS

The HTTP function provides a unified interface for users and features that use the HTTP protocol to transmit data. HTTP however does not have any security mechanisms; it transmits data in clear text and does not authenticate either communication party. Therefore, data transmitted over such a protocol is vulnerable to tampering, sacrificing transmission security. To overcome this, HTTPS establishes an SSL encryption layer on HTTP, and is therefore more secure.

1.2.4 Managing Files Using FTP

1.2.4.1 Configuring a Device as an FTP Client

Context

You can configure a device as an FTP client, through which you can log in to a remote FTP server to transfer files between the server and client and manage files and directories on the server. Before configuring a device to access files on another device as an FTP client, you have completed the following tasks:

- Ensure that there are reachable routes between the device and FTP server.
- Obtain the IP address, user name, and password of the FTP server.
- Obtain the port number configured for the FTP server if the server does not use the standard port number.

For details about configuration parameters, see **huawei-ftpc.yang**.

NOTICE

SFTP V3 is more secure than FTP, and is therefore recommended.

Procedure

Step 1 Enable the FTP client.

Operation	Command	Description
Enter the editing view.	**edit-config**	-

Operation	Command	Description
Enable the FTP client function.	**touch ftpc/client/ enabled true**	Files cannot be transferred before the FTP client function is enabled. After the FTP client function is enabled, you can run the **ftpc-transfer-file** command.
Commit the configuration.	**commit**	-
Return to the system view.	**return**	-

Step 2 Connect to the FTP server to perform file operations.

Operation	Command	Description
Enter the ftpc-transfer-file view.	**ftpc-transfer-file**	-
Enable the file download or upload function.	**command-type** [**get** | **put**]	• **get**: downloads files. • **put**: uploads files.
Configure the user name.	**user-name** *username*	-
Configure the password	**password** Enter password: Confim password:	-
Configure the IPv4 address or host name of the FTP server.	**server-ipv4-address** *host-address*	-
(Optional) Configure the listening port for the FTP server.	**server-port** *port*	By default, the FTP server uses port 21.
Specify the name of the file to be downloaded or uploaded.	**local-file-name** *localfilename*	The file name or directory/file name format is supported for file upload. The file name or directory (specified directory)/file name format is supported for file download. If this parameter is not specified, the file name specified by the *remotefilename* variable is used.

Operation	Command	Description
Specify the name of the file stored on the FTP server.	**remote-file-name** *remotefilename*	You can set *remotefilename* to a directory name or file name. If no file name is entered, the file name specified by *localfilename* is used.
Commit the configuration.	**emit**	-

NOTE

The local file of a newly started download task cannot be the same as that of a running download task.

After the **emit** command is executed, the value of **transfer-id** is returned. The successful output of the **emit** command only indicates that the device starts to connect to the remote FTP server, but does not indicate that the file transfer starts or is complete. You need to run the **display ftpc/transfer-tasks/** command to query the record corresponding to the value of **transfer-id**.

Step 3 (Optional) Cancel the ongoing file operation.

Operation	Command	Description
Enter the ftpc-cancel-transfer-task view.	**ftpc-cancel-transfer-task**	-
Cancel the ongoing file operation.	**transfer-id** *transferid*	Only file operations that are being performed can be canceled. Operations that have been completed or failed cannot be canceled.
Commit the configuration.	**emit**	-

----End

Verifying the Configuration

- Run the **display ftpc/transfer-tasks/** command to check the progress and status of all file operations.
- Run the **display ftpc/transfer-tasks/transfer-task[transfer-id=**"*transfer-id*"**]** command to check the progress and status of the specified file operation.

1.2.4.2 Example for Configuring a Device as an FTP Client

Networking Requirements

In Figure 1-2, the remote device with IP address 10.1.1.1/24 functions as the FTP server. The device with IP address 10.2.1.1/24 functions as the FTP client and has reachable routes to the FTP server.

The device needs to be upgraded. To be specific, you need to download the system software from the FTP server to the device.

Figure 1-2 Network diagram for accessing files on another device using FTP

Procedure

Step 1 Run the FTP software on the FTP server and configure an FTP user. For details, see the help document of the third-party software.

Step 2 Enable the FTP client function.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> touch ftpc/client/enabled true
[*(gl)user@HUAWEI]
MDCLI> commit
[(gl)user@HUAWEI]
MDCLI> return
```

Step 3 Download files through FTP on the device.

```
[user@HUAWEI]
MDCLI> ftpc-transfer-file
[(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> user-name user
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> password
Enter password: password
Confirm password: password
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> command-type get
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> local-file-name test.cc
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> remote-file-name test.cc
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> server-ipv4-address 10.1.1.1
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> emit
{
  "huawei-ftpc:transfer-id": 1
}
```

----End

Verifying the Configuration

Run the **display ftpc/transfer-tasks/** command to check the file operation status.

```
[user@HUAWEI]
MDCLI> display ftpc/transfer-tasks/
{
  "transfer-task": [
    {
      "transfer-id": 1,
      "command-type": "get",
      "server-address": "10.1.1.1",
      "local-file-name": "test.cc",
      "remote-file-name": "test.cc",
      "status": "succeeded",
      "percentage": 100
    }
  ]
}
[user@HUAWEI]
MDCLI>
```

1.2.5 Managing Files Using SFTP

1.2.5.1 Configuring a Device as an SFTP Client

Context

After a device is configured as an SFTP client, client authentication and bidirectional data encryption are used to ensure secure file transfer and file and directory management. Before configuring a device to access files on another device as an SFTP client, you have completed the following tasks:

- Ensure that there are reachable routes between the device and SSH server.
- Obtain the IP address of the SSH server and SSH user information, and ensure that the SSH user has been assigned the highest privilege level.
- Obtain the port number configured for the SSH server if the server does not use the standard port number.

For details about configuration parameters, see **huawei-sshc.yang**.

Procedure

Step 1 Connect to the SFTP server to perform file operations.

Operation	Command	Description
Enter the ssh-transfer-file view.	**ssh-transfer-file**	-
Enable the file download or upload function.	**command-type** [**get** | **put**]	• **get**: downloads files. • **put**: uploads files.
Configure the user name.	**user-name** *username*	-

Operation	Command	Description
Configure the password	**password** Enter password: Confim password:	-
Configure the IPv4 address or host name of the SFTP server.	**host-addr-ipv4** *host-address*	-
(Optional) Configure the listening port for the SFTP server.	**server-port** *port*	By default, the SFTP server uses port 22.
Specify the name of the file to be downloaded or uploaded.	**local-file-name** *localfilename*	Only the file name can be entered during download. The directory/file name can be entered during upload. If no file name is entered, the file name specified by the *remotefilename* variable is used.
Specify the name of the file stored on the SFTP server.	**remote-file-name** *remotefilename*	You can set *remotefilename* to a directory name or file name. If no file name is entered, the file name specified by *localfilename* is used.
Commit the configuration.	**emit**	-

NOTE

The local file of a newly started download task cannot be the same as that of a running download task.

After the **emit** command is executed, the value of **transfer-id** is returned. The successful output of the **emit** command only indicates that the device starts to connect to the remote SFTP server, but does not indicate that the file transfer starts or is complete. You need to run the **display sshc/transfer-tasks/** command to query the record corresponding to the value of **transfer-id**.

Step 2 Cancel the ongoing file operation.

Operation	Command	Description
Enter the ssh-cancel-transfer-task view.	**ssh-cancel-transfer-task**	-

Operation	Command	Description
Cancel the ongoing file operation.	**transfer-id** *transferid*	Only file operations that are being performed can be canceled. Operations that have been completed or failed cannot be canceled.
Commit the configuration.	**emit**	-

----End

Verifying the Configuration

- Run the **display sshc/transfer-tasks/** command to check the progress and status of all file operations.
- Run the **display sshc/transfer-tasks/transfer-task[transfer-id="***transfer-id***"]** command to check the progress and status of the specified file operation.

1.2.5.2 Example for Configuring a Device as an SFTP Client

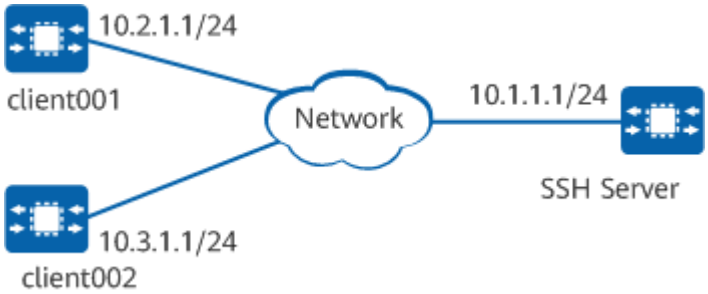
Networking Requirements

The SSH protocol uses encryption to secure the connection between a client and a server. All user authentication, commands, output, and file transfers are encrypted to protect against attacks in the network. A client can securely connect to the SSH server and transfer files using SFTP.

In Figure 1-3, routes between the SSH server and clients client001 and client002 are reachable. In this example, a Huawei device functions as the SSH server.

It is required that the two clients should connect to the SSH server in password authentication mode to ensure secure access to files on the server.

Figure 1-3 Network diagram for accessing files on another device using SFTP



Procedure

Step 1 Run the SFTP software on the SFTP server and configure an SFTP user. For details, see the help document of the third-party software.

Step 2 Download files through SFTP on the device.

```
[user@HUAWEI]
MDCLI> ssh-transfer-file
[(x)user@HUAWEI]/ssh-transfer-file
MDCLI> user-name user
[*(x)user@HUAWEI]/ssh-transfer-file
MDCLI> password
Enter password:
Confirm password:
[*(x)user@HUAWEI]/ssh-transfer-file
MDCLI> command-type get
[*(x)user@HUAWEI]/ssh-transfer-file
MDCLI> local-file-name test.cc
[*(x)user@HUAWEI]/ssh-transfer-file
MDCLI> remote-file-name test.cc
[*(x)user@HUAWEI]/ssh-transfer-file
MDCLI> host-addr-ipv4 10.1.1.1
[*(x)user@HUAWEI]/ssh-transfer-file
MDCLI> emit
{
  "huawei-sshc:transfer-id": 2
}
```

----End

Verifying the Configuration

Run the **display sshc/transfer-tasks/** command to check the file operation status.

```
MDCLI> display sshc/transfer-tasks/
{
  "transfer-task": [
    {
      "transfer-id": 2,
      "command-type": "get",
      "host-addr": "10.1.1.1",
      "server-port": 22,
      "local-file-name": "test.cc",
      "remote-file-name": "test.cc",
      "status": "succeeded",
      "percentage": 100
    }
  ]
}
```

1.2.6 Managing Files Using HTTPS

1.2.6.1 Configuring a Device as an HTTPS Client

Context

You can configure a device as an HTTPS client, through which you can log in to a TFTP server to upload and download files between the client and server. Before using HTTPS to perform file operations, you have completed the following tasks:

- Ensure that there are reachable routes between the device and HTTPS server.
- Obtain the host name or IP address of the HTTPS server and certificate of the HTTPS server.
- Obtain the port number configured for the HTTPS server if the server does not use the standard port number.

For details about configuration parameters, see **huawei-http.yang**.

Procedure

Step 1 Connect to the HTTPS server to perform file operations.

Operation	Command	Description
Enter the httpc-transfer-file view.	**httpc-transfer-file**	-
Enable the file download or upload function.	**operation-type** [**download** | **upload**]	• **download**: downloads files. • **upload**: uploads files.
Configure an SSL policy.	**ssl-policy-name** *username*	-
Configure a URL for the HTTPS server.	**file-url** *url*	The URL includes an IPv4 address (or a host name), a transmission port number, and a destination file name, for example, https://device-naas.huawei.com:18020/shell/3.test.
Specify the name of the file to be downloaded or uploaded.	**file-full-path** *localfilename*	Only the file name can be entered during download, and the directory/file name can be entered during upload.
Commit the configuration.	**emit**	-

NOTE

The local file of a newly started download task cannot be the same as that of a running download task.

After the **emit** command is executed, the value of **transfer-id** is returned. The successful output of the **emit** command only indicates that the device starts to connect to the remote HTTPS server, but does not indicate that the file transfer starts or is complete. You need to run the **display http/transfer-tasks/** command to query the record corresponding to the value of **transfer-id**.

Step 2 (Optional) Cancel the ongoing file operation.

Operation	Command	Description
Enter the httpc-cancel-transfer-task view.	**httpc-cancel-transfer-task**	-

Operation	Command	Description
Cancel the ongoing file operation.	**transfer-id** *transferid*	Only file operations that are being performed can be canceled. Operations that have been completed or failed cannot be canceled.
Commit the configuration.	**emit**	-

----End

Verifying the Configuration

- Run the **display http/transfer-tasks/** command to check the progress and status of all file operations.
- Run the **display http/transfer-tasks/transfer-task[transfer-id="***transfer-id***"]** command to check the progress and status of the specified file operation.

1.2.6.2 Example for Configuring a Device as an HTTPS Client

Networking Requirements

The HTTPS protocol uses encryption to secure the connection between a client and a server. All user authentication, commands, output, and file transfers are encrypted to protect against attacks in the network. A client can securely connect to the HTTPS server and transfer files using HTTPS.

In Figure 1-4, the remote device with IP address 10.1.1.1/24 functions as the HTTPS server. The device with IP address 10.2.1.1/24 functions as the HTTPS client and has reachable routes to the HTTPS server.

The device needs to be upgraded. To be specific, you need to download the system software from the HTTPS server to the device.

Figure 1-4 Network diagram for accessing files on another device using HTTPS

Procedure

Step 1 Run the HTTPS software on the HTTPS server and set the HTTPS working directory. For details, see the help document of the third-party software.

Step 2 Configure file download and upload through HTTPS on the device.

```
[user@HUAWEI]
MDCLI> httpc-transfer-file
[(x)user@HUAWEI]/httpc-transfer-file
MDCLI> ssl-policy-name default
```



```
[*(x)user@HUAWEI]/httpc-transfer-file
MDCLI> operation-type download
[*(x)user@HUAWEI]/httpc-transfer-file
MDCLI> file-full-path test.cc
[*(x)user@HUAWEI]/httpc-transfer-file
MDCLI> file-url https://device-naas.huawei.com/test.cc
[*(x)user@HUAWEI]/httpc-transfer-file
MDCLI> emit
{
  "huawei-http:transfer-id": 3
}
```

----End

Verifying the Configuration

Run the **display http/transfer-tasks/** command to check the file operation status.

```
[user@HUAWEI]
MDCLI> display http/transfer-tasks/
{
  "transfer-task": [
    {
      "transfer-id": 3,
      "operation-type": "download",
      "file-url": "https://device-naas.huawei.com/test.cc",
      "file-full-path": "test.cc",
      "transfer-status": "succeeded",
      "percentage": 100
    }
  ]
}
```

1.2.7 Troubleshooting File System Management Errors

1.2.7.1 Failed to Transfer Files Between the FTP Server and Client

Possible Causes

- The FTP source or destination file name contains characters not supported by the device, such as spaces.
- The FTP server is unreachable.
- The FTP server authentication failed.

Procedure

Step 1 The FTP source or destination file name contains characters not supported by the device, such as spaces.

The file name consists of letters, digits, and special characters including underscores (_), dots (.), hyphens (-), and @, and must start with a letter, digit, or underscore (_).

If the directory name contains any of these characters, change the directory name.

Step 2 The FTP server is unreachable.

Check the network connection and network configuration to ensure that the server is reachable.

Step 3 The FTP server authentication failed.

Enter the correct user name and password, and check whether the case lock status of the keyboard is correct.

----End

1.3 Configuration File Management Configuration

1.3.1 Overview of Configuration File Management

Definition

A configuration file is a collection of configurations that can be recorded on a device.

Purpose

With the configuration file management feature, you can perform the following operations on configuration files: view the current and next startup configuration files, save configurations to the configuration file for next startup, replace a configuration file, back up a configuration file, export a configuration file, import a configuration file as the configuration file for next startup or the default startup configuration file, clear the configuration file for next startup or the default startup configuration file, and automatically save configurations to the configuration file for next startup. This feature ensures that user configurations run properly on the device, prevents configuration loss, and facilitates configuration migration.

1.3.2 Configuration Precautions for Configuration File Management

Licensing Requirements

Configuration File Management is not under license control.

Hardware Requirements

Table 1-3 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

1.3.3 Managing Configuration Files

1.3.3.1 Understanding Configuration Files

Configuration File Format

Configuration files are compressed using the gzip algorithm and must meet the following requirements:

- The name of a configuration file consists of letters, digits, special characters including underscores (_), hyphens (-), and @, and must start with a letter, digit, or underscore (_). The file name extension is not limited to .gz.
- The length of the configuration file name ranges from 8 bytes to 64 bytes.
- Configuration files contain two text files: **startup.cfg** and **startup.meta**.
- The **startup.cfg** file can contain only configuration-related content and is in JSON format.
- The **startup.meta** contains the configuration metadata, which is used to record the master key information and configuration integrity information.

Configuration File Category

The following table lists the differences between the configurations loaded when the device is running. These configurations fall into the following types: factory configuration, initial configuration, current configuration, offline configuration, and next startup configuration.

Type	Description	Command
Factory configuration	Factory configurations are basic configurations provided for a new device. This type of configurations enables a device to start and operate correctly when there is no configuration file, or when the configuration file is lost or damaged.	-
Initial configuration	When a device is powered on, it reads the configuration file from the default directory to initialize the system. Therefore, the configurations in the file are called initial configurations. If no configuration file is stored in the default directory, the device uses the factory configurations for initialization.	-

Type	Description	Command
Current configuration	The configurations that are in effect during device running are current configurations.	Run the **display current-configuration** command to check the current configurations. Run the **display cfg/startup-infos** command to check the configuration file used for the current startup (specified by current-cfg-file). If no current startup configuration file is displayed, you can run the **display current-configuration** command. If the command output is displayed, factory configurations are loaded for current startup.
Next startup configuration	After the system starts, you can specify a configuration file as the initial configurations for the next startup, known as the next startup configurations	Run the **display cfg/startup-infos** command to check the configuration file for next startup (specified by next-cfg-file). If only the factory configuration file is configured, the configuration file for next startup cannot be queried.

To use modified configurations as the next startup configurations, run the **save** command to save them to the default storage medium.

1.3.3.2 Viewing a Configuration File

Procedure

For details about query parameters, see **huawei-cfg.yang** and **ietf-netconf.yang**.

Table 1-4 Viewing a configuration file

Operation	Command	Description
Check the configuration files for the current and next startup.	**display cfg/startup-infos**	It cannot be used to distinguish the startup with factory configurations from the startup with default configurations. To distinguish them, you also need to run the **display current-configuration** command.
Check configurations in the configuration file for the next startup.	1. Enter the configuration query RPC view from the system view. **get-config** 2. Access the source node. **source** 3. Set **source** to the startup target. **startup** [**null**] 4. Submit the RPC for query. **emit**	This is an RPC execution process on the CLI. You need to enter information one line by one.
Check device configuration change information.	**display cfg/config-change**	-

1.3.3.3 Saving a Configuration File

Context

In the current configuration file, the configurations that are different from those in the configuration file for the next startup will be lost after the restart. If such configurations are required after the restart, save them to the configuration file before restarting the device. Two methods are available for saving configurations to a configuration file:

- Enable the system to automatically save configurations.
- Manually save the configurations.

Device configurations are stored in the configuration file of the storage medium. During startup, the system reads the configuration file to restore configurations of the device, and then saves the restored configurations to memory.

You can run the **get-config** command (for details, see **1.3.3.2 Viewing a Configuration File**) to check configurations in the configuration file, and the **display current-configuration** command to check those in memory.

When the device has not run properly during system startup, the configurations in the configuration file are not completely restored in memory. If you run the **save** command at this time, incomplete configurations in memory will override those in the configuration file. As a result, some configurations may be lost.

On a device running properly, the configurations in the configuration file should be the same as those in memory. If you add, modify, or delete configurations, the latest configurations are saved in memory, and will be different from those in the configuration file. In this case, you can run the **save** command to save the current configurations in memory to the configuration file.

For details about configuration parameters, see **huawei-cfg.yang**.

Procedure

- Enable the system to automatically save configurations.
  a. Enter the editing view from the system view.
     edit-config
  b. Enable the function of periodically saving configurations.
     cfg autosave

     If you do not configure the subnodes under autosave, the default values of the subnodes are used.

     During automatic saving, if the startup data set is locked, configurations are being saved, imported, or exported, or the disk space is insufficient, the system cancels the automatic saving.
  c. (Optional) Configure the system to save the configurations at a scheduled time.
     delay-time *delay-time*
     internal-time *internal-time*

     There are two methods of setting the scheduled automatic saving time.

     i. **delay-time**: specifies the delay time, in minutes. The value ranges from 1 to 60. If the configuration changes again during the delay, the delay timer restarts. After the delay timer expires, configurations are automatically saved.
     ii. **internal-time**: specifies the interval, in minutes. The value ranges from 30 to 43200. The interval does not change with the configuration change. After the timer expires, configurations are automatically saved.
     iii. The automatic saving function is triggered when either of the timer expires first. After configurations are automatically saved, the automatic saving function enters the sleep state. The two timers start again when a new configuration change occurs.
     iv. If configurations fail to be automatically saved for the first time, the system retries a maximum of five times at an interval of up to five minutes (the smaller value between **delay-time** and 5 minutes is used as the interval). If the retry succeeds or the retry fails five times, the automatic saving function enters the sleep state.

d. Commit the configuration.
commit

- Manually save the configurations.
  - Save current configurations to the configuration file for next startup.
save

    After the **save** command is executed, the current system configurations overwrite those in the configuration file for next startup. If the configuration file for next startup has not been specified, the **startup.zip** file is used as the configuration file for next startup. The system startup configuration file must be stored in the root directory of the storage medium. You can run the **display cfg/startup-infos/startup-info** command to view the name of the current startup configuration file.
  - Save current configurations to the target file.
huawei-cfg:save filename *filename*

    You can run the **huawei-cfg:save** command to save configurations to a specified file that meets the naming requirements. If the specified file is an existing configuration file but not the configuration file for next startup or the default configuration file, the configuration file is overwritten. If the specified file does not exist, configurations are directly saved to the new file.
  - Enter the password to encrypt and save the configuration file.
huawei-cfg:save filename *filename* **shareable-mode password** *password*

    The device generates a key in the configuration file based on the password entered by the user. When the configuration file is used next time, the password must be entered for authentication.
  - Save the configuration file in white-box encryption mode.
huawei-cfg:save filename *filename* **shareable-mode** default

    The device generates a key in the configuration file using the white-box encryption algorithm. When the configuration file is used next time, white-box verification is required.

    NOTE

    After the weak password dictionary maintenance function is enabled, the passwords (which can be queried using the **display system/weak-passwords** command) defined in the weak password dictionary cannot be specified in this command.

----End

Example

1. Set the interval at which the configuration file is saved to 60 minutes.
```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> cfg autosave

[*(gl)user@HUAWEI]/cfg/autosave
MDCLI> interval-time 60

[*(gl)user@HUAWEI]/cfg/autosave
MDCLI> commit
```


2. After 60 minutes, you can view the saved configuration file for next startup in the root directory of the device. You can run the **display cfg/startup-infos** command to query the configuration file name.

```
[user@HUAWEI]
MDCLI> display cfg/startup-infos
{
  "startup-info": [
    {
      "position": "0",
      "current-cfg-file": "",
      "next-cfg-file": "startup.zip"
    }
  ]
}
```

1.3.3.4 Specifying the Configuration File for Next Startup

Context

When the system restarts, it uses the specified configuration file to restore configurations.

Before specifying the file for the next startup, you can run the **display cfg/startup-infos** command to view the current specified file.

If no configuration file is specified, the default configuration file (which can be set using the **set-default-config** command) will be used for the next startup. If no configuration file is stored in the storage medium, factory configurations are used for startup.

For details about configuration parameters, see **huawei-cfg.yang**.

NOTE

Manually constructing a configuration file is not recommended because manual construction is prone to file format errors. The format error may cause a failure to restore configurations or an error during configuration restoration.

The configuration file for the next startup must exist and be saved in the root directory of the storage medium.

After the **save** command is executed, the current system configurations overwrite those in the configuration file for next startup. If the configuration file for next startup has not been specified, the **startup.zip** file is used as the configuration file for next startup.

Procedure

- Configure the configuration file for the next startup.

  set-startup filename *filename*

  If the specified configuration file does not contain key information, the configuration file can be successfully set only after both the integrity check and content check succeed.

- Configure the configuration file containing key information of password encryption for the next startup.

  set-startup filename *filename* **sharable-mode password** *password*

  If the specified configuration file contains key information, the configuration file can be successfully set only after a correct password is entered and both the integrity check and content check succeed.

- Configure the configuration file containing key information of white-box encryption for the next startup.

  set-startup filename *filename* **sharable-mode** default

  If the specified configuration file contains key information, the configuration file can be successfully set only after the white box verification, integrity check, and content check succeed.
- Configure the default configuration file.

  set-default-config filename *filename*

  If no configuration file is specified for the next startup, the default configuration file will be used during system startup.

----End

Verifying the Configuration

Run the **display cfg/startup-infos** command to check the configuration file for the next startup and current startup configuration file.

1.3.3.5 Replacing the Configuration File

Context

When the configuration file of a device needs to be updated, you can load the configuration file from the local server to the local device to replace the running configuration file.

NOTE

The specified configuration file to be loaded must exist and meet the following conditions:

- The file name consists of letters, digits, and special characters including underscores (_), hyphens (-), and @, and must start with a letter, digit, or underscore (_).
- The file cannot be a package.
- The file content is in JSON format. You are advised to run the **export** command to export the file.
- The maximum file size is 10 MB.
- The configuration file must exist and be saved in the root directory of the storage medium.

Procedure

Step 1 Run the **import** command.

import *filename* { **candidate** | **running** | **startup** }

You can import the configuration file that meets the requirements in the root directory of the primary partition to the selected dataset.

----End

Verifying the Configuration

Perform the following operations to verify the configuration:

- Run the **display current-configuration candidate** command to check the uncommitted configurations and check whether the new configurations meet the expectation.
- Run the **display current-configuration** command to check the current configurations and check whether the new configurations meet the expectation.

1.3.3.6 Backing Up the Configuration File to an FTP Server

Prerequisites

Before backing up a configuration file, you have completed the following tasks:

- If the device functions as an FTP client, connect it to an FTP server. For details, see **1.2.4.1 Configuring a Device as an FTP Client** in the *File System Management Configuration*.

Context

If the device is damaged unexpectedly, configuration files cannot be restored. In case of that, you can back up configuration files through FTP.

Back up the configuration file to the FTP server when the device functions as an FTP client.

For details about configuration parameters, see **huawei-ftpc.yang**.

NOTE

Backing up the configuration file through FTP is a simple process, which however may pose security risks.

Procedure

- Back up the configuration file to the FTP server when the device functions as an FTP client.
  a. Enter the editing view.
     edit-config
  b. Enter the FTP client view.
     ftpc client
  c. Enable the FTP client function.
     enabled *true*
     By default, the FTP client function is disabled.
  d. Commit the configuration.
     commit
  e. Save the configuration.
     save
  f. Return to the system view.
     return
  g. Enter the ftpc-transfer-file view.
     ftpc-transfer-file

h. Configure the user name.

```
user-name user-name
```

i. Configure the password.

```
password
Enter password:
Confirm password:
```

j. Configure the operation type.

```
command-type put
```

k. Configure the local file name.

```
local-file-name file-name
```

l. Configure the remote file name.

```
remote-file-name file-name
```

m. Configure the IP address of the server.

```
server-ipv4-address ip-address
```

n. Submit the commands.

```
emit
```

----End

Verifying the Configuration

The configuration file is saved to the working directory of the FTP user, and the size of the configuration file on the device is the same as that on the FTP server or client.

1.3.3.7 Copying the Configuration File from an FTP Server to the Device

Prerequisites

Before copying a configuration file from an FTP server to the device, you have completed the following tasks:

- If the device functions as an FTP client, connect it to an FTP server. For details, see **1.2.4.1 Configuring a Device as an FTP Client** in the *File System Management Configuration*.

Context

If functions do not operate properly due to incorrect configurations, you can copy the backup configuration file to the device through FTP to restore the functions.

Copy the configuration file from the FTP server when the device functions as an FTP client.

For details about configuration parameters, see **huawei-ftpc.yang**.

NOTE

Restoring the configuration file through FTP is a simple process, which however may pose security risks.

Procedure

- Copy the configuration file from the FTP server when the device functions as an FTP client.
  a. Enter the editing view.
     edit-config
  b. Enter the FTP client view.
     ftpc client
  c. Enable the FTP client function.
     enabled *true*

     By default, the FTP client function is disabled.
  d. Commit the configuration.
     commit
  e. Return to the system view.
     return
  f. Enter the ftpc-transfer-file view.
     ftpc-transfer-file
  g. Configure the user name.
     user-name *user-name*
  h. Configure the password.
     password
     Enter password:
     Confirm password:
  i. Configure the operation type.
     command-type get
  j. Configure the local file name.
     local-file-name *file-name*
  k. Configure the remote file name.
     remote-file-name *file-name*
  l. Configure the IP address of the server.
     server-ipv4-address *ip-address*
  m. Submit the commands.
     emit
  n. Set the configuration file for the next startup.
     set-startup filename *file-name*
  o. Restart the device.
     reboot

----End

Verifying the Configuration

Run the **display cfg/startup-infos** command to check the current startup configuration file.

1.3.3.8 Clearing the Configuration File

Context

Clearing the configuration file is necessary in the following scenarios:

- The software and configuration file do not match after the device software is upgraded.
- The configuration file is damaged, or an incorrect configuration file is loaded.

NOTICE

The **clear-startup** command will clear the configuration file used for the next startup. You are advised to run this command under the guidance of technical support personnel.

The **clear-default-config** command will clear the default configuration file. You are advised to run this command under the guidance of technical support personnel.

For details about configuration parameters, see **huawei-cfg.yang**.

Procedure

- Delete configurations for the next startup.

  Cancel the configuration file specified for the next startup to restore the default configurations.

  ```
  clear-startup
  reboot save-flag false      //Restart the device to validate the configuration.
  ```

  NOTE

  After the configuration file specified for the next startup is canceled, the device will start with default configurations next time, unless the **set-startup** command is executed to specify a new configuration file, or the **save** command is executed to save configurations to the configuration file for the next startup. If the default configuration file does not exist, the device starts with factory configurations.

- Clear the default configuration file.

  ```
  clear-default-config
  ```

  NOTE

  Exercise caution when running this command. If this command is required, run it with assistance from technical support personnel.

  After the default configuration file is cleared, if the configuration file for next startup does not exist or fails to be loaded, the device starts with factory configurations.

----End

1.3.3.9 Example for Specifying the Configuration File to Be Loaded for Next Startup

Networking Requirements

The current system software cannot meet user needs. The device must load new software version with more features. Then the device software needs to be upgraded remotely.

Configuration Roadmap

For details about configuration parameters, see **huawei-cfg.yang** and **huawei-ftpc.yang**.

The configuration roadmap is as follows:

1. Upload the new system software to the root directory of the device.
2. Specify the system software to be loaded for next startup.
3. Configure an AAA user.
4. Save current system configurations to the configuration file for next startup.
5. Restart the device to complete upgrade.

Procedure

Step 1 Upload the new system software to the root directory of the device.

1. Run the **display software/startup-packages/startup-package** command to check the package for the next startup, and run the **display cfg/startup-infos** command to check the configuration file for the next startup.

```
[user@HUAWEI]
MDCLI> display software/startup-packages/startup-package
[
  {
    "slot-id": "0",
    "current-package": "S380-V600R022C10-0.cc",
    "next-package": "S380-V600R022C10-0.cc"
  }
]
[user@HUAWEI]
MDCLI> display cfg/startup-infos
{
  "startup-info": [
    {
      "position": "0",
      "current-cfg-file": "",
      "next-cfg-file": ""
    }
  ]
}
```

2. Enable the FTP client function on the device.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> ftpc client
[(gl)user@HUAWEI]/ftpc/client
MDCLI> enabled true
[*(gl)user@HUAWEI]/ftpc/client
MDCLI> commit
[(gl)user@HUAWEI]/ftpc/client
MDCLI> return
```


3. Download system software packages through FTP on the device. (For details, see **1.2.4 Managing Files Using FTP** in the *File System Management Configuration*.)

```
[user@HUAWEI]
MDCLI> ftpc-transfer-file
[(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> user-name admin
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> password
Enter password: password
Confirm password: password
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> command-type get
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> local-file-name S380-V600R022C10-1.cc
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> server-ipv4-address 10.64.24.151
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> remote-file-name S380-V600R022C10-1.cc
[*(x)user@HUAWEI]/ftpc-transfer-file
MDCLI> emit
```

Step 2 Specify the system software to be loaded for next startup.

```
[user@HUAWEI]
MDCLI> startup-by-mode name S380-V600R022C10-1.cc
```

Step 3 Configure an AAA user.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> aaa lam users user name admin
[*(gl)user@HUAWEI]/aaa/lam/users/user[name="admin"]
MDCLI> password
Enter password: password
Confirm password: password
[*(gl)user@HUAWEI]/aaa/lam/users/user[name="admin"]
MDCLI> service-terminal true
[*(gl)user@HUAWEI]/aaa/lam/users/user[name="admin"]
MDCLI> commit
[(gl)user@HUAWEI]/aaa/lam/users/user[name="admin"]
MDCLI> return
```

Step 4 Save the current configuration.

```
[user@HUAWEI]
MDCLI> save
Warning: The current configuration will be saved to the startup configuration database. Continue? [Y(yes)/
N(no)]:y
Info: Save the configuration successfully.
```

Step 5 Verify the configuration.

Run the following command to view the system software and configuration file for next startup.

```
[user@HUAWEI]
MDCLI> display software/startup-packages/startup-package
[
  {
    "slot-id": "0",
    "current-package": "S380-V600R022C10-0.cc",
    "next-package": "S380-V600R022C10-1.cc"
  }
]
[user@HUAWEI]
MDCLI> display cfg/startup-infos
{
  "startup-info": [
    {
```



```
    "position": "0",
    "current-cfg-file": "",
    "next-cfg-file": "startup.zip"
   }
  ]
}
```

Step 6 Restart the device.

Because the configuration file has been saved, run the following command to restart the device quickly.

```
[user@HUAWEI]
MDCLI> reboot save-flag false
```

----End

Verifying the Configuration

Wait for several minutes until the device restart is complete. Check the system software and configuration file to be loaded for next startup. If the current system software version is the target version, the upgrade is complete.

```
[user@HUAWEI]
MDCLI> display software/startup-packages/startup-package
[
  {
    "slot-id": "0",
    "current-package": "S380-V600R022C10-1.cc",
    "next-package": "S380-V600R022C10-1.cc"
  }
]
[user@HUAWEI]
MDCLI> display cfg/startup-infos
{
  "startup-info": [
    {
      "position": "0",
      "current-cfg-file": "startup.zip",
      "next-cfg-file": "startup.zip"
    }
  ]
}
```


2 Interface Management Configuration

2.1 Interface Basic Configuration

2.1.1 Overview of Interfaces

2.1.1.1 Interface Types

Interfaces of a device are used to exchange data and interact with other network devices. Interfaces are classified as physical or logical interfaces.

Physical Interfaces

Physical interfaces are real interfaces that are supported by components. They are classified as management or service interfaces.

Management Interfaces

Management interfaces are used to log in to devices for configuration and management purposes. They are not used for service transmission.

Table 2-1 describes the management interfaces that the device supports.

Table 2-1 Management interfaces

Interface	Description	Application
Console interface	A data connection equipment (DCE) interface that complies with the EIA/TIA-232 standard.	The console interface is connected to the COM serial interface of a configuration terminal to set up an on-site configuration environment.

Interface	Description	Application
MGMT interface	Complies with the 10/100/1000BASE-TX standard.	The MEth interface can be connected to a network interface of a configuration terminal or network management workstation to set up an on-site or remote configuration environment.
Micro USB interface	Complies with the USB 2.0 standard.	This interface is used to connect to the console for onsite configuration of the system.

NOTE

- The console interface and micro USB interface share the same internal serial interface. You can use the console interface or micro USB interface as required. When the micro USB interface is activated, the console interface cannot be used.
- If both interfaces are connected, the micro USB interface is preferred.

Service interfaces

Service interfaces are used for service transmission. Service interfaces are also referred to as ports. This document uses the term interface.

Table 2-2 describes the service interfaces that the device supports.

Table 2-2 Service interfaces

Interface Type	Description
LAN-side interface	A LAN-side GE interface works at the data link layer, processes Layer 2 protocol packets, implements fast Layer 2 forwarding, and provides a maximum transmission rate of 1000 Mbit/s.
WAN-side interface	A WAN-side GE interface works at the network layer, supports being configured with an IP address, processes Layer 3 protocol packets, provides the routing function, and provides a maximum transmission rate of 1000 Mbit/s.

Logical Interfaces

Logical interfaces are manually configured and do not physically exist. They are responsible for transmitting service data.

Table 2-3 describes the logical interfaces that the device supports.

Table 2-3 Logical interfaces

Interface Type	Description	Application Scenario
Vlanif interface	This is a Layer 3 logical interface that is created on a per VLAN basis. A Vlanif interface is named in the format "Vlanif + VLAN ID."	Vlanif interfaces are used to implement Layer 3 communication between users in different VLANs and on different network segments. After an IP address is configured for a Vlanif interface, the interface functions as the gateway for users in the VLAN to forward packets across network segments at Layer 3 based on the IP address.
lo interface	This interface is automatically created on device startup. An lo interface uses the loopback address 127.0.0.1/8 to receive data packets destined for the local device. This IP address cannot be changed or advertised using a routing protocol.	The lo interface functions as a device's internal loopback interface.
Null interface	This interface is automatically created on device startup. It is always up but cannot forward packets. Packets sent to a Null interface are all discarded. A Null interface cannot be configured with any IP address or encapsulated with any link layer protocol.	Null interfaces can be used to: • Prevent routing loops (most typical usage). For example, a route to a Null interface is always created during route summarization. • Filter traffic. Undesired packets can be sent to Null interfaces to avoid the need for an access control list (ACL). For example, a Null interface can be specified as the next hop of a static route to a network segment, thereby filtering out all the data packets destined for that network segment.

2.1.1.2 Interface Numbering Rules

Management Interface Numbering Rules

The following table lists the numbers of management interfaces.

Table 2-4 Management interface numbers

Interface	Management Interface Number
Console interface	console 0
MEth interface	MEth 0/0/0

Service Interface Numbering Rules

The following describes the numbering rules of service interfaces.

Service interfaces are numbered in the format of slot ID/subcard ID/interface number.

- Slot ID: indicates the slot where a board resides.
- Subcard ID: indicates the ID of a subcard on a board. Currently, boards do not support subcards, and the subcard ID is 0.
- Interface number: indicates the sequence number of an interface on a board.

2.1.2 Configuration Precautions for Interface Basic

Licensing Requirements

Interface Basics are not under license control.

Hardware Requirements

Table 2-5 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

2.1.3 Configuring an MTU for an Interface

Context

The size of each data packet is limited at the network layer. Upon receiving an IP packet to be forwarded, the network layer checks to which local interface it needs to send the packet and obtains the maximum transmission unit (MTU) of the interface. The network layer then compares the MTU with the packet length. If the

packet length is longer than the MTU, the network layer fragments the packet into chunks no longer than the MTU.

An MTU value determines the maximum number of bytes that an interface can send each time. A suitable MTU is therefore necessary for normal and efficient communication between network devices.

- If the MTU is too small and the size of packets is large, the packets may be broken into many fragments and discarded by QoS queues. This affects normal data transmission.
- If the MTU of the local interface is too large and the size of the packets sent by the interface exceeds the MTU supported by a transit node or a receiver, the transit node or receiver fragments the packets or may even discard them. This affects the network load and normal data transmission.

For details about configuration parameters, see huawei-ifm.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name { interface-name | interface-type interface-number }
```

Step 3 Configure an MTU for the interface.

```
mtu mtu
```

By default, the MTU is 1500 bytes. To ensure that large packets are not dropped due to MTU mismatch, perform this step to forcibly fragment large packets. The MTU ranges from 68 to 1610 bytes.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display this all** command to check the running status of interfaces.

2.1.4 Enabling or Disabling an Interface

Context

To implement the modifications made to an interface's parameters, run the **admin-status down** and **admin-status up** commands in turn.

When an interface is not connected to a cable or a fiber, disable the interface by using the **admin-status down** command to prevent exceptions caused by interference.

For details about configuration parameters, see huawei-ifm.yang.

NOTE

- A NULL interface is always up and cannot be enabled or disabled by commands.
- A loopback interface is always up and cannot be enabled or disabled by commands.

Procedure

- Disable an interface.
  a. Enter the edit-config view.
     edit-config
  b. Enter the interface view.
     ifm interfaces interface name { *interface-name* | *interface-type interface-number* }
  c. Disable the interface.
     admin-status down

     By default, an interface is enabled.
  d. Commit the configuration.
     commit
- Enable an interface.
  a. Enter the edit-config view.
     edit-config
  b. Enter the interface view.
     ifm interfaces interface name { *interface-name* | *interface-type interface-number* }
  c. Enable the interface.
     admin-status up

     By default, an interface is enabled.
  d. Commit the configuration.
     commit

----End

2.1.5 Maintaining Interfaces

Context

For details about configuration parameters, see huawei-ifm.yang.

NOTICE

Interface statistics cannot be restored after they are cleared. Confirm your action before you clear interface statistics.

Procedure

To perform interface maintenance operations, run the corresponding commands listed in **Table 2-6** in the user view.

Table 2-6 Interface maintenance operations

Operation	Command	Description
Check the interface status.	**display /ifm/interfaces/ interface[name="**[*interface-name* | *interface-type interface-number*]"]**/dynamic/ all**	If an interface fails, you can check its status.
Check packet statistics on interfaces.	**display /ifm/interfaces/ interface[name="***interface-name* | *interface-type interface-number*"]**/mib-statistics/ all**	By checking packet statistics on interfaces, you can analyze the network status based on traffic statistics and error packets.
Clear interface statistics.	**reset-if-mib-counters-by-name if-name** [*interface-name* | *interface-type* [*interface-number*]]	To monitor the status of an interface or locate faults on the interface, collect traffic statistics on the interface. Before collecting traffic statistics on an interface within a certain period, clear the existing traffic statistics on the interface.

2.2 Logical Interface Configuration

2.2.1 Overview of Logical Interfaces

Logical interfaces are virtual interfaces that need to be manually configured to exchange data. **Table 2-7** describes the logical interfaces supported by devices.

Table 2-7 Logical interfaces

Interface Type	Interface Description	Application Scenario
Vlanif interface	This is a Layer 3 logical interface that is created on a per VLAN basis. A Vlanif interface is named in the format "Vlanif + VLAN ID."	Vlanif interfaces are used to implement Layer 3 communication between users in different VLANs and on different network segments. After an IP address is configured for a Vlanif interface, the interface functions as the gateway for users in the VLAN to forward packets across network segments at Layer 3 based on the IP address.
lo interface	This interface is automatically created on device startup. An lo interface uses the loopback address 127.0.0.1/8 to receive data packets destined for the local device. This IP address cannot be changed or advertised using a routing protocol.	The lo interface functions as a device's internal loopback interface.

Interface Type	Interface Description	Application Scenario
Null interface	This interface is automatically created on device startup. It is always up but cannot forward packets. Packets sent to a Null interface are all discarded. A Null interface cannot be configured with any IP address or encapsulated with any link layer protocol.	Null interfaces can be used to: • Prevent routing loops (most typical usage). For example, a route to a Null interface is always created during route summarization. • Filter traffic. Undesired packets can be sent to Null interfaces to avoid the need for an access control list (ACL). For example, a Null interface can be specified as the next hop of a static route to a network segment, thereby filtering out all the data packets destined for that network segment.

Interface Type	Interface Description	Application Scenario
Loopback interface	This is a logical interface. Any data packet sent to a loopback interface is considered to be sent to the device itself. A loopback interface has the following characteristics: ● Once created, its physical status and link protocol status remain up until it is deleted, even if the interface has no IP address configured. A loopback IP address can be used if you need an IP address of an interface that is usually up. ● The IP address of a loopback interface can be advertised immediately after being configured. A loopback interface can be assigned an IP address with a 32-bit mask, which reduces address consumption. ● A device directly discards a packet if the outbound interface of the packet that needs to be sent out is a local loopback interface. ● A loopback interface cannot be encapsulated with any link layer protocol. Therefore, negotiation is not performed at the data link layer, and the data link protocol status is usually up. InLoopback0 is a special and fixed loopback	Loopback interfaces can be used to: ● Improve network reliability when their IP addresses are specified as the source IP addresses of packets. ● Control access interfaces and filter logs to simplify displayed information.

Interface Type	Interface Description	Application Scenario
	interface, which is automatically created on device startup. This interface uses the fixed loopback address 127.0.0.1/8 to receive all data packets destined for the device where the InLoopback0 interface resides. This IP address cannot be advertised using routing protocols.	

2.2.2 Configuration Precautions for Logical Interface

Licensing Requirements

Logical Interface are not under license control.

Hardware Requirements

Table 2-8 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

2.2.3 Default Settings for Logical Interfaces

Table 2-9 describes the default settings for logical interfaces.

Table 2-9 Default settings for logical interfaces

Interface Type	Default Setting
Vlanif interface	A Vlanif interface is created automatically on device startup.

Interface Type	Default Setting
Virtual-if interface	A Virtual-if interface is automatically generated and assigned an interface number during the creation of a virtual system.
lo interface	An lo interface is created automatically on device startup.
Null interface	A Null interface is created automatically on device startup.

2.2.4 Maintaining Logical Interfaces

Context

By collecting traffic statistics on an interface, you can monitor its status and locate faults. Before collecting traffic statistics on an Ethernet interface, clear the existing traffic statistics on the interface.

For details about configuration parameters, see huawei-ifm.yang.

To clear statistics, run the following command as required.

NOTICE

Statistics cannot be restored after being cleared. Exercise caution when clearing the statistics.

Table 2-10 Clearing statistics

Operation	Command
Clear traffic statistics on an interface.	**reset-if-mib-counters-by-name if-name** { *interface-name* | *interface-type interface-number* }

2.3 Ethernet Interface Configuration

2.3.1 Overview of Ethernet Interfaces

Definition

Ethernet interfaces, both electrical and optical, are used on local area networks (LANs).

To adapt to network requirements, Ethernet interfaces on a device are defined as follows:

- Layer 2 Ethernet interface: is a physical interface that works at the data link layer and cannot be configured with an IP address. It can forward received

packets at Layer 2 or be added to a VLAN to forward received packets at Layer 3 through corresponding VLANIF interfaces.

- Layer 3 Ethernet interface: is a physical interface that works at the network layer and can be configured with an IP address. It can forward received packets at Layer 3.

Layer 2 Ethernet interface

A Layer 2 Ethernet interface can be an electrical or optical interface. Electrical interfaces include GE electrical interfaces, and optical interfaces include GE optical interfaces.

Table 2-11 describes the attributes of Layer 2 Ethernet interfaces.

Table 2-11 Layer 2 Ethernet interfaces

Interface Type	Rate (Mbit/s)	Duplex Mode	Auto-negotiation
GE electrical interface	10	Full-duplex/Half-duplex	Supported
	100	Full-duplex/Half-duplex	
	1000	Full-duplex	
GE optical interface	100	Full-duplex	Supported
	1000	Full-duplex	

Layer 3 Ethernet interface

A Layer 3 Ethernet interface can be an electrical or optical interface. Electrical interfaces include GE electrical interfaces, and optical interfaces include GE optical interfaces.

Table 2-12 Layer 3 Ethernet interfaces

Interface Type	Rate (Mbit/s)	Duplex Mode	Auto-negotiation
GE electrical interface	10	Full-duplex/Half-duplex	Supported
	100	Full-duplex/Half-duplex	Supported
	1000	Full-duplex	Supported
GE optical interface	100	Full-duplex	Supported
	1000	Full-duplex	Supported

2.3.2 Configuration Precautions for Ethernet interface

Licensing Requirements

Ethernet interfaces are not under license control.

Hardware Requirements

Table 2-13 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

2.3.3 Default Settings for Ethernet Interfaces

Table 2-14 describes the default settings for Ethernet interfaces.

Table 2-14 Default settings for Ethernet interfaces

Parameter	Default Setting
Duplex mode	Full-duplex
Auto-negotiation	Enabled
Loopback detection	Disabled
Combo	Auto
Admin-status	Down
MTU	1500

2.3.4 Configuring an Ethernet Interface to Work in Auto-negotiation Mode

Context

On a network, devices may have different transmission capabilities and must negotiate a proper data transmission capability to communicate with each other. The auto-negotiation function enables connected devices at both ends of a

physical link to exchange information so that they can automatically choose the same working parameters and work at the maximum transmission capability that both devices support.

The parameters negotiated automatically include the duplex and FEC modes as well as the working rate. If the negotiation succeeds, the involved devices work in the same duplex and FEC modes, and at the same rate. If auto-negotiation is disabled on the devices, the operating parameters must be manually set.

For details about configuration parameters, see huawei-devm.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the Ethernet interface view.

```
devm ports port position position
ethernet
```

Step 3 Configure the Ethernet interface to work in auto-negotiation mode.

```
negotiation enabled
```

NOTE

If interface hardware complies with auto-negotiation standards, it is recommended that Ethernet interfaces work in auto-negotiation mode.

Manually setting interface rates usually complicates network planning and maintenance, and improper settings will affect or even interrupt the network communication.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display/devm/ports/port[position=*position*]/ all** command in any view to check the running status of the interface. The **negotiation** field in the command output shows the negotiation status.

2.3.5 Configuring Attributes for Ethernet Electrical Interfaces

2.3.5.1 Configuring the Duplex Mode

Context

The device supports half-duplex and full-duplex modes:

- In half-duplex mode, an Ethernet interface sends or receives data only within the specified maximum transmission distance at a time.
- In full-duplex mode, an Ethernet interface sends and receives data at the same time. The maximum throughput in full-duplex mode doubles that in half-duplex mode, and there is no limit on the maximum transmission distance.

You can configure the duplex mode of an Ethernet electrical interface working in either auto-negotiation or non-auto-negotiation mode.

- In auto-negotiation mode, interfaces on both ends of a link negotiate their duplex mode. If the negotiated duplex mode is not the required one, you can manually change the duplex mode. For example, two interfaces support both full duplex mode and half duplex mode. If the two interfaces negotiate to work in half duplex mode, but they are required to work in full duplex mode, you can set the full duplex mode for the two interfaces.
- In non-auto negotiation mode, you can set the required duplex mode for interfaces manually.

For details about configuration parameters, see huawei-devm.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the Ethernet interface view.

```
devm ports port position position
ethernet
```

Step 3 Enable the non-auto-negotiation mode.

```
negotiation disabled
```

By default, an Ethernet interface works in auto-negotiation mode.

Step 4 (Optional) Configure the rate of the Ethernet interface.

```
speed { 10M | 100M | 1000M }
```

Step 5 (Optional) Configure the working mode of the Ethernet interface.

```
duplex { half | full }
```

Step 6 Commit the configuration.

```
commit
```

----End

2.3.6 Configuring the Working Mode of a Combo Interface

Context

A combo interface is a logical interface and corresponds to a GE electrical interface and a GE optical interface on the device panel. The GE electrical interface and GE optical interface share one internal forwarding interface and are multiplexed and unable to work at the same time. This means that when one interface is enabled, the other is disabled. You can use the electrical or optical interface as required. In addition, the electrical and optical interfaces share one interface view. When enabling the electrical or optical interface and configuring interface parameters (such as the rate and duplex mode), ensure that you use the same interface view.

For details about configuration parameters, see huawei-devm.yang.

NOTE

If a combo interface is configured to work in a different mode from the peer interface, the two interfaces cannot communicate with each other.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the Ethernet interface view.

```
devm ports port position position
ethernet
```

Step 3 Configure the working mode of the combo interface.

```
combo { auto | copper | fiber }
```

By default, a combo interface works in auto mode.

Step 4 Commit the configuration.

```
commit
```

----End

2.3.7 Maintaining Ethernet Interfaces

2.3.7.1 Configuring Loopback Detection on an Interface

Context

Before testing some special functions such as locating an Ethernet fault, enable loopback detection on the desired Ethernet interface to check whether the interface is working properly. If no fault occurs on the Ethernet interface, the physical and protocol statuses of the interface are always up after loopback detection is enabled. If a fault occurs, the statuses remain down.

Loopback detection classification:

- Hardware loopback
  - The transmit and receive ends are connected with a cable to form a loop so that the device receives the signals sent by itself.
- Software loopback
  - Remote loopback (external loopback), which is used to locate a link fault or test the quality of a link. The local interface does not forward packets received from the remote interface based on their destination addresses. Instead, it sends the packets back to the remote interface.
  - Local loopback (internal loopback), which is used to locate a system fault. Packets sent from an interface are sent back to the local device.

  For details about configuration parameters, see huawei-devm.yang.

NOTICE

The loopback detection function affects other functions and may prevent the interface or link from working properly. When the test is complete, run the **loopback-mode noLoopback** command to disable loopback detection. The original configuration is restored after loopback detection is disabled.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the Ethernet interface view.

```
devm ports port position position
```

Step 3 Configure loopback detection on the Ethernet interface.

```
loopback-mode { localLoopback | noLoopback }
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display/devm/ports/port[position=*position*]/ all** command in any view to check the running status of the interface. The **loopback-mode** field in the command output shows the loopback status.

3 System Management Configuration

3.1 Hardware Management Configuration

3.1.1 Overview of Hardware Management

Definition

Hardware management is a feature designed to reduce the device failure rate and ensure secure, stable, and reliable running of the system.

Purpose

Ensuring that a device runs stably requires proper network planning and routine device management and maintenance. In terms of hardware management, operations include device reset, optical module management, and temperature

management for the device CPU/main control boards. The hardware management feature monitors the hardware status in real time so that the system can report an alarm as soon as an exception occurs. You can then check alarm information and take corresponding measures to ensure that the system runs securely, stably, and reliably.

3.1.2 Configuration Precautions for Hardware management

Licensing Requirements

Hardware management is not under license control.

Hardware Requirements

Table 3-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.1.3 Checking the Device Status

Context

You can check the device status to obtain hardware information, facilitating the detection and locating of exceptions.

For details about configuration parameters, see huawei-driver.yang, huawei-devm.yang, and huawei-devm-action.yang.

Procedure

Table 3-2 Checking the device status

Operation	Command	Description

Check the equipment serial number (ESN) of a device.	**display devm physical-entitys/physical-entity class mpuModule position 0 serial-number 0/esn all**	Enter the **class**, **position**, and **serial-number** of a device to query the corresponding ESN. The ESN is unique to each device, and is mandatory for license application.
Check electronic labels of a device.	**display driver/ electronic-labels all**	Electronic labels identify the hardware information of a device.
Check device information.	**display driver/device-health-checks all**	You can run this command to check the status of a device.
	display devm/physical-entitys all	You can run this command to check the type, status, and hardware version of a device. The command output excludes information about power modules and fan modules.
	display devm/mpu-boards all	You can run this command to check the memory and flash information of a device.
Check optical module information on device ports.	**display devm/ports all**	-
Check temperature information.	**display driver/ temperature2s all**	You can run this command to check the current temperature of a device. Hardware may be damaged if the device's temperature is too high or low. If the temperature of a main control board or CPU reaches a configured alarm threshold, a corresponding alarm is generated, prompting you to adjust working environment variables.

Check the CPU information of a device.	**display cpu-memory/board-cpu-infos/board-cpu-info[slot-id=0][cpu-id=0]/system-cpu-usage**	You can run this command to check the CPU usage of a device.
	display cpu-memory/board-cpu-infos/board-cpu-info[slot-id=0][cpu-id=0]/overload-threshold	You can run this command to check the CPU overload threshold of a device.
Check the memory information of a device.	**display cpu-memory/board-memory-infos/board-memory-info[slot-id=0][cpu-id=0]/os-memory-total**	You can run this command to check the total physical memory size of a device, in KB.
	display cpu-memory/board-memory-infos/board-memory-info[slot-id=0][cpu-id=0]/os-memory-free	You can run this command to check the remaining physical memory size of a device, in KB.
	display cpu-memory/board-memory-infos/board-memory-info[slot-id=0][cpu-id=0]/os-memory-use	You can run this command to check the size of the physical memory used by a device, in KB.
	display cpu-memory/board-memory-infos/board-memory-info[slot-id=0][cpu-id=0]/os-memory-usage	You can run this command to check the memory usage of a device.
	display cpu-memory/board-memory-infos/board-memory-info[slot-id=0][cpu-id=0]/overload-threshold	You can run this command to check the memory overload threshold of a device.
Check the health status of a device.	**display driver/device-health-checks all**	You can run this command to check the health status of a device.
Check the MAC address of a device.	**display driver/globbal-attribute all**	-
Check the reset information of a device.	**display get-reboot-info all**	-

Check the system basics of a device.	**display system/system-info** [**sys-desc** | **sys-object-id** | **hardware-model** | **product-name** | **esn** | **mac** | **uname** | **boot-time**]	• **sys-desc** indicates the device description. • **sys-object-id** indicates the device OID. • **product-name** indicates the product name. • **hardware-model** indicates the displayed hardware name. • **esn** indicates the device ESN. • **mac** indicates the system MAC address. • **boot-time** indicates the system startup time. • **uname** indicates the device hardware model.

Check the product information, working status, and exception locating information of a device.	**diagnose driver-dfx-infos driver-dfx-info** *info "<commandID>"*	*commandID* indicates the command parameter for querying detailed information. • **i2c common**: displays basic I2C public information. • **i2c statistics**: displays I2C statistics. • **i2c device**: displays information about the I2C device node. • **i2c driver**: displays information about the I2C driver node. • **localbus common**: displays basic public information about the local bus. • **localbus device**: displays information about the localbus device node. • **localbus driver**: displays information about the localbus driver node. • **smi common**: displays basic SMI public information. • **smi device**: displays information about the SMI device node. • **smi driver**: displays information about the smi driver node. • **spi common**: displays basic SPI public information. • **spi device**: displays information about the SPI device node. • **spi driver**: displays information about the SPI driver node. • **pll common**: displays basic PLL public information.

		• **pll reg** *inst-id chip-id*: displays PLL register information. *inst-id* specifies a board ID, and *chip-id* specifies a chip ID. • **pll trace** *inst-id chip-id dpll* : displays the PLL clock source selection information. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, and *dpll* specifies the internal PLL ID of the PLL. • **phy common**: displays basic PHY public information. • **phy reg** *chip-id port-id*: displays the PHY register information. *chip-id* specifies a chip ID, and *port-id* specifies a port ID. • **avs common**: displays basic AVS public information. • **avs chip** *inst-id chip-id*: displays the AVS chip information. *inst-id* specifies a board ID, and *chip-id* specifies a chip ID. • **avs bbox** *inst-id chip-id*: displays the AVS black box information. *inst-id* specifies a board ID, and *chip-id* specifies a chip ID. • **rtc info**: displays basic RTC public information. • **cpld** *inst-id chip-id mod-id*: displays CPLD register information. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, and

		mod-id specifies an internal module ID of the CPLD. • **cpldjtag common info**: displays basic public information about cpldjtag. • **temp common**: displays basic temp public information. • **temp chip** *dev-num*: displays temp chip information. *dev-num* specifies a chip ID. • **volt common**: displays basic volt public information. • **volt chip** *dev-num*: displays volt chip information. *dev-num* specifies a chip ID. • **cdr interface config** *chip-id inf-id inf-rate*: displays the CDR port configuration. *chip-id* specifies a chip ID, *inf-id* specifies a port number, and *inf-rate* specifies a port rate. • **cdr interface status** *chip-id inf-id inf-rate*: displays the CDR port status. *chip-id* specifies a chip ID, *inf-id* specifies a port number, and *inf-rate* specifies a port rate. • **cdr interface snr** *chip-id inf-id inf-rate*: displays the SNR of a CDR port. *chip-id* specifies a chip ID, *inf-id* specifies a port number, and *inf-rate* specifies a port rate. • **cdr interface ffeq** *chip-id chnl-id dir*: displays the FFEQ parameter information

		of a CDR port. *chip-id* specifies a chip ID, *chnl-id* specifies a channel ID, and *dir* specifies a direction. • **cdr chip temperature** *chip-id temp-id*: displays the CDR temperature information. *chip-id* specifies a chip ID, and *temp-id* specifies an internal temperature point of the chip. • **cdr chip reg read** *chip-id dfx-type module-type module-id*: displays CDR register information. *chip-id* specifies a chip ID, *dfx-type* specifies a detection type, *module-type* specifies a module type, and *module-id* specifies a sub-module ID. • **mac-chip status** *chip-id start-index status-num*: displays the register information of a MAC chip. *chip-id* specifies a chip ID, *start-index* specifies a start address, and *status-num* specifies the number of registers to be read. • **mac-chip io** *chip-id*: displays the overall fault information about a MAC chip. *chip-id* specifies a chip ID. • **mac-chip serdes cs** *chip-id serdes-id*: displays the overall parameters and status of the SerDes of a MAC chip. *chip-id* specifies a chip ID, and

		serdes-id specifies a SerDes ID. • **mac-chip serdes tx** *chip-id serdes-id*: displays the transmit parameters and status of the SerDes of a MAC chip. *chip-id* specifies a chip ID, and *serdes-id* specifies a SerDes ID. • **mac-chip serdes rx** *chip-id serdes-id*: displays the receive parameters and status of the SerDes of a MAC chip. *chip-id* specifies a chip ID, and *serdes-id* specifies a SerDes ID. • **mac-chip serdes digit-eye** *chip-id serdes-id*: displays the eye pattern of a MAC chip. *chip-id* specifies a chip ID, and *serdes-id* specifies a SerDes ID. • **mac-chip mib** *chip-id port-id port-type dir*: displays the count of a MAC chip. *chip-id* specifies a chip ID, *port-id* specifies a port number, *port-type* specifies a port type, and *dir* specifies a direction. • **optical dfx info read** *inst-id package-type panel-port*: displays the diagnostic information about an optical module. *inst-id* specifies a board ID, *package-type* specifies an encapsulation type, and *panel-port* specifies a sequence number of the port adopting the encapsulation type.

		• **flash nor statistics**: displays statistics about the NOR flash. • **flash nand statistics**: displays statistics about the NAND flash. • **flash nand part-info**: displays the partition information of the NAND flash. • **flash cpldsfc**: displays the controller information of the CPLD SFC. • **flash sfc** *inst-id*: displays information about the SFC controller. *inst-id* specifies a board ID. • **switch port-status** *inst-id chip-id port-id*: displays the port status of the LAN switch chip. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, and *port-id* specifies a port ID. • **switch port-mib** *inst-id chip-id port-id*: displays the count of LAN switch chip ports. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, and *port-id* specifies a port ID. • **switch serdes-config** *inst-id chip-id port-id*: displays SerDes parameter settings of a LAN switch chip port. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, and *port-id* specifies a port ID. • **switch serdes-eye** *inst-id chip-id port-id*

		lane-id: displays the SerDes eye pattern information of a LAN switch chip port. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, *port-id* specifies a port ID, and *lane-id* specifies a link ID. • **switch vlan-info** *inst-id chip-id vlan-id*: displays the information about a single VLAN of the LAN switch chip. *inst-id* specifies a board ID, *chip-id* specifies a chip ID, and *vlan-id* specifies a VLAN ID. • **switch vlan-all** *inst-id chip-id*: displays all VLAN information about a LAN switch chip. *inst-id* specifies a board ID, and *chip-id* specifies a chip ID. • **switch mac-table** *inst-id chip-id*: displays the MAC entry information of a LAN switch chip. *inst-id* specifies a board ID, and *chip-id* specifies a chip ID. • **chip9xxx info** *inst-id chip-id*: displays information about a 9xxx chip. *inst-id* specifies a board ID, and *chip-id* specifies a chip ID. • **clusterport all**: displays information about all cascade ports. • **clusterport port** *port-id*: displays information about a specified cascade port.

		• **console info**: displays console interface information. • **meth common**: displays management Ethernet information. • **meth phy**: displays physical information about the management Ethernet. • **meth statistics**: displays management Ethernet statistics. • **wwan info** *module-id file-index*: displays basic information about the WWAN module. *module-id* specifies a module ID, and *file-index* specifies an information ID. • **wwan forward info** *file-index*: displays information about the transparent transmission module. *file-index* specifies an information ID. • **wwan pcie info** *file-index*: displays the PCIe bus information. *file-index* specifies an information ID. • **wwan usb info** *file-index channel-id*: displays USB bus information. *file-index* specifies an information ID, and *channel-id* specifies a channel ID. • **wdt runtime**: displays watchdog runtime information. • **wdt last**: displays information about the last watchdog reset.

		• **intr runtime**: displays interrupt running information. • **intr cfg**: displays interrupt configurations. • **intr virq** *virq-id*: displays the status of the interrupt virq. *virq-id* specifies an interrupt ID. • **intr index-type index-id type**: displays the status of the interrupt index_type. *index-id* specifies an interrupt sequence number, and *type* specifies an interrupt type. • **intr monitor**: displays interrupt monitoring information. • **log common**: displays common log information. • **log connection**: displays log connection information. • **log mring-info**: displays log buffer information. • **log diag-cnt**: displays log diagnosis statistics. • **mcu info**: displays MCU information. • **env info**: displays information about environment variables. • **fan driver**: displays fan drive information. • **fan common** *slot-id*: displays the basic running information about the fan. *slot-id* specifies a slot number.

		• **fan cpld**: displays fan CPLD register information. • **fan eeprom** *slot-id*: displays the data stored on the EEPROM chip of the fan module. *slot-id* specifies a slot ID. • **power common**: displays the basic running information about the power supply. • **power config**: displays the power supply configuration. • **power eeprom-chip**: displays the data stored in the EEPROM chip of the power supply. • **power power-chip**: displays the register information of the chip on the power supply. • **backplane info**: displays backplane management information. • **board info**: displays board management information. • **version**: displays version information. • **card info** *card-id*: displays subcard information. *card-id* specifies a subcard ID. • **elabel info**: displays electronic label information. • **elabel runtime**: displays electronic label runtime information.

		• **elabel cfg**: displays the electronic label configuration. • **elabel hardware**: displays the electronic label data in hardware. • **elabel single-info** *inst-id elabel-id*: displays information about a specified electronic label. *inst-id* specifies a board ID, and *elabel-id* specifies an electronic label ID. • **elabel single-hardware** *inst-id elabel-id*: displays data in a specified electronic label. *inst-id* specifies a board ID, and *elabel-id* specifies an electronic label ID. • **mac board**: displays the MAC address of a board. • **mac card** *card-id*: displays the MAC address of a subcard. *card-id* specifies a subcard ID. • **mac cfg**: displays MAC configuration information. • **mac type**: displays the MAC type. • **msc info** *switch-mode*: displays information about the active/standby switchover. *switch-mode* specifies a type of the active/standby switchover. • **reset-mng**: displays reset information about the local board. • **cpu-reset-reason** *cpu-id index*: displays the cause of the CPU

		reset. *cpu-id* specifies a CPU ID, and *index* specifies a reset index. • **sata info**: displays SATA hardware information. • **usb info**: displays USB hardware information. • **woffline**: displays micro switch information. • **slave-cpu info** *cpu-id module*: displays information about the slave CPU. • **cpu hha key-regs** *dev-id*: displays the HHA common register. *dev-id* specifies an HHA ID, ranging from 0 to 64. • **cpu ddr reg** *dev-id offset len*: displays the DDR common register. *dev-id* specifies a DDR ID, ranging from 0 to 64. • **cpu ddr key-regs** *dev-id*: displays the DDR key register. • **cpu ddr ecc info** *dev-id*: displays ECC information of the DDR module. *dev-id* specifies a DDR ID, ranging from 0 to 64. • **cpu ddr config** *dev-id*: displays the DDR configuration. • **cpu ddr status** *dev-id*: displays the DDR status. • **cpu ddr isolate**: displays memory isolation information. • **cpu pcie common-reg** *dev-id offset len*: displays the PCIe common register.

		• **cpu pcie port-reg** *dev-id offset len*: displays the PCIe port register. • **cpu pcie config** *port-id*: displays the PCIe configuration. • **cpu pcie ltssm** *port-id*: displays the PCIe LTSSM register. • **cpu pcie status** *port-id*: displays the PCIe status register. • **cpu pcie config-reg** *dev-id reg-addr number*: displays the PCIe configuration register. *dev-id* specifies a device ID (domain:bus:devid:func), *reg-addr* specifies a register address, and *number* specifies the number of registers. • **cpu ge reg** *dev-id port-id offset len*: displays the GE register. *dev-id* specifies a device ID, *port-id* specifies a port ID, *offset* specifies a register offset, and *len* specifies the read length (4-byte aligned). • **cpu ge phy-reg** *dev-id port-id page reg-addr*: displays the GE internal PHY register. *dev-id* specifies a device ID, *port-id* specifies a port ID, which is reserved, *page* specifies the number of register pages, and *reg-addr* specifies a register address. • **cpu ge port stats** *dev-id start-port-id end-*

		port-id: displays GE port statistics. *dev-id* specifies a device ID, *start-port-id* specifies a starting port ID, and *end-port-id* specifies an end port ID. • **cpu ge port config** *dev-id start-port-id end-port-id* : displays the GE port configuration. • **cpu ge port status** *dev-id start-port-id end-port-id*: displays the GE port status. *dev-id* specifies a device ID, *start-port-id* specifies a starting port ID, and *end-port-id* specifies an end port ID. • **cpu ppe reg** *dev-id offset len*: displays the PPE register. • **cpu ppe tunnel-reg** *dev-id offset len*: displays the PPE TUNNEL register. • **cpu ppe common stats** *dev-id*: displays PPE common statistics. • **cpu ppe tunnel stats** *dev-id start-tnl end-tnl*: displays PPE TUNNEL statistics. • **cpu ppe pa stats** *dev-id*: displays PPE PA statistics. • **cpu ppe ssu stats** *dev-id*: displays PPE SSU statistics. • **cpu ppe ppp stats** *dev-id*: displays PPE PPP statistics. • **cpu ppe rpu stats** *dev-id*: displays PPE RPU statistics.

		• **cpu ppe tpu stats** *dev-id*: displays PPE TPU statistics. • **cpu ppe dma stats** *dev-id*: displays PPE DMA statistics. • **cpu ppe mam stats** *dev-id*: displays PPE MAM statistics. • **cpu ppe rcb stats** *dev-id*: displays PPE RCB statistics. • **cpu ppe tm stats** *dev-id*: displays PPE TM statistics. • **cpu ppe backpress stats** *dev-id*: displays PPE backpress statistics. • **cpu ppe drop stats** *dev-id*: displays PPE packet drop statistics. • **cpu ppe exception stats** *dev-id*: displays PPE exception statistics. • **cpu ppe cmdq stats** *dev-id*: displays PPE CMDQ statistics. • **cpu ppe bios stats** *dev-id*: displays PPE BIOS statistics. • **cpu ppe common config** *dev-id*: displays the PPE common configuration. • **cpu ppe tunnel config** *dev-id start-tnl end-tnl*: displays the PPE TUNNEL configuration. • **cpu ppe pa config** *dev-id*: displays the PPE PA configuration. • **cpu ppe ssu config** *dev-id*: displays the PPE SSU configuration.

		• **cpu ppe ppp config** *dev-id*: displays the PPE PPP configuration. • **cpu ppe rpu config** *dev-id*: displays the PPE RPU configuration. • **cpu ppe tpu config** *dev-id*: displays the PPE TPU configuration. • **cpu ppe dma config** *dev-id*: displays the PPE DMA configuration. • **cpu ppe mam config** *dev-id*: displays the PPE MAM configuration. • **cpu ppe rcb config** *dev-id*: displays the PPE RCB configuration. • **cpu ppe tm config** *dev-id*: displays the PPE TM configuration. • **cpu ppe backpress config** *dev-id*: displays the PPE backpress configuration. • **cpu ppe bios config** *dev-id*: displays the PPE BIOS configuration. • **cpu xge reg** *dev-id port-id offset len*: displays the XGE register. *dev-id* specifies a device ID, *port-id* specifies a port ID, *offset* specifies a register offset, and *len* specifies the read length (4-byte aligned). • **cpu xge port stats** *dev-id start-port-id end-port-id*: displays statistics about XGE ports.

		• **cpu xge port config** *dev-id start-port-id end-port-id* : displays the XGE port configuration. • **cpu xge port status** *dev-id start-port-id end-port-id*: displays the XGE port status. *dev-id* specifies a device ID, *start-port-id* specifies a starting port ID, and *end-port-id* specifies an end port ID. • **cpu rcb common-key-regs** *mac-id*: displays the RCB common key register. • **cpu rcb queue-key-regs** *mac-id tqp-id*: displays the RCB queue key register. • **cpu rcb stats** *mac-id tqp-id*: displays RCB statistics. • **cpu lbc reg** *dev-id offset len*: displays the LocalBus register. • **cpu lbc config** *dev-id*: displays the LocalBus configuration. • **cpu serdes tx-param** *macro-id lane-id*: displays SerDes transmit parameters. • **cpu serdes rx-param** *macro-id lane-id*: displays SerDes receive parameters. • **cpu serdes foureyes** *macro-id lane-id*: displays the SerDes four-eye pattern. • **cpu serdes pll** *macro-id cs-id*: displays the SerDes PLL status. • **cpu serdes common-reg** *macro-id offset*

		len: displays the SerDes common register. • **cpu serdes cs-reg** *macro-id cs-id offset len*: displays the SerDes CS register. • **cpu serdes ds-reg** *macro-id lane-id offset len*: displays the SerDes DS register. • **cpu serdes common-config** *macro-id*: displays the SerDes common configuration. • **cpu serdes cs-config** *macro-id cs-id*: displays the SerDes CS configuration. • **cpu serdes ds-config** *macro-id lane-id*: displays the SerDes DS configuration. • **cpu sllc key-regs** *dev-id*: displays the SLLC key register. • **cpu pll key-regs** *port-id*: displays the PLL key register. • **cpu sllc reg** *port-id offset len*: displays the SLLC register. • **cpu pll status**: displays the CPU PLL status. • **cpu sram key-regs** *dev-id*: displays the SRAM key register. • **cpu l1cache key-regs** *cache-id*: displays the L1 cache register. • **cpu l2cache key-regs** *cache-id*: displays the L2 cache register. *cache-id* specifies a cache ID. The command execution

		may be suspended when the L2 cache is in the WFX state. • **cpu l3cache key-regs** *cache-id*: displays the L3 cache register. • **cpu tsensor key-regs** *tensor-id*: displays the TSENSOR key register. • **cpu dsaf igu-egu** *mac-id*: displays DSAF IGU and EGU information. • **cpu dsaf ssu** *mac-id*: displays DSAF SSU information. • **cpu nic key-regs** *dev-id*: displays the NIC key register. • **cpu i2c key-regs** *bus-id*: displays the I2C key register. • **cpu i2c stats** *bus-id*: displays I2C statistics. • **cpu spi key-regs** *dev-id*: displays the SPI key register. *dev-id* specifies a device ID. • **cpu spi stats** *bus-id*: displays SPI statistics. • **cpu spi history** *bus-id*: displays SPI history information. • **cpu mdio key-regs** *dev-id*: displays the MDIO key register. • **cpu mdio stats** *bus-id*: displays MDIO statistics. • **cpu uart key-regs** *dev-id*: displays the key register of the serial port. *dev-id* specifies a serial port ID. • **cpu uart stats** *dev-id*: displays serial port statistics. *dev-id*

		specifies a serial port ID. • **cpu nand key-regs** *ctrl-id*: displays the NAND key register. • **cpu nand chip-info** *ctrl-id*: displays the NAND chip information. • **cpu nand bad-block** *ctrl-id*: displays the NAND bad block information. • **cpu nand oper-stats** *ctrl-id*: displays NAND operation statistics. • **cpu nand single-block-life** *ctrl-id start-block num*: displays the number of erase times of the NAND block. • **cpu nand mtd-info** *ctrl-id*: displays the NAND MTD information. • **cpu nand page-oob** *mtd-id offset len*: displays the management area of the NAND flash. • **cpu sec common stats** *dev-id*: displays SEC common statistics. • **cpu sec vf stats** *dev-id vf-id*: displays SEC VF statistics. • **cpu sec queue stats** *dev-id vf-id q-id*: displays SEC queue statistics. • **cpu sec common config** *dev-id*: displays the SEC common configuration. • **cpu sec vf config** *dev-id vf-id*: displays the SEC VF configuration.

		• **cpu sec queue config** *dev-id vf-id q-id*: displays the SEC queue configuration. • **cpu sec common status** *dev-id*: displays the SEC common status. • **cpu norflash ctrl-key-regs** *spictl-id*: displays the key register of the flash controller. • **cpu norflash chip-info**: displays the flash chip information. • **cpu norflash oper-stats**: displays flash operation statistics. • **cpu norflash single-block-life** *flash-id block-offset length*: displays the number of erase times of the flash block. • **cpu rsa stats** *dev-id*: displays RSA statistics. • **cpu rsa config** *dev-id*: displays the RSA configuration. • **cpu rsa status** *dev-id*: displays the RSA status. • **cpu smmu reg** *smmu-id offset len*: displays the SMMU register. • **cpu smmu key-regs** *smmu-id*: displays the SMMU key register. • **cpu usb key-regs** *usb-id*: displays the USB key register. • **cpu usb stats** *dev-id*: displays USB statistics. *dev-id* specifies a device ID. • **cpu aa key-regs** *aaf-id*: displays the AAF key register.

		• **cpu poe reg** *dev-id offset len*: displays the POE register. • **cpu poe common stats** *dev-id*: displays POE common statistics. • **cpu poe group stats** *dev-id start-grp-id end-grp-id*: displays POE group statistics. • **cpu poe lcpu stats** *dev-id start-lcpu-id end-lcpu-id*: displays POE LCPU statistics. • **cpu poe queue stats** *dev-id start-q-id end-q-id*: displays POE queue statistics. • **cpu poe backpress status** *dev-id*: displays the POE backpress status. • **cpu poe exception status** *dev-id*: displays the POE exception status. • **cpu poe drop stats** *dev-id*: displays POE packet drop statistics. • **cpu poe common config** *dev-id*: displays the POE common configuration. • **cpu poe group config** *dev-id start-grp-id end-grp-id*: displays the POE group configuration. • **cpu poe lcpu config** *dev-id start-lcpu-id end-lcpu-id*: displays the POE LCPU configuration. • **cpu poe queue config** *dev-id start-q-id end-q-id*: displays the POE queue configuration.

		• **cpu poe backpress config** *dev-id*: displays the POE backpress configuration. • **cpu bmu reg** *dev-id offset len*: displays the BMU register. • **cpu bmu common stats** *dev-id*: displays BMU common statistics. • **cpu bmu phy-pool stats** *dev-id start-pp-id end-pp-id*: displays BMU physical pool statistics. • **cpu bmu pool stats** *dev-id start-lp-id end-lp-id*: displays BMU logical pool statistics. • **cpu bmu backpress status** *dev-id*: displays the BMU backpress status. • **cpu bmu exception status** *dev-id*: displays the BMU exception status. • **cpu bmu abn-int status** *dev-id*: displays the BMU abnormal interruption status. *dev-id* specifies a device ID. • **cpu bmu common config** *dev-id*: displays the BMU common configuration. • **cpu bmu phy-pool config** *dev-id start-pp-id end-pp-id*: displays the BMU physical pool configuration. • **cpu bmu pool config** *dev-id start-lp-id end-lp-id*: displays the BMU logical pool configuration.

		• **cpu bmu backpress config** *dev-id*: displays the BMU backpress configuration. • **cpu tm reg** *dev-id offset len*: displays the TM register. • **cpu tm common stats** *dev-id*: displays TM common statistics. • **cpu tm port stats** *dev-id start-port-id end-port-id*: displays TM port statistics. • **cpu tm trunk stats** *dev-id start-tk-id end-tk-id*: displays TM TRUNK statistics. • **cpu tm gloc stats** *dev-id start-gloc-id end-gloc-id*: displays TM GLOC statistics. • **cpu tm loc stats** *dev-id start-loc-id end-loc-id*: displays TM LOC statistics. • **cpu tm queue stats** *dev-id start-q-id end-q-id*: displays TM queue statistics. • **cpu tm red stats** *dev-id start-red-id end-red-id*: displays TM RED statistics. • **cpu tm sid stats** *dev-id start-sid-id end-sid-id*: displays TM SID statistics. • **cpu tm backpress stats** *dev-id*: displays TM backpress statistics. • **cpu tm exception stats** *dev-id*: displays TM exception statistics. • **cpu tm drop stats** *dev-id*: displays TM packet drop statistics.

		• **cpu tm common config** *dev-id*: displays the TM common configuration. • **cpu tm port config** *dev-id start-port-id end-port-id*: displays the TM port configuration. • **cpu tm trunk config** *dev-id start-tk-id end-tk-id*: displays the TM TRUNK configuration. • **cpu tm gloc config** *dev-id start-gloc-id end-gloc-id*: displays the TM GLOC configuration. • **cpu tm loc config** *dev-id start-loc-id end-loc-id*: displays the TM LOC configuration. • **cpu tm queue config** *dev-id start-q-id end-q-id*: displays the TM queue configuration. • **cpu tm red config** *dev-id start-red-id end-red-id*: displays the TM RED configuration. • **cpu tm sid config** *dev-id start-sid-id end-sid-id*: displays the TM SID configuration. • **cpu tm backpress config** *dev-id*: displays the TM backpress configuration. • **cpu trng key-regs** *dev-id*: displays the TRNG key register. • **cpu cross-station** *key-regs cs-id*: displays the cross station register. • **cpu sysctl reg** *sysctl-id offset len*: displays the SYSCTL register.

		• **cpu sysctl key-regs** *sysctl-id*: displays the SYSCTL key register. • **cpu sysctl pinmode** *sysctl-id*: displays the SYSCTL pin multiplexing register. • **cpu mn key-regs** *mn-id*: displays the MN key register. • **cpu fabric key-regs** *dev-id*: displays the FABRIC key register. • **cpu hpre common-reg** *dev-id offset len*: displays the HPRE common register. • **cpu hpre cluster-reg** *dev-id offset len*: displays the HPRE cluster register. • **cpu hpre common config** *dev-id*: displays the HPRE common configuration. • **cpu hpre vf config** *dev-id vf-id*: displays the HPRE VF configuration. • **cpu hpre queue config** *dev-id vf-id q-id*: displays the HPRE queue configuration. • **cpu hpre status** *dev-id*: displays the HPRE common status. • **cpu hpre stats** *dev-id*: displays HPRE common statistics. • **cpu hpre vf stats** *dev-id vf-id*: displays HPRE VF statistics. • **cpu hpre queue stats** *dev-id vf-id q-id*: displays HPRE queue statistics.

		• **cpu ppp reg** *dev-id offset len*: displays the PPP register. • **cpu ppp common config** *dev-id*: displays the PPP common configuration. • **cpu ppp direct-table config** *dev-id start-tnl end-tnl*: displays the configuration of the PPP direct table. • **cpu ppp multi-table config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP multicast leaf table. • **cpu ppp vlan-table config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP VLAN filter table. • **cpu ppp qos-table config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP QOS parse table. • **cpu ppp flow-table config** *dev-id*: displays basic configuration of the PPP flow table. • **cpu ppp first-dis config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-1 distribution flow table. • **cpu ppp second-dis config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-2 distribution flow table.

		• **cpu ppp first-ingress config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-1 ingress flow table. • **cpu ppp second-ingress config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-2 ingress flow table. • **cpu ppp first-egress config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-1 egress flow table. • **cpu ppp first-dis-form config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-1 distribution flow table template. • **cpu ppp second-dis-form config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-2 distribution flow table template. • **cpu ppp first-ingress-form config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-1 ingress flow table template. • **cpu ppp second-ingress-form config** *dev-id start-table-id end-table-id*: displays the configuration of

		the PPP level-2 ingress flow table template. • **cpu ppp first-egress-form config** *dev-id start-table-id end-table-id*: displays the configuration of the PPP level-1 egress flow table template. • **cpu ppp dis-formwork match** *dev-id start-table-id end-table-id*: displays the match table of the PPP distribution table template. • **cpu ppp ingress-flow formwork-match** *dev-id start-table-id end-table-id*: displays the match table of the PPP ingress table template. • **cpu ppp egress-flow formwork-match** *dev-id start-table-id end-table-id*: displays the match table of the PPP egress table template. • **cpu ppp selfdef dis-table** *dev-id start-table-id end-table-id*: displays the PPP self-defined distribution table. • **cpu ppp rss-hash config** *dev-id*: displays the configuration of the PPP RSS hash algorithm. • **cpu ppp rss-hash table** *dev-id start-table-id end-table-id*: displays the configuration of the PPP RSS lookup table. • **cpu ppp rss-tc config** *dev-id start-table-id*

		end-table-id: displays the configuration of the PPP RSS TC. • **cpu ppp ingress property** *dev-id start-ingress-id end-ingress-id*: displays PPP ingress attributes. • **cpu ppp egress property dev-id** *start-egress-id end-egress-id*: displays PPP egress attributes. • **cpu ppp port-filtering config** *dev-id start-tnl-id end-tnl-id*: displays the PPP port filtering configuration. • **cpu ppp port-isolate config** *dev-id start-tnl-id end-tnl-id*: displays the PPP port isolation configuration. • **cpu ppp ingress-image config** *dev-id start-ingress-id end-ingress-id*: displays the PPP ingress image configuration. • **cpu ppp egress-image config** *dev-id start-egress-id end-egress-id*: displays the PPP egress image configuration. • **cpu ppp l2-base config** *dev-id start-table-id end-table-id*: displays the basic configuration of the PPP L2 table. • **cpu ppp l2-table config** *dev-id start-table-id end-table-id*: displays the entry configuration of the PPP L2 table. • **cpu ppp car-flow config** *dev-id start-*

		table-id end-table-id: displays the PPP CAR throttling configuration. • **cpu ppp promisc-table** *dev-id start-table-id end-table-id*: displays the PPP promiscuous table. • **cpu sata reg** *sata-id offset len*: displays the SATA register. • **cpu sata common-config** *sata-id*: displays the common SATA configurations. • **cpu sata port-config** *sata-id port-id*: displays the SATA port configuration. • **cpu sata status** *sata-id*: displays the SATA status. *sata-id* specifies the SATA ID. • **cpu gpio reg** *dev-id offset len*: displays the GPIO register. • **cpu gpio config** *gpio-id*: displays the GPIO configuration. • **cpu gpio ctrl-mode** *gpio-id*: displays the GPIO control mode. • **cpu mag common stats** *dev-id*: displays MAG common statistics. • **cpu mag port stats** *dev-id start-port-id end-port-id*: displays MAG port statistics. • **cpu mag common config** *dev-id*: displays the MAG common configuration. • **cpu mag port config** *dev-id start-port-id end-port-id*: displays

		the MAG port configuration. • **cpu toe stats** *dev-id*: displays TOE common statistics. • **cpu toe config** *dev-id*: displays the TOE common configuration. • **cpu toe status** *dev-id*: displays the TOE common status. • **cpu fec stats** *dev-id*: displays FEC common statistics. • **cpu fec vf stats** *dev-id vf-id*: displays FEC VF statistics. • **cpu fec queue stats** *dev-id vf-id q-id*: displays FEC queue statistics. • **cpu fec config** *dev-id*: displays the FEC common configuration. • **cpu fec vf config** *dev-id vf-id*: displays the FEC VF configuration. • **cpu fec queue config** *dev-id vf-id q-id*: displays the FEC queue configuration. • **cpu fec common status** *dev-id*: displays the FEC common status. • **cpu fec vf status** *dev-id vf-id*: displays the FEC VF status. • **cpu fec queue status** *dev-id vf-id q-id*: displays the FEC queue status. • **cpu dedup common stats** *dev-id*: displays DEDUP common statistics.

		• **cpu dedup vf stats** *dev-id vf-id*: displays DEDUP VF statistics. • **cpu dedup queue stats** *dev-id vf-id q-id*: displays DEDUP queue statistics. • **cpu dedup common config** *dev-id*: displays the DEDUP common configuration. • **cpu dedup vf config** *dev-id vf-id*: displays the DEDUP VF configuration. • **cpu dedup queue config** *dev-id vf-id q-id*: displays the DEDUP queue configuration. • **cpu dedup common status** *dev-id*: displays the DEDUP common status. • **cpu sa common stats** *dev-id*: displays SA common statistics. • **cpu sa vf stats** *dev-id vf-id*: displays SA VF statistics. • **cpu sa queue stats** *dev-id vf-id q-id*: displays SA queue statistics. • **cpu sa common config** *dev-id*: displays the SA common configuration. • **cpu sa vf config** *dev-id vf-id*: displays the SA VF configuration. • **cpu sa queue config** *dev-id vf-id q-id*: displays the SA queue configuration. • **cpu sa common status** *dev-id*: displays the SA common status.

		• **cpu lge stats** *dev-id start-port-id end-port-id*: displays LGE port statistics. • **cpu lge config** *dev-id start-port-id end-port-id*: displays the LGE port configuration. • **cpu lge status** *dev-id start-port-id end-port-id*: displays the LGE port status. • **cpu cge stats** *dev-id start-port-id end-port-id*: displays CGE port statistics. • **cpu cge config** *dev-id start-port-id end-port-id*: displays the CGE port configuration. • **cpu cge status** *dev-id start-port-id end-port-id*: displays the CGE port status. • **cpu xsec common stats** *dev-id*: displays XSEC common statistics. • **cpu xsec port stats** *dev-id start-port-id end-port-id*: displays XSEC port statistics. • **cpu xsec common config** *dev-id*: displays the XSEC common configuration. • **cpu xsec port config** *dev-id start-port-id end-port-id*: displays the XSEC port configuration. • **cpu dppa common stats** *dev-id*: displays DPPA common statistics. • **cpu dppa port stats** *dev-id start-port-id end-port-id*: displays DPPA port statistics.

		• **cpu dppa common config** *dev-id*: displays the DPPA common configuration. • **cpu dppa port config** *dev-id start-port-id end-port-id*: displays the DPPA port configuration. • **cpu dcfg key-regs**: displays the CPU DCFG register. • **cpu scfg key-regs**: displays the CPU SCFG register. • **cpu qman key-regs**: displays the CPU QMAN register. • **cpu bman key-regs**: displays the CPU BMAN register. • **cpu log imu** *page-id*: displays IMU logs. • **cpu log atf** *page-id*: displays ATF logs. • **cpu log hiboot**: displays HIBOOT logs. • **cpu log user**: displays user logs. • **cpu log hidrv**: displays soft forwarding logs. • **cpu log fmea**: displays FMEA logs. • **cpu hidrv-sys drop-stats**: displays statistics about discarded packets during forwarding. • **cpu hidrv-sys backpress-status**: displays the forwarding backpress status. • **cpu hidrv-sys packet-buffer**: queries statistics about packet congestion during forwarding.

		• **cpu hidrv-sys all-stats**: displays statistics about forwarded packets. • **cpu flowctl info** *flowctl-id*: displays information about the flow control module. *flowctl-id* specifies a network interface mapping ID. • **cpu msr reg** *core-id msr-id msr-num*: displays the MSR register. *core-id* specifies a core ID, *msr-id* specifies a register ID, and *msr-num* specifies the number of registers. • **cpu core info** *core-id*: displays the register information of the CORE (including the coprocessor). *core-id* specifies a core ID. • **cpu iob info** *core-id*: displays information about the IOB module (including register information). *core-id* specifies a core ID. • **cpu qlm config** *qlm-id*: displays the configuration information (including register information) of the QLM module. *qlm-id* specifies a QLM ID. • **cpu qlm status** *qlm-id*: displays the status (including register information) of the QLM module. *qlm-id* specifies a QLM ID. • **cpu bgx config** *bgx-id*: displays the configuration information (including

		register information) of the BGX module. *bgx-id* specifies a BGX ID. • **cpu bgx status** *bgx-id*: displays the status (including register information) of the BGX module. *bgx-id* indicates a BGX ID. • **cpu pki config**: displays the configuration information (including register information) of the PKI module. • **cpu pki group-config** *group-num*: displays the configuration information (including register information) of the PKI module group. *group-num* specifies the number of groups to be displayed. • **cpu pki port-config** *pkind-id*: displays the port configuration information (including register information) of the PKI module. *pkind-id* specifies a pkind ID corresponding to the input port. • **cpu pki pcam-config**: displays the pcam configuration information (including register information) of the PKI module. • **cpu pki status**: displays the status (including register information) of the PKI module. • **cpu pko config**: displays the configuration

		information (including register information) of the PKO module. • **cpu pko status**: displays the status (including register information) of the PKO module. • **cpu sso config**: displays the configuration information (including register information) of the SSO module. • **cpu sso status**: displays the status (including register information) of the SSO module. • **cpu fpa status**: displays the status (including register information) of the FPA module. • **cpu bp config**: displays the configuration information (including register information) of the BP module. • **cpu bp status**: displays the status (including register information) of the BP module. • **cpu ge nvm-reg** *dev-id reg-addr number*: displays the internal firmware information of the GE module. *dev-id* specifies a device ID, *reg-addr* specifies a register address, and *number* specifies the number of registers. • **cpu csr reg** *addr-high addr-low count*: displays the CPU

		registers by address. *addr-high* specifies the leftmost the 32 most significant bits, *addr-low* specifies the 32 least significant bits, and *count* specifies the number of registers. • **cpu csr reg-name** *module-id name-id*: displays the CPU registers by category. *module-id* specifies a module ID, and *name-id* specifies a register name ID. • **cpu dma key-regs**: displays the DMA key register. • **cpu disp key-regs** *dev_id*: displays the DISP key register. *dev_id* specifies a device ID. • **cpu espi key-regs** *dev_id*: displays the ESPI key register. *dev_id* specifies a device ID. • **cpu axi key-regs** *dev_id*: displays the AXI key register. *dev_id* specifies a device ID. • **cpu net-rtc key-regs** *dev_id*: displays the NET RTC key register. *dev_id* specifies a device ID. • **cpu mii stats** *dev_id*: displays MII statistics. *dev_id* specifies a device ID. • **cpu mii config** *dev_id*: displays the MII configuration. *dev_id* specifies a device ID. • **cpu mii status** *dev_id*: displays the MII status.

		dev_id specifies a device ID. • **cpu avs key-regs** *dev_id*: displays the AVS key register. *dev_id* specifies a device ID. • **cpu iosub key-regs** *dev_id*: displays the IOSUB key register. *dev_id* specifies a device ID. • **cpu sioe key-regs** *dev_id*: displays the SIOE key register. *dev_id* specifies a device ID. • **cpu scheduler key-regs** *dev_id*: displays the SCHEDULER key register. *dev_id* specifies a device ID. • **cpu pmc reg** *dev_id offset len*: displays the PMC register. *dev_id* specifies a device ID, *offset* specifies the address offset, and *len* specifies the data length.

3.1.4 Restoring Factory Settings

Context

You can restore a device to factory settings in order to clear all service configurations and data files.

- The configurations and files cleared in this process include the boot password, unused system software packages and patches, all configuration files, and log files.
- The following files are retained: .cc system software packages and .txt files that record the reset cause.

For details about configuration parameters, see huawei-driver.yang.

NOTICE

This operation will restore the system configuration file to factory settings and clear all service configurations and data files of the device. Exercise caution when performing this operation.

Procedure

- Restore the device to factory settings.
  a. Configure the device to restore factory settings.
     ```
     reboot-with-factory-configuration
     ```
  b. Enter **Y** in the confirmation message that is displayed. After that, the device restarts immediately and restores factory settings.
     ```
     Y
     ```
- Press and hold the RST button for more than 6 seconds to restore the device to factory settings.

----End

3.1.5 Resetting a Device

Context

If a device fails to be upgraded or works improperly, you can reset the device to upgrade its version or restore it to the normal state. After the reset operation is complete, you can check the reset time, cause, and error code. For details about configuration parameters, see huawei-devm-action.

Immediate reset: After the corresponding command is executed, a confirmation message is displayed. Enter **Y** to reset the device immediately.

Delayed reset: After the corresponding command is executed, a confirmation message is displayed. Enter **Y** to confirm the operation. The device will then reset after the specified delay. Delayed reset is useful if you want to specify the next-startup software package for a device, because you can schedule the device to reset when doing so will have minimal impact on services.

Scheduled reset: After the corresponding command is executed, no confirmation message is displayed. You can configure the device to reset at a specified time in the future.

Button-triggered reset: After you press and hold the RST button for less than 6 seconds, the device resets immediately.

The specified next-startup software package is used when a device resets.

If the specified next-startup software package is damaged, the software package used for the previous normal startup is used.

If the software package used for the previous normal startup does not exist on the device, the device searches the storage device for a valid software package.

For details about configuration parameters, see huawei-devm.yang.

NOTICE

Resetting the device will interrupt services. Only reset the device when absolutely necessary.

If you want the current configuration to take effect after the device resets, save the current configuration before resetting the device.

If a device is working improperly, rectify the underlying fault rather than resetting the device to prevent the fault from affecting services.

Procedure

Step 1 Reset the device using one of the following methods.

- Reset the device immediately.
  reboot
- Reset the device after a specified delay.
  schedule-reboot-at-time
  schedule-date YYYY-MM-DD
  schedule-datetime hh:mm
- Specify a delay and reset the device after the specified delay.
  schedule-reboot-delay-time
  delay-time *delay-time*
- Button-triggered reset: After you press and hold the RST button for less than 6 seconds, the device resets immediately.

Step 2 Commit the configuration and perform re-confirmation. Re-confirmation is required only for immediate device reset and delayed device reset.

```
emit
Are you sure you want to continue? [Y(yes)/N(no)]:y
```

Step 3 (Optional) Save the configuration.

```
save-flag true
```

Step 4 Commit the configuration.

```
commit
```

Step 5 Cancel the scheduled reset.

```
undo-schedule-reboot
```

----End

Verifying the Configuration

Run the following commands to check the reset cause. For details, see **Table 3-3**.

```
get-reboot-info
emit
```

Table 3-3 Reset causes

No.	Reset Cause	Description
1	Board button reset.	The Reset button was pressed.

No.	Reset Cause	Description
2	Reset board from command.	The reset command was run.
3	MPU temperature is too high, and reset board.	The MPU temperature was high.
4	CPU failed, and reset board.	The CPU temperature was high.
5	Board restore factory setting, and reset board.	A command was run to restore the factory settings.
6	Board Register	The device was registered.

3.1.6 Disabling the Function of Generating Alarms for Non-Huawei-Certified Optical Modules

Context

A device generates alarms if non-Huawei-certified optical modules are used. If you do not want the device to generate such alarms, disable the function for generating alarms for non-Huawei-certified optical modules.

For details about configuration parameters, see huawei-driver.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the global view.

```
driver global-attribute global
```

Step 3 Disable the function of generating alarms for non-Huawei-certified optical modules.

```
non-certified-optical-status-alarm false
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display this all** command to check optical module information on an interface. Check whether the *false* field is displayed for **non-certified-optical-status-alarm** in the command output.

3.1.7 Configuring a Boot Password

Context

If you do not have a boot password when logging in to a device for the first time, the system prompts you to configure a boot password to ensure device security.

For details about configuration parameters, see huawei-driver.yang.

NOTE

The password is entered in implicit mode. After entering the password, press **Enter**. The password must contain at least eight characters which comprise of the following: lowercase letters, uppercase letters, digits, and special characters. The special characters include `~!@#$%^&*()-_=+\|[]{};:'",<.>/.

Procedure

Step 1 Enter the boot password view.

```
set-boot-password
```

Step 2 Specify a slot ID.

```
slot-id id
```

Step 3 (Optional) Verify the old password. This step is not required if the boot password is configured for the first time.

```
old-password
Enter password:
Confirm password:
```

Step 4 Configure a new password.

```
new-password
Enter password:
Confirm password:
```

Step 5 Commit the configuration.

```
emit
```

----End

3.1.8 Enabling or Disabling the Power Alarm Function for an Optical Module

Context

You can enable or disable the power alarm function for an optical module, so that the power alarms of the optical module can be received or shielded, respectively.

For details about configuration parameters, see huawei-devm.yang and huawei-pic.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the ports view.

```
devm ports
```

Step 3 Select an optical module port.

```
port position interface-number
```

By default, GE0/0/3 and GE0/0/4 are optical module ports.

Step 4 Enter the optical module view.

```
optical-module
```

Step 5 Enable the power alarm function for the optical module.

```
rx-high-power-warn-en true
rx-low-power-warn-en true
tx-high-power-warn-en true
tx-low-power-warn-en true
```

By default, the power alarm function is enabled for the optical module. **rx** indicates the receive direction, and **tx** indicates the transmit direction.

Step 6 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display current-configuration** command to check the configuration of the power alarm function for the optical module.

3.2 PoE Configuration

3.2.1 Overview of PoE

Definition

Power over Ethernet (PoE) — also known as Power over LAN (PoL) or active Ethernet — provides electrical power through the Ethernet.

Purpose

As IP phones, network video surveillance, and wireless Ethernet networks become more widely used, the power supply demand for the Ethernet grows increasingly urgent. In most cases, terminal devices require a DC power supply. However, such devices are often installed outdoors or on high ceilings far from the ground and out of reach of most power sockets. Even if a suitable power socket is available, it can be challenging to install the AC/DC converter required by terminal devices. Large-scale LANs typically involve multiple terminal devices, and providing the uniform power supply and management they require can be difficult. The PoE function addresses this problem.

PoE technology is used on the wired Ethernet and is most widely applied on LANs. PoE allows power to be transmitted to terminal devices through data line pairs or unused line pairs. A PoE-capable device can provide DC power supply for multiple

terminal devices, facilitating centralized power supply management. In addition, supplying power to a terminal device requires only one network cable, which is easy to install. This technology provides power over an Ethernet, spanning a distance of up to 100 m.

With the development of the PoE technology, IEEE 802.3af, IEEE 802.3at, and IEEE 802.3bt power supply standards are introduced one after the other.

- PoE technology (IEEE 802.3af) is used to effectively provide centralized power for terminals such as IP phones, wireless access points (APs), chargers of portable devices, POS machines, cameras, and data collectors.
- PoE+ technology (IEEE 802.3at) provides high-power PoE power supply for terminals such as dual-band access terminals, video phones, and pan-tilt-zoom (PTZ) video surveillance systems.
- PoE++ technology (IEEE 802.3bt) provides higher-power PoE power supply to meet the power supply requirements of higher-power devices.

Benefits

- As there is no need to deploy independent power sockets or AC/DC converters, cabling and installation costs are reduced.
- PoE works with the uninterruptible power supply (UPS) to provide redundant power supply for devices such as IP cameras and IP phones, thereby preventing power outage and improving network reliability.

3.2.2 Understanding PoE

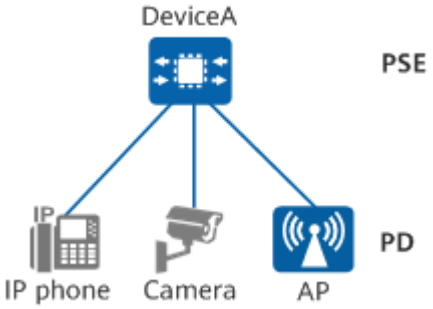
3.2.2.1 PoE Fundamentals

Components in a PoE System

A PoE power supply system consists of a power-sourcing equipment (PSE) and powered devices (PDs), as shown in **Figure 3-1**. The PSE and PDs are connected through network cables.

- PSE: a PoE device that feeds power to a PD through an Ethernet. It provides functions such as detection, analysis, and intelligent power management.
- PD: a device that receives power, such as an AP, portable device charger, POS machine, and camera. PDs are classified into standard and nonstandard PDs depending on whether they conform to IEEE standards.
- PoE power module: provides power to a PoE system. The number of PDs that can be connected to a PSE is limited by the power output of a PoE power module. PoE power modules are classified as either built-in or external depending on whether they are pluggable.

Figure 3-1 Components in a PoE system



PoE Power Supply Process

The PoE power supply process is as follows:

1. PD detection: A PSE periodically transmits a low voltage with limited current through PoE interfaces in order to detect PDs. If the PSE detects a resistance ranging from 19 kΩ to 26.5 kΩ, IEEE-compliant PDs are connected to the PSE. Low voltages for feature resistances of 19 kΩ to 26.5 kΩ are often 2.7 V to 10.1 V, and the detection period is 2 seconds.
2. Power supply capability negotiation: The PSE classifies PDs and negotiates the supplied power.
3. Power supply startup: The PSE starts to supply a low voltage to the PD within less than 15 μs (startup period), and then raises the power to the full 48 V DC voltage eventually.
4. Normal power supply: After the voltage reaches 48 V, the PSE supplies stable and reliable 48 V DC power to the PD. The PoE power output to the PD does not exceed the maximum capacity of the PSE.
5. Disconnection of power supply: The PSE keeps detecting the input current of the PD while supplying power. Upon detecting that the current of the PD falls below the minimum value or increases sharply, the PSE stops supplying power to the PD and re-initiates the PD detection process. This situation occurs when the PD is disconnected from the PSE, encounters a power overload or short circuit, or its power consumption exceeds the power supply capacity of the PSE.

PoE supports UPS, meaning that a PoE device can still supply power to PDs during software version upgrade or restart using the **reboot** command.

PoE Power Supply Mode

According to the IEEE standard, PSEs are used to supply power to PDs and are classified into MidSpan and Endpoint. MidSpan indicates that the PoE module is installed outside the device, whereas Endpoint indicates that the PoE module is integrated into the device. Huawei PSEs have PoE modules integrated inside, belonging to the Endpoint PSE category. Endpoint PSEs are compatible with 10BASE-T, 100BASE-TX, 1000BASE-T, and 2.5GE BASE-T interfaces and therefore can be more widely used than MidSpan PSEs.

Endpoint PSEs have two power supply modes: Alternative A (line pairs 1/2 and 3/6) and Alternative B (line pairs 4/5 and 7/8).

- In Alternative A mode, power is transmitted over data line pairs.

  The PSE provides power to the PD over line pairs 1/2 and 3/6. Pins 1/2 use the negative voltage while pins 3/6 use the positive voltage. 10BASE-T and 100BASE-TX interfaces use line pairs 1/2 and 3/6 to transmit data, while 1000BASE-T interfaces use all four line pairs to transmit data. As DC power and data frequency are independent, power and data can be transmitted over the same pair of lines.
- In Alternative B, power is transmitted over unused line pairs.

  The PSE provides power to the PD over line pairs 4/5 and 7/8. Pins 4/5 use the positive voltage while pins 7/8 use the negative voltage.

In general, standard PDs must support both modes, whereas PSEs can support only one. Huawei PSEs support only Alternative A.

PoE Power Supply Standards Compliance

PoE technical specifications vary according to PoE technologies. You can select the required PoE technology to supply power for PDs according to PD requirements.

Table 3-4 PoE technical specifications

Power Supply Technology	PoE	PoE+	PoE++
Power supply standards compliance	IEEE 802.3af	IEEE 802.3at	IEEE 802.3bt
Power supply distance	100 m	100 m	100 m
Power class	0–3	0–4	0–8
Maximum current	350 mA	600 mA	1730 mA
PSE output voltage	44 V DC–57 V DC	50 V DC–57 V DC	50 V DC–57 V DC (Type 3 PSE) 52 V DC–57 V DC (Type 4 PSE)
PSE output power	≤ 15400 mW	≤ 30000 mW	60000 mW (Type 3 PSE) 90000 mW (Type 4 PSE)
PD input voltage	36 V DC–57 V DC	42.5 V DC–57 V DC	39.9 V DC–57 V DC
Maximum PD power	12950 mW	25500 mW	51000 mW (Type 3 PSE) 71300 mW (Type 4 PSE)

Power Supply Technology	PoE	PoE+	PoE++
Cable requirements	No requirement	CAT-5e or better	CAT-5e or better
Number of power cable pairs	2	2	4

3.2.3 Configuration Precautions for PoE

Licensing Requirements

PoE is not under license control.

Hardware Requirements

Table 3-5 Hardware requirements

Series	Models
S380-L series	S380-L4P1T
S380-S series	S380-S8P2T

Feature Requirements

None

3.2.4 Default Settings for PoE

Table 3-6 describes the default settings for PoE.

Table 3-6 Default settings for PoE

Parameter	Default Setting
Status of the PoE function on a PoE interface	Enabled
Power supply priority of a PoE interface	low

Parameter	Default Setting
Maximum output power of a PoE interface	S380-S8P2T: 124 W S380-L4P1T: 50 W
PoE power alarm threshold	90%

3.2.5 Enabling the PoE Function

Context

Ensure that the PoE function is enabled on an interface before powering on a PD connected to the interface.

Procedure

Step 1 Enter the editing view.

```
edit-config
```

Step 2 Enter the interface view.

```
devm-poe poes poe position 0
ports port interface-name interface-name
```

Step 3 Enable PoE on the interface.

```
power-enable true
```

By default, the PoE function is enabled.

----End

3.2.6 Configuring PoE Power-On and Power-Off Management

3.2.6.1 Powering Off a PoE Interface

Context

A device generally starts providing power to a PD once the PD is connected to the device's PoE interface. In the following cases, you can power off a PoE interface as needed:

- If a PD does not operate constantly (for example, it has a fixed idle time range), you can configure a power-off time range for the PoE interface. In this way, the PD is automatically powered off during the configured power-off time range, reducing power consumption.
- When a PD needs to be temporarily powered off, you can manually power off the PD connected to a PoE interface.

Procedure

Step 1 Enter the system view.

```
edit-config
```

Step 2 Configure a time range.

```
time-range/time-range-instances/timerange-instance name name
absolute-ranges absolute-range start-time start-time end-time end-time
```

Step 3 Enter the interface view.

```
devm-poe poes poe position 0
ports port interface-name interface-name
```

Step 4 Set the power-off time range of the PoE interface.

```
power-off-time-range name
```

By default, no power-off time range is configured for a PoE interface.

----End

Verifying the Configuration

Run the **devm-poe poes poe position 0** and **display ports port interface-name** *interface-name* **all** commands in sequence to check the power supply status of a PoE interface based on the **power-enable** field.

3.2.7 Maintaining PoE

Checking PoE Information

Run the **display devm-poe/poes all** command to check information about devices that support the PoE function.

Resetting PSE Chips

Run the **poe cmdinfo cmd "poe reset chips"** command in the diagnostic view to reset PSE chips.

Querying Key Register Information

Run the **poe cmdinfo cmd "display poe chips"** command in the diagnostic view to query key register information.

3.3 Information Management Configuration

3.3.1 Overview of Information Management

Definition

The information management (IM) module operates as the information hub of a device. IM receives logs and debugging messages, and other device-generated information to implement unified management and flexible output.

Purpose

Immediate and accurate information about device operation facilitates troubleshooting if an exception or fault occurs on a device. During device operation, IM records relevant information generated by each module, including logs and debugging messages. You can configure IM to classify or filter device-generated information by information type or severity. This enables you to easily monitor the device status and locate faults.

3.3.2 Understanding IM

3.3.2.1 Information Classification

Information generated by the device is classified into logs and debugging information. For details, see **Table 3-7**.

Table 3-7 Information classification

Information Type	Description
Log	As defined in ITU-T, logs record the events and unexpected activities of managed objects, providing information that enables you to perform troubleshooting, obtain device running status, manage system security, and maintain the system. Some logs are intended for use by only technical support personnel during troubleshooting. Consequently, users are not notified when such logs are generated. For this reason, system logs are classified as user logs, diagnostic logs, or O&M logs. ● User logs: When a device is running, the log module in the host software records various running information and generates log information. Users can view the generated compressed files and file contents, which are called user logs. User logs related to security are called security logs, which are mainly comprised of account management, protocol, attack defense, and status logs. – Account management security logs: record account operations, including user accounts, IP addresses, login and logout time, and operating time, content, and results. – Protocol security logs: record information related to protocol security, including information about insecure interaction modes or algorithms used by protocols. – Attack defense security logs: record attack events, which include the time an event occurred, attack locations and sources, various IP attack types, and attack impacts. – Status security logs: record software and hardware abnormalities, real-time key performance indicators, real-time indicators of key bandwidths, entries and storage resources, and detected abnormal processes. ● Diagnostic log: When a device is running, the log module in the host software records the process information generated during the running of each service for fault locating and analysis. The recorded information is presented in diagnostic logs. ● O&M logs: When a device is running, the log module in the host software records the data generated during the running of each service and generates log information. Users can view the generated compressed files and file contents. **NOTE** The information recorded in diagnostic and O&M logs is used only for troubleshooting and does not contain any sensitive information. To obtain such logs, contact Huawei technical support.

Information Type	Description
Debugging message	Debugging messages record the internal running status of a device, such as information about service transmissions. These messages help you obtain the running status of the device. A device generates debugging messages only after debugging is enabled for a specific service on the device.

3.3.2.2 Information Severity

If a device generates large amounts of information, it can be difficult to identify the information you require. Setting information severities allows you to rapidly identify such information.

Information is classified into eight severities, with a smaller value indicating a higher severity. **Table 3-8** describes information severity.

Table 3-8 Description of information severity

Value	Severity	Description
0	Emergency	A fault prevents the device from running normally, requiring a restart. For example, the device restarts because of a program exception or a fault relating to memory usage.
1	Alert	A fault needs to be rectified immediately. For example, memory usage of the system reaches the upper limit.
2	Critical	A fault needs to be analyzed and processed. For example, the device temperature falls below the lower threshold or BFD detects that a device is unreachable.
3	Error	An irregular operation is performed or exceptions occur during service processing. The fault does not affect services but needs to be analyzed. For example, users enter incorrect commands or passwords, or error protocol packets are detected.
4	Warning	Some events or operations may affect operation of the device or lead to service processing faults, and therefore require attention. For example, a routing process is disabled, BFD detects packet loss, or error protocol packets are detected.
5	Notice	A key operation is performed to ensure that the device continues to run properly. For example, the **shutdown** command is used on an interface, a neighbor is discovered, or the protocol status is changed.
6	Informational	A normal operation is performed. For example, a **display** command is run.

Value	Severity	Description
7	Debugging	The running status of a device is recorded when the device is running properly.

After you set the severity value, the device outputs only the information with a severity value less than or equal to the set value. For example, if you set the severity value to 6, the device outputs only information with severity values 0 to 6.

3.3.2.3 Information Format

The information format is as follows:

<Int_16>TIMESTAMP HOSTNAME Process[ID] %%ddModule/Severity/Brief:CID=XXX; Description

Table 3-9 describes each field of the information format.

Table 3-9 Information format

Field	Description	Remarks
<Int_16>	Leading character	This character is added to information to be sent to the log host. It is not added to information saved locally.

Field	Description	Remarks
TIMESTAMP	Timestamp	Date and time when information was output. Five timestamp formats are available: ● boot: expressed in relative time (in this case, the time elapsed since system startup). The format is *xxxxxx.yyyyyy*, where *xxxxxx* is the leftmost 32 bits of the milliseconds elapsed since system startup, and *yyyyyy* is the rightmost 32 bits of the milliseconds elapsed since system startup. ● date: expressed as the current system date and time. The format is mm dd yyyy hh:mm:ss. ● short-date: indicates that the timestamp uses the short date format, similar to the date format but does not display the year. ● format-date: expressed in the format of YYYY-MM-DD hh:mm:ss. ● None: no timestamp in output information. The timestamp and hostname are separated by a space.
HOSTNAME	Hostname	The default value is HUAWEI.
Process[ID]	Process name [Process index]	Information about the process that generates the information.
%%	Huawei identifier	The information is output by a Huawei device.
Module	Module name	Name of a module that generates the information.
Severity	Information severity	Information severity.
Brief	Brief description	Brief description of the information.
CID=XXX	Internal component ID	Internal component to which the information belongs. XXX indicates the ID of an internal component.
Description	Detailed description	Detailed information about the message.

3.3.2.4 Information Suppression

Some service modules, such as ARP and VRRP, generate a large number of duplicate logs in quick succession in scenarios such as ARP attacks and route or link faults. This consumes device storage space and CPU resources. To reduce the number of such logs, you can configure information suppression.

By default, a device outputs a log immediately after receiving it. Before sending logs generated by service modules to the IM module, the system checks whether they are duplicates. If two consecutive logs contain different IDs or parameters, the system considers them unique and immediately sends both to the IM module. Conversely, the system considers them duplicate if they contain the same ID and parameters. In this case, only the number of duplicate logs is recorded until other logs are received.

For example, a module may generate logs in the following order: A(T1) A(T2) A(T2) B(T3) B(T4) B(T4) C1(T5) C2(T6) A(T7) B(T8) B(T8) B(T8) B(T9) A(T9) B(T10) C3(T11) A(T11) A(T12) A(T12) A(T13) A(T14) A(T15) A(T16) A(T17) A(T18) B(T18). Cn represents logs that do not repeat (C1 and C2 are two different logs), and Tn represents the sequence number. The log information output by the IM module is as follows:

```
T1:A
T3(1): last message repeated 2 times
T3:B
T5: last message repeated 2 times
T5:C1
T6:C2
T8:B
T9(1): last message repeated 3 times
T9:A
T10:B
T11:C3
T11:A
T13(2): last message repeated 3 times
T18(2): last message repeated 5 times
T18:B
```

According to the preceding log information, consecutive duplicate logs are triggered in either of the following scenarios:

- If the next log is not a duplicate, the module will output statistics about consecutive duplicate logs until the last duplicate log is generated. For example, **(1)**.
- If consecutive duplicate logs are generated again after a specific period of time, the module will update the statistics. For example, **(2)**.

The device resets the counter each time it outputs the statistics. In this case, log A is repeated nine times (1 + 3 + 5) in the period from T11 to T18.

The logs output by the device are sorted in the same order of log generation, facilitating subsequent review.

NOTE

Information suppression refers to the suppression about duplicate logs.

3.3.2.5 Information Output

Information generated by the device can be output to terminals and log files. This section describes information channels, output directions, output files, and typical application scenarios.

Information Output File

Information can be saved as log files on a device. **Table 3-10** describes the log files.

Table 3-10 Log files

Log File	Description
security.log	The security log file records security-related logs and is saved in log format.
diag.log	The diagnostic log file records service debugging information and is saved in log format.
event.log	Alarm logs and event logs record information about alarms and events generated by services and are saved in log format.
oper.log	The operation log records user operation information and is saved in log format.

Application Scenarios of Information Output

This section describes the typical application scenarios of IM.

- Outputting logs to a log file

  As shown in **Figure 3-2**, the IM module outputs logs to a log file, and maintenance personnel uploads the log file to the FTP server. By querying logs, maintenance personnel can learn about the running status of the device and locate faults.

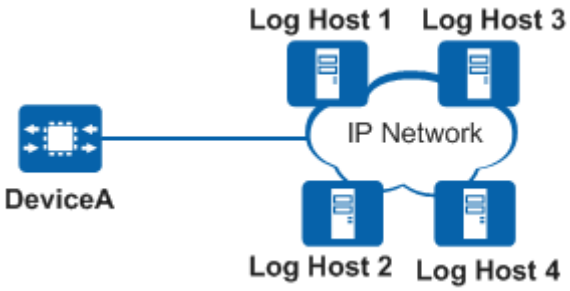
  Figure 3-2 Outputting logs to a log file

- Outputting logs to a log host

  As shown in **Figure 3-3**, the IM module sends logs to different log hosts. The network administrator can query logs to learn about the device running status and to rapidly locate faults.

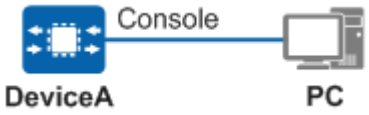
Figure 3-3 Outputting logs to a log host



- Outputting debugging messages to the console

  As shown in **Figure 3-4**, the IM module sends debugging messages to the console, and the maintenance personnel debugs the device based on the debugging messages.

Figure 3-4 Outputting debugging messages to the console



3.3.3 Configuration Precautions for Information management

Licensing Requirements

Information management is not under license control.

Hardware Requirements

Table 3-11 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.3.4 Default Settings for IM

Table 3-12 describes the default settings of common IM parameters.

Table 3-12 Default settings for IM

Parameter	Default Setting
Information management	Enabled
Number of displayed logs	200
Size of a single log file	2M
Maximum number of log files that can be stored for each type of log	5

3.3.5 Saving Logs to a Log File

Context

After a device is configured to output logs to a log file, the logs are saved in this log file on the device. If a fault occurs on a device, you can export and analyze the log file to locate the fault.

Procedure

Step 1 Save information in the log buffer to a log file in the root view.

```
save-logfile log-type { diaglog | log }
```

Logs are cached in the system memory before being written into a log file. When the buffer is full or if the log saving timer expires, the device immediately writes the cached logs into the log file. If the log buffer is not full or the timer does not expire, run this command to manually write logs in the memory into the log file.

----End

Follow-up Procedure

After a log file is generated, you can run the **display syslog/logfiles/logfile[logfile-type=***value***]** command in the root view to view the content of the log file.

To clear a log file, run the **delete-file file-name** *value* command in the root view.

3.3.6 Configuring the Log Aging Period

Context

You can set the aging period of logs to periodically clear log files to save system memory space.

Procedure

Step 1 Enter the editing view.

```
edit-config
```


Step 2 Enter the log aging view.

```
syslog log-storage-time
```

Step 3 Configure the log aging period.

```
storage-time value
```

Step 4 Commit the configuration.

```
commit
```

----End

3.3.7 Configuring Log Reporting Suppression

Context

You can configure log reporting suppression to suppress the reporting of specified logs. The suppression rules do not take effect for security logs and operation logs.

Procedure

Step 1 Enter the editing view.

```
edit-config
```

Step 2 Specify a log name.

```
syslog log-switch-list log-switch feature-name feature-name log-name log-name
```

Step 3 Enable log reporting suppression.

```
suppress true
```

Step 4 Commit the configuration.

```
commit
```

----End

3.3.8 Maintaining IM

Monitoring IM

During routine maintenance, you can run the following commands in the root view to check the desired device information.

Table 3-13 Monitoring IM

Operation	Command
View the supported logs.	**display syslog/log-switch-list/ all**
View the content of a specified log file.	**display syslog/logfiles/logfile[logfile-type=** *value* **] all**
Collect logs.	**collect-log** **http-url** *http URL* **ssl-policy-name transfer** **emit**

3.4 LLDP Configuration

The Link Layer Discovery Protocol (LLDP) is a Layer 2 discovery protocol defined in IEEE 802.1ab. Deploying LLDP improves network management system (NMS) capabilities. LLDP supplies the NMS with detailed information about network topology and changes to topology, and it detects inappropriate configurations existing on the network. The information provided by LLDP helps administrators monitor network status in real time to keep the network secure and stable.

3.4.1 Overview of LLDP

Definition

The Link Layer Discovery Protocol (LLDP), a Layer 2 discovery protocol defined in IEEE 802.1ab, provides a standard link-layer discovery method that encapsulates information about the capabilities, management address, device ID, and interface ID of a local device into LLDP packets and sends the packets to neighboring devices. These neighboring devices save the information received in a standard management information base (MIB) to help the network management system (NMS) query and determine the link communication status.

Purpose

Diversified network devices are deployed on a network, and configurations of these devices are complicated. Therefore, NMSs must be able to meet increasing requirements for network management capabilities, such as the capability to automatically obtain the topology status of connected devices and the capability to detect configuration conflicts between devices. A majority of NMSs use an automated discovery function to trace changes in the network topology, but most can only analyze the network layer topology. Network layer topology information notifies you of basic events, such as the addition or deletion of devices, but gives you no information about the interfaces to connect a device to other devices. The NMSs can identify neither the device location nor the network operation mode.

LLDP is developed to resolve these problems. LLDP can identify interfaces on a network device and provide detailed information about connections between devices. LLDP can also display information about paths between clients, switches, routers, application servers, and network servers, which helps you efficiently locate network faults.

Benefits

Deploying LLDP improves NMS capabilities. LLDP supplies the NMS with detailed information about network topology and topology changes, and it detects inappropriate configurations existing on the network. The information provided by LLDP helps administrators monitor network status in real time to keep the network secure and stable.

3.4.2 Understanding LLDP

3.4.2.1 Basic LLDP Concepts

LLDP Packets

LLDP packets are Ethernet packets that have LLDP data units (LLDPDUs) encapsulated. LLDP packets support two encapsulation modes: Ethernet II and Subnetwork Access Protocol (SNAP). Currently, the Ethernet II encapsulation mode is supported. **Figure 3-5** shows the format of an Ethernet II LLDP packet.

Figure 3-5 LLDP packet format

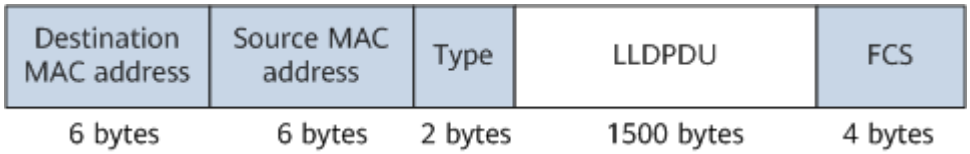


Table 3-14 describes the fields in an LLDP packet.

Table 3-14 Fields in an LLDP packet

Field	Description
Destination MAC address	Destination MAC address, which is a fixed multicast MAC address 0x0180-C200-000E
Source MAC address	Source MAC address, which is the interface MAC address or system MAC address (Use the interface MAC address if there is one; otherwise, use the system MAC address)
Type	Packet type, which is fixed at 0x88CC
LLDPDU	Main body of an LLDP packet
FCS	Frame check sequence

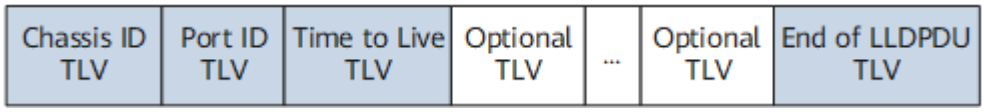
LLDPDU

An LLDPDU is a data unit encapsulated in the data field in an LLDP packet.

A device encapsulates local device information in type-length-value (TLV) format and combines several TLVs in an LLDPDU for transmission. You can combine various TLVs to form an LLDPDU as required. TLVs allow a device to advertise its own status and learn the status of neighboring devices.

Figure 3-6 shows the LLDPDU format.

Figure 3-6 LLDPDU format



Each LLDPDU carries a maximum of 28 types of TLVs, and each LLDPDU starts with the Chassis ID TLV, Port ID TLV, and Time to Live TLV, and ends with the End of LLDPDU TLV. The Chassis ID TLV, Port ID TLV, and Time to Live TLV are mandatory TLVs, and other TLVs are selected as needed.

TLV

A TLV is the smallest unit of an LLDPDU. It provides type, length, and other information for a device object. Each TLV represents a type of device information. For example, a device ID is carried in the Chassis ID TLV, an interface ID in the Port ID TLV, and a network management address in the Management Address TLV.

LLDPDUs can carry basic TLVs and IEEE 802.3-defined TLVs.

- Basic TLVs: Basic TLVs are the basis for network device management.

Table 3-15 Basic TLVs

TLV Name	TLV Type Value	Description	Mandatory
End of LLDPDU TLV	0	End of an LLDPDU	Yes
Chassis ID TLV	1	System MAC address of the transmit device	Yes
Port ID TLV	2	ID of an interface on a transmit device	Yes
Time To Live TLV	3	Time to live (TTL) of the local device information stored on a neighbor device	Yes
Port Description TLV	4	Description of an Ethernet interface	No
System Name TLV	5	Device name	No
System Description TLV	6	System description	No
System Capabilities TLV	7	Main functions of the system and the functions that have been enabled	No
Management Address TLV	8	Management address	No

TLV Name	TLV Type Value	Description	Mandatory
Reserved	9–126	Reserved for special use	No
Organizationally Specific TLVs	127	TLVs defined by organizations	No

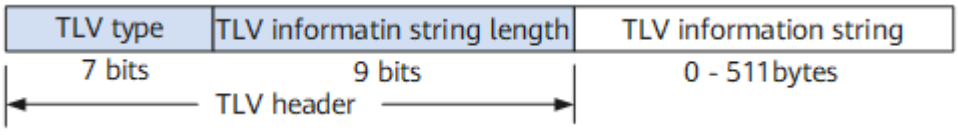
- IEEE 802.3-defined TLVs: IEEE 802.3-defined TLVs are used to enhance network device management. Use these TLVs as needed.

Table 3-16 IEEE 802.3-defined TLVs

TLV Name	TLV Type Value	Description
Reserved	0	Reserved for special use
MAC/PHY Configuration/Status TLV	1	Rate auto-sensing, current duplex and rate settings, and auto-sensing status
Power Via MDI TLV	2	Power supply capability of an interface, that is, whether an interface supplies or requires power
Link Aggregation TLV	3	Link aggregation status
Maximum Frame Size TLV	4	Maximum frame length supported by an interface, that is, the maximum transmission unit (MTU) of an interface
Reserved	5–255	Reserved for special use

Figure 3-7 shows the TLV format.

Figure 3-7 TLV format



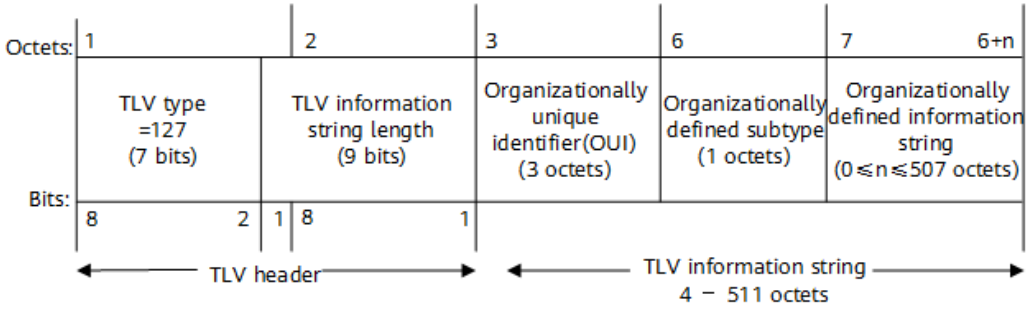
The TLV contains the following fields:

- TLV type: a 7–bit field. Each value uniquely identifies a TLV type. For example, value 0 indicates the End of LLDPDU TLV, and value 1 indicates the Chassis ID TLV.
- TLV information string length: a 9–bit field indicating the length of a TLV string.
- TLV information string: a string that contains TLV information. This field contains a maximum of 511 bytes.

When the value of TLV type is 127, it indicates that the TLV is an organization-defined TLV. In this case, the TLV structure is the following shown in **Figure 3-8**.

The Organizationally unique identifier (OUI) field identifies the organization that defines the TLV.

Figure 3-8 TLV structure with TLV type being 127



LLDP Management Addresses

LLDP management addresses are used by the NMS to identify devices and implement network management. Management addresses uniquely identify network devices, facilitating network topology layout and network management.

Each management address is encapsulated in the Management Address TLV in an LLDP packet and then advertised.

The management address can be the system MAC address of the device or the IP address of the management interface. Either of them can be used, with the IP address of the management interface preferred.

3.4.2.2 LLDP Fundamentals

Implementation

LLDP must be used together with Management Information Bases (MIBs). LLDP requires that each device interface be provided with four MIBs. An LLDP local system MIB that stores status information of a local device and an LLDP remote system MIB that stores status information of a neighboring device are most important. Status information includes the device ID, interface ID, system name, system description, interface description, device capability, and network management address.

LLDP requires that each interface of a device be provided with an LLDP agent to manage LLDP operations on the device. The LLDP agent performs the following functions:

- Maintains information in the LLDP local system MIB.
- Sends LLDP packets to notify neighboring devices of the status of the local device.
- Identifies and processes LLDP packets sent by neighboring devices and maintains information in the LLDP remote system MIB.
- Sends LLDP alarms to the NMS when detecting changes in information stored in the LLDP local or remote system MIB.

Figure 3-9 LLDP schematic diagram

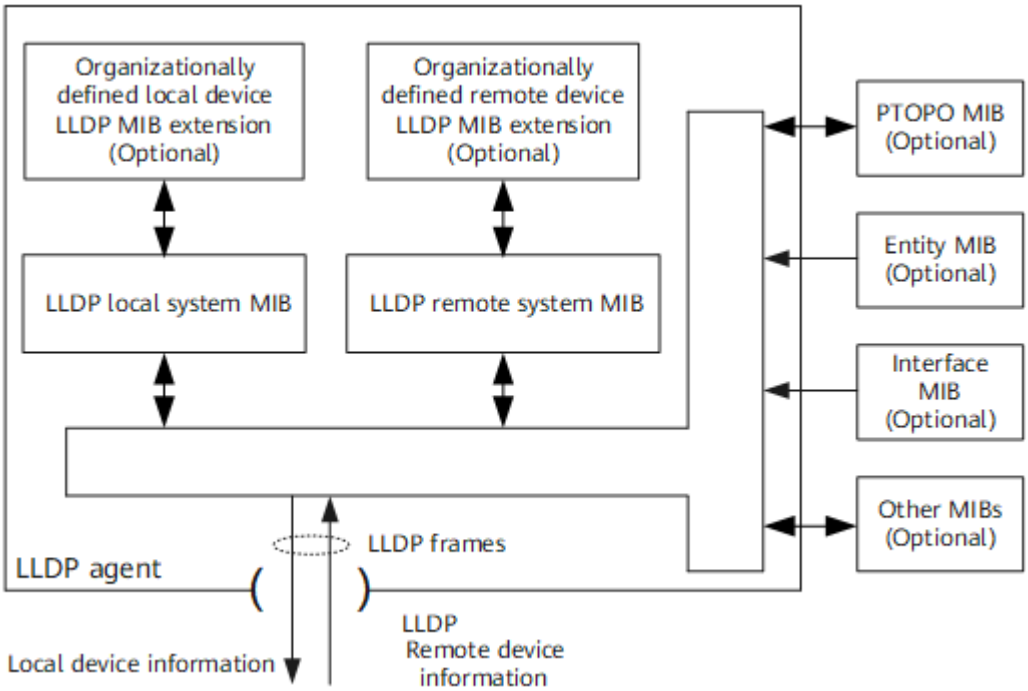


Figure 3-9 shows the LLDP implementation process:

- The LLDP module maintains the LLDP local system MIB by exchanging information with the PTOPO MIB, Entity MIB, Interface MIB, and Other MIBs of the device.
- An LLDP agent sends LLDP packets carrying local device information to neighboring devices that are directly connected to the local device.
- An LLDP agent updates the LLDP remote system MIB after receiving LLDP packets from neighboring devices.

The NMS collects and analyzes topology information stored in LLDP local and remote system MIBs on all managed devices and determines the network topology. The information helps rapidly detect and rectify network faults.

Working Mechanism

LLDP working modes

LLDP works in one of the following modes:

- Tx mode: enables a device only to send LLDP packets.
- Rx mode: enables a device only to receive LLDP packets.
- Tx/Rx mode: enables a device to send and receive LLDP packets.
- Disabled mode: disables a device from sending or receiving LLDP packets.

The default working mode is Tx/Rx.

NOTE

When the LLDP working mode changes on an interface, the interface initializes the LLDP state machine. To avoid repeated initializations caused by frequent working mode changes on an interface, an initialization delay can be configured for the interface. When the working mode of the interface changes, the interface initializes the LLDP state machine after the configured delay elapses.

Mechanism for sending LLDP packets

- After LLDP is enabled on a device, the device periodically sends LLDP packets to neighboring devices. If the configuration is changed on the local device, the device immediately sends LLDP packets to notify neighboring devices of the changes. If information changes frequently, a delay for an interface to send LLDP packets can be set. After an interface sends an LLDP packet, the interface does not send another LLDP packet until the configured delay time elapses. This reduces the number of LLDP packets to be sent.
- The fast sending mechanism allows a device to override a pre-configured delay time and quickly advertise local information to other devices in the following situations:
  - The device receives an LLDP packet sent by a transmit device, whereas the device has no information about the transmit device.
  - LLDP is enabled on the device that previously has LLDP disabled. An interface on the device goes up.
  - The fast sending mechanism shortens the interval at which LLDP packets are sent to 1 second.

  After a specified number of LLDP packets are sent, the pre-configured delay time is restored.

Mechanism for receiving LLDP packets

A device verifies TLVs carried in LLDP packets it receives. If the TLVs are valid, the device saves information about neighboring devices and sets the TTL value carried in the LLDPDU so that the information ages after the TTL expires. If the TTL value carried in a received LLDPDU is 0, the device immediately ages information about neighboring devices.

3.4.3 Configuration Precautions for LLDP

Licensing Requirements

LLDP is not under license control.

Hardware Requirements

Table 3-17 Hardware requirements

Series	Model
S380-H series	S380-H8T3S
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Limitations

None

3.4.4 Configuring Basic LLDP Functions

After devices are configured with LLDP, the NMS can obtain network topology information, and information about the capabilities, management address, device ID, and interface ID of each device.

3.4.4.1 Enabling LLDP

When LLDP is enabled on a device, the device sends LLDP packets carrying its status information to its LLDP-capable neighbors and obtains their status information by receiving LLDP packets from them.

Context

LLDP can be enabled globally or on an interface. The relationships are as follows:

- LLDP is enabled on all interfaces after LLDP is enabled globally.
- LLDP is disabled on all interfaces after LLDP is disabled globally.
- An interface can send and receive LLDP packets only when LLDP is enabled globally and on the interface.
- The command to enable or disable LLDP on an interface is invalid when LLDP is disabled globally.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enable LLDP globally.

```
lldp
```

Step 3 Commit the configuration.

```
commit
```

Step 4 Return to the editing view.


```
quit
```

Step 5 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 6 Configure an LLDP working mode for the interface.

```
lldp session
admin-status { disabled | tx-only | rx-only | tx-rx }
```

Step 7 Commit the configuration.

```
commit
```

----End

3.4.4.2 (Optional) Configuring Types of TLVs Allowed to Be Advertised by LLDP

During the process of exchanging LLDP packets between devices, the LLDPDU encapsulated in an LLDP packet carries different TLVs as needed. A device sends its status information and receives neighbor status information based on these different TLVs.

Context

LLDPDUs can carry basic TLVs and TLVs defined by IEEE 802.3.

- Basic TLVs implement basic LLDP functions. Basic TLVs include four mandatory TLVs, which must be encapsulated into LLDPDUs to be advertised. Basic TLVs also include five optional TLVs: **management-address**, **port-description**, **system-capability**, **system-description**, and **system-name**.
- TLVs defined by IEEE 802.3 are optional TLVs used to enhance the LLDP function. You can determine whether to encapsulate these TLVs into LLDPDUs to be advertised based on their functions and your actual requirements.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Configure the types of TLVs allowed to be advertised by LLDP.

```
lldp session tlv-enable
{ link-aggregatoin | mac-physic | management-address | port-description | system-capability | system-description | system-name } { true | false }
```

Step 4 Commit the configuration.

```
commit
```

----End

3.4.4.3 (Optional) Optimizing LLDP Performance

This section describes how to adjust LLDP parameters based on the load of a network to reduce the consumption of system resources and optimize the LLDP performance.

Context

LLDP parameters include: interval for sending LLDP packets, delay for sending LLDP packets, time multiplier of device information held in neighbors, delay for initializing LLDP on interfaces, and number of LLDP packets being sent in quick succession to neighbors. Values of these parameters should be appropriate. You can adjust these parameters based on the load of a network. **Table 3-18** describes the usage scenarios of LLDP parameters.

Table 3-18 LLDP parameters

Parameter Name	Parameter Description	Value Description
message-transmission interval	Sets the interval for sending LLDP packets to adjust the frequency of network topology discovery.	• The longer the interval, the lower the frequency of LLDP packets being exchanged. This saves system resources. However, if the interval for sending LLDP packets is too long, the device cannot notify neighbors of its status in a timely manner, reducing network topology discovery efficiency. • The shorter the interval, the higher the frequency of the local status information being sent to neighbors. This ensures prompt network topology discovery. However, if the interval is too short, LLDP packets are exchanged too frequently, increasing the system load and wasting resources.

Parameter Name	Parameter Description	Value Description
message-transmission delay	Sets the delay for sending LLDP packets to avoid network flapping of neighbors caused by LLDP packets being frequently sent to neighbors.	When the status of a device changes frequently: • The longer the delay, the lower the frequency of the local status information being sent to neighbors. This saves system resources. However, if the delay for sending LLDP packets is too long, the device cannot notify neighbors of its status in a timely manner, reducing network topology discovery efficiency. • The shorter the delay, the higher the frequency of the local status information being sent to neighbors. This ensures prompt network topology discovery. However, if the delay is too short, LLDP packets are exchanged too frequently, increasing the system load and wasting resources.
message-transmission hold-multiplier	Sets the time multiplier of device information held in neighbors to calculate the valid time of LLDP packets being sent to neighbors. The time of device information held in neighbors can be adjusted by setting this parameter.	• The higher the time multiplier, the lower the frequency of network topology changes of neighbors. However, if the time of device information held in neighbors is too long, the device cannot notify neighbors of its status in a timely manner, reducing network topology discovery efficiency. • The lower the time multiplier, the higher the frequency of network topology changes of neighbors. This ensures prompt network topology discovery. However, if the time multiplier is too low, neighbors refresh local status information frequently, increasing the system load and wasting resources.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Configure a delay for sending LLDP packets. It is advised that you use the default delay for sending LLDP packets.

```
lldp global-attribute
message-transmission delay delay
```

Step 3 Configure a time multiplier of device information held in neighbors.

```
message-transmission hold-multiplier hold-multiplier
```

Step 4 Configure an interval for sending LLDP packets.

```
message-transmission-interval interval
```

NOTE

The interval and delay for sending LLDP packets affect each other. When you decrease the value of *interval*, ensure that the target value of *interval* is greater than or equal to four times the value of *delay*. Otherwise, the value of *delay* must be adjusted to be less than or equal to a quarter of the target value of *interval*.

The time of device information held in neighbors is Min(65535, (*interval* x *hold-multiplier*)). That is, the smaller value between 65535 and (*interval* x *hold-multiplier*) is used.

Step 5 Commit the configuration.

```
commit
```

----End

3.4.4.4 Verifying the Configuration

After configuring basic LLDP functions, verify the configuration.

Prerequisites

All configurations of basic LLDP functions are complete.

Procedure

- Run the **display lldp/local-info** command to check the global local LLDP status.
- Run the **display ifm/interfaces/interface[name="***interface-name* **"]/lldp/session/local-info** command to check the local LLDP status of all interfaces or a specified interface.
- Run the **display ifm/interfaces/interface[name="***interface-name***"]/lldp/session/neighbors/neighbor[index=***neighbor-index***]** command to check the LLDP status of neighbors connected to specified interfaces. If *neighbor-index* is not specified, the LLDP status of all neighbors of a specified interface is displayed.
- Run the **display ifm/interfaces/interface[name="***interface-name***"]/lldp/session/tlv-enable** command to check information about optional TLVs that are allowed to be advertised by all interfaces or a specified interface.

----End

3.4.5 Maintaining LLDP

Maintaining LLDP includes clearing information about LLDP statistics and monitoring LLDP status.

3.4.5.1 Checking LLDP Status

Context

In routine maintenance, you can run the following commands to check LLDP status.

Procedure

- Run the **display ifm/interfaces/interface[name="***interface-name***"]/lldp/session/statistics** command to check statistics about LLDP packets transmitted by all interfaces or a specified interface.
- Run the **diagnose lldp debugging enabled** [**true** | **false**] { **if-name** *if-name* } { **level error** | **event** | **fsm** | **packet** } command to enable debugging for specified content.
- Run the **diagnose lldp error-packet if-name** *if-name* command to check the error packets received by a specified interface.
- Run the **diagnose lldp trouble-shooting** command to view LLDP debugging information.

----End

3.4.6 Configuration Examples for LLDP

The configuration examples provide networking requirements, configuration roadmap, data preparation, and configuration procedures

3.4.6.1 Example for Configuring Basic LLDP Functions

This section provides an example for configuring basic LLDP functions to help the NMS obtain information about network topology.

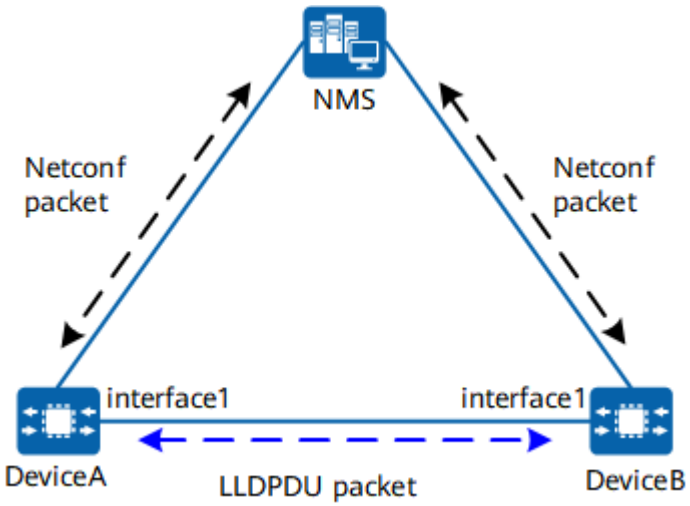
Networking Requirements

As shown in **Figure 3-10**, there is a reachable link between DeviceA and DeviceB. DeviceA and DeviceB have reachable routes to the NMS. Before the LLDP function is configured, DeviceA cannot obtain status information about DeviceB, and the NMS cannot obtain topology information between DeviceA and DeviceB by exchanging NETCONF packets with them. After the LLDP function is configured, devices can obtain status information about neighbors by exchanging LLDP packets between them.

NOTE

In this example, interface1 represents GE0/0/1.

Figure 3-10 Networking diagram for configuring basic LLDP functions



Configuration Roadmap

The configuration roadmap is as follows:

1. Enable the LLDP function for DeviceA and DeviceB globally.
2. Configure LLDP parameters for DeviceA and DeviceB to optimize the LLDP performance.

Data Preparation

To complete the configuration, ensure link reachability between DeviceA and DeviceB, and route reachability between DeviceA, DeviceB, and the NMS.

Procedure

Step 1 Enable the LLDP function for DeviceA and DeviceB globally.

Configure LLDP globally on DeviceA.

```
[root@HUAWEI]
MDCLI> edit-config
[(gl)root@HUAWEI]
MDCLI> lldp
[*(gl)root@HUAWEI]/lldp
MDCLI> commit
[(gl)root@HUAWEI]/lldp
MDCLI>
```

Configure LLDP globally on DeviceB. For details, see the configuration of DeviceA.

Step 2 Configure LLDP parameters for DeviceA and DeviceB. These parameters include the interval and delay for sending LLDP packets.

Configure an interval and a delay for DeviceA to send LLDP packets.

```
[root@HUAWEI]
MDCLI> edit-config
[(gl)root@HUAWEI]
MDCLI> lldp global-attribute
[*(gl)root@HUAWEI]/lldp/global-attribute
```



```
MDCLI> message-transmission-delay 9
[*(gl)root@HUAWEI]/lldp/global-attribute
MDCLI> message-transmission-interval 60
[*(gl)root@HUAWEI]/lldp/global-attribute
MDCLI> message-transmission-hold-multiplier 10
[*(gl)root@HUAWEI]/lldp/global-attribute
MDCLI> commit
[(gl)root@HUAWEI]/lldp/global-attribute
MDCLI>
```

Configure an interval and a delay for DeviceB to send LLDP packets. For details, see the configuration of DeviceA.

Step 3 Verify the configuration.

Check the configuration of DeviceA.

- Check the global local configuration of DeviceA.

```
[root@HUAWEI]
MDCLI> display lldp/global-attribute/
{
  "message-transmission-interval": 60,
  "message-transmission-delay": 9,
  "message-transmission-hold-multiplier": 10
}
```

- Check the global LLDP local information of DeviceA.

```
[root@HUAWEI]
MDCLI> display lldp/local-info/
{
  "chassis-id-sub-type": "mac-address",
  "chassis-id": "00e0-fc12-3456",
  "system-name": "Huawei",
  "system-description": "Huawei YunShan OS\u000D\u000AVersion 1.22.0.1 (S380
V600R022C10)\u000D\u000ACopyright (C) 2021-2022 Huawei Technologies Co., Ltd.\u000D
\u000AHUAWEI S380",
  "up-time": "2022-01-01T21:54:19+00:00",
  "system-capabilities-supported": [
    "bridge",
    "router"
  ],
  "system-capabilities-enabled": [
    "bridge",
    "router"
  ],
  "management-addresss": {
    "management-address": [
      {
        "type": "ipv4",
        "value": "10.1.1.1",
        "length": 5,
        "if-sub-type": "if-index",
        "if-id": 10001,
        "oid": "1.3.6.1.4.1.2011.5.25.41.1.2.1.1.1"
      }
    ]
  }
}
```

- Check the LLDP information about neighbors connected to DeviceA.

```
[root@HUAWEI]
MDCLI> display ifm/interfaces/interface[name="GE0/0/1"]/lldp/session/neighbors/
{
  "neighbor": [
    {
      "index": 1,
      "chassis-id-sub-type": "mac-address",
      "chassis-id": "00e0-fc12-3457",
      "port-id-sub-type": "mac-address",
```



```
    "port-id": "00e0-fc12-3412",
    "system-capabilities-enabled": "no",
    "system-capabilities-supported": "no",
    "expired-time": 3601,
    "port-vlan-id": 0,
    "auto-negotiation-supported": "yes",
    "auto-negotiation-enabled": "yes",
    "auto-negotiation-capability": "0001(Other)",
    "oper-mau-type": "unknown",
    "link-aggregation-supported": "no",
    "link-aggregation-enabled": "no",
    "aggregation-port-id": 0,
    "maximum-frame-size": 0,
    "discovered-time": "2022-01-01T22:01:18+00:00",
    "power": {
      "port-class": "pd",
      "pse-support": "no",
      "pse-state": "no",
      "pse-pairs-control-ability": "not-controlled",
      "pse-pairs": "unknown",
      "classification": "unknown"
    },
    "med-tlv": {
      "capability": {
        "capabilities": [
          "capabilities"
        ],
        "device-type": "endpoint-class-1"
      }
    }
  }
 ]
}
```


Check the configuration of DeviceB. For details, see how the configuration is checked on DeviceA.

----End

3.5 System Time Configuration

3.5.1 Overview of System Time

Definition

System time refers to the current time on a device and is an important parameter for device running.

Purpose

System time recorded in device logs and alarms enables administrators to identify when specific events occurred. In addition, a correctly configured system time ensures the accuracy and consistency of device collaboration when multiple devices interwork on a network.

3.5.2 Configuration Precautions for System time

Licensing Requirements

System time is not under license control.

Hardware Requirements

Table 3-19 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.5.3 Configuring the System Time

Context

System time is the current time that a device keeps track of and is recorded in timestamps of sent packets. Users in different regions can configure the system clock according to their own country's or region's regulations.

System time = Coordinated Universal Time (UTC) + Time zone offset

The system time needs to be set correctly so that a device can coordinate properly with other devices. However, on networks that contain multiple devices, setting or adjusting the system time manually involves a heavy workload and may compromise the clock accuracy. To ensure that all devices enabled with clock synchronization can obtain the same time, you can configure the Network Time Protocol (NTP) feature. This allows devices to synchronize their clocks so that they can provide diverse applications based on consistent time.

For details about configuration parameters, see huawei-tm.yang.

Procedure

Step 1 Configure the UTC time.

date-time date-time *YYYY-MM-DDTHH:MM:SSZ*

Value range: 2000-01-01T00:00:00Z to 2037-12-31T23:59:59Z. If the UTC time configured on the device is earlier than the delivery time, the pre-configuration certificate is unavailable. As a result, the web page and Qiankun Cloud cannot go online.

Step 2 Enter the editing view.

edit-config

Step 3 Enter the local time zone setting view.

tm timezone-configuration

Step 4 Configure the local time zone.

timezone-name timezone-name option [**add** | **minus**] **timezone-offset** *HH:MM:SS*

The default values of **timezone-name**, **option**, and **timezone-offset** are *DefaultZoneName*, *add*, and *00:00:00*, respectively.

add: adds the specified time zone offset on the basis of the UTC time. The default system time (UTC time) plus the time zone offset (*HH:MM:SS*) is the time in the time zone specified by *timezone-name*.

minus: subtracts the specified time zone offset from the UTC time. The default system time (UTC time) minus the time zone offset (*HH:MM:SS*) is the time in the time zone specified by *timezone-name*.

After the local time zone is configured, the current system time is the result of the following: UTC time ± *offset*.

The value of timezone-name can contain a maximum of 32 characters.

The time zone offset ranges from -12:00:00 to 14:00:00.

Step 5 Commit the configuration.

```
commit
```

----End

Example

The following uses the Beijing time zone as example. To set the system time to 2021-05-26 07:31:08, set the UTC time to *2021-05-25T23:30:46Z*, and then configure the local time zone as follows: set *timezone-name* to *BEIJING*, specify *add*, and set *timezone-offset* to *08:00:00* in the corresponding command. After committing the configuration, query the local system time.

```
[user@HUAWEI]
MDCLI> date-time date-time
2021-05-25T23:30:46Z

[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> tm timezone-configuration

[(gl)user@HUAWEI]/tm/timezone-
configuration
MDCLI> timezone-name BEIJING option add timezone-offset
08:00:00

[*(gl)user@HUAWEI]/tm/timezone-
configuration
MDCLI> commit

[(gl)user@HUAWEI]/tm/timezone-
configuration
MDCLI> return

[user@HUAWEI]
MDCLI> display tm/local-time
{
  "current-time": "2021-05-26T07:31:08+08:00",
  "weekday": "Wednesday"
}
```


Verifying the Configuration

Perform the following operations in the initial view of the MDCLI. After each operation, run the **return** command to return to the initial view of the MDCLI.

- Run the **display tm/date-and-time** command to check the UTC time.
- Run the **display tm/timezone-configuration** command to check information about the local time zone.
- Run the **display tm/local-time** command to check the local system time.

3.6 NTP Configuration

3.6.1 Overview of NTP

Definition

Network Time Protocol (NTP) is an application layer protocol in the TCP/IP protocol suite. It is used to synchronize clocks between a series of distributed time servers and clients.

Purpose

NTP is used to synchronize the time of all clock devices on a network. If time is not synchronized on a network, time errors may occur because devices run their own clocks. NTP synchronizes the time on all devices, enabling them to provide various applications based on consistent time.

NTP is mainly used in the following scenarios where the clocks of all network devices must be consistent:

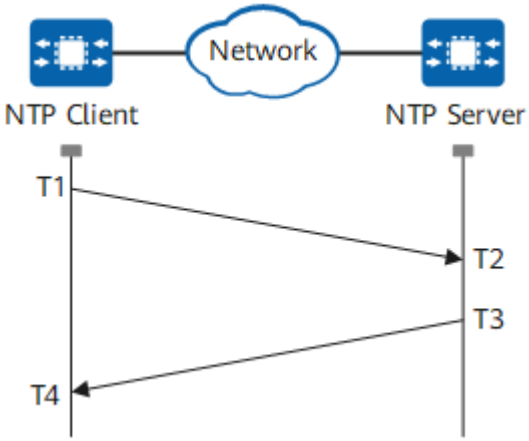
- Network management: Synchronized time is used as a reference when a network management system (NMS) analyzes the logs and debugging information collected from different devices.
- Charging system: Synchronized time is required to ensure the accuracy and trustworthiness of charging information.
- Several systems interworking on the same complex event: The systems must use the same clock for reference to ensure proper sequencing of operations.
- Incremental backup between the backup server and client: Synchronized time ensures integrity of the backup data which can be used for production system recovery.

3.6.2 Understanding NTP

3.6.2.1 NTP Fundamentals

Figure 3-11 shows the NTP message exchange process.

Figure 3-11 NTP message exchange process



The NTP message exchange process is as follows:

1. The NTP client sends an NTP request message to the NTP server. The NTP request message is timestamped as T1 when it leaves the NTP client.
2. The NTP server receives the NTP request message and adds the timestamp T2 to this message.
3. The NTP server sends an NTP response message to the NTP client. The NTP response message is timestamped as T3 when it leaves the NTP server.
4. The NTP client receives the NTP response message and adds the timestamp T4 to this message.

Following this message exchange, the NTP client has four timestamps: T1, T2, T3, and T4.

An example assumes that the one-way delay of a link is Delay and the time difference between the NTP client and NTP server is Offset (the current time of the NTP server minus the time of the NTP client). The calculation formulas are as follows:

T2 - T1 = Delay + Offset

T4 - T3 = Delay - Offset

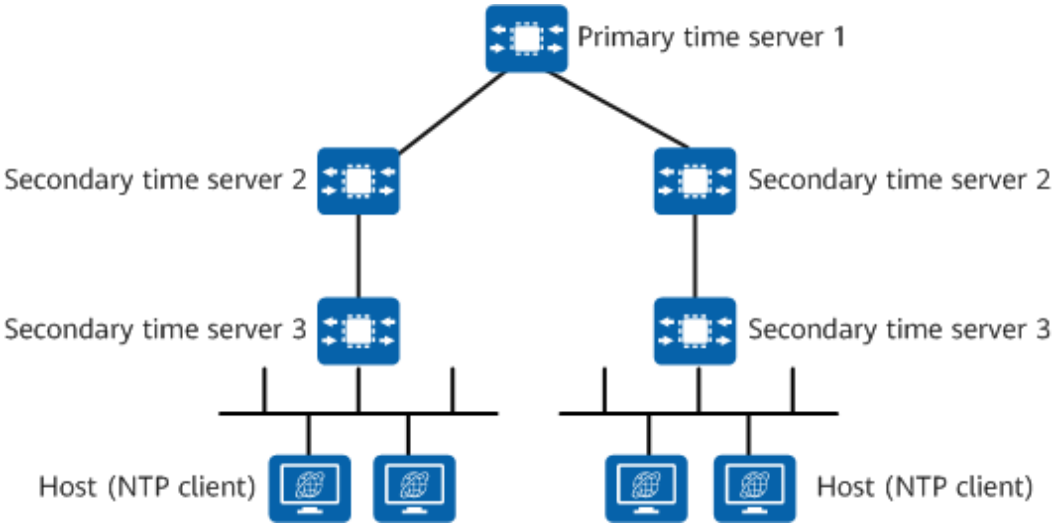
Therefore, Offset is calculated as follows: Offset = [(T2 - T1) - (T4 - T3)]/2. The NTP client adjusts its clock based on the Offset value to achieve synchronization with the NTP server.

The preceding is a brief introduction of how NTP works. Standard NTP algorithms can be used to further guarantee the precision of clock synchronization.

3.6.2.2 Network Structure of NTP

Figure 3-12 shows the network structure of NTP. An NTP synchronization subnet is a network of primary and secondary time servers, clients, and interconnecting transmission paths.

Figure 3-12 Network structure of NTP



A primary time server is directly synchronized to a primary reference source, usually a radio clock or a global positioning system (GPS). A secondary time server derives synchronization through a primary time server or other secondary time servers, and uses NTP to transmit time information to other hosts on a local area network (LAN).

In normal cases, the master and slave servers in a synchronization subnet form a hierarchical master-slave structure. In this hierarchical structure, the master server is located at the root, the secondary server is close to the leaf node, the number of layers increases, and the accuracy decreases.

If one or more primary or secondary time servers fail or the network paths between them fail, the synchronization subnet is automatically adjusted to maintain accurate and reliable time.

- When all primary time servers being used on a synchronization subnet fail, one or more backup primary time servers continue operation.
- When all primary time servers on the synchronization subnet fail, the secondary time servers synchronize among themselves. They drop off the synchronization subnet and free-run using the last determined time and frequency. Timekeeping errors are no greater than a few milliseconds per day if oscillators are appropriately stabilized, even in extended outage periods.

3.6.2.3 Operating Modes of NTP

Overview

Table 3-20 lists the operating modes of NTP.

Table 3-20 Operating modes of NTP

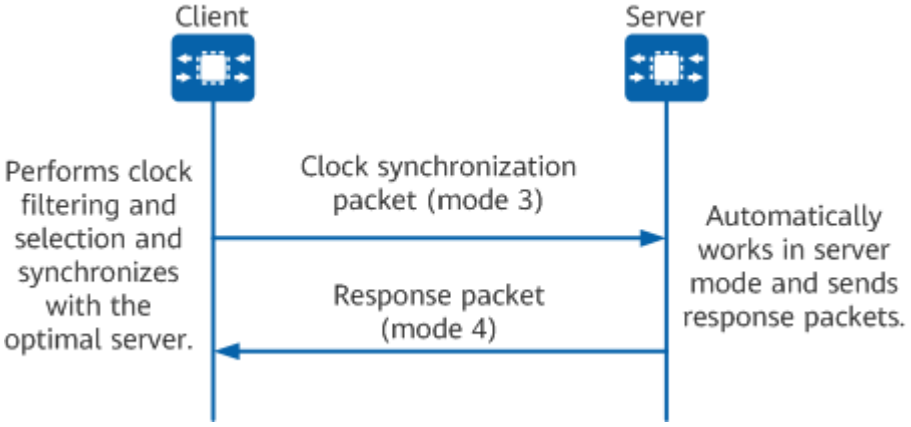
Operating Mode	Description	Usage Scenario
Client/Server mode	The client synchronizes its clock with the clock on the server.	In this mode, the server and client run at a high stratum level on the synchronization subnet. In this mode, the IP address of the server needs to be obtained in advance.
Peer mode	The symmetric active peer and symmetric passive peer can synchronize with each other. The peer with a lower stratum level (larger stratum value) is synchronized with the peer with a higher stratum level (smaller stratum value).	In this mode, the host runs at a relatively low stratum level on the synchronization subnet.
Broadcast mode	The server periodically sends clock synchronization packets to a broadcast address.	This mode applies to high-speed networks that have numerous workstations but lower requirements on synchronization accuracy. In typical scenarios, one or more time servers periodically send broadcast packets to workstations, which then determine the time based on a millisecond-level delay.
Multicast mode	The server periodically sends clock synchronization packets to a multicast address.	This mode applies to scenarios where a large number of clients are distributed on the network. In this mode, the NTP server multicasts an NTP packet to all clients, thereby lowering the number of NTP packets sent on the network.
Manycast mode	Manycast servers continuously listen for request packets that manycast clients periodically send to a multicast address in search of a server with the fewest number of hops.	This mode applies to scenarios where multiple servers are sparsely distributed on the network. Clients can discover and synchronize with the closest manycast server. This mode applies to networks where servers are not stable and clients do not need to be reconfigured if a server fails.

Client/Server Mode

Figure 3-13 shows the packet exchange process in client/server mode. The client synchronizes its clock with the clock on the server. The server provides

synchronization information for the client but does not alter its own clock. In client/server mode, the server is also called a unicast server to distinguish it from the broadcast server, multicast server, and manycast server in other modes.

Figure 3-13 Client/Server mode



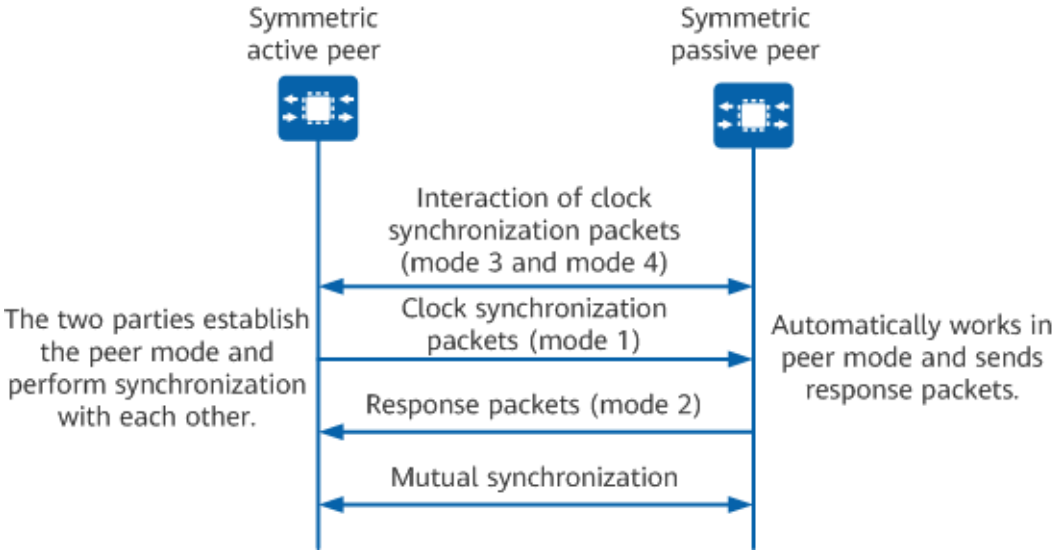
The packet exchange process in client/server mode is as follows:

1. The client periodically sends packets to the server. The value of the Mode field in the packets is set to 3 (client mode). A client will not verify the reachability and stratum of the server.
2. After receiving the request packet, the server sends a response packet in which the Mode field is set to 4 (server mode). The server fills in the required information to the response packet before sending it to the client. The server does not need to retain any status information.
3. After receiving the response packet, the client performs the clock filter and selection procedure and then synchronizes its clock to the server that provides the optimal clock.

Peer Mode

Figure 3-14 shows the packet exchange process in peer mode.

Figure 3-14 Peer mode



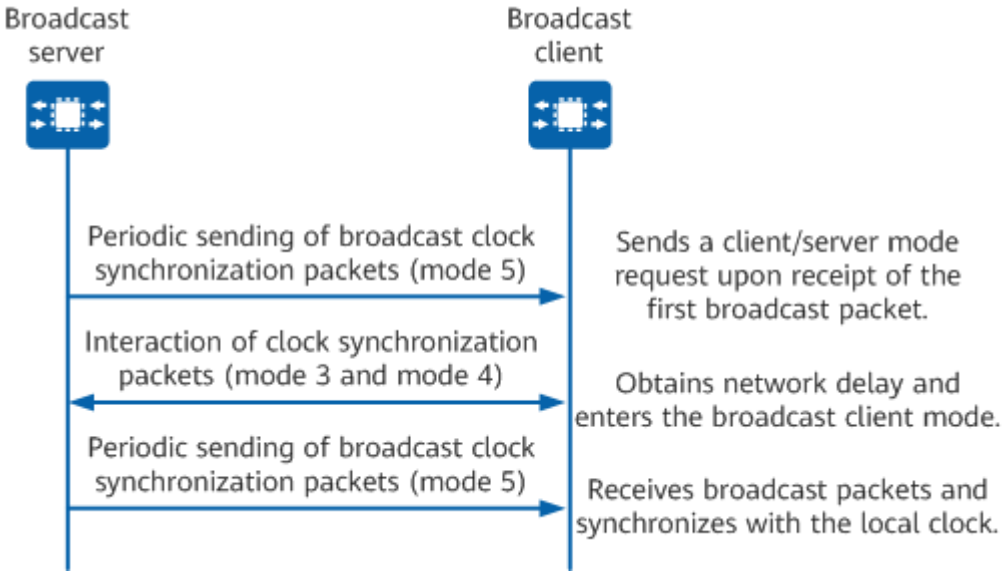
The packet exchange process in peer mode is as follows:

1. The symmetric active peer sends an NTP request packet to the symmetric passive peer, with the Mode field being 3 (client mode). The symmetric passive peer replies with an NTP response packet, in which the Mode field is set to 4 (server mode).
2. The active peer periodically sends packets to the passive peer. The value of the Mode field in a packet is set to 1, indicating that the packet is sent by the active peer. Whether the peer is reachable and the number of layers of the peer are not considered.
3. After receiving the request packet, the symmetric passive peer sends a response packet in which the Mode field is set to 2 (symmetric passive peer). The symmetric passive peer does not need to be configured. A host establishes a connection and sets relevant state variables only after an NTP packet is received.
4. After the peer relationship is set up, the symmetric active and passive peers can synchronize with each other. The peer with a lower stratum level (larger stratum value) is synchronized with the peer with a higher stratum level (smaller stratum value).

Broadcast Mode

Figure 3-15 shows the packet exchange process in broadcast mode. In this mode, servers typically run high-speed broadcast media over the network. They provide synchronization information to all clients, but do not alter their own clocks.

Figure 3-15 Broadcast mode



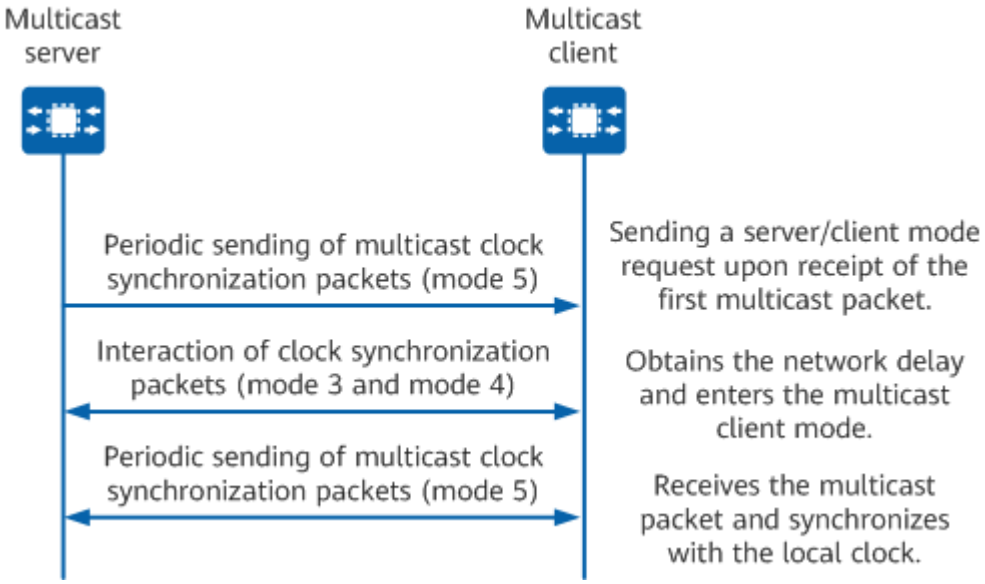
The packet exchange process in broadcast mode is as follows:

1. The broadcast server periodically sends clock synchronization packets to the broadcast address 255.255.255.255. The Mode field in the packets is set to 5 (broadcast mode or multicast mode), regardless of whether the client is reachable or the number of layers.
2. The client listens for the broadcast packets sent from the server. After receiving the first broadcast packet, the client temporarily starts in client/server mode to exchange packets with the server. This allows the client to estimate the network delay.
3. The client then enters the broadcast mode, continues to listen for the subsequent broadcast packets, and synchronizes the local clock according to the subsequent broadcast packets.

Multicast Mode

Figure 3-16 shows the packet exchange process in multicast mode. In this mode, servers typically run high-speed broadcast media over the network. They provide synchronization information to all clients, but do not alter their own clocks.

Figure 3-16 Multicast mode



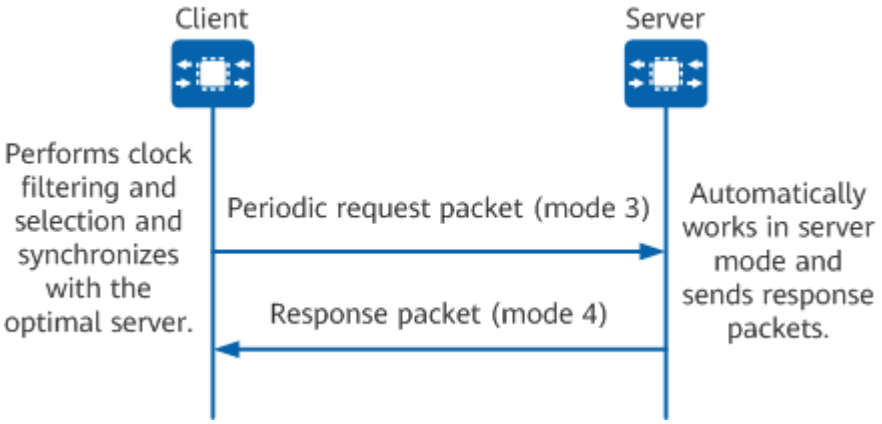
The packet exchange process in multicast mode is as follows:

1. The multicast server periodically sends clock synchronization packets to an IPv4 or IPv6 multicast address. The Mode field in the packets is set to 5 (broadcast or multicast mode).
2. The client listens for the multicast packets sent from the server. After receiving the first multicast packet, the client temporarily starts in client/server mode to exchange packets with the server. This allows the client to estimate the network delay.
3. The client then enters the multicast mode, continues to listen for the subsequent multicast packets, and synchronizes the local clock according to the subsequent multicast packets.

Manycast Mode

Figure 3-17 shows the packet exchange process in manycast mode. In this mode, servers provide synchronization information to all clients and do not alter their own clocks.

Figure 3-17 Manycast mode



The packet exchange process in manycast mode is as follows:

1. The manycast client periodically sends request packets to an IPv4 or IPv6 multicast address to search for the closest manycast server (smallest TTL). The value of the Mode field is set to 3 (client mode).

   The initial TTL value of a request packet sent by the client is 1. The value increases by 1 each time a request packet is sent until either the client receives a response packet or the TTL value reaches the upper limit. Receipt of a response packet indicates that the client has found the closest manycast server. To subsequently maintain the connection with this server, the client sends a packet every time a timeout period expires. If the client does not receive a response packet when the TTL reaches the upper limit, the client stops sending request packets for a certain period of time (a timeout period). This timeout period allows all connections to be cleared. After the timeout period expires, the client repeats the preceding process.
2. The manycast server continuously listens for packets. If server synchronization is possible, the server unicasts a response packet to the client, with the Mode field set to 4 (server mode).
3. After receiving the response packet, the client performs the clock filter and selection procedure and then synchronizes its clock to the server that provides the optimal clock.

3.6.2.4 NTP Clock Source Selection

When multiple valid server candidates exist for the clock source selection, a client selects the most accurate and reliable server according to certain rules. NTP determines the quality of each clock source based on the following parameters:

- Offset: indicates the maximum-likelihood time offset of the server clock relative to the system clock.
- Delay: indicates the round-trip delay between the client and server.
- Dispersion: indicates the maximum error inherent in the measurement.
- Jitter: indicates the root mean square (RMS) average of the most recent offset differences.

Upon receipt of each NTP response packet, a client updates the values of these four parameters to measure the system clock relative to each server clock. The client uses the selection, cluster, and combine algorithms to determine the most accurate and reliable candidates to synchronize the system clock. The selection algorithm is used to select a list of accurate and reliable servers based on the values of the offset, delay, dispersion, and jitter parameters. The cluster algorithm uses statistical principles to find the most accurate set of truechimers (a truechimer is a clock that maintains timekeeping accuracy to a previously published and trusted standard). The combine algorithm computes the final clock offset by statistically averaging the surviving truechimers.

3.6.2.5 NTP Packet Format

Figure 3-18 shows the NTP packet format.

Figure 3-18 NTP packet format

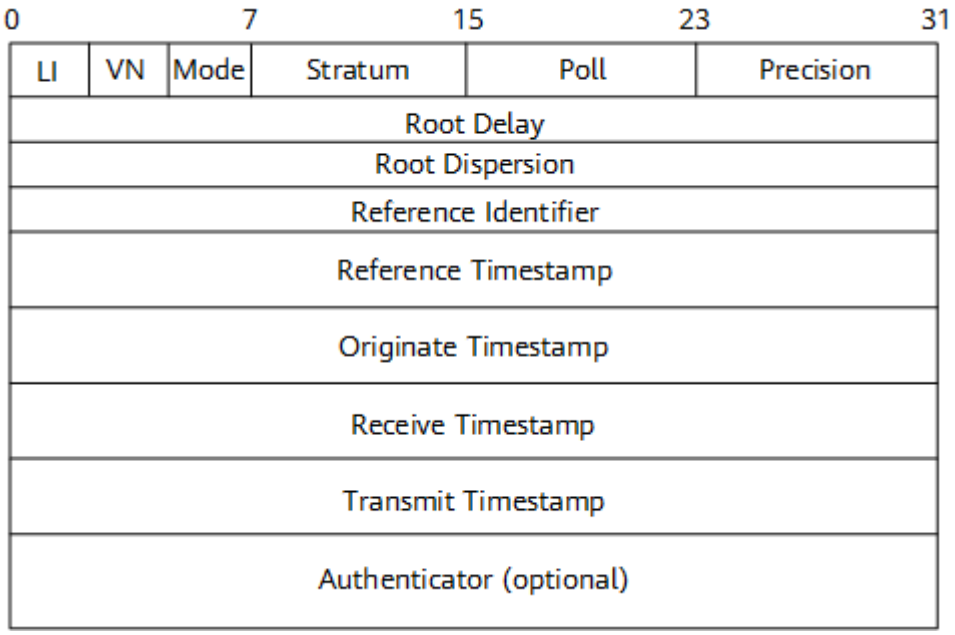


Table 3-21 Description of each field in an NTP packet

Field	Length	Description
LI (Leap Indicator)	2 bits	A code warning of an impending leap second to be inserted or deleted in the NTP timescale. The bit values are defined as follows: • 00: no warning • 01: last minute has 61 seconds • 10: last minute has 59 seconds • 11: alarm condition (clock not synchronized)
VN (Version Number)	3 bits	NTP version number. The current version is 3.
Mode	3 bits	NTP mode. The values are defined as follows: • 0: reserved • 1: symmetric active • 2: symmetric passive • 3: client mode • 4: server mode • 5: broadcast mode • 6: reserved for NTP control messages • 7: reserved for private use

Field	Length	Description
Stratum	8 bits	Stratum level of the local clock. It defines the precision of the clock. The value of this field ranges from 1 to 15. A stratum 1 clock has the highest precision.
Poll	8 bits	Maximum interval between successive messages.
Precision	8 bits	Precision of the local clock.
Root Delay	32 bits	Total round-trip delay to the primary reference source.
Root Dispersion	32 bits	Maximum error relative to the primary reference source.
Reference Identifier	32 bits	ID of a reference clock.
Reference Timestamp	64 bits	Local time at which the local clock was last set or corrected. Value 0 indicates that the local clock is never synchronized.
Originate Timestamp	64 bits	Local time at which an NTP request packet departed the client for the server.
Receive Timestamp	64 bits	Local time at which an NTP request packet arrived at the server.
Transmit Timestamp	64 bits	Local time at which an NTP response packet departed the server for the client.
Authenticator	96 bits	(Optional) Authenticator information.

3.6.3 Configuration Precautions for NTP

Licensing Requirements

NTP is not under license control.

Hardware Requirements

Table 3-22 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.6.4 Configuring Basic NTP Functions

3.6.4.1 Configuring the NTP Client

Context

If the server clock changes or multiple NTP servers are available, you need to set the clock synchronization parameters on the client clock. Such parameters include the interval and the maximum synchronization distance threshold for synchronizing the client clock. The client clock synchronizes with the clock source based on the configured clock synchronization parameters.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the NTP unicast configuration view.

```
ntp unicasts
```

Step 3 Configure the NTP server.

```
unicast ip-address ipv4-address type Server vpn-name _public_
```

Step 4 (Optional) Set the interface for sending and receiving packets.

```
ifname interface-name
```

Step 5 Commit the configuration.

```
commit
```

----End

Example

To enable the NTP service, you need to configure the NTP client on the local device. Enter the NTP unicast configuration interface in the editing view, configure the NTP server information, set the interface name, and save the configuration. After the configuration is complete, you can run the **display** command to check whether the current configuration is invalid.

```
[device@HUAWEI]
MDCLI> edit-config

[(gl)device@HUAWEI]
MDCLI> ntp unicasts

[(gl)device@HUAWEI]/ntp/unicasts
MDCLI> unicast ip-address 192.168.1.10 type Server vpn-name
_public_

[*(gl)device@HUAWEI]/ntp/unicasts/unicast[ip-address="192.168.1.10"][type="Server"][vpn-
name="_public_"]
MDCLI> ifname GE0/0/0
```



```
[*(gl)device@HUAWEI]/ntp/unicasts/unicast[ip-address="192.168.1.10"][type="Server"][vpn-
name="_public_"]
MDCLI> commit

[(gl)device@HUAWEI]/ntp/unicasts/unicast[ip-address="192.168.1.10"][type="Server"][vpn-
name="_public_"]
MDCLI> display this
{
  "ip-address": "192.168.1.10",
  "type": "Server",
  "vpn-name": "_public_",
  "ifname": "GE0/0/0"
}
```

3.6.4.2 (Optional) Configuring the Peer NTP Server

Context

After configuring the client, you can configure the peer server based on service requirements.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the NTP unicast configuration view.

```
ntp unicasts
```

Step 3 Configure the NTP server.

```
unicast ip-address ipv4-address type Server vpn-name _public_
```

Step 4 (Optional) Specify the source interface for sending NTP packets.

```
ifname interface-name
```

Step 5 (Optional) Set the priorities of different time synchronization servers.

```
is-preferred [ true | false ]
```

By default, the priority of a time synchronization server is **false**.

Step 6 (Optional) Set the maximum interval for synchronizing the client clock.

```
max-poll-interval max-number
```

By default, the maximum interval for synchronizing the client clock is 36.4 hours, and the default interval is 1024 seconds.

Step 7 (Optional) Set the minimum interval for synchronizing the client clock.

```
min-poll-interval min-number
```

By default, the minimum interval for synchronizing the client clock is 16 seconds, and the default interval is 64 seconds.

Step 8 Commit the configuration.

```
commit
```

----End

3.6.4.3 Verifying the Configuration

Prerequisites

All the basic NTP functions have been configured.

Procedure

- Run the **display ntp/unicasts** command to check the configured NTP service.
- Run the **display ntp/status** command to check the status of the NTP service.
- Run the **display ntp/full-sessions** command to check the tracing path between the local device and the reference clock source.

----End

3.6.5 Maintaining NTP

Monitoring the NTP Operating Status

During routing maintenance, you can run the following commands in any view to monitor the NTP operating status.

Table 3-23 Monitoring the NTP operating status

Operation	Command
Check the status of dynamic sessions maintained by the NTP service.	**display ntp/full-sessions**
Check the status of the NTP service.	**display ntp/status**

3.6.6 Troubleshooting NTP

3.6.6.1 NTP Authentication Does Not Take Effect

Fault Symptom

The NTP authentication function does not take effect. Clock synchronization can be performed even when the server or client is invalid (for example, the keys on the server and client are inconsistent).

Possible Causes

NTP authentication is not configured before basic NTP functions are configured.

Procedure

Step 1 Check the current NTP configuration.

```
display ntp/unicasts/
```

Step 2 Configure basic NTP functions.

```
edit-config
ntp unicasts
unicast ip-address ipv4-address type Server vpn-name _public_
commit
```

----End

3.7 NETCONF Configuration

3.7.1 NETCONF Overview

Definition

The Network Configuration Protocol (NETCONF) is an extensible markup language (XML)-based network configuration and management protocol. NETCONF uses a simple remote procedure call (RPC) mechanism to implement communication between a client and a server. NETCONF provides a method for a network management station (client), which is a central computer that runs network management software, to remotely manage and monitor devices.

The NMS uses NETCONF to implement local management and perform operations such as adding, modifying, and deleting configurations of remote devices. This protocol allows a device to provide a complete and formal application programming interface (API). Network management applications can use this API to send and receive complete or partial configuration data sets.

Purpose

As networks become larger and more complex, the Simple Network Management Protocol (SNMP) can no longer meet the requirements for managing and configuring networks. This is where NETCONF comes into play.

Table 3-24 lists the differences between SNMP and NETCONF.

Table 3-24 Comparison between SNMP and NETCONF

Function	SNMP	NETCONF
Configuration management	SNMP does not provide a lock mechanism when multiple users perform operations on the same configuration.	NETCONF provides a lock mechanism to ensure that operations performed by multiple users do not conflict with each other.
Querying	SNMP can be used to query one or more records in a table, requiring multiple interactions.	NETCONF allows you to directly query the configuration data of the system and supports data filtering.

Function	SNMP	NETCONF
Scalability	Poor.	Good. • The NETCONF protocol framework adopts a hierarchical structure with four independent layers. Extensions to one layer have little impact on the other layers. • XML encoding helps expand NETCONF's management capabilities and system compatibility.
Safety	SNMPv2, released by the International Architecture Board (IAB) in 1996, provides only limited security improvements over SNMPv1. SNMPv3, released in 2002, provides important security improvements over the previous two versions, but it is inextensible. This is because SNMPv3 security parameters depend on the use of the security model.	NETCONF uses existing security protocols to ensure network security and is not specific to any security protocols. NETCONF is therefore more flexible than SNMP in security protection. **NOTE** Secure Shell (SSH) is the preferred transport protocol in NETCONF and is used to transmit XML information.

Benefits

- Facilitates configuration data management and interoperability between different vendors' devices by using XML encoding to define messages and using the RPC mechanism to modify configuration data.
- Reduces network faults caused by manual configuration errors.
- Improves the efficiency of tool-based upgrades to system software.
- Provides high extensibility, allowing different vendors to define additional NETCONF operations.
- Improves data security using authentication and authorization mechanisms.

NOTE

This document is written according to device information obtained under lab conditions and therefore may not cover all scenarios. Due to factors such as version upgrades and differences in device models, the content provided in this document may differ from the information on user device interfaces. Such information takes precedence over the content provided by this document.

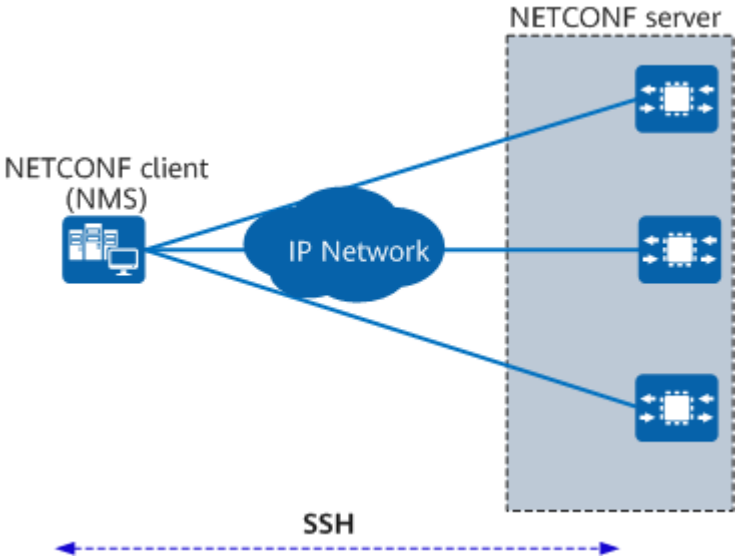
3.7.2 Understanding NETCONF

3.7.2.1 NETCONF Network Architecture

Basic Network Architecture of NETCONF

Figure 3-19 shows the basic network architecture of NETCONF. It must contain at least one network management system (NMS) that runs on an NMS server and manages devices.

Figure 3-19 Basic network architecture of NETCONF



The architecture consists of two main elements: client and server.

Table 3-25 Main elements in the basic network architecture of NETCONF

Main Element	Function
NETCONF client	A client manages network devices using NETCONF. • The client sends RPC requests to a server to query or modify one or more parameter values. • The client learns the status of a managed device based on the alarms and events sent by the server.

Main Element	Function
NETCONF server	A server maintains information of managed devices, responds to RPC requests of the client, and sends the requested management data to the client. • When receiving a request from the client, the server parses the request and sends a reply to the client. • If a fault or another type of event occurs on a managed device, the server reports the alarm or event to the client through the notification mechanism. This allows the client to learn the status of the managed device.

A network device must support at least one NETCONF session, which is a logical connection between a client and a server. The information that a client obtains from a server can be configuration or status data.

- The client can modify and operate configuration data to implement a user-expected status of the server.
- The client cannot modify status data, which mainly includes the running status and statistics of the server.

Establishing a Basic NETCONF Session

1. A device triggers the establishment of a NETCONF session. It then completes SSH connection setup after authentication and authorization are complete.
2. The client and server complete NETCONF session establishment and capability negotiation.
3. The client sends one or more requests to the server for RPC interaction (authorization). For example:
   - Modify and commit the configuration.
   - Query the configuration data or status.
   - Perform maintenance operations on the device.
4. Terminate the NETCONF session.
5. Tear down the SSH connection.

Session Interaction Between the Client and Server

After a NETCONF session is established, the client and server immediately exchange Hello messages with each other (these messages include the <hello> element, which contains the set of capabilities supported locally). If both ends support a capability, they can implement specific management functions based on this capability.

Capability set negotiation result: For a standard capability set (except Notification), the capability set supported by the server is used as the negotiation

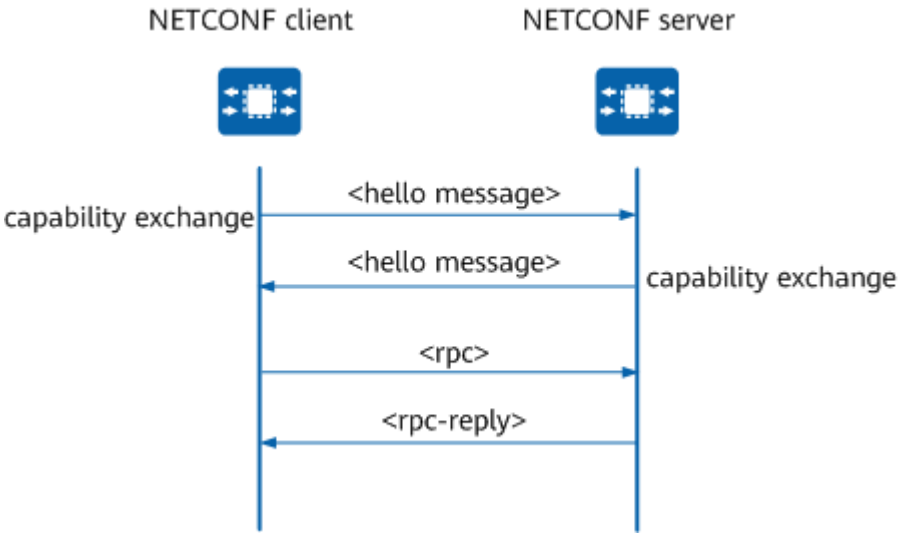
result. For an extended capability set, the intersection of the capability sets supported by both the client and server is used as the negotiation result.

NOTE

A NETCONF server can send a <hello> element to advertise the capabilities that it supports. In scenarios where a Huawei device connects to a non-Huawei device, if the capabilities contained in a <hello> element sent from the peer are all standard capabilities, the Huawei device replies with a YANG packet.

After a NETCONF server exchanges <hello> elements with a NETCONF client, the server waits for <rpc> elements from the client. The server returns an <rpc-reply> element in response to each <rpc> element. **Figure 3-20** shows the capability interaction between the NETCONF server and client.

Figure 3-20 Capability interaction between the server and client



3.7.2.2 NETCONF Protocol Architecture

NETCONF Protocol Framework

Like the open systems interconnection (OSI) model, the NETCONF protocol framework also uses a hierarchical structure. Each layer encapsulates certain functions and provides services for its upper layer.

This hierarchical structure enables each layer to focus only on a single aspect of NETCONF and reduces the dependencies between different layers. In this way, the impact that internal implementation imposes on other layers can be minimized.

Table 3-26 describes the four layers of the NETCONF protocol framework.

Table 3-26 NETCONF protocol framework

Layer	Example	Description
Layer 1: transport	BEEP, SSH, and SSL	The transport layer provides a communication path for interaction between a NETCONF client and server. NETCONF can be carried over any transport protocol that meets the following basic requirements: • The transport protocol is connection-oriented and establishes a persistent connection between the NETCONF client and server. This connection provides reliable, sequenced data transmission. • NETCONF relies on the transport layer for user authentication, data integrity, and security encryption. • The transport protocol provides a mechanism to distinguish the session type (client or server) for NETCONF. **NOTE** Currently, the device supports only SSH as the transport layer protocol for NETCONF.
Layer 2: messages	<rpc>, <rpc-reply>, <notification>	The message layer provides a simple RPC request and response mechanism independent of transport protocols. The client encapsulates RPC request information in the <rpc> element and sends it to the server through a secure and connection-oriented session. The server encapsulates RPC reply information (content at the operations and content layers) into the <rpc-reply> element and sends it to the client. Normally, the <rpc-reply> element encapsulates data required by the client or a configuration success message. If the client sends an incorrect request or the server fails to process a request from the client, the server encapsulates the <rpc-error> element containing detailed error information into the <rpc-reply> element and sends the <rpc-reply> element to the client.
Layer 3: operations	<get-config>, <edit-config>	The operations layer defines a series of basic operations used in RPC. These operations constitute basic capabilities of NETCONF.
Layer 4: content	Configuration data	The content layer describes configuration data involved in network management. The configuration data depends on vendors' devices. To date, only the content layer remains to be standardized for NETCONF. This layer has no standard NETCONF data modeling language or data model.

NETCONF Modeling Language

YANG is a data modeling language developed to design NETCONF-oriented configuration data, status data models, RPC models, and notification mechanisms.

The YANG data model is a machine-oriented model interface, which defines data structures and constraints to provide more flexible and complete data description.

Encoding Format

XML encoding is used in NETCONF, allowing complex hierarchical data to be expressed in a text format that can be read, saved, and manipulated with both traditional text tools and XML-specific tools.

XML-based network management uses XML to describe managed data and management operations. In this way, management information forms a database that is understandable to computers, allowing them to efficiently process network management data using enhanced management capabilities.

The format of the file header used in XML encoding is **<?xml version="1.0" encoding="UTF-8"?>**, where:

- **<?**: indicates the start of an instruction.
- **xml**: identifies an XML file.
- **version="1.0"**: indicates that the XML1.0 standard version is used.
- **encoding**: indicates the character set encoding format. Only UTF-8 encoding is supported.
- **?>**: indicates the end of an instruction.

Communication Modes

The NETCONF client and server communicate through the RPC mechanism. To implement the communication, a secure and connection-oriented session must be established. After receiving an RPC request from the client, the server processes the request and sends a response message to the client. The RPC request from the client and the response message from the server are encoded in XML format. The XML-encoded <rpc> and <rpc-reply> elements provide a request and response message framework independent of transport layer protocols. **Table 3-27** lists some basic RPC elements.

Table 3-27 Element description

Element	Description
<rpc>	Encapsulates a request that the client sends to the NOTCONF server.
<rpc-reply>	Encapsulates a response message that the NETCONF server sends in reply to each <rpc> request it receives.
<rpc-error>	Notifies a client of an error that occurs during processing of an <rpc> request. The server encapsulates the <rpc-error> element in the <rpc-reply> element and sends the <rpc-reply> element to the client.

Element	Description
<ok>	Notifies a client that no errors occur during processing of an <rpc> request. The server encapsulates the <ok> element in the <rpc-reply> element and sends the <rpc-reply> element to the client.

Capability Set

NETCONF defines the syntax and semantics of capabilities. The protocol allows the client and server to notify each other of supported capabilities. The client can send the operation requests only within the capability range supported by the server.

A capability set includes basic and extended functions implemented based on NETCONF. The NETCONF capability set includes a standard capability set defined by the IETF standards organization. In addition, a device can use the capability set to add a protocol operation so that the operation range of the existing configuration object is extended.

Each capability is identified by a unique uniform resource identifier (URI). The URI format of the capability set defined by NETCONF is as follows:

```
urn:ietf:params:xml:ns:netconf:capability:{name}:{version}
```

In addition to the NETCONF-defined capability set, a vendor can define additional capability sets to extend management functions. A module that supports the YANG model needs to add YANG notifications to Hello messages before sending the messages. The message format is as follows:

```
<capability>urn:huawei:yang:huawei-ifm?module=huawei-ifm&amp;revision=2022-03-30</capability>
```

Configuration Database

A configuration database is a collection of complete configuration parameters for a device. **Table 3-28** describes NETCONF-defined configuration databases.

Table 3-28 NETCONF-defined configuration databases

Configuration Database	Description
<running/>	Stores the device's currently running configuration, status information, and statistics. Unless the NETCONF server supports the candidate capability, this configuration database is the only standard database that is mandatory. To support modification of the <running/> configuration database, a device must have the writable-running capability.

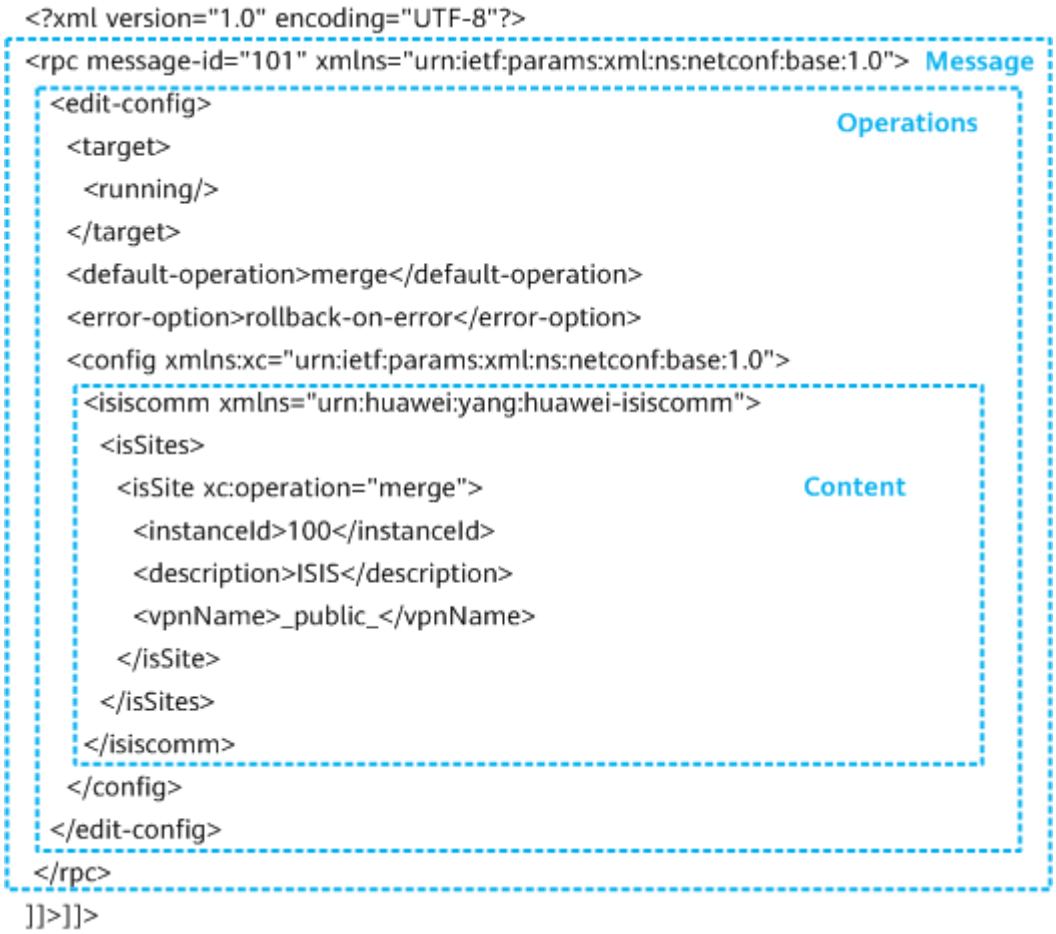
Configuration Database	Description
<candidate/>	Stores the configuration data to be run on a device. An administrator can perform operations on the <candidate/> configuration database. Any change to the <candidate/> database does not directly affect the configurations currently running on the device. To support the <candidate/> configuration database, a device must have the candidate capability. **NOTE** The <candidate/> configuration databases supported by devices allow inter-session data sharing.
<startup/>	Stores the configuration data loaded during device startup, which is similar to the saved configuration file. To support the <startup/> configuration database, a device must have the distinct startup capability.

3.7.2.3 NETCONF Message Formats

Request Message

Figure 3-21 shows the structure of a complete NETCONF request message.

Figure 3-21 Structure of a NETCONF YANG request message



A NETCONF request message consists of three layers. **Table 3-29** describes the fields in a NETCONF request message.

- Message: The message layer provides a simple and independent mechanism of transmitting frames for RPC messages. The client encapsulates an RPC request into an <rpc> element and sends it to the server, which encapsulates the result of processing this request into the <rpc-reply> element and sends it to the client.
- Operations: The operations layer defines a set of basic NETCONF operations. These operations are invoked by RPC methods that are based on XML encoding parameters.
- Content: The content (managed object) layer defines a configuration data model. Currently, mainstream configuration data models include the schema model and YANG model.

Table 3-29 Fields in a NETCONF message

Field	Description
message-id	Indicates the information code. The value is specified by the client that initiates the RPC request. After receiving the RPC request message, the server saves the message-id attribute, which is used in an <rpc-reply> message to be generated.
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"	Indicates the NETCONF XML namespace. **base** indicates that basic operation types are supported. base1.0: supports the <running/> configuration database and defines the following basic operations: <get-config>, <get>, <edit-config>, <copy-config>, <delete-config>, <lock>, <unlock>, <close-session>, and <kill-session>. You can set the <error-option> parameter to rollback-on-error. base1.1: an upgrade of base1.0, with the following changes: ● The remove operation is added to the operation attribute of <edit-config>. ● The well-known error-tagmalformed-message is added, and the well-known error-tagpartial-operation is obsolete. ● The namespace wildcarding mechanism is added for subtree filtering. ● The chunked framing mechanism is added to resolve the security issues in the end-of-message (EOM) mechanism. If you want to perform an operation in base1.1, the client must support base1.1 so that this capability can be advertised during capability set exchange.
<edit-config>	Indicates the operation type.

Field	Description
<target>	Indicates the target data set to be operated: • running • candidate • startup
<default-operation>	Indicates the default operation type.
<error-option>	Indicates the mode for processing subsequent operations if an error occurs during an <edit-config> operation. The options are as follows: • **rollback-on-error**: stops the operation after an error occurs and rolls back the configuration to the state before the <edit-config> operation is performed. This operation is supported only when the device supports the rollback-on-error capability.
<config>	Indicates a group of hierarchical configuration items defined in the data model. The configuration items must be placed in the specified namespace and meet the constraints of that data model, as defined by its capability set.
]]>]]>	Indicates the end character of an XML message. **NOTE** When a server exchanges XML packets with a client, the packets must be concluded with the end character **]]>]]>**. Otherwise, the device cannot identify the XML packets and does not respond to them. By default, the end character is automatically added to XML messages sent by a device. The examples provided in this document omit the end character for brevity. If the capability set in the <hello> elements contains base1.1, the RPC messages in the YANG model support the chunk format. Messages in chunk format can be fragmented. The end character is **\n##\n**.

Response Message

If a request message is successfully executed, the device returns a successful response. Otherwise, the device returns a failed response.

- For a successful response, an <rpc-reply> message carrying the <ok> element is returned.

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           message-id="10">
  <ok/></rpc-reply>
```

- For a failed response, an <rpc-reply> message carrying the <rpc-error> element is returned.

```
<?xml version="1.0" encoding="utf-8"?
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>data-exists</error-tag>
    <error-severity>error</error-severity>
    <error-app-tag>43</error-app-tag>
    <error-path xmlns:acl="urn:huawei:yang:huawei-acl">acl:acl/acl:groups/
acl:group[acl:identity='2000']</error-path>
    <error-message xml:lang="en">Invalid ACL number:Number can not be the number of an existent
ACL.</error-message>
  </rpc-error>
</rpc-reply>
```

Table 3-30 Description of each field in a response message

Field	Description
xmlns	Indicates the NETCONF XML namespace.
message-id	Indicates the information code. The value is specified by the client that initiates the RPC request. After receiving the RPC request message, the server saves the message-id attribute, which is used in an <rpc-reply> message to be generated.
<error-type>	Defines the protocol layer at which an error occurs. The value can be transport, RPC, protocol, or application.
<error-tag>	Indicates the error information.
<error-severity>	Indicates the severity of an error. The value can be error or warning.
<error-app-tag>	Indicates a specific error type. This element is not present if no <error-tag> is associated with the error type.
<error-path>	Indicates the location where the error occurs and the name of the file where the error occurs.
<error-message>	Indicates the description of the error.

3.7.2.4 NETCONF Subtree Filtering

Overview

Subtree filtering is a mechanism that allows an application to query particular data for a <get> or <get-config> operation.

Subtree filtering provides a small set of filters for inclusion, simple content exact-match, and selection. The NETCONF agent does not need to use any semantics specific to any particular data model during processing, allowing for simple and centralized implementation policies.

Subtree Filter Components

Each node specified in subtree filtering represents a filter. The filter only selects nodes associated with the basic data model of a specified database on the NETCONF server. A node matching any filtering rule and element hierarchy is selected. **Table 3-31** describes subtree filter components.

Table 3-31 Subtree filter components

Component	Description
Namespace selection	If namespaces are used, the filter output will include only elements from the specified namespace.
Containment node	A containment node is a node that contains child elements within a subtree filter. For each containment node specified in a subtree filter, all data model instances that are exact matches for the specified namespaces and element hierarchy are included in the filter output.
Content match node	A content match node is a leaf node that contains simple content within a subtree filter. This node is used to select some or all of its relevant nodes for filter output and represents an exact-match filter of the leaf node element content.
Selection node	A selection node is an empty leaf node within a subtree filter. This node represents an explicit selection filter of the underlying data model. Presence of any selection nodes within a set of sibling nodes will cause the filter to select the specified subtrees and suppress automatic selection of the entire set of sibling nodes in the underlying data model.

- Namespace selection

  If the XML namespace associated with a specific node in the <filter> element is the same as that in the underlying data model, the namespace is matched.


```
<filter type="subtree">
 <top xmlns="urn:huawei:yang:example"/>
</filter>
```

In this example, the <top> element is a selection node. If the node namespace complies with **urn:huawei:yang:example**, the node and its child nodes will be included in the filter for output.

- Containment node

  The child element of a containment node can be a node of any type, including another containment node. For each containment node specified in the subtree filter, all data model instances that completely match the specified namespace and element hierarchy, and any attribute-matching expression are included in the output.

  ```
  <filter type="subtree">
   <top xmlns="urn:huawei:yang:example">
    <users/>
   </top>
  </filter>
  ```

  In this example, the <top> element is a containment node.

- Content match node

  A leaf node that contains simple content is called a content match node. It is used to select some or all of its sibling nodes for filter output and represents exact match of the leaf node element content.

  ```
  <filter type="subtree">
   <top xmlns="urn:huawei:yang:example">
    <users>
     <user>
      <name>fred</name>
     </user>
    </users>
   </top>
  </filter>
  ```

  In this example, both the <users> and <user> nodes are containment nodes, and the <name> node is a content match node. Because the sibling nodes of the <name> node are not specified, only <user> nodes that comply with namespace **urn:huawei:yang:example**, with their element hierarchies matching the **name** element and their values being **fred**, can be included in the filter output. All sibling nodes of the <name> node are included in the filter output.

- Selection node

  A node with empty content is called a selection node. The selection node suppresses the filtering output of non-selection nodes among sibling nodes. To choose a filtering expression mode, an empty tag (such as <foo/>) or an expression with explicit start and end tags (such as <foo> </ foo>) can be used to specify an empty leaf node. In this case, all blank characters will be ignored.

  ```
  <filter type="subtree">
   <top xmlns="urn:huawei:yang:example">
    <users/>
   </top>
  </filter>
  ```

  In this example, the <top> node is a containment node, and the <users> node is a selection node. The <users> node can be included for filter output only when the <users> node complies with namespace **urn:huawei:yang:example** and is contained in the <top> element in the root directory of the configuration database.

Subtree Filter Processing

First, the subtree filter output is set as empty. Each subtree filter can contain one or more data model segments, each of which represents one of the selected output parts of the selected data model. Each subtree data segment is composed of data models supported by the NETCONF server. If the entire subtree data segment completely matches part of the data models supported by the NETCONF server, all nodes and child nodes of the subtree data segment are selected and output to the query result.

- If no filter is used, all data in the current data model is returned in the query result.

  RPC request

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

  RPC reply

  ```
  <rpc-reply message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
    <!-- ... entire set of data returned ... -->
   </data>
  </rpc-reply>
  ```

- If an empty filter is used, the query result contains no data output, in that no content match or selection node is specified.

  RPC request

  ```
  <rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get>
    <filter type="subtree">
    </filter>
   </get>
  </rpc>
  ```

  RPC reply

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

- Multi-subtree filtering

  The following example uses the **root**, **fred**, and **barney** subtree filters.

  The **root** subtree filter contains two containment nodes (<users> and <user>), one content match node (<name>), and one selection node (<company-info>). As for subtrees that meet selection criteria, only <company-info> is selected.

  The **fred** subtree filter contains three containment nodes (<users>, <user>, and <company-info>), one content match node (<name>), and one selection node (<id>). As for subtrees that meet the selection criteria, only the <id> element in <company-info> is selected.

  The **barney** subtree filter contains three containment nodes (<users>, <user>, and <company-info>), two content match nodes (<name> and <type>), and one selection node (<dept>). User **barney** is not of the userbarney type and does not comply with the subtree filtering rule. Therefore, the entire subtree of **barney** (including its parent node <user>) is not selected.

RPC request

```
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <top xmlns="urn:huawei:yang:example">
    <users>
     <user>
      <name>root</name>
      <company-info/>
     </user>
     <user>
      <name>fred</name>
      <company-info>
       <id/>
      </company-info>
     </user>
     <user>
      <name>barney</name>
      <type>userbarney</type>
      <company-info>
       <dept/>
      </company-info>
     </user>
    </users>
   </top>
  </filter>
 </get-config>
</rpc>
```

RPC reply

```
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <top xmlns="urn:huawei:yang:example">
   <users>
    <user>
     <name>root</name>
     <company-info>
      <dept>1</dept>
      <id>1</id>
     </company-info>
    </user>
    <user>
     <name>fred</name>
     <company-info>
      <id>2</id>
     </company-info>
    </user>
   </users>
  </top>
 </data>
</rpc-reply>
```

3.7.3 Configuration Precautions for NETCONF

Licensing Requirements

NETCONF is not under license control.

Hardware Requirements

Table 3-32 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.7.4 NETCONF Operation Capabilities (YANG)

NETCONF provides a set of basic operations for you to manage device configurations and retrieve device configuration and status data. NETCONF also provides additional operations based on the capabilities advertised by a device.

3.7.4.1 Basic NETCONF Operations (YANG)

3.7.4.1.1 get-config

The <get-config> operation queries all or specified configuration data sets.

- **source**: specifies a configuration database in which configuration data is being queried. The value can be <running/>, <candidate/>, or <startup/>.
- **filter**: specifies a range to be queried in the configuration database. If this parameter is not specified, all configurations on the device are returned.

The following example shows how to query interface configuration of the IFM feature in the <running/> database. The queried interface information is contained in an RPC reply message.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="827">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <ifm:ifm xmlns:ifm="urn:huawei:yang:huawei-ifm">
    <ifm:interfaces>
     <ifm:interface/>
    </ifm:interfaces>
   </ifm:ifm>
  </filter>
 </get-config>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```



```
 <ifm xmlns="urn:huawei:yang:huawei-ifm">
  <interfaces>
   <interface>
    <name>GE0/0/1</name>


   </interface>
  </interfaces>
 </ifm>
</data>
```

3.7.4.1.2 get-data

The <get-data> operation can be used to retrieve all or specified configuration or status data sets from the NMDA data set.

- source: indicates the name of the database being queried. If the database name is <ietf-datastores:running/>, <ietf-datastores:candidate/>, or <ietf-datastores:startup/>, the configuration data is being queried. If the database name is <ietf-datastores:operational/>, the running configuration and status data of the device is being queried.
- xpath-filter: uses an XPath to specify the range of the configuration database to be queried. If this parameter is not specified, all configurations on the device are returned.
- subtree-filter: uses a subtree to specify the range of the configuration database to be queried. If this parameter is not specified, all configurations on the device are returned.

The following example shows how to query the task group configuration of the AAA feature in the <ietf-datastores:running/> database. The queried group information is returned in an RPC reply message.

Use an XPath to specify the range of the configuration database to be queried.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
  <datastore>ds:running</datastore>
  <subtree-filter>
   <ifm:ifm xmlns:ifm="urn:huawei:yang:huawei-ifm">
    <ifm:interfaces>
     <ifm:interface/>
    </ifm:interfaces>
   </ifm:ifm>
  </subtree-filter>
 </get-data>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
   <ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
     <interface>
      <name>GE0/0/1</name>
     </interface>
    </interfaces>
   </ifm>
  </data>
</rpc-reply>
```


Use a subtree to specify the range of the configuration database to be queried.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
   <datastore>ds:running</datastore>
   <xpath-filter xmlns:aaa="urn:huawei:yang:huawei-ifm">/ifm:ifm/ifm:interfaces/ifm:interface</
xpath-filter>
 </get-data>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">
    <ifm xmlns="urn:huawei:yang:huawei-ifm">
      <interfaces>
        <interface>
          <name>GE0/0/1</name>
        </interface>
      </interfaces>
    </ifm>
  </data>
</rpc-reply>
```

3.7.4.1.3 get

The <get> operation only retrieves data from the <running/> configuration database.

If the <get> operation is successful, the NETCONF server returns an <rpc-reply> element containing a <data> element with the results of the query. If the operation fails, the server returns an <rpc-reply> element containing an <rpc-error> element.

NOTE

The differences between <get> and <get-config> operations are as follows:

- The <get-config> operation can retrieve data from the <running/>, <candidate/>, and <startup/> configuration databases, whereas the <get> operation can only retrieve data from the <running/> configuration database.
- The <get-config> operation can only retrieve configuration data, whereas the <get> operation can retrieve both configuration and status data.

The following example shows how to query interface configuration of the IFM feature in the database. The queried interface information is contained in an RPC reply message.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="831">
 <get>
  <filter type="subtree">
   <ifm:ifm xmlns:ifm="urn:huawei:yang:huawei-ifm">
    <ifm:interfaces>
     <ifm:interface/>
    </ifm:interfaces>
   </ifm:ifm>
  </filter>
 </get>
</rpc>
```


- RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ifm xmlns="urn:huawei:yang:huawei-ifm">
  <interfaces>
   <interface>
    <name>GE0/0/1</name>
    <admin-status>up</admin-status>
    <mtu>1500</mtu>
    <vrf-name>_public_</vrf-name>
    <mac-address>00e0-fc12-3456</mac-address>
    <index>1001</index>
    <is-l2-switch>false</is-l2-switch>
    <dynamic>
     <oper-status>up</oper-status>
     <physical-status>down</physical-status>
     <link-status>down</link-status>
     <mtu>1500</mtu>
     <bandwidth>0</bandwidth>
     <ipv4-status>down</ipv4-status>
     <ipv6-status>down</ipv6-status>
     <is-control-flap-damp>false</is-control-flap-damp>
     <mac-address>00e0-fc12-3456</mac-address>
     <is-offline>true</is-offline>
     <link-quality-grade>good</link-quality-grade>
     <sub-if-counts>0</sub-if-counts>
    </dynamic>
    <mib-statistics>
     <receive-byte>0</receive-byte>
     <send-byte>0</send-byte>
     <receive-packet>0</receive-packet>
     <send-packet>0</send-packet>
     <receive-unicast-packet>0</receive-unicast-packet>
     <receive-multicast-packet>0</receive-multicast-packet>
     <receive-broad-packet>0</receive-broad-packet>
     <send-unicast-packet>0</send-unicast-packet>
     <send-multicast-packet>0</send-multicast-packet>
     <send-broad-packet>0</send-broad-packet>
     <receive-error-packet>0</receive-error-packet>
     <receive-drop-packet>0</receive-drop-packet>
     <send-error-packet>0</send-error-packet>
     <send-drop-packet>0</send-drop-packet>
    </mib-statistics>
    <arp-entry xmlns="urn:huawei:yang:huawei-arp">
     <expire-time>1200</expire-time>
     <arp-learn-disable>false</arp-learn-disable>
     <route-proxy-enable>false</route-proxy-enable>
     <dest-mac-check>false</dest-mac-check>
     <src-mac-check>false</src-mac-check>
    </arp-entry>
    <vlanif-attribute xmlns="urn:huawei:yang:huawei-vlan">
     <band-width>1000</band-width>
     <damping-time>0</damping-time>
    </vlanif-attribute>
    <ethernet xmlns="urn:huawei:yang:huawei-ethernet">
     <main-interface>
      <vlan-swap>disable</vlan-swap>
      <qinq-protocol>0x8100</qinq-protocol>
     </main-interface>
     <l2-sub-interface>
      <local-switch>disable</local-switch>
     </l2-sub-interface>
    </ethernet>
    <bdif-attribute xmlns="urn:huawei:yang:huawei-bd">
     <band-width>1000</band-width>
     <damping-time>0</damping-time>
    </bdif-attribute>
    <cellular xmlns="urn:huawei:yang:huawei-cellular">
     <dialer-enable>
```



```
        <ip-address-alloc>false</ip-address-alloc>
       </dialer-enable>
       <modem-event>
        <dial-failed>
         <fail-times>6</fail-times>
         <action>modem-reboot</action>
         <retry-times>3</retry-times>
        </dial-failed>
       </modem-event>
      </cellular>
      <dhcp-client-if xmlns="urn:huawei:yang:huawei-dhcp">
       <bootp-alloc>
        <enable>false</enable>
        <unicast>false</unicast>
       </bootp-alloc>
      </dhcp-client-if>
     </interface>
    </interfaces>
   </ifm>
</data>
```

3.7.4.1.4 edit-config

The <edit-config> operation loads all or some configurations to a specified target configuration database (<running/> or <candidate/>).

The <edit-config> operation supports multiple modes for loading configurations. If a NETCONF server supports the URL capability, the <url> parameter (which identifies a local configuration file) can be used to replace the <config> parameter.

Parameters in an RPC message of the <edit-config> operation are described as follows:

- <config>: indicates a group of hierarchical configuration items defined in the data model.

  The <config> parameter may contain the optional operation attribute, which is used to specify an operation type for a configuration item. If the operation attribute is not present, the <merge> operation is performed by default. The values of the operation attribute are as follows:

  - **merge**: modifies or creates data in the database. This is the default operation.
  - **create**: adds configuration data to the configuration database only if such data does not exist.
  - **delete**: deletes a specified configuration data record from the configuration database.
  - **remove**: removes a specified configuration data record from the configuration database. If the data exists, it is deleted. If the data does not exist, a success message is returned.
  - **replace**: replaces existing data or creates data that does not exist in the database.
- <target>: indicates the configuration database to be edited. The configuration database can be set based on scenarios.
  - In immediate validation mode, set the database to <running/>.
  - In two-phase validation mode, set the database to <candidate/>. After editing the database, perform the <commit> operation so that the modified configuration takes effect.

- <default-operation>: sets a default operation for the <edit-config> operation.

  The <default-operation> parameter is optional. Its values are as follows:

  – **merge**: merges the configuration data in the <config> parameter with that in the target configuration database. This is the default operation.

  – **replace**: completely replaces the configuration data in the target configuration database with the configuration data in the <config> parameter.

  – **none**: The target configuration database is not affected by the configuration in the <config> parameter, unless and until the incoming configuration data uses the operation attribute to request a different operation. If the <config> parameter contains configuration data that does not exist at the corresponding data level in the target configuration database, <rpc-error> is returned, in which the <error-tag> value is **data-missing**. This prevents redundant elements from being created when a specified operation is performed. For example, in the condition that <config> contains the parent hierarchical structure of a child element to be deleted but the target database does not contain the configuration of the parent element, if the value of the <default-operation> parameter is not **none**, the configuration of the parent element is created in the database when the child element is deleted; if the <default-operation> parameter is **none**, only the child element is deleted, and the configuration of the parent element is not created.

- <error-option>: sets a mode for processing subsequent instance configurations if an error occurs in the current instance configuration. Its values are as follows (the default value is **rollback-on-error**):

  – If the target configuration database is <running/>:

    ▪ **continue-on-error**: records the error information and continues the execution after an error occurs. If an error occurs, the NETCONF server returns an RPC reply message to the client, indicating an operation failure.

    ▪ **rollback-on-error**: stops the operation after an error occurs and rolls back the configuration to the state before the <edit-config> operation is performed. This operation is supported only when the device supports the rollback-on-error capability.

  – If the target configuration database is <candidate/>, set the value of <error-option> to **rollback-on-error** for subsequent instances after an error occurs in the current instance configuration.

  The following example shows how to change the **admin-status** value of GE0/0/1 in the running configuration database to **up**.

  – RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="15">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
   <ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
     <interface>
      <name>GE0/0/1</name>
```



```
          <admin-status>up</admin-status>
        </interface>
      </interfaces>
    </ifm>
  </config>
 </edit-config>
</rpc>
```

– RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
         message-id="15">
 <ok/>
</rpc-reply>
```

The following example shows how to remove the configuration on interface GE0/0/1 from the running configuration database.

– RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="844">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
   <ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
     <interface xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="delete">
      <name>GE0/0/1</name>
     </interface>
    </interfaces>
   </ifm>
  </config>
 </edit-config>
</rpc>
```

– RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
         message-id="15">
 <ok/>
</rpc-reply>
```

3.7.4.1.5 edit-data

For the NMDA data set, the <edit-data> operation can be used to load all or some configuration data to a specified target configuration database (<ietf-datastores:running/> or <ietf-datastores:candidate/>). A device authorizes the operation in <edit-data>. After the authorization succeeds, the device performs corresponding modification.

The <edit-data> operation supports multiple modes for loading configurations. If a NETCONF server supports the URL capability, the <url> parameter (which identifies a local configuration file) can be used to replace the <config> parameter.

Parameters in an RPC message of the <edit-data> operation are described as follows:

- <config>: indicates a group of hierarchical configuration items defined in the data model.

  The <config> parameter may contain the optional operation attribute, which is used to specify an operation type for a configuration item. If the operation attribute is not present, the <merge> operation is performed by default. The values of the operation attribute are as follows:

  - **merge**: modifies or creates data in the database. This is the default operation.
  - **create**: adds configuration data to the configuration database only if such data does not exist. If such configuration data exists, <rpc-error> is returned.
  - **delete**: deletes a specified configuration data record from the configuration database. If the data exists, it is deleted. If the data does not exist, <rpc-error> is returned.
  - **remove**: removes a specified configuration data record from the configuration database. If the data exists, it is deleted. If the data does not exist, a success message is returned.
  - **replace**: replaces configuration data records in the configuration database. If the data exists, all relevant data is replaced. If the data does not exist, the data is created. Different from the <copy-config> operation (which completely replaces the configuration data in the target configuration database), this operation affects only the configuration that exists in the <config> parameter.
- **target**: indicates the configuration database to be edited. The configuration database can be set based on scenarios.
  - In immediate validation mode, set the database to <ietf-datastores:running/>.
  - In two-phase validation mode, set the database to <ietf-datastores:candidate/>. After editing the database, perform the <commit> operation to submit the configuration for the modification to take effect.
- **default-operation**: sets a default operation for the <edit-data> operation.

  The <default-operation> parameter is optional. Its values are as follows:
  - **merge**: merges the configuration data in the <config> parameter with that in the target configuration database. This is the default operation.
  - **replace**: completely replaces the configuration data in the target configuration database with the configuration data in the <config> parameter.
  - **none**: The target configuration database is not affected by the configuration in the <config> parameter, unless and until the incoming configuration data uses the operation attribute to request a different operation. If the <config> parameter contains configuration data that does not exist at the corresponding data level in the target configuration database, <rpc-error> is returned, in which the <error-tag> value is **data-missing**. This prevents redundant elements from being created when a specified operation is performed. For example, when a specified child element is deleted and <config> contains the parent hierarchical structure of the child element but the target database does not contain the configuration of the parent element, if the value of the <default-operation> parameter is not **none**, the configuration of the parent element is created in the database when the child element is deleted. If the <default-operation> parameter is set to **none**, only the child element is deleted, and the configuration of the parent element is not created.

The following example shows how to change the description of GE0/0/1 in the <ietf-datastores:running/> configuration database to **huawei**.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
 <edit-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
  <datastore>ds:running</datastore>
  <config>
   <ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
     <interface>
      <name>GE0/0/1</name>
     </interface>
    </interfaces>
   </ifm>
  </config>
 </edit-data>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           message-id="5">
 <ok/>
</rpc-reply>
```

The following example shows how to remove the configuration on interface GE0/0/1 from the running configuration database.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
 <edit-data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
  <datastore>ds:running</datastore>
  <config>
   <ifm xmlns="urn:huawei:yang:huawei-ifm">
    <interfaces>
     <interface xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="delete">
      <name>GE0/0/1</name>
     </interface>
    </interfaces>
   </ifm>
  </config>
 </edit-data>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           message-id="5">
 <ok/>
</rpc-reply>
```

3.7.4.1.6 copy-config

The <copy-config> operation replaces the target configuration database with the source configuration database. The target database is overwritten if it exists, or a new one is created, if allowed.

- The configuration data in the <candidate/> and <running/> databases can be saved to a specified URL file.
- The configuration data in the <running/> database can be copied to the <candidate/> or <startup/> database.
- The configuration data in a specified URL file can replace that in the <candidate/> or <startup/> configuration database.

NOTE

The size of the specified file must be less than 1 MB. If the size exceeds 1 MB, "The configuration is too large." is displayed.

The following example shows how to save the configuration data in the <running/> database to the local **abc.xml** file.

- RPC request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <target>
    <url>file:///abc.xml</url>
  </target>
  <source>
    <running/>
  </source>
 </copy-config>
</rpc>
```

- RPC reply

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

3.7.4.1.7 delete-config

This operation deletes a configuration database. The <running/> configuration database cannot be deleted.

If the <delete-config> operation is successful, the server sends an <rpc-reply> element containing an <ok> element. Otherwise, the server sends an <rpc-reply> element containing an <rpc-error> element.

The following example shows how to delete the <startup/> database.

- RPC request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <delete-config>
    <target>
      <startup/>
    </target>
  </delete-config>
</rpc>
```

- RPC reply

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

3.7.4.1.8 lock

This operation locks a configuration database of a device, preventing it from being modified by other users. The lock operation prevents configuration conflicts with other sessions.

If the configuration database is already locked by an authorized user, the <error-tag> element will be displayed as **lock-denied** and the <error-info> element will include <session-id> of the lock owner in the reply message.

The following example shows a successful locking of the <running/> configuration database.

- RPC request

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

- RPC reply

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

If the NMDA data set is supported, the data set format in the target configuration database is different, as shown in the following:

- RPC request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <lock xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
  <target>
   <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">ds:running</datastore>
  </target>
 </lock>
</rpc>
```

- RPC reply

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

The following example shows a failed locking of the <running/> configuration database.

- RPC request

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

- RPC reply

```
<rpc-reply message-id="101" 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-app-tag>43</error-app-tag>
  <error-message>The configuration is locked by other user. [Session ID = 629] </error-message>
  <error-info>
   <session-id>629</session-id>
   <error-paras>
    <error-para>629</error-para>
   </error-paras>
  </error-info>
 </rpc-error>
</rpc-reply>
```

If the NMDA data set is supported, the data set format in the target configuration database is different, as shown in the following:

- RPC request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <lock xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
```

```
  <target>
    <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">ds:running</datastore>
  </target>
 </lock>
</rpc>
```

- RPC reply

```
<rpc-reply message-id="101" 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-app-tag>43</error-app-tag>
  <error-message>The configuration is locked by other user. [Session ID = 629] </error-message>
  <error-info>
   <session-id>629</session-id>
   <error-paras>
    <error-para>629</error-para>
   </error-paras>
  </error-info>
 </rpc-error>
</rpc-reply>
```

3.7.4.1.9 unlock

This operation cancels the <lock> operation performed by the specified user, rather than the <lock> operation performed by other users.

If the <unlock> operation is successful, the server sends an <rpc-reply> element containing an <ok> element. Otherwise, the server sends an <rpc-reply> element containing an <rpc-error> element.

The following example shows how to unlock the <running/> database.

- RPC request

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

- RPC reply

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

If the NMDA data set is supported, the data set format in the target configuration database is different, as shown in the following:

- RPC request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <unlock xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">
  <target>
   <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-nmda">ds:running</datastore>
  </target>
 </unlock>
</rpc>
```

- RPC reply

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


3.7.4.1.10 close-session

This operation terminates a NETCONF session.

After receiving a <close-session> request, the NETCONF server terminates the current NETCONF session. The server releases all locks and resources associated with the session. After receiving a <close-session> request, the NETCONF server ignores all request messages of the session.

The following example shows how to terminate a NETCONF session.

- RPC request

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

- RPC reply

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

3.7.4.1.11 kill-session

The <kill-session> operation forcibly terminates a NETCONF session. Only an administrator is authorized to perform this operation.

After receiving a <kill-session> request, the NETCONF server stops all operations that are being performed for the session, releases all the locks and resources associated with the session, and terminates the session.

If the NETCONF server receives a <kill-session> request when performing the <commit> operation, it must restore the configuration to the state before the configuration is committed.

The following example shows how to forcibly terminate the NETCONF session with the session ID of 4.

- ```
  <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <kill-session>
      <session-id>4</session-id>
    </kill-session>
  </rpc>
  ```

- RPC reply

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

3.7.4.2 NETCONF Standard Capability Set (YANG)

3.7.4.2.1 Writable-running

This capability indicates that the device supports direct writes to the <running/> configuration database. Specifically, the device supports <edit-config> and <copy-config> operations on the <running/> configuration database.

- RPC request

```
<?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>
    <ifm xmlns="urn:huawei:yang:huawei-ifm">
      <interfaces>
        <interface>
          <name>GE0/0/1</name>
        </interface>
      </interfaces>
    </ifm>
  </config>
 </edit-config>
</rpc>
```

- RPC reply

```
<?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>
```

3.7.4.2.2 Candidate Configuration

This capability indicates that the device supports the <candidate/> configuration database.

The <candidate/> configuration database holds a complete set of configuration data that can be manipulated without impacting the device's current configuration. This configuration database serves as a work place for manipulating configuration data.

Additions, deletions, and changes can be made to the data in the <candidate/> configuration database to construct the desired configuration data. The following operations can be performed at any time:

- <commit>: converts all configuration data in the <candidate/> configuration database into running configuration data.

  If the <commit> operation fails, the content in the <candidate/> configuration database remains unchanged.
- <discard-changes>: discards uncommitted configuration data in the <candidate/> configuration database. After this operation is performed, the configuration data in the <candidate/> configuration database is the same as that in the <running/> configuration database again.

A device has only one <candidate/> configuration database, which is shared by different NETCONF sessions.

- RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
    <candidate/>
  </target>
  <config>
    <ifm xmlns="urn:huawei:yang:huawei-ifm">
      <interfaces>
        <interface>
          <name>GE0/0/1</name>
        </interface>
      </interfaces>
    </ifm>
  </config>
```



```
  </edit-config>
</rpc>
```

- RPC reply

```
<?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>
```

3.7.4.2.3 Rollback on Error

This capability allows the device to perform a rollback when an error occurs. Specifically, "rollback-on-error" can be carried in the <error-option> parameter of the <edit-config> operation. If an error occurs and the <rpc-error> element is generated, the server stops performing the <edit-config> operation and restores the specified configuration to the state before the <edit-config> operation is performed.

This capability is valid only when the device supports the candidate configuration capability.

- RPC request

```
<?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>
    <error-option>rollback-on-error</error-option>
    <config>
      <ifm xmlns="urn:huawei:yang:huawei-ifm">
        <interfaces>
          <interface>
            <name>GE0/0/1</name>
          </interface>
        </interfaces>
      </ifm>
    </config>
  </edit-config>
</rpc>
```

- RPC reply

```
<?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>
```

3.7.4.2.4 Distinct Startup

This capability indicates that the device can perform an independent startup. Specifically, the device can distinguish the <running/> configuration database from the <startup/> configuration database.

The NETCONF server needs to independently maintain the running configuration and restore the configuration after the device restarts. Because the configuration data of the <running/> configuration database is not automatically synchronized to the <startup/> configuration database, you must copy the data from the <running/> configuration database to the <startup/> configuration database by using an operation such as a <copy-config>.

The following is an example of executing the <copy-config> operation to copy the data from the <running/> database to the <startup/> database.

- RPC request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <source>
   <running/>
  </source>
  <target>
   <startup/>
  </target>
 </copy-config>
</rpc>
```

- RPC reply

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

3.7.4.2.5 URL

This capability indicates that the device can modify or copy files in a specified path. Currently, the <edit-config> and <copy-config> operations are supported. Password information in URLs is protected. When configuration data is exported, password information is exported in ciphertext.

- <edit-config>: submits the configuration file in a specified path to <candidate/> or <running/>.
- <copy-config>: copies data in <candidate/> or <running/> to a file in a specified path.

NOTE

For the <copy-config> operation, if the file specified in the <url> element does not exist, the file is directly created. If the file exists, it is overwritten.

When you perform the <edit-config> operation, the file specified in <url> must exist.

The following example shows how to copy the data in the <running/> configuration database to the local **abc.xml** file.

- RPC request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <target>
   <url>file:///abc.xml</url>
  </target>
  <source>
   <running/>
  </source>
 </copy-config>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
 <ok/>
</rpc>
```

The following example shows how to commit the content of the local **config.xml** file to the <candidate/> database.

- RPC request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
```



```
<edit-config>
 <target>
  <candidate/>
 </target>
 <url>file://config.xml</url>
</edit-config>
</rpc>
```

- RPC reply

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
<ok/>
</rpc>
```

3.7.4.2.6 Notification Capabilities

Notification 1.0

A device can send alarms and events to a client using the NETCONF notification capability, thereby allowing the client to promptly detect device configuration or other changes. You can perform the <create-subscription> operation to subscribe to device alarms and events. If the <rpc-reply> element returned by the device contains an <ok> element, the <create-subscription> operation is successful. In this case, the device will proactively report the generated alarms and events to the client through NETCONF.

1. Alarms and events can be subscribed to in either of the following modes: long-term subscription and subscription within a specified period.
   - Long-term subscription: After the subscription is successful, if the <startTime> element is specified in the subscription message, the device sends historical alarms and events to the NMS and then sends a <replayComplete> message to notify the NMS that the replay is complete. If a new alarm or event is generated, the device also sends it to the NMS. If the <startTime> element is not specified in the subscription message, the device sends all newly generated alarms and events to the NMS. After a NETCONF session is terminated, the subscription is automatically canceled.
   - Subscription within a specified period: After the subscription is successful, the device sends the alarms and events that are generated during the specified period and that meet the filtering conditions to the NMS. Because the <startTime> element is specified in the subscription message, the device sends historical alarms and events to the NMS and then sends a <replayComplete> message to notify the NMS that the replay is complete. When the specified <stopTime> has been reached, the NETCONF module sends a <notificationComplete> message to notify the NMS that the subscription is terminated.

   Historical alarms and events refer to those generated from the <startTime> specified in the subscription message to when the user performs the subscription operation. If <stopTime> is not specified, the subscription is a long-term one. If both <startTime> and <stopTime> are specified, the subscription is within a specified period. The format of the subscription message sent by the device to the NMS is as follows:

   RPC request (NETCONF subscription)

```
<rpc message-id="101" 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>
```



```
    <startTime>2016-10-20T14:50:00Z</startTime>
    <stopTime>2016-10-23T06:22:04Z</stopTime>
  </create-subscription>
</rpc>
```

RPC reply (NETCONF subscription)

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

RPC request (NETCONF-WITH-RES-CONFIG subscription)

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
  <create-subscription xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <stream>NETCONF-WITH-RES-CONFIG</stream>
  </create-subscription>
</rpc>
```

RPC reply (NETCONF-WITH-RES-CONFIG subscription)

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="5">
  <ok/>
</rpc-reply>
```

Example of reporting a notification (NETCONF-WITH-RES-CONFIG subscription)

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2020-12-09T20:41:14Z</eventTime>
  <alt-resource-config xmlns="urn:huawei:yang:huawei-notification-common">
    <notification-id>135598331</notification-id>
    <notification-class>event</notification-class>
    <event-level>informational</event-level>
  </alt-resource-config>
  <netconf-config-change xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-notifications">
    <changed-by>
      <server/>
    </changed-by>
    <datastore>running</datastore>
  </netconf-config-change>
</notification>
```


Table 3-33 Element descriptions

Element	Description	Value Range	Mandatory	Constraints
stream	Event flow type	The value is a case-sensitive enumerated type and can be: • NETCONF: indicates that the NETCONF notification mechanism is used to report alarms and events. • NETCONF-WITH-RES-CONFIG: indicates that the reported alarms or events carry Huawei's proprietary extension attribute huawei-notification-common.	No	N/A
startTime	Start time	The value is in the time format.	No	The start time must be earlier than the time when the subscription operation is performed.

Element	Description	Value Range	Mandatory	Constraints
stopTime	End time	The value is in the time format.	No	The end time must be later than the start time.

2. After the subscription is successful, the device encapsulates the alarm or event information into notification messages and sends them to the NMS.

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>2016-11-26T13:51:00Z</eventTime>
 <hwCPUUtilizationResume xmlns="urn:huawei:yang:huawei-sem">
 <ProbableCause>0</ProbableCause>
 <PhysicalIndex>0</PhysicalIndex>
 <PhysicalName>SimulateStringData</PhysicalName>
 <RelativeResource>SimulateStringData</RelativeResource>
 <UsageType>0</UsageType>
 <SubIndex>0</SubIndex>
 <CpuUsage>0</CpuUsage>
 <CpuUsageThreshold>0</CpuUsageThreshold>
 </hwCPUUtilizationResume>
</notification>
```

3. After alarms and events are reported to the NMS, the NETCONF module sends a subscription completion message to the NMS.
   - After historical alarms and events are reported to the NMS, the NETCONF module sends a replayComplete message to the NMS.

     ```
     <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
       <eventTime>2016-11-29T11:57:15Z</eventTime>
       <replayComplete xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0" />
     </notification>
     ```

   - When <stopTime> specified in the subscription message has been reached, the NETCONF module sends a notificationComplete message to notify the NMS that the subscription is terminated.

     ```
     <notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
       <eventTime>2016-11-29T11:57:25Z</eventTime>
       <notificationComplete xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0" />
     </notification>
     ```

Table 3-34 Element descriptions

Element	Description	Value Range	Mandatory	Constraints
replayComplete	After historical alarms and events are reported to the NMS, the NETCONF module sends a replayComplete message to the NMS.	N/A	No	N/A

Element	Description	Value Range	Mandatory	Constraints
notificationComplete	When <stopTime> specified in the subscription message has been reached, the NETCONF module sends a notificationComplete message to notify the NMS that the subscription is terminated.	N/A	No	N/A

3.7.4.2.7 YANG-library

This capability indicates that a device is capable of providing information about supported YANG modules. Basic information about YANG modules that a server supports can be viewed on a NETCONF client. The information includes the module name, YANG model version, namespace, list of submodules, and resource file path, and is saved in the local buffer.

XML example: Query basic information about the YANG module named **huawei-aaa**.

RPC request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="23">
  <get>
    <filter type="subtree">
      <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
        <module-set>
          <name />
          <module>
            <name>huawei-aaa</name>
          </module>
        </module-set>
      </yang-library>
    </filter>
  </get>
</rpc>
```

RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <yang-library xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-library">
    <module-set>
```



```
    <name>complete</name>
    <module>
     <name>huawei-aaa</name>
     <revision>2022-04-02</revision>
     <namespace>urn:huawei:yang:huawei-aaa</namespace>
     <location>file:///usr/local/share/omu-res/yang/huawei-aaa.yang</location>
     <submodule>
      <name>huawei-aaa-lam</name>
      <revision>2022-03-31</revision>
      <location>file:///respath/huawei-aaa-lam.yang</location>
     </submodule>
     <deviation>huawei-aaa-deviations-SOHO</deviation>
    </module>
   </module-set>
  </yang-library>
</data>
```

Information contained in the reply includes the YANG model version used by the **huawei-aaa** module, namespace, list of submodules, revision time, and resource file path.

3.7.4.3 NETCONF Extended Capability Set (YANG)

3.7.4.3.1 With-defaults

The <with-defaults> capability indicates that a device can process default values of the model. The <get>, <get-config>, and <copy-config> operations can carry the <with-defaults> parameter.

The available options of the <with-defaults> parameter are as follows:

- **report-all**: queries all nodes and does not perform any operation on the nodes.
  - RPC request

    ```
    <?xml version="1.0" encoding="utf-8"?>
    <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="4">
     <get xmlns:wsss="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">
      <filter type="subtree">
       <system xmlns="urn:huawei:yang:huawei-system"/>
      </filter>
      <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all</with-
    defaults>
     </get>
    </rpc>
    ```

  - RPC reply

    ```
    <?xml version="1.0" encoding="utf-8"?>
    <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="4">
     <data>
      <system xmlns="urn:huawei:yang:huawei-system">
       <system-info>
        <sys-name>example</sys-name>
        <sys-contact>R&amp;amp;D Beijing, Huawei Technologies co.,Ltd.</sys-contact>
        <sys-location>Beijing China</sys-location>
       </system-info>
      </system>
     </data>
    </rpc-reply>
    ```

- trim: trims the nodes whose values equal the default ones from the query results.
  - RPC request

    ```
    <?xml version="1.0" encoding="utf-8"?>
    <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
    ```



```
<get xmlns:wsss="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">
 <filter type="subtree">
  <system xmlns="urn:huawei:yang:huawei-system"/>
 </filter>
 <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">trim</with-
defaults>
 </get>
</rpc>
```

– RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="4">
 <data>
  <system xmlns="urn:huawei:yang:huawei-system">
   <system-info>
    <sys-name>example</sys-name>
   </system-info>
  </system>
 </data>
</rpc-reply>
```

- report-all-tagged: queries all nodes and uses namespace:default="true" to identify the nodes whose values equal the default ones.
  – RPC request

```
<?xml version="1.0" encoding="utf-8"?>
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
 <get xmlns:wsss="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">
  <filter type="subtree">
   <system xmlns="urn:huawei:yang:huawei-system"/>
  </filter>
  <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all-
tagged</with-defaults>
 </get>
</rpc>
```

  – RPC reply

```
<?xml version="1.0" encoding="utf-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns:wd="urn:ietf:params:xml:ns:netconf:default:1.0"
          message-id="2">
 <data>
  <system xmlns="urn:huawei:yang:huawei-system">
   <system-info>
    <sys-name>example</sys-name>
    <sys-contact wd:default="true">R&amp;amp;D Beijing, Huawei Technologies co.,Ltd.</sys-contact>
    <sys-location wd:default="true">Beijing China</sys-location>
   </system-info>
  </system>
 </data>
</rpc-reply>
```

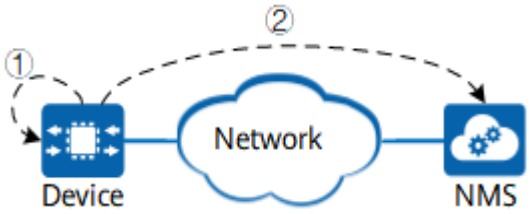
3.7.5 Establishing a NETCONF Session

3.7.5.1 Understanding How to Establish a NETCONF Session

The device functions as the NETCONF server, and the NMS functions as the NETCONF client.

A device can proactively establish a NETCONF session with an NMS after proactive NETCONF registration is enabled.

Figure 3-22 Establishing a NETCONF session between a device and an NMS by enabling proactive NETCONF registration



Enable proactive NETCONF registration:

1. Configure a user and enable proactive NETCONF registration on the device.
2. The device proactively establishes a NETCONF session with the NMS.

3.7.5.2 Configuring an SSH User

Context

Configuring an SSH user includes creating an SSH user and configuring a PKI domain for the SSH user.

NOTE

The SSH username in the current configuration must be **huawei**.

Procedure

Step 1 Enter the editing view.

```
edit-config
```

Step 2 Create an SSH user.

```
sshs users user name user-name
```

Step 3 Configure a PKI domain for the SSH user.

```
pub-key-type PKI key-name key-name
```

pub-key-type specifies the public key type. Currently, only PKI is supported. **key-name** specifies the name of the PKI domain bound to the user.

Step 4 Commit the configuration.

```
commit
```

----End

3.7.5.3 Establishing a NETCONF Connection Between a Device and an NMS

Context

Proactive NETCONF registration is required for a device to establish a NETCONF connection with an NMS. With this function, an on-board device proactively sends a NETCONF connection request to the NMS and establishes a NETCONF connection with the NMS. This enables users to manage the device through the NMS in a timely manner.

Procedure

- Establish a NETCONF connection between a device and an NMS by enabling proactive NETCONF registration. If the address and port number of the NMS are not configured, the system attempts to obtain the address and port number of the NMS from the registration center. If the address and port number are obtained successfully, the system writes them into the configuration and re-establishes a connection with the NMS.
  a. Enter the editing mode.
  edit-config
  b. Enter the SSHS view.
  sshs
  c. Create a callhome template and enter the callhome view.
  call-homes call-home call-home-name *name*
  d. Create a NETCONF connection instance and enter the NETCONF connection instance view.
  end-points end-point end-point-name *name*
  e. Configure an NMS address that is used to establish a NETCONF connection with the device.
  { **address** *ip-address* | **host-name** *host-domain-name* }
  f. Configure a TCP port number that is used to establish a NETCONF connection with the device.
  port *port-number*
  g. Commit the configuration.
  commit
  h. Exit the editing view.
  return
- Disable the connection to the NMS.
  a. Enter the editing mode.
  edit-config
  b. Enter the SSHS view.
  sshs
  c. Create a callhome template and enter the callhome view.
  call-homes call-home call-home-name *name*
  d. Create a NETCONF connection instance and enter the NETCONF connection instance view.
  end-points end-point end-point-name *name*
  e. Disconnect the device from the NMS.
  enabled false
  f. Commit the configuration.
  commit
  g. Exit the editing view.
  return

----End

Follow-up Procedure

Run the **display sshs/call-homes/call-home[call-home-name=*callhome-name*]/end-points/end-point[end-point-name=*endpoint-name*]/connection-status** command to check the status of the connection between the device and controller.

When a device goes online, you can perform the following operations to check the registration status and logout reason of the device:

- Run the **display system-controller/register-fail-records/register-fail-record** command to check the device registration failure records and determine the cause of the failure.
- Run the **display system-controller/offline-records/offline-record** command to check the reason why the device goes offline from the NMS.
- After the connection is established, run the **display system/system-info/upstream-info** command to check the uplink interface connected to the Qiankun cloud.

3.7.5.4 (Optional) Configuring NETCONF Reliability

Context

If a device has been managed by Qiankun Cloud and some configurations are modified, the device may go offline. In this scenario, you can configure NETCONF robustness. After a device goes online and remains stable for a period of time, the device proactively saves the current configuration. If the device goes offline due to incorrect configuration, the device rolls back the configuration and goes online again.

Procedure

Step 1 Enter the editing mode.

```
edit-config
```

Step 2 Enter the NETCONF robustness configuration node.

```
system-controller online-robustness-enhancement
```

Step 3 Enable the NETCONF robustness function.

```
enabled true
```

Step 4 (Optional) Configure the steady-state duration of the device.

```
stable-time stable-time
```

By default, the steady-state duration is 8 minutes.

Step 5 (Optional) Configure the device offline duration.

```
retry-time retry-time
```

By default, the offline duration is 5 minutes.

Step 6 Commit the configuration.

```
commit
```

----End

Example

1. After the device is managed by Qiankun Cloud, ensure that the NETCONF reliability function is enabled. For details about how to enable and configure the device stable duration and device offline duration, see "Procedure" in section 1.8.6. If the NETCONF reliability function is enabled, check the NETCONF reliability configuration in the MDCLI view. The values of stable-time and retry-time may be different from the following values.

```
[device@HUAWEI]
MDCLI> display system-controller/online-robustness-enhancement/ all
{
  "enabled": true,
  "stable-time": 8,
  "retry-time": 5
}
```

2. When the NETCONF reliability function is enabled on a device and the device goes offline, the device configuration is rolled back and the device goes online again after the retry-time specified in step 1 (in minutes) expires. The device cloud indicator blinks from fast to slow. On the MDCLI, the cloud connection status of the device is restored to "connected".

```
[device@HUAWEI]
MDCLI> sshs call-homes call-home call-home-name QiankunCloudService

[device@HUAWEI]/sshs/call-homes/call-home[call-home-name="QiankunCloudService"]
MDCLI> end-points end-point end-point-name DefaultEndPoint

[device@HUAWEI]/sshs/call-homes/call-home[call-home-name="QiankunCloudService"]/end-points/
end-point[end-point-name="DefaultEndPoint"]
MDCLI> display connection-status
"connected"
```

Verifying the Configuration

Run the **display system-controller/online-robustness-enhancement all** command to check the NETCONF robustness status, stable duration for triggering configuration backup, and offline duration for triggering configuration rollback.

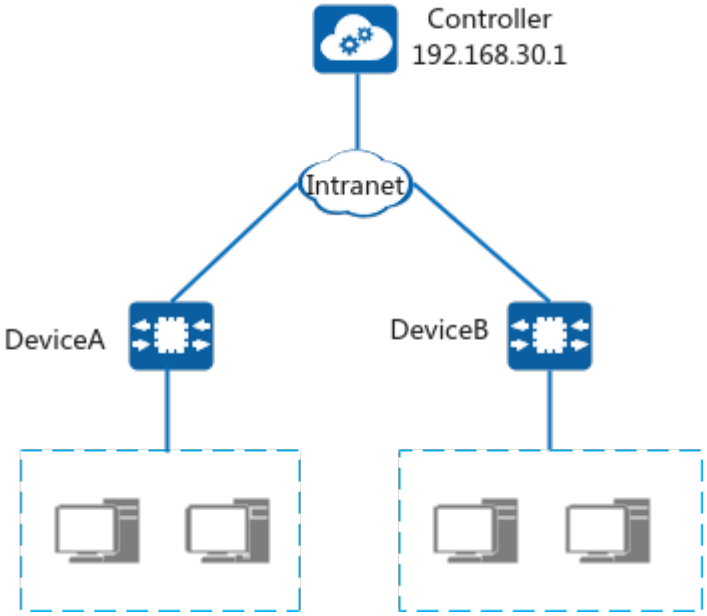
3.7.5.5 Example for Configuring a Device to Communicate with iMaster NCE-Campus Using NETCONF

Networking Requirements

When the NMS is used to centrally manage devices on a network that requires high security and scalability, you can use NETCONF to ensure communication between the NMS and the devices.

In **Figure 3-23**, a Huawei iMaster NCE-Campus server is used as the NMS. The administrator wants to configure and manage devices on the network through the NMS to improve network configuration and maintenance efficiency.

Figure 3-23 Network diagram of communication with iMaster NCE-Campus using NETCONF



Configuration Roadmap

Configure NETCONF connections between the NMS and devices so that the NMS can be used to configure and manage the devices. Configure a reachable route between the devices and the NMS based on the actual networking. The following uses DeviceA as an example. The configuration of DeviceB is similar to that of DeviceA, and is not described here.

1. Configure an SSH User.

   NOTE

   The SSH username in the current configuration must be **huawei**.
2. Configure an SSH server.
3. Enable NETCONF.
4. Establish a NETCONF connection between the device and the NMS.
5. Configure iMaster NCE-Campus.

Procedure

Step 1 Configure an AAA user.

```
[device@HUAWEI]
MDCLI> edit-config

[(gl)device@HUAWEI]
MDCLI> aaa lam users user name huawei

[*(gl)device@HUAWEI]/aaa/lam/users/user[name="huawei"]
If the value of MDCLI> service-api true level 3

[*(gl)device@HUAWEI]/aaa/lam/users/user[name="huawei"]
MDCLI> commit
```



```
[(gl)device@HUAWEI]/aaa/lam/users/user[name="huawei"]
MDCLI> quit 4
```

Step 2 Configure an SSH server.

```
[*(gl)device@HUAWEI]
MDCLI> sshs server-enable

[*(gl)device@HUAWEI]/sshs/server-enable
MDCLI> stelnet-ipv4-enable enable

[*(gl)device@HUAWEI]/sshs/server-enable
MDCLI> quit 2

[*(gl)device@HUAWEI]
MDCLI> sshs server

[*(gl)device@HUAWEI]/sshs/server
MDCLI> pki-domain default

[*(gl)device@HUAWEI]/sshs/server
MDCLI> commit

[(gl)device@HUAWEI]/sshs/server
MDCLI>quit 2
```

Step 3 Configure an SSH User.

```
[(gl)device@HUAWEI]
MDCLI> sshs users user name huawei

[*(gl)device@HUAWEI]/sshs/users/user[name="huawei"]
MDCLI> pub-key-type PKI

[*(gl)device@HUAWEI]/sshs/users/user[name="huawei"]
MDCLI> key-name default

[*(gl)device@HUAWEI]/sshs/users/user[name="huawei"]
MDCLI> commit

[(gl)device@HUAWEI]/sshs/users/user[name="huawei"]
MDCLI> quit 3
```

Step 4 Establish a NETCONF connection between the device and the NMS. (No address or port number is configured. Addressing is automatically performed from the registration center.)

```
[(gl)device@HUAWEI]
MDCLI> sshs call-homes call-home call-home-name cloud

[*(gl)device@HUAWEI]/sshs/call-homes/call-home[call-home-name="cloud"]
MDCLI> end-points end-point end-point-name cloud

[*(gl)device@HUAWEI]/sshs/call-homes/call-home[call-home-name="cloud"]/end-points/end-point[end-point-name="cloud"]
MDCLI> commit
```

NOTE

NETCONF is carried over SSH to ensure data transmission security. Before authentication, you need to import the ESN, device type, and CA certificate of each device to iMaster NCE-Campus. A local certificate and CA certificate are preconfigured on each device before delivery. To perform certificate-related operations, such as updating the local certificate, run commands on the device. For details, see "PKI Configuration" in the Configuration Guide - Security.

Step 5 Configure iMaster NCE-Campus.

Log in to iMaster NCE-Campus and add a device using the ESN of the device. For details about how to configure iMaster NCE-Campus, see iMaster NCE-Campus Product Documentation.

----End

Verifying the Configuration

Run the display display sshs/call-homes/call-home[call-home-name=cloud]/end-points/end-point[end-point-name=cloud]/connection-status command to check the status of the connection between the device and NMS.

```
MDCLI> display sshs/call-homes/ all
{
  "call-home": [
    {
      "call-home-name": "cloud",
      "end-points": {
        "end-point": [
          {
            "end-point-name": "cloud",
            "address": "10.1.1.1",
            "port": 10020,
            "connection-status": "connected",
            "enabled": true
          }
        ]
      }
    }
  ]
}
```

3.8 Fault Management Configuration

3.8.1 Overview of Fault Management

Definition

The fault management (FM) function manages alarms generated by devices in a centralized manner and provides guaranteed alarm reporting. It monitors the operating status of devices and networks in real time and records abnormalities, analyzes the abnormalities, and determines whether to generate and report alarms. This feature also enables a device to report alarms to notify users of faults, so that users can take measures to isolate and rectify faults for service recovery.

Purpose

With the expansion of network scale and increase of network complexity, when a device module is faulty, a large number of alarms may be generated on one or more devices. The devices and NMS, however, may not be able to process all alarms, resulting in loss of alarms during alarm transmission. If the alarms that users are concerned about are lost, network management will be difficult. To resolve this issue, a more intelligent and effective FM mechanism is required to implement the following improvements:

- Fewer alarms generated

To prevent alarm loss and ensure that valuable fault information can be collected quickly, you can configure alarm severities, alarm suppression, and delayed alarm reporting on devices before these devices report alarms.

- Guaranteed alarm reporting

  The internal reliability mechanism of the devices ensures that alarms are displayed promptly and reliably to support quick and accurate fault locating and diagnosis.

3.8.2 Understanding FM

FM Fundamentals

FM dynamically manages and reports alarms generated on devices in a centralized manner. If a device does not run properly, the device generates alarms to notify the maintenance personnel of the device's operating status, facilitating fault locating.

The common types of alarms are as follows:

- Active alarm

  An alarm indicating occurrence of a fault. For example, the hwFanInvalid alarm indicates that a fan is faulty.

- Clear alarm

  An alarm indicating that a fault is rectified. For example, the hwFanInvalidResume alarm indicates that a fault on a fan is rectified. Each active alarm has a corresponding clear alarm.

FM receives alarms generated by devices, saves the alarms based on the default severities, and records the time when alarms are generated.

Alarm Severity

Alarm information is classified to enable users to roughly determine the alarm severity and take measures for prompt fault recovery. In this way, alarms of high severity are handled at a high priority, preventing service interruption.

According to X.733, alarms are classified into four severities, as shown in **Table 3-35**. A smaller value indicates a higher severity.

Table 3-35 Definition of alarm severities

Value	Severity	Description
1	Critical	Services have been affected, and an immediate rectification measure is required.
2	Major	Services are being affected, and an urgent rectification measure is required.
3	Minor	A fault that does not yet affect services has occurred, and a rectification measure is required to prevent the fault from affecting services.

Value	Severity	Description
4	Warning	A potential fault that will affect services is detected. Actions should be taken to further diagnose the fault (if necessary) and rectify the fault before the fault grows in severity and affects services.

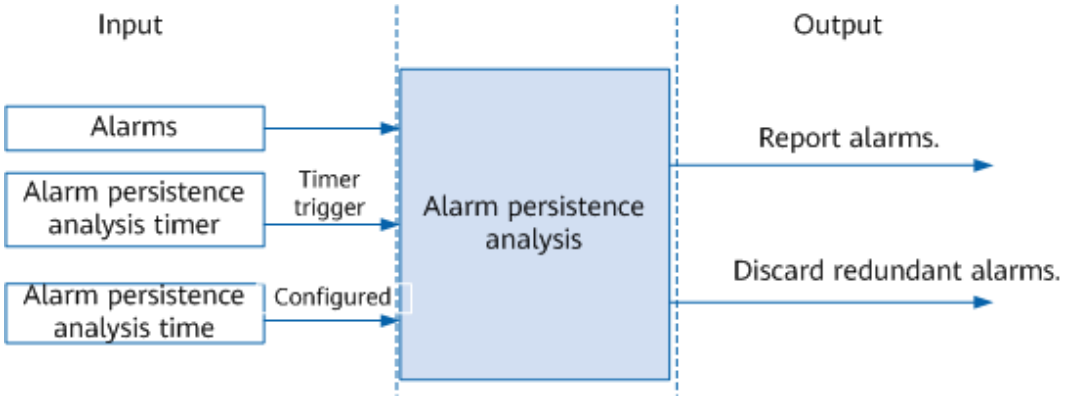
Alarm Suppression

Jitter suppression

Jitter suppression, focusing on analysis of alarm persistence, enables a device to report only alarms that persist after a set interval expires. This prevents a large number of invalid alarms from being reported.

Alarm persistence analysis provides a basis for a device to filter out non-persistent alarms and report only persistent alarms. **Figure 3-24** illustrates principles of alarm persistence analysis.

Figure 3-24 Alarm persistence analysis



Alarm persistence analysis measures the duration after an alarm (a fault alarm or clear alarm) is generated. If the period defined for an alarm has elapsed but the alarm persists, a notification is sent. If an alarm is cleared within the defined period, the alarm is filtered out and no notification is sent. That is, if an alarm persists for a short time, it is filtered out and no notification is reported. Only stable fault information is displayed in the case of fault flapping.

3.8.3 Configuration Precautions for Fault management

Licensing Requirements

Fault Management is not under license control.

Hardware Requirements

Table 3-36 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.8.4 Default Settings for FM

Table 3-37 describes the default settings for FM.

Table 3-37 Default settings for FM

Parameter	Default Setting
Delayed alarm reporting	Enabled
Wrap function of historical alarm records	Enabled

3.8.5 Maintaining FM

Clearing Alarms

After identifying alarms to be cleared, perform the following operations in the alarm management view.

CAUTION

After an alarm is cleared, an NMS cannot obtain the alarm information. Cleared alarms cannot be restored. In addition, the alarm is no longer reported even if the original fault persists. Therefore, exercise caution when you decide to clear alarms.

Table 3-38 Clearing alarms

Operation	Command
Clear all active alarms.	**clear-all-active-alarm**

Operation	Command
Clear specified active alarms.	**clear-active-alarm alarm-sequence** *sequence*
Clear all historical alarms.	**clear-all-history-alarm**
Clear specified historical alarms.	**clear-history-alarm alarm-sequence** *sequence*

Monitoring Alarms

You can run the following commands in the root view to check the alarm information in routine maintenance.

Table 3-39 Monitoring alarms

Operation	Command
View alarm configurations.	**display fm/alarms/alarm**[**alarm-name** *alarm-name*]
View active alarms.	**display fm**/**active-alarms**/[**active-alarm**[**sequence**=*sequence*]]
View historical alarms.	**display fm**/**history-alarms**/[**history-alarm**[**alarm-sequence**=*sequence*]]

3.9 Upgrade Maintenance Configuration

3.9.1 Overview of Upgrade Maintenance

Background

To improve the performance of existing system software on a device, you can maintain or upgrade the device so that the device with high performance is properly running on a network. Specific operations involve:

- Upgrading system software

  A system software upgrade helps you optimize device performance, add new features, and remove defects existing in versions that are not the latest.
- Installing patches

  Patches are a type of software compatible with system software. They are used to fix a few urgent defects in the system software. By installing a patch, you can upgrade the system without upgrading the system software version.

Purpose

When you need to add new features, improve existing system performance, or replace existing resource files to meet existing user requirements, you can upgrade a device by installing the target-version software packages and patch packages.

Benefits

Upgrading and maintaining devices optimizes system performance and helps devices be properly running on live networks.

3.9.2 Understanding Upgrade Maintenance

3.9.2.1 Overview of the Basic Software Package

Overview of the Basic Software Package

The upgrade and maintenance operations are performed based on the basic software package. The basic software package provides basic capabilities for software running, such as hardware drivers, common components, an operating system, and a boot file. It is the basis for the running of components and services on a device.

You can manage a basic software package separately. The basic software package can be updated in in-service mode. In addition, patches or MOD files with a matching version can be installed for the basic software package. Before managing system software, note the following:

- Obtain a system software package of a specified target version and matching documentation from the Huawei support website.
- Before uploading the system software package onto a device, ensure that sufficient storage space is available on the master and slave main control boards.
- Install or upgrade the system software by following the procedure described in an installation or upgrade guide released by Huawei.
- Purchase and install licenses if you need some service function modules or capacity-based capabilities that are controlled by licenses.

When you install or upgrade system software, enable log and alarm management functions to record installation or upgrade operations on a device. The recorded information helps diagnose faults if installation or an upgrade fails.

Basic Upgrade Maintenance Functions

The basic upgrade maintenance operations on the basic software package are as follows:

- Specifying the target-version basic software package (*.cc) that takes effect at a next startup, which updates basic software of a device
- Dynamic in-service installation of a patch package (*.PAT) for basic software to rectify software defects

NOTE

"In-service" in the upgrade maintenance sections indicates that the device does not need to be restarted.

- Dynamic in-service installation of a module file (*.MOD) for basic software to strengthen system software

Integrity Check

After a software package is released, it may be modified or its components may be tampered with during transmission, download, storage, and installation. During the installation of a software package (patch, MOD patch, or basic software package), digital signatures and hash values are used to verify the validity and integrity of software packages. Software is verified before being installed, ensuring security.

When a software package is released, digital signature verification is performed on the software package, and the digital signature information is packed into the software package for release. When the software package is installed, the device verifies the digital signature. The software package is considered complete and trusted and applications can be installed only after the verification succeeds. A hash value is a unique character string consisting of short random letters and digits. When a software package is installed, the device verifies the hash value. The software package is considered complete and trusted and applications can be installed only after the verification succeeds.

3.9.2.2 Basic Software Package Management

Installing a Basic Software Package

When a device starts for the first time, you can load a basic software package on the BIOS interface, restart the device, and use O&M commands to configure service parameters. An installed basic software package takes effect only after a device is restarted. Such a package does not support dynamic in-service installation or uninstallation.

Loading a Basic Software Package for a Version Upgrade

When you want to add new features, improve existing system performance, or remove defects in an existing version, upgrade basic software by loading a target-version software package and restarting the device. Device upgrades are closely related to newly released versions. Each new version comes with a corresponding upgrade guide, which you can follow during the upgrade process.

3.9.2.3 Management of Patches Mapped to a Basic Software Package

During device running, you may need to modify the system software of the device. For example, you may need to remove system defects or add new functions based on service requirements. The conventional method is to disconnect a device from a network and upgrade system software in offline mode. This method adversely affects services and deteriorates communication service quality.

To address these issues, you can load a patch package dynamically on a running device to improve communication service quality, without affecting services.

Patch Classification

Based on the impact of patch effectiveness on running services, patches are classified as hot patches or cold patches.

- Hot patch (HP): takes effect without affecting or interrupting services. Using hot patches helps reduce device upgrade costs and prevent upgrade risks.
- Cold patch (CP): takes effect only after a board is reset or a device is restarted, which adversely affects services.

Based on patch dependency, patches are classified as incremental patches or non-incremental patches.

- Incremental patch: is dependent on previous patches. An incremental patch package must contain all patch information that is contained in a previous patch package. You can install the incremental patch package without uninstalling an existing patch package.
- Non-incremental patch: is the sole patch package of the same type installed on an existing device. Before you install another non-incremental patch package, uninstall the existing one and then install the new non-incremental patch package.

NOTE

All patches released for products are hot patches and incremental patches. All patches mentioned in the following sections are such patches, unless otherwise specified.

Patch Status

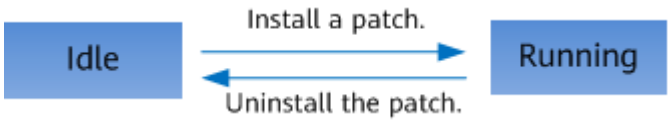
Each patch has its own state that can only be changed statically. **Table 3-40** describes patch states.

Table 3-40 Patch status

Status	Description	Status Transition
Idle	A patch package is stored in a storage medium of a device, but patches in the package are not loaded into the patch area in the memory. In this situation, the patches in the package are in the idle state.	After the patches in the storage device are loaded into the patch area in the memory, the patch status becomes running.
Running	After the patches in the storage device are loaded into the patch area in the memory, the patch status becomes running. After a board or device is reset, the running state remains if the patches are in the running state before the reset.	You can uninstall a running patch package so that the device deletes the patches from the patch area in the memory and sets the status to idle.

Figure 3-25 shows patch state transition.

Figure 3-25 Transition between patch states



Patch Installation

For basic software, you can install patches in either of the following modes:

- Dynamic in-service patch installation: A patch is installed on a device, without interrupting services. This mode is also called hot patch installation.

  For details on how to install a patch package in this mode, see a corresponding patch installation guide that is released with a patch version.
- Next-startup patch installation: You can specify a patch that takes effect at a next startup. This mode is also called cold patch installation. This mode is usually used during a device upgrade.

Patch management simplifies patch operations by supporting one-click patch installation (idle -> running) and one-click patch removal (no patch).

Manual Patch Uninstallation

After a patch is installed, if services become abnormal due to the patch, you can uninstall the patch to ensure normal services.

Automatic Patch Rollback

If an exception occurs during the patch installation, the patch may fail to be installed. In this case, the system automatically rolls back to the previous patch version.

3.9.2.4 Smart Upgrade

Smart upgrade is an upgrade mode in which a device automatically selects the software version to be upgraded from the Huawei HOUP (https://houp.huawei.com) through HTTPS. It supports automatic or manual download of a software package or patch package for automatic or manual upgrade of the device. After the smart upgrade is complete, the upgrade result is automatically reported to the HOUP.

This function simplifies the upgrade operations and enables customers to upgrade the system software version by themselves, greatly reducing the device maintenance costs. In addition, the upgrade policy on the HOUP standardizes the upgrade path, which effectively prevents engineers from selecting an incorrect software version and reduces the possibility of upgrade failures.

Smart upgrade supports the following modes:

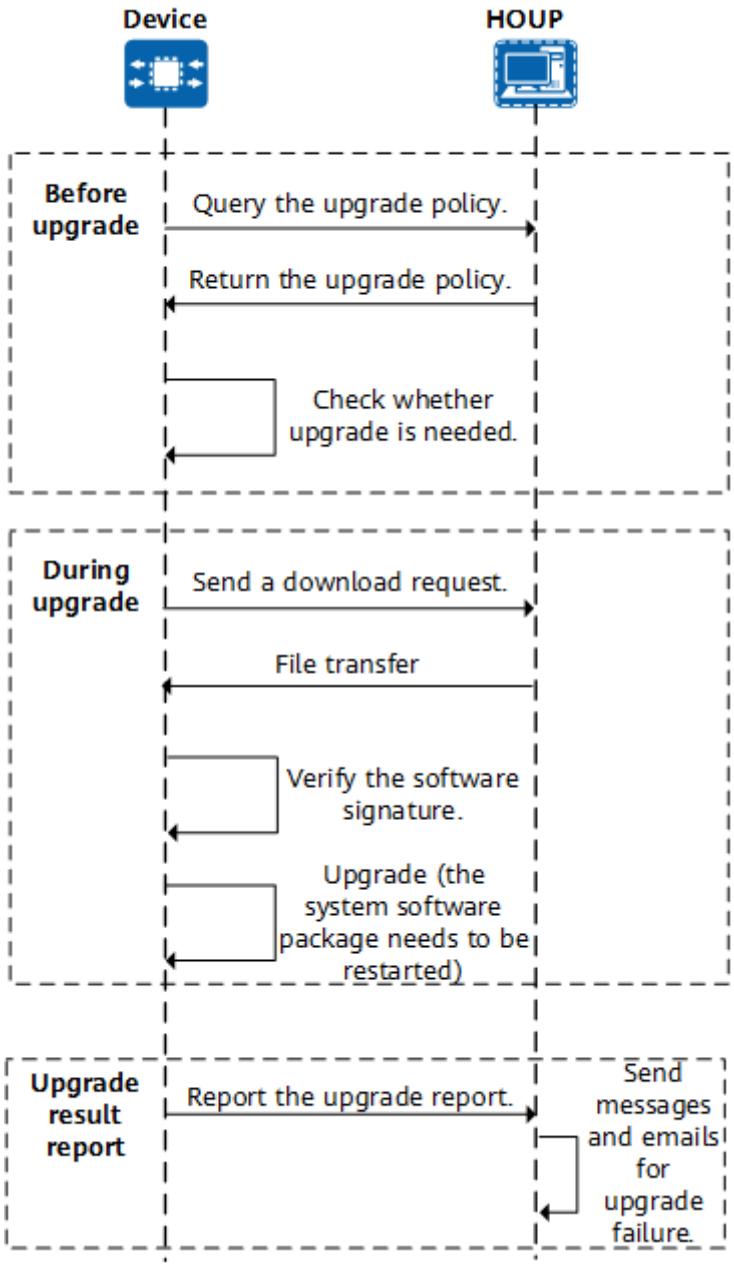
- Scheduled smart upgrade

  The time for scheduled smart upgrade is configured on the device. When the specified time arrives, the device automatically starts smart upgrade and reports the upgrade result to the HOUP. Scheduled smart upgrade supports the setting of the upgrade time by year, month, or week.

- Manual smart upgrade

  If scheduled smart upgrade is not configured or you want to manually perform a smart upgrade before the scheduled smart upgrade time arrives, you can run the command to download the system software of the target version and manually perform the smart upgrade.

Figure 3-26 Process of manually triggering a smart upgrade



To improve smart upgrade security, the following security measures are provided:

- Signature file of the software or patch package

  A signature file with the file name extension .asc is generated when the system software package or patch package is archived on the HOUP. During a smart upgrade, the software package and its signature file are automatically downloaded and verified.
- SSL Policy

  A smart upgrade is performed based on the HTTPS protocol, which uses the SSL channel. The SSL policy is used on the device to describe the SSL channel configuration. In a smart upgrade, the CA certificate loaded in the SSL policy can be used to check whether the server is valid. If the server does not need to be verified, the CA certificate is not required, but the SSL policy still needs to be specified.

3.9.3 Configuration Precautions for Upgrade and Maintenance

Licensing Requirements

Upgrade maintenance is not under license control.

Hardware Requirements

Table 3-41 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

3.9.4 Default Settings for Upgrade Maintenance

Table 3-42 describes the default settings for upgrade maintenance.

Table 3-42 Default settings for upgrade maintenance

Parameter	Default Value
Name of a system software package	Storage device name
Patch file name	No patch package name is specified.
Module file name	No module file name is specified.

3.9.5 Preparing for Upgrade Maintenance

Items to Be Prepared for Upgrade Maintenance

To ensure the smooth upgrade maintenance of a device, prepare for the upgrade maintenance strictly based on requirements.

Preparing for Upgrade Maintenance	Purpose	Operation
Check the existing software version of a device.	Prevent an upgrade failure due to a version mismatch.	Run the **display version** command to check the existing basic software version of a device.

3.9.6 Upgrading a Version by Specifying the Basic Software Package That Takes Effect at the Next Startup

Context

Upgrading the basic software version of the device can optimize existing device performance, add new performance, and remove defects existing because software in the current version is not updated in time.

NOTE

Be sure to check the correctness of each uploaded file by comparing the file size and date.

Procedure

Step 1 Upload the basic software package to the storage medium. For details, see "File System Management Configuration" in the *Configuration Guide - Basic Configuration*.

Step 2 Specify the basic software package that takes effect at the next startup.

```
startup-by-mode name name
```

NOTE

If a patch has been loaded for a software package before the package is upgraded, when you roll back to the source version, you must run the **startup patch** *patch-name* command for the patch to take effect.

Step 3 Restart the device.

```
reboot
```

----End

Verifying the Configuration

Run the **display software/startup-packages/** command to check whether the displayed information is the same as the name of the basic software package to be started.

3.9.7 Configuring Dynamic In-Service Patch Installation and Uninstallation

Prerequisites

Before configuring dynamic in-service patch installation and uninstallation, complete the following tasks:

- Log in to the desired device.
- **3.9.5 Preparing for Upgrade Maintenance**.

Context

During device running, you may need to modify the system software of the device. For example, you may need to remove system defects or add new functions based on service requirements. Configuring dynamic in-service installation of a patch package helps you optimize software without interrupting device running.

For details about configuration parameters, see huawei-patch.yang.

NOTE

Be sure to check the correctness of each uploaded file by comparing the file size and date.

Procedure

Step 1 Upload the patch file to the storage medium of the main control board. For details, see "File System Management Configuration" in the *Configuration Guide - Basic Configuration*.

Step 2 Configure dynamic in-service installation of a patch file.

load-patch load-type run name *patch_name*

NOTE

- When loading a patch package, the device checks whether the patch version is consistent with the system software version. If not, the device fails to load the patch package.
- If there are running patch files in the system during non-incremental patching, the system displays a patch file installation failure. The **delete-patch delete-type all** command needs to be run to delete the running patch files.
- This command can be used to load a hot patch or a cold patch. After loading a cold patch, you need to restart the device for the cold patch to take effect. If you run the **reset-startup-patch delete-type all** command to delete the patch file from the device after the cold patch is installed, you do not need to restart the device.

----End

Verifying the Configuration

- Run the **display patch/patch-infos** command to check whether the installed patch file is in the running state.

Follow-up Procedure

If a patch file needs to be deleted after having been installed, run the **delete-patch delete-type all** command to delete all patch files in the system.

3.9.8 Specifying the Patch Package That Takes Effect at the Next Startup

Prerequisites

Before specifying the patch package that takes effect at the next startup, you have completed the following tasks:

- Log in to the desired device.
- **3.9.5 Preparing for Upgrade Maintenance**.

Context

During device running, you may need to modify the system software of the device. For example, you may need to remove system defects or add new functions based on service requirements. You can specify a patch package that takes effect at the next startup to optimize software.

For details about configuration parameters, see huawei-patch.yang.

NOTE

Be sure to check the correctness of each uploaded file by comparing the file size and date.

Procedure

Step 1 Upload the patch file to the storage medium of the main control board. For details, see "File System Management Configuration" in the *Configuration Guide - Basic Configuration*.

Step 2 Specify the patch package that takes effect at the next startup.

```
startup-next-patch name patch-name
```

Step 3 Restart the device.

```
reboot
```

----End

Verifying the Configuration

- Run the **display patch/next-startup-patchs** command to check whether the installed patch file is in the running state.
- Run the **diagnose hpe query-patch app-name** *process_name* command to view the patch status of the process in the HPE.

Follow-up Procedure

To remove the configuration of specifying a patch file for next startup, run the **reset-startup-patch delete-type all** command to delete the startup patch file.

3.9.9 Configuring Module Loading

Prerequisites

Before configuring module loading, you have completed the following tasks:

- Log in to the desired device.
- **3.9.5 Preparing for Upgrade Maintenance**.

Context

If a desired module does not exist in the system, you can perform in-service installation of the module so that the functions of the module can be used. In this process, services are not interrupted. You can also perform dynamic in-service uninstallation of a module if you no longer need it.

NOTE

Be sure to check the correctness of each uploaded file by comparing the file size and date.

Procedure

Step 1 Upload the module file to the storage medium of the main control board. For details, see "File System Management Configuration" in the *Configuration Guide - Basic Configuration*.

Step 2 Load a module file.

install-module name *module*

----End

Verifying the Configuration

- Run the **display module-management/module-infos** command to check whether the displayed information is the same as the name of the installed module file and whether the module takes effect.

Follow-up Procedure

If a module is no longer needed, run the **uninstall-module name** *name* command to uninstall it.

3.9.10 Configuring Smart Upgrade

Prerequisites

Before performing a smart upgrade, ensure that there are reachable routes between the device and HOUP server.

Context

The smart upgrade function enables a device to obtain upgrade files from Huawei HOUP and upgrade the device in one-click or scheduled mode. This simplifies upgrade operations and reduces upgrade and maintenance costs.

Procedure

Step 1 Configure the device to communicate with the HOUP server.

Ensure that there are reachable routes between the device and HOUP network, and configure the URL and HTTPS port number of the HOUP server on the device. This allows the smart upgrade function to connect to the HOUP for files required for by upgrade.

To ensure the security of smart upgrade, the device supports the configuration of CA certificate authentication for the server.

1. Enter the global smart upgrade view.
   smart-upgrade global
2. Enter the editing mode.
   edit

1. Configure the URL and HTTPS port number of the server.
   { **url** *host* | **port** *port-value* }
2. Bind an SSL policy. After an SSL policy is configured, the device generates a certificate and loads a trusted-CA file for the SSL policy. The trusted-CA file is used to verify the authenticity of the digital certificate sent by the HOUP server.
   ssl-policy *policy-name*
3. Commit the configuration.
   commit

Step 2 Set the smart upgrade mode.

A smart upgrade can be triggered manually or at a scheduled time. You can select either one or two of the following upgrade modes based on network maintenance requirements:

- Configure a smart upgrade to be triggered at a scheduled time.
  create-scheduled-upgrade { **scheduled-time** *scheduled-time-value* | **software-version** *software-version-value* | **patch-version** *patch-version-value* }
- Configure a smart upgrade to be triggered instantly.
  a. (Optional) Check whether the HOUP has the system software that can be upgraded.
     check-new-version
  b. (Optional) Download the system software.
     download-software
     version-lists version-list version version-value
     emit
  c. (Optional) Delete unnecessary software packages.
     clean-discarded-package
     clean-type all
     emit
  d. Configure the device to be upgraded immediately after the system software is downloaded successfully.


```
upgrade-right-now
recheck-new-version true
emit
```

e. Configure the device to be upgraded to a specified version after the system software is downloaded successfully. The **base-name** and **base-version** commands and the **patch-name** and **patch-version** commands must be configured in pairs.

```
upgrade-right-now
base-name base-name-value
base-version base-version-value
patch-name patch-name-value
patch-version patch-version-value
emit
```

----End

Verifying the Configuration

After the smart upgrade configuration is complete, run the **display smart-upgrade smart-upgrade-info all** command to check the configuration, download status, upgrade version, upgrade status, and local information of the smart upgrade function.

3.9.11 Loading a Digital Signature Certificate Revocation List (CRL)

Prerequisites

Before loading a CRL, complete the following tasks:

- Log in to the desired device.
- Upload the CRL to the flash path of the device.

Context

If an issued digital signature certificate needs to be revoked due to key disclosure or other reasons, a third-party tool can be used to mark the certificate invalid and add the certificate to the CRL. After you load the latest CRL to a device, the device does not verify the digital signature certificate upon next startup.

Procedure

Step 1 Load the CRL file to the system.

```
software-crl-load name crl-name
```

----End

Verifying the Configuration

Run the **display codesign/software-crls/** command to check information about the loaded CRL file.

3.9.12 Maintaining Software

Procedure

After performing software maintenance, you can verify the current software version and delete patch files as needed to ensure the normal running of the system.

Table 3-43 describes operations related to software package file maintenance.

Table 3-43 Operations related to software package file maintenance

Operation	Command	Description
Check the software version.	**display software/ versions/**	You can view the current software version to determine whether the device needs to be upgraded or has been upgraded successfully.
Check whether the running software package is reliable.	**display software/ startup-packages/**	-
Check the digital signature certificate revocation list (CRL).	**display codesign/ software-crls/**	-

4 System Forwarding Configuration

4.1 Session Management Configuration

4.1.1 Overview of Session Management

Definition

A session on a device is an entry used to record the connection status of protocols such as TCP, UDP, and ICMP, and is fundamental for packet forwarding. Session management enables you to change the aging time of sessions based on service requirements.

Purpose

A device performs complex processing when packets pass through it. If a large number of packets pass through a device, they pose a big challenge to device performance and packet processing efficiency.

The session mechanism is applied to improve packet processing efficiency. The device establishes a session for packets that meet specific conditions. If subsequent packets match the session, the device skips some procedures, and processes the packets according to the measures recorded in the session.

A session table records multi-tuple information and processing measures of packets. When forwarding TCP, UDP, and ICMP packets, the device queries the session table and quickly processes packets based on the session table.

4.1.2 Configuration Precautions for Session Management

Licensing Requirements

Session Management is not under license control.

Hardware Requirements

Table 4-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

4.1.3 Default Settings for Session Management

Table 4-2 lists the default aging time for sessions of different types.

Table 4-2 Default aging time for sessions of different types.

Session Type	Aging Time (Seconds)
TCP	240
UDP	120
ICMP	20
DNS	30
HTTP	240
FTP	240
SIP	600
RTSP	240
PPTP	240

4.1.4 Configuring Session Aging

Context

After a session is created, if no packet matches the session for a long time, the connection is torn down, and the session is considered unnecessary. To save system resources, the system deletes the session that is not matched for a certain period of time; that is, the session is aged out.

The impact of aging time on forwarding is as follows:

- If the aging time is too long, a large number of sessions whose corresponding connections have been torn down may exist in the system and consume system resources. In addition, new sessions may not be created, thus affecting the forwarding of other services.
- If the aging time is too short, some sessions may be aged out before packet transmission is finished, resulting in service interruptions.

Generally, you can use the default aging time of sessions. For details about the default aging time for sessions of different types, see **4.1.3 Default Settings for Session Management**. You can also modify the aging time based on the traffic type and number of connections.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Configure the session aging time based on the protocol type.

```
sec-session-mgmt protocol-ttls protocol-ttl protocol { tcp | udp | icmp }
aging-time aging-time
```

Step 3 Configure the session aging time based on the application.

```
sec-session-mgmt application-ttls application-ttl application { dns | http | ftp | sip | rtsp | pptp }
aging-time aging-time
```

Step 4 Commit the configuration.

```
commit
```

----End

4.1.5 Managing Sessions

Clearing a Session Table

The session table on the device controls packet forwarding, and session entries do not age if they are used to direct traffic all the time. Therefore, in certain cases, the session table must be cleared for the device to generate a new session table.

NOTICE

After you clear the session table, all connections and services that need to search the session table for packet forwarding are forcibly disconnected. Users must re-initiate a connection to resume the communication. Therefore, exercise caution before you clear a session table. If you must clear it, limit the range of session entries to be cleared by setting one or more conditions. Avoid clear all session entries. Only some common commands are listed here.

- Run the following command in the system view to clear all session entries on the device.

```
diagnose hpf reset-session
emit
```

- Run the following command in the system view to clear specified session entries on the device.

```
diagnose hpf reset-session option option-value
```


4.2 Server Mapping Entries Configuration

4.2.1 Overview of Server Mapping Entries

Definition

Server mapping entries record the connection status for multi-channel protocol packets or address translation relationships in NAT scenarios.

Purpose

- Server mapping entries record the connection status for multi-channel protocol packets to ensure normal communication initiated by Internet users.
- Server mapping entries help the device translate IP addresses and port numbers in packets.

  In addition to the connection status for packets, the server mapping entries record the processing measures on the packet. If a packet matches a server mapping entry, the device translates the IP address and port number in the packet according to the matched server mapping entry. For example, when NAT Server is configured, the device translates the destination public IP address of a packet destined for the intranet into a private IP address.

4.2.2 Understanding Server Mapping Entries

4.2.2.1 Generation of Server Mapping Entries

Currently, the device generates server mapping entries when processing multi-channel protocol packets and some NAT services.

1. Server mapping entries are generated when the device forwards the traffic of multi-channel protocols, such as FTP and RTSP, after the application specific packet filter (ASPF) mechanism is configured.
2. Static server mapping entries are generated when NAT Server is configured.
3. Dynamic server mapping entries are generated when NAT No-PAT is configured.

Server Mapping Entries Generated When the Device Forwards Traffic of Multi-Channel Protocols, such as FTP and RTSP

For a multi-channel protocol, the data channel is dynamically negotiated through the control channel between the client and the server. That is, the port numbers of the communication parties are not fixed.

After ASPF is configured, the device detects the negotiation process of the channels, and dynamically creates a server mapping entry according to the address information in the key packet payload. The mapping entry will be looked up during the connection initiation of the data channel. This server mapping entry

contains information about the data channel negotiated in the packets of the multi-channel protocol.

For example, in FTP port mode, the device randomly selects a port (**2165** in this example) for the data channel on the FTP client and sends the port number to the FTP server through the control channel. Then the FTP server initiates a data connection to the port. The corresponding server mapping entry is as follows:

```
Type: ASPF,  10.40.0.5 -> 10.40.0.10:2165,  Zone:---
Protocol: tcp(Appro: ftp-data),  Left-Time:00:04:47
Vpn: public -> public
```

Static Server Mapping Entries Generated When NAT Server Is Configured

After NAT Server is configured, Internet users can send access requests to the intranet server. The IP addresses and port numbers of the users are unknown, but the IP address and port number of the intranet server are known. Therefore, after NAT Server is configured on a device to determine the mappings between public and private IP addresses, the device generates server mapping entries to store the mappings. Then the device can translate IP addresses and forward packets according to the mappings.

A static server mapping entry is generated for each valid NAT Server mapping. If you delete the NAT Server configuration, the server mapping entries are also deleted.

The server mapping entries generated by NAT Server are as follows:

```
Type: Nat Server,  ANY -> 10.10.1.100:21[10.1.1.2:21],  Zone:---,  protocol:tcp
Vpn: public --> public

Type: Nat Server Reverse,  10.1.1.2[10.10.1.100] -> ANY,  Zone:--- ,  protocol:tcp
Vpn: public --> public,  counter: 1
```

- **Nat Server** indicates the server mapping entry for forward traffic (initiated from the clients on the Internet to the intranet server).
- **Nat Server Reverse** indicates the server mapping entry for return traffic (initiated from the intranet server to the clients on the Internet).
- **protocol** indicates the specified protocol when you configure NAT Server.
- **10.10.1.100** indicates the public IP address for NAT Server.
- **10.1.1.2** indicates the private IP address for NAT Server.

Dynamic Server Mapping Entries Generated When NAT No-PAT Is Configured

If you configure NAT and specify the No-PAT parameter, the device translates only the IP addresses but not the port numbers. In this case, all the ports at internal IP addresses are translated to ports at external IP addresses, and Internet users can initiate connections to any ports on the intranet. Therefore, after NAT No-PAT is configured and traffic matches a NAT policy, the device generates server mapping entries to store the mappings between internal and external IP addresses. Then the device translates the IP addresses and forwards the packets according to the mappings.

The server mapping entries generated when NAT No-PAT is configured are as follows:

```
Type: No-Pat,  10.1.1.100[10.10.1.1] -> ANY,  Zone:---
Protocol: ANY, TTL:360, Left-Time:353,  Pool: 3, Section: 0
```



```
Vpn: public

Type: No-Pat Reverse, ANY -> 10.10.1.1[10.1.1.100],  Zone:---
Protocol: ANY, TTL:---, Left-Time:---,  Pool: 3, Section: 0
Vpn: public
```

- **No-Pat** indicates the server mapping entry for the forward traffic (initiated from an intranet IP address).
- **No-Pat Reverse** indicates the server mapping entry for the return traffic (initiated from an Internet IP address to an intranet IP address).
- **10.1.1.100** indicates the pre-NAT intranet IP address.
- **10.10.1.1** indicates the post-NAT Internet IP address.

4.2.2.2 Function of Server Mapping Entries

Position of Server Mapping Entries During Packet Forwarding

After receiving and processing a packet, the device searches for a session entry first. If the session entry is matched, the device processes and forwards the packet based on the session entry. If no session entry is matched, the device searches for a server mapping entry and then starts the session creation process. The device searches for a matched server mapping entry for two purposes:

- Allow special service packets to pass through: For some special service packets (for example, FTP data packets forwarded through a port that is temporarily negotiated), it may be difficult to configure refined security policies to permit the packets. In this case, you can create server mapping entries to record information about the temporarily negotiated connection. Special service packets that match the server mapping entries are permitted, and no security policy check is required.
- Translate the IP address and port number of the packet: The device matches the IP address and port number of the packet against server mapping entries. If a match is found, the device translates the IP address and port number of the packet based on the entry.

The server mapping entries do not record information such as the next hop of a packet. Therefore, after a packet matches a server mapping entry, the device still needs to search for the routing table and then create session entries for subsequent packet forwarding.

The following uses FTP in port mode as an example. Host A at 192.168.1.1 initiates an FTP connection from port 20000 to port 21 on server B at 10.11.1.1. A security policy has been configured to allow 192.168.1.1 to initiate a connection to port 21 at 10.11.1.1.

1. After the packet for initiating a connection is permitted, the device creates a session entry, and the control channel is established.

Type	Source IP Address	Source Port	Destination IP Address	Destination Port	Protocol
Session entry	192.168.1.1	20000	10.11.1.1	21	FTP

2. After the negotiation of the control channel, host A randomly selects a port (for example, port 2165) for the data channel, and then sends the port number to the FTP server through the control channel. After receiving the packet, the device establishes a server mapping entry accordingly.

Type	Source IP Address	Destination IP Address	Destination Port	Protocol
Server mapping entry	10.11.1.1	192.168.1.1	2165	FTP-Data

3. After receiving the negotiation packet from host A, server B initiates a connection from port 20 to port 2165 at 192.168.1.1. The connection does not match any session entry in the session table. If the default interzone security policy is **deny**, the packet may be discarded. Therefore, the device checks the server mapping.

   Since the source IP address, destination IP address, destination port, and protocol of the packet are the same as those in the server mapping entry, the device allows this packet to pass through. Moreover, the 5-tuple of the new data channel is determined, and the device creates a session entry. Then the data channel between the communication parties is established.

Type	Source IP Address	Source Port	Destination IP Address	Destination Port	Protocol
Session entry	10.11.1.1	20	192.168.1.1	2165	FTP-Data

4. Subsequent packets in the data channel can match this session entry and be forwarded without matching any server mapping entry. If the server mapping entry is not matched by any packet for a long time, it will be deleted after the aging time expires. This mechanism ensures that the server mapping entries can be deleted in a timely manner, ensuring network security.

4.2.2.3 Aging of Server Mapping Entries

Server mapping entries occupy device resources. The device provides an aging mechanism to clear the server mapping entries that meet the specified aging conditions.

Table 4-3 lists the aging mechanisms for the server mapping entries.

Table 4-3 Aging mechanisms for server mapping entries

Server Mapping Type	Aging Time	Description
Server mapping entries generated when the device forwards the traffic of multi-channel protocols	If no traffic is matched, the entries age according to the default aging time of various protocols. For details, see **Table 4-4**.	The aging time for server mapping entries cannot be configured.
Static server mapping entries generated when NAT Server is configured	These server mapping entries are not aged, but they are deleted automatically when NAT Server configuration is deleted.	-
Dynamic server mapping entries generated when NAT No-PAT is configured	If no traffic matches a dynamic server mapping entry, the entry is aged out based on the aging time configured for the No-PAT type after the corresponding session entry is aged out.	This type of dynamic server mapping entry is triggered by traffic. The entry applies to all intranet ports and ages out when no traffic is forwarded to ensure intranet security. You can run the **firewall server-map aging-time** command to set the aging time for server mapping entries of the No-PAT type.

Table 4-4 Aging time of server mapping entries for common protocols

Protocol	Aging Time (Second)
RTP	55
RTCP	65
FTP DATA	15
PPTP DATA	15
SIP	240
RTSP	20

5 Ethernet Switching Configuration

5.1 MAC Configuration

5.1.1 Overview of MAC Addresses

Definition

A Media Access Control (MAC) address, also called a physical address, hardware address, or link address, is burned into the network interface card (NIC) of a network device by a vendor to uniquely identify the device's location. A MAC address consists of 48 bits and is displayed as a 12-digit hexadecimal number. Among the 48 bits, bits 0 to 23 are assigned by the Internet Engineering Steering Group (IETF) or other institutions to identify vendors, and bits 24 to 47 are the unique ID assigned by vendors to identify their NICs.

Purpose

An IP address cannot identify a user over the Internet because the IP address is only a logical identifier and can be modified. To address this problem, a MAC address is used to uniquely identify a user.

MAC addresses fall into the following types:

- Physical MAC address: uniquely identifies a terminal on an Ethernet network and is the globally unique hardware address of the terminal.
- Broadcast MAC address: identifies all terminals on a LAN and is all 1s (FF-FF-FF-FF-FF-FF).
- Multicast MAC address: identifies a group of terminals on a LAN. MAC addresses, excluding broadcast MAC addresses, are multicast ones if their eighth bit is 1, for example, 01-00-00-00-00-00. The multicast MAC address starting from 01-80-c2 is the bridge protocol data unit (BPDU) MAC address, and is often used as the destination MAC address in protocol packets to indicate the protocol type.

On network devices, physical MAC addresses that are often mentioned include bridge MAC addresses, system MAC addresses, and interface MAC addresses. By default, the system MAC address of a device is the bridge MAC address, which is the physical MAC address of the device when it is delivered. In addition, the device provides a certain number of available system MAC address ranges that are used for dynamically allocating MAC addresses to Layer 3 interfaces.

5.1.2 Configuration Precautions for MAC

Licensing Requirements

MAC is not under license control.

Hardware Requirements

Table 5-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

5.1.3 Understanding MAC Address Tables

5.1.3.1 Definition and Classification of MAC Address Entries

Definition of a MAC Address Table

A MAC address table records the mapping between interfaces, VLANs, and MAC addresses. Before forwarding a packet, a device looks up the destination MAC address of the packet in the MAC address table. If a MAC address entry matches the destination MAC address, the device forwards the packet to the outbound interface in the MAC address entry. If no MAC address entry matches the destination MAC address, the device broadcasts the packet to all interfaces in the corresponding VLAN, except the inbound interface receiving the packet.

Classification of MAC Address Entries

MAC address entries are classified in to dynamic, static, and blackhole entries.

Table 5-2 Characteristics and functions of different MAC address entries

MAC Address Entry Type	Characteristics	Function
Dynamic MAC address entry	Dynamic MAC address entries are obtained by learning source MAC addresses of packets received on an interface, and will age out. Dynamic MAC address entries are lost after a system restart, card hot swap, or card reset.	You can check whether data is forwarded between two connected devices by checking dynamic MAC address entries. You can obtain the number of users connected to an interface by checking the number of specified dynamic MAC address entries.
Static MAC address entry	Static MAC address entries are manually configured and delivered to each card. Static MAC address entries never age out. The static MAC address entries saved in the system are not lost after a system restart, card hot swap, or card reset. Each static MAC address entry can have only one outbound interface. After an interface is statically bound to a MAC address, other interfaces discard packets originating from that MAC address. Statically binding an interface to a MAC address does not affect the learning of dynamic MAC address entries on the interface.	When static MAC address entries are configured, authorized users can use network resources, and other users are prevented from using the bound MAC addresses. This can block certain attacks.

MAC Address Entry Type	Characteristics	Function
Blackhole MAC address entry	Blackhole MAC address entries are manually configured and delivered to each card. Blackhole MAC address entries never age out. The blackhole MAC address entries saved in the system are not lost after a system restart, card hot swap, or card reset. After blackhole MAC address entries are configured, the device discards packets originating from or destined for the blackhole MAC addresses.	Blackhole MAC address entries can filter out unauthorized users.

5.1.3.2 Elements and Functions of a MAC Address Table

Elements

Each entry in a MAC address table is uniquely identified by a MAC address and a VLAN ID or VSI. If a destination host is added to multiple VLANs or VSIs, one MAC address corresponds to multiple VLAN IDs or VSIs in the MAC address table. **Table 5-3** lists MAC address entries, which specify the outbound interfaces for packets with specified destination MAC addresses and VLAN IDs. For example, the first MAC address entry is used to forward the packets with destination MAC address 00e0-fc12-1234 and VLAN ID 10 through the outbound interface, interface 1.

Table 5-3 MAC address entries

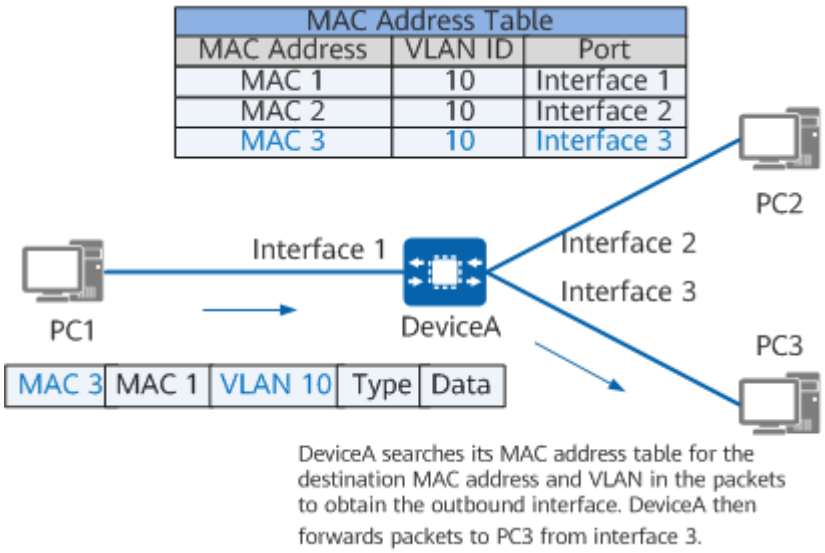
MAC Address	VLAN ID/VSI Name	Outbound Interface
00e0-fc12-1234	10	Interface 1
00e0-fc12-5678	20	Interface 2
00e0-fc12-1278	huawei	Interface 3

Functions

A MAC address table is used for forwarding unicast packets. In **Figure 5-1**, when packets sent from PC1 to PC3 reach DeviceA, DeviceA searches its MAC address

table for the outbound interface (interface 3) that matches the destination MAC address MAC 3 and VLAN 10 in the packets. DeviceA then forwards the packets to PC3 through interface 3.

Figure 5-1 Forwarding based on the MAC address table

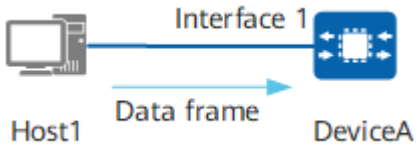


5.1.3.3 MAC Address Entry Learning and Aging

MAC Address Entry Learning

Most MAC address entries are dynamically created based on source MAC addresses learned from received data frames.

Figure 5-2 MAC address entry learning



In **Figure 5-2**, Host1 sends a data frame to DeviceA. When receiving the data frame, DeviceA obtains the source MAC address (Host1's MAC address) and VLAN ID of the frame.

- If Host1's MAC address does not exist in the MAC address table, DeviceA adds a new entry with Host1's MAC address, interface 1, and VLAN ID to the MAC address table.
- If Host1's MAC address exists in the MAC address table, DeviceA resets the aging timer of the MAC address entry and updates the entry.

NOTE

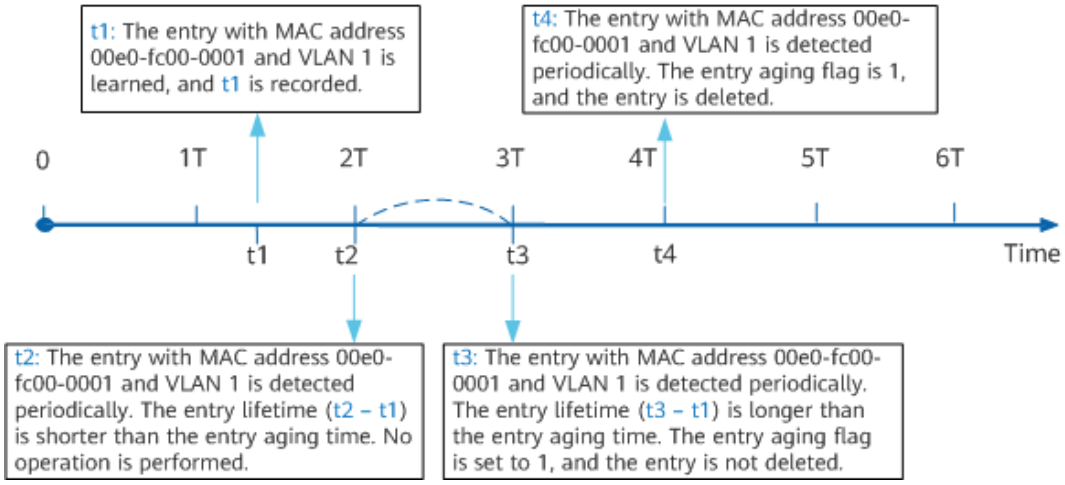
- All device interfaces belong to VLAN 1 by default. If the VLAN to which an interface belongs is not changed, the VLAN ID of the corresponding MAC address entry is VLAN 1.
- A device does not learn the BPDU MAC address, of which the format is 0180-c200-xxxx.

MAC address entry learning and update are triggered on a device only when the device receives data frames.

MAC Address Entry Aging

A device needs to update its MAC address table continuously to adapt to changing network topologies. The dynamic entries automatically created in a MAC address table are not always valid. Each entry has a lifecycle (aging time) and will be deleted if it is not updated within the aging time. If an entry is updated within the aging time, the aging timer for the entry is reset.

Figure 5-3 MAC address entry aging



As shown in **Figure 5-3**, the aging time of MAC address entries is set to 70 seconds in the following example.

At t1, packets with source MAC address 00e0-fc00-0001 and VLAN ID 1 arrive at an interface. Assume that the interface has been added to VLAN 1. If no entry with MAC address 00e0-fc00-0001 and VLAN 1 exists in the MAC address table, the MAC address is learned as a dynamic MAC address entry, and t1 is recorded as the entry update time.

The device checks all learned dynamic MAC address entries at an interval of T (60 seconds).

1. At t2, the device detects that the lifetime (t2 – t1) of the dynamic MAC address entry with MAC address 00e0-fc00-0001 and VLAN 1 is shorter than the aging time (70 seconds) of the MAC address entry, and does not perform any operation.

2. If no such packet enters the device between t2 and t4, the entry update time t1 remains unchanged.

3. At t3, the device detects that the lifetime (t3 – t1) of the dynamic MAC address entry with MAC address 00e0-fc00-0001 and VLAN 1 is longer than the aging time (70 seconds) of the MAC address entry, sets the aging flag of the MAC address entry to 1, and does not delete the MAC address entry.

4. At t4, the device detects that the aging flag of the dynamic MAC address entry with MAC address 00e0-fc00-0001 and VLAN 1 is 1, and deletes the MAC address entry.

5. If such packets enter the device between t2 and t4, the device records the current time to replace the entry update time t1, and sets the aging flag to 0. The device then repeats the aging process from t2 to t4.

As mentioned above, through automatic aging, the shortest time for a dynamic entry to exist in the MAC address table is the aging time configured on the device plus the time for the timer to execute one round (60 seconds), and the longest time is the aging time configured on the device plus the time for the timer to execute two rounds (120 seconds). You can set the aging time of dynamic MAC address entries to control their lifecycle in the MAC address table.

5.1.4 Default Settings for MAC Address Tables

Table 5-4 lists default settings for MAC address tables.

Table 5-4 Default Settings for MAC Address Tables

Parameter	Default Setting
Aging time of a dynamic MAC entry	300 seconds
MAC address learning	Enabled
Limit on the number of MAC addresses learned on an interface	2048
MAC address learning priority of an interface	0
MAC address flapping between interfaces with the same priority	Allowed
MAC address flapping detection	Enabled
MAC address-triggered ARP entry update	Disabled

5.1.5 Configuring a Static MAC Address Entry

Context

A device cannot distinguish packets from authorized and unauthorized users when it learns source MAC addresses of packets to maintain the MAC address table. This causes network risks. If an unauthorized user uses the MAC address of an authorized user as the source MAC address of attack packets and connects to another interface of the device, the device learns an incorrect MAC address entry. As a result, packets destined for the authorized user are forwarded to the unauthorized user. For security purposes, you can create static MAC address entries to bind MAC addresses of authorized users to specified interfaces. This prevents unauthorized users from intercepting the data of authorized users.

Static MAC address entries have the following characteristics:

- A static MAC address entry will not age out, and will not be lost after a system restart, and can only be deleted manually.
- The VLAN bound to a static MAC address entry must have been created and assigned to the interface bound to the entry.
- The MAC address in a static MAC address entry must be a unicast MAC address, not a multicast or broadcast MAC address.
- Static MAC address entries take precedence over dynamic MAC address entries. The system discards packets with flapping static MAC addresses.

  For details about configuration parameters, see huawei-mac.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the VLAN view.

```
vlan vlans vlan id vlan_id
```

Step 3 Configure a static MAC address entry.

```
mac-addresss mac-address address mac_address
out-interface-name port_name
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display vlan/vlans/vlan[id=*id*]/mac-addresss/** command to check the configured static MAC address entries.

5.1.6 Configuring a Blackhole MAC Address Entry

Context

Blackhole MAC address entries can be used to filter out invalid MAC addresses. To prevent a MAC address from being used to attack a user device or network, configure the MAC address of an untrusted user as a blackhole MAC address. After a device receives packets with a destination blackhole MAC address, the device directly discards these packets if the VLAN information carried in these packets matches that specified in a blackhole MAC address entry. For details about configuration parameters, see huawei-mac.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the VLAN view.

```
vlan vlans vlan id vlan_id
```


Step 3 Configure a blackhole MAC address entry.

```
mac-addresss mac-address address mac_address
black-hole [ null ]
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display vlan/vlans/vlan[id=*id*]/mac-addresss/** command to check the configured blackhole MAC address entries.

5.1.7 Configuring an Aging Time for Dynamic MAC Address Entries

Context

Dynamic MAC address entries do not need to be created manually and will age out automatically. You can set an aging time for dynamic MAC address entries.

- If the aging time is set too high, the MAC address table on a device may store many useless MAC address entries, and exhaust the address table. As a result, the device cannot learn new MAC addresses.
- If the aging time is set too low, MAC address entries in the MAC address table of a device may be deleted when the device receives packets destined for the MAC addresses. As a result, the device broadcasts a large number of data packets, increasing the network load.

Therefore, it is important that the aging time be set according to the situation. For example, if the devices on the network do not change often, you can set a longer aging time to prevent devices from broadcasting a large number of data packets. For details about configuration parameters, see huawei-mac.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the MAC view.

```
mac global-attribute
```

Step 3 Configure an aging time for dynamic MAC address entries.

```
aging-time timeValue
```

The aging time is an integer, in seconds. The default value is 300 seconds.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display this** command in the MAC view to check the aging time of dynamic MAC address entries.

NOTE

After an aging time is configured for dynamic MAC address entries, existing dynamic MAC address entries may be aged out in the next aging period.

5.1.8 Configuring MAC Address Flapping Prevention and Detection

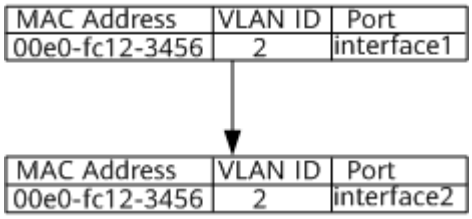
5.1.8.1 Understanding MAC Address Flapping

What Is MAC Address Flapping

MAC address flapping occurs when a MAC address is learned by two or three interfaces in the same VLAN, and a more recently learned MAC address entry overrides the earlier version. Normally, the first interface that learns a MAC address is the correct outbound interface, and is referred to as the original interface. An interface that learns the same MAC address later is called the move interface, and this is usually an interface on a loop (or where its downlink network has a loop). **Figure 5-4** shows how MAC address flapping occurs. In the MAC address entry with MAC address 00e0-fc12-3456 and VLAN ID 2, the outbound interface is changed from interface 1 to interface 2. MAC address flapping can lead to increased CPU usage on devices.

MAC address flapping frequently occurs on networks where a network loop exists. If this issue frequently occurs on your network, it may be due to a network loop or the result of unauthorized users attacking the network. In such cases, check the alarms and MAC address flapping records to quickly locate and eliminate the loops.

Figure 5-4 MAC address flapping



How to Prevent MAC Address Flapping

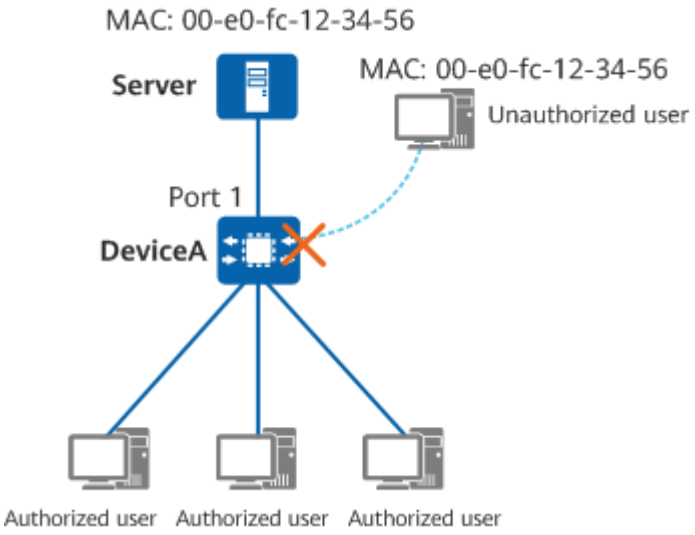
During network planning, you can use the following methods to prevent MAC address flapping:

- Increase the MAC address learning priority of an interface. If the same MAC addresses are learned by interfaces of different priorities, those learned by the interface with the highest priority overrides any learned by other interfaces.
- Prevent MAC address entries from being overridden on interfaces with the same priority. If an interface connected to an unauthorized network device

has the same priority as that connected to an authorized device, any MAC address learned by the former will not override the original correct MAC address. However, if the authorized device is powered off, the MAC address of the unauthorized device will be learned, and that of the authorized device cannot be learned once it is powered on again.

In **Figure 5-5**, port 1 of DeviceA is connected to a server. To prevent unauthorized users from connecting to DeviceA using the server's MAC address, set a high MAC address learning priority for port 1.

Figure 5-5 Networking of MAC address flapping prevention

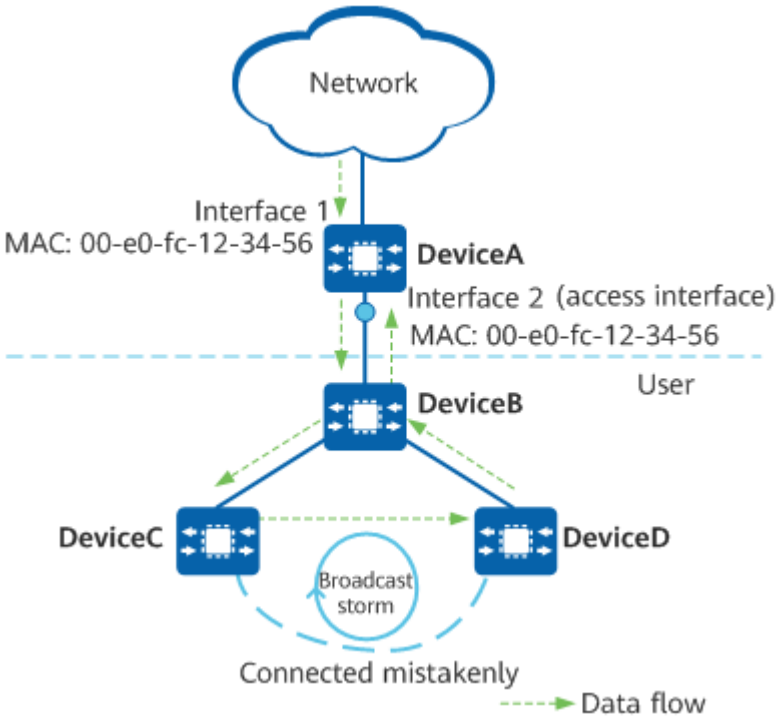


How to Detect MAC Address Flapping

MAC address flapping detection checks whether outbound interfaces in MAC address entries flap.

After MAC address flapping detection is enabled, a device can report an alarm when MAC address flapping occurs. The alarm contains the flapping MAC address, VLAN ID, and outbound interfaces between which the MAC address flaps. A loop may exist between the outbound interfaces. You can locate the cause by referring to the alarm's information. Alternatively, you can configure what action the device will take following MAC address flapping detection, and this can include quit-vlan (remove the interface from the VLAN) or error-down (shut down the interface), which ensures the device automatically removes the loop.

Figure 5-6 Networking of MAC address flapping detection



In **Figure 5-6**, a network loop occurs between DeviceB, DeviceC, and DeviceD because DeviceC and DeviceD are mistakenly connected using a network cable. When interface 1 of DeviceA receives a broadcast packet, DeviceA forwards the packet to DeviceB. The packet is then sent to interface 2 of DeviceA. After DeviceA is configured with MAC address flapping detection, it can detect that the source MAC address of the packet flaps from interface 1 to interface 2. If MAC address flapping continuously occurs, DeviceA reports a MAC address flapping alarm to inform users of maintenance.

Processing Mechanism After MAC Address Flapping Occurs

The device supports MAC address flapping detection by default. When MAC address flapping occurs on an interface, the device generates an alarm.

5.1.9 Maintaining MAC Address Tables

5.1.9.1 Displaying MAC Address Entries

During routine maintenance, you can run the following commands in the MD-CLI view to check MAC address information.

Table 5-5 Commands used to display MAC address entries

Operation	Command
Display the system MAC address.	**display /driver/global-attribute/system-mac-address**

Operation	Command
Display the number of MAC addresses.	**display /driver/global-attribute/system-mac-number**

5.1.9.2 Deleting MAC Address Entries

During routine maintenance, you can run the following commands to delete MAC address entries, including both static and dynamic MAC address entries.

NOTICE

Exercise caution when deleting MAC address entries. Deleted MAC address entries cannot be restored.

Table 5-6 Commands used to delete MAC address entries

Operation	View	Command
Delete static and blackhole MAC address entries.	System view	**remove /vlan/vlans/vlan[id=***id***]/mac-addresss/mac-address[address="***mac_address***"]**
Delete all dynamic MAC address entries learned by a specified interface.	System view	**reset-dynamic-macs-by-interface interface-name** *port_name*
Delete all dynamic MAC address entries in a specified VLAN.	System view	**reset-vlan-dynamic-macs vlan-id** *vlan_id*

5.2 VLAN Configuration

5.2.1 Overview of VLANs

Definition

Virtual Local Area Network (VLAN) technology logically divides a physical LAN into multiple broadcast domains, each of which is called a VLAN.

Purpose

VLAN was originally introduced to address the problems of conflicts, broadcast storms, data security risks that occurred on early Ethernet, which was itself

originally intended for small and simple networks using bus technology and carrier-sense multiple access/collision detection (CSMA/CD). These problems have become even more prominent as Ethernet has expanded to larger and more complex networks, with LANs carrying such diversified data as graphics, voice, and video:

- Conflicts: Multiple hosts on the network send frames at the same time, causing a conflict. Additional hosts lead to increased conflicts.
- Broadcast storms: Frames sent by any host on the network are sent to all other hosts, causing broadcast storms. More hosts lead to more severe broadcast storms.
- Data security risks: As all hosts on the network share a single data transmission channel, data security cannot be ensured. Increasingly complex data leads to greater security risks.

Using Layer 2 devices for fast Layer 2 switching can restrict data transmission within a LAN. However, this resolves only the conflicts.

To reduce broadcast traffic, you can configure different network segments to isolate devices, but the drawbacks of this solution include high costs. As such, VLAN technology was introduced.

VLAN technology allows a physical LAN to be divided into multiple logical LANs (multiple VLANs). Each VLAN functions as a separate broadcast domain, with devices in the same VLAN able to directly communicate with one another, while those in different VLANs cannot. As a result, broadcast packets are confined within a single VLAN, thereby strengthening network security.

Benefits

VLAN technology offers the following benefits:

- Confines each broadcast domain to a single VLAN, conserving bandwidth and improving network processing capabilities.
- Enhances LAN security. Frames in different VLANs are separately transmitted, so that hosts in a VLAN cannot directly communicate with those in another VLAN.
- Improves network robustness. A fault in one VLAN does not affect hosts in another VLAN.
- Allows for flexible virtual groups. VLAN technology allows hosts to be divided into different groups, and hosts in different geographical locations can be grouped together, simplifying network construction and maintenance.

5.2.2 Understanding VLANs

5.2.2.1 VLAN Tags

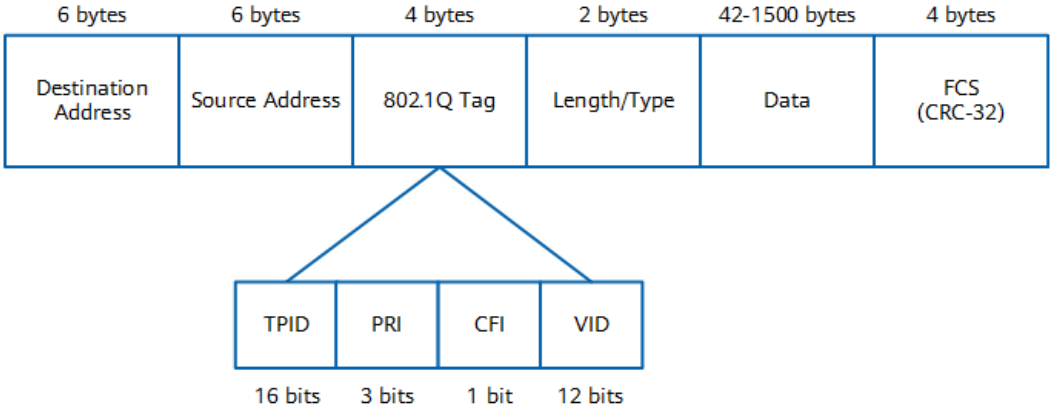
Definition

Each VLAN on a LAN is identified by a unique VLAN tag, which is also called an 802.1Q tag.

Format

IEEE 802.1Q adds a 4-byte 802.1Q tag between the Source Address field and the Length/Type field of an Ethernet frame. **Figure 5-7** shows the VLAN-tagged frame format defined in IEEE 802.1Q.

Figure 5-7 VLAN-tagged frame format defined in IEEE 802.1Q



An 802.1Q tag contains four fields:

- Tag protocol identifier (TPID): determines whether a VLAN frame carries an 802.1Q tag. This field is 16 bits long and defaults to 0x8100, which indicates an 802.1Q-tagged frame. A device that does not support 802.1Q discards 802.1Q-tagged frames.

  Device vendors can define their own TPID values. When the TPID value of a neighbor device is set to a value other than 0x8100, the TPID value of the local device must be changed to that of the neighbor device. This enables the local device to identify the frames sent by, and communicate with, the neighbor device.
- Priority (PRI): indicates the frame priority. This field is 3 bits long and its value ranges from 0 to 7, with a larger value indicating a higher priority. If network congestion occurs, a device preferentially sends frames with a higher priority.
- Canonical Format Indicator (CFI): indicates whether a MAC address is encapsulated in canonical format. This field is 1 bit long and can be set to 0 or 1 (0 by default). The value 0 indicates that the MAC address is encapsulated in canonical format while the value 1 indicates non-canonical format.
- VID: indicates the VLAN to which a frame belongs. This field is 12 bits long and ranges from 0 to 4095. The values 0 and 4095 are reserved, and therefore available VLAN IDs are in the range from 1 to 4094.

Frame Types

Each 802.1Q-capable device identifies the VLAN to which a frame belongs based on the VLAN ID, and processes the frame based on whether it carries a VLAN tag and the specific VLAN tag value. Frames are classified into the following types based on whether they carry VLAN tags:

- Tagged frame: a frame with a 4-byte 802.1Q tag

- Untagged frame: an original frame without a 4-byte 802.1Q tag

In most cases, devices process tagged and untagged frames differently:

- User hosts, servers, hubs, and unmanaged switches can only receive and send untagged frames.
- Switches, routers, firewalls, and access controllers (ACs) can send and receive both tagged and untagged frames.
- Voice terminals can send and receive the tagged or untagged frames of only one VLAN.

5.2.2.2 Default VLAN

The default VLAN ID of an interface is called the PVID. Each interface has a PVID.

5.2.2.3 Adding and Removing VLAN Tags

Interfaces process data frames as tagged or untagged based on their interface types and default VLANs. **Table 5-7** describes how interfaces process data frames.

NOTE

To improve the efficiency of data frame processing, all data frames inside a device carry VLAN tags so that the device can process them in a unified manner.

Table 5-7 Data frame processing modes of different interfaces

Interface Type	Processing a Received Frame	Processing a Frame to Be Sent
pvid	Checks whether the received frame carries a VLAN tag: • Tags the frame with its PVID if the frame does not carry a VLAN tag. • Accepts the frame only when the frame carries a VLAN tag with the same VLAN ID as the PVID. Otherwise, the frame is discarded.	Removes the PVID tag from the frame before sending it.

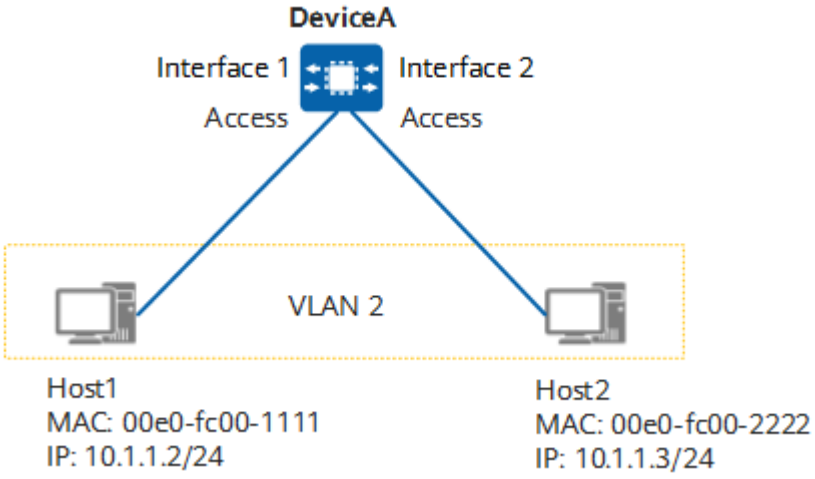
Interface Type	Processing a Received Frame	Processing a Frame to Be Sent
trunk-vlans	Checks whether the received frame carries a VLAN tag: • Tags the frame with its PVID if the frame does not carry a VLAN tag. • Accepts the frame if it carries a tag with a VLAN ID specified in the allowed VLAN ID list. Otherwise, the frame is discarded.	Checks the VLAN tag of the frame to be sent: • If the VLAN ID in the tagged frame is the same as the PVID of the interface and the VLAN is allowed, the interface removes the VLAN tag from the frame before sending it. • If the VLAN ID in the tagged frame is different from the PVID of the interface and the VLAN is allowed, the interface sends the frame without removing the carried tag. • If the VLAN ID in the tagged frame is different from the PVID of the interface and the VLAN is not allowed, the interface discards the frame.
untag-vlans	Checks whether the received frame carries a VLAN tag: • Tags the frame with its PVID if the frame does not carry a VLAN tag. • Accepts the frame if it carries a tag with a VLAN ID specified in the allowed VLAN ID list. Otherwise, the frame is discarded.	Checks the VLAN tag of the frame to be sent: • If the VLAN ID in the tagged frame is the same as the PVID of the interface and the VLAN is allowed, the interface removes the VLAN tag from the frame before sending it. • If the VLAN ID in the tagged frame is different from the PVID of the interface and the VLAN is allowed, the interface removes the VLAN tag from the frame before sending it. • If the VLAN ID in the tagged frame is different from the PVID of the interface and the VLAN is not allowed, the interface discards the frame.

5.2.2.4 Intra-VLAN Communication

Intra-VLAN Communication Through a Single Device

In **Figure 5-8**, Host1 and Host2 connect to the same device, belong to VLAN 2, and are located on the same network segment.

Figure 5-8 Intra-VLAN communication through a single device



When Host1 sends a packet to Host2, the packet is transmitted as follows (assuming that no forwarding entry is created on DeviceA):

1. Host1 determines that the destination IP address is on the same network segment as its IP address, and broadcasts an ARP Request packet to obtain the MAC address of Host2. The ARP Request packet carries the all-F destination MAC address and the destination IP address 10.1.1.3 (Host2's IP address).
2. When the packet reaches interface 1 on DeviceA, DeviceA determines that the ARP Request packet is untagged and adds a tag with VLAN ID 2 (which is the PVID of interface 1) to the packet. DeviceA then adds the mapping between the source MAC address, VLAN ID, and interface (00e0-fc00-1111, 2, interface 1) to its MAC address table.
3. As DeviceA does not find a MAC address entry matching the destination MAC address and VLAN ID of the ARP Request packet, it broadcasts the ARP Request packet through all interfaces that allow VLAN 2 (interface 2 in this example).
4. Before sending the ARP Request packet, interface 2 on DeviceA removes the tag with VLAN ID 2 from the packet.
5. Host2 receives the ARP Request packet from interface 2 and records the mapping between the MAC address and the IP address of Host1 in its ARP table. Host2 then compares the destination IP address with its own IP address. If they are the same, Host2 sends an ARP Reply packet carrying Host2's MAC address (00e0-fc00-2222) and Host1's IP address (10.1.1.2) as the destination IP address.
6. After receiving the ARP Reply packet, interface 2 on DeviceA tags the packet with VLAN ID 2.

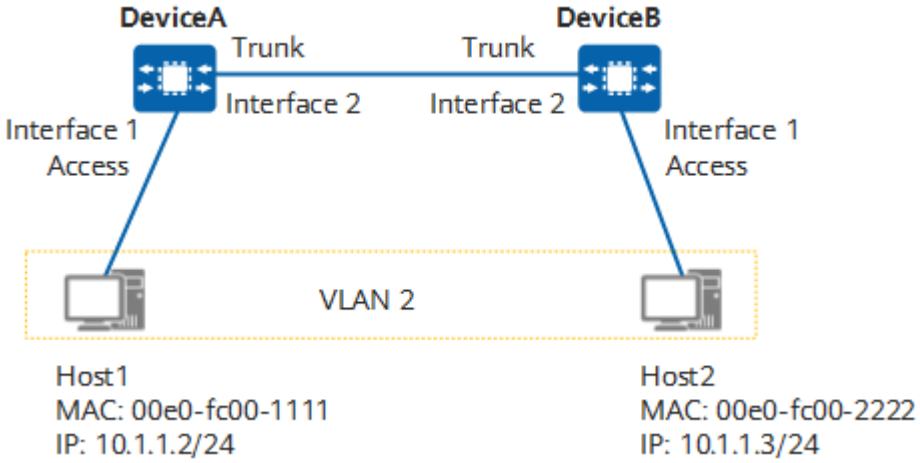
7. DeviceA adds the mapping between the source MAC address, VLAN ID, and interface (00e0-fc00-2222, 2, interface 2) to its MAC address table, and then searches for an entry in its MAC address table based on the destination MAC address and VLAN ID (00e0-fc00-1111, 2). DeviceA finds the matching entry and sends the ARP Reply packet through interface 1.
8. Before sending the ARP Reply packet through interface 1, DeviceA removes the tag with VLAN ID 2 from the packet based on the interface configuration.
9. Host1 receives the ARP Reply packet from interface 1 and records the mapping between the MAC address and the IP address of Host2 in its ARP table.

Intra-VLAN Communication Through Multiple Devices

In **Figure 5-9**, Host1 and Host2 connect to different devices, belong to VLAN 2, and are located on the same network segment. DeviceA and DeviceB are connected using a trunk link over which frames tagged with VLAN ID 2 can be identified and transmitted between them.

Users in the same VLAN but on different network segments cannot communicate with each other at Layer 2 through DeviceA and DeviceB. The VLANIF technology can be used to implement Layer 3 communication between them.

Figure 5-9 Intra-VLAN communication through multiple devices



When Host1 sends a packet to Host2, the packet is transmitted as follows (assuming that no forwarding entry is created on DeviceA and DeviceB):

1. Host1 determines that the destination IP address is on the same network segment as its IP address, and broadcasts an ARP Request packet to obtain the MAC address of Host2. The ARP Request packet carries the all-F destination MAC address and destination IP address 10.1.1.3 (Host2's IP address).
2. When the packet reaches interface 1 on DeviceA, DeviceA determines that the ARP Request packet is untagged and adds a tag with VLAN ID 2 (which is the PVID of interface 1) to the packet. DeviceA then adds the mapping between the source MAC address, VLAN ID, and interface (00e0-fc00-1111, 2, interface 1) to its MAC address table.

3. As DeviceA does not find a MAC address entry matching the destination MAC address and VLAN ID of the ARP Request packet, it broadcasts the ARP Request packet through all interfaces that allow VLAN 2 (interface 2 in this example).
4. Interface 2 on DeviceA transparently transmits the ARP Request packet to interface 2 on DeviceB without removing the packet's VLAN tag, as the VLAN ID of the packet is different from the PVID (which is 1 in this example) of interface 2 on DeviceA.
5. After receiving the ARP Request packet, interface 2 on DeviceB determines that VLAN 2 is allowed and accepts the packet.
6. As DeviceB does not find a MAC address entry matching the destination MAC address and VLAN ID of the ARP Request packet, it broadcasts the ARP Request packet through all interfaces that allow VLAN 2 (interface 1 in this example).
7. Before sending the ARP Request packet, interface 1 on DeviceB removes the tag with VLAN ID 2 from the packet.
8. Host2 receives the ARP Request packet from interface 1 on DeviceB and records the mapping between the MAC address and IP address of Host1 in its ARP table. Host2 then compares the destination IP address with its own IP address. If they are the same, Host2 sends an ARP Reply packet carrying Host2's MAC address (00e0-fc00-2222) and Host1's IP address (10.1.1.2) as the destination IP address.
9. After interface 1 on DeviceB receives the ARP Reply packet, DeviceB adds a tag with VLAN ID 2 to the packet, and then adds the mapping between the source MAC address, VLAN ID, and interface (00e0-fc00-2222, 2, interface 1) to its MAC address table.
10. DeviceB transparently transmits the ARP Reply packet of Host2 through interface 2 to interface 2 on DeviceA. This is because interface 2 on DeviceB is a trunk interface and its PVID (which is 1 in this example) is different from the VLAN ID of the packet. As a result, interface 2 on DeviceB does not remove the VLAN tag of the packet.
11. After receiving the ARP Reply packet, interface 2 on DeviceA determines that VLAN 2 is an allowed VLAN and accepts the packet.
12. DeviceA adds the mapping between the source MAC address, VLAN ID, and interface (00e0-fc00-2222, 2, interface 2) to its MAC address table, and then searches for an entry in its MAC address table based on the destination MAC address and VLAN ID (00e0-fc00-1111, 2). DeviceA finds the matching entry and sends the ARP Reply packet through interface 1.
13. Before sending the ARP Reply packet through interface 1, DeviceA removes the tag with VLAN ID 2 from the packet based on the interface configuration.
14. Host1 receives the ARP Reply packet from interface 1 and records the mapping between the MAC address and the IP address of Host2 in its ARP table.

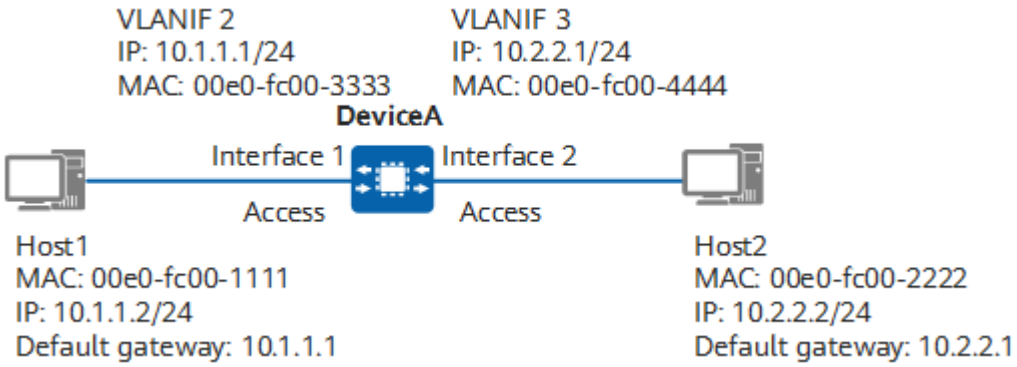
In addition to transmitting frames from multiple VLANs, a trunk link can transparently transmit frames without adding or removing VLAN tags of packets.

5.2.2.5 Inter-VLAN Communication

Inter-VLAN Communication Through a Single Device (Using VLANIF Interfaces)

In **Figure 5-10**, Host1 and Host2 connect to the same device, are located on different network segments, and belong to VLAN 2 and VLAN 3, respectively. After VLANIF 2 and VLANIF 3 are created on DeviceA and configured with IP addresses, the default gateway addresses of Host1 and Host2 are set to the IP addresses of VLANIF 2 and VLANIF 3, respectively.

Figure 5-10 Using VLANIF interfaces to implement inter-VLAN communication through a single device



When Host1 sends a packet to Host2, the packet is transmitted as follows (assuming that no forwarding entry is created on DeviceA):

1. Host1 determines that the destination IP address is on a different network segment from its own IP address, and therefore sends an ARP Request packet to request the gateway MAC address. The ARP Request packet carries the destination IP address 10.1.1.1 (gateway's IP address) and all-F destination MAC address.
2. When the ARP Request packet reaches interface 1 on DeviceA, DeviceA tags the packet with VLAN ID 2 (PVID of interface 1). DeviceA then records the mapping between the source MAC address, VLAN ID, and inbound interface (00e0-fc00-1111, 2, interface 1) in its MAC address table.
3. DeviceA determines that the packet is an ARP Request packet and the destination IP address is that of its own VLANIF 2. DeviceA then encapsulates VLANIF 2's MAC address 00e0-fc00-3333 into the ARP Reply packet and removes the tag with VLAN ID 2 before sending it through interface 1. In addition, DeviceA records the mapping between the IP address and MAC address of Host1 in its ARP table.
4. After receiving the ARP Reply packet from DeviceA, Host1 records the mapping between the IP address and MAC address of VLANIF 2 on DeviceA in its ARP table and sends a packet to DeviceA. The packet carries the destination MAC address 00e0-fc00-3333 and destination IP address 10.2.2.2 (Host2's IP address).
5. After receiving the packet, interface 1 on DeviceA tags the packet with VLAN ID 2.

6. DeviceA updates its MAC address table based on the source MAC address, VLAN ID, and inbound interface of the packet, and compares the destination MAC address of the packet with the MAC address of VLANIF 2. If they are the same, DeviceA determines that the packet should be forwarded at Layer 3 and searches for a Layer 3 forwarding entry based on the destination IP address. If no entry is found, DeviceA sends the packet to the CPU, which then searches for a routing entry to forward the packet.
7. The CPU searches the routing table based on the destination IP address of the packet and determines that the destination IP address matches a directly connected network segment (the network segment where VLANIF 3 is located). The CPU continues to search the ARP table but finds no matching ARP entry. As a result, DeviceA broadcasts an ARP Request packet with the destination address of 10.2.2.2 to all interfaces in VLAN 3. Before sending the ARP Request packet from interface 2, DeviceA removes the tag with VLAN ID 2.
8. After receiving the ARP Request packet, Host2 determines that the destination IP address in the packet is its own IP address and sends an ARP Reply packet carrying its own MAC address. At the same time, Host2 records the mapping between the MAC address and IP address of VLANIF 3 in its ARP table.
9. After interface 2 on DeviceA receives the ARP Reply packet, DeviceA tags the packet with VLAN ID 3 and records the mapping between the MAC address and IP address of Host2 in its ARP table. Before forwarding the packet from Host1 to Host2, DeviceA removes the tag with VLAN ID 3 from the packet. At the same time, DeviceA records the mapping between the Host2's IP address, MAC address, VLAN ID, and outbound interface in its Layer 3 forwarding table.

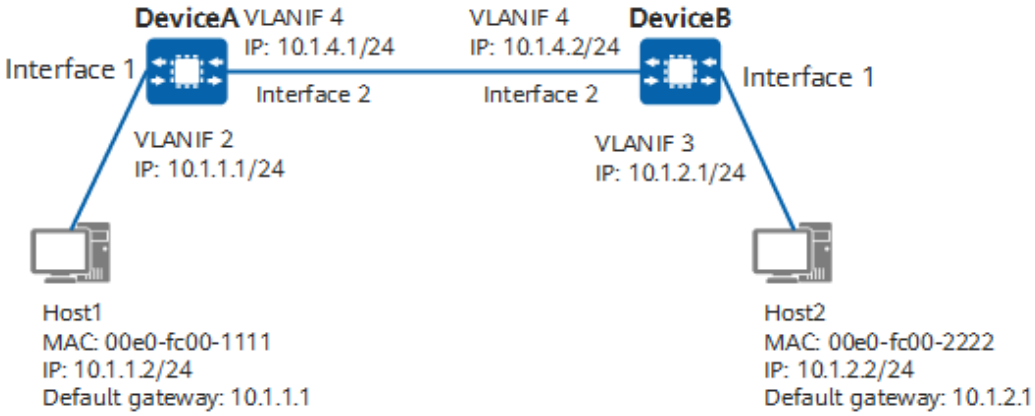
At this point, Host1 accesses Host2 successfully. The same process is used for Host2 to access Host1.

Inter-VLAN Communication Through Multiple Devices Using VLANIF Interfaces

When hosts in different VLANs connect to multiple devices, you need to configure static routes or a dynamic routing protocol in addition to configuring VLANIF interfaces and their IP addresses, as the IP addresses of VLANIF interfaces can only be used to generate direct routes.

In **Figure 5-11**, Host1 and Host2 connect to different devices, are located on different network segments, and belong to VLAN 2 and VLAN 3, respectively. DeviceA and DeviceB connect to hosts using access interfaces and connect to each other using trunk interfaces. On DeviceA, VLANIF 2 and VLANIF 4 are created and configured with IP addresses 10.1.1.1 and 10.1.4.1, respectively. On DeviceB, VLANIF 3 and VLANIF 4 are created and configured with IP addresses 10.1.2.1 and 10.1.4.2, respectively. Static routes are configured on DeviceA and DeviceB. On DeviceA, the destination network segment in the static route is 10.1.2.0/24 and the next-hop address is 10.1.4.2. On DeviceB, the destination network segment in the static route is 10.1.1.0/24 and the next-hop address is 10.1.4.1.

Figure 5-11 Using VLANIF interfaces to implement inter-VLAN communication through multiple devices



When Host1 sends a packet to Host2, the packet is transmitted as follows (assuming that no forwarding entry is created on DeviceA and DeviceB):

1. The first six steps are the same as steps 1 to 6 in **Inter-VLAN Communication Through a Single Device (Using VLANIF Interfaces)**. After those steps are complete, DeviceA sends the packet to its CPU which then searches the routing table to forward the packet.
2. The CPU of DeviceA searches the routing table based on the destination IP address 10.1.2.2, and finds a static route with the destination network segment of 10.1.2.0/24 and the next-hop address of 10.1.4.2. The CPU continues to search the ARP table but finds no matching ARP entry. Therefore, DeviceA broadcasts an ARP Request packet with the destination address 10.1.4.2 to all interfaces in VLAN 4. Interface 2 on DeviceA transparently transmits the ARP Request packet to interface 2 on DeviceB without removing the tag from the packet.
3. After the ARP Request packet reaches DeviceB, DeviceB determines that the destination IP address of the ARP Request packet is the IP address of its own VLANIF 4. DeviceB then sends an ARP Reply packet with the MAC address of VLANIF 4 to DeviceA.
4. Interface 2 on DeviceB transparently transmits the ARP Reply packet to DeviceA. After DeviceA receives the ARP Reply packet, it records the mapping between the MAC address and IP address of VLANIF 4 in its ARP table.
5. Before forwarding the packet of Host1 to DeviceB, DeviceA changes the destination MAC address of the packet to the MAC address of VLANIF 4 on DeviceB, and the source MAC address to the MAC address of its own VLANIF 4. In addition, DeviceA records the forwarding entry (10.1.2.0/24, destination MAC address, VLAN, and outbound interface) in its Layer 3 forwarding table. Similarly, the packet is transparently transmitted to interface 2 on DeviceB.
6. After DeviceB receives the packet of Host1 forwarded by DeviceA, steps 6 to 9 in **Inter-VLAN Communication Through a Single Device (Using VLANIF Interfaces)** are performed. In addition, DeviceB records the forwarding entry (Host2's IP address, MAC address, VLAN, and outbound interface) in its Layer 3 forwarding table.

At this point, Host1 accesses Host2 successfully. The same process is used for Host2 to access Host1.

5.2.3 Configuration Precautions for VLAN

Licensing Requirements

VLAN is not under license control.

Hardware Requirements

Table 5-8 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

5.2.4 Default Settings for VLANs

Table 5-9 describes the default settings for VLAN.

Table 5-9 Default settings for VLAN

Parameter	Default Setting
Default VLAN	VLAN 1
VLAN to which interfaces are added	The default PVID of GE0/0/1, GE0/0/2, GE0/0/3, GE0/0/4, GE0/0/5, GE0/0/6, GE0/0/7, and GE0/0/8 is 1, and the default value of **untag-vlans** is 1.

5.2.5 Creating and Deleting a VLAN

Procedure

- **Creating a VLAN**
  a. Enter the edit-config view.
     edit-config
  b. Create a VLAN.
     vlan vlans vlan id *vlan_id*
  c. Commit the configuration.
     commit
- **Deleting a VLAN**

a. Enter the edit-config view.
edit-config
b. Delete a VLAN.
remove vlan/vlans/vlan[id="*vlan_id***"]**
c. Commit the configuration.
commit

----End

Verifying the Configuration

Run the **display /vlan/vlans/vlan/ all** command to check all created VLANs.

5.2.6 Modifying the Default VLAN

Procedure

Step 1 Enter the edit-config view.
edit-config

Step 2 Enter the interface view.
ifm interfaces interface name *interface_name*

Step 3 Enter the Ethernet view.
ethernet main-interface l2-attribute

Step 4 (Optional) Configure the link type.
link-type { **access** | **trunk** | **hybrid** }

Step 5 (Optional) Configure the allowed VLAN ID.
trunk-vlans *vlan_id*

Step 6 (Optional) Configure the allowed untagged VLAN ID.
untag-vlans *vlan_id*

Step 7 Configure the default VLAN ID.
pvid *vlan_id*

Step 8 Commit the configuration.
commit

----End

Verifying the Configuration

Run the **display /ifm/interfaces/interface[name="***interface_name***"]/ethernet/main-interface/l2-attribute/ all** command to check the PVID of an interface.

5.2.7 Configuring Intra-VLAN Communication

5.2.7.1 Understanding Intra-VLAN Communication

This section describes VLAN assignment modes. For details about the implementation of intra-VLAN communication, see **Intra-VLAN Communication**.

Table 5-10 VLAN assignment modes

VLAN Assignment Mode	Fundamentals	Advantages	Disadvantages
Interface-based VLAN assignment	VLANs are assigned based on interface numbers. A network administrator pre-configures a unique PVID for each interface (the VLAN to which the interface belongs by default). • When a data frame arrives at an interface, the frame is tagged with the PVID of the interface if it does not carry any VLAN tag. • If the data frame already carries a VLAN tag, the device does not add more VLAN tags to the frame even if the interface has a PVID configured. Different types of interfaces process frames in different ways.	Defining VLAN members is simple.	The network administrator must reconfigure VLANs when VLAN members change.

5.2.7.2 Configuring Interface-based VLAN Assignment

Prerequisites

Before configuring interface-based VLAN assignment, you have completed the following task:

- Create VLANs. For details, see **5.2.5 Creating and Deleting a VLAN**.

Context

Interface-based VLAN assignment is the easiest and most effective method for assigning VLANs. After you add an interface to a VLAN, the interface can only forward packets from that VLAN. This limits broadcast packets to a single VLAN, as hosts in the same VLAN can directly communicate with each other at Layer 2, while those in different VLANs cannot.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```


Step 2 Enter the interface view.

ifm interfaces interface name *interface_name*

Step 3 Enter the l2-attribute view of the interface.

ethernet main-interface l2-attribute

Step 4 Configure the Layer 2 Ethernet interface attribute.

link-type { **access** | **hybrid** | **trunk** }

Step 5 (Optional) Configure the interface not to remove the VLAN tag from outgoing packets.

trunk-vlans *vlan-range*

Step 6 (Optional) Configure the interface to remove the VLAN tag from outgoing packets.

untag-vlans *vlan-range*

Step 7 Commit the configuration.

commit

----End

Verifying the Configuration

- Run the **display /ifm/interfaces/interface[name="***br_name***"]/ethernet/main-interface/l2-attribute/ all** command to check information about the VLAN to which an interface is added.
- Run the **display /vlan/vlans/vlan[id=***vlan_id***]/member-ports/ all** command to check member interfaces of a VLAN.

5.2.7.3 Example for Configuring Interface-based VLAN Assignment to Implement Intra-VLAN Communication (Through a Single Device)

Networking Requirements

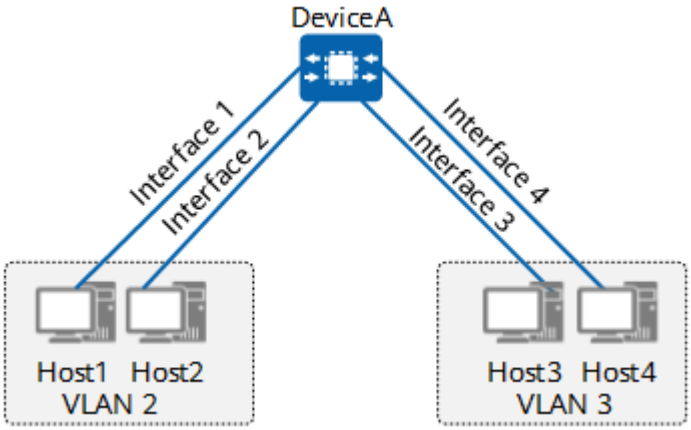
On DeviceA in **Figure 5-12**, the interfaces connected to Host1 and Host2 are added to VLAN 2, and the interfaces connected to Host3 and Host4 are added to VLAN 3. With this configuration, hosts in the same VLAN can directly communicate with each other at Layer 2, but hosts in different VLANs cannot. Specifically:

- Host1 and Host2 can communicate with each other, and Host3 and Host4 can communicate with each other.
- Host1 and Host2 cannot communicate with Host3 and Host4 in VLAN 3.

Figure 5-12 Networking diagram of configuring interface-based VLAN assignment for intra-VLAN communication through a single device

NOTE

In this example, interface 1, interface 2, interface 3, and interface 4 represent GE0/0/0, GE0/0/1, GE0/0/2, and GE0/0/3, respectively.



Procedure

Step 1 Create VLANs.

```
MDCLI> edit-config
MDCLI> vlan vlans vlan id 2
MDCLI> quit 2
MDCLI> vlans vlan id 3
MDCLI> quit 2
MDCLI> commit
```

Step 2 Add interfaces to VLANs.

Add GE0/0/1 and GE0/0/2 to VLAN 2.

```
MDCLI> ifm interfaces interface name GE0/0/1
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 7
MDCLI> ifm interfaces interface name GE0/0/2
MDCLI> link-type hybrid
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 2
MDCLI> commit
```

Add GE0/0/3 and GE0/0/4 to VLAN 3.

```
MDCLI> ifm interfaces interface name GE0/0/3
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> pvid 3
MDCLI> untag-vlans 3
MDCLI> quit 7
MDCLI> ifm interfaces interface name GE0/0/4
MDCLI> link-type hybrid
MDCLI> pvid 3
MDCLI> untag-vlans 3
MDCLI> quit 2
MDCLI> commit
```

----End

Verifying the Configuration

Run the **display /vlan/vlans/vlan[id=*vlan_id*]/member-ports/ all** command to check member interfaces of VLANs.


```
MDCLI> display vlan/vlans/vlan[id=2]/member-ports/ all
{
  "member-port": [
    {
      "interface-name": "GE0/0/1",
      "access-type": "hybrid",
      "state": "down",
      "tag-mode": "untag"
    },
    {
      "interface-name": "GE0/0/2",
      "access-type": "hybrid",
      "state": "up",
      "tag-mode": "untag"
    }
  ]
}
[(gl)user@HUAWEI]
MDCLI> display vlan/vlans/vlan[id=3]/member-ports/ all
{
  "member-port": [
    {
      "interface-name": "GE0/0/3",
      "access-type": "hybrid",
      "state": "down",
      "tag-mode": "untag"
    },
    {
      "interface-name": "GE0/0/4",
      "access-type": "hybrid",
      "state": "up",
      "tag-mode": "untag"
    }
  ]
}
```

Hosts in VLAN 2 cannot ping hosts in VLAN 3, but those in the same VLAN can ping each other.

5.2.7.4 Example for Configuring Interface-based VLAN Assignment to Implement Intra-VLAN Communication (Through Multiple Devices)

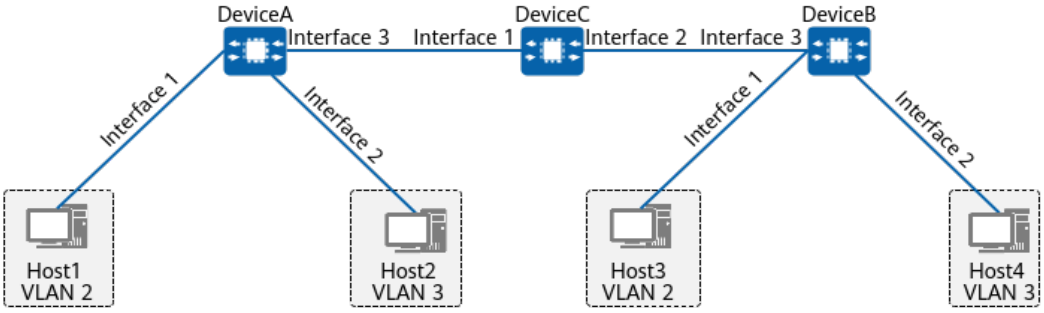
Networking Requirements

In **Figure 5-13**, Host1 and Host3 are added to VLAN 2, and Host2 and Host4 are added to VLAN 3. The interfaces on the link between DeviceA and DeviceC and those on the link between DeviceC and DeviceB allow packets sourced from VLAN 2 and VLAN 3 to pass through. This ensures that hosts in the same VLAN on DeviceA and DeviceB can directly communicate with each other at Layer 2, but hosts in different VLANs cannot.

Figure 5-13 Networking diagram of configuring interface-based VLAN assignment for intra-VLAN communication through multiple devices

NOTE

In this example, interface 1, interface 2, and interface 3 represent GE0/0/1, GE0/0/2, and GE0/0/3, respectively.



Procedure

Step 1 On DeviceA and DeviceB, configure the interfaces connecting to hosts as access interfaces, add Host1 and Host3 to VLAN 2, and add Host2 and Host4 to VLAN 3.

Configure DeviceA.

```
MDCLI> edit-config
MDCLI> vlan vlans vlan id 2
MDCLI> quit 2
MDCLI> vlan vlans vlan id 3
MDCLI> quit 2
MDCLI> commit
MDCLI> quit 3
MDCLI> ifm interfaces interface name GE0/0/1
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 7
MDCLI> ifm interfaces interface name GE0/0/2
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 2
MDCLI> commit
```

Configure DeviceB.

```
MDCLI> edit-config
MDCLI> vlan vlans vlan id 2
MDCLI> quit 2
MDCLI> vlans vlan id 3
MDCLI> quit 2
MDCLI> commit
MDCLI> quit 3
MDCLI> ifm interfaces interface name GE0/0/1
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 7
MDCLI> ifm interfaces interface name GE0/0/2
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 2
MDCLI> commit
```

Step 2 Configure the link between DeviceA and DeviceC and that between DeviceB and DeviceC as trunk links.

Configure DeviceA.


```
MDCLI> edit-config
MDCLI> ifm interfaces interface name GE0/0/3
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> trunk-vlans 2-3
MDCLI> commit
```

Configure DeviceB.

```
MDCLI> edit-config
MDCLI> ifm interfaces interface name GE0/0/3
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> trunk-vlans 2-3
MDCLI> commit
```

Configure DeviceC.

```
MDCLI> edit-config
MDCLI> vlan vlans vlan id 2
MDCLI> quit 2
MDCLI> vlans vlan id 3
MDCLI> quit 3
MDCLI> commit
MDCLI> ifm interfaces interface name GE0/0/1
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> trunk-vlans 2-3
MDCLI> quit 7
MDCLI> ifm interfaces interface name GE0/0/2
MDCLI> ethernet main-interface l2-attribute
MDCLI> link-type hybrid
MDCLI> trunk-vlans 2-3
MDCLI> commit
```

----End

Verifying the Configuration

Run the **display /vlan/vlans/vlan[id=*vlan_id*]/member-ports/ all** command to check member interfaces of VLANs. The following example uses the command output on DeviceA.

```
MDCLI> display vlan/vlans/vlan[id=2]/member-ports/ all
{
  "member-port": [
    {
      "interface-name": "GE0/0/1",
      "access-type": "hybrid",
      "state": "down",
      "tag-mode": "untag"
    },
    {
      "interface-name": "GE0/0/2",
      "access-type": "hybrid",
      "state": "up",
      "tag-mode": "untag"
    }
  ]
}
[(gl)user@HUAWEI]
MDCLI> display vlan/vlans/vlan[id=3]/member-ports/ all
{
  "member-port": [
    {
      "interface-name": "GE0/0/2",
      "access-type": "hybrid",
      "state": "down",
```



```
    "tag-mode": "untag"
   },
   {
    "interface-name": "GE0/0/3",
    "access-type": "hybrid",
    "state": "down",
    "tag-mode": "untag"
   }
  ]
}
```


Hosts in VLAN 2 can ping one another, as can those in VLAN 3. However, hosts in VLAN 2 cannot ping hosts in VLAN 3.

5.2.8 Configuring Inter-VLAN Communication

5.2.8.1 Understanding Inter-VLAN Communication

The VLANIF technology is commonly used to implement inter-VLAN communication across network segments.

A VLANIF interface is a Layer 3 logical interface that implements Layer 3 communication between users in different VLANs across different network segments. This technology is most commonly used to implement inter-VLAN communication because it is easy to configure.

Each VLANIF interface corresponds to a VLAN. After an IP address is configured for a VLANIF interface, the VLANIF interface becomes the gateway of the user hosts within that VLAN and forwards packets across network segments at Layer 3.

If users on multiple network segments need to communicate with each other, a primary IP address and multiple secondary IP addresses need to be configured on a VLANIF interface.

5.2.8.2 Configuring VLANIF Interfaces to Implement Inter-VLAN Communication

Prerequisites

Before configuring VLANIF interfaces to implement inter-VLAN communication, you have completed the following task:

Create VLANs. For details, see **5.2.5 Creating and Deleting a VLAN**.

Context

A VLANIF interface is a Layer 3 logical interface that implements Layer 3 communication between users in different VLANs across different network segments. This technology is most commonly used to implement inter-VLAN communication because it is easy to configure.

Each VLANIF interface corresponds to a VLAN. After an IP address is configured for a VLANIF interface, the VLANIF interface becomes the gateway of the user hosts within that VLAN and forwards packets across network segments at Layer 3.

For details about configuration parameters, see huawei-ifm.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Create a VLANIF interface and enter the VLANIF interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Configure an IP address for the VLANIF interface.

```
ip addresses address ip ip-address
mask mask
type main
```

If users on multiple network segments need to communicate with each other, a primary IP address and multiple secondary IP addresses need to be configured on a VLANIF interface. In addition, ensure that the IP addresses of VLANIF interfaces are on different network segments.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display /ifm/interfaces/interface[name="***interface-name***"]/dynamic/all** command to check the status of a VLANIF interface.

5.2.8.3 Example for Configuring VLANIF Interfaces to Implement Inter-VLAN Communication (Through a Single Device)

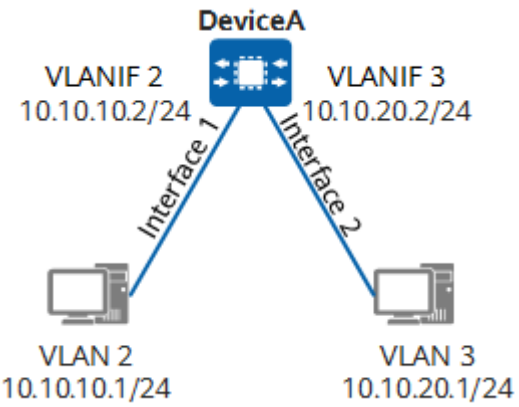
Networking Requirements

On the network shown in the figure, the two hosts connected to DeviceA are located on different network segments and belong to VLAN 2 and VLAN 3, respectively. Both hosts need to communicate with each other.

Figure 5-14 Network diagram of configuring VLANIF interfaces to implement inter-VLAN communication (through a single device)

NOTE

In this example, interface 1 and interface 2 represent GE0/0/1 and GE0/0/2, respectively.



Procedure

Step 1 Create VLANs.

```
MDCLI> edit-config
MDCLI> vlan vlans vlan id 2
MDCLI> quit 2
MDCLI> vlan vlans vlan id 3
MDCLI> quit 2
MDCLI> commit
```

Step 2 Add interfaces to VLANs.

```
MDCLI> ifm interfaces interface name GE0/0/1
MDCLI> ethernet main-interface l2-attribute
MDCLI> pvid 2
MDCLI> untag-vlans 2
MDCLI> quit 4
MDCLI> ifm interfaces interface name GE0/0/2
MDCLI> ethernet main-interface l2-attribute
MDCLI> pvid 3
MDCLI> untag-vlans 3
MDCLI> q 6
MDCLI> commit
```

Step 3 Configure IP addresses for VLANIF interfaces.

```
MDCLI> ifm interfaces interface name Vlanif2
MDCLI> ipv4 addresses address ip 10.10.10.2
MDCLI> mask 255.255.255.0 type main
MDCLI> commit
MDCLI> quit 6
MDCLI> ifm interfaces interface name Vlanif3
MDCLI> ipv4 addresses address ip 10.10.20.2
MDCLI> mask 255.255.255.0 type main
MDCLI> commit
```

----End

Verifying the Configuration

Set the IP address of the host in VLAN 2 to 10.10.10.1/24 and default gateway address to 10.10.10.2/24 (IP address of VLANIF 2), and set the IP address of the host in VLAN 3 to 10.10.20.1/24 and default gateway address to 10.10.20.2/24 (IP address of VLANIF 3). After the configuration is complete, hosts in VLAN 2 and VLAN 3 can ping each other.

6 WAN Access Configuration

6.1 PPP Configuration

Configuring PPP enables PPPoE dial-up access to the Internet and connectivity between campus networks.

6.1.1 Overview of PPP

Definition

Point-to-Point Protocol (PPP) is a link-layer protocol used to transmit point-to-point data over full-duplex synchronous or asynchronous links.

Purpose

PPP is built on the Serial Line Internet Protocol (SLIP). SLIP supports only the asynchronous transfer mode (ATM), transmits only IP packets, and does not support negotiation (such as negotiation of network-layer attributes including IP addresses of the two ends). Such shortfalls are why SLIP is gradually being replaced by PPP.

PPP has the following advantages:

- PPP supports both synchronous and asynchronous links.
- PPP features high extensibility. For example, PPP can be extended as Point-to-Point Protocol over Ethernet (PPPoE) when PPP packets need to be transmitted over an Ethernet.
- PPP uses Link Control Protocol (LCP) to negotiate link-layer parameters.

- PPP uses Network Control Protocols (NCPs) such as IP Control Protocol (IPCP) and Internetwork Packet Exchange Control Protocol (IPXCP) to negotiate network-layer parameters.
- PPP supports the Challenge Handshake Authentication Protocol (CHAP) and Password Authentication Protocol (PAP) to ensure network security.
- PPP has no retransmission mechanism, reducing the network cost and accelerating packet transmission.

6.1.2 Understanding PPP

6.1.2.1 Typical Networking of PPP

On a device that functions as the egress gateway of an enterprise, the LAN-side interfaces connect to hosts on the intranet, and a WAN-side interface connects to a carrier's device. The carrier's device can be an optical line terminal (OLT) or a wireless base station, depending on the WAN-side interface type.

PPP can be used in the following scenarios:

- PPP links are widely used for communication between enterprise branches and their headquarters. For example, in the following networking diagram, DeviceA in an enterprise branch connects to DeviceB in an enterprise headquarter through their WAN interfaces over a PPP link. DeviceA obtains an IP address through IPCP negotiation for connecting to the WAN.

  Figure 6-1 Communication over a PPP link

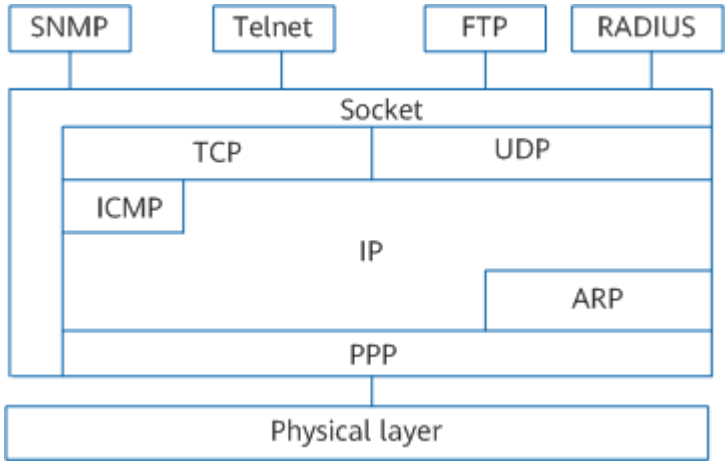
- PPP can be used with other technologies to provide various services, such as PPPoE.

6.1.2.2 PPP Packet Format

Basic Architecture of PPP

PPP is used at the data link layer of the TCP/IP protocol suite for point-to-point data transmission over full-duplex synchronous or asynchronous links.

Figure 6-2 Location of PPP in the protocol suite



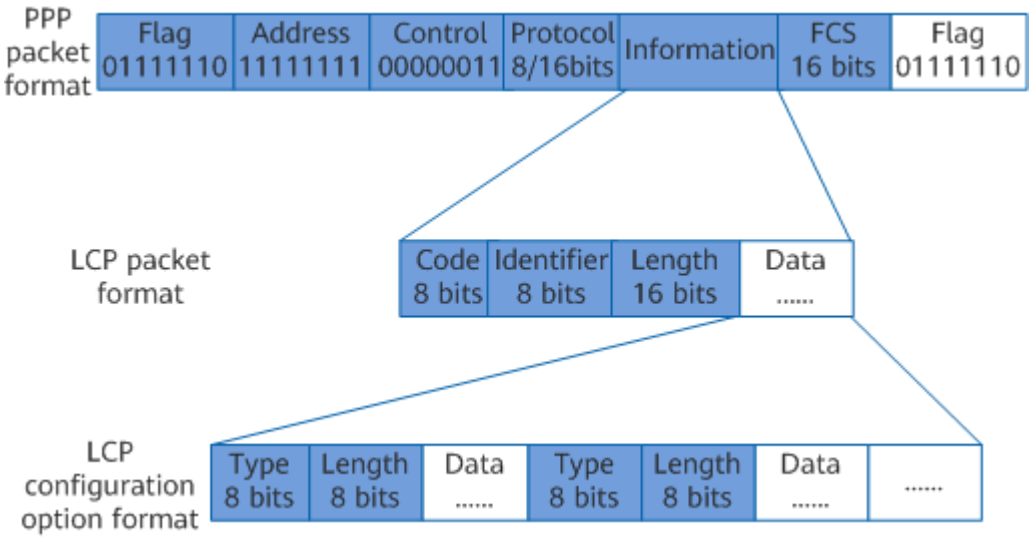
PPP consists of three types of protocols:

- LCP: used to establish, monitor, and tear down PPP data links.
- NCP: used to negotiate the format and type of packets transmitted on data links.
- CHAP and PAP: used for network security authentication.

PPP Packet Format

Figure 6-3 shows the PPP packet format.

Figure 6-3 PPP packet format



The meanings of the fields are as follows:

- Flag field

  The Flag field identifies the start and end of a physical frame and is always 0x7E.

- Address field

  The Address field identifies a peer. Two communicating devices that are connected through PPP do not need to know the data link layer address of each other because PPP is used on point-to-point links. As such, this field must be filled with a broadcast address of all 1s and is of no significance to PPP.

- Control field

  The Control field value defaults to 0x03, indicating unnumbered information. By default, PPP does not use sequence numbers or acknowledgement mechanisms to ensure transmission reliability.

  The Address and Control fields together identify a PPP packet. That is, the PPP packet header value is FF03.

- Protocol field

  The Protocol field identifies the protocol of the data encapsulated in the Information field of a PPP packet.

  The structure of this field complies with the ISO 3309 extension mechanism for address fields. All Protocol field values must be odd, meaning that the least significant bit of the least significant byte must be 1.

  If a receiver receives a PPP data packet that does not comply with these rules from a sender, the receiver considers the packet unrecognizable and sends a Protocol-Reject packet padded with the protocol code of the rejected packet to the sender.

  Table 6-1 Common protocol codes

Protocol Code	Protocol Type
0021	Internet Protocol
002b	Novell IPX
002d	Van Jacobson Compressed TCP/IP
002f	Van Jacobson Uncompressed TCP/IP
8021	Internet Protocol Control Protocol
802b	Novell IPX Control Protocol
8031	Bridging NC
C021	Link Control Protocol
C023	Password Authentication Protocol
C223	Challenge Handshake Authentication Protocol

- Information field

  The Information field contains data. The maximum length for the Information field, including the padding, is the maximum receive unit (MRU). The MRU defaults to 1500 bytes and can be negotiated.

  Padding is required only when the length of the Information field does not meet the MRU requirements. To ensure proper communication, both

communicating parties must be able to identify and distinguish the padding bytes from real information.

- FCS field

  The frame check sequence (FCS) field checks the correctness of PPP packet transmission.

  Some mechanisms are used to ensure correct data packet transmission. However, they increase the cost and delay in data exchange at the application layer.

LCP Packet Format

Figure 6-3 shows the LCP packet format.

Communicating devices exchange LCP packets to negotiate with each other before establishing a PPP link. The LCP packets are encrypted in the Information field of a PPP data packet as the payload, and the Protocol field of the PPP data packet is fixed to 0xC021.

While a PPP link is being established, the Information field is variable and contains various packets, which need to be identified by corresponding fields.

- Code field

  The Code field is 1 byte in length and identifies the LCP packet type.

  If a receiver receives an LCP packet with an invalid Code field from a sender, the receiver sends a Code-Reject packet to the sender.

  Table 6-2 Common code values

Code Value	Packet Type
0x01	Configure-Request
0x02	Configure-Ack
0x03	Configure-Nak
0x04	Configure-Reject
0x05	Terminate-Request
0x06	Terminate-Ack
0x07	Code-Reject
0x08	Protocol-Reject
0x09	Echo-Request
0x0A	Echo-Reply
0x0B	Discard-Request
0x0C	Reserved

- Identifier field

The Identifier field is 1 byte long and is used to match and respond to requests. A packet with an invalid Identifier field will be discarded.

The sequence number of a Configure-Request packet usually begins with 0x01 and increments by 1 each time a Configure-Request packet is sent. After a receiver receives a Configure-Request packet, it sends a reply packet of which the sequence number must be the same as that of the received Configure-Request packet, regardless of which type of the reply it sends.

- Length field

  The Length field indicates the total number of bytes in the LCP packet, which equals the total length of the Code, Identifier, Length and Data fields.

  The Length field value cannot exceed the MRU of the link. Bytes outside the range of the Length field are treated as padding and are ignored after the packet is received.

- Data field

  The Data field contains the contents of a negotiation packet, including the following fields:

  – Type field: indicates the negotiation option type.
  – Length field: indicates the total length of the Data field. That is, the total length of Type, Length, and Data fields.
  – Data field: contains the contents of the negotiation option.

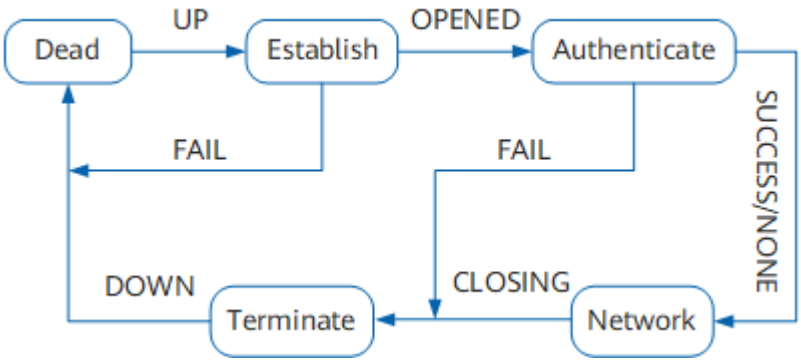
Table 6-3 Negotiation options in the Type field

Negotiation Option Value	Negotiation Packet Type
0x01	Maximum-Receive-Unit
0x02	Async-Control-Character-Map
0x03	Authentication-Protocol
0x04	Quality-Protocol
0x05	Magic-Number
0x06	RESERVED
0x07	Protocol-Field-Compression
0x08	Address-and-Control-Field-Compression

6.1.2.3 PPP Link Establishment Process

The following figure shows the PPP link establishment process.

Figure 6-4 PPP link establishment process



The PPP link establishment process is as follows:

1. Two communicating devices enter the Establish phase when starting to set up a PPP link.
2. In the Establish phase, the two devices perform an LCP negotiation to negotiate the working mode, maximum receive unit (MRU), authentication mode, and magic number. The working mode can be either Single-Link PPP (SP) or Multilink PPP (MP). If the LCP negotiation succeeds, LCP turns Opened, which indicates that a lower-layer link has been established.
3. If authentication is configured, the two devices enter the Authenticate phase and perform CHAP or PAP authentication. If no authentication is configured, the two devices enter the Network phase.
4. In the Authenticate phase, if authentication fails, the devices enter the Terminate phase. The link is removed and LCP turns Down. If authentication succeeds, the devices enter the Network phase and LCP remains Opened.
5. In the Network phase, the two devices perform an NCP negotiation to select and configure a network protocol and to negotiate network-layer parameters. After the two devices succeed in negotiating a network protocol, packets can be sent over this PPP link using the network protocol.

   Various control protocols, such as IP Control Protocol (IPCP) and Multiprotocol Label Switching Control Protocol (MPLSCP), can be used in NCP negotiation. IPCP mainly negotiates the IP addresses of the two devices.
6. After NCP negotiation succeeds, packets can be sent over the PPP link. During the PPP operation, the PPP connection can be terminated at any time. A physical link disconnection, authentication failure, timeout timer expiry, or connection close by administrators through configuration can cause the two devices to enter the Terminate phase.
7. In the Terminate phase, the two devices enter the Dead phase after all resources are released. The two devices remain in the Dead phase until they start to establish a new PPP connection.

The following describes the phases involved in PPP negotiation.

Dead Phase

The physical layer is unavailable during the Dead phase. A PPP link begins and ends with this phase.

When two communicating devices detect that the physical link between them is activated (for example, carrier signals are detected on the physical link), the two devices enter the Establish phase from the Dead phase.

After the link is terminated, the two devices return to the Dead phase.

Establish Phase

In the Establish phase, the two devices perform an LCP negotiation to negotiate parameters including the working mode (SP or MP), MRU, authentication mode, and magic number. After the LCP negotiation is complete, the two devices enter the next phase.

In the Establish phase, the LCP status changes as follows:

- When the link is unavailable, LCP is in the Initial or Starting state. When detecting that the link is available, the physical layer sends an up event to the link layer. After receiving the up event, the link layer changes the LCP status to Request-Sent. Then the devices at both ends send Configure-Request packets to configure a data link.
- If the local device first receives a Configure-Ack packet from the peer, the LCP status changes from Request-Sent to Ack-Received. After the local device sends a Configure-Ack packet to the peer, the LCP status changes from Ack-Received to Opened.
- If the local device first sends a Configure-Ack packet to the peer, the LCP status changes from Request-Sent to Ack-Sent. After the local device receives a Configure-Ack packet from the peer, the LCP status changes from Ack-Sent to Opened.
- After LCP enters the Opened state, the two devices enter the next phase.

The next phase is the Authenticate or Network phase, depending on whether authentication is required.

Authenticate Phase

The Authenticate phase is optional. By default, PPP does not perform authentication during PPP link establishment. If authentication is required, an authentication protocol must be specified in the Establish phase.

PPP authentication is mainly performed between hosts and devices that connect to a PPP network server through switched circuits or dial-up lines, and can also be used on dedicated links.

PPP provides two authentication modes: PAP authentication and CHAP authentication.

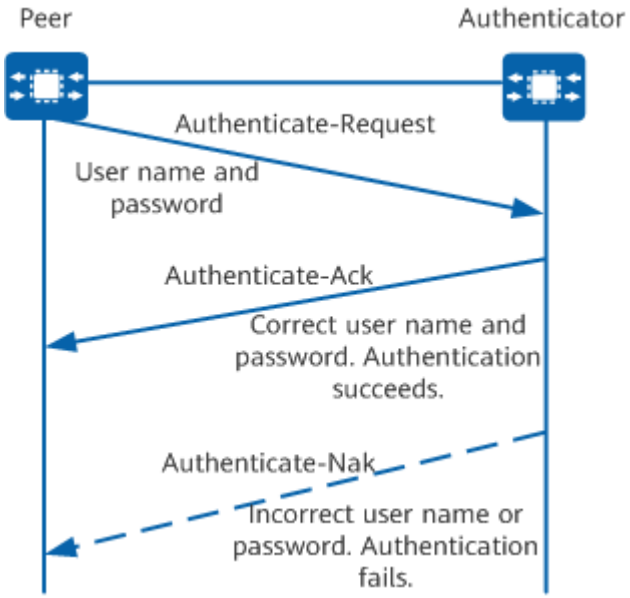
In unidirectional authentication, the device on one end functions as the authenticator, and the device on the other end functions as the peer. In bidirectional authentication, each device functions as both the authenticator and peer. In practice, unidirectional authentication is typically used.

PAP Authentication Process

PAP is a two-way handshake authentication protocol that transmits passwords in plain text.

Figure 6-5 shows the PAP authentication process.

Figure 6-5 PAP authentication process



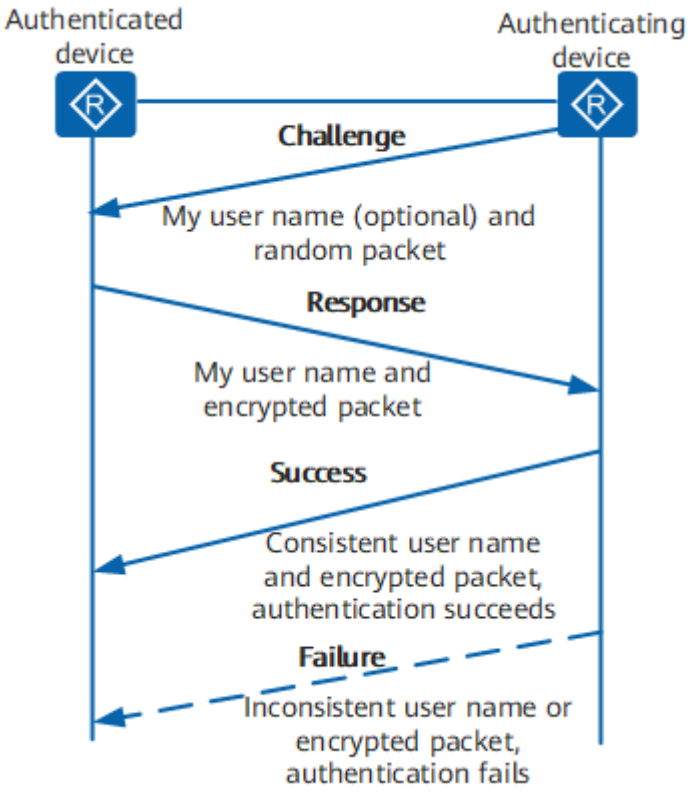
- The peer sends the local user name and password to the authenticator.
- The authenticator checks whether the received user name is in the local user table.
  - If the received user name is in the local user table, the authenticator checks whether the received password is correct. If the password is correct, the authentication succeeds. If the password is incorrect, the authentication fails.
  - If the received user name is not in the local user table, the authentication fails.

CHAP Authentication Process

CHAP is a three-way handshake authentication protocol. CHAP transmits only user names but not passwords, so it is more secure than PAP.

Figure 6-6 shows the CHAP authentication process.

Figure 6-6 CHAP authentication process



Unidirectional CHAP authentication is applicable to two scenarios:

- The authenticator is configured with a user name.
- The authenticator is not configured with a user name.

It is recommended that the authenticator be configured with a user name.

- When the authenticator is configured with a user name:
  - The authenticator initiates an authentication request by sending a Challenge packet, which contains a random number, the ID field, and the local user name, to the peer.
  - The peer checks whether the **ppp chap password** command is configured on the local interface after receiving the authentication request of the authenticator. If this command is configured, the peer uses the ID field and random number contained in the received Challenge packet as well as the user password configured in the command for hash or MD5 calculation. It then sends an authentication response that contains the generated hash value and its user name to the authenticator. If this command is not configured, the peer searches its local user table for the corresponding password based on the user name in the received Challenge packet, uses the ID field and random number contained in the packet and the searched password for hash or MD5 calculation. Thereafter, it sends an authentication response that contains the generated hash value and its user name to the authenticator.
- When the authenticator is not configured with a user name:

- The authenticator initiates an authentication request by sending a Challenge packet, which contains a random number and ID field, to the peer.
- The peer uses the ID field and random number contained in the received Challenge packet as well as the CHAP password configured in the **ppp chap password** command for hash or MD5 calculation, and then sends an authentication response that contains the generated hash value and its user name to the authenticator.

Comparison Between CHAP and PAP Authentication Processes

- In PAP authentication, passwords are sent over links in plain text. After a PPP link is established, the peer repeatedly sends the user name and password until authentication finishes. A high level of security is not ensured for this mode, so it is used on networks that do not require high security.
- CHAP is a three-way handshake authentication protocol. In CHAP authentication, the peer sends only a user name to the authenticator. Compared with PAP, CHAP features higher security because passwords are not transmitted. On networks requiring high security, CHAP authentication is used to establish a PPP connection.

Network Phase

In the Network phase, NCP negotiation is performed to select and configure a network protocol and to negotiate network-layer parameters. Each NCP may be in Opened or Closed state at any time. After an NCP enters the Opened state, network-layer data can be transmitted over the PPP link.

Terminate Phase

A PPP link can be terminated at any time. A link can be terminated manually by an administrator, or be terminated due to the loss of carrier, an authentication failure, or other causes.

6.1.3 Configuration Precautions for PPP

Hardware Requirements

Table 6-4 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

6.1.4 Default Settings for PPP

Table 6-5 Default settings for PPP

Parameter	Default Setting	Description
Negotiation timeout period	3 seconds	In PPP negotiation, if the local end does not receive any response from the peer within the specified timeout period, it retransmits the same packet.
Negotiation polling interval	10 seconds	The polling interval of an interface is the interval at which the interface sends a Keepalive packet. Keepalive packets are used to monitor and maintain the link status. If an interface does not receive any Keepalive packet through the PPP link after five Keepalive intervals, it considers that the link fails.
Number of heartbeat packet retransmissions	4	A device sends PPP heartbeat packets to monitor the PPP link quality. If the link quality deteriorates and a heartbeat packet is retransmitted for the specified number of times, the device terminates the PPP link.

6.1.5 Configuring Basic PPP Functions

Prerequisites

Before configuring PPP authentication, you have completed the following task:

- Configure physical attributes for synchronous serial interfaces on the router to ensure that the interfaces are physically up.

6.1.5.1 Configuring PPP as the Link Layer Protocol of an Interface

Context

PPP is a point-to-point link layer protocol. Before using PPP, you need to set the link type of a specified interface to PPP.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```


Step 3 Set the link type to PPP.

```
link-protocol ppp
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

After the dial-up connection is set up, run the **display ifm/interfaces/interface[name="*interface-name*"]/ppp-net/ppp/ppp-link-info** command to view PPP session information.

6.2 PPPoE Configuration

Point-to-Point Protocol over Ethernet (PPPoE) is a PPP running on the Ethernet and widely used on campus networks.

6.2.1 Overview of PPPoE

Definition

PPP over Ethernet (PPPoE) is a link layer protocol that encapsulates Point-to-Point Protocol (PPP) frames into Ethernet frames. It enables multiple hosts on an Ethernet network to connect to a broadband remote access server (BRAS).

Purpose

Using PPPoE, carriers can connect multiple hosts at a site to a remote access device that provides access control and accounting for these hosts in a manner similar to dial-up access. PPPoE enables carriers to achieve this at a lower cost because Ethernet is the most cost-effective among all access technologies and PPP implements access control and accounting.

PPPoE allows a large number of hosts on an Ethernet network to connect to the Internet through a remote access device and controls each host using PPP. PPPoE can be used in various scenarios and provides high security as well as convenient accounting.

6.2.2 Understanding PPPoE

6.2.2.1 PPPoE Packet Type

Based on the value of the Code field, PPPoE packets are classified into the following types:

- The value 0x09 indicates PPPoE Active Discovery Initiation (PADI) packets.
- The value 0x07 indicates PPPoE Active Discovery Offer (PADO) packets.
- The value 0x19 indicates PPPoE Active Discovery Request (PADR) packets.

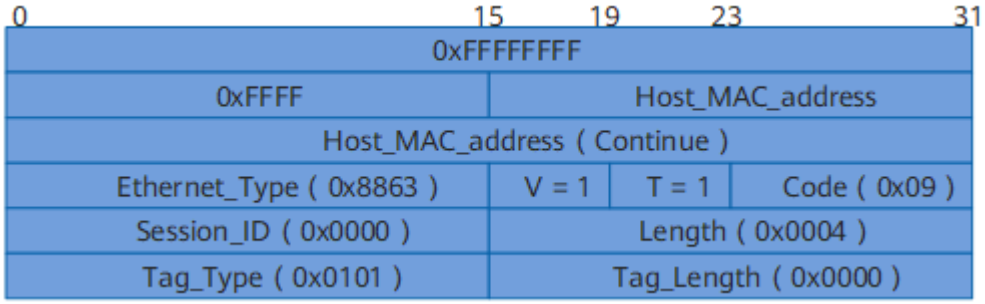
- The value 0x65 indicates PPPoE Active Discovery Session-confirmation (PADS) packets.
- The value 0x00 indicates session data.
- The value 0xa7 indicates PPPoE Active Discovery Terminate (PADT) packets.

Discovery Stage

PADI

- The Code field is set to 0x09.
- The Session_ID field is set to 0x0000.
- The Tag_Type field is set to 0x0101 (Service-Name). It is followed by a service name. A PADI packet contains only one tag with Tag_Type being Service-Name, and other tags are optional.

Figure 6-7 Example of a PADI packet

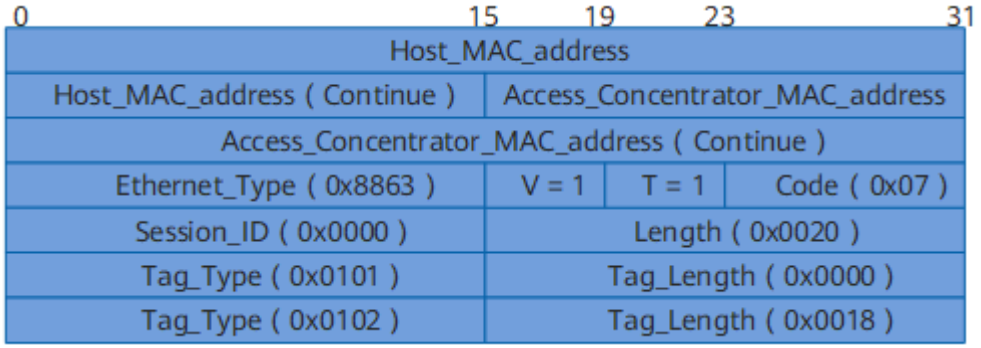


PADO

- The Code field is set to 0x07.
- The Session_ID field is set to 0x0000.
- Different values of the Tag_Type field represent different tag types.
  - When the Tag_Type field is set to 0x0101 (Service-Name), it is followed by a service name.
  - When the Tag_Type field is set to 0x0102 (AC-Name), it is followed by a string that uniquely identifies an access controller (AC).

  A PADO packet contains only one tag with Tag_Type being AC-Name and at least one tag with Tag_Type being Service-Name.

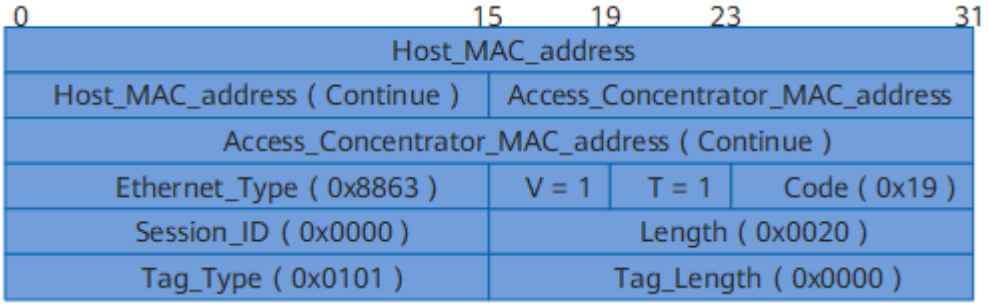
Figure 6-8 Example of a PADO packet



PADR

- The Code field is set to 0x19.
- The Session_ID field is set to 0x0000.
- The Tag_Type field is set to 0x0101 (Service-Name). It is followed by a service name. A PADR packet contains only one tag with Tag_Type being Service-Name, and other tags are optional.

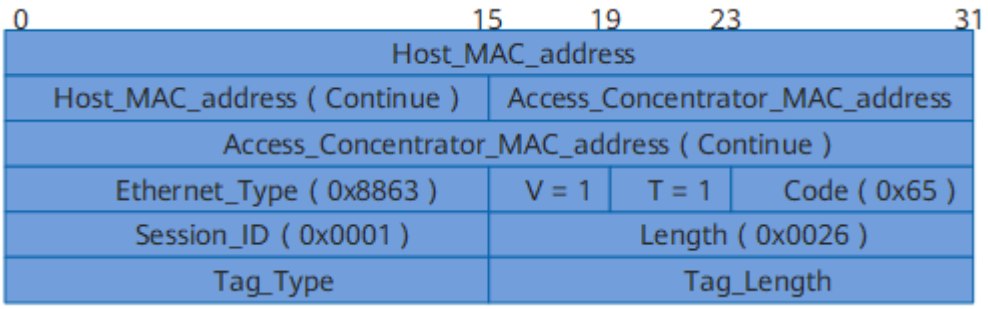
Figure 6-9 Example of a PADR packet



PADS

- The Code field is set to 0x65.
- The value of the Session_ID field is that specified at the Discovery stage.
- Tags are optional.

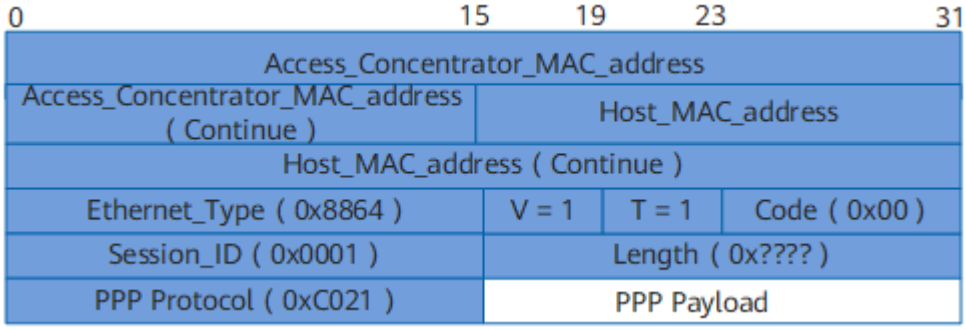
Figure 6-10 Example of a PADS packet



Session Stage

- The Ethernet_Type field is set to 0x8864.
- The Code field is set to 0x00.
- The value of the Session_ID field must be that specified at the Discovery stage.
- The Tag_Type field contains a PPP frame with the first field being PPP Protocol-ID.

Figure 6-11 Example of a packet at the Session stage

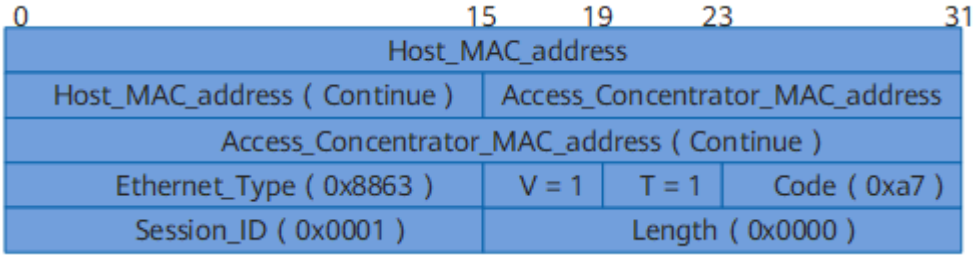


Terminate Stage

PADT

- The Code field is set to 0xa7.
- The value of the Session_ID field is that specified at the Discovery stage.
- No tag is required in a PADT packet.

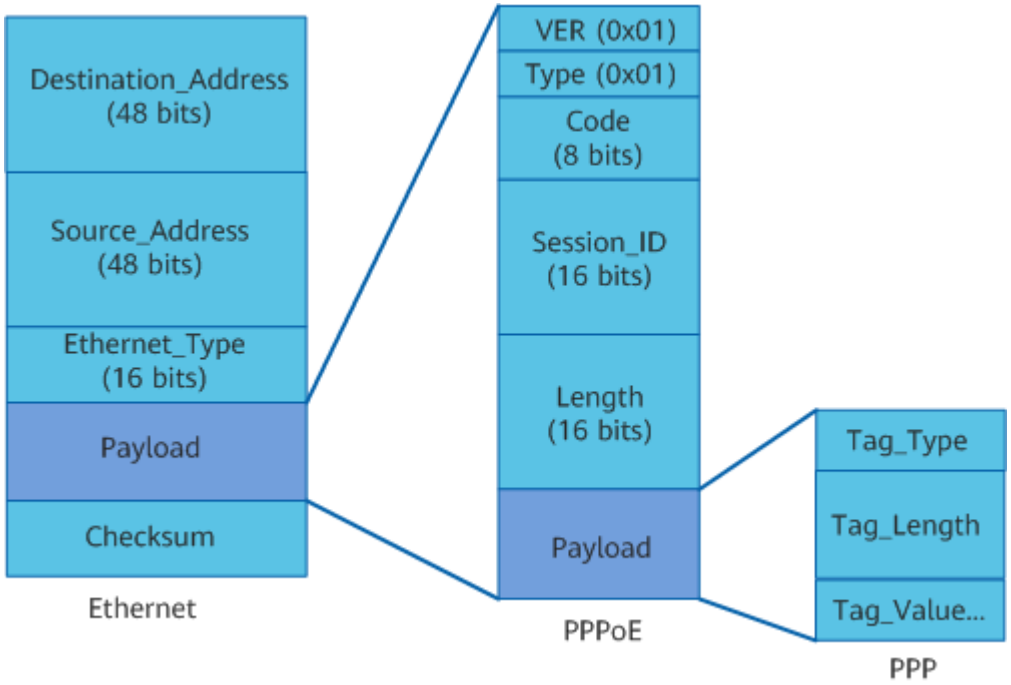
Figure 6-12 Example of a PADT packet



6.2.2.2 PPPoE Packet Format

A PPPoE packet is a PPP packet encapsulated in an Ethernet frame. **Figure 6-13** shows the format of a PPPoE packet.

Figure 6-13 PPPoE packet format



The following table describes each field in a PPPoE packet.

Table 6-6 Fields in a PPPoE packet

Field	Length	Description
Destination_Address	48 bits	An Ethernet unicast destination address or Ethernet broadcast address 0xFFFFFFFF. • For Discovery packets, the value is a unicast or broadcast address. When a PPPoE client sends the packet to search for the PPPoE server, the value is a broadcast address. When the PPPoE client has discovered the PPPoE server, the value is a unicast address. • At the Session stage, the value must be the unicast address determined by the PPPoE server and client at the Discovery stage.
Source_Address	48 bits	Ethernet MAC address of the source device.
Ethernet_Type	16 bits	Stage of PPPoE dial-up. • The value 0x8863 indicates the Discovery or Terminate stage. • The value 0x8864 indicates the Session stage.
VER	4 bits	PPPoE version number. This field has a fixed value of 0x01.
Type	4 bits	PPPoE type. This field has a fixed value of 0x01.
Code	8 bits	PPPoE packet type. • 0x00: indicates session data. • 0x09: indicates PADI packets. • 0x07: indicates PADO packets. • 0x19: indicates PADR packets. • 0x65: indicates PADS packets. • 0xa7: indicates PADT packets. For details about PPPoE packets, see **6.2.2.1 PPPoE Packet Type**.
Session_ID	16 bits	An unsigned number in network byte order. The value is fixed for a given PPPoE session and defines a PPPoE session along with Ethernet Source_address and Destination_address. The value 0xFFFF is reserved.
Length	16 bits	Length of the PPPoE payload, excluding the length of the Ethernet header and PPPoE header.
Tag_Type	16 bits	Network byte order.

Field	Length	Description
Tag_Length	16 bits	Number of bytes in the Tag_Value field. The value is an unsigned number in network byte order.
Checksum	32 bits	Checksum. It is used to check the packet validity.

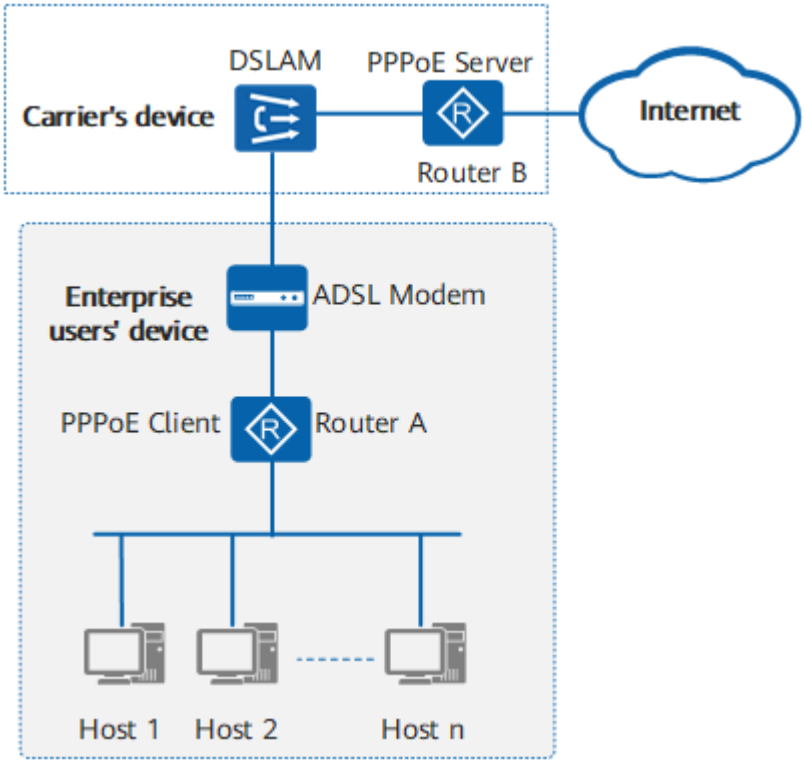
6.2.2.3 PPPoE Typical Networking

PPPoE uses the client/server architecture. A PPPoE client sends a connection request to the PPPoE server, and the PPPoE server provides access control and authentication functions for the PPPoE client.

Device Functioning as a PPPoE Client

As shown in **Figure 6-14**, Device A functions as a PPPoE client and connects to LAN users. Device B is a carrier's device. It is not necessary for all hosts on the LAN to have the PPPoE client dial-up software installed; instead, they share the same account and establish PPPoE sessions with Device B through Device A.

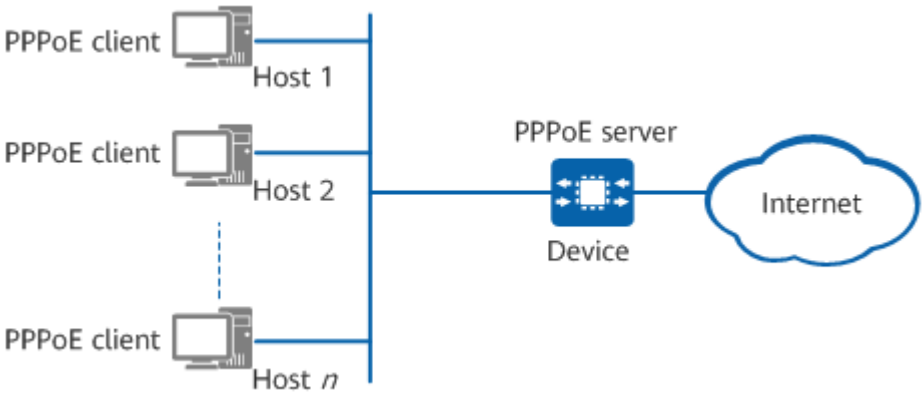
Figure 6-14 Device functioning as a PPPoE client



A device functions as a PPPoE server to dynamically allocate IP addresses to clients and provide multiple authentication methods. The PPPoE server is applicable to Ethernet networks connecting to the Internet, such as school campus networks and residential networks.

As shown in **Figure 6-15**, all hosts have the PPPoE client dial-up software installed. Each host functions as a PPPoE client and establishes a PPPoE session with the PPPoE server (**Device** in the figure). Each host uses a unique account, which facilitates user accounting and control by the carrier.

Figure 6-15 Device functioning as a PPPoE server

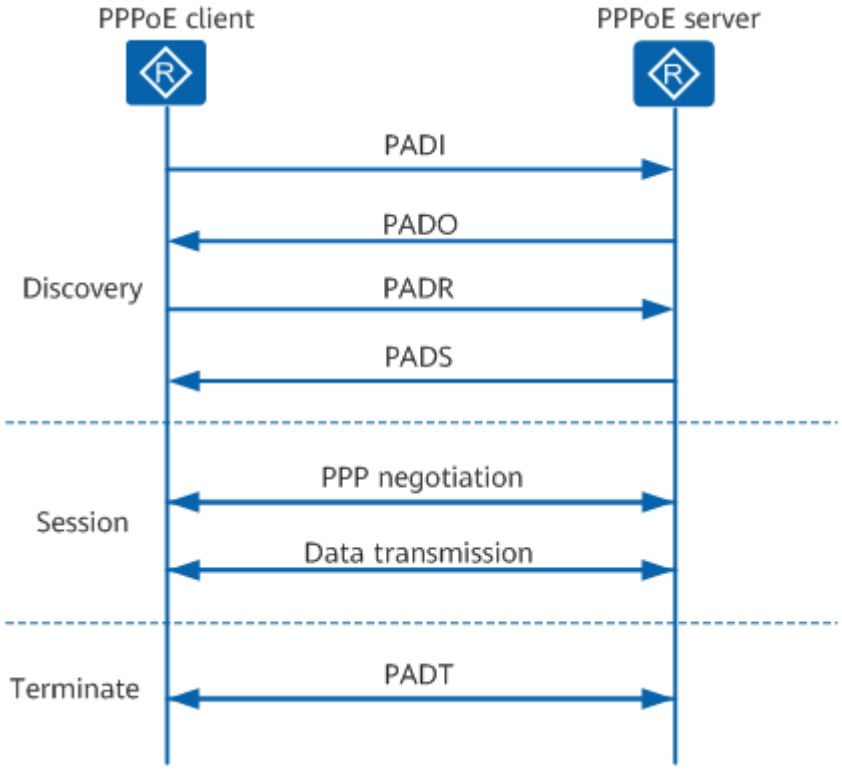


The PPPoE client at the peer end of a PPPoE server can be a host that has the dial-up software installed or a router.

6.2.2.4 PPPoE Dial-up Implementation

PPPoE dial-up is used to establish a PPPoE session between a PPPoE client and a PPPoE server. **Figure 6-16** shows the PPPoE dial-up process.

Figure 6-16 PPPoE dial-up process



The PPPoE dial-up process includes three stages: Discovery, Session, and Terminate.

Discovery Stage

The Discovery stage consists of the following steps:

1. A PPPoE client broadcasts a PPPoE Active Discovery Initiation (**PADI**) packet that contains the service type required by the PPPoE client.
2. After receiving the PADI packet, all PPPoE servers compare the requested service with the services they can provide. The PPPoE servers that can provide the requested service unicast PPPoE Active Discovery Offer (**PADO**) packets to the PPPoE client.
3. The PPPoE client receives PADO packets from one or more PPPoE servers. If multiple PPPoE servers reply, the PPPoE client selects the PPPoE server from which the PADO packet is received first and unicasts a PPPoE Active Discovery Request (**PADR**) packet to the selected PPPoE server.
4. The PPPoE server generates a unique session ID to identify the PPPoE session with the PPPoE client, and then sends a PPPoE Active Discovery Session-confirmation (**PADS**) packet containing this session ID to the PPPoE client. When the PPPoE session is established, the PPPoE server and PPPoE client enter the PPPoE Session stage.

After the PPPoE session is established, the PPPoE server and client learn the session ID and the peer Ethernet address. Therefore, the PPPoE server has a unique PPPoE session with the client.

Session Stage

The PPPoE Session stage involves PPP negotiation and PPP data transmission.

PPP negotiation at the PPPoE Session stage is the same as common PPP negotiation, which includes the Link Control Protocol (LCP), authentication, and Network Control Protocol (NCP) phases.

1. In the LCP phase, the PPPoE server and PPPoE client establish and configure a data link, and verify the data link status.
2. When LCP negotiation is complete, authentication starts. The authentication protocol depends on the LCP negotiation result. The authentication protocol can be Challenge Handshake Authentication Protocol (CHAP) or Password Authentication Protocol (PAP).
3. After authentication succeeds, PPP enters the NCP phase. NCP is a protocol suite used to configure network-layer protocols. A commonly used network-layer protocol is IP Control Protocol (IPCP), which is used to negotiate IP addresses for users and the domain name server (DNS).

When PPP negotiation succeeds, PPP data packets can be forwarded over the established PPP link.

At the PPPoE Session stage, all Ethernet data packets are sent in unicast mode.

Terminate Stage

The PPPoE server and client use PPP to terminate the PPPoE session. If PPP cannot be used, the server and client can use PPPoE Active Discovery Terminate (**PADT**) packets to terminate the PPPoE session.

After a PPPoE session is established, the PPPoE client or the PPPoE server can unicast a PADT packet to terminate the PPPoE session at any time. After transmitting or receiving the PADT packet, the PPPoE server and PPPoE client are not allowed to use this session to send any PPP traffic.

6.2.3 Configuration Precautions for PPPoE

Licensing Requirements

PPPoE is not under license control.

Hardware Requirements

Table 6-7 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

6.2.4 Default Settings for PPPoE

Table 6-8 Default settings for PPPoE

Parameter	Default Setting
PPPoE client function on an interface	Disabled
Accepting the DNS server address specified on a PPPoE server	Enabled
Accepting the IP address specified on a PPPoE server	Enabled

6.2.5 Configuring the Device as a PPPoE Client

A PPPoE client allows all hosts on a LAN to use the same account to dial up to the Internet.

Prerequisites

Before configuring the device as a PPPoE client, you have completed the following task:

- Power on the device.

6.2.5.1 Configuring the PPPoE Client as the Authentication Peer

Prerequisites

You have set the link type of the corresponding interface to PPP.

Context

After the PPP authentication mode is set to PAP or CHAP on the PPPoE server, you need to configure the user name and password corresponding to the authentication mode on the PPPoE client.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Enter the ppp-base container node view.

```
ppp-net ppp ppp-base
```

Step 4 Configure the user name and password.

- Configure the user name and password for CHAP authentication.

```
chap-user-name username
chap-password
Enter password: password
Confirm password: password
```

- Configure the user name and password for PAP authentication.

```
pap-user-name username
pap-password
Enter password: password
Confirm password: password
```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display ifm/interfaces/interface[name="*interface-name*"]/ppp-net/ppp/ppp-base** command to check the user name and password corresponding to the authentication mode configured on the PPPoE client.

6.2.5.2 Enabling the PPPoE Client Function on an Interface

Context

The PPPoE client function can take effect only after it is configured on a Layer 3 Ethernet interface.

PPPoE auto dial-up (permanently online mode):

- When the physical link is up, the device immediately initiates a PPPoE call and establishes a PPPoE session. The PPPoE session exists permanently unless it is deleted.

  Auto dial-up applies to users who are not charged based on traffic or duration, for example, users who subscribe to services on a yearly basis.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
remove dhcp-client-if/
link-protocol ppp
```

Step 3 Enter the ppp-base container node view.

```
ppp-net ppp ppp-base
```

Step 4 Configure an authentication mode.

- Configure CHAP authentication.

  ```
  chap-authen-flag enable
  ```

- Configure PAP authentication.

  ```
  pap-authen-flag enable
  ```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

- Run the **display ifm/interfaces/interface[name="*interface-name*"]/ppp-net/ppp/ppp-base** command to check the authentication mode configured on the PPPoE client.
- After the dial-up succeeds, run the **display ifm/interfaces/interface[name="*interface-name*"]/pppoe-client-session-summarys** command to check PPPoE session information.
- Run the **display ifm/interfaces/interface[name="interface-name"]/pppoe-client-session-summarys** command to check PPPoE session information on a specified interface (enabled with the PPPoE client function).

6.2.5.3 Example for Configuring the Device as a PPPoE Client

Networking Requirements

In **Figure 6-17**, the device functioning as a PPPoE client connects to hosts on the LAN using GE0/0/1 and connects to a PPPoE server using GE0/0/9.

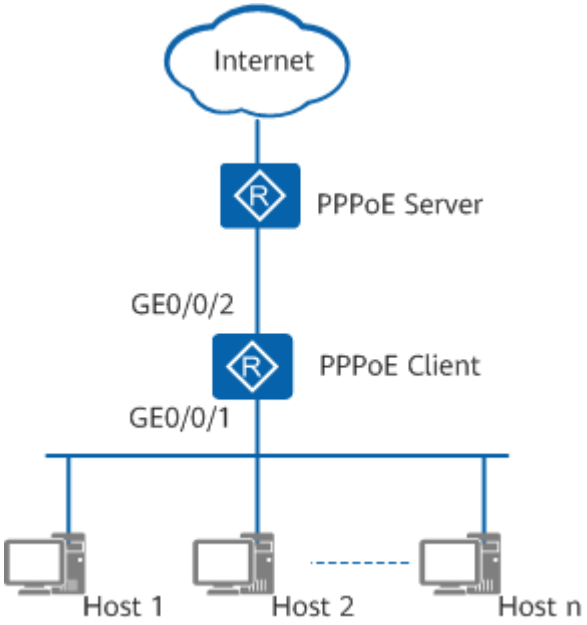
The hosts need to share an account. During connection establishment, if the account is authenticated successfully on the PPPoE server, a PPPoE session is established. Service requirements are as follows:

- The device establishes a PPPoE session with the PPPoE server using PPP authentication.
- After the connection is disconnected, the device attempts to re-establish a dial-up connection at intervals.

Figure 6-17 Network diagram of the device functioning as a PPPoE client

NOTE

In this example, interface1 and interface2 represent GE0/0/1 and GE0/0/9, respectively.



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure CHAP authentication on the dialer interface so that the device can establish a connection with the PPPoE server through PPP authentication.
2. Set the dial-up mode to auto dial-up so that the device will attempt to re-establish a dial-up connection at intervals after a disconnection.

Procedure

Step 1 Configure the PPPoE client.

```
MDCLI> edit-config
MDCLI> ifm interfaces interface name GE0/0/9
[(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> remove dhcp-client-if/
[(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> link-protocol ppp
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> ppp-net ppp ppp-base
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> chap-authen-flag enable
```



```
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> chap-user-name usersoho@system
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI>chap-password
Enter password:
Confirm password:
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> commit
```

Step 2 Verify the configuration.

```
MDCLI> display ifm/interfaces/interface[name="GE0/0/9"]/pppoe-client-session-summarys
{
  "pppoe-client-session-summary": [
    {
      "session-id": 117,
      "if-name": "GE0/0/9",
      "client-mac": "0a0e-2e96-b4af",
      "server-mac": "22e1-fc05-0204",
      "state": "UP"
    }
  ]
}
```

----End

6.2.6 Maintaining PPPoE

6.2.6.1 Resetting PPPoE Sessions

Context

To disconnect users or trigger re-negotiation, reset PPPoE sessions on either the PPPoE server or client.

NOTICE

Resetting PPPoE sessions interrupts user services. Exercise caution when performing this operation.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Reset sessions.

```
reset-pppoe-client-session
```

Step 3 Specify an interface on which sessions are reset.

- Reset sessions on all interfaces.
  ```
  all true
  ```
- Reset sessions on a specified interface.
  ```
  interface interface-name
  ```

After running either of these commands, PPPoE sessions are terminated, and the device automatically re-establishes PPPoE sessions 16 seconds later.

Step 4 Commit the configuration.

```
emit
```

----End

6.2.7 Configuration Examples for PPPoE

This section provides several PPPoE configuration examples, including network requirements, configuration roadmap, configuration procedure, and configuration files.

6.2.7.1 Example for Configuring Dual-Uplink PPPoE Dial-up Access for Load Balancing

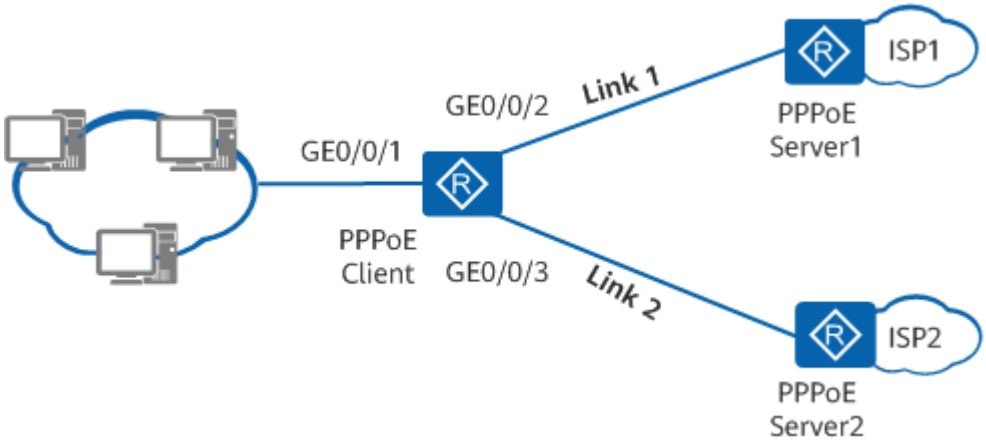
Networking Requirements

In **Figure 6-18**, the device functions as an enterprise's egress gateway and connects to the Internet through PPPoE dial-up over two uplinks. The device uses equal-cost egress routes for load balancing on link 1 and link 2. NAT is configured on the device so that users on the enterprise's private network can access the Internet.

Figure 6-18 Network diagram of the PPPoE client using two uplinks for load balancing

NOTE

In this example, interface1, interface2, and interface3 represent GE0/0/1, GE0/0/9, and GE0/0/10, respectively.



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure CHAP authentication on GE0/0/9 and GE0/0/10 so that the device can establish PPPoE sessions with PPPoE Server1 and PPPoE Server2 after passing PPP authentication.
2. The device automatically configures equal-cost routes for load balancing.

Procedure

Step 1 Enable the PPPoE client function on GE0/0/9.

```
MDCLI> ifm interfaces interface name GE0/0/9
[root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> edit-config
[(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> link-protocol ppp
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> ppp-net ppp ppp-base
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> chap-authen-flag enable
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> chap-user-name usersoho@system
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> chap-password
Enter password:
Confirm password:
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/ppp-net/ppp/ppp-base
MDCLI> commit
```

Step 2 Enable the PPPoE client function on GE0/0/10.

```
MDCLI> ifm interfaces interface name GE0/0/10
[root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]
MDCLI> edit-config
[(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]
MDCLI> link-protocol ppp
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]
MDCLI> ppp-net ppp ppp-base
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]/ppp-net/ppp/ppp-base
MDCLI> chap-authen-flag enable
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]/ppp-net/ppp/ppp-base
MDCLI> chap-user-name usersoho@system
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]/ppp-net/ppp/ppp-base
MDCLI> chap-password
Enter password:
Confirm password:
[*(gl)root@HUAWEI]/ifm/interfaces/interface[name="GE0/0/10"]/ppp-net/ppp/ppp-base
MDCLI> commit
```

Step 3 Verify the configuration.

```
MDCLI> display ifm/interfaces/interface[name="GE0/0/9"]/pppoe-client-session-summarys
{
  "pppoe-client-session-summary": [
    {
      "session-id": 117,
      "if-name": "GE0/0/9",
      "client-mac": "0a0e-2e96-b4af",
      "server-mac": "22e1-fc05-0204",
      "state": "UP"
    }
  ]
}
MDCLI> display ifm/interfaces/interface[name="GE0/0/10"]/pppoe-client-session-summarys
{
  "pppoe-client-session-summary": [
    {
      "session-id": 118,
      "if-name": "GE0/0/10",
      "client-mac": "0a0e-2e96-b4bf",
      "server-mac": "22e1-fc05-0214",
      "state": "UP"
    }
  ]
}
```

----End

6.2.8 PPPoE FAQ

6.2.8.1 What Should I Do If the PPP Authentication Mode of a PPPoE Server Is Unknown?

PPP authentication will fail if inconsistent PPP authentication modes are used on the PPPoE server and client. If the PPPoE authentication mode provided by the carrier is unknown, you are advised to configure both CHAP and PAP authentication modes on the device that functions as a PPPoE client. The device will dynamically determine the authentication mode based on packets exchanged with the PPPoE server.

7 IP Addresses and Services Configuration

7.1 IPv4 Basic Configuration

7.1.1 Overview of IPv4 Basic

Definition

Internet Protocol version 4 (IPv4) is the core protocol of the Transmission Control Protocol (TCP)/IP protocol suite. All TCP, User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) data is carried in IP datagrams. IPv4 works at the Internet layer of the TCP/IP model. This layer corresponds to the network layer in the OSI model. The network layer provides connectionless data transmission. Each IP datagram is transmitted independently without needing to establish a connection before IP datagrams are sent. A large number of datagrams need to be forwarded on the network, which may cause network congestion and degrade network performance. You can adjust parameters or forwarding modes for datagrams to improve network performance.

Purpose

IP provides unreliable and connectionless data transmission services. Unreliable transmission means that IP does not ensure that IP datagrams successfully arrive at their destination. IP only provides best effort delivery. Once an error occurs, for example, a device exhausts the buffer, IP discards the excess datagrams and sends

ICMP messages to the source. The upper-layer protocols, such as TCP, are responsible for resolving reliability issues.

Connectionless transmission means that IP does not maintain status information for subsequent datagrams. Every datagram is processed independently, meaning that IP datagrams may not be received in the same order they are sent. If a source sends two consecutive datagrams A and B in sequence to the same destination, each datagram is possibly routed over a different path, and therefore B may arrive ahead of A.

Each host on an IP network must have an IP address. An IPv4 address is 32 bits long and consists of two parts: network ID and host ID.

- A network ID uniquely identifies a network segment or a group of network segments. A network ID can be obtained by converting an IP address and subnet mask into binary numbers and performing an AND operation on the numbers.
- A host ID uniquely identifies a device on a network segment. A host ID can be obtained by converting an IP address and subnet mask into binary numbers, reversing the post-conversion subnet mask, and performing an AND operation on the numbers.

Network devices with the same network ID are located on the same network, regardless of their physical locations.

Benefits

IPv4 is used at the network layer to ensure proper data transmission between the data link and transport layers. IPv4 shields the differences at the data link layer and provides a uniform format for datagrams transmitted at the transport layer.

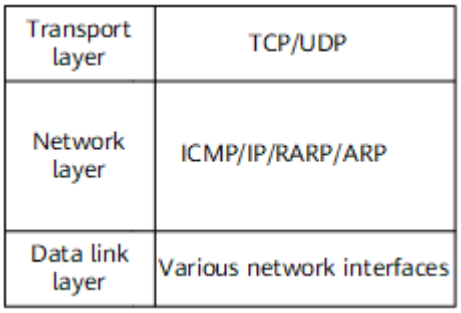
7.1.2 Understanding IPv4 Basic

7.1.2.1 IPv4 Protocol Suite

Definition

IPv4 is the core protocol in the TCP/IP protocol suite. The IPv4 protocol suite includes Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), ICMP, TCP, and UDP.

Figure 7-1 IPv4 protocol suite



As shown in **Figure 7-1**, ARP and RARP work between the data link and network layers for address resolution. ICMP works between the network and transport layers to ensure the correct forwarding of IP datagrams.

ARP

ARP maps an IP address to a MAC address, and can be implemented in dynamic or static mode. ARP provides some extended functions, such as proxy ARP, gratuitous ARP, ARP security, and ARP-Ping.

RARP

RARP maps a MAC address to an IP address.

ICMP

ICMP works at the network layer to ensure the correct forwarding of IP datagrams, and allows hosts or devices to report errors during datagram transmission. An ICMP message is encapsulated in an IP datagram as the data, and forms a complete IP datagram together with an IP header.

7.1.2.2 IPv4 Address

To connect a PC to the Internet, you must apply for an IP address from the Internet service provider (ISP).

An IP address is a numerical label assigned to each device on a computer network. An IPv4 address is a 32-bit binary number, expressed in dotted decimal notation, which helps you memorize and identify it. In dotted decimal notation, an IPv4 address is written as four decimal numbers, one for each byte of the address. For example, the binary IPv4 address 00001010 00000001 00000001 00000010 is written as 10.1.1.2 in dotted decimal notation.

An IPv4 address consists of two parts:

- Network ID (Net-id): identifies a network.
- Host ID (Host-id): identifies a host on a network. Network devices with the same network ID are located on the same network, regardless of their physical locations.

Characteristics of IPv4 Addresses

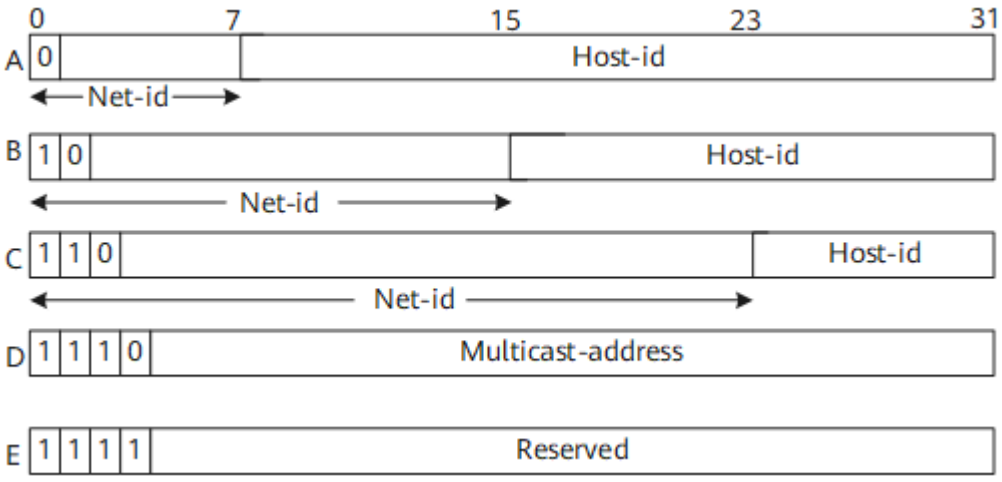
IPv4 addresses have the following characteristics:

- IPv4 addresses do not show any geographical information. The network ID represents the network to which a host belongs.
- When a host connects to two networks, it must have two IPv4 addresses with different network IDs. In this case, the host is called a multihomed host.
- Networks allocated with network IDs are in the same class.

IPv4 Address Classification

As shown in **Figure 7-2**, IPv4 addresses are classified into five classes to facilitate IPv4 address management and networking.

Figure 7-2 Five classes of IPv4 addresses



Most IP addresses in use belong to Class A, Class B, or Class C. Class D addresses are multicast addresses, and Class E addresses are reserved. The easiest way to determine the class of an IP address is to check the first bits in its network ID. The class fields of Class A, Class B, Class C, Class D, and Class E are binary numbers 0, 10, 110, 1110, and 1111, respectively.

Table 7-1 IPv4 address classes and ranges

Class	Range	Description
A	0.0.0.0 to 127.255.255.255	IP addresses with a host ID comprising all 0s are network addresses and are used for routing. IP addresses with a host ID comprising all 1s are broadcast addresses and are used for broadcasting datagrams to all hosts on a network.
B	128.0.0.0 to 191.255.255.255	IP addresses with a host ID comprising all 0s are network addresses and are used for routing. IP addresses with a host ID comprising all 1s are broadcast addresses and are used for broadcasting datagrams to all hosts on a network.
C	192.0.0.0 to 223.255.255.255	IP addresses with a host ID comprising all 0s are network addresses and are used for routing. IP addresses with a host ID comprising all 1s are broadcast addresses and are used for broadcasting datagrams to all hosts on a network.
D	224.0.0.0 to 239.255.255.255	Class D addresses are multicast addresses.
E	240.0.0.0 to 255.255.255.255	Reserved. 255.255.255.255 is a LAN broadcast address.

Special IPv4 Addresses

Table 7-2 Special IPv4 addresses

Network ID	Host ID	Used as a Source Address	Used as a Destination Address	Description
All 0s	All 0s	Supported	Not supported	Used for routing on a network
All 0s	Host ID	Supported	Not supported	Used by a specific host on a network
127	Any value that does not comprise all 0s or all 1s	Supported	Supported	Used as a loopback address
All 1s	All 1s	Not supported	Supported	Used as a limited broadcast address (datagrams with this IP address are never forwarded)
Net-id	All 1s	Not supported	Supported	Used as a directed broadcast address (datagrams with this IP address is broadcast on a specified network)

NOTE

Net-id is neither all 0s nor all 1s.

Private IPv4 Addresses

Private IPv4 addresses were proposed to resolve IPv4 address shortage. They are used for internal networks or hosts, and cannot be used for public networks. RFC standards describe three IPv4 address segments, which are reserved for private networks.

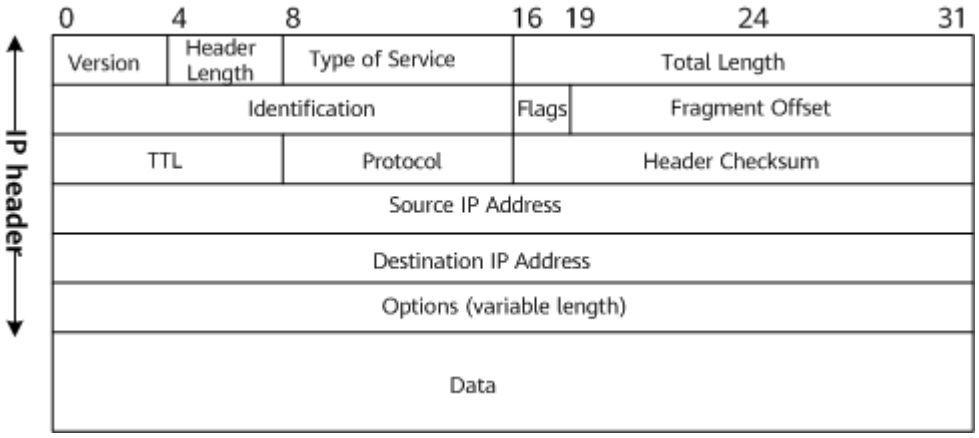
Table 7-3 Private IPv4 addresses

Class	Range
A	10.0.0.0 to 10.255.255.255
B	172.16.0.0 to 172.31.255.255
C	192.168.0.0 to 192.168.255.255

7.1.2.3 IPv4 Datagram Format

Figure 7-3 shows the IPv4 datagram format.

Figure 7-3 IPv4 datagram format



An IPv4 datagram consists of a header and a data field. The first 20 bytes in the header are mandatory for all IPv4 datagrams. The Options field following the 20 bytes has a variable length.

Table 7-4 describes the meaning of each field in an IPv4 datagram.

Table 7-4 Meaning of each field in an IPv4 datagram

Field	Length	Meaning
Version	4 bits	IP version, which can be IPv4 or IPv6.
Header Length	4 bits	Length of the IPv4 header.
Type of Service	8 bits	This field takes effect only in the differentiated service model.
Total Length	16 bits	Total length of the header and data.
Identification	16 bits	A device maintains a counter on a storage device to record the number of IPv4 datagrams. The counter value increases by 1 each time an IPv4 datagram is sent.
Flags	3 bits	This field has three bits. The most significant bit is reserved and must be set to 0. If the middle bit is 0, the datagram can be fragmented. If the middle bit is 1, the datagram cannot be fragmented. If the least significant bit is 0, the fragment is the last one. If the least significant bit is 1, there are more fragments.

Field	Length	Meaning
Fragment Offset	13 bits	Location of a fragment in a datagram.
TTL	8 bits	Life span of a datagram on a network. TTL is measured by the number of hops.
Protocol	8 bits	Type of the protocol carried in a datagram.
Header Checksum	16 bits	A device calculates the header checksum for each datagram received. If the checksum is 0, the device knows that the header remains unchanged and retains the datagram. This field checks only the header but not the data.
Source IP Address	32 bits	IPv4 address of a sender.
Destination IP Address	32 bits	IPv4 address of a receiver.
Options	0 to 40 bytes (variable length)	This field supports various options, such as error correction, measurement, and security. Pad bytes with a value of 0 are added if necessary.
Data	Variable	Pads an IP datagram.

7.1.2.4 Subnetting

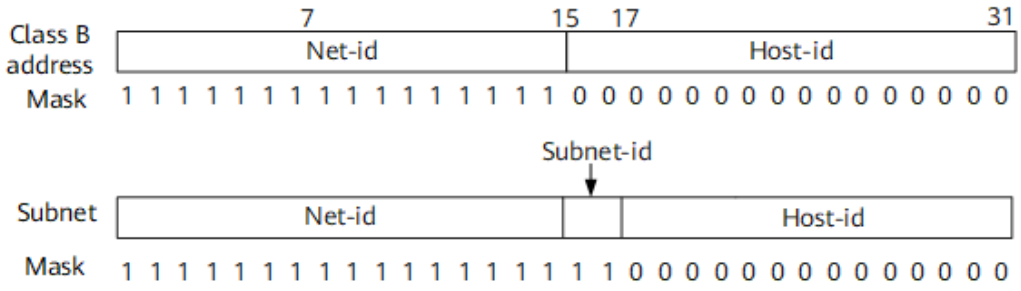
A network can be divided into multiple subnets to conserve IPv4 address space and support flexible IPv4 addressing.

When many hosts are distributed on an internal network, the internal host IDs can be divided into multiple subnet IDs to facilitate management. In this case, the entire network contains multiple small networks.

The network ID of subnets is visible to the external network, but the subnet IDs are not. The subnet IDs take effect in route selection and destination host addressing only after datagrams enter the internal network.

Figure 7-4 shows subnetting of a Class B IPv4 address. The subnet mask consists of a string of continuous 1s and 0s. All 1s correspond to the Net-id and Subnet-id fields, and all 0s correspond to the Host-id field.

Figure 7-4 IPv4 address subnetting



The Class B IP address shown in **Figure 7-4** is used as an example. Assume that the network address is 172.16.0.0 and the mask is 255.255.0.0. The two most significant bits of the host ID are used for subnetting. The subnet ID ranges from 00 to 11 (binary), allowing a maximum of 4 (2^2) subnets. Each subnet ID has a subnet mask, which changes after subnetting. Specifically, the subnet mask of the Class B IP address is changed from 255.255.0.0 to 255.255.192.0. After performing an AND operation on the IPv4 address and the subnet mask, you can obtain the network address. **Figure 7-5** shows the network addresses of the four subnets.

Figure 7-5 IPv4 address subnets

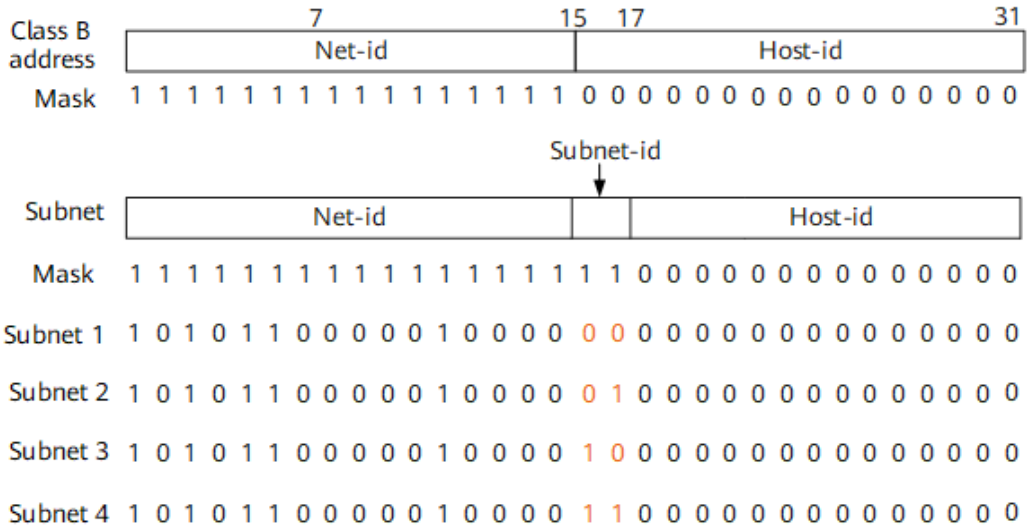


Table 7-5 lists the network addresses (with the all-0 host ID), broadcast addresses (with the all-1 host ID), and host addresses of the preceding four subnets.

Table 7-5 Network address and host address ranges (in binary format)

Subnet	Network Address	Broadcast Address	Host Address Range
10101100 00010000 00000000 00000000	10101100 00010000 00000000 00000000	10101100 00010000 00111111 11111111	10101100 00010000 00000000 00000001 to 10101100 00010000 00111111 11111110
10101100 00010000 01000000 00000000	10101100 00010000 01000000 00000000	10101100 00010000 01111111 11111111	10101100 00010000 01000000 00000001 to 10101100 00010000 01111111 11111110
10101100 00010000 10000000 00000000	10101100 00010000 10000000 00000000	10101100 00010000 10111111 11111111	10101100 00010000 10000000 00000001 to 10101100 00010000 10111111 11111110
10101100 00010000 11000000 00000000	10101100 00010000 11000000 00000000	10101100 00010000 11111111 11111111	10101100 00010000 11000000 00000001 to 10101100 00010000 11111111 11111110

Typically, the subnetting result is presented in decimal format. Converting a result in binary format to that in decimal format is complex. The following describes how to conveniently divide a network into subnets in decimal notation.

Take the preceding Class B IP address with the network address of 172.16.0.0 and the mask of 255.255.0.0 as an example. After subnetting, the subnet mask is 255.255.192.0. The length of the subnet mask equals 18 (16 + 2) bits. The calculation method is as follows:

- The number of bits in a subnet ID is *m*, and the number of subnets is 2^m. In this example, *m* equals 2, so the number of subnets is 4.
- The number of bits in a host ID is *n*, and the number of valid host addresses in each subnet is 2^n - 2 (excluding the number of host addresses that comprise all 1s or all 0s). In this example, *n* equals 14, so the number of valid host addresses in each subnet is 16382.
- The span between two neighboring subnet IDs is called block size, which is the value of deducting a subnet mask that is neither 0.0.0.0 nor 255.255.255.255 from 256. The first subnet network ID starts from 0 (located in the subnet mask in decimal format that is neither 0.0.0.0 nor 255.255.255.255), and the IDs of the following subnets increase individually by block size. In this example, the block size is 64 (256 - 192). The four

subnets are 172.16.**0**.0/18, 172.16.**64**.0/18, 172.16.**128**.0/18, and 172.16.**192**.0/18, respectively.

Table 7-6 lists the subnetting result in decimal format.

Table 7-6 Network address and host address ranges (in decimal format)

Subnet	Network Address	Broadcast Address	Host Address Range
172.16.0.0/18	172.16.0.0	172.16.63.255	172.16.0.1 to 172.16.63.254
172.16.64.0/18	172.16.64.0	172.16.127.255	172.16.64.1 to 172.16.127.254
172.16.128.0/18	172.16.128.0	172.16.191.255	172.16.128.1 to 172.16.191.254
172.16.192.0/18	172.16.192.0	172.16.255.255	172.16.192.1 to 172.16.255.254

Borrowing bits from the Host-id field to create a Subnet-id field reduces the number of supported hosts. For example, a Class B IP address can contain 65534 (2^{16} - 2) host addresses. If a 2-bit Subnet-id field is created, the four subnets have a maximum of 65528 [4 x (2^{14} - 2)] host addresses, which is six less than the number of host addresses when no 2-bit Subnet-id field is created.

To implement efficient network planning, subnetting and IPv4 addressing should abide by the rules described below.

Hierarchy

To divide a network into multiple layers, you need to consider geographic and service factors. Use a top-down subnetting mode to facilitate network management and simplify routing tables. Generally:

- A network consisting of a backbone network and a metro network is divided into hierarchical subnets.
- An administrative network is divided into subnets based on administrative levels.

Consecutiveness

Consecutive addresses facilitate route summarization on a hierarchical network, which greatly reduces the number of routing entries and improves route searching efficiency. When allocating IP addresses, note the following:

- Allocate consecutive IP addresses to each area.
- Allocate consecutive IP addresses to devices that have the same services and functions.

Scalability

When allocating addresses, reserve certain addresses at each layer to ensure consecutive address allocation in future network expansion.

A backbone network must have sufficient consecutive addresses for independent autonomous systems (ASs) and further network expansion.

Efficiency

When planning subnets, fully utilize address resources to ensure that the subnets are sufficient for hosts.

- Allocate IP addresses by using variable length subnet mask (VLSM) to fully utilize address resources.
- Consider the routing mechanism in subnetting to improve address utilization in the allocated address space.

7.1.2.5 IP Address Resolution

When using an IP address, note the following:

- An IP address is a network layer address of a host. To transmit datagrams to a destination host, the system must obtain the physical address of the host. Therefore, the IP address must be resolved to a physical address.
- A host name is easier to remember than an IP address. Therefore, the host name needs to be resolved to the IP address.

On an Ethernet, the physical address of a host is the MAC address. A DNS server resolves a host name to an IP address, and ARP resolves an IP address to a MAC address.

7.1.3 Configuration Precautions for IPv4 basic

Licensing Requirements

IPv4 Basics is not under license control.

Hardware Requirements

Table 7-7 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

7.1.4 Default Settings for IPv4 Basic

Table 7-8 describes the default settings for IPv4 basic.

Table 7-8 Default settings for IPv4 basic

Parameter	Default Setting
Function of processing IPv4 datagrams with the Source Route option	Enabled
Interface IPv4 address	Not configured
Maximum number of fragments that can be cached for an IPv4 datagram	32
Direct fragment forwarding	Disabled
Fragment discarding	Disabled
Fragment first in first out (FIFO)	Disabled
Function of discarding fragments that match a specified ACL	Disabled
Forced fragmentation for IPv4 datagrams on an outbound interface	Disabled
DSCP priority of datagrams	Not configured
SYN-Wait timer for TCP connections	20s
FIN-Wait timer for TCP connections	FIN-WAIT1 20s FIN-WAIT2 60s
Receive or send buffer size for a TCP socket	Send buffer size: 8 x 1460; receive buffer size: 7 x 1460
Maximum maximum segment size (MSS) value of a TCP connection	1460
Minimum MSS value of a TCP connection	32
MSS for a TCP packet that a device can receive	1460 bytes
TCP handshake check	Disabled

7.1.5 Configuring IP Addresses for an Interface

Prerequisites

Before configuring IP addresses for an interface, configure link layer protocol parameters for the interface to ensure that its link layer protocol status is up.

Context

To run IP services on an interface, configure IP addresses for the interface.

Each interface on a device can be configured with multiple IP addresses. If IP addresses are configured in primary/secondary mode, one primary and multiple secondary IP addresses can be configured on an interface.

If IP addresses are configured in primary/secondary mode, you need to configure only one primary IP address for an interface. In some special cases, you need to configure one or more secondary IP addresses for an interface. For example, a device connects to a physical network through one of its interfaces. Hosts on the physical network belong to two different Class C networks. To allow the device to communicate with all the hosts on the physical network, configure a primary IP address and a secondary IP address for the interface on the device.

For details about configuration parameters, see huawei-ip.yang.

NOTE

For IP addresses in primary/secondary mode:

- One primary IP address and multiple secondary IP addresses can be configured on an interface.
- If a secondary IP address exists, the primary IP address cannot be deleted.

Procedure

- **For IP addresses in primary/secondary mode:**
  a. Enter the edit-config view.
     edit-config
  b. Enter the interface view.
     ifm interfaces interface name *interface-name*
  c. Configure a primary IP address for the interface.
     ipv4 addresses address ip *ipv4-address*
     mask *ipv4-netmask* **type main**

     Each interface can have only one primary IP address. If an interface already has a primary IP address, configuring another primary IP address for the interface fails.
  d. (Optional) Configure a secondary IP address for the interface.
     ipv4 addresses address ip *ipv4-address*
     mask *ipv4-netmask* **type sub**

     To save IP address space, you can configure secondary IP addresses with 31-bit masks for an interface. A maximum of 255 secondary IP addresses can be configured for each interface.

----End

Verifying the Configuration

- Run the **display /ifm/interfaces/interface[name="GE0/0/0"]/ipv4/state/addresses/address all** command to check the IP address configuration of the interface.
- Run the **display /ifm/interfaces/interface[name="***interface-name***"]** command to check information about an interface.

Example

Set the primary and secondary IP addresses of DeviceA's GE 0/0/0 to 172.16.1.1 and 172.16.2.1, respectively.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> ifm interfaces interface name GE0/0/0
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/0"]
MDCLI> ip addresses address ip 172.16.1.1
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="172.16.1.1"]
MDCLI> mask 255.255.255.0 type main
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="172.16.1.1"]
MDCLI> quit 3
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/0"]
MDCLI> ip addresses address ip 172.16.2.1
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="172.16.2.1"]
MDCLI> mask 255.255.255.0 type sub
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="172.16.2.1"]
MDCLI> commit
```

NOTE

Both DeviceA and DeviceB use GE 0/0/0 for connection. Set the primary and secondary IP addresses of DeviceB's GE 0/0/0 to 172.16.1.2 and 172.16.2.2, respectively. For configuration details, see the configuration of DeviceA.

Verify the configuration.

- Ping a host on the network segment 172.16.1.0 from DeviceA. The ping operation is successful.

```
[DeviceA] ping 172.16.1.2
 PING 172.16.1.2: 56  data bytes, press CTRL_C to break
   Reply from 172.16.1.2: bytes=56 Sequence=1 ttl=255 time=614 ms
   Reply from 172.16.1.2: bytes=56 Sequence=2 ttl=255 time=16 ms
   Reply from 172.16.1.2: bytes=56 Sequence=3 ttl=255 time=2 ms
   Reply from 172.16.1.2: bytes=56 Sequence=4 ttl=255 time=3 ms
   Reply from 172.16.1.2: bytes=56 Sequence=5 ttl=255 time=2 ms
 --- 172.16.1.2 ping statistics ---
   5 packet(s) transmitted
   5 packet(s) received
   0.00% packet loss
   round-trip min/avg/max = 25/26/27 ms
```

- Ping a host on the network segment 172.16.2.0 from DeviceA. The ping operation is successful.

```
[DeviceA] ping 172.16.2.2
 PING 172.16.2.2: 56  data bytes, press CTRL_C to break
   Reply from 172.16.2.2: bytes=56 Sequence=1 ttl=255 time=13 ms
   Reply from 172.16.2.2: bytes=56 Sequence=2 ttl=255 time=2 ms
   Reply from 172.16.2.2: bytes=56 Sequence=3 ttl=255 time=2 ms
   Reply from 172.16.2.2: bytes=56 Sequence=4 ttl=255 time=2 ms
   Reply from 172.16.2.2: bytes=56 Sequence=5 ttl=255 time=2 ms
 --- 172.16.2.2 ping statistics ---
   5 packet(s) transmitted
   5 packet(s) received
   0.00% packet loss
   round-trip min/avg/max = 25/25/26 ms
```

7.1.6 Maintaining IPv4 Basic

7.1.6.1 Displaying the IPv4 Operating Status

Procedure

During routine maintenance, you can run the following commands to check the IPv4 operating status.

For details about configuration parameters, see huawei-diagnose-hps.yang.

Table 7-9 Displaying the IPv4 operating status

Operation	Command
Display the status of IPv4 RawIP connections.	**diagnose hps query-rawip-status** [**pid** *pid*] [**fd** *fd*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*]
Display the status of IPv4 RawLink connections.	**diagnose hps query-rawlink-status** [**pid** *pid*] [**fd** *fd*]
Display the current TCP connection status.	**diagnose hps query-tcp-status** [**pid** *pid*] [**fd** *fd*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**source-port** *source-port*] [**destination-port** *destination-port*] [**control-block-id** *control-block-id*]
Display the status of all IPv4 UDP connections.	**diagnose hps query-udp-status** [**pid** *pid*] [**fd** *fd*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**source-port** *source-port*] [**destination-port** *destination-port*]
Display information about created IPv4 sockets. If no optional parameter is specified, this command displays information about all types of sockets.	**diagnose hps query-ip-socket** [**type** { *tcp* | *udp* | *rawip* | *rawlink* }]
Display the status of IPv4 TCP control blocks.	**diagnose hps query-tcp-control-block** [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**source-port** *source-port*] [**destination-port** *destination-port*]
Display statistics about RawIP packets.	**diagnose hps query-statistics type** *rawip*
Display statistics about RawLink packets.	**diagnose hps query-statistics type** *rawlink*

Operation	Command
Display statistics about TCP packets.	**diagnose hps query-statistics type** *tcp*
Display statistics about UDP packets.	**diagnose hps query-statistics type** *udp*
Reset statistics about RawIP packets.	**diagnose hps reset-statistics type** *rawip*
Reset statistics about RawLink packets.	**diagnose hps reset-statistics type** *rawlink*
Reset statistics about TCP packets.	**diagnose hps reset-statistics type** *tcp*
Reset statistics about UDP packets.	**diagnose hps reset-statistics type** *udp*
Debug IPv4 RawIP packets.	**diagnose hps debugging-rawip-packet** [**enable** [*true* | *false*]] [**packet-number** [*number*]] [**source-or-destination-ip** *ip*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*]
Debug IPv4 RawLink packets.	**diagnose hps debugging-rawlink-packet** [**enable** [*true* | *false*]] [**packet-number** [*number*]] [**source-or-destination-mac** *source-or-destination-mac*] [**source-mac** *source-mac*] [**destination-mac** *destination-mac*] [**verbose** [*true* | *false*]]
Debug IPv4 TCP packets.	**diagnose hps debugging-tcp-packet** [**enable** [*true* | *false*]] [**packet-number** [*number*]] [**source-or-destination-ip** *ip*] [**source-or-destination-port** *port*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**source-port** *source-port*] [**destination-port** *destination-port*]
Debug IPv4 UDP packets.	**diagnose hps debugging-udp-packet** [**enable** [*true* | *false*]] [**packet-number** [*number*]] [**source-or-destination-ip** *ip*] [**source-or-destination-port** *port*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**source-port** *source-port*] [**destination-port** *destination-port*]
Debug IPv4 IP packets.	**diagnose hps debugging-ip-packet** [**enable** [*true* | *false*]] [**packet-number** [*number*]] [**source-or-destination-ip** *ip*] [**source-or-destination-port** *p*ort] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**ifindex** *ifindex*] [**verbose** [*true* | *false*]]

Operation	Command
Debug IPv4 TCP state transition.	**diagnose hps debugging-tcp-event** [**enable** [*true* | *false*]] [**packet-number** [*number*]] [**source-or-destination-ip** *ip*] [**source-or-destination-port** *port*] [**source-ip** *source-ip*] [**destination-ip** *destination-ip*] [**source-port** *source-port*] [**destination-port** *destination-port*]

7.2 Load Balancing Configuration

7.2.1 Overview of Load Balancing

Definition

Load balancing distributes traffic among multiple available links to the same destination.

Purpose

- Network resources are fully utilized.
- If a link used for load balancing fails, traffic can be forwarded through other links, improving link reliability.

7.2.2 Understanding Load Balancing

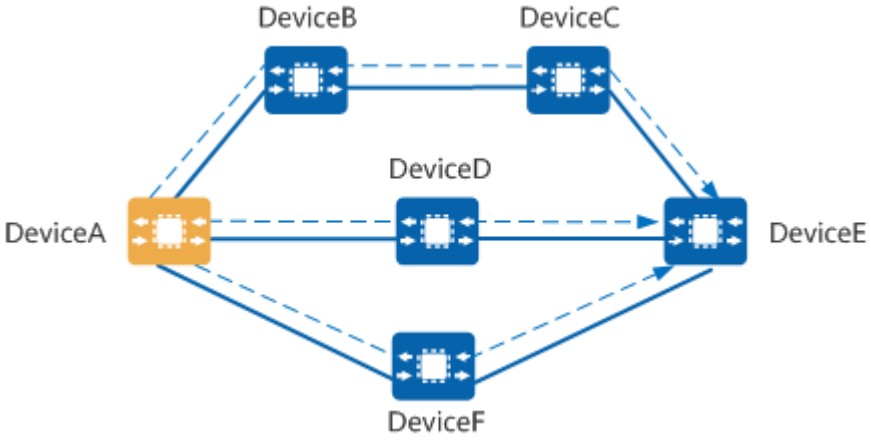
7.2.2.1 Definition and Classification of Load Balancing

Load balancing allows a network node to distribute traffic among multiple links for forwarding. It is classified as route, tunnel, or trunk load balancing.

Route Load Balancing

Route load balancing enables traffic to be balanced among multiple forwarding paths to the same destination. If there are multiple routes with the same destination address and mask but different next hops, outbound interfaces, or tunnel IDs, route load balancing can be implemented, as shown in **Figure 7-6**.

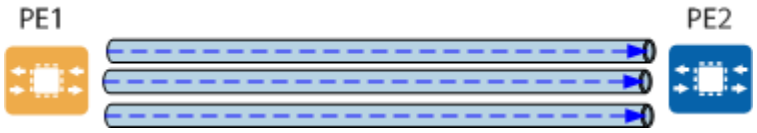
Figure 7-6 Route load balancing



Tunnel Load Balancing

On a VPN, tunnel load balancing enables traffic to be balanced among the ingress PE's multiple tunnels that are destined for the same destination PE, as shown in **Figure 7-7**.

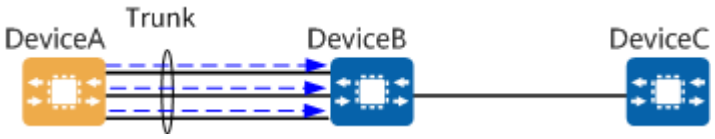
Figure 7-7 Tunnel load balancing



Trunk Load Balancing

Trunk load balancing enables traffic to be balanced among multiple member links of a trunk after multiple physical interfaces with the same link layer protocol are bundled into the trunk, as shown in **Figure 7-8**.

Figure 7-8 Trunk load balancing



7.2.2.2 Per-Flow and Per-Packet Load Balancing

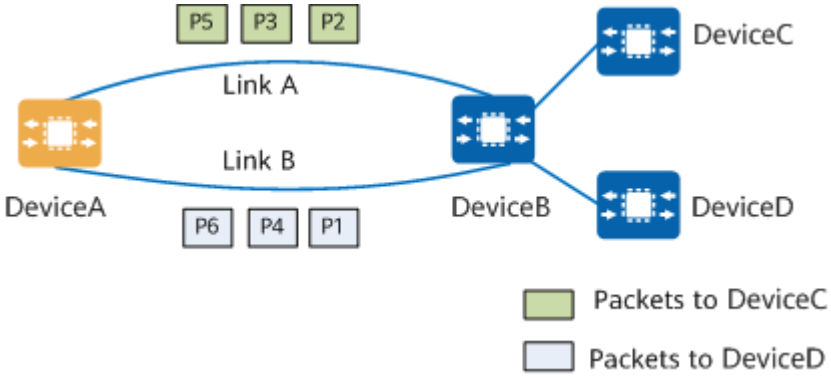
Load balancing works in per-flow or per-packet mode, irrespective of whether it is route, tunnel, or trunk load balancing.

Per-Flow Load Balancing

Per-flow load balancing classifies packets into different flows based on a certain rule, such as the IP 5-tuple (source IP address, destination IP address, protocol number, source port number, and destination port number). Packets of the same flow go over the same link.

On the network shown in **Figure 7-9**, DeviceA sends six packets, P1, P2, P3, P4, P5, and P6 in sequence to DeviceB over links A and B in load balancing mode. P2, P3, and P5 are destined for DeviceC, and P1, P4, and P6 for DeviceD. If per-flow load balancing is used, packets destined for DeviceC can go over link A, and packets destined for DeviceD can go over link B. Alternatively, packets destined for DeviceC can go over link B, and packets destined for DeviceD can go over link A.

Figure 7-9 Network diagram of per-flow load balancing



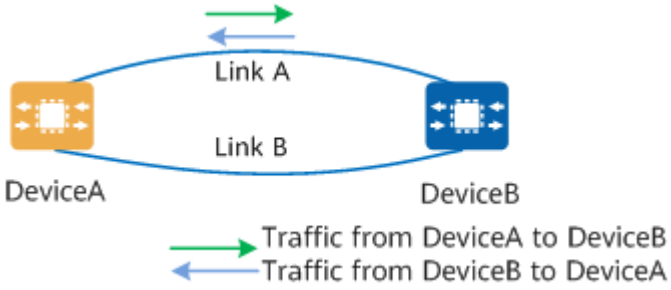
Symmetric load balancing

Symmetric load balancing is a special per-flow load balancing mode.

This mode distinguishes data flows based on the IP addresses of packets. In this way, the same data flow goes over the same link.

On the network shown in **Figure 7-10**, DeviceA forwards the same data flow to DeviceB over link A; DeviceB obtains the same link index by exchanging the source and destination IP addresses, and then hashes the flow to link A.

Figure 7-10 Network diagram of symmetric load balancing

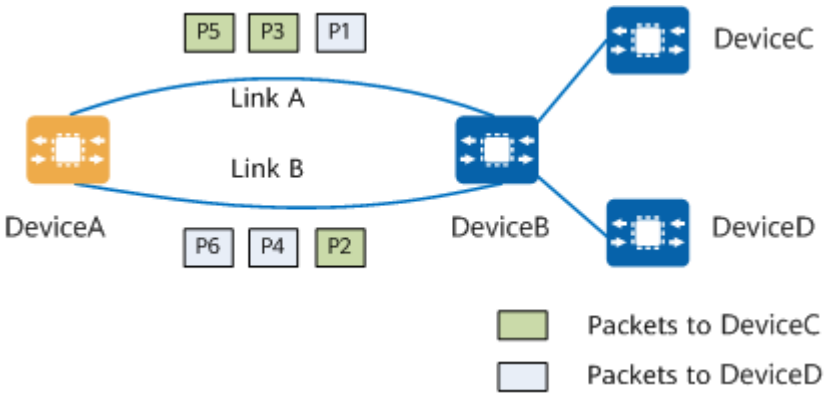


Symmetric load balancing ensures packet sequence but not bandwidth utilization.

Per-Packet Load Balancing

Per-packet load balancing evenly distributes packets among links used for load balancing based on their incoming sequence, as shown in **Figure 7-11**.

Figure 7-11 Network diagram of per-packet load balancing



7.2.2.3 ECMP and UCMP

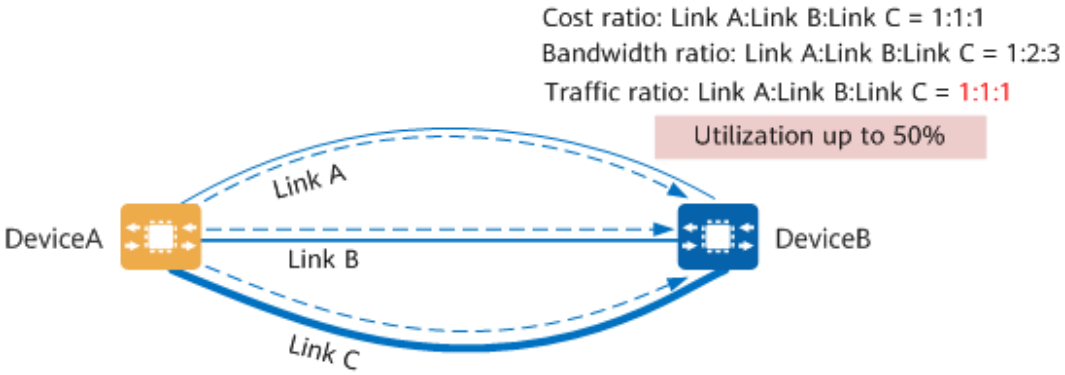
Route load balancing can be further classified as equal-cost multi-path (ECMP) or unequal cost multipath (UCMP).

ECMP

ECMP evenly balances traffic among multiple equal-cost paths to the same destination, irrespective of bandwidth. Equal-cost paths have the same cost to the destination.

When these paths have very different bandwidths, the bandwidth utilization is low. On the network shown in **Figure 7-12**, traffic is balanced among three paths, with the bandwidth of 10 Mbit/s, 20 Mbit/s, and 30 Mbit/s, respectively. If ECMP is used, the total bandwidth can reach 30 Mbit/s, but the highest bandwidth utilization can only reach 50%.

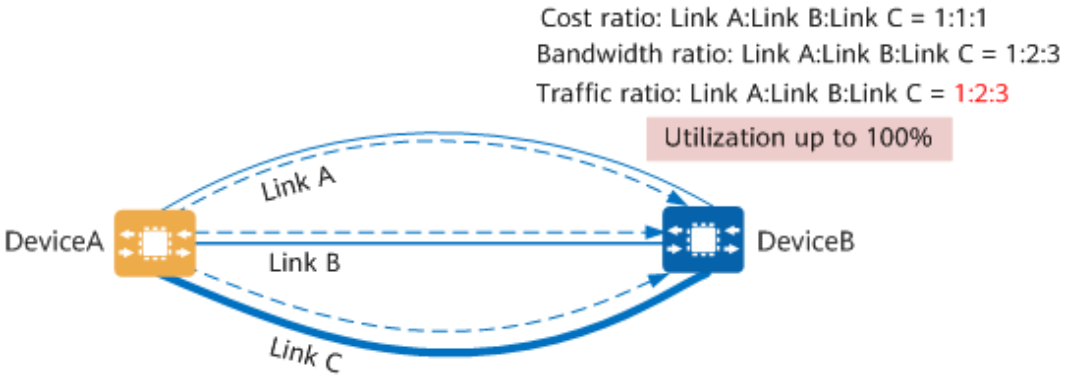
Figure 7-12 ECMP networking



UCMP

On the network shown in **Figure 7-13**, UCMP proportionally balances traffic among multiple equal-cost paths to the same destination based on different bandwidths, improving bandwidth utilization.

Figure 7-13 UCMP networking



7.2.3 Default Settings for Load Balancing

Table 7-10 Default settings for load balancing

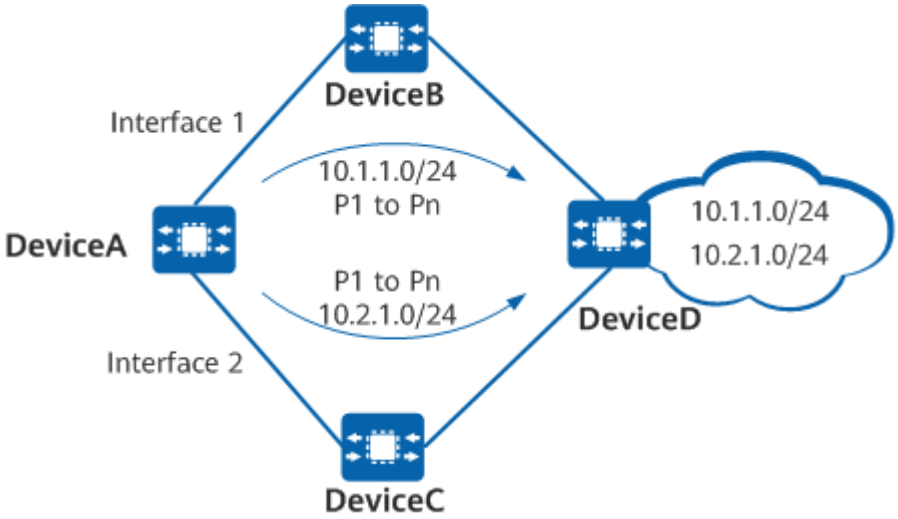
Parameter	Default Setting
Load balancing mode for IP packets	Combination of the source IP address (src-ip), destination IP address (dst-ip), transport-layer source port number (src-port), and transport-layer destination port number (dst-port)

7.2.4 Configuring ECMP

7.2.4.1 Understanding ECMP

ECMP applies to a network where multiple links to the same destination are available. If traditional routing technology is used, packets are forwarded to the destination through only one link; the other links remain in the backup or inactive state; switching between links requires a certain period when dynamic routes are used. Unlike traditional routing technology, ECMP can use multiple links to increase transmission bandwidth and to take over traffic from a faulty link without any delay or packet loss. Load balancing can be performed among multiple routes discovered by the same routing protocol if they have the same destination and cost. **Figure 7-14** shows the packet forwarding process when the load balancing mode is set to 5-tuple.

Figure 7-14 Network diagram of load balancing



DeviceA has forwarded the first packet P1 to 10.1.1.0/24 through interface 1 and needs to forward subsequent packets to 10.1.1.0/24 and 10.2.1.0/24. The forwarding process is as follows:

- When forwarding the second packet P2 to 10.1.1.0/24, DeviceA finds that the 5-tuple information of this packet is the same as that of the first packet P1 to 10.1.1.0/24. Therefore, DeviceA forwards all subsequent packets to 10.1.1.0/24 through interface 1.
- When forwarding the first packet P1 to 10.2.1.0/24, DeviceA finds that the 5-tuple information of this packet is different from that of the first packet P1 to 10.1.1.0/24. Therefore, DeviceA forwards this packet and all subsequent packets to 10.2.1.0/24 through interface 2.

7.2.4.2 Configuring an ECMP Mode

Context

Devices support load balancing for IP and tunnel packets.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the load-balance-ecmp-profile view.

```
loadbalance ip-load-balance
```

Step 3 (Optional) Configure a load balancing mode.

```
mode flow
```

Step 4 Configure hash factors for load balancing.

```
{ src-ip | dst-ip | src-port | dst-port }* true
```

Load balancing can be implemented based on any combination of the source IP address, destination IP address, transport-layer source port number, and transport-layer destination port number.

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display loadbalance/ip-load-balance all** command to check the load balancing configuration.

7.3 ARP Configuration

7.3.1 Overview of ARP

Definition

The Address Resolution Protocol (ARP) resolves IP addresses to MAC addresses.

Purpose

If two hosts need to communicate, the sender must know the network-layer IP address of the receiver. IP datagrams, however, must be encapsulated with MAC addresses before they can be transmitted over the physical network. If the sender does not know the MAC address of the receiver, ARP is needed to map the receiver's IP address to the receiver's MAC address.

Benefits

ARP maps IP addresses at the network layer to MAC addresses at the data link layer on Ethernet networks to ensure communication between the data link and network layers.

7.3.2 Understanding ARP

ARP Message Format

Figure 7-15 shows the ARP message format. An Ethernet frame in which an ARP message is encapsulated in the payload is called an ARP frame. An ARP message is only 28 bytes long. The minimum data length of an Ethernet frame is 46 bytes and the maximum data length is 1500 bytes. If the data length of an Ethernet frame is less than 46 bytes, padding bytes are added to meet the requirement for the minimum data length. The padding length may vary with devices. The minimum padding length is 18 bytes.

Figure 7-15 ARP message format

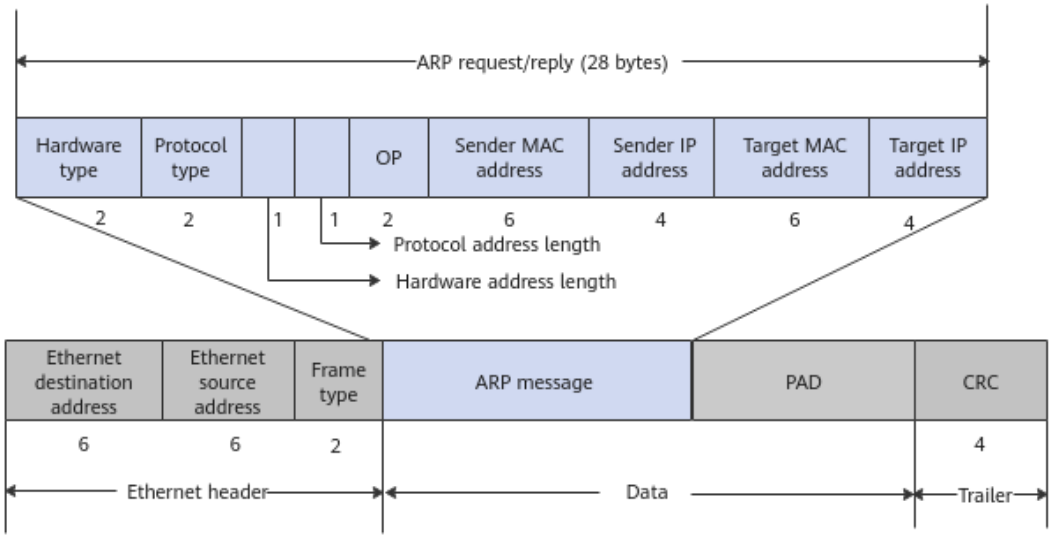


Table 7-11 Description of main fields

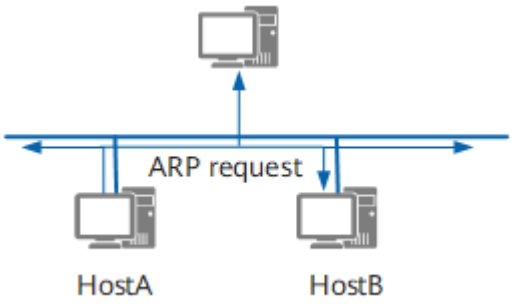
Field	Description
Ethernet destination address	Ethernet destination MAC address. For an ARP request message, the value of this field is the broadcast MAC address 0xFF-FF-FF-FF-FF-FF.
Ethernet source address	Ethernet source MAC address.
Frame type	Data type. For an ARP request or reply message, the value of this field is 0x0806.
Data	Payload. The minimum data length of an Ethernet frame is 46 bytes and the maximum data length is 1500 bytes. If the data length of an Ethernet frame is less than 46 bytes, padding bytes are added to meet the requirement for the minimum data length.
Hardware type	Hardware address type. For an Ethernet network, the value of this field is 1.
Protocol type	Mapped protocol address type. For an IP address, the value of this field is 0x0800.
Hardware address length	Hardware address length. For an ARP request or reply message, the value of this field is 6.

Field	Description
Protocol address length	Protocol address length. For an ARP request or reply message, the value of this field is 4.
OP	Operation type. Values 1 and 2 indicate an ARP request and reply, respectively.
Sender MAC address	MAC address of the sender.
Sender IP address	IP address of the sender.
Target MAC address	Destination MAC address. For an ARP request message, the value of this field is 0x00-00-00-00-00-00.
Target IP address	IP address of the receiver.
CRC	Cyclic redundancy check, which is used to check whether an error occurs in the frame.

Address Resolution Process

ARP uses ARP request and reply messages to complete address resolution.

Figure 7-16 ARP request

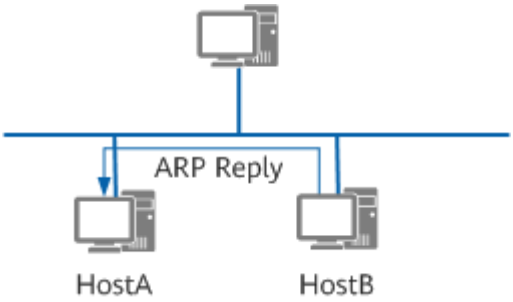


As shown in **Figure 7-16**, HostA and HostB are on the same network segment, and HostA needs to send IP datagrams to HostB.

HostA searches the local ARP table for the ARP entry corresponding to HostB. If the corresponding ARP entry is found, HostA encapsulates the IP datagrams into Ethernet frames and forwards them to HostB based on its MAC address.

If the corresponding ARP entry is not found, HostA caches the IP datagrams and broadcasts an ARP request message. In the ARP request message, the IP and MAC addresses of the sender are the IP and MAC addresses of HostA. The destination IP address is the IP address of HostB, and the destination MAC address comprises all 0s. All hosts on the same network segment can receive the ARP request message, but only HostB processes the message.

Figure 7-17 ARP reply



HostB compares its IP address with the destination IP address in the ARP request message. If the two addresses are the same, HostB adds the IP and MAC addresses of the sender (HostA) to the local ARP table. HostB then unicasts an ARP reply message, which contains its MAC address, to HostA, as shown in **Figure 7-17**.

After receiving the ARP reply message, HostA adds HostB's MAC address into the local ARP table. In addition, HostA encapsulates the IP datagrams and forwards them to HostB.

7.3.3 Configuration Precautions for ARP

Licensing Requirements

ARP is not under license control.

Hardware Requirements

Table 7-12 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

7.3.4 Default Settings for ARP

Table 7-13 Default settings for ARP

Parameter	Default Setting
Static ARP entries	Not configured

Parameter	Default Setting
Interface-based aging time and global aging time of dynamic ARP entries	1200s (interface-based); 1200s (global)
Number of probes for aging dynamic ARP entries	3
Probe interval for aging dynamic ARP entries	5s
Mode in which an interface sends ARP aging probe messages	Unicast mode
Routed proxy ARP	Disabled
Intra-VLAN proxy ARP	Disabled
Layer 2 proxy ARP	Disabled
IP address conflict detection	Enabled
Host IP address conflict detection options	Detection period: 5s; maximum number of retransmissions: 5
Active ARP learning	Disabled

7.3.5 Configuring Static ARP

7.3.5.1 Understanding Static ARP

Definition

Static ARP allows a network administrator to create a mapping between IP and MAC addresses.

Purpose

Configuring static ARP entries improves communication security. If a static ARP entry is configured on a device, the device can communicate with the peer device using only the specified MAC address. This improves communication security, because network attackers cannot modify the mapping between the IP and MAC addresses using ARP messages. Static ARP applies to networks with simple topologies, high stability, and high information security requirements.

Table 7-14 Static ARP entries

Type	Description
Static ARP entries	In addition to IP and MAC addresses, a static ARP entry contains an outbound interface. Static ARP entries can be directly used to forward messages. Configure a static ARP entry if you want a device and host to communicate only through the specified interface.

Benefits

To ensure communication stability and security, deploy static ARP based on actual requirements and network resources.

- IP addresses can be bound to the MAC address of a specified gateway to ensure that only this gateway forwards the IP datagrams destined to these IP addresses.
- The destination IP addresses of certain IP datagrams sent by a specified host can be bound to a nonexistent MAC address, helping filter out unnecessary IP datagrams.

7.3.5.2 Configuring Static ARP

Prerequisites

Before configuring static ARP, you have completed the following tasks:

- Connect interfaces and configure physical parameters for the interfaces to ensure that the physical status of the interfaces is up.
- Configure link layer protocol parameters for the interfaces to ensure that the link layer protocol status of the interfaces is up.

Context

You can deploy static ARP on important network devices such as servers to set up a static mapping between IP and MAC addresses of the peers communicating with the devices. The mapping cannot be modified by forged ARP messages, and it prevents the devices from responding to invalid ARP request messages. In this way, the devices are protected against network attacks.

For details about configuration parameters, see huawei-arp.yang.

NOTE

Static ARP entries will never be overwritten, but configuring a large number of ARP entries is laborious. Therefore, static ARP is applicable to small networks on which host IP addresses seldom change.

Procedure

Step 1 Enter the edit-config view.

edit-config

Step 2 Enter the interface view.

ifm interfaces interface name *interface-name*

Step 3 Configure the IP address of a static ARP entry.

ipv4 static-arps static-arp ip-addr *ip-address*

Step 4 Configure the MAC address of a static ARP entry.

mac-addr *mac-address*

Step 5 Configure the PEVID of a static ARP entry. If the interface is a Layer 2 interface, the PEVID is the ID of the VLAN to which the interface is added. If the interface is a Layer 3 interface, the PEVID is optional (if configured, the value must be 0).

pevid vlanid

Step 6 Commit the configuration.

commit

----End

Verifying the Configuration

Run the **display /arp/query-entries all** command to check the static ARP mapping table.

7.3.6 Configuring Dynamic ARP

7.3.6.1 Understanding Dynamic ARP

Definition

Dynamic ARP allows devices to dynamically learn and update the mapping between IP and MAC addresses through ARP messages. That is, you do not need to manually configure the mapping. Dynamic ARP entries are dynamically created and updated using ARP messages. In this way, they do not need to be manually maintained, greatly reducing maintenance workload.

Creating and Updating Dynamic ARP Entries

After receiving an ARP message whose source IP address is on the same network segment as the IP address of the inbound interface, a device automatically creates or updates an ARP entry if the message meets either of the following conditions:

- The destination IP address is the IP address of the inbound interface.
- The destination IP address is the Virtual Router Redundancy Protocol (VRRP) virtual IP address of the inbound interface.

Aging Dynamic ARP Entries

After the aging timer of a dynamic ARP entry on a device expires, the device sends ARP aging probe messages to the peer device. In this case, if the device does not

receive an ARP reply message after sending a specified maximum number of aging probe messages, the dynamic ARP entry is aged.

Application Scenarios

The dynamic ARP aging mechanism ensures that ARP entries unused during a specified period are automatically deleted. This mechanism helps save the storage space of ARP tables and speed up ARP table lookups. Dynamic ARP applies to networks with complex topologies and high real-time communication requirements.

Table 7-15 Dynamic ARP aging mechanism

Concept	Description	Scenario
Aging probe mode	Before a dynamic ARP entry on a device is aged, the device sends ARP aging probe messages to other devices on the same network segment. An ARP aging probe message can be a unicast or broadcast message. Currently, ARP aging probe messages can only be unicast.	If the IP address of the peer device remains unchanged but its MAC address changes frequently, it is recommended that you configure the local device to broadcast ARP aging probe messages. If the MAC address of the peer device remains unchanged, network bandwidth resources are insufficient, and the aging time of ARP entries is set to a small value, it is recommended that you configure the local device to unicast ARP aging probe messages. Currently, ARP aging probe messages can only be unicast.
Aging time	Every dynamic ARP entry has a lifecycle, which is also called aging time. If a dynamic ARP entry is not updated after its lifecycle ends, this dynamic ARP entry is deleted from the ARP table.	Two interconnected devices can use ARP to learn the mapping between their IP and MAC addresses and save the mapping in their ARP tables. Then, the two devices can communicate using the ARP entries. When the peer device becomes faulty or its NIC is replaced but the local device does not receive any status change information about the peer device, the local device continues to send IP datagrams to the peer device. As a result, network traffic is interrupted because the ARP table of the local device is not promptly updated. To reduce the risk of network traffic interruptions, an aging timer can be set for each ARP entry. After the aging timer of a dynamic ARP entry expires, the entry is automatically deleted.

Concept	Description	Scenario
Number of probes for aging dynamic ARP entries	Before a dynamic ARP entry is aged, a device sends ARP aging probe messages to the peer device. If the device does not receive an ARP reply message after sending a specified maximum number of aging probe messages, the dynamic ARP entry is deleted. The current number of probes for aging dynamic ARP entries is 3 and cannot be configured.	The ARP aging timer can help reduce the risk of network traffic interruptions that occur because an ARP table is not updated quickly enough, but it cannot eliminate problems caused by delays. For example, if the aging time of a dynamic ARP entry is N seconds, the local device can detect the status change of the peer device after N seconds. During this period, the ARP table of the local device is not updated. You can set the number of probes for aging dynamic ARP entries to ensure that the ARP table is updated in time in the preceding situation.

7.3.6.2 Configuring Dynamic ARP

Prerequisites

Before configuring dynamic ARP, you have completed the following tasks:

- Connect interfaces and configure physical parameters for the interfaces to ensure that the physical status of the interfaces is up.
- Configure link layer protocol parameters for the interfaces to ensure that the link layer protocol status of the interfaces is up.

Context

Hosts and devices can learn dynamic ARP entries by default. You can adjust dynamic ARP aging parameters or optimize the ARP entry update policy as required. For details about configuration parameters, see huawei-arp.yang.

NOTE

- If the aging time of dynamic ARP entries is too short, for example, 1 minute, the device will be busy updating dynamic ARP entries. This consumes a lot of system resources and affects the processing of other services.
- If the aging time of dynamic ARP entries is configured both globally and on an interface, the aging time configured on the interface takes precedence. For example, if the global aging time and interface-based aging time are set to 300s and 60s respectively, dynamic ARP entries are updated every 60s.

Procedure

- Configure the global aging parameters of dynamic ARP entries.
  a. Enter the edit-config view.
     edit-config
  b. Enter the global ARP view.
     arp global
  c. Configure an aging time for dynamic ARP entries.
     expire-time *expire-time-values*
  d. Commit the configuration.
     commit
- Configure the aging parameters of dynamic ARP entries in the view of an interface except Layer 2, null, and lo interfaces.
  a. Enter the edit-config view.
     edit-config
  b. Enter the interface view.
     ifm interfaces interface name *interface-name*
  c. Enter the ARP entry view.
     arp-entry
  d. Configure an aging time for dynamic ARP entries.
     expire-time *expire-time-value*
  e. Commit the configuration.
     commit

----End

Verifying the Configuration

- Run the **display /arp/global/expire-time all** command to check the global aging time of dynamic ARP entries.
- Run the **display /ifm/interfaces/interface**[**name=**"*interface-name*"]**/arp-entry/expire-time all** command to check the interface-based aging time of dynamic ARP entries.

7.3.7 Configuring Proxy ARP

7.3.7.1 Understanding Proxy ARP

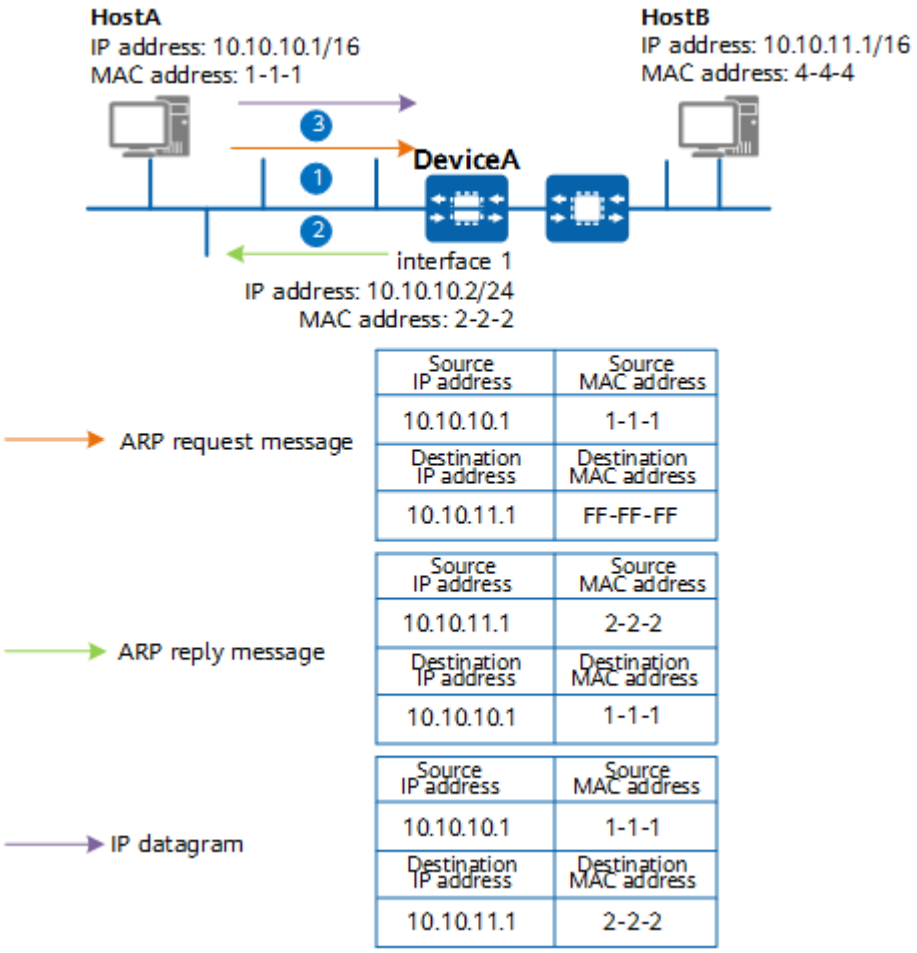
ARP is applicable only to devices on the same physical network. When a device on a physical network needs to send IP datagrams to another physical network, the gateway is used to query the routing table to implement communication between the two networks. However, routing table query consumes system resources and affects other services. To resolve this problem, deploy proxy ARP on an intermediate device. Proxy ARP enables devices that reside on different physical network segments but on the same IP network to resolve IP addresses to MAC addresses. This feature helps reduce system resource consumption caused by routing table queries and improves the efficiency of system processing.

Routed Proxy ARP

A large company network is usually divided into multiple subnets to facilitate management. The routing information of a host in a subnet can be modified so that IP datagrams sent from this host to another subnet are first sent to the gateway that connects different subnets and then to another subnet. However, this solution makes it difficult to manage and maintain devices. If the gateways connected to hosts have different addresses, you can deploy routed proxy ARP so that the gateways send their interface MAC addresses to the hosts.

Figure 7-18 illustrates how routed proxy ARP is implemented between HostA and HostB.

Figure 7-18 Routed proxy ARP implementation



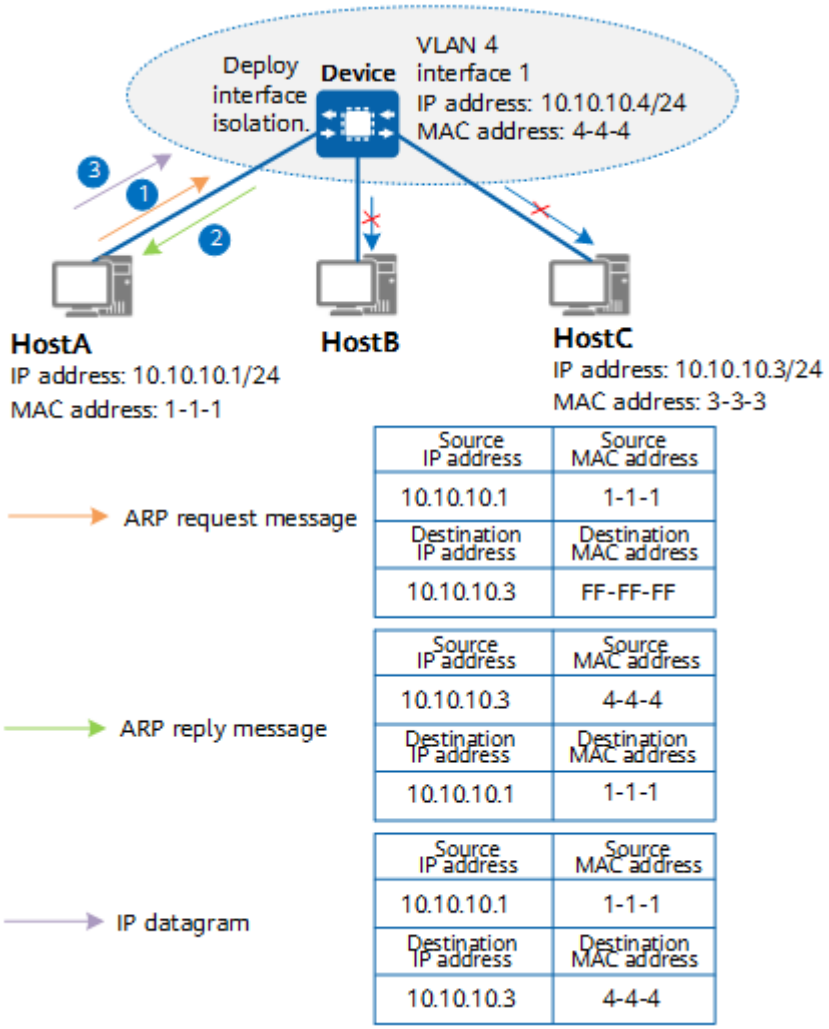
1. HostA sends an ARP request message for the MAC address of HostB.
2. After receiving the ARP request message, DeviceA checks the destination IP address of the message and finds that the requested MAC address is not its own MAC address. DeviceA then checks whether there are routes to HostB. If a route to HostB is available, DeviceA checks whether routed proxy ARP is enabled on interface 1. If routed proxy ARP is enabled on interface 1, DeviceA sends interface 1's MAC address to HostA. If routed proxy ARP is not enabled on interface 1, DeviceA discards the ARP request message. If no route to HostB is available, DeviceA discards the ARP request message.

3. After learning interface 1's MAC address, HostA sends IP datagrams to DeviceA using this MAC address.
4. DeviceA receives the IP datagrams and forwards them to HostB.

Intra-VLAN Proxy ARP

Figure 7-19 illustrates how intra-VLAN proxy ARP is implemented between HostA and HostC.

Figure 7-19 Intra-VLAN proxy ARP implementation



HostA, HostB, and HostC belong to the same VLAN, but HostA and HostC cannot communicate at Layer 2 because interface isolation is enabled on Device. To allow HostA and HostC to communicate, configure interface 1 on Device and enable intra-VLAN proxy ARP.

1. HostA sends an ARP request message for the MAC address of HostC.
2. After receiving the ARP request message, Device checks the destination IP address of the message and finds that the requested MAC address is not the MAC address of interface 1. Device then searches its ARP table for the ARP entry indicating the mapping between the IP and MAC addresses of HostC. If

Device finds this ARP entry in its ARP table, it checks whether intra-VLAN proxy ARP is enabled on interface 1.

- If intra-VLAN proxy ARP is enabled on interface 1, Device sends the MAC address of interface 1 to HostA.
- If intra-VLAN proxy ARP is not enabled on interface 1, Device discards the ARP request message.

If Device does not find this ARP entry in its ARP table, it discards the ARP request message and checks whether intra-VLAN proxy ARP is enabled.

- If intra-VLAN proxy ARP is enabled, Device broadcasts the ARP request message with the IP address of HostC as the destination IP address within VLAN 4. After receiving an ARP reply message from HostC, Device generates an ARP entry indicating the mapping between the IP and MAC addresses of HostC.
- If intra-VLAN proxy ARP is not enabled, Device does not perform any operations.

3. After learning the MAC address of interface 1, HostA sends IP datagrams to Device using this MAC address.
4. Device receives the IP datagrams and forwards them to HostC.

Application Scenarios

Table 7-16 Application scenarios of proxy ARP

Proxy ARP Type	Application Scenario
Routed proxy ARP	Two hosts that need to communicate belong to the same network segment but different physical networks. The gateways to which hosts are connected have different IP addresses.
Intra-VLAN proxy ARP	Two hosts that need to communicate belong to the same network segment and the same VLAN in which user isolation is configured.

Benefits

- Proxy ARP enables the source host on a network to mistakenly consider that the destination host and itself are on the same network segment. In this way, the network details can be hidden, thereby achieving transparent subnet division.
- All operations related to proxy ARP are performed on a gateway. No configuration is required for hosts connected to the gateway. In addition, proxy ARP affects only the ARP tables on hosts and does not affect the ARP table or routing table on a gateway.
- Proxy ARP can be used when no default gateway is configured for a host or when a host cannot route messages.

7.3.7.2 Configuring Routed Proxy ARP

Context

Hosts that belong to the same network segment but different physical networks are unable to communicate with each other if the gateways connected to the hosts have different IP addresses. In this case, you can enable routed proxy ARP on a device's interface connected to the hosts to enable the hosts to communicate. For details about configuration parameters, see huawei-arp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the view of an interface except Layer 2, null, and lo interfaces.

```
ifm interfaces interface name interface-name
```

Step 3 Enable or disable routed proxy ARP on the interface.

```
arp-entry
route-proxy-enable { true | false }
```

After routed proxy ARP is enabled on a device, the aging time of ARP entries on hosts must be reduced. This ensures that invalid ARP entries are aged out as soon as possible, reducing the number of packets that are sent to but cannot be forwarded by the device.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display ifm/interfaces/interface** [**name=***interface-name*] **/arp-entry** command to check the routed proxy ARP configuration on an interface.

7.3.7.3 Configuring Intra-VLAN Proxy ARP

Context

Hosts that belong to the same VLAN are unable to communicate with each other if Layer 2 interface isolation is configured in the VLAN. In this case, you can enable intra-VLAN proxy ARP on a device's interface associated with the VLAN to enable the hosts to communicate.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the view of an interface except Layer 2, null, and lo interfaces.

```
ifm interfaces interface name interface-name
```


Step 3 Enable or disable intra-VLAN proxy ARP on the interface.

```
arp-entry
inner-proxy-enable { true | false }
```

After intra-VLAN proxy ARP is enabled on a device, the aging time of ARP entries on hosts must be reduced. This ensures that invalid ARP entries are aged out as soon as possible, reducing the number of packets that are sent to but cannot be forwarded by the device.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display ifm/interfaces/interface** [**name=***interface-name*] **/arp-entry** command to check the intra-VLAN proxy ARP configuration on an interface.

7.3.8 Configuring Layer 2 Proxy ARP

7.3.8.1 Understanding Layer 2 Proxy ARP

Definition

After receiving an ARP request message, a device broadcasts the message in its broadcast domain (BD). If a device receives a large number of ARP request messages within a period of time and broadcasts the messages, a large amount of network resources are consumed, causing network congestion. As a result, network performance deteriorates and user services are affected. Layer 2 proxy ARP can relieve the pressure on processing ARP messages by isolating ARP BDs. With this function enabled, a device preferentially uses learned ARP snooping entries to respond to received ARP request messages.

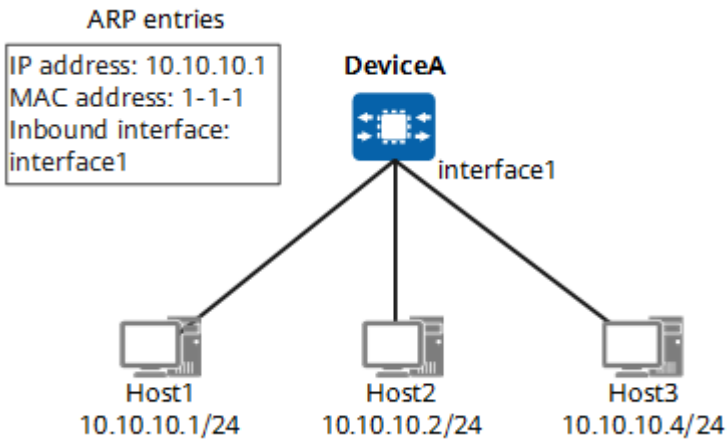
Application Scenarios

VLAN-based Layer 2 proxy ARP

As shown in **Figure 7-20**, VLAN-based Layer 2 proxy ARP is enabled on the Layer 2 device. When a user-side interface receives an ARP request message, the device searches for a matching ARP entry based on the destination IP address in the message.

- If a matching ARP entry exists, and the VLAN to which the interface in the entry belongs is the same as that of the interface that receives the ARP request message, the device directly sends an ARP reply message based on the entry.
- If a matching ARP entry exists, but the VLAN to which the interface in the entry belongs is different from that of the interface that receives the ARP request message, the device processes the ARP request message according to the original standard process.
- If no matching entry exits, the device processes the ARP request message according to the original standard process.

Figure 7-20 Network diagram of VLAN-based Layer 2 proxy ARP



7.3.8.2 Configuring Layer 2 Proxy ARP

Prerequisites

Before configuring Layer 2 proxy ARP, configure a VLAN and add a vlan interface.

Context

Layer 2 proxy ARP can effectively isolate ARP BDs and reduce the impact of ARP broadcast messages on the network. Layer 2 proxy ARP is configured based on a VLAN. If the VLAN filter function is disabled, proxy ARP is not performed.

Procedure

- Configure VLAN-based Layer 2 proxy ARP.
  a. Enter the edit-config view.
     edit-config
  b. Enter the VLAN view
     vlans vlan id *vlan-id*
  c. Enter the ARP security view.
     arp-security
  d. Enable Layer 2 proxy ARP.
     l2proxy-enable true
  e. (Optional) Disable Layer 2 proxy ARP.
     l2proxy-enable false
  f. Commit the configuration.
     commit

----End

Verifying the Configuration

Run the **display /vlan/vlans/vlan[id="***vlan-id***"]/arp-security** command to check whether Layer 2 proxy ARP based on a specified VLAN is enabled.

7.3.9 Configuring IP Address Conflict Detection

Prerequisites

Before configuring IP address conflict detection, you have completed the following tasks:

- Connect interfaces and configure physical parameters for the interfaces to ensure that the physical status of the interfaces is up.
- Configure link layer protocol parameters for the interfaces to ensure that the link layer protocol status of the interfaces is up.

Context

IP address conflicts cause problems such as route flapping and traffic interruptions, affecting user services. IP address conflicts are often caused by incorrect networking or configurations. Customers expect that devices can automatically detect IP address conflicts on a network and immediately notify them of conflict reasons, so they can rapidly eliminate the conflicts and minimize adverse impacts on services.

IP address conflict detection helps you quickly locate and modify conflicting IP addresses. Currently, the system configures and manages IP addresses automatically and does not support manual configuration or management of IP addresses.

7.3.10 Maintaining ARP

Context

For details about configuration parameters, see huawei-arp.yang.

NOTICE

- Static ARP entries cannot be restored after being cleared. Exercise caution when you clear static ARP entries.
- After ARP entries are cleared, the mapping between IP and MAC addresses is deleted, which may cause a failure to access some nodes. Exercise caution when you clear ARP entries.

Table 7-17 Clearing ARP information

Operation	Command
Clear all dynamic ARP entries.	1. Enter the ARP entry clearing view. **arp-entry-clear** 2. Configure the function of clearing all dynamic ARP entries. **clear-type dynamic-arp** 3. Perform the clear operation. **emit**
Clear static ARP entries on a specified interface.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Enter the IPv4 view. **ipv4** 4. Clear static ARP entries on the specified interface. **remove static-arps** 5. Commit the configuration. **commit**
Clear static ARP entries containing a specified IP address.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Enter the IPv4 static ARP view. **ipv4 static-arps** 4. Clear static ARP entries containing a specified IP address. **remove static-arp** [**ip-addr=***ip-address*] 5. Commit the configuration. **commit**
Clear dynamic ARP entries containing a specified IP address and interface name.	1. Enter the ARP entry clearing view. **arp-entry-clear** 2. Configure the function of clearing dynamic ARP entries containing a specified IP address and interface name. For a Layer 2 interface, the value of *interface-name* is the name of the Layer 2 interface or the name of the VLANIF interface to which the Layer 2 interface is added. For a Layer 3 interface, the value of *interface-name* is the name of the Layer 3 interface (this name is the same as the value of the if-name field in the ARP entry). **if-name** *interface-name* **ip-addr** *ip-address* 3. Perform the clear operation. **emit**

Table 7-18 Querying ARP information

Operation	Command
Query all ARP entries.	1. Enter the ARP entry query view. **arp query-entries** 2. Query all ARP entries. **display this**
Query ARP entries based on a specified IP address.	1. Enter the ARP entry query view. **arp query-entries** 2. Specify an IP address. **query-entry ni-name 0 ip-addr ip-address** *ip-address* 3. Query ARP entries. **display this**

7.4 ARP Security Configuration

7.4.1 Overview of ARP Security

Definition

Address Resolution Protocol (ARP) security protects devices against attacks that tamper with or forge ARP messages, improving device and communication security.

Purpose

If two hosts need to communicate, the sender must know the network-layer IP address of the receiver. IP datagrams, however, must be encapsulated with media access control (MAC) addresses before they can be transmitted over the physical network. Therefore, ARP is needed to map IP addresses to MAC addresses to ensure the transmission of datagrams. ARP is easy to use but lacks security protection mechanisms. Attackers may use ARP to attack network devices. The following ARP attacks exist on networks:

- ARP spoofing attack

  Attackers send bogus ARP messages to modify ARP entries on gateways or valid hosts, interrupting the transmission of valid ARP messages. ARP anti-spoofing protects the device against ARP attacks, improving communication security and reliability. **Table 1** describes the ARP anti-spoofing security solution.

Table 7-19 ARP anti-spoofing security solution

ARP Security Function	Description
ARP message validity check	After receiving an ARP message, a device checks whether the source and destination MAC addresses in the Ethernet header match those in the Data field of the message. If they match, the device considers the message valid and allows it to pass. If they do not match, the device considers the message an attack one and discards it. If no inconsistency is detected, the ARP message is accepted.
Fixed ARP	After a device learns the ARP entry of a user, it does not update the ARP entry or only modifies some fields in the ARP entry when receiving ARP messages from other users. This ensures that valid ARP entries are not replaced by attackers using forged ARP messages.
Dynamic ARP inspection	After dynamic ARP inspection (DAI) is enabled on a device, the device compares the source IP address, source MAC address, interface, and VLAN information in a received ARP message with DHCP snooping binding entries. If they match, the device considers the message valid and forwards it. If they do not match, the device considers the message invalid and discards it. This function applies only to DHCP snooping scenarios.
ARP gateway anti-collision	If an attacker forges the gateway IP address to send ARP messages to other user hosts on a LAN, the user hosts record incorrect gateway address mappings in their ARP tables. As a result, all traffic from the user hosts to the gateway is sent to the attacker and the attacker can intercept related data, causing network access failures of these user hosts. To defend against attacks from a bogus gateway, enable ARP gateway anti-collision on the gateway if user hosts are directly connected to the gateway.

ARP Security Function	Description
Strict ARP learning	A device learns the MAC addresses of only the ARP reply messages in response to the ARP request messages sent by itself. This prevents attacks initiated by ARP request messages and the ARP reply messages that are not in response to the request messages that the device itself sends.
Gratuitous ARP message discarding	A device discards all received gratuitous ARP messages to prevent an ARP entry overflow.

- ARP flood attack (denial of service)

  Attackers forge and send to a device excessive ARP request messages and gratuitous ARP messages with IP addresses that cannot be mapped to MAC addresses. As a result, the device's ARP buffer overflows, and the device is incapable of caching valid ARP entries. In this case, valid ARP messages cannot be transmitted. ARP anti-flood relieves CPU loads and prevents an ARP entry overflow, ensuring normal device running. **Table 2** describes the ARP anti-flood security solution.

  Table 7-20 ARP anti-flood security solution

ARP Security Function	Description
ARP entry limiting	A device limits the number of ARP entries that an interface can learn to prevent an ARP entry overflow, improving ARP entry security. Currently, the maximum number of ARP entries is 1024.
Rate limiting on ARP messages	A device counts the number of received ARP messages. If the number of ARP messages received in a specified period exceeds an upper limit, the device does not process the excess ARP messages. This function prevents an ARP entry overflow.
Rate limiting on ARP Miss messages	A device counts the number of received ARP Miss messages. If the number of ARP Miss messages received in a specified period exceeds the limit, the device does not process excess ARP Miss messages, relieving the burden on the CPU.

7.4.2 Configuration Precautions for ARP security

Licensing Requirements

ARP Security is not under license control.

Hardware Requirements

Table 7-21 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

7.4.3 Default Settings for ARP Security

Table 7-22 Default settings for ARP Security

Parameter	Default Setting
Maximum number of dynamic ARP entries that an interface can learn	1024
Disabling ARP learning on an interface	Disabled
ARP gateway anti-collision	Disabled
ARP message validity check	Disabled
Strict ARP learning	Disabled

7.4.4 Disabling ARP Learning on an Interface

Context

If a user host connected to a device's interface initiates an ARP attack, the device's ARP resources may be exhausted. To resolve this issue, you can disable the interface from learning dynamic ARP entries, ensuring device security. For details about configuration parameters, see huawei-arp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the view of an interface except Layer 2, null, and lo interfaces.

```
ifm interfaces interface name interface-name
```


Step 3 Enable or disable ARP learning on the interface.

```
arp-entry
arp-learn-disable { true | false }
```

Step 4 Commit the configuration.

```
commit
```

NOTICE

Disabling ARP learning on an interface may cause a traffic forwarding failure. Exercise caution when performing this operation.

----End

Verifying the Configuration

Run the **display ifm/interfaces/interface**[**name=***interface-name*]**/arp-entry** command to check the configuration of disabling ARP entry learning on an interface.

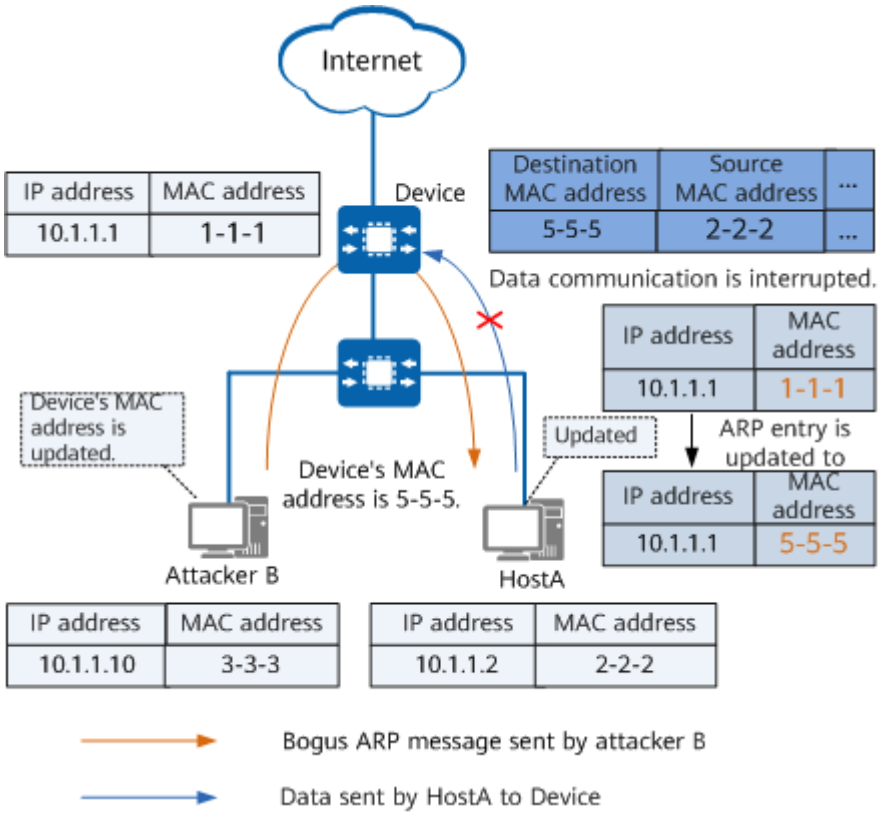
7.4.5 Configuring ARP Gateway Anti-Collision

7.4.5.1 Understanding ARP Gateway Anti-Collision

Fundamentals

As shown in **Figure 7-21**, attacker B forges the gateway (Device) address to send a bogus ARP message to HostA. HostA incorrectly considers the attacker Device and records an incorrect ARP entry for Device. As a result, Device cannot receive messages from HostA and their communication is interrupted.

Figure 7-21 ARP gateway collision



To prevent bogus gateway attacks, enable ARP gateway anti-collision on Device. In addition, you can enable gratuitous ARP message sending on Device so that Device can broadcast correct gratuitous ARP messages to all user hosts. In this manner, the gateway address mappings recorded by the attacked user hosts can be corrected.

7.4.5.2 Configuring ARP Gateway Anti-Collision

Context

If an attacker forges the gateway IP address to send ARP messages to other user hosts on a LAN, the user hosts record incorrect gateway address mappings in their ARP tables. As a result, all traffic from the user hosts to the gateway is sent to the attacker and the attacker can intercept related data, causing network access failures of these user hosts.

To defend against attacks from a bogus gateway, enable ARP gateway anti-collision on the gateway if user hosts are directly connected to the gateway.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the global ARP view.

```
arp global
```


Step 3 Enable ARP gateway anti-collision.

```
gateway-dup-enable { true | false }
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display /arp/global/gateway-dup-enable all** command to check the ARP gateway anti-collision configuration.

7.4.6 Configuring ARP Message Validity Check

7.4.6.1 Understanding ARP Message Validity Check

Definition

ARP is easy to implement but lacks security mechanisms, making it vulnerable to attacks. An attacker can forge an ARP message by changing the MAC addresses carried in the Data field of an authorized user's message, which makes the source and destination MAC addresses carried in the Data field different from those carried in the Ethernet header.

This issue can be resolved by configuring ARP message validity check on devices. When a device capable of checking ARP message validity receives an ARP message, the device checks whether the source and destination MAC addresses carried in the Data field match those carried in the Ethernet header. If they match, the device considers the message valid. If they do not match, the device considers the message invalid and discards it.

Related Concepts

Table 7-23 ARP message validity check modes

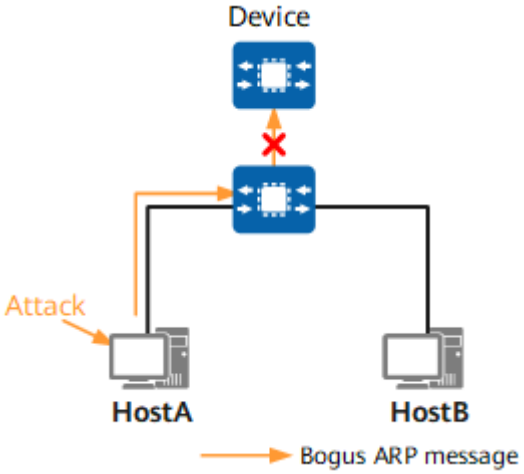
Check Mode	Description
Source MAC address-based check	After receiving an ARP message, a device checks whether the source MAC address in the Data field matches that in the Ethernet header. If they match, the device considers the message valid. If they do not match, the device considers the message invalid and discards it.

Check Mode	Description
Destination MAC address-based check (only for ARP reply messages)	After receiving an ARP message, a device checks whether the destination MAC address in the Data field matches that in the Ethernet header. If they match, the device considers the message valid. If they do not match, the device considers the message invalid and discards it.
Source and destination MAC addresses-based check	After receiving an ARP request message, a device checks whether the source MAC address in the Data field matches that in the Ethernet header. If they match, the device considers the message valid. If they do not match, the device considers the message invalid and discards it. After receiving an ARP reply message, a device checks whether the source and destination MAC addresses in the Data field match those in the Ethernet header. If the source and destination MAC addresses in the Data field match those in the Ethernet header, the device considers the message valid. Otherwise, the device considers the message invalid and discards it.

Fundamentals

Figure 7-22 shows how the validity of an ARP request message is checked.

Figure 7-22 Validity check of an ARP request message



As shown in **Figure 7-22**, HostA is attacked, and the attacker changes the source MAC address of an ARP request message sent by HostA. As a result, the source MAC address in the Data field of the message is different from that in the Ethernet header. If ARP message validity check is not configured, both HostB and Device learn the fake address information carried in HostA's ARP request message.

If ARP message validity check is configured on Device, Device checks whether the source MAC address in the Data field of the received ARP request message matches that in the Ethernet header. If they match, Device learns the address information carried in the message. If they do not match, Device discards the message.

7.4.6.2 Configuring ARP Message Validity Check

Context

You can configure ARP message validity check to reject ARP messages from authorized users who have been attacked, improving communication security and reliability. For details about configuration parameters, see huawei-arp.yang.

Procedure

- Configure validity check for source MAC addresses of ARP messages on an interface.
  a. Enter the edit-config view.
     edit-config
  b. Enter the view of an interface except Layer 2, null, and lo interfaces.
     ifm interfaces interface name *interface-name*
  c. Enable or disable validity check for source MAC addresses of ARP messages.
     arp-entry
     src-mac-check { **true** | **false** }
  d. Commit the configuration.
     commit
- Configure validity check for destination MAC addresses of ARP messages on an interface.
  a. Enter the edit-config view.
     edit-config
  b. Enter the view of an interface except Layer 2, null, and lo interfaces.
     ifm interfaces interface name *interface-name*
  c. Enable or disable validity check for destination MAC addresses of ARP messages.
     arp-entry
     dest-mac-check { **true** | **false** }
  d. Commit the configuration.
     commit

----End

Verifying the Configuration

Run the **display ifm/interfaces/interface**[**name=***interface-name*]**/arp-entry** command to check the configuration of ARP message validity check on the device.

7.4.7 Configuring Strict ARP Learning

7.4.7.1 Understanding Strict ARP Learning

Fundamentals

If strict ARP learning is not configured, a device processes ARP entries as follows:

- After receiving an ARP reply message in response to the ARP request message that the device itself sends, the device checks whether the source IP address in the message matches an ARP entry. If no matching entry exists, the device creates an ARP entry using the source IP and MAC addresses carried in the message. If a matching entry exists, the device updates the entry based on the source IP and MAC addresses carried in the message.
- After receiving an ARP request message, the device sends an ARP reply message and then creates an ARP entry.

If strict ARP learning is configured, a device processes ARP messages as follows:

- After receiving an ARP reply message, the device checks whether the message is in response to an ARP request message sent by itself. If so, the device learns and updates ARP entries. If not, the device does not learn or update ARP entries.
- After receiving an ARP request message, the device sends an ARP reply message but does not learn or update ARP entries.

Application Scenarios

If many user hosts simultaneously send a large number of ARP messages to a device, or attackers send bogus ARP messages to the device, the following issues occur:

- Processing ARP messages consumes many CPU resources. The device learns many invalid ARP entries, which exhausts ARP entry resources and prevents the device from learning ARP entries for ARP messages from authorized users. Consequently, communication of authorized users is interrupted.
- After receiving bogus ARP messages, the device incorrectly updates the ARP entries, resulting in user communicate failures.

To avoid the preceding issues, configure strict ARP learning on the device. After strict ARP learning is configured, the device learns only ARP entries for ARP reply messages in response to ARP request messages sent by itself. In this way, the device can defend against most ARP attacks.

7.4.7.2 Configuring Strict ARP Learning

Context

Strict ARP learning can be enabled globally or for an interface (interface-based learning is not currently supported). This function allows the device or interface to learn information carried in the ARP reply messages it receives in response to the ARP request messages it sends. For details about configuration parameters, see huawei-arp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the global ARP view.

```
arp global
```

Step 3 Enable or disable strict ARP learning.

```
strict-learn-enable { true | false }
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display arp/global/strict-learn-enable** command to check whether strict ARP learning is enabled on the device.

7.5 DHCPv4 Configuration

7.5.1 Overview of DHCPv4

Definition

The Dynamic Host Configuration Protocol (DHCP) dynamically configures and uniformly manages IPv4 addresses of hosts. To distinguish from the Dynamic Host Configuration Protocol for IPv6 (DHCPv6), use DHCPv4 in the following sections.

DHCPv4 is defined in RFC 2131 and uses the client/server communication model. A DHCPv4 client requests configuration information from a DHCPv4 server, and the server returns the configuration information allocated to the client.

DHCPv4 supports dynamic and static IPv4 address allocation. Network administrators can use either of the two mechanisms to allocate IPv4 addresses to hosts based on network requirements.

- Dynamic allocation: DHCPv4 allocates an IPv4 address with a limited validity period (known as a lease) to a client.

This mechanism applies to scenarios where hosts temporarily access the network and the number of idle IPv4 addresses is less than the total number of hosts.

- Static allocation: Network administrators use DHCPv4 to allocate fixed IPv4 addresses to specified hosts.

  Compared with manual IPv4 address configuration, DHCPv4 static allocation prevents manual configuration errors and helps network administrators perform unified maintenance and management.

Purpose

As networks expand and become more complex, network configurations also become more complex. In addition, a sharp increase in computers and their location changes cause IPv4 addresses to frequently change and become insufficient. To properly and dynamically allocate IPv4 addresses to hosts, DHCPv4 is used.

DHCPv4 is developed based on the Bootstrap Protocol (BOOTP) which runs in a static environment where each host has a fixed network connection. For each host using BOOTP, an administrator must configure a specific BOOTP parameter file that keeps unchanged in a long period. DHCPv4 is an extension of BOOTP by:

- Dynamically allocating an IPv4 address to each host, instead of specifying an IPv4 address for each host.
- Allocating other configuration parameters, such as the boot file of a client, so that the client can obtain all the required configuration information by using only one message.

DHCPv4 properly and dynamically allocates IPv4 addresses, which improves IPv4 address utilization and prevents the waste of IPv4 addresses. It simplifies network deployment and scale-out, even for small networks.

Benefits

DHCPv4 offers the following benefits:

- Reduced client configuration and maintenance costs

  DHCPv4 is easy to configure and deploy. For non-technical users, DHCPv4 minimizes configuration-related operations on clients and reduces remote deployment and maintenance costs.

- Centralized management

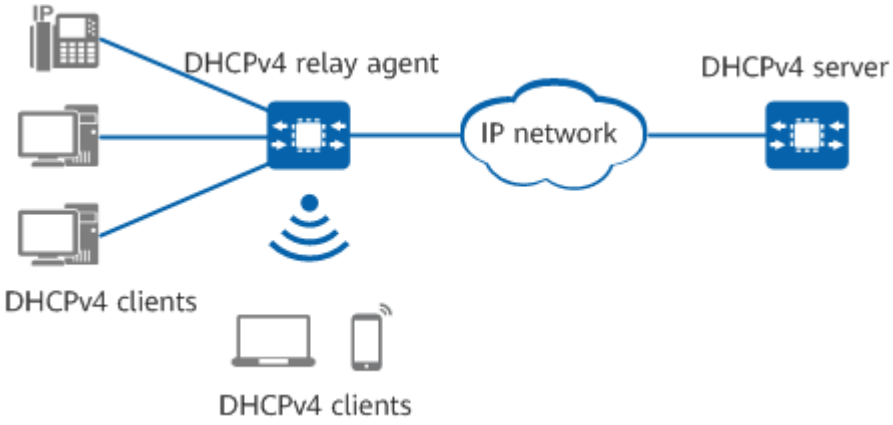
  A DHCPv4 server can manage the configurations of multiple network segments. When the configuration of a network segment changes, an administrator only needs to update the corresponding configuration on the DHCPv4 server.

7.5.2 Understanding DHCPv4

7.5.2.1 Typical DHCPv4 Networking

Figure 7-23 shows the typical DHCPv4 networking.

Figure 7-23 Typical DHCPv4 networking



A typical DHCPv4 network consists of the following roles:

- **DHCPv4 server**

  A DHCPv4 server selects IPv4 addresses from an address pool and allocates them to DHCPv4 clients. It can also provide other network parameters, such as the default gateway address, DNS server address, and Windows Internet Name Service (WINS) server address, for DHCPv4 clients. A DHCPv4 server can receive and process intra- or inter-network segment DHCPv4 request messages forwarded by a DHCPv4 relay agent.
- **DHCPv4 client**

  A DHCPv4 client sends DHCPv4 request messages to obtain network parameters, such as IPv4 addresses, through BOOTP or DHCPv4. A DHCPv4 client can be an IP phone, PC, mobile phone, or diskless workstation.
- **DHCPv4 relay agent**

  A DHCPv4 relay agent forwards DHCPv4 messages between a DHCPv4 server and client and helps the DHCPv4 server dynamically allocate network parameters to the DHCPv4 client.

  When a DHCPv4 client broadcasts DHCPv4 request messages with the destination IPv4 address 255.255.255.255, only the DHCPv4 server on the same network segment as the DHCPv4 client can receive the messages. If the DHCPv4 client and server are located on different network segments, the DHCPv4 server cannot receive DHCPv4 request messages from the client. In this case, a DHCPv4 relay agent is required to forward DHCPv4 messages. Different from traditional IP packet forwarding, the DHCPv4 relay agent changes the format of a received DHCPv4 request or reply message, generates a new DHCPv4 message, and forwards the message.

  NOTE

  The device can function only as a DHCPv4 server or client.

7.5.2.2 Introduction to DHCPv4 Messages

DHCPv4 Message Types

A DHCPv4 server and client communicate through DHCPv4 messages, which are transmitted using the User Datagram Protocol (UDP). A DHCPv4 client sends

messages to a DHCPv4 server through UDP port 68, and a DHCPv4 server sends messages to a DHCPv4 client through UDP port 67. The following table describes DHCPv4 message types.

Table 7-24 DHCPv4 message types

Message Name	Description
DHCPDISCOVER	A DHCPDISCOVER message is broadcast by a DHCPv4 client to locate a DHCPv4 server when the client attempts to connect to a network for the first time.
DHCPOFFER	A DHCPOFFER message is sent by a DHCPv4 server to respond to a DHCPDISCOVER message.
DHCPREQUEST	A DHCPREQUEST message is sent in the following scenarios: • After a DHCPv4 client starts, it broadcasts a DHCPREQUEST message to respond to a DHCPOFFER message sent by a DHCPv4 server. • After a DHCPv4 client restarts, it broadcasts a DHCPREQUEST message to confirm the configuration including the previously allocated IPv4 address. • After a DHCPv4 client obtains an IPv4 address, it unicasts or broadcasts a DHCPREQUEST message to renew the IPv4 address lease.
DHCPACK	A DHCPACK message is sent by a DHCPv4 server to acknowledge a DHCPREQUEST message from a DHCPv4 client. After receiving a DHCPACK message, the DHCPv4 client obtains the configuration parameters including the IPv4 address.
DHCPNAK	A DHCPNAK message is sent by a DHCPv4 server to reject a DHCPREQUEST message from a DHCPv4 client. For example, if a DHCPv4 server cannot find matching lease records after receiving a DHCPREQUEST message, it sends a DHCPNAK message to notify the DHCPv4 client that no IPv4 address is available.
DHCPDECLINE	A DHCPDECLINE message is sent by a DHCPv4 client to notify a DHCPv4 server that the allocated IPv4 address conflicts with another IPv4 address. The DHCPv4 client then applies to the DHCPv4 server for another IPv4 address.
DHCPRELEASE	A DHCPRELEASE message is sent by a DHCPv4 client to release its IPv4 address. After receiving a DHCPRELEASE message, a DHCPv4 server can allocate this IPv4 address to another DHCPv4 client.
DHCPINFORM	A DHCPINFORM message is sent by a DHCPv4 client to obtain other network configuration parameters such as the gateway address and DNS server address after the client has obtained an IPv4 address.

DHCPv4 Message Format

The DHCPv4 message format is developed based on the BOOTP message format. Therefore, a DHCPv4 server can also interact with a BOOTP client.

Figure 7-24 shows the DHCPv4 message format. The number in the brackets indicates the field length, in octets. **Table 7-25** describes each field in a DHCPv4 message.

Figure 7-24 DHCPv4 message format

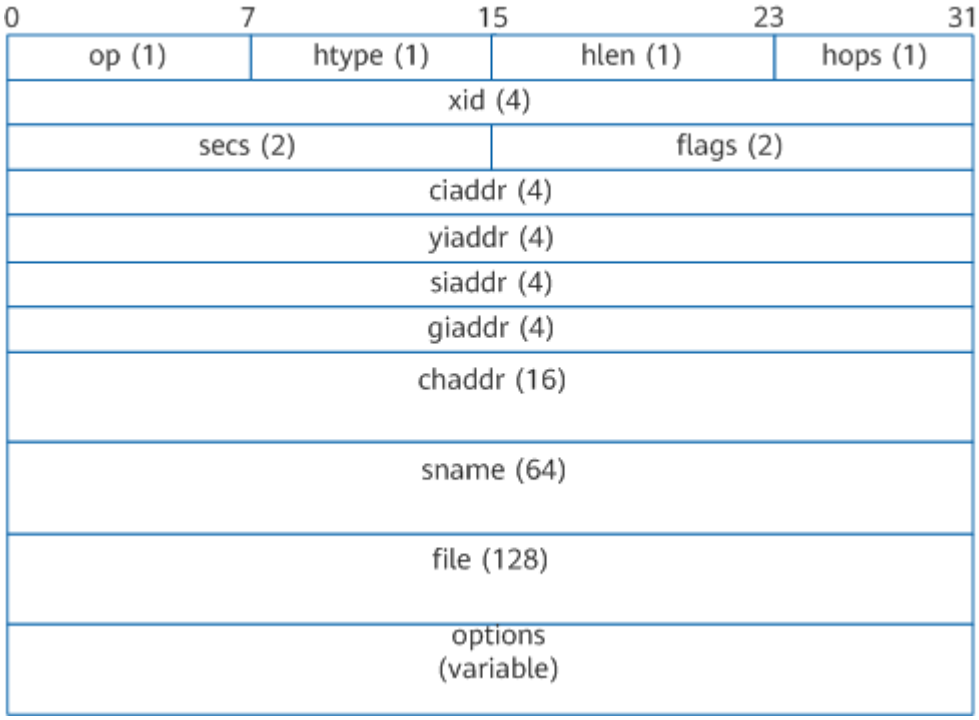


Table 7-25 Description of each field in a DHCPv4 message

Field	Length	Description
op	1 octet	Message type: • 1: BOOTREQUEST • 2: BOOTREPLY
htype	1 octet	Hardware type. The most common value is 1, indicating an Ethernet.
hlen	1 octet	Hardware address length. For an Ethernet address, the value of this field is 6.

Field	Length	Description
hops	1 octet	Number of DHCPv4 relay agents that have relayed a DHCPv4 message. This field is set to 0 by a DHCPv4 client or server. Its value increases by 1 each time the message passes through a DHCPv4 relay agent. This field limits the number of DHCPv4 relay agents that a DHCPv4 message can pass through. A maximum of 16 DHCPv4 relay agents are allowed between a DHCPv4 server and client. If the value of hops is greater than 16, DHCPv4 messages are discarded.
xid	4 octets	Random number chosen by a DHCPv4 client, used by the client and server to associate messages and responses between them.
secs	2 octets	Seconds elapsed since a DHCPv4 client obtained or renewed an IPv4 address.
flags	2 octets	Flags field. Only the leftmost bit in this field is valid and other bits are set to 0. The leftmost bit determines whether a DHCPv4 server unicasts or broadcasts a reply message. Options for this field are as follows: ● 0: The DHCPv4 server unicasts a reply message. ● 1: The DHCPv4 server broadcasts a reply message.
ciaddr	4 octets	Client IPv4 address. The IPv4 address can be an existing IPv4 address of a DHCPv4 client or an IPv4 address allocated by a DHCPv4 server to a DHCPv4 client. During initialization, the client has no IPv4 address, and the value of this field is 0.0.0.0. The IPv4 address 0.0.0.0 is only used by a DHCPv4-enabled device to temporarily communicate with other devices during startup. It is an invalid destination address.
yiaddr	4 octets	IPv4 address that a DHCPv4 server assigns to a DHCPv4 client. The DHCPv4 server fills this field into a DHCPv4 reply message.
siaddr	4 octets	IPv4 address of a server from which a DHCPv4 client obtains the boot file.

Field	Length	Description
giaddr	4 octets	IPv4 address of the first DHCPv4 relay agent. If a DHCPv4 server and client are located on different network segments, the first DHCPv4 relay agent fills its own IPv4 address into this field of a DHCPv4 request message and forwards the message to the DHCPv4 server. The DHCPv4 server determines the network segment where the client resides based on the giaddr field, and allocates an IPv4 address on this network segment to the client. The DHCPv4 server also returns a reply message to the first DHCPv4 relay agent based on the giaddr field. The DHCPv4 relay agent then forwards the message to the client. If the DHCPv4 request message passes through multiple DHCPv4 relay agents before reaching the DHCPv4 server, the value of this field is the IPv4 address of the first DHCPv4 relay agent and remains unchanged. However, the value of the hops field increases by 1 each time the DHCPv4 request message passes through a DHCPv4 relay agent.
chaddr	16 octets	Client MAC address. This field must be consistent with the htype and hlen fields. When sending a DHCPv4 request message, the client fills its hardware address in this field. For an Ethernet, a 6-octet Ethernet MAC address must be filled in this field when the htype and hlen fields are set to 1 and 6, respectively.
sname	64 octets	Name of the server from which a client obtains the configuration. This field is optional and is filled in by a DHCPv4 server. This field must be filled in with a character string that ends with 0.
file	128 octets	Boot file name specified by the DHCPv4 server for a DHCPv4 client. The DHCPv4 server fills this field and delivers it together with an IPv4 address to the client. This field is optional and must be filled in with a character string that ends with 0.
options	Variable	DHCPv4 options field, which has a maximum of 312 octets. This field contains the DHCPv4 message type and configuration parameters allocated by a DHCPv4 server to a client. The configuration parameters include the gateway IPv4 address, DNS server IPv4 address, and IPv4 address lease. For details about the options field, see **Options Field in a DHCPv4 Message**.

Options Field in a DHCPv4 Message

The options field is located at the end of a DHCPv4 message and is used to store the control information and parameters allocated to a DHCPv4 client. As shown in **Figure 7-25**, the options field consists of three parts: Type, Length, and Value. **Table 7-26** describes the three parts.

Figure 7-25 Format of the options field

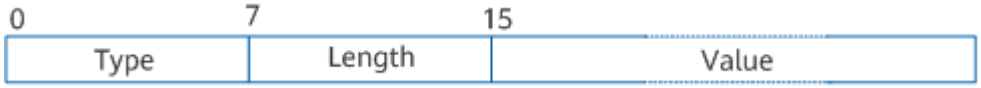


Table 7-26 Description of the options field

Option	Length	Description
Type	1 octet	Information type
Length	1 octet	Length of the information content
Value	Depending on the Length option	Information content

The value of the DHCPv4 options field ranges from 1 to 255. DHCPv4 options include predefined and user-defined options. **Table 7-27** describes some predefined DHCPv4 options.

Table 7-27 Description of DHCPv4 options

Option No.	Description
1	Subnet mask.
3	Gateway address.
4	Time server address.
6	DNS server address.
7	Log server address.
12	DHCPv4 client's hostname.
15	Domain name suffix.
17	Root path.
28	Multicast address.

Option No.	Description
33	Static route. After a DHCPv4 client receives DHCPv4 messages with this option, it adds the classful static routes contained in the option to its routing table. In classful routes, destination address masks are natural masks and cannot be used for subnetting. If Option 121 exists, Option 33 is ignored.
42	NTP server address.
43	Vendor-specific information.
44	NetBIOS name server.
46	NetBIOS node type.
50	Requested IPv4 address.
51	IPv4 address lease.
52	Additional option.
53	DHCPv4 message type.
54	Server identification.
55	Parameter request list. A DHCPv4 client uses this option to request specified configuration parameters from a DHCPv4 server. The content of this option is the option values corresponding to the parameters requested by the client.
56	Message option, which is used to describe the reason why an IP address fails to be allocated. DHCPv4 messages encapsulated with this option are as follows: ● DHCPNAK message sent by a DHCPv4 server. ● DHCPDECLINE or DHCPRELEASE message sent by a DHCPv4 client. ● DHCPDECLINE message sent by a DHCPv4 relay agent when an IPv4 address conflict is detected or DHCPRELEASE message sent by a DHCPv4 client to release its IPv4 address. ● DHCPRELEASE message sent by a DHCP snooping device to release its IPv4 address.
58	Lease renewal time (T1), which is 50% of the lease.
59	Lease renewal time (T2), which is 87.5% of the lease.
60	Vendor category, which identifies the DHCPv4 client type and configuration.
61	Client identifier.
66	TFTP server name allocated to DHCPv4 clients.
67	Boot file name allocated to a DHCPv4 client.
77	User type.

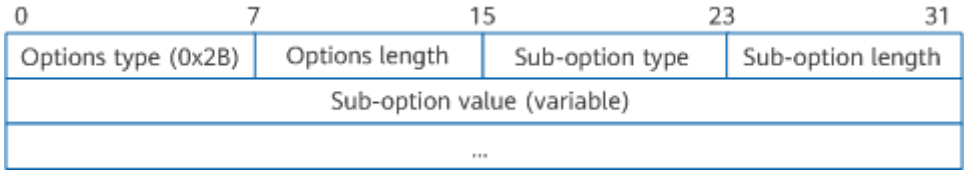
Option No.	Description
120	SIP server IPv4 address. **NOTE** Currently, only IPv4 addresses can be parsed and domain names cannot be parsed.
121	Classless route option. This option contains a group of classless static routes. After a DHCPv4 client receives DHCPv4 messages with this option, it adds the classless static routes contained in the option to its routing table. In classless routes, destination address masks can be any value and can be used for subnetting. **NOTE** A device functioning as a DHCPv4 client can receive static routes delivered from a DHCPv4 server through Option 121.
129	Call server address.
143	BootStrap server address list.
184	Reserved option. You can customize information carried in this option.

In addition to predefined options, a device supports user-defined options to connect to different terminals, such as IP phones.

- Vendor-specific information option (Option 43)

  Figure 7-26 shows the format of Option 43.

  Figure 7-26 Format of Option 43

  DHCPv4 servers and clients use Option 43 to exchange vendor-specific information. When a DHCPv4 server receives a DHCPv4 request message with parameter 43 encapsulated in Option 55, it encapsulates Option 43 in a reply message and sends the message to the DHCPv4 client.

  When a device functions as the DHCPv4 server, it can deliver the AC's IPv4 address to connected APs (Huawei devices), facilitating the connection setup between the AC and APs.

  Option 43 supports suboptions, as shown in **Figure 7-26**.

  - Sub-option type: type of the suboption. When the device delivers the AC's IPv4 address to APs, the value can be 0x01 (hexadecimal type), 0x02 (IPv4 address type), or 0x03 (ASCII code type).
  - Sub-option length: length of the suboption.

  - Sub-option value: value of the suboption.
- Relay agent information option (Option 82)

  Option 82 records the location of a DHCPv4 client. A DHCPv4 relay agent or DHCP snooping device appends Option 82 to a DHCPv4 request message sent from a DHCPv4 client and forwards the message to a DHCPv4 server.

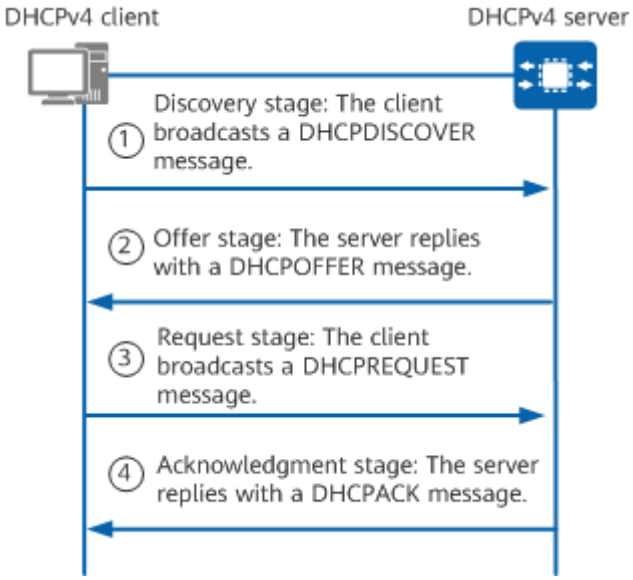
  An administrator can use Option 82 to locate a DHCPv4 client and control the security and accounting of the DHCPv4 client. A DHCPv4 server that supports Option 82 can determine policies to flexibly allocate IPv4 addresses and other parameters based on the information in this option.

  Option 82 contains a maximum of 254 suboptions. If Option 82 is defined, at least one suboption must be defined.

7.5.2.3 DHCPv4 Server Allocating Network Parameters to Newly Connected DHCPv4 Clients

As shown in **Figure 7-27**, when no DHCPv4 relay agent is deployed, the newly connected DHCPv4 client and server exchange DHCPv4 messages through four steps.

Figure 7-27 Message exchange between the newly connected DHCPv4 client and server



Step 1: Discovery Stage

The newly connected DHCPv4 client does not know the IPv4 address of the DHCPv4 server. To learn the IPv4 address of the DHCPv4 server, the DHCPv4 client broadcasts a DHCPDISCOVER message with the destination IPv4 address of 255.255.255.255 to all devices on the same network segment, including the DHCPv4 server and relay agent (if possible). The DHCPDISCOVER message carries information, such as the client's MAC address (**chaddr field**), requested parameter list option (**Option 55**), and broadcast flag (**flags field**).

Step 2: Offer Stage

All the DHCPv4 servers on the same network segment as the DHCPv4 client receive the DHCPDISCOVER message. Each DHCPv4 server selects an address pool on the same network segment as the IPv4 address of the interface receiving the DHCPDISCOVER message, and from the address pool allocates an idle IPv4 address. The DHCPv4 server then sends a DHCPOFFER message carrying the allocated IPv4 address to the DHCPv4 client.

In most cases, the leases of IPv4 addresses are specified in an address pool. If the DHCPDISCOVER message carries an expected lease, the DHCPv4 server compares the expected lease with the specified lease and allocates the IPv4 address with a smaller lease to the DHCPv4 client.

The DHCPv4 server allocates an IPv4 address from the address pool to a client in the following sequence:

NOTE

The IPv4 address allocation sequence cannot be modified.

1. IPv4 address statically bound to the MAC address of the client on the DHCPv4 server.
2. IPv4 address specified in **Option 50** (requested IPv4 address) in the DHCPDISCOVER message.
3. IPv4 address in the Expired state in the address pool, that is, the IPv4 address that has been assigned to the client and whose lease has expired.
4. Random IPv4 address in the Idle state in the address pool.
5. If no IPv4 address is available for allocation, the DHCPv4 server automatically reclaims the expired and conflicting IPv4 addresses in sequence. If an available IPv4 address is found after the reclaim, the DHCPv4 server allocates the IPv4 address. If no IPv4 address is available, the DHCPv4 client resends a DHCPDISCOVER message to apply for an IPv4 address after waiting for a response times out.

DHCPv4 servers can exclude some IPv4 addresses that cannot be allocated through DHCPv4 from address pools. For example, if 192.168.1.100/24 has been manually configured for a DNS server, the DHCPv4 server excludes this IPv4 address from the address pool on network segment 192.168.1.0/24 so that it is not allocated through DHCPv4. This helps prevent IPv4 address conflicts.

To prevent the allocated IPv4 address from conflicting with the IPv4 addresses of other clients on the network, before sending a DHCPOFFER message, the DHCPv4 server sends an ICMP Echo request message with the source address being the IPv4 address of the DHCPv4 server and the destination address being the pre-allocated IPv4 address to detect conflicts for the allocated IPv4 address. If the DHCPv4 server receives no ICMP Echo reply message within the detection period, no client is using this IPv4 address, and the DHCPv4 server can allocate it. If the DHCPv4 server receives an ICMP Echo reply message within the detection period, this IPv4 address is being used by another client, and the DHCPv4 server lists this IPv4 address as a conflicting one. The DHCPv4 server then waits for the next DHCPDISCOVER message to start the IPv4 address selection process again.

The IPv4 address allocated in this stage may not be the final IPv4 address used by the client. This is because the IPv4 address can be allocated to another client if the

DHCPv4 server receives no response 16 seconds after the DHCPOFFER message is sent. The IPv4 address for the client can be determined only after the request and acknowledgment stages.

Step 3: Request Stage

If multiple DHCPv4 servers reply with a DHCPOFFER message to the DHCPv4 client, the client accepts only the first received DHCPOFFER message. The client then broadcasts a DHCPREQUEST message carrying the selected DHCPv4 server identifier (**Option 54**) and IPv4 address (**Option 50**, with the IPv4 address specified in the yiaddr field of the accepted DHCPOFFER message).

The DHCPv4 client broadcasts a DHCPREQUEST message to notify all the DHCPv4 servers that it has selected the IPv4 address offered by a DHCPv4 server. Then the other servers can allocate IPv4 addresses to other clients.

NOTE

If a device functions as a DHCPv4 client and multiple DHCPv4 servers exist on the network, the DHCPv4 client polls the DHCPv4 servers according to the sequence of receiving DHCPOFFER messages. If a DHCPv4 server fails to assign an IPv4 address, the client selects the next DHCPv4 server.

Step 4: Acknowledgement Stage

After receiving the DHCPREQUEST message, the DHCPv4 server sends a DHCPACK message to the client, carrying the IPv4 address specified in **Option 50** of the DHCPREQUEST message.

After receiving the DHCPACK message, the DHCPv4 client broadcasts gratuitous ARP packets to check whether any other terminal is using the IPv4 address allocated by the DHCPv4 server. If no response is received within the specified time, the DHCPv4 client can use the IPv4 address. If the DHCPv4 client receives a response within the specified time, this IPv4 address is being used by another terminal. The client then sends a DHCPDECLINE message to the DHCPv4 server and applies for a new IPv4 address. The DHCPv4 server lists this IPv4 address as a conflicting one. The DHCPv4 server allocates conflicting IPv4 addresses only when there is no idle IPv4 address in the address pool, minimizing IPv4 address conflicts.

Occasionally, the DHCPv4 server may fail to allocate the IPv4 address specified in **Option 50** because, for example, an error occurs during negotiation or it takes a long time to receive the DHCPREQUEST message. In this case, the DHCPv4 server replies with a DHCPNAK message to notify the DHCPv4 client that the requested IPv4 address cannot be allocated. In this case, the DHCPv4 client has to send another DHCPDISCOVER message for a new application.

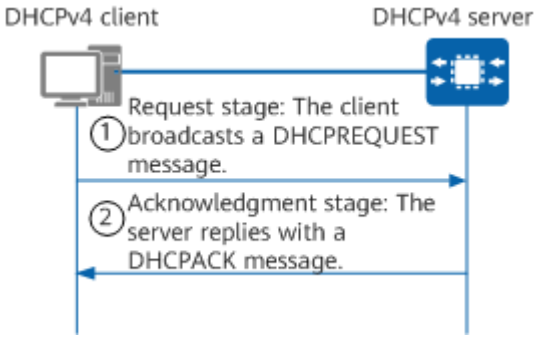
7.5.2.4 DHCPv4 Client Reusing an IPv4 Address

A non-newly connected DHCPv4 client can reuse an IPv4 address that has been allocated to it. As shown in **Figure 7-28**, the DHCPv4 client exchanges DHCPv4 messages with the DHCPv4 server to re-obtain network parameters such as the previously used IPv4 address. This process is performed through two steps.

NOTE

Not all clients can reuse IPv4 addresses that have been allocated to them. The following figure uses a PC as the DHCPv4 client to describe how the client reuses an IPv4 address.

Figure 7-28 Message exchange for IPv4 address reuse between a DHCPv4 client and server



Step 1: Request Stage

The DHCPv4 client broadcasts a DHCPREQUEST message carrying the IPv4 address that the client has used. The requested IPv4 address is added in Option 50.

Step 2: Acknowledgement Stage

After receiving the DHCPREQUEST message, the DHCPv4 server checks whether there is a lease record based on the MAC address in the message. If there is a lease record matching the MAC address, the DHCPv4 server replies with a DHCPACK message to notify the DHCPv4 client that the requested IPv4 address can be used. Otherwise, the DHCPv4 server performs no operation and waits for a new DHCPDISCOVER message from the client.

7.5.2.5 DHCPv4 Client Renewing Its IPv4 Address Lease

A DHCPv4 server defines a validity period for each IPv4 address assigned to a DHCPv4 client. The validity period is called a lease. All IPv4 addresses dynamically allocated by the DHCPv4 server are limited by the lease. The leases configured for different DHCPv4 servers can be different. Statically allocated IPv4 addresses are not limited by the lease, that is, the lease is infinite.

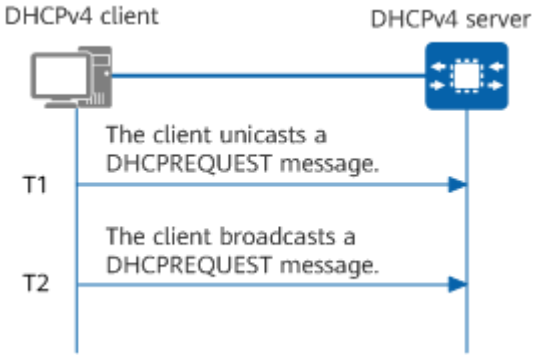
A DHCPDISCOVER message from a DHCPv4 client can carry an expected lease. When allocating network parameters, the DHCPv4 server compares the expected lease with the specified lease in the address pool and allocates an IPv4 address with a smaller lease to the DHCPv4 client. If the DHCPv4 client still needs to use the IPv4 address before the lease expires, it can request to extend the lease. If the DHCPv4 client does not need to use the IPv4 address, it can proactively release the IPv4 address. After the lease expires or the client goes offline, the server reclaims the IPv4 address. If no idle IPv4 address is available, the DHCPv4 server allocates the IPv4 address released by the client to another client, improving IPv4 address utilization.

If the DHCPv4 client does not renew the lease of an IPv4 address, the server reclaims the IPv4 address and allocates it to another client after the lease expires.

To continue to use the IPv4 address, the client must renew the IPv4 address lease before the lease expires (for example, extending the IPv4 address lease).

Figure 7-29 shows how a DHCPv4 client renews its IPv4 address lease.

Figure 7-29 IPv4 address lease renewal by a DHCPv4 client



1. When the lease reaches 50% (T1) of its validity period, the DHCPv4 client unicasts a DHCPREQUEST message to the DHCPv4 server to request lease renewal. If the DHCPv4 client receives a DHCPACK message, the IPv4 address lease is successfully renewed (counted from 0). If the DHCPv4 client receives a DHCPNAK message, the DHCPv4 client must send a DHCPDISCOVER message to apply for a new IPv4 address.
2. If no response is received from the DHCPv4 server when the lease reaches 87.5% (T2) of its validity period, the DHCPv4 client broadcasts a DHCPREQUEST message to request lease renewal. If the DHCPv4 client receives a DHCPACK message, the IPv4 address lease is successfully renewed (counted from 0). If the DHCPv4 client receives a DHCPNAK message, the DHCPv4 client must send a DHCPDISCOVER message to apply for a new IPv4 address.
3. If no response is received when the lease expires, the DHCPv4 client stops using the current IPv4 address and sends a DHCPDISCOVER message to apply for a new one.

If a DHCPv4 client does not need to use the allocated IPv4 address before the lease expires, it sends a DHCPRELEASE message to the DHCPv4 server to request IPv4 address release. The DHCPv4 server saves the configuration of this DHCPv4 client and records the IPv4 address in the allocated IPv4 address list. The IPv4 address can then be allocated to this DHCPv4 client or other clients. A DHCPv4 client can send a DHCPINFORM message to the DHCPv4 server to request configuration update.

7.5.3 Configuration Precautions for DHCPv4

Licensing Requirements

DHCPv4 is not under license control.

Hardware Requirements

Table 7-28 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

7.5.4 Default Settings for DHCPv4

Table 7-29 describes the default settings for DHCPv4.

Table 7-29 Default settings for DHCPv4

Function	Parameter	Default Setting
DHCPv4 server	DHCPv4 function	Enabled
	IPv4 addresses that cannot be automatically allocated to clients from the IPv4 address pool	Not configured
	Fixed IPv4 addresses allocated to specified clients	Not configured
	IPv4 address lease	1 day
	Number of conflict detections during IPv4 address allocation and maximum waiting time for each conflict detection	2 and 500 ms
DHCPv4 client	DHCPv4 client function on a WAN-side interface	Enabled
	DHCPv4 client function on a LAN-side interface	Disabled

NOTE

Currently, the default configuration cannot be queried using commands.

7.5.5 DHCPv4 Data Planning Guidance

Server Planning

A DHCPv4 client broadcasts DHCPv4 request messages. If multiple DHCPv4 servers (or DHCPv4 relay agents) are deployed on the same network segment as the client, the client accepts only the first received DHCPOFFER message and therefore may obtain an IPv4 address from an unexpected DHCPv4 server. Proper server planning can ensure that a client applies for network parameters from an expected DHCPv4 server.

When planning DHCPv4 servers, plan VLANs properly to ensure that only one DHCPv4 server (or DHCPv4 relay agent) can receive DHCPv4 request messages from clients in a VLAN.

IPv4 Address Planning

Plan the range of IPv4 addresses that can be automatically allocated by a DHCPv4 server and the IPv4 address allocation mechanism (dynamic or static allocation).

Plan IPv4 addresses that cannot be automatically allocated. For example, an enterprise requires that the device function as a DHCPv4 server, network segment 192.168.1.0/24 be used as the IPv4 addresses of employees' office computers, and 192.168.1.10 be used as the IPv4 address of the DNS server. During the configuration, 192.168.1.10 needs to be excluded from DHCPv4 automatic allocation.

Lease Planning

Plan an IPv4 address lease for a client based on the online duration of the client. The default IPv4 address lease is one day.

- In locations where clients often move and stay online for a short period of time, for example, in cafes, Internet bars, and airports, plan a short lease to ensure that IPv4 addresses are released promptly after the clients go offline.
- In locations where clients seldom move and stay online for a long period of time, for example, in office areas of an enterprise, plan a long lease to prevent system resources from being occupied by frequent lease or address renewals.

Planning of Other Network Parameters

In addition to allocating IPv4 addresses to clients, a DHCPv4 server allocates other network parameters to them. You can configure a DHCPv4 server to allocate other network parameters as required. For example, to enable a client to communicate with other network devices through a domain name and to obtain DNS parameters using DHCPv4, plan the IPv4 address of the DNS server and the domain name of the client.

7.5.6 Configuring a Device to Function as a DHCPv4 Server

7.5.6.1 Configuring DHCPv4

Context

Before configuring the DHCPv4 server function, you must enable DHCPv4.

For details about configuration parameters, see huawei-dhcp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the configuration path.

```
dhcp common global
```

Step 3 Enable DHCPv4.

```
enable true
```

By default, DHCPv4 is enabled.

Step 4 Commit the configuration.

```
commit
```

----End

7.5.6.2 Configuring an Interface Address Pool

Context

An address pool is a collection of IPv4 addresses that a DHCPv4 server can assign to clients. In addition to IPv4 addresses, network parameters such as the lease and default gateway can be configured in the address pool. When the DHCPv4 server assigns IPv4 addresses to clients, these network parameters are also assigned to them.

An interface address pool is created by configuring IPv4 addresses on an interface connecting a DHCPv4 server to a DHCPv4 client. IPv4 addresses in the interface address pool are on the same network segment as the interface's address, and can be allocated only to clients connected to the interface. An interface address pool applies only to the scenario where a DHCPv4 server and client are on the same network segment.

A DHCPv4 server selects an address pool based on whether a DHCPv4 relay agent is deployed. If no DHCPv4 relay agent is deployed, the DHCPv4 server selects an address pool that is on the same network segment as the IPv4 address of the interface that receives DHCPv4 request messages. If a DHCPv4 relay agent is deployed, the DHCPv4 server selects an address pool that is on the same network segment as the IPv4 address specified in the giaddr field of received DHCPv4 request messages.

You need to determine the number of IPv4 addresses to be deployed in the address pool based on the number of clients and the time and frequency of connection and disconnection.

IPv4 addresses in an address pool can be classified by their usage as follows:

- **Used**: indicates that the IPv4 address has been used.
- **Idle**: indicates that the IPv4 address is idle.
- **Static-bind**: indicates that the IPv4 address has been bound to a MAC address and is not in use.
- **Static-bind used**: indicates that the IPv4 address has been bound to a MAC address and used.
- **Disable**: indicates that the IPv4 address cannot be used.

  The IPv4 addresses excluded using the **excluded-ip-addresses excluded-ip-address start-ip-address** *start-ip-address* **end-ip-address** *end-ip-address* command are in the Disable state.
- **Expired**: indicates that the lease of the IPv4 address has expired and the IPv4 address is idle.

  After an IPv4 address in the address pool expires, it enters the Expired state. The records of allocating IPv4 addresses in the Expired state are retained in the address pool. This allows the original IPv4 address to be allocated to a user, ensuring IPv4 address stability.

  When IPv4 addresses in the Idle state in the address pool are exhausted, the address pool automatically reclaims the IPv4 addresses in the Expired state and allocates them to new users.
- **Conflict**: indicates that the IPv4 address conflicts with another one on the network.

  Setting an IPv4 address to the Conflict state can prevent an IPv4 address conflict. An IPv4 address in the Conflict state exists in either of the following situations:

  – After receiving a DHCPDISCOVER message from a client, a DHCPv4 server pings an IPv4 address before allocating the IPv4 address. If the ping operation succeeds, the DHCPv4 server sets the IPv4 address to the Conflict state and allocates another IPv4 address to the client.

  – After obtaining an IPv4 address, a DHCPv4 client immediately sends a gratuitous ARP packet. If the client receives a response, it sends a DHCPDECLINE message to notify the DHCPv4 server of the IPv4 address conflict. The DHCPv4 server sets the IPv4 address to the Conflict state, and the client sends a DHCPDISCOVER message to apply for an IPv4 address again.

  When IPv4 addresses in the Idle and Expired states in the address pool are exhausted, the address pool automatically reclaims the IPv4 addresses in the Conflict state and allocates them to new users. An IPv4 address in the Conflict state cannot be allocated within 1 minute.

  Some wireless terminals respond to ping messages after sending DHCPDISCOVER messages to request IPv4 addresses, causing IPv4 address conflicts to be reported incorrectly. To prevent this issue, you can disable the ping function for the address pool using the **ping-packet-nub 0** command (the default value is 2, and the value 0 indicates that the ping function is disabled).

  For details about configuration parameters, see huawei-dhcp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Switch the interface working mode to Layer 3.

```
remove ethernet/main-interface/l2-attribute/
```

Determine whether to perform this step based on the current interface working mode.

Step 4 Configure an IPv4 address for the interface.

```
ipv4 addresses address ip ip-address
type main mask mask
```

The IPv4 address segment of the interface is the interface address pool. The interface address mask cannot be set to 31; otherwise, the interface address pool fails to be configured.

Step 5 Configure an interface address pool in the interface view.

```
interface-ip-pool select-interface
```

Step 6 (Optional) Configure network parameters for the interface address pool as required.

Table 7-30 Configuring network parameters for the interface address pool

Operation	Command	Description
Configure IPv4 addresses that cannot be automatically allocated to clients from the address pool in the interface address pool view.	**excluded-ip-addresses excluded-ip-address start-ip-address** *start-ip-address* **end-ip-address** *end-ip-address*	To configure multiple IPv4 addresses that cannot be automatically allocated to clients from the address pool, run this command multiple times. You do not need to exclude the gateway address configured using the **gateway-list** command, because the device automatically adds it to the list of IPv4 addresses that cannot be automatically allocated. You do not need to exclude the IPv4 address of the server's interface connected to a client, because the device automatically sets it to the Conflict state during address allocation. By default, all IPv4 addresses in an address pool can be automatically allocated.

Operation	Command	Description
Configure a fixed IPv4 address allocated to a specified client in the interface address pool view.	**static-binds static-bind static-bind-ip** *ip-address* **static-bind-mac** *mac-address* [**description** *description*]	Ensure that the bound IPv4 address is not set to an IPv4 address that cannot be allocated. The occupied IPv4 addresses can also be statically bound or unbound. When statically binding an IPv4 address to a MAC address, ensure that the MAC address to be bound is the same as the MAC address of the user who uses the IPv4 address. After a static binding is configured for a dynamically allocated IPv4 address, the lease becomes infinite. After the static binding is deleted, the lease is the same as that configured in the address pool. By default, no IP address in an address pool is bound to any MAC address.
Change an address lease in the interface address pool view.	**lease** { **day** *day* [**hour** *hour* [**minute** *minute*]] | **unlimited** }	After the lease of IPv4 addresses in the address pool is changed, the newly allocated IPv4 addresses use the new lease. The IPv4 addresses that have been allocated before the change still use the original lease before the lease is renewed. After the lease is renewed, these IPv4 addresses use the new lease. A BOOTP client does not support Option 51 (IPv4 address lease option). The lease of an IPv4 address allocated to a BOOTP client is infinite, and the DHCPv4 server does not reclaim the IPv4 address. The fixed IPv4 address allocated to a specified client can be used permanently and is not restricted by the lease. By default, the lease of IPv4 addresses in an interface address pool is one day.

Operation	Command	Description
Configure the device to automatically reclaim conflicting IPv4 addresses in the address pool in the interface address pool view.	**auto-recycle** **day** *day* [**hour** *hour* [**minute** *minute*]]	The DHCPv4 server selects available IPv4 addresses from the conflicting addresses for allocation only after all available addresses are allocated. You can configure the function of automatically reclaiming conflicting IPv4 addresses and set an automatic reclaiming interval, quickly reclaiming the conflicting IPv4 addresses. By default, a device is disabled from automatically reclaiming conflicting IPv4 addresses in an address pool.
Configure a gateway address for a client in the interface address pool view.	**gateway-list** *ip-address* &<1-8>	If a gateway address is configured for a client on the DHCPv4 server, the client obtains the gateway address from the DHCPv4 server and automatically generates a default route to the gateway address. The client can then access the hosts on other network segments. If the DHCPv4 server is configured to allocate a classless static route to the client using Option 121, the client does not automatically generate a default route to the gateway address. To balance traffic and improve network reliability, configure multiple gateways. Each address pool can be configured with a maximum of eight gateway addresses. If the DHCPv4 server and clients are on the same network segment and the DHCPv4 server functions as the gateway of the clients, this task is not required. By default, the default gateway address pre-allocated by the DHCPv4 server to a DHCPv4 client is not specified.

Operation	Command	Description
Configure a boot file for a client.	**server-name** *sname*: configures the name of a server from which a DHCPv4 client obtains the boot file. **next-server** *ip-address*: configures a file server IPv4 address used by a DHCPv4 client after the client obtains an IPv4 address.	The boot file can be stored on the DHCPv4 server or a dedicated file server. If the boot file is stored on a dedicated file server, you need to specify the file server address for a DHCPv4 client on the DHCPv4 server and ensure that the client and file server are routable. By default, the name of a server and the file server IPv4 address are not configured for a DHCPv4 client.
Configure DNS information allocated by the DHCPv4 server in the interface address pool view.	**dns-list**: configures the IPv4 address of a DNS server for a DHCPv4 client. **ip-address** *ip-address* &<1-8> **domain-name** *domain-name*: configures a DNS domain name suffix assigned to a DHCPv4 client.	You can perform this task to dynamically obtain the DNS configuration of a client through DHCPv4. Each address pool can be configured with a maximum of eight DNS server addresses. By default, no DNS server address or DNS domain name suffix assigned to a DHCPv4 client is configured in an address pool. By default, the DNS server IP address in the address pool on Vlanif1 is 114.114.114.114.
Configure a SIP server IPv4 address allocated by the DHCPv4 server.	**sip-server** { **sip-server-ip1** *ip-address1* [**sip-server-ip2** *ip-address2*] | **sip-server-name1** *domain-name1* [**sip-server-name2** *domain-name2*] }	After a SIP server IPv4 address is configured in the address pool of the DHCPv4 server, the DHCPv4 server specifies the SIP server IPv4 address when allocating IPv4 addresses to DHCPv4 clients. By default, no SIP server IPv4 address is configured in an address pool.

Step 7 Commit the configuration.

```
commit
```

----End

7.5.6.3 (Optional) Configuring Options

Context

Vendors can define options. A device functioning as a DHCPv4 server can allocate vendor-defined network parameters to clients.

For details about configuration parameters, see huawei-dhcp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-conifg
```

Step 2 Configure a user-defined option so that the DHCPv4 server allocates network parameters to a client based on the option requested by the client.

- Configure a user-defined option in the interface address pool view.

```
ifm interfaces interface name interface-name
options option option-code code
[ sub-options sub-option sub-option-code sub-code ]
{ ascii-string ascii-string | hex-string hex-string | ip-addresses ip-address &<1-8> }
quit 2
```

  Determine whether to switch the interface working mode to Layer 3 based on the current interface working mode.

By default, no user-defined option that a DHCPv4 server allocates to a DHCPv4 client is configured.

When configuring Option 33 (static route option), you must specify both the destination and next hop IP addresses.

Step 3 Commit the configuration.

```
commit
```

----End

Follow-up Procedure

Some options need to be configured using other commands in an address pool. The following table lists the options and their configuration methods.

Table 7-31 Options and their configuration methods

Option	Configuration Method	Description
Option 1	• Interface address pool: **ipv4 addresses address ip** *ip-address* Configured using the *mask* parameter in the **type main mask** *mask* command	Subnet mask
Option 3	• Interface address pool: configured using the **ipv4 addresses address ip** *ip-address* command Configured using the *ip-address* parameter in the **type main mask** *mask* command	Gateway address

Option	Configuration Method	Description
Option 6	• Interface address pool: configured using the **dns-list** command **ip-address** *ip-address* &<1-8>	DNS server address
Option 15	• Interface address pool: configured using the **domain-name** *domain-name* command	Domain name
Option 50	This option does not need to be configured on the DHCPv4 server.	Requested IPv4 address
Option 51	• Interface address pool: configured using the **lease** command { **day** *day* [**hour** *hour* [**minute** *minute*]] | **unlimited** }	IPv4 address lease
Option 52	This option does not need to be configured on the DHCPv4 server.	Additional option
Option 53	This option does not need to be configured on the DHCPv4 server.	DHCPv4 message type
Option 54	This option does not need to be configured on the DHCPv4 server.	Server identification
Option 55	This option does not need to be configured on the DHCPv4 server.	Parameter request list
Option 57	This option does not need to be configured on the DHCPv4 server.	Maximum DHCPv4 message length
Option 58	This option does not need to be configured on the DHCPv4 server.	Lease renewal time (T1), which is generally 50% of the lease
Option 59	This option does not need to be configured on the DHCPv4 server.	Lease renewal time (T2), which is generally 87.5% of the lease
Option 61	This option does not need to be configured on the DHCPv4 server.	Client identifier
Option 82	This option does not need to be configured on the DHCPv4 server.	Relay agent information
Option 120	• Interface address pool: configured using the **sip-server** command { **sip-server-ip1** *ip-address1* [**sip-server-ip2** *ip-address2*] | **sip-server-name1** *domain-name1* [**sip-server-name2** *domain-name2*] }	SIP server IPv4 address

7.5.6.4 Configuring the DHCPv4 Server Function

Context

If a device is configured as a DHCPv4 server, you must enable the DHCPv4 server function for an address pool after enabling DHCPv4 on the device.

For details about configuration parameters, see huawei-dhcp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enable the DHCPv4 server function for an address pool.

- Enable the interface-based DHCPv4 server function.
  a. Configure an IPv4 address for an interface.

     ```
     ifm interfaces interface name interface-name
     ipv4 addresses address ip ip-address
     type main mask mask
     ```

     Determine whether to switch the interface working mode to Layer 3 based on the current interface working mode.
  b. Enable the DHCPv4 server function on the interface to allocate an IPv4 address to a client from the interface address pool.

     ```
     interface-ip-pool select-interface
     ```

     If the device functions as a DHCPv4 server to provide DHCPv4 services for clients connected to multiple interfaces, repeat this step on each interface to enable the DHCPv4 server function.

Step 3 Enter the DHCP server view.

```
quit 5
dhcp server common
```

Step 4 (Optional) Configure optional functions for the DHCPv4 server as required.

Table 7-32 Configuring optional functions for the DHCPv4 server

Operation	Command	Description
Configure IPv4 address conflict detection in the DHCP server view.	**ping-packet-nub** *number* **ping-packet-timeout** *milliseconds*: configures the number of conflict detections during IPv4 address allocation and maximum waiting time for each conflict detection.	After this function is configured, the DHCPv4 server, before sending a DHCPOFFER message, sends an ICMP Echo request message with the source address being the IPv4 address of the DHCPv4 server and the destination address being the pre-allocated IPv4 address to detect conflicts for the allocated IPv4 address. • If the DHCPv4 server receives no ICMP Echo reply message within the detection period (number of detections x maximum waiting time for each detection), this IPv4 address is not used. In this case, the DHCPv4 server allocates the IPv4 address to the client. • If the DHCPv4 server receives an ICMP Echo reply message within the detection period, this IPv4 address is being used. In this case, the DHCPv4 server lists this IPv4 address as a conflicting one and waits for the next DHCPv4 request message. Do not set a long IPv4 address conflict detection period. Otherwise, clients cannot obtain IPv4 addresses. You are advised to set the detection period to less than 8 seconds. By default, a DHCPv4 server sends a maximum of two ping packets, and the maximum timeout period of each ping reply is 500 ms.

Operation	Command	Description
Configure the DHCPv4 server to dynamically allocate IPv4 addresses to BOOTP clients in the DHCP server view.	**bootp-enable true**: enables the DHCPv4 server to respond to BOOTP requests. **bootp-auto-enable true**: enables the DHCPv4 server to allocate IPv4 addresses to BOOTP clients.	If a BOOTP client exists on the network and the client needs to dynamically obtain an IPv4 address through DHCPv4, this step must be performed on the DHCPv4 server. By default, a DHCPv4 server is enabled to respond to BOOTP requests and dynamically allocate IPv4 addresses to BOOTP clients.

Step 5 Commit the configuration.

```
commit
```

----End

7.5.6.5 Verifying the Configuration

Procedure

- Run the **display /ifm/interfaces/interface[name="***interface-name***"]/ interface-ip-pool/select-interface/ip-pool-statistics/** command to check address allocation in the current address pool.
- Run the **display /dhcp/server/ip-pool-querys/** command to check information about conflicting, expired, and used IPv4 addresses in the current address pool.
- Run the **display /dhcp/server/packet-statistics/** command to check statistics about DHCPv4 messages sent and received on the DHCPv4 server.

----End

7.5.6.6 Example for Configuring a DHCPv4 Server Based on an Interface Address Pool

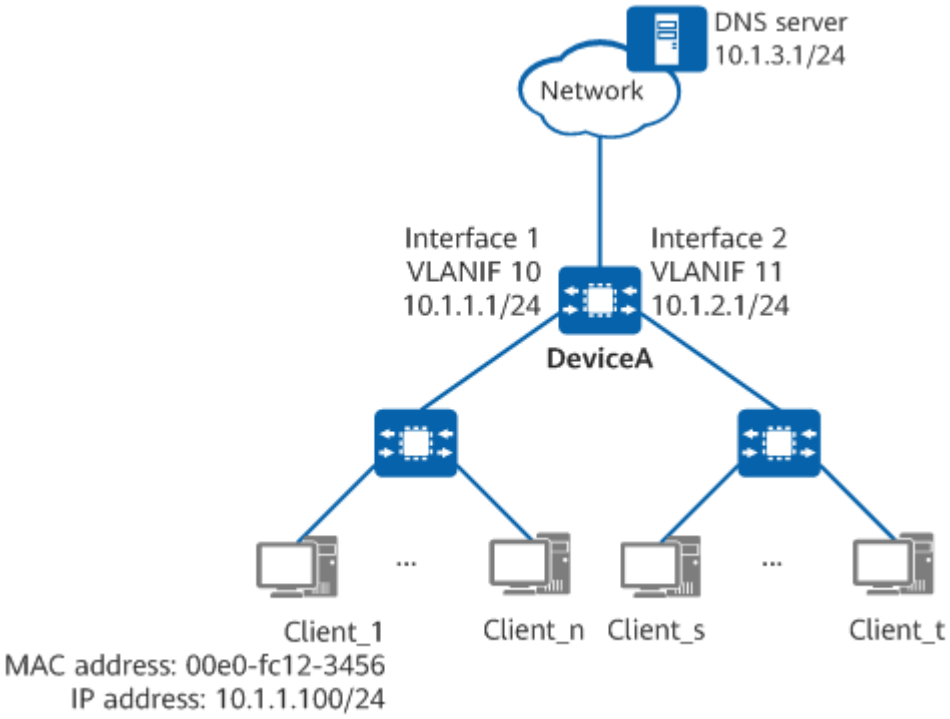
Networking Requirements

As shown in **Figure 7-30**, DeviceA functions as a DHCPv4 server; the PCs on network segment 10.1.1.0/24 are fixed terminals; network segment 10.1.2.0/24 is used for the terminals' temporary access. To facilitate unified management, the administrator requires the terminals to automatically obtain IPv4 addresses and the IPv4 address of the DNS server (if users need to access the network using domain names, a DNS server must be configured). A PC named Client_1 requires a fixed IPv4 address of 10.1.1.100/24 to meet service requirements.

Figure 7-30 Network diagram of configuring a DHCPv4 server based on an interface address pool

NOTE

In this example, interface 1 and interface 2 represent GE 0/0/1 and GE 0/0/2, respectively.



Configuration Roadmap

The configuration roadmap is as follows:

1. Set an IPv4 address lease to 30 days for the PCs (Client_1 to Client_n) on network segment 10.1.1.0/24, and allocate a fixed IPv4 address of 10.1.1.100/24 to Client_1 statically.
2. Set an IPv4 address lease to two days for the PCs (Client_s to Client_t) on network segment 10.1.2.0/24 for temporary access.

Procedure

Step 1 Enable DHCPv4.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> dhcp common global
[(gl)device@HUAWEI]/dhcp/common/global
MDCLI> enable true
[*(gl)device@HUAWEI]/dhcp/common/global
MDCLI> commit
```

Step 2 Add interfaces to VLANs.

Add GE 0/0/1 and GE 0/0/2 to VLAN 10 and VLAN 11, respectively.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> vlan vlans vlan id 10
[*(gl)device@HUAWEI]/vlan/vlans/vlan[id="10"]
```



```
MDCLI> quit 3
[*(gl)device@HUAWEI]
MDCLI> vlan vlans vlan id 11
[*(gl)device@HUAWEI]/vlan/vlans/vlan[id="11"]
MDCLI> quit 3
[*(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name GE0/0/1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> ethernet main-interface l2-attribute
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ethernet/main-interface/l2-attribute
MDCLI> pvid 10
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ethernet/main-interface/l2-attribute
MDCLI> quit 4
[*(gl)device@HUAWEI]/ifm/interfaces
MDCLI> interface name GE0/0/2
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/2"]
MDCLI> ethernet main-interface l2-attribute
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/2"]/ethernet/main-interface/l2-attribute
MDCLI> pvid 11
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/2"]/ethernet/main-interface/l2-attribute
MDCLI> commit
```

Step 3 Configure IPv4 addresses for VLANIF interfaces.

Configure an IPv4 address for VLANIF 10.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name Vlanif10
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]
MDCLI> ipv4 addresses address ip 10.1.1.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/ipv4/addresses/address[ip="10.1.1.1"]
MDCLI> type main mask 255.255.255.0
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/ipv4/addresses/address[ip="10.1.1.1"]
MDCLI> commit
```

Configure an IPv4 address for VLANIF 11.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name Vlanif11
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]
MDCLI> ipv4 addresses address ip 10.1.2.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/ipv4/addresses/address[ip="10.1.2.1"]
MDCLI> type main mask 255.255.255.0
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/ipv4/addresses/address[ip="10.1.2.1"]
MDCLI> commit
```

Step 4 Configure interface address pools.

Configure the clients connected to VLANIF 10 to obtain IPv4 addresses and other network parameters from the address pool on VLANIF 10.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name Vlanif10
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]
MDCLI> interface-ip-pool select-interface
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface
MDCLI> gateway-list 10.1.1.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface
MDCLI> lease
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/lease
MDCLI> day 30
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/lease
MDCLI> quit
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface
MDCLI> domain-name huawei.com
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface
```



```
MDCLI> dns-list
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/dns-list
MDCLI> ip-address 10.1.3.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/dns-list
MDCLI> quit
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface
MDCLI> static-binds static-bind static-bind-ip 10.1.1.100
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/static-
binds/static-bind[static-bind-ip="10.1.1.100"]
MDCLI> static-bind-mac 00e0-fc12-3456
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/static-
binds/static-bind[static-bind-ip="10.1.1.100"]
MDCLI> commit
```

Configure the clients connected to VLANIF 11 to obtain IPv4 addresses and other network parameters from the address pool on VLANIF 11.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name Vlanif11
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]
MDCLI> interface-ip-pool select-interface
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface
MDCLI> gateway-list 10.1.2.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface
MDCLI> lease
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface/lease
MDCLI> day 2
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface/lease
MDCLI> quit
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface
MDCLI> domain-name huawei.com
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface
MDCLI> dns-list
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface/dns-list
MDCLI> ip-address 10.1.3.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface/dns-list
MDCLI> commit
```

----End

Verifying the Configuration

Check IPv4 address configuration and allocation in address pools on DeviceA. The **used-ip-count** field displays the number of used IPv4 addresses in each address pool.

```
MDCLI> display /ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/
{
  "domain-name": "huawei.com",
  "gateway-list": [
    "10.1.1.1"
  ],
  "lease": {
    "day": 30
  },
  "dns-list": {
    "ip-address": [
      "10.1.3.1"
    ]
  },
  "static-binds": {
    "static-bind": [
      {
        "static-bind-ip": "10.1.1.100",
        "static-bind-mac": "00e0-fc12-3456"
      }
    ]
```



```
  }
}
MDCLI> display /ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/ip-pool-statistics/
{
  "used-ip-count": 3,
  "idle-ip-count": 250,
  "expired-ip-count": 0,
  "conflict-ip-count": 0,
  "disable-ip-count": 0,
  "total-ip-count": 253,
  "start-ip": "10.1.1.1",
  "end-ip": "10.1.1.254"
}
MDCLI> display /ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface/
{
  "domain-name": "huawei.com",
  "gateway-list": [
    "10.1.2.1"
  ],
  "lease": {
    "day": 2
  },
  "dns-list": {
    "ip-address": [
      "10.1.3.1"
    ]
  }
}
MDCLI> display /ifm/interfaces/interface[name="Vlanif11"]/interface-ip-pool/select-interface/ip-pool-statistics/
{
  "used-ip-count": 3,
  "idle-ip-count": 250,
  "expired-ip-count": 0,
  "conflict-ip-count": 0,
  "disable-ip-count": 0,
  "total-ip-count": 253,
  "start-ip": "10.1.2.1",
  "end-ip": "10.1.2.254"
}
```

Check IPv4 address information on Client_1. You can check that Client_1 has obtained the IPv4 address 10.1.1.100/24.

Check IPv4 address information on other DHCPv4 clients. You can check that the clients have obtained IPv4 addresses.

7.5.7 Configuring a Device to Function as a DHCPv4 Client

7.5.7.1 Configuring the DHCPv4 Client Function

Context

After the DHCPv4 client function is configured on a device, the device can obtain configuration parameters such as an IPv4 address from a DHCPv4 server.

If the IPv4 address allocated by a DHCPv4 server to a device's interface is on the same network segment as the IPv4 address of another interface on the device, the interface does not use this address or apply for a new IPv4 address from the server. To enable the interface to apply for a new IPv4 address, run the **admin-status down** and **admin-status up** commands in sequence on the interface, or

run the **remove dhcp-client-if/address-allocation**, **dhcp-client-if**, and **address-allocation [null]** commands in sequence.

After the DHCPv4 client function is enabled on a device's interface, the interface cannot communicate with the DHCPv4 server when the device functions as a DHCPv4 relay agent.

For details about configuration parameters, see huawei-dhcp.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Switch the interface working mode to Layer 3.

```
remove ethernet/main-interface/l2-attribute/
```

Determine whether to perform this step based on the current interface working mode.

Step 4 Enable the DHCPv4 client function on the interface.

```
dhcp-client-if
address-allocation [ null ]
```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

- Run the **display /ifm/interfaces/interface[name="***interface-name***"]/dhcp-client-if/client-status/** command to check DHCPv4 client lease information.
- Run the **display /ifm/interfaces/interface[name="***interface-name***"]/dhcp-client-if/client-statistics/** command to check message statistics on the DHCPv4 client.

7.5.7.2 Example for Configuring a DHCPv4 Client

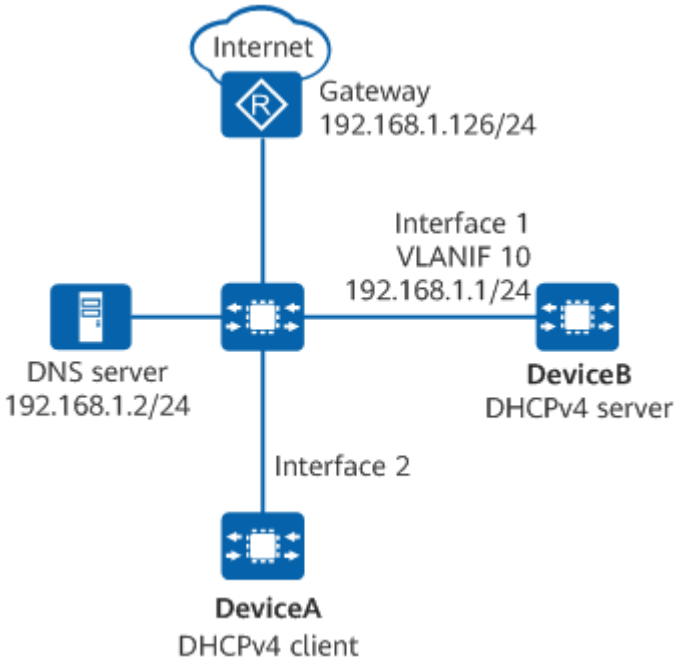
Networking Requirements

As shown in **Figure 7-31**, DeviceA functions as a DHCPv4 client and needs to obtain information such as an IPv4 address, DNS server address, and gateway address from DeviceB functioning as a DHCPv4 server.

Figure 7-31 Network diagram of configuring a DHCPv4 client

NOTE

In this example, interface 1 and interface 2 represent GE 0/0/1 and GE 0/0/9, respectively.



Procedure

Step 1 Configure the DHCPv4 client function on DeviceA.

Enable the DHCPv4 client function on the WAN interface GE 0/0/9.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name GE0/0/9
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]
MDCLI> dhcp-client-if
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/dhcp-client-if
MDCLI> address-allocation [ null ]
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/9"]/dhcp-client-if
MDCLI> commit
```

Step 2 Create an interface address pool for the DHCPv4 server on DeviceB and configure network parameters.

1. Enable DHCPv4.

   ```
   MDCLI> edit-config
   [(gl)device@HUAWEI]
   MDCLI> dhcp common global
   [(gl)device@HUAWEI]/dhcp/common/global
   MDCLI> enable true
   [*(gl)device@HUAWEI]/dhcp/common/global
   MDCLI> commit
   ```

2. Create VLAN 10 and add GE 0/0/1 to it.

   ```
   MDCLI> edit-config
   [(gl)device@HUAWEI]
   MDCLI> vlan vlans vlan id 10
   [*(gl)device@HUAWEI]/vlan/vlans/vlan[id="10"]
   MDCLI> quit 3
   [*(gl)device@HUAWEI]
   MDCLI> ifm interfaces interface name GE0/0/1
   [*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]
   MDCLI> ethernet main-interface l2-attribute
   [*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ethernet/main-interface/l2-
   attribute
   MDCLI> pvid 10
   ```



```
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="GE0/0/1"]/ethernet/main-interface/l2-
attribute
MDCLI> commit
```

3. Enable the DHCPv4 server function on VLANIF 10 to allocate an IPv4 address to a client from the interface address pool.

```
MDCLI> edit-config
[(gl)device@HUAWEI]
MDCLI> ifm interfaces interface name Vlanif10
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]
MDCLI> ipv4 addresses address ip 192.168.1.1
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/ipv4/addresses/
address[ip="192.168.1.1"]
MDCLI> type main mask 255.255.255.0
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/ipv4/addresses/
address[ip="192.168.1.1"]
MDCLI> quit 3
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]
MDCLI> interface-ip-pool select-interface
[*(gl)device@HUAWEI]/ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface
MDCLI> commit
```

----End

Verifying the Configuration

After the interface obtains an IPv4 address, check the status of the DHCPv4 client on DeviceA.

```
MDCLI> display /ifm/interfaces/interface[name="GE0/0/9"]/dhcp-client-if/client-status
{
  "fsm-state": "bound",
  "mac-address": "00e0-fc12-7901",
  "ip-address": "192.168.1.3",
  "ip-mask": "255.255.255.0",
  "server-address": "192.168.1.2",
  "lease-obtained-time": "2022-11-16T11:06:26Z",
  "lease-expire-time": "2022-11-17T11:06:26Z",
  "lease-renew-time": "2022-11-16T23:06:26Z",
  "lease-rebind-time": "2022-11-17T08:06:26Z"
}
```

Check IPv4 address allocation in the interface address pool on DeviceB. The **used-ip-count** field displays the number of used IPv4 addresses in the interface address pool.

```
MDCLI> display /ifm/interfaces/interface[name="Vlanif10"]/interface-ip-pool/select-interface/ip-pool-
statistics/
{
  "used-ip-count": 1,
  "idle-ip-count": 253,
  "expired-ip-count": 0,
  "conflict-ip-count": 0,
  "disable-ip-count": 0,
  "total-ip-count": 254,
  "start-ip": "192.168.1.1",
  "end-ip": "192.168.1.254"
}
```


7.5.8 Maintaining DHCPv4

Clearing DHCPv4 Message Statistics

NOTICE

Statistics cannot be restored after being cleared. Exercise caution when clearing the statistics.

To clear DHCPv4 message statistics, run reset commands in the user view.

Table 7-33 Clearing DHCPv4 message statistics

Operation	Command
Clear statistics about DHCPv4 messages sent and received by the device functioning as a DHCPv4 server.	**reset-server-statistics**
Clear statistics about DHCPv4 messages sent and received by the device functioning as a DHCPv4 client.	**reset-client-statistics** [**if-name** *interface-name*]

Resetting an Address Pool

To allow a device that functions as a DHCPv4 server to re-allocate IPv4 addresses from an address pool to clients or reset IPv4 addresses to idle, reset the address pool. Note that idle IPv4 addresses are preferentially allocated.

If a device functions as a DHCPv4 relay agent, it can request a DHCPv4 server to release client IPv4 addresses. A DHCPv4 relay agent can send a DHCPRELEASE message to a specified DHCPv4 server. After receiving the message, the server resets the corresponding IPv4 address to idle, allowing the released IPv4 address to be allocated to another DHCPv4 client.

Table 7-34 Resetting an address pool

Operation	Command
Reset an interface address pool configured on the device.	User view: **reset-interface-pool-address-state if-name** *interface-name* **reset-flag** { **ip start-ip-address** *start-ip-address* [**end-ip-address** *end-ip-address*] | **all** | **conflict** | **expired** | **used** }

Canceling the Allocation of Fixed IPv4 Addresses to Clients

On a device functioning as a DHCPv4 server, you can cancel the allocation of a specified IPv4 address to a static client. For example, you can cancel the allocation

of the IPv4 address 10.1.1.5 in the address pool whose network segment address is 10.1.1.0 and mask length is 24 to a client. You can run the **display /dhcp/server/ip-pool-querys** command to check the static bindings between clients and IPv4 addresses.

Table 7-35 Canceling the allocation of fixed IPv4 addresses to clients

Operation	Command
Reclaim IPv4 addresses.	User view: **reset-interface-pool-address-state if-name** *interface-name* **reset-flag** { **ip start-ip-address** *start-ip-address* [**end-ip-address** *end-ip-address*] | **all** | **conflict** | **expired** | **used** }
Remove a static binding.	• If an interface address pool is used, run the following command in the corresponding interface address pool view: **remove static-binds/static-bind[static-bind-ip="***ip-address***"]/**

7.5.9 Troubleshooting DHCPv4

7.5.9.1 IPv4 Address Obtained by a Client Conflicts with That of Another Client

Possible Cause	The IPv4 address is manually configured for another host on the network. The DHCPv4 server does not exclude this IPv4 address from the address pool, leading to an IPv4 address conflict.
Troubleshooting Roadmap	Disable the network adapter of the client or disconnect the network cable. Perform a ping operation on another host to check whether any host with this IPv4 address exists. If the ping operation is successful, the IPv4 address has been manually configured.
Solution	• Change the manually configured IPv4 address. • Exclude the conflicting IPv4 address from the DHCPv4 server's address pool. – If the device functions as a DHCPv4 server and an interface address pool is configured, run the **excluded-ip-addresses excluded-ip-address start-ip-address** *start-ip-address* **end-ip-address** *end-ip-address* command. To prevent the client from obtaining a conflicting IPv4 address, configure IPv4 address conflict detection on the DHCPv4 server. After detecting an IPv4 address conflict, the server allocates another available IPv4 address. For details, see "Configure IPv4 address conflict detection." in **7.5.6.4 Configuring the DHCPv4 Server Function**.

7.5.9.2 Client Cannot Obtain an IPv4 Address from a DHCPv4 Server

Possible Cause 1	DHCPv4 is not enabled.
Troubleshooting Roadmap	Run the **display /dhcp/common/global/** command in the user view to check whether DHCPv4 is enabled. If the command output is empty, DHCPv4 is not enabled.
Solution	In the edit-config view, run the **dhcp common global** command and then the **enable true** command to enable DHCPv4.

Possible Cause 2	The configuration is incorrect.
Troubleshooting Roadmap	For the DHCPv4 server: • Run the **display interface-ip-pool/select-interface/** command in the interface view to check whether an address pool on the same network segment as the client is configured, and check whether the configuration is correct. • Run the **display this** command in the view of the interface connected to the client to check whether the DHCPv4 server function is enabled on the interface. • If a DHCPv4 relay agent is deployed between the server and client, check whether a route to the network segment of the client is configured on the server.
Solution	Modify the configuration or deployment of the DHCPv4 server. • For configuration details about the DHCPv4 server, see **7.5.6 Configuring a Device to Function as a DHCPv4 Server**.

Possible Cause 3	The address pool has no available IPv4 addresses.
Troubleshooting Roadmap	Run the **display interface-ip-pool/select-interface/ip-pool-statistics** command to check whether there are available IPv4 addresses in the address pool. The **idle-ip-count** field in the command output indicates the number of idle IPv4 addresses in the address pool. If the value of this field is 0, there are no available IPv4 addresses in the address pool.

Solution	Determine the number of DHCPv4 clients on the network. • If the number of clients is greater than the number of available IPv4 addresses in the address pool, increase the address range. – In the interface view, run the **ipv4 addresses address ip** *ip-address* command and then the **mask** *mask* command to increase the address range by reducing the mask length. • If the number of clients is less than the number of available IPv4 addresses in the address pool, reduce the address lease to ensure that unused IPv4 addresses can be reclaimed promptly. You can configure the function of automatically reclaiming conflicting IPv4 addresses and set an automatic reclaiming interval, quickly reclaiming the conflicting IPv4 addresses. – Run the **auto-recycle day** *day* [**hour** *hour* [**minute** *minute*]] command in the interface view.

Possible Cause 4	The IPv4 address is manually configured for another host on the network. The DHCPv4 server does not exclude this IPv4 address from the address pool, leading to an IPv4 address conflict.
Troubleshooting Roadmap	Disable the network adapter of the client or disconnect the network cable. Perform a ping operation on another host to check whether any host with this IPv4 address exists. If the ping operation is successful, the IPv4 address has been manually configured.
Solution	• Change the manually configured IPv4 address. • Exclude the conflicting IPv4 address from the DHCPv4 server's address pool. – If the device functions as a DHCPv4 server and an interface address pool is configured, run the **excluded-ip-addresses excluded-ip-address start-ip-address** *start-ip-address* **end-ip-address** *end-ip-address* command. To prevent the client from obtaining a conflicting IPv4 address, configure IPv4 address conflict detection on the DHCPv4 server. After detecting an IPv4 address conflict, the server allocates another available IPv4 address. For details, see "Configure IPv4 address conflict detection." in **7.5.6.4 Configuring the DHCPv4 Server Function**.

7.6 DNS Configuration

7.6.1 Overview of DNS

Definition

TCP/IP allows devices to communicate through IP addresses, but users find it difficult to remember these IP addresses. To resolve this problem, a domain name system (DNS) was designed to match IP addresses with human-readable hostnames.

Purpose

A DNS server is configured on a network to establish mappings between domain names and IP addresses. These mappings give users easy-to-remember alternatives for identifying devices.

7.6.2 Understanding DNS

DNS over the Internet

Initially, the domain names of devices consisted of a sequence of characters. All of these domain names formed a non-hierarchical domain name structure, which makes it inconvenient for administrators to manage a large number of domain names for the following reasons:

- Domain names consist of characters, which may result in a name conflict.
- The domain name structure is not hierarchical. As the number of hostnames increases, so does the management workload.
- The mappings between domain names and IP addresses frequently change. Therefore, maintaining the domain namespace is a huge undertaking.

To address this, a hierarchical domain name structure was defined for the Internet by the DNS in the TCP/IP protocol stack. The DNS divides the Internet into multiple top-level domains (TLDs). **Table 7-36** lists the domain name of each TLD. TLDs typically represent either an organization type or a geographical location. Geographic TLDs are used to classify domain names based on countries. Before joining the Internet, each country registers a country code TLD that represents their country with the NIC. For example, "cn" represents China, and "us" represents the United States.

Table 7-36 TLDs and their meanings

TLD	Meaning
com	Commercial organizations
edu	Educational agencies
gov	Governmental agencies
mil	Military departments
net	Main network support centers

TLD	Meaning
int	International organizations
org	Other organizations
Country code	Countries (classified in geography mode)

NOTE

The first seven domains are defined in organization mode, and the country code domain is defined in geography mode.

The NIC authorizes management agencies to classify domains into sub-domains. The agencies in charge of this can authorize subordinate agencies to continue classifying domains. As a result, the Internet has a hierarchical domain name structure.

Static Domain Name Resolution

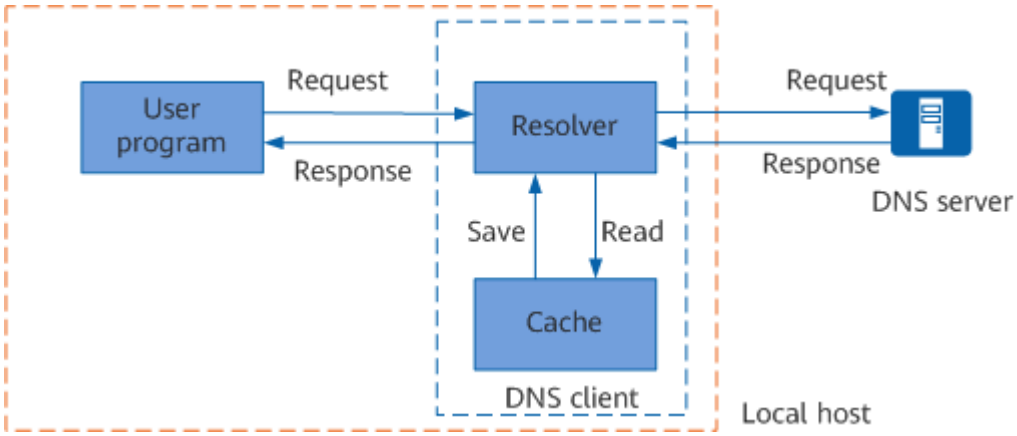
The DNS supports dynamic and static domain name resolution. Static domain name resolution is first used to resolve a domain name. If the resolution fails, dynamic domain name resolution is used.

Static domain name resolution requires a static domain name resolution table, which is manually created and holds mappings between commonly used domain names and IP addresses. A DNS client first searches the static domain name resolution table for a domain name to resolve it into an IP address. This is an efficient method for domain name resolution.

Dynamic Domain Name Resolution

Dynamic domain name resolution requires a dedicated DNS server. This server runs the domain name resolution program, maps domain names to IP addresses, and receives DNS requests from clients.

Figure 7-32 Implementation diagram of dynamic domain name resolution



In Figure 7-32, the DNS client, consisting of the resolver and cache, is used to receive and respond to DNS requests from the user program. Typically, the user

program, cache, and resolver are located on the same host, whereas the DNS server is on another host. The dynamic domain name resolution process is as follows:

1. When a user program (such as ping or Telnet) uses a domain name to access the network, it sends a DNS request to the resolver of the DNS client.
2. After receiving the request, the resolver first checks the local cache.
   - If the resolver finds the mapping entry for the domain name in the local cache, it directly returns the mapped IP address to the user program.
   - If the resolver does not find such a mapping entry in the local cache, it sends a request to the DNS server.
3. The DNS server checks whether the requested domain name is within a sub-domain it manages and then responds to the DNS client accordingly.
   - If the requested domain name is within a sub-domain it manages, the DNS server searches for the IP address corresponding to the domain name in its own database.
   - If the requested domain name is not within a sub-domain it manages, the DNS server forwards the request to upper-level DNS servers. After completing the resolution, the corresponding upper-level DNS server returns the result to the DNS client.
4. The resolver receives and resolves the response sent by the DNS server, and returns the result to the user program.

Dynamically resolved mappings between domain names and IP addresses are stored in the cache. If a domain name is searched for again, the DNS client obtains the corresponding IP address from the cache directly instead of sending a request to the DNS server. Mappings stored in the cache will expire and be deleted after a period to ensure that the latest mappings can be obtained from the DNS server.

DNS Query Types

The IPv4 DNS supports the following query types:

- A query is the most commonly used type of query, and is used to obtain the IP address corresponding to a specified domain name. For example, when you ping or tracert a domain name, a query is sent to the DNS client for the IP address corresponding to the domain name. If the corresponding IP address does not exist on the DNS client, the DNS client sends an A query to the DNS server for obtaining.
- Pointer record (PTR) query means that the DNS client obtains the corresponding domain name based on the IP address. PTRs constitute the mappings between domain names and IP addresses provided by the DNS server for PTR query.

The IPv6 DNS supports the following query types:

- AAAA query: uses a domain name to query an IPv6 address.
- IPv6 PTR query: uses an IPv6 address to query a domain name.

7.6.3 Configuration Precautions for DNS

Licensing Requirements

DNS is not under license control.

Hardware Requirements

Table 7-37 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

7.6.4 Default Settings for DNS

Table 7-38 describes the default settings for DNS.

Table 7-38 Default settings for DNS

Parameter	Default Setting
Dynamic domain name resolution	Enabled
Domain name suffix list function	Enabled

7.6.5 Configuring Basic DNS Functions

7.6.5.1 Configuring Static Domain Name Resolution

Context

A static domain name resolution table is set up manually, with mappings between domain names and IP addresses. Common domain names can be added to the table for static domain name resolution. A DNS client first searches the static domain name resolution table for a domain name to resolve it into an IP address. This is an efficient method for domain name resolution.

For details about configuration parameters, see huawei-dns.yang.

Procedure

- Configure IPv4 static domain name resolution.
  a. Enter the edit-config view.
     edit-config
  b. Enter the IPv4 static DNS host view.
     dns ipv4-hosts
  c. Configure an IPv4 static DNS hostname.
     ipv4-host vpn *vpn-value* **host** *host-name*

     vpn-value is always set to **_public_** because the VPN function is currently not supported.
  d. Configure an IPv4 static DNS address.
     address *ip-address*

     By default, no IPv4 static DNS address is configured.

     Each hostname can be mapped to only one IPv4 address. If a hostname is mapped to multiple IPv4 addresses, only the latest configuration takes effect. If multiple hostnames need to be resolved, repeat Step 3 and Step 4.
  e. Commit the configuration.
     commit
- Remove IPv4 static domain name resolution.
  a. Enter the configuration path.
     dns ipv4-hosts
  b. Enter the edit-config view.
     edit-config
  c. Remove an IPv4 static DNS hostname.
     remove ipv4-host[**vpn**=*vpn-value*][**host**=*host-name*]

     vpn-value is always set to **_public_** because the VPN function is currently not supported.
  d. Commit the configuration.
     commit

----End

Verifying the Configuration

Run the **display query-host-ip**[**host-name=***host-name*] command in the **dns/query-host-ips** path to check static DNS entries.

7.6.5.2 Configuring Dynamic Domain Name Resolution

Prerequisites

Before configuring dynamic domain name resolution, you have completed the following task:

- Configure link layer protocol parameters for the interfaces to ensure that the link layer protocol status of the interfaces is up.

- Configure a route between the device and DNS server.

Context

Dynamic domain name resolution requires a dedicated DNS server, which receives domain name resolution requests from DNS clients. Upon receiving a request, the DNS server searches for the corresponding IP address of the domain name in its DNS database. If no matching entry is found, it sends the request to a higher-level DNS server. This process continues until the DNS server finds the corresponding IP address or detects that the corresponding IP address of the domain name does not exist. The DNS server then returns a result to the DNS client.

For details about configuration parameters, see huawei-dns.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the DNS server view.

```
dns ipv4-servers
```

Step 3 Select the corresponding configuration as required.

- Configure the IPv4 address of the DNS server.
  ipv4-server vpn *vpn-value* **address** *ip-address*
- Remove the IPv4 address of the DNS server.
  remove ipv4-server[**vpn**=*vpn-value*][**address**=*ip-address*]

  vpn-value is always set to **_public_** because the VPN function is currently not supported.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display ipv4-servers/ all** command in the **dns** path to check information about the DNS server.

7.6.5.3 (Optional) Configuring a Suffix for a Device Name

Context

Devices in different domains may have the same system name. You can use a fully qualified domain name (FQDN) to uniquely identify such a device. An FQDN consists of a device name and a domain name. To add a suffix for a device name, perform this configuration. In this situation, the System Name TLV in an LLDP packet is *device name.suffix*. For example, if the device name is HUAWEI and the suffix is area1, the System Name TLV in an LLDP packet is HUAWEI.area1. By default, a device name does not have a suffix.

For details about configuration parameters, see huawei-dns.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the DNS domain name suffix configuration view.

```
dns domains domain-name
```

Step 3 Select the corresponding configuration as required.

- Configure a DNS domain name suffix.
  ```
  domain vpn vpn-value name domain-value
  ```
- Delete a DNS domain name suffix.
  ```
  remove domain[vpn="vpn-value][name="domain-vaule"]
  ```

vpn-value is always set to **_public_** because the VPN function is currently not supported.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display /dns/domains all** command to check the configured DNS domain name.

7.6.6 Troubleshooting DNS

7.6.6.1 Dynamic Domain Name Resolution Cannot Be Implemented on a Device

Fault Symptom

The device functions as a DNS client that is configured with dynamic domain name resolution but is unable to resolve a domain name to a correct IP address.

Procedure

Step 1 Run the **display ipv4-servers/ all** command in the **dns** path to check whether the DNS server IP address configured on the DNS client is correct.

Step 2 If the DNS server IP address is incorrect, enter the **dns/ipv4-servers** path, run the **remove ipv4-server**[**vpn**=0][**address**=*ip-address*] command to remove the configured DNS server IP address, and run the **ipv4-server vpn 0 address** *ip-address* command to reconfigure a correct one.

----End

7.7 ACL Configuration

7.7.1 Overview of ACL

Definition

An access control list (ACL) is a set of one or more rules that define packet filtering conditions, such as source addresses, destination addresses, and port numbers of packets.

An ACL is used by a device to filter packets based on rules contained in the ACL. Then, the device permits or denies matched packets according to the policy used by the service module where the ACL is applied.

Purpose

ACLs accurately identify and control packets on a network to manage network access behaviors, prevent network attacks, and improve bandwidth utilization. In this way, ACLs ensure security and service quality.

7.7.2 Understanding ACL

7.7.2.1 ACL Classification

Table 7-39 describes ACL classification by function.

Table 7-39 ACL classification

Category	Function	Number Range
Basic ACL	Defines packet filtering rules based on information such as source IPv4 addresses, fragment information, and time ranges.	2000 to 2999
Advanced ACL	Defines packet filtering rules based on information such as source and destination IPv4 addresses, IP protocol types, TCP source and destination port numbers, UDP source and destination port numbers, fragment information, and time ranges.	3000 to 3999
Layer 2 ACL	Defines packet filtering rules based on the information in Ethernet frame headers, such as source and destination MAC addresses, VLAN IDs, and Layer 2 protocol types.	4000 to 4999

You can create a numbered or named ACL.

- A named ACL is created using a name. This type of ACL is named in the format of name + number. The number can be manually specified or automatically assigned by the system. If a number is not manually specified for an ACL, the system assigns the largest number available for the ACL.

NOTE

Once a named ACL is created, its name cannot be changed. If a different name is required, the named ACL must be deleted and a new one created.

- A numbered ACL is created using a number.

7.7.2.2 ACL Matching Process

ACL Rule ID

- All rules in an ACL are identified by rule IDs and arranged in ascending order of ID. Rule IDs can be manually configured or automatically generated based on the ACL step.
- If rule IDs are automatically generated, they are incremented in predefined steps. For example, if the ACL step is set to 5, the difference between rule IDs will be 5. In this case, the first rule ID is 5, and subsequent IDs will increase by increments of 5. This facilitates ACL management, enabling you to add new rules between existing ones.
- In the configuration file, the rules are displayed in ascending order of ID, not in the order of configuration.

ACL Matching Mechanism

A device stops matching packets against ACL rules once the packets match a rule.

After a packet is filtered by an ACL, the packet may be matched or unmatched:

- Matched: The packet matches a rule in the ACL.
- Unmatched: The packet does not match any rule in the ACL, the ACL is unavailable, or the ACL does not contain rules.

Whether packets are permitted or denied is determined by actions specified in matched rules of an ACL and the service module that has this ACL applied. Different service modules may process the packets that are filtered by ACL rules in different ways.

ACL Matching Order

The system matches packets against ACL rules in ascending order of rule IDs. That is, the rule with the smallest ID takes effect first.

- If a smaller rule ID is manually specified for a rule, the rule takes effect earlier than those with larger rule IDs.
- If no ID is manually specified for a rule, the system allocates one. This rule ID is the largest in the ACL and has the minimum multiple of the step. Therefore, this rule is the last one that functions.

ACL Step

An ACL step is the difference between two adjacent ACL rule IDs that are automatically allocated. For example, if the ACL step is set to 5, the rule IDs are multiples of 5, such as 5, 10, 15, and 20.

NOTE

If no rule is configured in an ACL, the step can be modified. Otherwise, the step cannot be modified.

7.7.2.3 ACL Configuration Guidelines

When configuring ACL rules, follow these guidelines:

1. Some rules in an ACL may overlap, and matching stops for a packet once the packet matches a rule in an ACL. To prevent a packet from matching a rule with loose conditions earlier than a rule with strict conditions in the same ACL, ensure that rules with strict conditions are arranged above those with loose conditions.
2. The ACL configuration guidelines vary according to the default ACL actions taken by service modules. Take a service module whose default action is permit as an example. If the module must deny packets from only some IPv4 addresses, you can configure a deny rule only for these IPv4 addresses. Because the default action is permit, you do not need to add a permit rule for other IPv4 addresses as the last rule in the ACL. The opposite is true for a service module whose default action is deny. **Table 7-40** describes the ACL configuration guidelines.

   NOTE

   The following rules are for reference only. The command line syntax shall prevail when you configure ACL rules.

   The **rule permit** and **rule deny** commands permit and deny packets, respectively. In **Table 7-40**, *a* and *b* indicate packet attributes, with *b* covering a wider range of packets than *a*.

Table 7-40 ACL configuration guidelines

Default ACL Action	Permit All Packets	Deny All Packets	Permit a Few Packets and Deny Most Packets	Deny a Few Packets and Permit Most Packets
permit	No ACL is required.	Configure **rule deny**.	Configure **rule permit** *a* first, and then **rule deny** *b* or **rule deny**. **NOTE** This guideline applies to packet filtering. When an ACL is applied to traffic policing or traffic statistics collection in a traffic policy, configure **rule permit** *a* if you only need to count rate or collect statistics on the specified packets.	Only **rule deny** *a* is required, and **rule permit** *b* or **rule permit** is not required. **NOTE** If **rule permit** is configured and an ACL is applied to a traffic policy in which the traffic behavior's action is set to **deny**, all packets are rejected and all services are interrupted.
deny	Configure **rule permit**.	No ACL is required.	Only **rule permit** *a* is required, and **rule deny** *b* or **rule deny** is not required.	Configure **rule deny** *a* first, and then **rule permit** *b* or **rule permit**.

7.7.3 Configuration Precautions for ACL

Licensing Requirements

ACL is not under license control.

Hardware Requirements

Table 7-41 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST

Series	Models
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

7.7.4 Default Settings for ACL

Table 7-42 describes the default settings for ACL.

Table 7-42 Default settings for ACL

Parameter	Default Setting
Step between ACL rules	5
ACL's effective time range	Not configured
ACL description	Not configured
ACL rule description	Not configured

7.7.5 Creating a Time Range in Which an ACL Takes Effect

Context

A time range defines when ACL rules are in effect, for example, during a specific time range or a fixed time range of each week. This allows network administrators to configure different policies during different time ranges for network optimization.

For details about configuration parameters, see huawei-acl.yang and huawei-time-rang.yang.

Time ranges associated with ACL rules are classified as follows:

- Absolute time range: defined by a period of time, in the format of from YYYY/MM/DD HH:MM to YYYY/MM/DD HH:MM. That is, ACL rules take effect only during this period.
- Periodic time range: defined by week. That is, ACL rules can take effect at an interval of one week. For example, if the time range of ACL rules is 08:00 to 12:00 on Monday, the ACL rules take effect during this time at each week.

A time name can be specified for multiple time ranges, which take effect based on the following principles:

- If only periodic time ranges are configured, all the periodic time ranges take effect.

- If only absolute time ranges are configured, all the absolute time ranges take effect.
- You are advised not to configure both periodic and absolute time ranges. If both periodic and absolute time ranges are configured, all the periodic and absolute time ranges take effect.

CAUTION

- To associate a time range with an ACL rule, ensure that the system time of the device is the same as that of other devices on the network; otherwise, the rule may not take effect. A time name must have been configured for the time range; otherwise, the time range cannot be associated with the rule.
- Before deleting a time range, you must delete the ACL rules associated with the time range or delete the ACL to which the ACL rules belong. Deleting the time range of an ACL may cause some ACL rules to become invalid. Exercise caution when performing this operation.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Create a time range instance.

```
time-range time-range-instances time-range-instance name name
```

Step 3 Create an absolute or periodic time range.

- Create an absolute time range.
  ```
  absolute-ranges absolute-range start-time start-time end-time end-time
  ```
- Create a periodic time range.
  ```
  period-ranges period-range day-of-week "day-of-week" start-time start-time end-time end-time
  ```
  day-of-week can be set to **sunday**, **monday**, **tuesday**, **wednesday**, **thursday**, **friday**, **saturday**, or a combination of them.

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

- Run the **display time-range/time-range-instances/time-range-instance**[**name**=*time-name*] **all** command to check the configuration and status of a specified time range.
- Run the **display time-range/time-range-instances/time-range-instance all** command to check the configurations and status of all time ranges.

Follow-up Procedure

After a time range is created, you need to create an ACL and configure the ACL rules to be associated with the time range.

7.7.6 Configuring an ACL

7.7.6.1 Configuring a Basic ACL

Context

A basic ACL defines packet filtering rules based on information such as source IPv4 addresses, fragment information, and time ranges.

To filter packets based only on source IPv4 addresses, configure a basic ACL.

When ACL rules are configured:

- If the specified rule ID already exists and the new rule conflicts with the original, the original is replaced.
- Matching stops for a packet once the packet matches a rule in an ACL.

When ACL rules are removed: The **remove** command can remove an ACL rule even if this rule is referenced. Exercise caution when performing this operation.

For details about configuration parameters, see huawei-acl.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Create a basic ACL. You can create a numbered or named basic ACL.

- Create a numbered basic ACL (its number ranges from 2000 to 2999) and enter its view.

  ```
  acl groups group identity basic-acl-number
  ```

- Create a named basic ACL and enter its view.

  ```
  acl groups group identity basic-acl-name
  ```

  You must specify a type or number for a named ACL.

  ```
  type basic                      //Set the ACL type to basic.
  number basic-acl-number         //Specify an ACL number.
  ```

Step 3 (Optional) Configure an ACL step.

```
step step-value
```

The default ACL step is 5. Change the step value as required.

NOTE

If no rule is configured in an ACL, the step can be modified. Otherwise, the step cannot be modified.

Step 4 (Optional) Configure a description for the basic ACL.

```
description text
```

The ACL description helps you understand and remember the functions or purpose of the ACL.

Step 5 Configure a basic ACL rule.

1. Create a basic ACL rule and enter its view.

rule-basics rule-basic name *rule_name*

2. Configure the action of the basic ACL rule.
   action { **permit** | **deny** }
3. (Optional) Configure the ID of the basic ACL rule.
   id *number*

   By default, the system automatically assigns a rule ID, which is greater than the maximum rule ID in the current ACL and is the smallest integer that is a multiple of the step. A smaller ID means that the rule is more preferential to take effect. Therefore, this rule is the last one that functions.
4. (Optional) Configure packet filtering based on a source IPv4 address for the basic ACL rule.
   source-ipaddr *ip-address* **source-wild** *mask*
5. (Optional) Configure a description for the basic ACL rule.
   description *text*
6. (Optional) Configure a fragment type for the basic ACL rule.
   fragment-type { **fragment-subseq** | **non-fragment** | **non-subseq** | **fragment-spe-first** }

NOTE

To configure a time-based basic ACL, you must create a time range. For details, see **7.7.5 Creating a Time Range in Which an ACL Takes Effect**.

Step 6 Commit the configuration.
commit

----End

Example

- Configure a packet filtering rule based on the source IPv4 address (host address).

  To permit packets from a host, add a rule to an ACL. For example, to permit packets from the host at 192.168.1.3, create the following rule in ACL 2001.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl groups group identity 2001
[*(gl)user@localhost]/acl/groups/group[identity="2001"]
MDCLI> rule-basics rule-basic name rule1
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> action permit source-ipaddr 192.168.1.3 source-wild 0.0.0.0
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> commit
```

- Configure a packet filtering rule based on the source IPv4 address segment.

  To permit packets from a host and reject packets from other hosts on the same network segment, configure rules in an ACL. For example, to permit packets from the host at 192.168.1.3 and reject packets from other hosts on the network segment 192.168.1.0/24, configure the following rules in ACL 2001 and configure the description "permit only 192.168.1.3 through" for the ACL.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl groups group identity 2001
[*(gl)user@localhost]/acl/groups/group[identity="2001"]
MDCLI> description "permit only 192.168.1.3 through"
```



```
[*(gl)user@localhost]/acl/groups/group[identity="2001"]
MDCLI> rule-basics rule-basic name rule1
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> action permit source-ipaddr 192.168.1.3 source-wild 0.0.0.0
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> commit
[(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> quit
[(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics
MDCLI> rule-basic name rule2
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule2"]
MDCLI> action deny source-ipaddr 192.168.1.0 source-wild 0.0.0.255
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule2"]
MDCLI> commit
```

- Configure a time range-based ACL rule.

  Create a time range **working-time** (for example, 08:00 to 18:00 on Monday through Friday). Configure a rule in the ACL **work-acl** to reject packets from the network segment 192.168.1.0/24 within the time range.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> time-range time-range-instances time-range-instance name tt1
[*(gl)user@localhost]/time-range/time-range-instances/time-range-instance[name="tt1"]
MDCLI> period-ranges period-range day-of-week "monday tuesday wednesday thursday friday" start-time 10:00 end-time 18:00
[*(gl)user@localhost]/time-range/time-range-instances/time-range-instance[name="tt1"]/period-ranges/period-range[day-of-week="monday tuesday wednesday thursday friday"][start-time="10:00"][end-time="18:00"]
MDCLI> commit
[(gl)user@localhost]/time-range/time-range-instances/time-range-instance[name="tt1"]/period-ranges/period-range[day-of-week="monday tuesday wednesday thursday friday"][start-time="10:00"][end-time="18:00"]
MDCLI> quit 5
[(gl)user@localhost]
MDCLI> acl groups group identity 2001
[(gl)user@localhost]/acl/groups/group[identity="2001"]
MDCLI> rule-basics rule-basic name rule1
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> action deny source-ipaddr 192.168.1.0 source-wild 0.0.0.255 time-range-name tt1
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> commit
```

- Configure a packet filtering rule based on the IP fragment information and source IPv4 address segment.

  For example, to reject all fragments except the first from the network segment 192.168.1.0/24, configure the following rule in ACL 2001.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl groups group identity 2001
[*(gl)user@localhost]/acl/groups/group[identity="2001"]
MDCLI> rule-basics rule-basic name rule1
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> action deny source-ipaddr 192.168.1.3 source-wild 0.0.0.255 fragment-type fragment-subseq
[*(gl)user@localhost]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="rule1"]
MDCLI> commit
```

Verifying the Configuration

Run the **display acl/groups/group**[**identity**=*identity*]**/rule-basics/rule-basic**[**name**=*rule_name*] **all** command to check the basic ACL configuration.

Follow-up Procedure

Apply the basic ACL to a service module so that the basic ACL rules can be delivered and take effect.

7.7.6.2 Configuring an Advanced ACL

Context

Advanced ACLs give more flexibility and functionality than basic ACLs, allowing you to filter packets more accurately. For example, with advanced ACLs, you can define packet filtering rules based on information such as source and destination IPv4 addresses, IP protocol types, TCP source and destination port numbers, UDP source and destination port numbers, fragment information, and time ranges.

To match multiple source and destination IPv4 addresses using an advanced ACL, configure an ACL IPv4 address pool. This helps reduce configuration workloads. After an ACL IPv4 address pool is configured, you only need to configure an ACL rule with a specified ACL IPv4 address pool name (*pool-name*) to match multiple IPv4 addresses.

To match multiple source and destination port numbers using an advanced ACL, configure an ACL port pool. This helps reduce configuration workloads. After an ACL port pool is configured, you only need to configure an ACL rule with a specified ACL port pool name (*pool-name*) to match multiple port numbers.

When ACL rules are configured:

- If the specified rule name already exists and the new rule conflicts with the original, the original is replaced.
- Matching stops for a packet once the packet matches a rule in an ACL.

When ACL rules are removed: The **remove** [**rule-advances** | **rule-advance**] *rule_name* command can remove an ACL rule even if this rule is referenced. Exercise caution when performing this operation.

For details about configuration parameters, see huawei-acl.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Create a numbered or named advanced ACL.

- Create a numbered advanced ACL (its number ranges from 3000 to 3999) and enter its view.

  ```
  acl groups group identity advance-acl-number
  ```

- Create a named advanced ACL and enter its view.

  ```
  acl groups group identity advance-acl-name
  ```

  You must specify a type or number for a named ACL.

  ```
  type advance                          //Set the ACL type to advanced.
  number advance-acl-number             //Specify an ACL number.
  ```

Step 3 (Optional) Configure an ACL step.

```
step step-value
```


The default ACL step is 5. Change the step value as required.

NOTE

If no rule is configured in an ACL, the step can be modified. Otherwise, the step cannot be modified.

Step 4 (Optional) Configure a description for the ACL.

description *text*

The ACL description helps you understand and remember the functions or purpose of the ACL.

Step 5 Enter the advanced ACL view using either of the following methods:

acl groups group identity *advance-acl-number*
acl groups group identity *advance-acl-name*
type advance

Step 6 Configure an advanced ACL rule.

1. Create an advanced ACL rule and enter its view.
   rule-advances rule-advance name *rule-name*
2. Configure the action of the advanced ACL rule.
   action { **permit** | **deny** }
3. Configure fields for the advanced ACL rule as required.
   - Configure an ID for the advanced ACL rule.
     id *number*
   - Configure a source IP address for the advanced ACL rule.
     source-ipaddr *ip-address*
     source-wild *mask*
   - Configure a destination IP address for the advanced ACL rule.
     dest-ipaddr *ip-address*
     dest-wild *mask*
   - Use either of the following methods to configure a protocol number for the advanced ACL rule.
     protocol *number*
     protocol-zero [null]
   - Configure a fragment type for the advanced ACL rule.
     fragment-type { **fragment-subseq** | **non-fragment** | **non-subseq** | **fragment-spe-first** }
   - Configure a time range for the advanced ACL rule.
     time-range-name *time_range_name*
   - Configure an IP precedence for the advanced ACL rule.
     precedence *number*
   - Configure a ToS value for the advanced ACL rule.
     tos *number*
   - Configure a DSCP priority for the advanced ACL rule.
     dscp *number*
   - Configure a TCP flag for the advanced ACL rule.
     tcp-flag-value *value*
   - Configure a source port range for the advanced ACL rule.
     source-port-begin *port_number* **source-port-end** *port_number*
   - Configure a destination port range for the advanced ACL rule.
     dest-port-begin *port_number* **dest-port-end** *port_number*
   - Configure an IGMP packet type for the advanced ACL rule.
     igmp-type *number*

- Configure an ICMP packet type for the advanced ACL rule.
  icmp-type *number*
- Configure an ICMP message code for the advanced ACL rule.
  icmp-code *number*
- Configure TTL information for the advanced ACL rule.
  ttl-expired { **true** | **false** }
- Configure a description for the advanced ACL rule.
  description *text*

In this example, only one permit or deny rule is configured. In practice, you can configure multiple ACL rules and decide the matching order of the rules according to service requirements.

NOTE

To configure a time-based advanced ACL, you must create a time range. For details, see **7.7.5 Creating a Time Range in Which an ACL Takes Effect**.

When a rule is configured for an advanced ACL:

- If a destination IP address, destination port number, source IP address, and source port number are specified, the system filters only packets with the specified destination IP address, destination port number, source IP address, and source port number.
- If **time-range** is specified, the specified time range name must exist. Otherwise, the ACL rule configuration fails.
- DSCP value and ToS value configurations are mutually exclusive. If you configure both, only the one configured last takes effect and it overrides the other one.

Step 7 Commit the configuration.

commit

----End

Example

- Configure a filtering rule for ICMP packets based on the source IPv4 address (host address) and destination IPv4 address segment.

  To permit ICMP packets from a host and destined for a network segment, configure a rule in an ACL. For example, to permit ICMP packets from the host at 192.168.1.3 and destined for the network segment 192.168.2.0/24, configure the following rule in ACL 3001.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl groups group identity 3001
[*(gl)user@localhost]/acl/groups/group[identity="3001"]
MDCLI> rule-advances rule-advance name rule1
[*(gl)user@localhost]/acl/groups/group[identity="3001"]/rule-advances/rule-advance[name="rule1"]
MDCLI> source-ipaddr 192.168.1.3 source-wild 0.0.0.0
[*(gl)user@localhost]/acl/groups/group[identity="3001"]/rule-advances/rule-advance[name="rule1"]
MDCLI> dest-ipaddr 192.168.2.0 dest-wild 0.0.0.255
[*(gl)user@localhost]/acl/groups/group[identity="3001"]/rule-advances/rule-advance[name="rule1"]
MDCLI> protocol 1
[*(gl)user@localhost]/acl/groups/group[identity="3001"]/rule-advances/rule-advance[name="rule1"]
MDCLI> action permit
[*(gl)user@localhost]/acl/groups/group[identity="3001"]/rule-advances/rule-advance[name="rule1"]
MDCLI> commit
```

- Configure a filtering rule for TCP packets based on the TCP destination port number, source IPv4 address (host address), and destination IPv4 address segment.

To prohibit Telnet connections between a specified host and the hosts on a network segment, configure a rule in an advanced ACL. For example, to prohibit Telnet connections between the host at 192.168.1.3 and hosts on the network segment 192.168.2.0/24, configure the following rule in the advanced ACL **deny-telnet**.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl groups group identity deny-telnet
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]
MDCLI> type advance
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]
MDCLI> rule-advances rule-advance name rule1
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]/rule-advances/rule-advance[name="rule1"]
MDCLI> source-ipaddr 192.168.1.3 source-wild 0.0.0.0
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]/rule-advances/rule-advance[name="rule1"]
MDCLI> dest-ipaddr 192.168.2.0 dest-wild 0.0.0.255
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]/rule-advances/rule-advance[name="rule1"]
MDCLI> protocol 6
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]/rule-advances/rule-advance[name="rule1"]
MDCLI> dest-port-begin 23 dest-port-end 23
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]/rule-advances/rule-advance[name="rule1"]
MDCLI> action deny
[*(gl)user@localhost]/acl/groups/group[identity="deny-telnet"]/rule-advances/rule-advance[name="rule1"]
MDCLI> commit
```

- To prohibit the specified hosts from accessing web pages (HTTP is used to access web pages, and the TCP port number is 80), configure rules in an advanced ACL. For example, to prohibit hosts at 192.168.1.3 and 192.168.1.4 from accessing web pages, configure the following rules in ACL **no-web** and configure the description "Web access restrictions" for the ACL.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl
[(gl)user@localhost]/acl
MDCLI> groups group identity no-web
[*(gl)user@localhost]/acl/groups/group[identity="no-web"]
MDCLI> type advance
[(gl)user@localhost]/acl/groups/group[identity="no-web"]
MDCLI> description "Web access restrictions"
[*(gl)user@localhost]/acl/groups/group[identity="no-web"]
MDCLI> rule-advances rule-advance name rule1
[*(gl)user@localhost]/acl/groups/group[identity="no-web"]/rule-advances/rule-advance[name="rule1"]
MDCLI> dest-port-begin 80 dest-port-end 80
[*(gl)user@localhost]/acl/groups/group[identity="no-web"]/rule-advances/rule-advance[name="rule1"]
MDCLI> protocol 6
[*(gl)user@localhost]/acl/groups/group[identity="no-web"]/rule-advances/rule-advance[name="rule1"]
MDCLI> action deny
[*(gl)user@localhost]/acl/groups/group[identity="no-web"]/rule-advances/rule-advance[name="rule1"]
MDCLI> commit
```

- Configure a packet filtering rule for TCP packets based on the source IPv4 address segment and TCP flags.

  To implement unidirectional access control on a network segment, configure rules in an ACL. For example, to implement unidirectional access control on

the network segment 192.168.2.0/24, configure the following rules in ACL 3002. In the rules, the hosts on 192.168.2.0/24 can only respond to TCP handshake packets, but cannot send the packets. Set the descriptions of the ACL rules to "Allow the ACK TCP packets through," "Allow the RST TCP packets through," and "Do not Allow the other TCP packet through."

Configure two permit rules to permit the packets with the ACK or RST field being 1 from 192.168.2.0/24, and then configure a deny rule to reject other TCP packets from this network segment.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> acl groups group identity 3002
[*(gl)user@localhost]/acl/groups/group[identity="3002"]
MDCLI> rule-advances rule-advance name rule1
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule1"]
MDCLI> description "Allow the ACK TCP packets through"
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule1"]
MDCLI> action permit source-ipaddr 192.168.2.0 source-wild 0.0.0.255 protocol 6 tcp-flag-value
16
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule1"]
MDCLI> commit
[(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule1"]
MDCLI> quit
[(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances
MDCLI> rule-advance name rule2
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule2"]
MDCLI> description "Allow the RST TCP packets through"
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule2"]
MDCLI> action permit source-ipaddr 192.168.2.0 source-wild 0.0.0.255 protocol 6
tcp-flag-value 4
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule2"]
MDCLI> commit
[(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule2"]
MDCLI> quit
[(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances
MDCLI> rule-advance name rule3
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule3"]
MDCLI> description "Do not Allow the other TCP packet through"
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule3"]
MDCLI> action deny source-ipaddr 192.168.2.0 source-wild 0.0.0.255 protocol 6
[*(gl)user@localhost]/acl/groups/group[identity="3002"]/rule-advances/rule-advance[name="rule3"]
MDCLI> commit
```

Verifying the Configuration

Run the **display acl/groups/group**[**identity**=*acl_name*]**/rule-advances/rule-advance**[**name**=*rule_name*] **all** command to check the advanced ACL configuration.

Follow-up Procedure

Apply the advanced ACL to a service module so that the advanced ACL rules can be delivered and take effect.

7.7.6.3 Configuring a Layer 2 ACL

Context

A Layer 2 ACL defines packet filtering rules based on the information in Ethernet frame headers of the packets, such as source and destination MAC addresses, VLAN IDs, and Layer 2 protocol types.

When ACL rules are configured:

- If the specified rule ID already exists and the new rule conflicts with the original, the original is replaced.
- Matching stops for a packet once the packet matches a rule in an ACL.

When ACL rules are removed: The **remove** [**rule-ethernets** | **rule-ethernet**] *rule_name* command can remove an ACL rule even if this rule is referenced.

For details about configuration parameters, see huawei-acl.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Create a Layer 2 ACL. You can create a numbered or named Layer 2 ACL.

- Create a numbered Layer 2 ACL (its number ranges from 4000 to 4999) and enter its view.

  ```
  acl groups group identity ethernet-acl-number
  ```

- Create a named Layer 2 ACL and enter its view.

  ```
  acl groups group identity ethernet-acl-name
  ```

  You must specify a type or number for a named ACL.

  ```
  type link                          //Set the ACL type to Layer 2.
  number ethernet-acl-number                 //Specify an ACL number.
  ```

Step 3 (Optional) Configure an ACL step.

```
step step-value
```

The default ACL step is 5. Change the step value as required.

NOTE

If no rule is configured in an ACL, the step can be modified. Otherwise, the step cannot be modified.

Step 4 (Optional) Configure a description for the ACL.

```
description text
```

The ACL description helps you understand and remember the functions or purpose of the ACL.

Step 5 Configure a rule for the Layer 2 ACL.

1. Create a Layer 2 ACL rule and enter its view.

   ```
   rule-ethernets rule-ethernet name rule-name
   ```

2. Configure the action of the Layer 2 ACL rule.

   ```
   action { permit | deny }
   ```

3. Configure fields for the Layer 2 ACL rule as required.
   - Configure an ID for the Layer 2 ACL rule.

     ```
     id number
     ```

   - Configure frame information for the Layer 2 ACL rule.

     ```
     frame-type type 0x0800
     frame-mask mask 0xffff
     ```

   - Configure a source MAC address for the Layer 2 ACL rule.

     ```
     source-mac mac-address
     source-mac-mask mask
     ```


- Configure a destination MAC address for the Layer 2 ACL rule.
  dest-mac *mac-address*
  dest-mac-mask *mask*
- Configure a VLAN for the Layer 2 ACL rule.
  vlan-id *id*
  vlan-id-mask *mask*
- Configure an 802.1p priority for the Layer 2 ACL rule.
  value-8021p *number*
- Configure a description for the Layer 2 ACL rule.
  description *text*
- Configure a time range for the Layer 2 ACL rule.
  time-range-name *time_range_name*

Step 6 Commit the configuration.

commit

----End

Example

- Configure packet filtering rules based on the source MAC address, destination MAC address, and Layer 2 protocol type.

  To permit ARP packets with the specified destination and source MAC addresses and Layer 2 protocol type, configure a rule in a Layer 2 ACL. For example, to permit ARP packets with the destination MAC address of 00e0-fc00-0001, source MAC address of 00e0-fc00-0002, and Layer 2 protocol type of 0x0806, configure the following rule in ACL 4001.

  ```
  [user@HUAWEI]
  MDCLI> edit-config
  [(gl)user@HUAWEI]
  MDCLI> acl groups group identity 4001
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]
  MDCLI> rule-ethernets rule-ethernet name r1
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r1"]
  MDCLI> action permit
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r1"]
  MDCLI> dest-mac 00e0-fc00-0001 source-mac 00e0-fc00-0002
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r1"]
  MDCLI> frame-type 0x0806
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r1"]
  MDCLI> vlan-id 1
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r1"]
  MDCLI> commit
  ```

  To reject packets with a specified Layer 2 protocol type, configure a rule in a Layer 2 ACL. For example, to reject packets with the Layer 2 protocol type of 0x8863, configure the following rule in ACL 4001.

  ```
  [user@HUAWEI]
  MDCLI> edit-config
  [(gl)user@HUAWEI]
  MDCLI> acl groups group identity  4001
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]
  MDCLI> rule-ethernets rule-ethernet name r2
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r2"]
  MDCLI> action deny
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r2"]
  MDCLI> frame-type 0x8863
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r2"]
  MDCLI> vlan-id 1
  [*(gl)user@HUAWEI]/acl/groups/group[identity="4001"]/rule-ethernets/rule-ethernet[name="r2"]
  MDCLI> commit
  ```


- Configure a packet filtering rule based on the source MAC address segment and VLAN ID.

  To reject packets from a specified MAC address segment in a VLAN, configure a rule in a Layer 2 ACL. For example, to reject packets from the source MAC address segment 00e0-fc01-0000 to 00e0-fc01-ffff in VLAN 10, configure the following rule in the Layer 2 ACL **deny-vlan10-mac**.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> acl groups group identity deny-valn10-mac
[*(gl)user@HUAWEI]/acl/groups/group[identity="deny-valn10-mac"]
MDCLI> type link
[*(gl)user@HUAWEI]/acl/groups/group[identity="deny-valn10-mac"]
MDCLI> rule-ethernets rule-ethernet name r1
[*(gl)user@HUAWEI]/acl/groups/group[identity="deny-valn10-mac"]/rule-ethernets/rule-ethernet[name="r1"]
MDCLI> action deny
[*(gl)user@HUAWEI]/acl/groups/group[identity="deny-valn10-mac"]/rule-ethernets/rule-ethernet[name="r1"]
MDCLI> vlan-id 10
[*(gl)user@HUAWEI]/acl/groups/group[identity="deny-valn10-mac"]/rule-ethernets/rule-ethernet[name="r1"]
MDCLI> source-mac 00e0-fc01-0000 source-mac-mask ffff-ffff-0000
[*(gl)user@HUAWEI]/acl/groups/group[identity="deny-valn10-mac"]/rule-ethernets/rule-ethernet[name="r1"]
MDCLI> commit
```

Verifying the Configuration

Run the **display acl/groups/group**[**identity**=*acl_name*]**/rule-ethernets/rule-ethernet**[**name**=*rule_name*] **all** command to check the Layer 2 ACL configuration.

Follow-up Procedure

Apply the Layer 2 ACL to a service module so that the Layer 2 ACL rules can be delivered and take effect.

7.7.7 Applying an ACL

Context

After an ACL is configured, it must be applied to a service module so that the ACL rules can be delivered and take effect. Typically, an ACL is applied to an SACL to filter packets to be forwarded. For details about configuration parameters, see huawei-sacl.yang.

NOTE

ACL can be applied to many features, and the devices with these features process the classified packets according to specific service requirements. For details about the features, see the configuration guide of each feature. For details on how to configure an ACL-based simplified traffic policy, see **12.1 ACL-based Simplified Traffic Policy Configuration**.

7.7.8 Modifying an ACL

Context

You can add and delete ACL rules as required. However, it is recommended not to directly modify existing rules. This is because modified rules may conflict with and

replace other rules, leading to unexpected results. In this manner, the ACL rules may fail to achieve the expected effect.

For details about configuration parameters, see huawei-acl.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Run either of the following commands to enter the view of an ACL to be modified:

```
acl groups group identity acl-number
acl groups group identity acl-name
```

Step 3 Configure new ACL rules and delete unused ACL rules.

NOTE

If an existing rule is edited and the edited content conflicts with the original one, the edited content takes effect.

Before updating ACL rules, the device deletes all existing ones. To reduce the pressure on the system caused by frequent configurations, the device caches user changes for a short time and periodically updates and delivers configurations in batches.

Step 4 Commit the configuration.

```
commit
```

----End

7.7.9 Deleting an ACL.

Context

When the usage of ACL resources on a device reaches the maximum, you can deploy a new ACL only after deleting unnecessary ACL configurations.

For details about configuration parameters, see huawei-acl.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Delete an ACL using either of the following methods:

- Delete a numbered ACL.

```
acl groups group identity acl-number    //Enter the view of a numbered ACL.
remove rule-advances/                   //Delete all rules from the ACL.
acl groups group identity
remove group[identity=acl-number]       //Delete the ACL.
```

- Delete a named ACL.

```
acl groups group identity acl-name    //Enter the view of a named ACL.
remove rule-advances/                 //Delete all rules from the ACL.
acl groups group identity
remove group[identity=acl-name]       //Delete the ACL.
```

NOTE

Before deleting an ACL, ensure that it is not referenced by any service module.

Step 3 Commit the configuration.

```
commit
```

----End

8 IP Routing Configuration

8.1 Route Management Configuration

8.1.1 Overview of Route Management

Definition

Route management refers to the capability of a routing device to establish and refresh a routing table, and forward data packets over routes contained in the routing table.

Purpose

Route management helps users realize that routing devices, routing tables, and routing protocols are indispensable for data forwarding. Route management also helps users gain a preliminary understanding of routing devices and routing tables. Routing protocols are used to discover routes and contribute to routing entry generation, routing tables store routes discovered by various routing protocols, and routing devices select routes as well as implement data forwarding.

8.1.2 Understanding Route Management

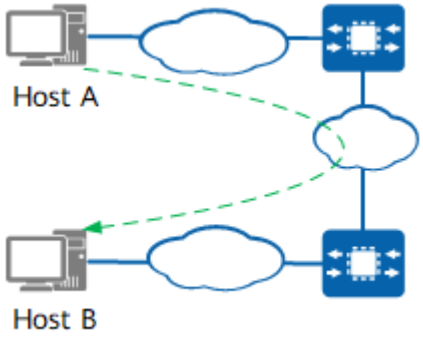
8.1.2.1 Routing Device

Routing devices select routes and forward packets over the routes. In this process, a routing device selects a proper path to a specified destination address carried in a received packet and sends the packet to a next routing device. The routing device at an endpoint of a path sends the packet to a destination host. In addition, the routing devices can select optimal paths for data transmission.

For example, in **Figure 8-1**, traffic from host A to host B passes through three network segments and two routing devices. The number of hops between a

routing device and its directly connected network segment is zero, and that between the routing device and another network segment through the other routing device is one, and so forth. If two routing devices are connected through a network segment, they are considered to be adjacent on the Internet.

Figure 8-1 Network segment and hop count



8.1.2.2 Routing Protocols

Route selection and packet forwarding are the main functions of a routing device. To implement the two functions, routing protocols are required. Routing protocols are necessary to implement these two functions, and they are rules used by routing devices to discover and add routes, as well as maintain routing tables for packet forwarding.

Differences Between Static and Dynamic Routes

Routes are classified into the following types according to their origins:

- Direct route: discovered by a data link layer protocol
- Static route: manually configured
- Dynamic route: discovered by a dynamic routing protocol

  Dynamic and static routes have their own advantages and disadvantages. You can determine whether to use static or dynamic routes based on real-world situations. **Table 8-1** describes the advantages and disadvantages of static and dynamic routes.

Table 8-1 Advantages and disadvantages of static and dynamic routes

Route Type	Advantage	Disadvantage
Static route	Poses low system requirements and is applicable to small-scale networks with simple and stable topologies.	Static routes cannot dynamically adapt to network topology changes; therefore, manual intervention is required.

Route Type	Advantage	Disadvantage
Dynamic route	Has its own routing algorithm and can automatically adapt to network topology changes. Dynamic routes are applicable to networks with a large number of Layer 3 devices.	In addition to consuming network and system resources, dynamic routes have complex configurations, and they pose higher system requirements than static routes.

Classification of Dynamic Routing Protocols

Dynamic routing protocols are classified based on application ranges and algorithms.

Based on the application range, dynamic routing protocols are classified into the following types:

- Interior Gateway Protocols (IGPs): run inside an autonomous system (AS), including the Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Intermediate System to Intermediate System (IS-IS).
- Exterior Gateway Protocols (EGPs): run between ASs, including the Border Gateway Protocol (BGP).

Based on the type of algorithm used, dynamic routing protocols are classified into the following types:

- Distance-vector routing protocols: include RIP and BGP. BGP is also called a path-vector protocol.
- Link-state routing protocols: include OSPF and IS-IS.

The preceding algorithms mainly differ in route discovery and calculation methods.

8.1.2.3 Routing Tables

A routing device searches a routing table for routes, and each routing device maintains at least one routing table.

Routing tables store routes related to various routing protocols. Based on the generation method, the routes in a routing table consist of the following types:

- Direct route
- Static route
- Dynamic route

Routing Table Types

Each routing device maintains a global routing table, and each routing protocol maintains its own routing table.

- Protocol routing table: stores routing information related to a specific protocol.

  A routing protocol can import and advertise routes generated by other routing protocols. For example, OSPF can import other types of routes, such as direct routes, static routes, and IS-IS routes, into the OSPF routing table, and then advertise them to guide packet forwarding.
- Global routing table: stores protocol routes and preferred routes. The routes in the global routing table are selected based on preferences of routing protocols and costs of routes. To check the global routing table, run the **display network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/routing-manage/topologys/topology[name="base"]/routes/ipv4-unicast-routes/ipv4-unicast-route all** command.

Routing Table Content

To check brief information about the routing table, run the **display network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/routing-manage/topologys/topology[name="base"]/routes/ipv4-unicast-routes/ipv4-unicast-route all** command.

```
[user@HUAWEI]
MDCLI> display network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/routing-manage/topologys/topology[name="base"]/routes/ipv4-unicast-routes/ipv4-unicast-route all
[
  {
    "prefix": "127.0.0.1",
    "mask-length": 32,
    "protocol-type": "direct",
    "interface-name": "lo",
    "process-id": 0,
    "direct-nexthop": "127.0.0.1",
    "indirect-id": "0x1",
    "preference": 0,
    "cost": 0,
    "flag": "d",
    "active": true,
    "state": "active-noadv"
  }
]
```

Table 8-2 describes the keywords in the routing table.

Table 8-2 Key items in the routing table

Item	Meaning	Function
prefix	Destination address	Identifies the destination address or destination network segment of IP packets.

Item	Meaning	Function
mask-length	Network mask length	Is used together with a destination address to identify the address of a network segment where a destination host or routing device resides. • The address of the network segment where the destination host or routing device resides can be calculated after the destination IP address and network mask are ANDed. For example, if a destination address is 10.1.1.1 and a mask is 255.255.255.0, the address of the network segment where the destination host or routing device resides is 10.1.1.0. • A mask consists of several consecutive 1s, which can be expressed in dotted decimal notation or the number of consecutive 1s in the mask. For example, the length of the mask 255.255.255.0 is 24, and therefore, it can also be expressed as 24.
protocol-type	Routing protocol	• Static • OSPF • IS-IS • direct • BGP
preference	Preference value of a route that is added to the IP routing table	There may be multiple routes to the same destination, which have different next hops and outbound interfaces. These routes can be discovered by different routing protocols or manually configured. A routing device selects the route with the highest preference (indicated by the smallest value) as the optimal route. For the route preference value of each routing protocol, see **Table 8-3**.
cost	Route cost	When multiple routes to the same destination have the same preference, the route with the smallest cost is selected as the optimal route. **NOTE** Preference is used to compare the priorities of routes of different routing protocols, and Cost is used to compare the priorities of routes of the same routing protocol. When selecting routes in the routing table, a routing device compares their preference values first. If their preference values are the same, the routing device compares their cost values.

Item	Meaning	Function
flag	Flag of a route	• R: stands for "relay", indicating a recursive route. • D: stands for "download to fib", indicating a route delivered to the FIB. • T: stands for "to vpn-instance", indicating a route with a VPN instance as a next hop. • B: stands for a blackhole route.
direct-nexthop	Next-hop IP address	Identifies an interface on a next routing device through which an IP packet passes.
interface-name	Outbound interface name	Identifies a local interface through which an IP packet is forwarded.

According to the destination, routes can be divided into the following types:

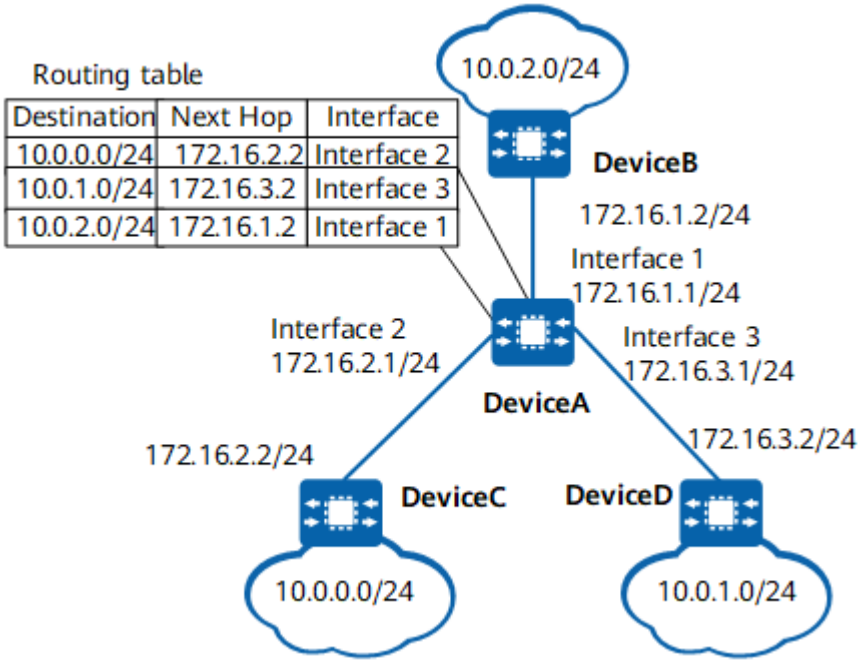
- Network segment route: is destined for a network segment.
- Host route: is destined for a host.

In addition, depending on whether the destination is directly connected to a local routing device, routes fall into the following types:

- Direct route: The routing device is directly connected to the destination network.
- Indirect route: The routing device is indirectly connected to the destination network.

As shown in **Figure 8-2**, DeviceA is connected to three network segments, and therefore, it has three destination IP addresses and three outbound interfaces. **Figure 8-2** shows the routing table of DeviceA.

Figure 8-2 Routing table



8.1.2.4 Route Preference

Routing protocols (including static route) may discover different routes to the same destination, but not all routes are optimal. Only one routing protocol is used each time to determine the optimal route to a destination. To help select the optimal route, routing protocols, including static route, are assigned preferences, and the route with the highest preference becomes the optimal route. **Table 8-3** lists routing protocols and their default preference values.

In **Table 8-3**, value 0 indicates a direct route, and value 255 indicates any route learned from an unreliable source. A smaller value indicates a higher preference.

Table 8-3 Routing protocols and their default preference values

Routing Protocol or Route Type	Route Preference Value
Direct	0
OSPF	10
IS-IS	15
Static	60
RIP	100
OSPF ASE	150
OSPF NSSA	150
IBGP	255

Routing Protocol or Route Type	Route Preference Value
EBGP	255

Preference values of various routing protocols can be manually set. In addition, the preference of each static route can be different.

External and internal preferences are used. The external preference refers to the preference set by users for each routing protocol. **Table 8-3** lists the default external preference values.

When various routing protocols are assigned the same preference, the system selects the optimal route based on the internal preference. Internal preferences of routing protocols cannot be manually modified. **Table 8-4** lists the internal preferences of routing protocols.

Table 8-4 Internal preference values of routing protocols

Routing Protocol or Route Type	Route Preference Value
Direct	0
OSPFv2 inter-area	10
OSPFv3 inter-area	10
IS-IS Level-1	15
IS-IS Level-2	18
EBGP	20
Static	60
RIP	100
RIPng	100
OSPFv2 ASE	150
OSPFv3 ASE	150
OSPFv2 NSSA	150
OSPFv3 NSSA	150
IBGP	200

For example, both an OSPF route and a static route are destined for 10.1.1.0/24, and their protocol preference values are set to 5. In this case, a device determines the optimal route according to internal preferences listed in **Table 8-4**. The

internal preference of OSPF (10) is higher than that of the static route (60). Therefore, the device selects the route discovered by OSPF as the optimal route.

NOTE

- If multiple OSPFv2/OSPFv3 processes learn routes to the same destination and the external and internal preferences of the routes are the same, the device selects the route with the smallest link cost; if the link costs of the routes are the same, the routes perform load balancing.
- If multiple IS-IS processes learn routes to the same destination and the external and internal preferences of the routes are the same, the device selects the route with the smallest link cost; if the link costs of the routes are the same, the routes perform load balancing.
- If multiple RIP/RIPng processes learn routes to the same destination and the external and internal preferences of the routes are the same, the device selects the route with the smallest link cost; if the link costs of the routes are the same, the routes perform load balancing.

8.1.2.5 Priority-based Route Convergence

Definition

To improve network reliability, priority-based route convergence is essential. It provides faster route convergence for key services. In priority-based convergence, different convergence priorities are set for routes on a device, and the device converges routes based on priorities at a specified scheduling ratio to ensure uninterrupted service forwarding.

Purpose

With the network convergence, requirements on service differentiation increase. Carriers require that routes for key services, such as voice over IP (VoIP) and video conferencing services, converge faster than those for common services. In this case, route convergence needs to be performed based on convergence priorities to improve network reliability.

Route Convergence Priority

Route convergence priorities are critical, high, medium, and low. Critical routes have the highest convergence priority, and low routes have the lowest convergence priority. **Table 8-5** lists the default convergence priorities of public network routes. You can set convergence priorities for routes as needed for a specific topology.

Table 8-5 Default convergence priorities of public network routes

Routing Protocol or Route Type	Convergence Priority
Direct	Critical
Static	Medium
OSPF and IS-IS host routes with 32-bit masks	Medium

Routing Protocol or Route Type	Convergence Priority
OSPF (except 32-bit host routes)	Low
IS-IS (except 32-bit host routes)	Low
RIP	Low
BGP	Low

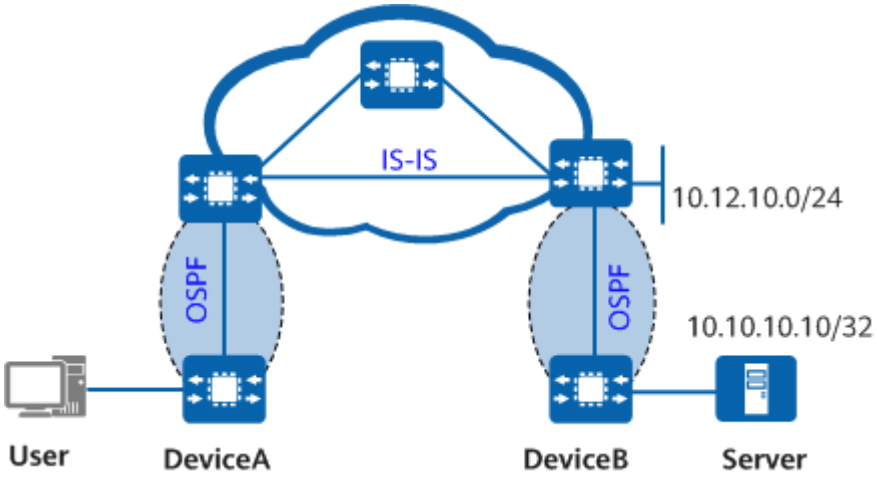
NOTE

For private network routes, only convergence priorities of OSPF and IS-IS host routes with 32-bit masks are medium, and those of the other routes are low.

Typical Application

An IGP runs on the network shown in **Figure 8-3**. DeviceA functions as the receiver, and DeviceB is connected to a server with IP address 10.10.10.10/32. The route to the server must be converged faster than other routes, for example, a route to 10.12.10.0/24. You can set a higher convergence priority for the route to 10.10.10.10/32 than that for the route to 10.12.10.0/24. In this case, the route to the server 10.10.10.10/32 is converged first, which ensures the proper transmission of key services.

Figure 8-3 Network diagram of priority-based route convergence



8.1.2.6 Route Recursion

Routes can be used to forward traffic only when they have directly connected next hops. However, a generated route may contain an indirect next hop. Therefore, in a process referred to as route recursion, a device needs to search for a directly connected next hop and matching outbound interface for the route.

For example, the next-hop IP address of a BGP route is often the IP address of an indirectly connected peer's loopback interface. In this case, this route cannot be

used to forward traffic unless route recursion is performed. During route recursion, the device searches its IP routing table for a recursive route (an IGP route in most cases) with a directly connected next hop and outbound interface based on the next-hop IP address of the BGP route. The device then adds the recursive route's next-hop IP address and outbound interface to the IP routing table and generates a forwarding entry.

8.1.2.7 Default Route

Default routes are special routes. Generally, administrators can manually configure default static routes. Default routes can also be generated through dynamic routing protocols, such as OSPF and IS-IS.

Default routes are used when no matching routing entry is found in the routing table. In a routing table, a default route is a route with the network address and mask of all 0s.

If the destination address of a packet does not match any entry in the routing table, the packet is sent over a default route. If no default route exists and the destination address of the packet does not match any entry in the routing table, the packet is discarded. An Internet Control Message Protocol (ICMP) message is then sent, informing the source that the destination host or network is unreachable.

8.1.3 Configuration Precautions for Route Management

Licensing Requirements

Route Management is not under license control.

Hardware Requirements

Table 8-6 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

8.2 IPv4 Static Route Configuration

8.2.1 Overview of IPv4 Static Routes

Definition

A static route is a manually configured route that allows network traffic to reach a specified destination.

Purpose

Static routes alone can implement interworking for simple networks. Static routes can also be used if devices cannot use dynamic routing protocols or do not have available routes to destination networks.

Static routes can accurately control route selection on a network. Properly configuring and using static routes can improve network performance and guarantee the required bandwidth for important applications.

8.2.2 Understanding IPv4 Static Routes

A routing device forwards data packets over routes. The routes can be manually configured or automatically calculated using a dynamic routing algorithm. Static routes are manually configured.

Compared with using dynamic routes, using static routes consumes less bandwidth and does not consume device resources to calculate, analyze, and update routes.

Compared with dynamic routes, static routes have the disadvantage that if a network fault occurs or the topology changes, static routes can only be manually adjusted but cannot automatically change.

A static route may contain the destination address, mask length, outbound interface name, and next-hop address.

Destination Address and Mask

An IPv4 destination address in a static route is expressed in dotted decimal notation, while a mask can be expressed either in dotted decimal or CIDR notation.

Outbound Interface and Next-Hop IP Address

When configuring a static route, you can specify an outbound interface only, a next-hop IP address only, or both. Actually, a next-hop IP address must be explicitly contained in each route. Before sending a packet, a device searches its routing table for a route matching the destination IP address in the packet by following the longest match rule. Rules for specifying an outbound interface are as follows:

- When a P2P interface is specified as an outbound interface, this operation also implicitly specifies a next-hop IP address. This is because the IP address of the interface directly connected to the outbound interface is used as the next-hop IP address.
- Non-Broadcast Multiple-Access (NBMA) interfaces apply to point-to-multipoint (P2MP) networks. In addition to static IP routes, mappings

between IP and MAC addresses must be configured. In this case, next-hop IP addresses must be specified.

- When configuring a static route, you are recommended not to specify a broadcast interface (for example, an Ethernet interface) as an outbound interface. Because broadcasting involves multiple next hops, using such an outbound interface leads to difficulty in determining a correct next hop. In applications, if a broadcast interface (for example, an Ethernet interface) must be used as an outbound interface, a next-hop IP address must also be specified.

8.2.3 Configuration Precautions for IPv4 Static Route

Licensing Requirements

IPv4 Static Route is not under license control.

Hardware Requirements

Table 8-7 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

8.2.4 Default Settings for IPv4 Static Routes

Table 8-8 describes the default settings for IPv4 static routes.

Table 8-8 Default settings for IPv4 static routes

Parameter	Default Setting
IPv4 static route	No IPv4 static route is configured.
Default preference value of IPv4 static routes	60
Default cost of IPv4 static routes	0
Whether an IPv4 static route inherits the cost of the recursive route	No

8.2.5 Configuring an IPv4 Static Route

8.2.5.1 Creating an IPv4 Static Route

Prerequisites

Before creating an IPv4 static route, you have completed the following task:

- Set link layer protocol parameters for interfaces to ensure that the link layer protocol status of the interfaces is up.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Configure an IPv4 static route.

- Configure an IPv4 static route on the public network.
  a. Enter the public network view.
  ```
  network-instance instances instance name _public_
  ```
  b. Enter the IPv4 unicast address family view.
  ```
  afs af type ipv4-unicast
  ```
  c. Configure a route prefix.
  ```
  routing static-routing unicast-route2s unicast-route2 topology-name base prefix ip-address mask-length mask-length
  ```
  d. Configure a next-hop outbound interface or address. Select one of the following methods:
     - Configure a next-hop outbound interface for the route.
     ```
     nexthop-interfaces nexthop-interface interface-name interface-name
     ```
     (Optional) Configure a route preference and cost.
     ```
     preference preference cost cost
     ```
     - Configure a next-hop outbound interface and address for the route.
     ```
     nexthop-interface-addresses nexthop-interface-address interface-name interface-name address nexthop-address
     ```
     (Optional) Configure a route preference and cost.
     ```
     preference preference cost cost
     ```
     - Configure a next-hop address for the route.
     ```
     nexthop-addresses nexthop-address address nexthop-address
     ```
     (Optional) Configure a route preference and determine whether the route inherits the cost.
     ```
     preference preference [ inherit-cost { true | false } ]
     ```

You can set different preference values for different static routes to facilitate the flexible application of route management policies.

Setting tag values for static routes classifies these routes, helping a device implement different route management policies. For example, a routing protocol can import static routes with a specified tag value using a routing policy.

NOTE

If you configure a broadcast or NBMA interface as an outbound interface when configuring a static route, you must specify a next-hop address as well.

Step 3 Commit the configuration.

```
commit
```

----End

8.2.5.2 Example for Configuring IPv4 Static Routes

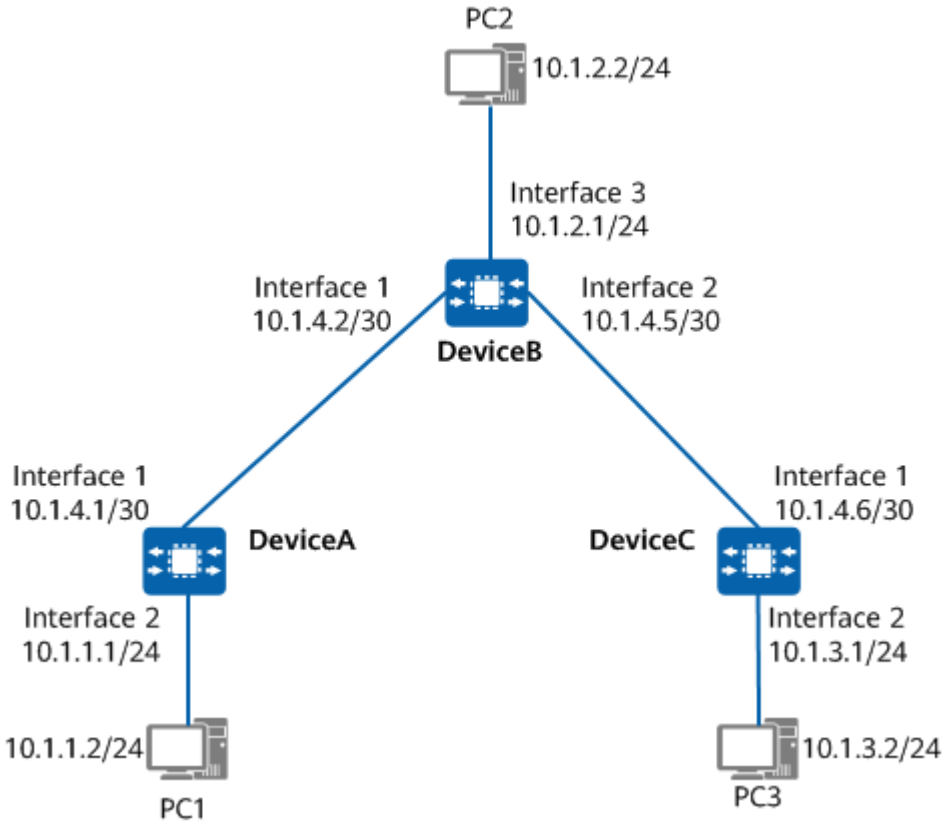
Networking Requirements

On the network shown in **Figure 8-4**, static routes need to be configured on DeviceA, DeviceB, and DeviceC to enable any two hosts to communicate.

Figure 8-4 Configuring IPv4 static routes

NOTE

In this example, interface1, interface2, and interface3 represent GE0/0/1, GE0/0/2, and GE0/0/3, respectively.



Precautions

When configuring IPv4 static routes, note the following:

- If a broadcast interface is used as the outbound interface of an IPv4 static route, a next-hop IP address must be specified.

Configuration Roadmap

The configuration roadmap is as follows:

1. Configure IPv4 addresses for interfaces on each device.
2. Configure IPv4 static routes destined for specified destination addresses or a default IPv4 static route on each device.

Procedure

Step 1 Configure IPv4 addresses for interfaces on each device.

Configure DeviceA.

```
[user@localhost]
MDCLI> edit-config
[(gl)user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> ipv4 addresses address ip 10.1.4.1
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/ipv4/addresses/address[ip="10.1.4.1"]
MDCLI> mask 255.255.255.252 type main
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/ipv4/addresses/address[ip="10.1.4.1"]
MDCLI> commit
[(gl)user@localhost]
MDCLI> quit
[(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> quit
[(gl)user@localhost]/ifm/interfaces
MDCLI> interface name GE0/0/2
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/2"]
MDCLI> ipv4 addresses address ip 10.1.1.1
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/2"]/ipv4/addresses/address[ip="10.1.1.1"]
MDCLI> mask 255.255.255.0 type main
[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/2"]/ipv4/addresses/address[ip="10.1.1.1"]
MDCLI> commit
```

The configurations of Device B and Device C are similar to the configuration of Device A. For configuration details, see Configuration Scripts.

Step 2 Configure IPv4 static routes on each device.

On DeviceA, configure a default IPv4 static route with DeviceB as the next hop.

```
[(gl)user@localhost]
MDCLI> network-instance instances instance name _public_
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]
MDCLI> afs af type ipv4-unicast
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]
MDCLI> routing static-routing unicast-route2s unicast-route2 topology-name base prefix 0.0.0.0 mask-length 0
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="0.0.0.0"][mask-length="0"]
MDCLI> nexthop-addresses nexthop-address address 10.1.4.2
[*(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="0.0.0.0"][mask-length="0"]/nexthop-addresses/nexthop-address[address="10.1.4.2"]
MDCLI> commit
```

On DeviceB, configure one IPv4 static route with DeviceA as the next hop and another IPv4 static route with DeviceC as the next hop.

```
[(gl)user@localhost]
MDCLI> network-instance instances instance name _public_
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]
```



```
MDCLI> afs af type ipv4-unicast
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]
MDCLI> routing static-routing unicast-route2s unicast-route2 topology-name base prefix 10.1.1.0
mask-length 24
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="10.1.1.0"][mask-
length="24"]
MDCLI> nexthop-addresses nexthop-address address 10.1.4.1
[*(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="10.1.1.0"][mask-
length="24"]/nexthop-addresses/nexthop-address[address="10.1.4.1"]
MDCLI> quit
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="10.1.1.0"][mask-
length="24"]
MDCLI> quit
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]
MDCLI> routing static-routing unicast-route2s unicast-route2 topology-name base prefix 10.1.3.0
mask-length 24
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="10.1.3.0"][mask-
length="24"]
MDCLI> nexthop-addresses nexthop-address address 10.1.4.6
[*(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="10.1.3.0"][mask-
length="24"]/nexthop-addresses/nexthop-address[address="10.1.4.6"]
MDCLI> commit
```

On DeviceC, configure a default IPv4 static route with DeviceB as the next hop.

```
[(gl)user@localhost]
MDCLI> network-instance instances instance name _public_
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]
MDCLI> afs af type ipv4-unicast
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]
MDCLI> routing static-routing unicast-route2s unicast-route2 topology-name base prefix 0.0.0.0 mask-
length 0
[(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="0.0.0.0"][mask-
length="0"]
MDCLI> nexthop-addresses nexthop-address address 10.1.4.5
[*(gl)user@localhost]/network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/static-routing/unicast-route2s/unicast-route2[topology-name="base"][prefix="0.0.0.0"][mask-
length="0"]/nexthop-addresses/nexthop-address[address="10.1.4.5"]
commit
```

----End

Verifying the Configuration

Check the IP routing table of DeviceA.

```
[user@localhost]
MDCLI> display /network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/
routing/routing-manage/topologys/topology[name="base"]/routes/ipv4-unicast-routes/ipv4-unicast-
route/ all
[
  {
    "prefix": "0.0.0.0",
    "mask-length": 0,
    "protocol-type": "static",
    "interface-name": "10GE1/0/1",
    "process-id": 0,
    "direct-nexthop": "10.1.4.2",
    "indirect-id": "0x0",
    "preference": 60,
    "cost": 0,
    "flag": "rd",
    "active": true,
```



```
  "state": "active-noadv"
},
{
  "prefix": "10.1.1.0",
  "mask-length": 24,
  "protocol-type": "direct",
  "interface-name": "10GE1/0/1",
  "process-id": 0,
  "direct-nexthop": "10.1.1.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "10.1.1.1",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "10GE1/0/1",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "10.1.1.255",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "10GE1/0/1",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "10.1.4.0",
  "mask-length": 30,
  "protocol-type": "direct",
  "interface-name": "10GE1/0/1",
  "process-id": 0,
  "direct-nexthop": "10.1.4.2",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "10.1.4.2",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "10GE1/0/1",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
```

```
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "10.1.4.255",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "10GE1/0/1",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "127.0.0.1",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "lo",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "127.0.0.0",
  "mask-length": 8,
  "protocol-type": "direct",
  "interface-name": "lo",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "127.255.255.255",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "lo",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
  "flag": "d",
  "active": true,
  "state": "active-noadv"
},
{
  "prefix": "255.255.255.255",
  "mask-length": 32,
  "protocol-type": "direct",
  "interface-name": "lo",
  "process-id": 0,
  "direct-nexthop": "127.0.0.1",
  "indirect-id": "0x0",
  "preference": 0,
  "cost": 0,
```



```
    "flag": "d",
    "active": true,
    "state": "active-noadv"
  }
]
```

The preceding command output shows that DeviceA's routing table contains a default IPv4 static route 0.0.0.0/0 and the next-hop address is 10.1.4.2.

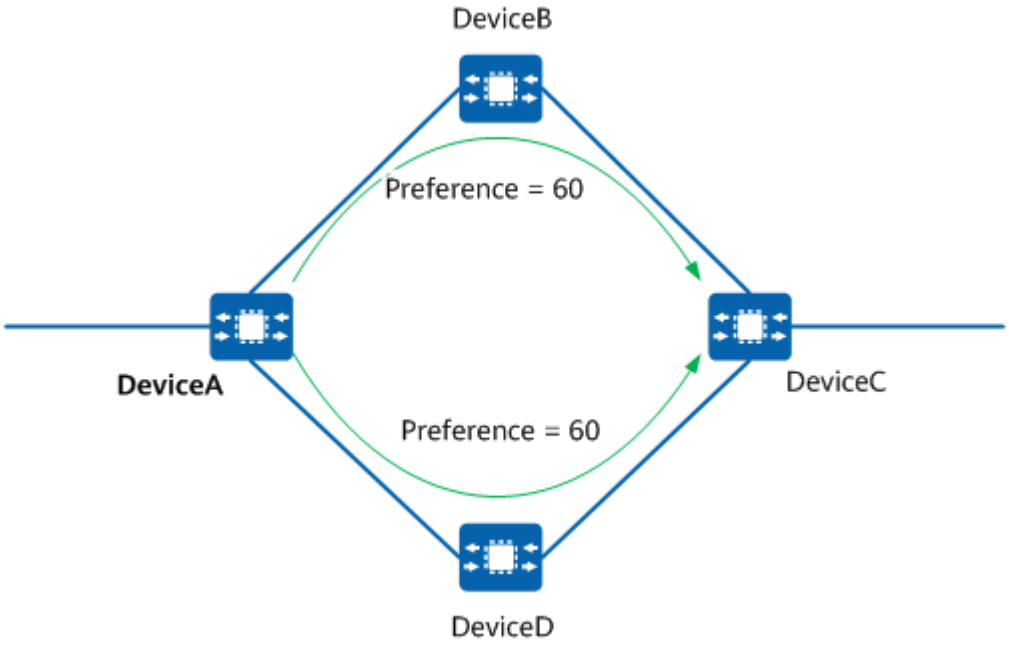
8.2.6 Configuring Load Balancing Among IPv4 Static Routes

Context

According to route selection rules, if IPv4 static routes have the same prefix, mask length, and preference, they load-balance traffic by default.

On the network shown in **Figure 8-5**, two static routes with the same prefix, mask length, and preference from DeviceA to DeviceC need to be configured. Both routes will be added to the routing table and used to forward data. For details about configuration parameters, see huawei-routing.yang.

Figure 8-5 Network diagram of load balancing among static routes



Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Perform the following operations as needed to configure an IPv4 static route. Then repeat the operations to configure another or more IPv4 static routes with the same prefix, mask, and preference.

- Configure an IPv4 static route on the public network.
  a. Enter the public network view.
  ```
  network-instance instances instance name _public_
  ```


b. Enter the IPv4 unicast address family view.
afs af type ipv4-unicast

c. Configure a route prefix.
routing static-routing unicast-route2s unicast-route2 topology-name base prefix *ip-address* **mask-length** *mask-length*

d. Configure a next-hop outbound interface or address. Select one of the following methods:

- Configure a next-hop outbound interface for the route.
nexthop-interfaces nexthop-interface interface-name *interface-name*

  (Optional) Configure a route preference and cost.
preference *preference* **cost** *cost*

- Configure a next-hop outbound interface and address for the route.
nexthop-interface-addresses nexthop-interface-address interface-name *interface-name* **address** *nexthop-address*

  (Optional) Configure a route preference and cost.
preference *preference* **cost** *cost*

- Configure a next-hop address for the route.
nexthop-addresses nexthop-address address *nexthop-address*

  (Optional) Configure a route preference and determine whether the route inherits the cost.
preference *preference* [**inherit-cost** { **true** | **false** }]

NOTE

By default, IPv4 static routes with the same prefix, mask, and preference can implement load balancing.

Step 3 Commit the configuration.
commit

----End

Verifying the Configuration

Run the **display /network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/routing-manage/topologys/topology[name="base"]/routes/ipv4-unicast-routes/ipv4-unicast-route/ all** command to check the routes with the same prefix and mask but different outbound interfaces and next hops in the IPv4 routing table.

8.2.7 Configuring a Default IPv4 Static Route

Context

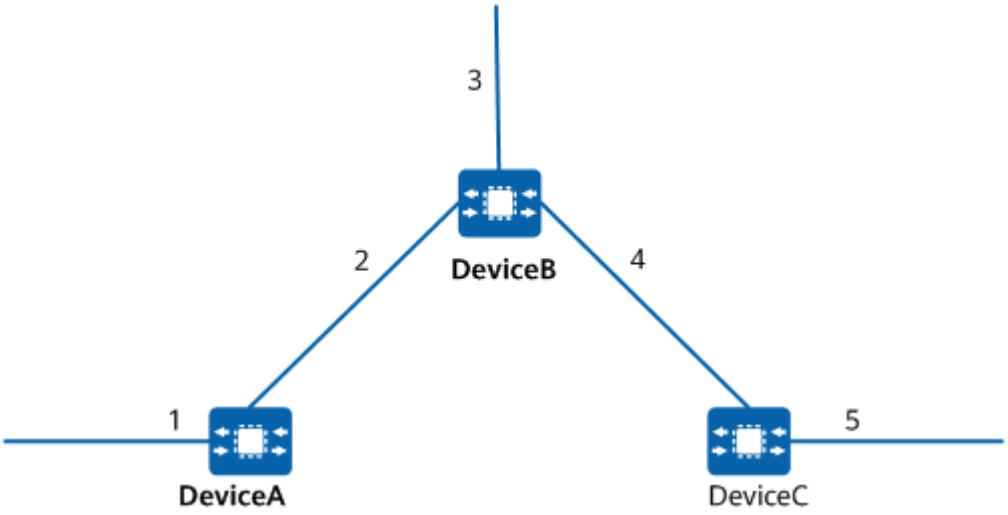
Default static routes are special routes that can be manually configured. Default static routes are used only when packets to be forwarded do not match any entry in the routing table. The destination address and subnet mask of the IPv4 default route are all 0s in the routing table.

If the destination address of a packet does not match any entry in the routing table, a device selects the default route to forward this packet. If no default route exists and the destination address of the packet does not match any entry in the routing table, the packet is discarded. An Internet Control Message Protocol

(ICMP) message is then sent to the source end to inform that the destination or network is unreachable.

If you set the destination address and mask to all 0s (0.0.0.0 0.0.0.0) when configuring a static route using the **routing static-routing unicast-route2s unicast-route2** command, the configured static route is a default route. This simplifies the network configuration. On the network shown in **Figure 8-6**, the next hop of packets sent from DeviceA to networks 3, 4, and 5 is DeviceB. Therefore, you can configure a default route on DeviceA to replace three static routes destined for networks 3, 4, and 5. Similarly, only one default route to DeviceB needs to be configured on DeviceC to replace the three static routes destined for networks 1, 2, and 3. For details about configuration parameters, see huawei-routing.yang.

Figure 8-6 Network diagram of static routes



Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Configure a default IPv4 static route.

- Configure a default IPv4 static route on the public network.
  a. Enter the public network view.
  ```
  network-instance instances instance name _public_
  ```
  b. Enter the IPv4 unicast address family view.
  ```
  afs af type ipv4-unicast
  ```
  c. Configure a route prefix.
  ```
  routing static-routing unicast-route2s unicast-route2 topology-name base prefix ip-address mask-length mask-length
  ```
  d. Configure a next-hop outbound interface or address. Select one of the following methods:
     - Configure a next-hop outbound interface for the route.
     ```
     nexthop-interfaces nexthop-interface interface-name interface-name
     ```
     (Optional) Configure a route preference and cost.
     ```
     preference preference cost cost
     ```


- Configure a next-hop outbound interface and address for the route.
  nexthop-interface-addresses nexthop-interface-address interface-name *interface-name* **address** *nexthop-address*

  (Optional) Configure a route preference and cost.
  preference *preference* **cost** *cost*

- Configure a next-hop address for the route.
  nexthop-addresses nexthop-address address *nexthop-address*

  (Optional) Configure a route preference and determine whether the route inherits the cost.
  preference *preference* [**inherit-cost** { **true** | **false** }]

Step 3 Commit the configuration.
commit

----End

Verifying the Configuration

Run the **display /network-instance/instances/instance[name="_public_"]/afs/af[type="ipv4-unicast"]/routing/routing-manage/topologys/topology[name="base"]/routes/ipv4-unicast-routes/ipv4-unicast-route/ all** command to check the routes with the prefix 0.0.0.0 in the IPv4 routing table.

9 User Access and Authentication Configuration

9.1 AAA Configuration

9.1.1 Overview of AAA

Access control is the way you control who is allowed access to the network server and what services they are allowed to use once they have access. Authentication, authorization, and accounting (AAA) provides a management framework for configuring access control on a Network Access Server (NAS).

Definition

AAA is an architectural framework for configuring a set of three independent security functions in a consistent manner. AAA provides a modular way of performing the following services:

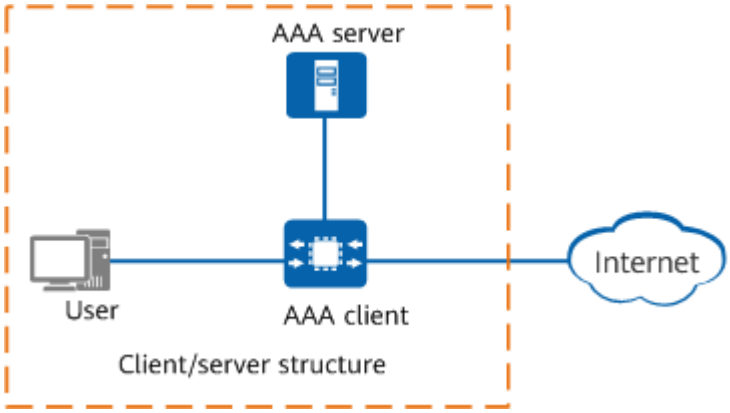
- Authentication: confirms the identities of users accessing the network and determines whether the users are authorized.
- Authorization: assigns differentiated rights to authorize users to use specific services.
- Accounting: records all the operations of a user during the network service process, including the used service type, start time, and data traffic, to collect and record the network resource usage of the user for implementing time- or traffic-based accounting and network monitoring.

Basic Architecture

AAA uses the client/server structure. The access device on which an AAA client runs is usually called an NAS. The NAS is responsible for user identity verification and user access management. An AAA server provides a collection of

authentication, authorization, and accounting functions and is responsible for centralized user information management. **Figure 9-1** shows the basic AAA architecture.

Figure 9-1 Basic architecture of AAA



For the AAA server in **Figure 9-1**, you can determine which protocols that the AAA server uses to implement authentication, authorization, and accounting functions respectively based on actual networking requirements. Users can use only one or two security services provided by AAA. For example, if a company only wants to authenticate employees who access certain network resources, the network administrator only needs to configure an authentication server. If the company also wants to record operations performed by employees on the network, an accounting server is required.

Purpose

AAA provides authentication, authorization, and accounting functions for users, preventing unauthorized users from logging in to a switch and improving system security.

9.1.2 Understanding AAA

In the AAA implementation, a set of AAA configuration policies can be defined using an AAA scheme. An AAA scheme is a set of authentication, authorization, and accounting methods defined on a device. Such methods can be used in combination depending on user access characteristics and security requirements.

9.1.2.1 Authentication Scheme

An authentication scheme defines the authentication methods to be used and the order in which authentication methods take effect.

Authentication Methods Supported by a Device

- Local authentication: The device functions as an authentication server and user information is configured on the device. This method features fast processing and low operation costs. However, the information storage capacity is subject to the device hardware.

9.1.2.2 Authorization Scheme

An authorization scheme defines the authorization methods and the order in which authorization methods take effect.

Authorization Methods Supported by a Device

- Local authorization: The device functions as an authorization server to authorize users based on user information configured on the device.

In addition, the "authentication + privilege level" method is typically used to control access of the administrators (login users) to the device, improving device operation security. Authentication restricts the administrators' access to the device and the privilege level defines commands that the administrators can enter after logging in to the device. For details about this method, see "First Login to a Device Configuration" in the *Configuration Guide - Basic Configuration*.

9.1.3 Configuration Precautions for AAA

Licensing Requirements

AAA is not under license control.

Hardware Requirements

Table 9-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

9.1.4 Default Settings for AAA

Table 9-2 describes the default settings for AAA.

Table 9-2 Default settings for AAA

Parameter	Default Setting
Local user	No local user is created.

9.1.5 Configuring a Local User

9.1.5.1 Understanding Local Authentication and Authorization

Local AAA Server

A device functioning as an AAA server is called a local AAA server, which performs user authentication and authorization but not user accounting.

Similar to the remote AAA server, the local AAA server requires the configurations of local user names, passwords, and authorization information. The authentication and authorization speed of a local AAA server is faster than that of a remote AAA server, which reduces operation costs. However, the information storage capacity of a local AAA server is subject to the device hardware.

Maximum Number of Local Users

The device has the following restrictions on the user quantity:

- A maximum of eight users can be configured on the device.
- The maximum number of concurrent online sessions for all users is 8.
- The configured user name cannot be the same as the default user name.

Local User Password Policy

The password policy of local users is vital to user security. Devices support the security policy of local user accounts, password complexity check, and password policy of local user accounts.

User Name, Password Length, and Password Complexity

The device has the following restrictions on the user name and password:

- The user name must meet the following requirements:
  - The user name must contain 1 to 31 characters.
  - The user name must consist of digits, uppercase letters, lowercase letters, periods (.), underscores (_), and hyphens (-).
  - The user name cannot start with a hyphen (-) or period (.).
- The password must meet the following requirements:
  - The password must contain 8 to 128 characters.
  - The password must contain three types of the following characters: digits, uppercase letters, lowercase letters, and special characters excluding question marks (?) and spaces.
  - The password cannot repeat the user name.
  - The password cannot be the same as the last five passwords including the current password.

Password Change Policy

- The local administrator can change the password of a local user at the same or lower privilege level.

- Local users cannot directly change their passwords. They need to run the **change-my-password** command in the root view to change their passwords.
- When a local user logs in to the device using the initial password or password changed by another administrator, the device prompts the user to change the password by default. To configure whether to prompt the user to change the password, run the **password-force-change** command.

Security Policy for Local Authentication

If a local account fails local authentication 5 consecutive times, the account will be locked for 5 minutes.

9.1.5.2 Configuring a Local User

Context

When an administrator uses local authentication and authorization, you need to configure a local administrator. For details about configuration parameters, see **huawei-aaa.yang**.

NOTE

- To keep the device secure, change the password periodically.
- When logging in to the device for the first time, a local user must change the password. You can run the **password-force-change false** command to prevent a specified user from changing the initial password upon the first login.
- After the user group to which local users belong is changed, the permissions of online users remain unchanged, and new users obtain new permissions when they go online.
- After the weak password dictionary is loaded, the passwords defined in the weak password dictionary (which can be queried using the **display system/weak-passwords** command) cannot be specified in the following commands for configuring local users:
  - Commands in the user node view:
    password
    Enter password: *password*
    Confirm password: *password*
  - Commands in the root view:
    change-my-password
    old password
    Enter password:*password*
    Confim password:*password*
    new password
    Enter password:*password*
    Confim password:*password*

Procedure

Step 1 Enter the editing mode.

```
edit-config
```

Step 2 Enter the AAA view.

```
aaa
```

Step 3 Configure a user name.

```
lam users user name user-name
```

The device has the following restrictions on the user name and password:

- The user name must meet the following requirements:
  - The user name must contain 1 to 31 characters.
  - The user name must consist of digits, uppercase letters, lowercase letters, periods (.), underscores (_), and hyphens (-).
  - The user name cannot start with a hyphen (-) or period (.).
- The password must meet the following requirements:
  - The password must contain 8 to 128 characters.
  - The password must contain three types of the following characters: digits, uppercase letters, lowercase letters, and special characters excluding question marks (?) and spaces.
  - The password cannot repeat the user name.
  - The password cannot be the same as the last five passwords including the current password.

Step 4 Configure a password for the local user. In the node view of the user, configure the password in interactive mode.

- Configure the password in interactive mode.
  password
  Enter password: *password*
  Confirm password: *password*

  NOTE

  The password is hidden if it is configured in interactive mode.

Step 5 (Optional) Configure a user group for the local user as required.

group-name admin

By default, a local user belongs to the user group admin.

Step 6 (Optional) Configure the terminal login permission for the local user as required.

service-terminal { **true** | **false** }

By default, the terminal login permission of a local user is **false**.

Step 7 (Optional) Configure the machine-to-machine login permission for the local user as required.

service-api { **true** | **false** }

By default, the machine-to-machine login permission of a local user is **false**.

Step 8 (Optional) Configure whether to prompt a local user to change the password upon the first login.

password-force-change { **true** | **false** }

By default, a local user must change the password configured for the first time upon the first login. You can run the **password-force-change false** command to prevent a specified user from changing the initial password upon the first login.

Step 9 (Optional) Configure the privilege level for the local user as required.

level *level*

By default, the permissions of the group to which the local user belongs take effect. Configuring a privilege level will overwrite the permissions of the group. Privilege levels 3 to 15 correspond to administrative permissions.

Step 10 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

In the root view, run the **display aaa/lam/users** command to check information about configured local users.

9.1.6 Maintaining AAA

9.1.6.1 Disconnecting Online Users

Context

You can disconnect online users by specifying their user names. This is required if an online user's AAA configuration is modified, because the new configuration takes effect only after the user is disconnected. It may also be necessary if the number of online users reaches the maximum value or online users are unauthorized users.

NOTE

Before deleting the AAA configuration of an online user, disconnect the online user.

Procedure

- In the root view, run the **cut-user-by-user-name user-name** *user-name* command to disconnect all sessions of the specified user so that the user can be disconnected.

----End

9.1.7 Configuration Examples for AAA

9.1.7.1 Example for Configuring a Local AAA User

Networking Requirements

As shown in Figure 9-2, the enterprise requires that AAA local authentication be used for login after the administrator configures a new local user. The administrator can log in to the device only after entering a correct user name and password.

Figure 9-2 Configuring AAA local authentication and authorization

Precautions

Ensure that there are reachable routes between the user terminal and DeviceA before the configuration.

Procedure

Step 1 Configure a local AAA user.

```
[admin@HUAWEI]
MDCLI> edit-config
[(gl)admin@HUAWEI]
MDCLI> aaa
[(gl)admin@HUAWEI]/aaa/lam/users
MDCLI> lam users user name user1-huawei
[*(gl)admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> password
Enter password:
Confirm password:
[*(gl)admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> group-name admin
[*(gl)admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> service-terminal true
[*(gl)admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> commit
```

----End

Verifying the Configuration

After a user connects to the device and enters the correct user name and password, the user can pass AAA local authentication and is required to change the initial password.

```
Authorized uses only. All activity may be monitored and reported.
HUAWEI login: user1-huawei
Password:
Warning: The initial password poses security risks.
The password needs to be changed. Change now? [Y/N]:y
Please enter old password:
Please enter new password:
Please confirm new password:

[user1-huawei@HUAWEI]
MDCLI>
```

In the root view, run the **display aaa/lam/users** command to check information about configured local users.

```
[user1-huawei@HUAWEI]
MDCLI> display aaa/lam/users
{
  "user": [
    {
      "name": "user1-huawei",
      "group-name": "admin",
      "password": "******",
      "service-terminal": true
    }
  ]
}
```

9.1.8 Troubleshooting AAA

9.1.8.1 Users Cannot Log In to the Device When AAA Local Authentication Is Used

Fault Symptom

When AAA local authentication is used, a user cannot log in to the device through SSH.

Possible Causes

- The user does not have an account on the device.
- The user name or password entered by the user is incorrect.
- The terminal login permission is not enabled for the user.

Troubleshooting Procedure

1. Run the **display this all** command in the AAA user node view to check whether the user has an account on the device and whether the terminal login permission of the user is true.

```
[admin@HUAWEI]
MDCLI> aaa lam users user name user1-huawei

[admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> display this all
{
  "name": "user1-huawei",
  "group-name": "admin",
  "password": "******",
  "service-terminal": true,
  "service-api": false
}
```

   – If the user does not have an account on the device, run the following command in the AAA user node view to create a local user and run the **service-terminal** *true* command to configure the terminal login permission for the user.

```
password
Enter password: password
Confirm password: password
```

   – If the user has an account on the device, ensure that the user name and password entered by the user are the same as those configured on the device.

     The password is displayed in cipher text on the device. If you forget the password, run the following command in the AAA user node view to change the password.

```
password
Enter password: password
Confirm password: password
```

   – If the user has an account on the device, ensure that the terminal login permission of the user has been enabled.

     To configure the terminal login permission for the user, run the **service-terminal** *true* command in the AAA user node view.

9.1.8.2 When AAA Local Authentication Is Used, a User Cannot Run Configuration-Level Commands After Logging In to the Device

Fault Symptom

A user logs in to the device successfully, but cannot run configuration-level commands.

Possible Causes

The permission of the group to which the user belongs is insufficient.

Troubleshooting Procedure

The administrator logs in to the device and reconfigures the user privilege level.

```
[admin@HUAWEI]
MDCLI> edit-config
[(gl)admin@HUAWEI]
MDCLI> aaa lam users user name user1-huawei
[(gl)admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> group-name admin
[*(gl)admin@HUAWEI]/aaa/lam/users/user[name="user1-huawei"]
MDCLI> commit
```

9.1.8.3 Failed to Create a Local AAA User Due to the Configuration of a Simple Password

Fault Symptom

A local AAA user fails to be created, and an error message is displayed.

Possible Causes

- The password length does not meet requirements.
- The password does not meet complexity requirements.

Troubleshooting Procedure

Rectify the fault based on the error message displayed on the device. The following table describes the troubleshooting methods.

Message	Possible Causes and Troubleshooting Method	Solution
Invalid password.	The password must contain 8 to 128 characters.	When creating a local user, ensure that the password contains 8 to 128 characters.

Message	Possible Causes and Troubleshooting Method	Solution
The password is too simple, it must consist of at least 3 types of characters, including lowercase letters, uppercase letters, numerals and special characters.	The password must contain at least three types of the following characters: digits, uppercase letters, lowercase letters, and special characters excluding question marks (?) and spaces.	During the creation of a local user, ensure that the configured password contains at least three types of the following characters: digits, uppercase letters, lowercase letters, and special characters excluding question marks (?) and spaces.
The password is a weak password.	The password matches the password in the weak password dictionary.	When creating a local user, ensure that the configured password is different from any password in the weak password dictionary. To check the loaded weak passwords that have taken effect, run the **display system/weak-passwords** command.
The password is same as the user name.	The password is the same as the user name.	When creating a local user, ensure that the password is different from the user name.

9.2 System Master Key Configuration

Context

A system master key is used to encrypt and decrypt locally stored data. When the device starts for the first time, it generates a system master key and uses it to encrypt data. The system master key is randomly generated by the system and is secure; therefore, you are not advised to change the system master key frequently.

- The system periodically generates a new master key to reduce the workload of manually maintaining the master key.
- If you need to use a user-defined key to encrypt and decrypt locally stored data, you can manually change the system master key.

Feature Requirements

- After a configuration file is copied from another device to the local device for next startup, if the master key on the source device does not exist on the local

device, the configuration fails. To resolve this problem, perform either of the following operations:

- Change the master key on the device to be configured to be the same as that on the device that provides the configuration file.
- Change the master key on the device that provides the configuration file to be the same as that on the device to be configured. After that, save and export the configuration file, upload it to the device to be configured, and specify it for next startup.

- After the master key is changed and an encrypted file is copied from another device to the local device, if the master key on the source device does not exist on the local device, the local device cannot decrypt the copied file due to master key mismatch. To resolve this problem, perform either of the following operations:
  - Change the master key on the local device to be the same as that on the device that provides the encrypted file.
  - Change the master key on the device that provides the encrypted file to be the same as that on the local device. Then export the encrypted file and upload it to the local device for decryption.

If an error occurs during master key change, the system prompts a message indicating a master key change failure and instructs the user to try again. If the failure persists, contact technical support.

Procedure

- Set the interval for automatically updating the system master key.

  edit-config //Enter the editing mode.
  masterkey auto-update //Enter the view for setting the interval for automatically updating the system master key.
  interval *interval* //Set the interval for automatically updating the system master key.
  commit //Commit the configuration.

  By default, the interval for automatically updating the system master key is 1825 days.

- Trigger the device to immediately generate a system master key.

  set-masterkey //Enter the system master key configuration view.
  emit //Trigger the device to immediately generate a system master key.

- Configure a user-defined system master key.

  set-masterkey new-masterkey *new-masterkey*

  A user-defined system key is a string of 20 to 32 characters, including uppercase letters, lowercase letters, digits, and special characters (excluding spaces).

- Clear historical system master keys.

  clear-masterkey

  NOTE

  If a system master key has been generated or customized on the device, running this command will cause the ciphertext encrypted using the historical master key unable to be decrypted. Therefore, you are advised to restart the device before running this command.

----End

Verifying the Configuration

- Run the **display masterkey**/**current-masterkey** command in the system view to check the effective system master key.
- Run the **display masterkey**/**current-type** command in the system view to check the effective system master key type.
- Run the **display masterkey**/**modify-result** command in the system view to check the delivery result of the user-defined system master key configuration.
- Run the **check-masterkey-sync-status** command in the system view to check the synchronization status of the system master key.

10 NAT Configuration

10.1 NAT Configuration

10.1.1 Overview of NAT

Definition

Network Address Translation (NAT) translates IP addresses in the packet header into other IP addresses.

Purpose

As the Internet expands and becomes more accessible, an increasing number of private network users access the Internet using public IPv4 addresses. The rapid exhaustion of IPv4 address space causes a significant depletion of public addresses. Although IPv6 technology can be used to solve this problem, most contents and applications are still based on IPv4, and therefore they cannot be completely switched to IPv6 within a short period of time. NAT technology allows public IPv4 addresses to be reused, offering a short-term solution to IPv4 address exhaustion.

Benefits

NAT has the following advantages:

- Translates a large number of private addresses into a few public IP addresses, resulting in private network users being able to access the Internet while the number of required public IP addresses is reduced.
- Ensures that external network users are unaware of private IP addresses, preventing threats to these addresses.
- Allows for flexible internal networking modes when the public IP addresses of services remain unchanged.

- Addresses the IP address overlapping problem.

10.1.2 Understanding NAT

10.1.2.1 NAT Types

NAT is categorized into different types based on the translation mode, as listed in **Table 10-1**.

Table 10-1 NAT types

Type		Translated Item	Port Translation	Application Scenario
Source NAT	**NAT No-PAT**	Source IP address	No	Public IP addresses are sufficient and only a small number of private network users require access to the Internet. There are one-to-one mappings between private and public addresses.
	NAPT	Source IP address	Yes	Many private network users require access to the Internet, and a large number of private addresses are translated into a small number of public addresses. In this mode, both the source IP addresses and port numbers in packets are translated.
	Easy IP	Source IP address	Yes	WAN interfaces dynamically obtain public IP addresses through dial-up, and the post-NAT public IP addresses are not fixed. Therefore, fixed IP addresses cannot be configured in a NAT address pool. Easy IP can also be applied to scenarios where only a few public addresses are available, which are just enough for configuring WAN interfaces.
Destination NAT	**NAT Server**	Destination IP address	Optional	There are fixed mappings between private IP addresses and public IP addresses, private IP addresses and public port numbers, private port numbers and public IP addresses, or private port numbers and public port numbers. NAT Server is implemented by running the **nat server** command.

10.1.2.2 NAT Policy

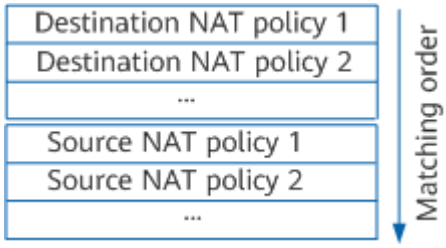
The source NAT function can be implemented by configuring a NAT policy that consists of post-NAT addresses (addresses in an address pool or outbound interface addresses), matching conditions, and actions.

- Source NAT address pools (NAT No-PAT and NAPT) are supported.
- Matching conditions include the source address, destination address, outbound interface, and service. You can configure different matching conditions to perform NAT translation on the traffic matching these conditions.
- Actions include different types of source address translation and destination address translation. NAT can be performed on the traffic matching certain conditions regardless of the chosen action.

In the NAT policy list shown in **Figure 10-1**, destination NAT policies (considered as a group) have higher matching priorities than source NAT policies (considered as another group), and are placed before source NAT policies. Destination NAT policies as well as source NAT policies are sorted according to their configuration sequences. A newly added policy or a policy with the NAT action modified is placed at the end of NAT policies of its own group.

You can adjust the matching order of NAT policies in the same group as required. For example, destination NAT policy 2 can be placed above destination NAT policy 1, or source NAT policy 2 above source NAT policy 1. However, a source NAT policy cannot be placed above destination NAT policies. For example, source NAT policy 1 cannot be placed above destination NAT policy 2.

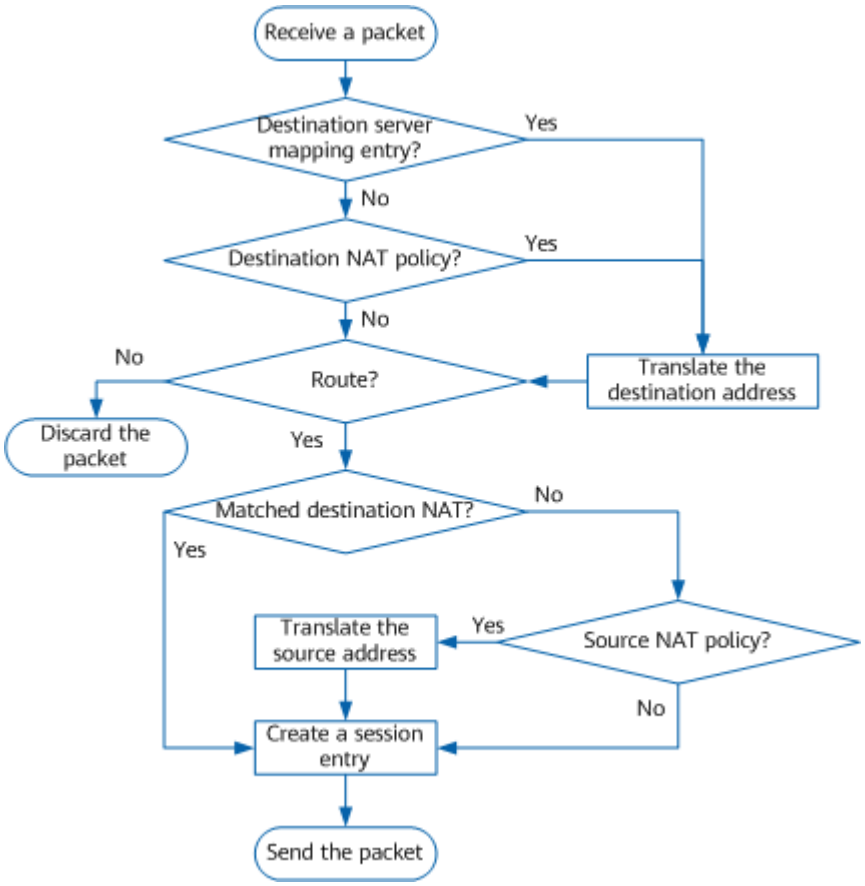
Figure 10-1 Example of a NAT policy list



10.1.2.3 NAT Processing Flow

Different NAT types correspond to different NAT policies. The device matches NAT policies based on their priorities.

Figure 10-2 NAT processing flow



As shown in **Figure 10-2**, the NAT processing flow is described as follows:

1. When receiving a packet, the device searches for a matching server mapping entry generated by NAT Server. If a match is found, the device translates the destination address of the packet accordingly and proceeds to step **3**. If no match is found, the device proceeds to step **2**.
2. The device checks whether the packet meets the conditions of a destination NAT policy. If so, the device translates the destination address of the packet, and then proceeds to step **3**. If the packet does not match any destination NAT policy, the device proceeds to step **3**.
3. The device searches for a matching route (including routes available in PBR scenarios) for the packet. If a match is found, the device proceeds to step **4**. If no match is found, the device discards the packet.
4. If the packet did not match any destination NAT policy before, the device proceeds to step **5**. If the packet matched a destination NAT policy, the device proceeds to step **6**.
5. The device searches for a source NAT policy matching the packet. If a match is found, the device translates the source address of the packet into a public address, and creates a session for the packet. If no match is found, the device creates a session for the packet without address translation and proceeds to step **6**.
6. The device sends the packet out according to the matching route.

Destination NAT, routing, and source NAT are performed in sequence. Therefore, the source address configured in a route is the pre-NAT source address, and the destination address configured in a route is the post-NAT destination address.

The NAT processing on the device is as follows:

10.1.3 Configuration Precautions for NAT

Licensing Requirements

The NAT function is not under license control.

Hardware Requirements

Table 10-2 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

10.1.4 Default Settings for NAT

Table 10-3 describes the default settings for NAT.

Table 10-3 Default settings for NAT

Parameter	Default Setting
NAT mode of an address pool	**pat** mode
Status of an address pool	Active

10.1.5 Configuring Source NAT

10.1.5.1 Understanding Source NAT

10.1.5.1.1 Overview of Source NAT

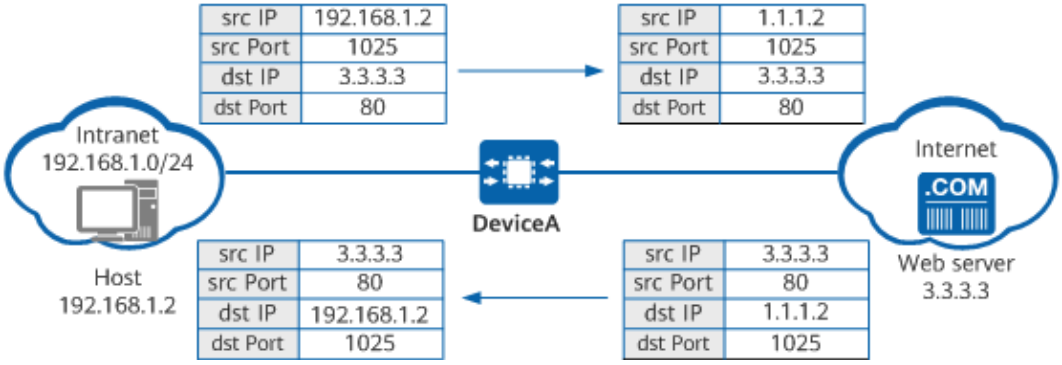
Definition

Source NAT translates the source addresses of packets.

Implementation

Source NAT translates private IP addresses into public IP addresses so that users on an intranet can access the Internet using public IP addresses. **Figure 10-3** shows the source NAT process.

Figure 10-3 Source NAT process



When a host on the intranet accesses the web server on the Internet, the source NAT process on DeviceA is as follows:

1. When receiving a packet from the host, DeviceA translates the private source IP address of the packet into a public address.
2. When receiving a return packet from the web server, DeviceA translates the public destination IP address back into the private IP address of the host.

Classification

Based on whether port translation is performed during source address translation, source NAT is categorized into two types:

- NAT involving only source address translation: NAT No-PAT
- NAT involving both source address translation and source port translation: NAPT and Easy IP

10.1.5.1.2 NAT No-PAT

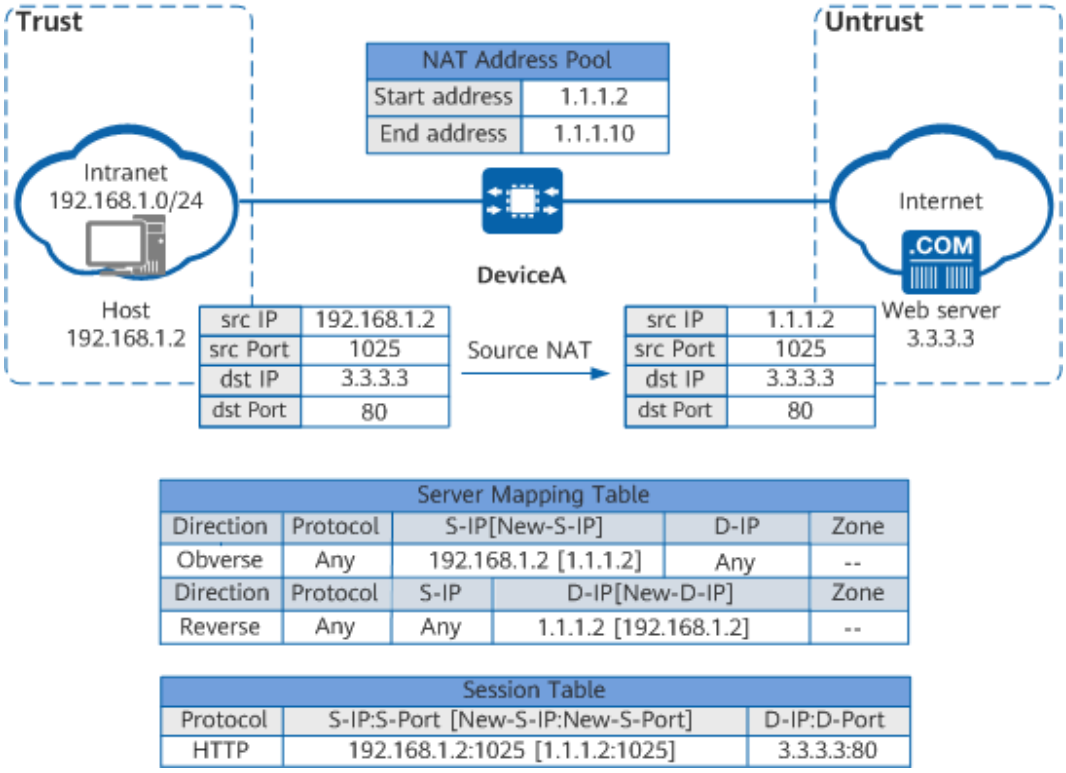
Definition

NAT No-PAT translates a private source IP address into a unique public address, but does not translate source port numbers.

Implementation

NAT No-PAT applies to scenarios where each private network user usually has a public IP address. **Figure 10-4** shows the NAT No-PAT process.

Figure 10-4 NAT No-PAT process



When a host on the intranet accesses the web server on the Internet, the NAT No-PAT process on DeviceA is as follows:

1. When receiving a packet from the host, DeviceA searches for a matching NAT policy and discovers that the packet requires NAT translation.
2. DeviceA replaces the source IP address of the packet with a public IP address selected from the NAT address pool using the polling algorithm, creates a server mapping entry and session entry accordingly, and then forwards the packet to the web server on the Internet.
3. The web server returns a response packet destined for the host. When receiving this packet, DeviceA searches the session table for the entry created in step 2, translates the destination address of the packet into the private IP address of the host accordingly, and then forwards the packet to the host.

In NAT No-PAT, there are one-to-one mappings between the private and public IP addresses. If all public addresses in an address pool are allocated, NAT cannot be performed for other intranet hosts.

The server mapping table stores the mappings between the private and public IP addresses of hosts.

Forward server mapping entries enable fast address translation when a private network user accesses the Internet, improving the processing efficiency of the device. Return server mapping entries allow for address translation when an Internet user proactively accesses a private network user.

10.1.5.1.3 NAPT

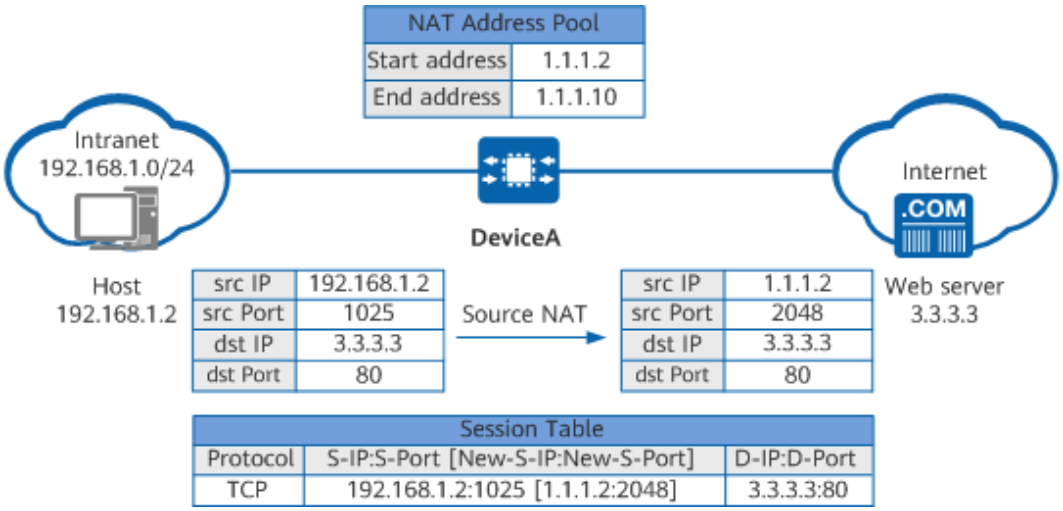
Definition

NAPT translates both source IP addresses and ports to allow multiple private addresses to be translated into the same public address.

Implementation

NAPT applies to scenarios where a limited number of public addresses exist but many private users require access to the Internet. **Figure 10-5** shows the NAPT process.

Figure 10-5 NAPT process



When a host on the intranet accesses the web server on the Internet, the NAPT process on DeviceA is as follows:

1. When receiving a packet from the host, DeviceA searches for a matching NAT policy and discovers that the packet requires NAT translation.
2. DeviceA replaces the source IP address of the packet with a public IP address selected from the NAT address pool based on the hash result of the source IP address, replaces the source port number with a new one, creates a session entry accordingly, and forwards the packet to the web server on the Internet.
3. The web server returns a response packet destined for the host. When receiving this packet, DeviceA searches the session table for the entry created in step 2, translates the destination address of the packet into the host's private IP address and the destination port number into the host's private port number based on this entry, and then forwards the packet to the host.

As both addresses and ports are translated, multiple private network users can use the same public address to access the Internet. The device distinguishes users based on ports, enabling more users to access the Internet simultaneously. Unlike NAT No-PAT, NAPT does not generate server mapping entries.

10.1.5.1.4 Easy IP

Definition

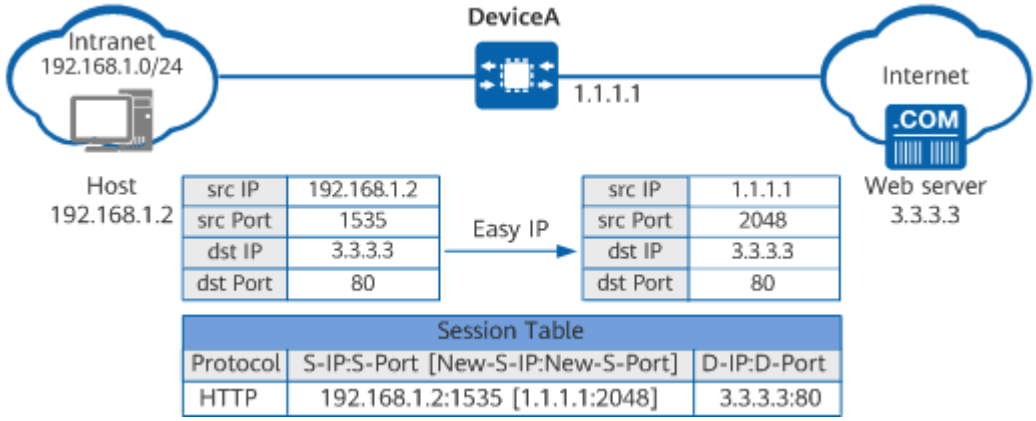
Easy IP translates both source IP addresses and ports and uses the public IP address of the outbound interface (a WAN interface connected to the Internet) as the post-NAT address.

Implementation

Easy IP applies to scenarios where the outbound interface obtains an IP address dynamically.

If the WAN interface of the device obtains a public IP address through dial-up and you want to use only this address as the post-NAT address, you cannot configure a fixed public IP address in the NAT address pool. Easy IP can be used in this scenario as it allows the device to translate private source addresses into the new public IP address of the WAN interface. **Figure 10-6** shows the Easy IP process.

Figure 10-6 Easy IP process



When a host on the intranet accesses the web server on the Internet, the Easy IP process on DeviceA is as follows:

1. When receiving a packet from the host, DeviceA searches for a matching NAT policy and discovers that the packet requires NAT translation.
2. DeviceA replaces the source IP address of the packet with the public IP address of its WAN interface connected to the Internet, and replaces the source port number with a public port number, creates a session entry accordingly, and forwards the packet to the web server on the Internet.
3. The web server returns a response packet destined for the host. When receiving this packet, DeviceA searches the session table for the entry created in step 2, translates the destination address of the packet into the host's private IP address and the destination port number into the host's private port number based on this entry, and then forwards the packet to the host.

As both addresses and ports are translated, multiple private network users can use the same public address to access the Internet. The device distinguishes users based on ports, enabling more users to access the Internet simultaneously.

10.1.5.2 Configuring a Source NAT Address Pool

Context

Before creating a NAT policy, create a NAT address pool, specify the post-NAT address ranges and source NAT mode, and configure a blackhole route.

You do not need to configure a source NAT address pool if source NAT in Easy IP mode is used.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Create a NAT address pool and enter the NAT address pool view.

```
nat-address-group snat-address-groups snat-address-group name
```

Step 3 Specify IP address ranges in the address pool using either of the following methods:

- Configure one or more IP address ranges.

  ```
  sections section [ id ]
  start-ip [ end-ip ]
  ```

- Specify a large address range and exclude some addresses or ports from the range.

  ```
  sections section [ id ]
  start-ip [ end-ip ]
  exclude-ips exclude-ip start-ip [ end-ip ]
  exclude-ports exclude-port start-port [ end-port ]
  ```

Step 4 Configure a source NAT mode for the address pool.

Table 10-4 Configuring a source NAT mode

Source NAT Mode	Command	Description
NAT No-PAT	**mode no-pat**	There are one-to-one mappings between private and public IP addresses, and port translation is not performed.
NAPT	**mode pat**	Multiple private IP addresses share one or more public IP addresses, and both source IP addresses and port numbers are translated.

Step 5 Commit the configuration.

```
commit
```

----End

10.1.5.3 Configuring a Source NAT Policy

Prerequisites

You have **configured a source NAT address pool**.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the view of the inbound interface of packets.

```
ifm interfaces interface name interface-number
```

Step 3 (Optional) Enable the NAT function on the interface.

```
nat nat-enable { true | false }
```

By default, the NAT function is enabled on an interface.

Step 4 Return to the edit-config view.

```
quit value
```

value indicates the number of times that you need to exit views.

Step 5 Enter the NAT policy view.

```
nat-policy
```

Step 6 Create a NAT rule and enter the NAT rule view.

```
rules rule name rule-name
```

The NAT function is implemented using NAT rules in a NAT policy. A NAT rule must be created in order to specify the data flow that requires NAT translation and translation action. If multiple NAT rules are configured, the device matches packets against the rules from top to bottom in the NAT rule list until a match is found. As such, ensure that NAT rules are configured in the correct sequence.

Step 7 Configure matching conditions in the NAT rule.

Matching conditions are optional configurations. By default, the matching condition of a NAT rule is **any**, indicating that all traffic matches this rule. If matching conditions are configured for a NAT rule, traffic must satisfy all matching conditions in order to match this rule.

Table 10-5 Configuring matching conditions in the NAT rule

Matching Condition	Command	Description
Source IP address	**source-address** { **address-ipv4s address-ipv4 ipv4** *ipv4-address* **mask** *mask-address* | **address-ipv4-ranges address-ipv4-range start-ipv4** *ipv4-address* **end-ipv4** *ipv4-address* }	Configures the source IP addresses of traffic. You can exclude addresses from the address range that is to be translated. For example, source NAT translation is required for users on the

Matching Condition	Command	Description
	source-address { **address-ipv4s-excludes address-ipv4-exclude ipv4** *ipv4-address* **mask** *mask-address* | **address-ipv4-ranges-excludes address-ipv4-range-exclude start-ipv4** *ipv4-address* **end-ipv4** *ipv4-address* }	10.1.1.0/24 network segment, excluding the users from 10.1.1.2 to 10.1.1.5. **nat-policy rules rule name rule1** **source-address address-ipv4s address-ipv4 ipv4 10.1.1.0 mask 255.255.255.0** **source-address address-ipv4-range-excludes address-ipv4-range-exclude start-ipv4 10.1.1.2 end-ipv4 10.1.1.5**
Destination IP address	**destination-address** { **address-ipv4s address-ipv4 ipv4** *ipv4-address* **mask** *mask-address* | **address-ipv4-ranges address-ipv4-range start-ipv4** *ipv4-address* **end-ipv4** *ipv4-address* }	Configures the destination IP addresses of traffic.
	destination-address { **address-ipv4s-excludes address-ipv4-exclude ipv4** *ipv4-address* **mask** *mask-address* | **address-ipv4-ranges-excludes address-ipv4-range-exclude start-ipv4** *ipv4-address* **end-ipv4** *ipv4-address* }	Excludes addresses from the address range that is to be translated.
Outbound Interface	**egress-interface** *interface-name*	Configure the outbound interface of traffic.

Matching Condition	Command	Description
Service	• **service service-items protocol-and-ports protocol-and-port protocol** { **udp** | **tcp** | **sctp** } [**source-port** *source-port* **destination-port** *destination-port*] * • **service service-items protocol protocol-id** *protocol-number* • **service service-items icmpv4s icmpv4 type** *icmp-type-number* **code** *icmp-code-number* • **service service-items-exclude protocol-and-ports protocol-and-port protocol** { **udp** | **tcp** | **sctp** } [**source-port** *source-port* **destination-port** *destination-port*] * • **service service-items-exclude protocol protocol-id** *protocol-number* • **service service-items-exclude icmpv4s icmpv4 type** *icmp-type-number* **code** *icmp-code-number*	Configure the port number and protocol number of a service. For example, you can configure services with source TCP ports 123 to 128 as a matching condition using either of the following methods: **nat-policy rules rule name rule1** **service service-items protocol-and-ports protocol-and-port protocol tcp source-port 123-128 dest-port 255-258**

Step 8 Define an action in the NAT rule.

action { **source-nat** { **address-group** *address-group-name* | **easy-ip** } | **no-nat** }

source-nat indicates that source NAT is performed for the matched traffic, whereas **no-nat** indicates that NAT is not performed for the matched traffic.

no-nat primarily applies to certain special clients. For example, to perform NAT translation for all hosts except 192.168.1.2 on the 192.168.1.0/24 network segment, you can first configure a policy that does not translate 192.168.1.2 and then configure a policy to translate the 192.168.1.0/24 network segment.

Step 9 Commit the configuration.

commit

----End

Verifying the Configuration

Run the **display nat-policy rule** command to check the configuration of NAT rules.

10.1.5.4 Example for Configuring NAT No-PAT for Intranet Users to Access the Internet

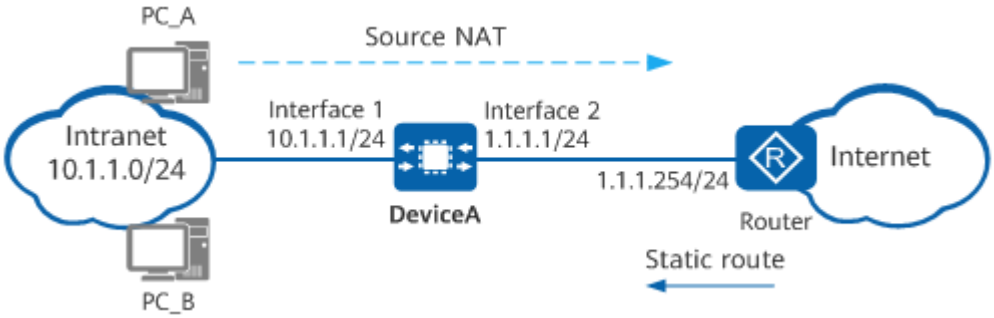
Networking Requirements

An enterprise has deployed DeviceA as a security gateway at the intranet border. A source NAT policy needs to be configured on DeviceA so that intranet users on the 10.1.1.0/24 network segment can access the Internet. As only a small number of users require access to the Internet, NAT No-PAT can be configured on DeviceA to perform one-to-one translation between private and public addresses. The enterprise has obtained six public IP addresses (1.1.1.10 to 1.1.1.15) from the Internet service provider (ISP) for address translation. **Figure 10-7** shows the network diagram, in which the router is the access gateway provided by the ISP.

Figure 10-7 Network diagram of NAT No-PAT

NOTE

In this example, interface 1 and interface 2 represent GE 0/0/1 and GE 0/0/2, respectively.



Item		Data	Description
GE0/0/1		IP address: 10.1.1.1/24	Set the default gateway address on each intranet host to 10.1.1.1.
GE0/0/2		IP address: 1.1.1.1/24	1.1.1.1/24 is a public address provided by the ISP.
Private network segment allowed to access the Internet		10.1.1.0/24	-
Post-NAT public addresses		1.1.1.10 to 1.1.1.15	Configure NAT No-PAT on DeviceA as a small number of users require access to the Internet.
Routes	Default route on DeviceA	Destination address: 0.0.0.0 Next-hop address: 1.1.1.254	Configure a default route to the Internet on DeviceA, enabling it to forward traffic from the intranet to the ISP router.

Item		Data	Description
	Static route on the ISP router	Destination address: 1.1.1.10 to 1.1.1.15 Next-hop address: 1.1.1.1	The post-NAT public addresses are not assigned to interfaces, which results in the ISP router not being able to use a routing protocol to discover routes to these public addresses. Therefore, contact the ISP network administrator to configure a static route to the network segment in the address pool on the router.

Procedure

Step 1 Configure a NAT address pool and disable port translation.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> nat-address-group snat-address-groups snat-address-group name test
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]
MDCLI> mode no-pat
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]
MDCLI> sections section id 1
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]/sections/
section[id="1"]
MDCLI> start-ip 1.1.1.10 end-ip 1.1.1.15
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]/sections/
section[id="1"]
MDCLI> commit
[(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]/sections/
section[id="1"]
MDCLI> quit 255
```

Step 2 Configure a source NAT policy to enable source address translation for intranet users on a specified network segment to access the Internet.

```
[(gl)user@HUAWEI]
MDCLI> nat-policy rules rule name rule1
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]
MDCLI> source-address address-ipv4s address-ipv4 ipv4 10.1.1.0 mask 255.255.255.0
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/source-address/address-ipv4s/address-
ipv4[ipv4="10.1.1.0"][mask="255.255.255.0"]
MDCLI> quit 3
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]
MDCLI> action source-nat
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/action/source-nat
MDCLI> address-group-name test
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/action/source-nat
MDCLI> commit
```

Step 3 Configure a default route on DeviceA, enabling it to forward traffic sent from intranet users to the ISP router. The configuration details are not provided here.

Step 4 Configure the default gateway address on each intranet host, enabling them to send traffic destined for the Internet to DeviceA. The detailed configuration process is not provided here.

Step 5 On the ISP router, configure a static route to addresses in the NAT address pool (1.1.1.10 to 1.1.1.15), with the next-hop address set to 1.1.1.1. The ISP router can then forward traffic returned by the Internet to DeviceA.

Contact the ISP network administrator to perform this step.

----End

Verifying the Configuration

1. Verify that intranet PCs can access the Internet.

10.1.5.5 Example for Configuring NAPT for Intranet Users to Access the Internet

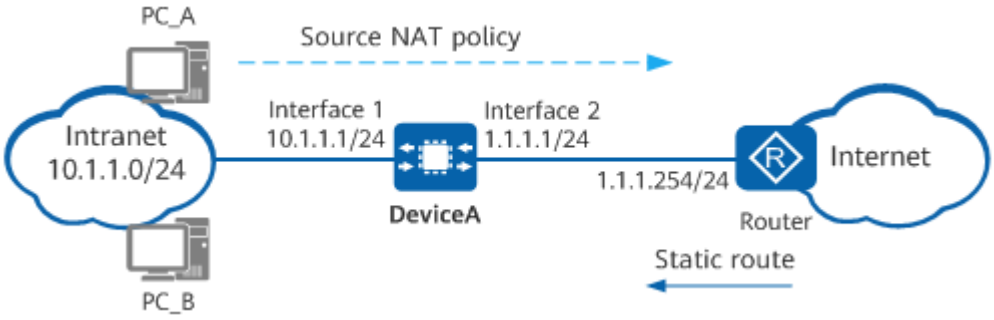
Networking Requirements

An enterprise has deployed DeviceA as a security gateway at the intranet border. A source NAT policy needs to be configured on DeviceA so that intranet users on the 10.1.1.0/24 network segment can access the Internet. In addition to the public IP address of the WAN interface on DeviceA, the enterprise has obtained six public IP addresses (1.1.1.10 to 1.1.1.15) from the ISP for address translation. If a large number of users on the intranet access the Internet, you need to configure NAPT to enable both address and port translation; however, port conflicts may occur during NAT translation. To avoid this, set the maximum number of private addresses that correspond to a public address. **Figure 10-8** shows the network diagram, in which the router is the access gateway provided by the ISP.

Figure 10-8 Network diagram of NAPT

NOTE

In this example, interface 1 and interface 2 represent GE 0/0/1 and GE 0/0/2, respectively.



Item	Data	Description
GE0/0/1	IP address: 10.1.1.1/24	Set the default gateway address on each intranet host to 10.1.1.1.
GE0/0/2	IP address: 1.1.1.1/24	1.1.1.1/24 is a public address provided by the ISP.

Item		Data	Description
Private network segment allowed to access the Internet		10.1.1.0/24	-
Post-NAT public addresses		1.1.1.10 to 1.1.1.15	As private addresses far outnumber public addresses, one-to-one mapping cannot be implemented. To translate all private addresses into public addresses, enable port translation.
(Optional) Maximum number of private addresses corresponding to a single public address		256	-
Routes	Default route on DeviceA	Destination address: 0.0.0.0 Next-hop address: 1.1.1.254	Configure a default route to the Internet on DeviceA, enabling it to forward traffic from the intranet to the ISP router.
	Static route on the ISP router	Destination address: 1.1.1.10 to 1.1.1.15 Next-hop address: 1.1.1.1	The post-NAT public addresses are not assigned to interfaces, which results in the ISP router not being able to use a routing protocol to discover routes to these public addresses. Therefore, contact the ISP network administrator to configure a static route to the network segment in the address pool on the router.

Procedure

Step 1 Configure a NAT address pool, enable port translation, and set the maximum number of private addresses to be translated into one public address to 256.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> nat-address-group snat-address-groups snat-address-group name test
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]
MDCLI> mode pat
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]
MDCLI> sections section id 1
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]/sections/
section[id="1"]
MDCLI> start-ip 1.1.1.10 end-ip 1.1.1.15
[*(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]/sections/
section[id="1"]
```



```
MDCLI> commit
[(gl)user@HUAWEI]/nat-address-group/snat-address-groups/snat-address-group[name="test"]/sections/
section[id="1"]
MDCLI> quit 255
```

Step 2 Configure a source NAT policy to enable source address translation for intranet users on a specified network segment to access the Internet.

```
[user@HUAWEI]
MDCLI> nat-policy rules rule name rule1
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]
MDCLI> source-address address-ipv4s address-ipv4 ipv4 10.1.1.0 mask 255.255.255.0
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/source-address/address-ipv4s/address-
ipv4[ipv4="10.1.1.0"][mask="255.255.255.0"]
MDCLI> quit 3
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]
MDCLI> action source-nat
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/action/source-nat
MDCLI> address-group-name test
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/action/source-nat
MDCLI> commit
```

Step 3 Configure a default route on DeviceA, enabling it to forward traffic sent from intranet users to the ISP router. The configuration details are not provided here.

Step 4 Configure the default gateway address on each intranet host, enabling them to send traffic destined for the Internet to DeviceA. The detailed configuration process is not provided here.

Step 5 On the ISP router, configure a static route to addresses in the NAT address pool (1.1.1.10 to 1.1.1.15), with the next-hop address set to 1.1.1.1. The ISP router can then forward traffic returned by the Internet to DeviceA.

Contact the ISP network administrator to perform this step.

----End

Verifying the Configuration

1. Verify that intranet PCs can access the Internet.

10.1.5.6 Example for Configuring Easy IP for Intranet Users to Access the Internet

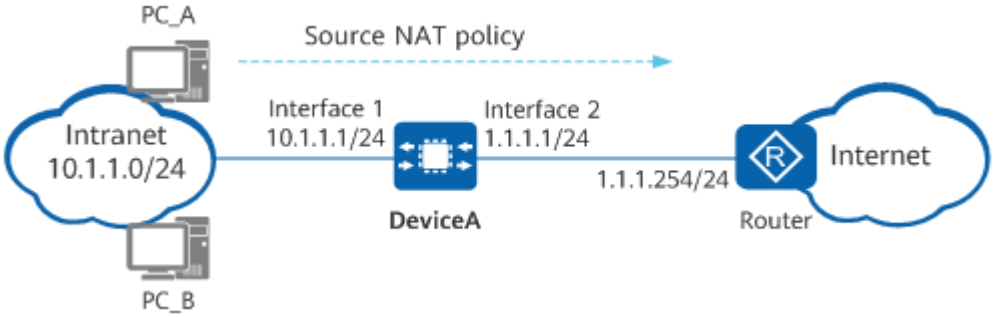
Networking Requirements

An enterprise has deployed DeviceA as a security gateway at the intranet border. This public IP address is assigned to the WAN interface of DeviceA so that it can communicate with the ISP router, which is an access gateway on the ISP network. To allow intranet users on the 10.1.1.0/24 network segment to access the Internet, you need to configure a source NAT policy in Easy IP mode on DeviceA. This policy allows DeviceA to translate private IP addresses of intranet users into the public IP address of the outbound interface (the WAN interface of DeviceA) of user packets. **Figure 10-9** shows the network diagram, in which the router is the access gateway provided by the ISP.

Figure 10-9 Network diagram of Easy IP

NOTE

In this example, interface 1 and interface 2 represent GE 1/0/1 and GE 1/0/2, respectively.



Item	Data	Description
GE1/0/1	IP address: 10.1.1.1/24	Set the default gateway address on each intranet host to 10.1.1.1.
GE1/0/2	IP address: 1.1.1.1/24	1.1.1.1/24 is a public address provided by the ISP.
Private network segment allowed to access the Internet	10.1.1.0/24	-
Default route on DeviceA	Destination address: 0.0.0.0 Next-hop address: 1.1.1.254	Configure a default route to the Internet on DeviceA, enabling it to forward traffic from the intranet to the ISP router.

Procedure

Step 1 Configure a NAT policy in Easy IP mode.

```
[user@HUAWEI]
MDCLI> nat-policy rules rule name rule1
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]
MDCLI> source-address address-ipv4s address-ipv4 ipv4 10.1.1.0 mask 255.255.255.0
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/source-address/address-ipv4s/address-ipv4[ipv4="10.1.1.0"][mask="255.255.255.0"]
MDCLI> quit 3
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]
MDCLI> action source-nat
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/action/source-nat
MDCLI> easy-ip [ null ]
[*(gl)user@HUAWEI]/nat-policy/rules/rule[name="rule1"]/action/source-nat
MDCLI> commit
```


Step 2 Configure the default gateway address on each intranet host, enabling them to send traffic destined for the Internet to DeviceA. The detailed configuration process is not provided here.

----End

Verifying the Configuration

1. Verify that intranet PCs can access the Internet.

10.1.6 Configuring Destination NAT

10.1.6.1 Understanding Destination NAT

10.1.6.1.1 Overview of Destination NAT

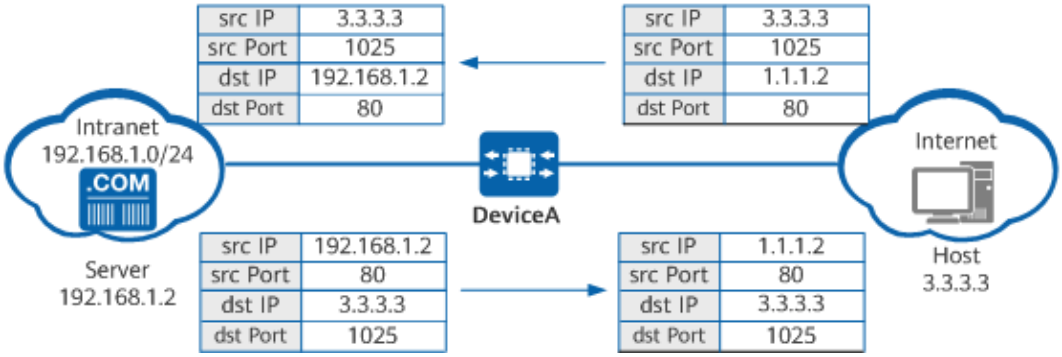
Definition

Destination NAT translates both the destination addresses and ports of packets.

Implementation

Destination NAT translates public IP addresses into private IP addresses so that users on the Internet can access intranet servers using public IP addresses. **Figure 10-10** shows the destination NAT translation process.

Figure 10-10 Implementation of destination NAT



When an Internet user accesses a server on the intranet, the destination NAT process on DeviceA is as follows:

1. When receiving a packet from the host, DeviceA translates the destination public address of the packet into a private address.
2. When receiving a return packet from the server on the intranet, DeviceA translates the source private address back into the public address of the host.

10.1.6.1.2 NAT Server

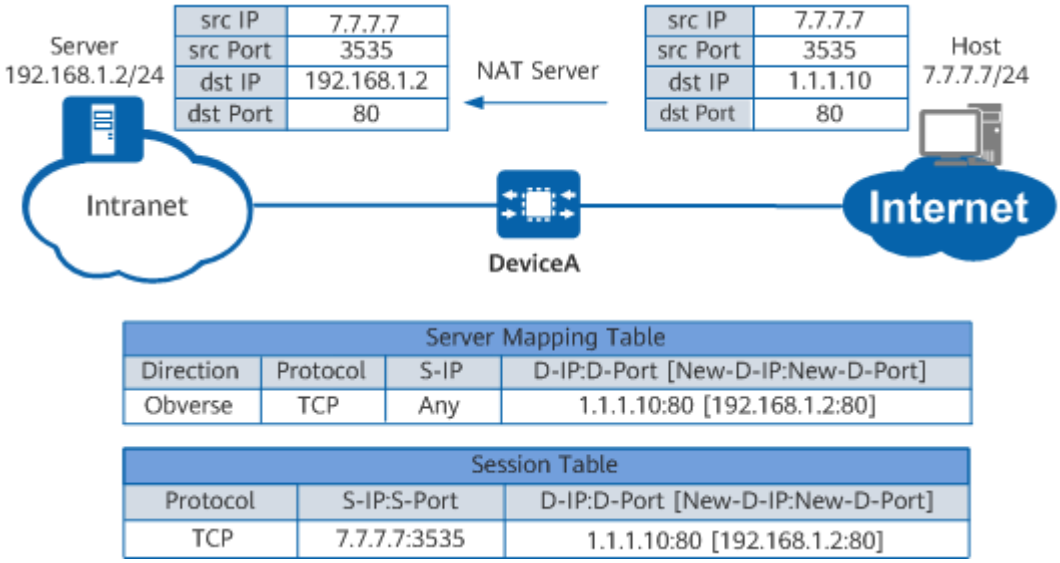
Definition

NAT Server is a static destination address translation technology. It translates the public IP addresses in packets into private IP addresses in fixed mappings.

Implementation

In some scenarios, schools or companies provide services for Internet users. However, the servers that provide these services are usually configured with private IP addresses during network deployment, which are inaccessible to Internet users. To address this, the device functioning as the egress gateway can employ NAT Server to translate private IP addresses into public IP addresses for Internet users' access. **Figure 10-11** shows the implementation of NAT Server.

Figure 10-11 Implementation of NAT Server



Server Mapping Table			
Direction	Protocol	S-IP	D-IP:D-Port [New-D-IP:New-D-Port]
Obverse	TCP	Any	1.1.1.10:80 [192.168.1.2:80]

Session Table		
Protocol	S-IP:S-Port	D-IP:D-Port [New-D-IP:New-D-Port]
TCP	7.7.7.7:3535	1.1.1.10:80 [192.168.1.2:80]

After NAT Server is configured on the device to determine the mappings between public and private IP addresses, the device generates server mapping entries to store the mappings. The entries will always exist unless the NAT Server configuration is deleted.

DeviceA performs the following when a host on the Internet accesses a server on the intranet:

1. When receiving the first packet destined for 1.1.1.10 from the host on the Internet, DeviceA searches for a matching server mapping entry, and translates the destination IP address of the packet into 192.168.1.2 accordingly.
2. DeviceA creates a session entry based on the destination IP address of the packet, and forwards the packet to the server on the intranet.
3. When receiving a return packet from the server, DeviceA searches the session table for the entry created in step 2, replaces the source address of the packet with 1.1.1.10, and forwards the packet to the host on the Internet.

4. Upon receipt of subsequent packets sent from the host to the server, DeviceA performs NAT translation for the packets according to the session entry, instead of searching for a server mapping entry.

You can configure NAT Server flexibly to meet various requirements of different scenarios. For example, you can specify parameters such as the port and protocol to determine whether an intranet server is allowed to use a public IP address to access the Internet.

10.1.6.2 Configuring NAT Server

10.1.6.2.1 Configuring NAT Server

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the view of the inbound interface of packets.

```
ifm interfaces interface name interface-number
```

Step 3 (Optional) Enable the NAT function on the interface.

```
nat nat-enable { true | false }
```

By default, the NAT function is enabled on an interface.

Step 4 Return to the edit-config view.

```
quit value
```

value indicates the number of times that you need to exit the view.

Step 5 Configure NAT Server.

Whether a Protocol Type Is Specified	Command	Description
No protocol type is specified.	**nat-server server-mappings server-mapping name** *name* **global start-ip** *ip-address* [**end-ip** *ip-address*] **inside start-ip** *ip-address* [**end-ip** *ip-address*] [**reverse-enable** { **true** | **false** }]	When the **reverse-enable** parameter is specified, you can configure mappings between multiple public IP addresses specified by **global** and one private IP address specified by **inside**. In this case, however, the intranet server cannot access the Internet proactively. To rectify this, you can configure a source NAT policy between the security zone of the intranet server and that of the Internet to translate

Whether a Protocol Type Is Specified	Command	Description
A protocol type is specified.	**nat-server server-mappings server-mapping name** *name* **protocol** *protocol* **global start-ip** *ip-address* [**end-ip** *ip-address*] **inside start-ip** *ip-address* [**end-ip** *ip-address*] [**global-port start-port** *port-number* **end-port** *port-number*] [**inside-port start-port** *port-number* **end-port** *port-number*] [**reverse-enable** { **true** | **false** }]	the private IP address of the intranet server into a public IP address. The source NAT policy can reference an address specified by **global** or another public IP address. When the protocol type is set to **gre**, **esp**, **ah**, or **sctp**, **global-port** and **inside-port** cannot be configured simultaneously.

NOTE

When NAT Server and DNS ALG are used together, you are advised not to specify both the port and protocol in the NAT Server configuration simultaneously. Otherwise, DNS ALG address translation fails.

- If the public address configured in NAT Server and the WAN interface address are on different network segments, a blackhole route is required.
- If the public address configured in NAT Server and the WAN interface address are on the same network segment, a blackhole route is recommended.
- If the public address configured in NAT Server is consistent with the WAN interface address, a routing loop will not occur. In this case, a blackhole route is not required.

Step 6 Commit the configuration.

```
commit
```

Step 7 **Optional:** Configure the NAT ALG function.

In scenarios where the intranet server provides multi-channel protocol services (such as FTP), configure NAT ALG to translate random port numbers negotiated between the server and hosts. For details, see "ASPF/ALG Configuration" in CLI Configuration Guide > Security Configuration.

Step 8 Configure a route to the Internet on the intranet server or configure the IP address of the interface connecting DeviceA to the intranet server as the default gateway address. This allows the intranet server to send packets to DeviceA by default, after which DeviceA forwards them to Internet users.

----End

10.1.6.2.2 Example for Configuring NAT Server for Internet Users to Access Intranet Servers

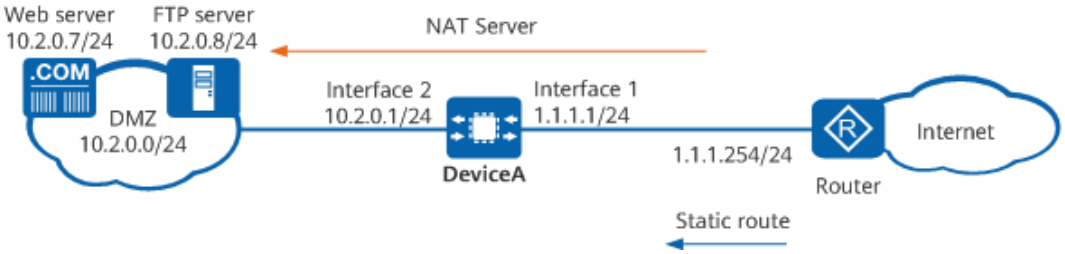
Networking Requirements

An enterprise has deployed DeviceA as a security gateway at the intranet border. NAT Server needs to be configured on DeviceA so that the web and FTP servers on the enterprise's private network can provide services to Internet users. In addition to the public IP address of the WAN interface on DeviceA, the enterprise has obtained another public IP address (1.1.1.10) from the ISP, which is used by intranet servers to provide services to Internet users. **Figure 10-12** shows the network diagram, in which the router is the access gateway provided by the ISP.

Figure 10-12 Network diagram of NAT Server

NOTE

In this example, interface 1 and interface 2 represent GE 0/0/1 and GE 0/0/2, respectively.



Item	Data	Description
GE0/0/1	IP address: 1.1.1.1/24	1.1.1.1/24 is a public address provided by the ISP.
GE0/0/2	IP address: 10.2.0.1/24	Intranet servers use 10.2.0.1 as the default gateway address.
NAT Server	Name: policy_web Public IP address: 1.1.1.10 Private IP address: 10.2.0.7 Public port: 8080 Private port: 80	When Internet users send traffic to 1.1.1.10 through port 8080, DeviceA can forward the traffic to the web server based on this mapping entry. The web server has the private address 10.2.0.7 and private port number 80.
	Name: policy_ftp Public IP address: 1.1.1.10 Private IP address: 10.2.0.8 Public port: 21 Private port: 21	When Internet users send traffic to 1.1.1.10 through port 21, DeviceA can forward the traffic to the FTP server based on this mapping entry. The FTP server has the private address 10.2.0.8 and private port number 21.

Item	Data	Description
Default route	Destination address: 0.0.0.0 Next-hop address: 1.1.1.254	Configure a default route to the Internet on DeviceA so that it can forward traffic from intranet servers to the ISP router.

Procedure

Step 1 Configure NAT Server.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> nat-server server-mappings server-mapping name policy_web
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]
MDCLI> protocol tcp
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]
MDCLI> global
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/global
MDCLI> start-ip 1.1.1.10
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/global
MDCLI> quit 1
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]
MDCLI> inside
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/inside
MDCLI> start-ip 10.2.0.7
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/inside
MDCLI> quit 1
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]
MDCLI> global-port
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/global-port
MDCLI> start-port 8080
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/global-port
MDCLI> quit 1
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]
MDCLI> inside-port
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/inside-port
MDCLI> start-port 80
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/inside-port
MDCLI> commit
[(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_web"]/inside-port
MDCLI> quti 255
[(gl)user@HUAWEI]
MDCLI> nat-server server-mappings server-mapping name policy_ftp
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]
MDCLI> protocol tcp
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]
MDCLI> global
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/global
MDCLI> start-ip 1.1.1.10
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/global
MDCLI> quit 1
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]
MDCLI> global-port
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/global-port
MDCLI> start-port 21
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/global-port
MDCLI> quit 1
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]
MDCLI> inside
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/inside
MDCLI> start-ip 10.2.0.8
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/inside
```



```
MDCLI> quit 1
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]
MDCLI> inside-port
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/inside-port
MDCLI> start-port 21
[*(gl)user@HUAWEI]/nat-server/server-mappings/server-mapping[name="policy_ftp"]/inside-port
MDCLI> commit
```

Step 2 Configure a default route on DeviceA to enable it to forward traffic from intranet servers to the ISP router.

Step 3 On the ISP router, configure a static route to the public address (1.1.1.10) of intranet servers, with the next-hop address set to 1.1.1.1. The ISP router can then forward traffic destined for the intranet server to DeviceA.

Contact the ISP network administrator to perform this step.

----End

Verifying the Configuration

1. Verify that Internet users can access the intranet servers.

11 Security Configuration

11.1 Local Attack Defense Configuration

11.1.1 Overview of Local Attack Defense

Definition

Local attack defense is a central processing unit (CPU) protection mechanism designed to ensure that the CPU can properly process normal services. The CPU of a device may receive both normal service packets and malicious packets targeting the CPU.

- If a large number of normal service packets are sent to the CPU, its usage surges. This severely impacts device performance, ultimately disrupting services.
- If the CPU is busy processing attack packets for an extended period, normal services will be interrupted, and in some cases even the entire system will crash.

To solve these issues, the local attack defense function is introduced. With this function enabled, the CPU can run properly even when receiving a large number of normal service packets or attack packets, ensuring normal service running.

Fundamentals

Local attack defense includes CPU attack defense and attack source tracing.

CPU attack defense

CPU attack defense can rate-limit packets sent to the CPU. This ensures that the CPU can properly process services. CPU attack defense protects the device by using a multi-level security mechanism, which provides the following policies:

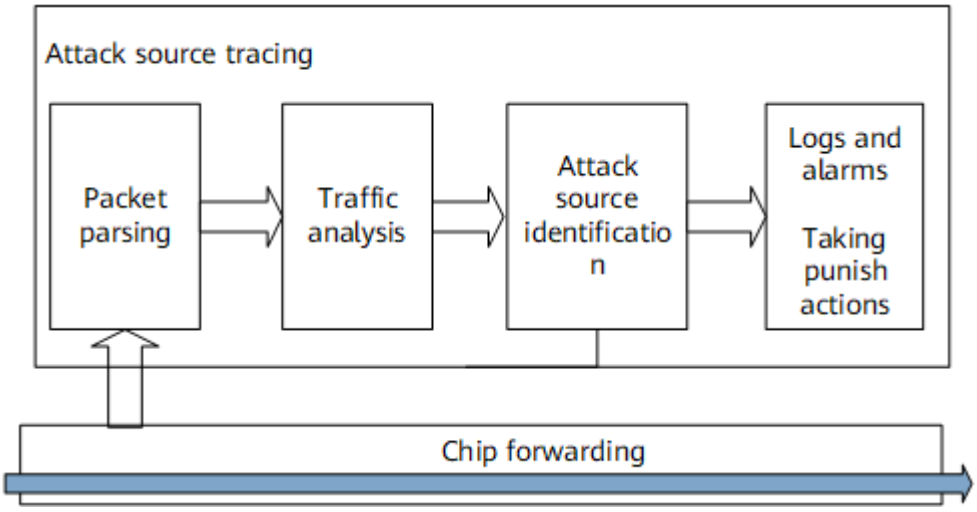
- Level 1: The device uses a blacklist to filter out unauthorized packets sent to the CPU.
- Level 2: The device rate-limits the packets sent to the CPU based on the protocol type, and discards excess packets based on the Control Plane Committed Access Rate (CPCAR) value. This prevents excess packets of a protocol from being sent to the CPU.
- Level 3: The device rate-limits all the packets sent to the CPU and randomly discards excess packets.

Attack source tracing

Attack source tracing defends against denial of service (DoS) attacks. A device enabled with attack source tracing analyzes packets sent to the CPU, collects statistics on the packets, and allows a user to set a packet rate threshold. Packets exceeding the threshold are considered attack packets. The device finds the source IP address or source MAC address by analyzing the attack packets, generates logs or alarms to alert network administrators, and discards the attack packets to punish the attack source.

As shown in **Figure 11-1**, attack source tracing involves four steps: packet parsing, traffic analysis, attack source identification, log and alarm generation as well as punishment action taking.

Figure 11-1 Attack source tracing process



In this process, the device finds the attack source, triggers an alarm when detecting the packet rate exceeds the threshold, and takes punishment actions to protect the CPU.

11.1.2 Configuration Precautions for Local Attack Defense

Licensing Requirements

Local Attack Defense is not under license control.

Hardware Requirements

Table 11-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

11.1.3 Default Settings for Local Attack Defense

Table 11-2, **Table 11-4**, and **Table 11-5** describe the default settings for local attack defense.

Table 11-2 Default settings for CPU attack defense

Parameter	Default Setting
Attack defense policy	No default policy
Blacklist	No blacklist
Rate limiting	Enabled. The device has a default rate limit for each type of protocol packets sent to the CPU. For details, see **Table 11-3**.
Rate limit for all packets sent to the CPU	1000 pps

Parameter	Default Setting
Dynamic adjustment of the default CPCAR value for protocol packets	Disabled Applicable packet types: ARP Request, unicast ARP Request, ARP Reply, DHCP Reply, and DHCP Request

Table 11-3 Default rate limits for protocol packets

Protocol Packet	Default Rate Limit (pps)
ntp	32
dns-request	32
dns-reply	50
dhcp-client	100
arp	128
arp-miss	64
icmp	64
igmp	64
lldp	64
arp-request	64
arp-reply	64
http-client	256
http-server	256
https-client	256
https-server	256
dhcp-server	128
ssh	256
ftp	256
pppoe-discovery	64
pppoe-session	512
ip-cloud	256
coap-server	64
coap-server-dtls	512

Table 11-4 Default settings for attack source tracing

Parameter	Default Setting
Attack defense policy	No default policy
Attack source tracing	Enabled
Attack source tracing mode	Source tracing based on source IP addresses and source MAC addresses
Alarm function for attack source tracing	Enabled
Alarm threshold for attack source tracing	128pps
Punishment for attack source tracing	Disabled
Punishment threshold for attack source tracing	128pps

Table 11-5 Default settings for dynamic CPCAR adjustment based on the CPU usage

CAR Protocol Packet	Description	Default CIR (CPU 65%)	Default CIR (CPU 85%)
arp-reply	ARP Reply packet	128 pps	64 pps
arp-request	ARP Request packet	128 pps	64 pps
arp-request-uc	Unicast ARP Request packet	128 pps	64 pps
dhcp-client	Message sent by a DHCP client	96 pps	64 pps
dhcp-server	Message sent by a DHCP server	96 pps	64 pps

11.1.4 Configuring CPU Attack Defense

Prerequisites

Before configuring CPU attack defense, you have completed the following tasks:

- Connect interfaces and set physical parameters for the interfaces to ensure that the physical status of the interfaces is up.
- Configure an ACL if a blacklist needs to be configured.

11.1.4.1 Creating an Attack Defense Policy

Context

You must first configure an attack defense policy before you can configure the local attack defense function in it.

For details about configuration parameters, see **huawei-host-security.yang**.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Configure an attack defense policy.

```
host-security policys policy name p_name
```

Step 3 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/** command to check the attack defense policy configuration.

11.1.4.2 (Optional) Configuring a Blacklist

Context

A blacklist is a group of users with particular characteristics. The device discards packets sent by users in the blacklist. You can apply an ACL to a blacklist.

For details about configuration parameters, see **huawei-host-security.yang**.

NOTE

- A maximum of one blacklist can be configured on the device.
- The ACL referenced by a blacklist can be a basic ACL, an advanced ACL, or a Layer 2 ACL.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the attack defense policy view.

```
host-security policys policy name p_name
```


Step 3 Configure a blacklist.

```
filters filter filter-id 1
```

By default, no blacklist is configured.

Step 4 Configure an ACL to be referenced by the blacklist.

```
ipv4-acl acl_group
```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/policy[name=*p_name*]/filters/** command to check the blacklist configuration.

11.1.4.3 (Optional) Configuring Dynamic CPCAR Adjustment

Context

When CPCAR values are manually set, a device cannot immediately control the rate of packets sent to the CPU when undergoing an attack. As a result, the CPU usage exceeds the acceptable range and CPU performance degrades.

After dynamic CPCAR adjustment is configured, the device periodically checks its CPU usage, and automatically adjusts CPCAR values of protocol packets when the CPU usage exceeds the preset upper threshold, reducing impact on the CPU.

For details about configuration parameters, see **huawei-host-security.yang**.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the dynamic CPCAR adjustment view.

```
host-security adjust-car
```

Step 3 Set the type of protocol packets for which the CPCAR value will be dynamically adjusted.

```
adjust-protocol-type { arp-request | arp-reply | arp-request-uc | dhcp-discovery | dhcp-reply | dhcp-request }
```

Step 4 Enable or disable dynamic CPCAR adjustment.

```
enable { true | false }
```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/adjust-car/** command to check the dynamic CPCAR adjustment configuration.

11.1.4.4 (Optional) Configuring a Rate Limit

Context

You can set rate limits for different types of packets to reduce the number of packets sent to the CPU and the impacts of different types of packets on each other.

For details about configuration parameters, see **huawei-host-security.yang**.

NOTE

If both dynamic CPCAR adjustment and a rate limit are configured for a type of protocol packet, dynamic CPCAR adjustment takes precedence.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the attack defense policy view.

```
host-security policys policy name p_name
```

Step 3 Configure protocol types for applying CPCAR to.

```
cpcars cpcar protocol-type { arp | arp-miss | arp-reply | arp-request | dhcp-reply | dhcp-request | dns-reply | dns-request | ftp | http-client | http-server | https-client | https-server | icmp | igmp | lldp | mqtt | ntp | ssh | coap-server | coap-server-dtls | pppoe-discovery | pppoe-session }
```

Step 4 Configure the rate limit, in pps.

```
cir pps
```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/policy[name=*p_name*]/cpcars/** command to check the rate limit information.

11.1.4.5 Applying an Attack Defense Policy

Context

After an attack defense policy is created, apply it so that it can take effect.

For details about configuration parameters, see **huawei-host-security.yang**.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the attack defense policy view.

```
host-security policys policy name p_name
```

Step 3 Apply the policy globally.

```
applied-policys applied-policy applied-type all
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/policy[name=*p_name*]/applied-policys/** command to check the configuration of the applied attack defense policy.

11.1.4.6 Verifying the CPU Attack Defense Configuration

Procedure

- Run the **display host-security/policys/policy[name=*p_name*]** command to check information about the attack defense policy.
- Run the **display host-security/packet-statistics/packet-statistic[slot=*slot_0*][packet-type=*arp*]** command to check CPCAR statistics.

----End

11.1.5 Configuring Attack Source Tracing

Pre-configuration Tasks

Before configuring attack source tracing, complete the following tasks:

Connect interfaces and set physical parameters for the interfaces to ensure that the physical status of the interfaces is up.

11.1.5.1 Creating an Attack Defense Policy

Context

You need to first configure an attack defense policy and then configure the local attack defense function in the policy.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Configure an attack defense policy.

```
host-security policys policy name p_name
```

Step 3 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/** command to check the attack defense policy configuration.

11.1.5.2 Configuring Attack Source Tracing

Context

Attackers may send a large number of attack packets targeting the CPUs of network devices. After a device is configured with attack source tracing and a threshold for attack source tracing, the device can analyze attack packets sent to the CPU. If the number of packets sent from an attack source in a specified period exceeds the threshold for attack source tracing, the device sends logs to notify the administrator, who can then take protective measures.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the attack defense policy view.

```
host-security policys policy name p_name
```

Step 3 Enter the attack source tracing view.

```
auto-defend
```

Step 4 (Optional) Enable attack source tracing.

```
enable { true | false }
```

Step 5 (Optional) Enable the alarm function for attack source tracing.

```
alarm-enable { true | false }
```

Step 6 (Optional) Set the alarm threshold for attack source tracing.

```
alarm-threshold pps
```

Step 7 (Optional) Enable the punishment function for attack source tracing.

```
penalty-enable { true | false }
```

Step 8 (Optional) Set the punishment threshold for attack source tracing.

```
penalty-threshold pps
```

Step 9 (Optional) Configure the attack source tracing mode. Currently, the device supports two source tracing modes: source tracing based on source IP addresses and source tracing based on source MAC addresses.

```
defend-type { source-ip | source-mac } [ { source-ip | source-mac } ]
```

Step 10 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/policy[name=*p_name*]/auto-defend/** command to check the attack defense policy configuration.

11.1.5.3 Applying an Attack Defense Policy

Context

After an attack defense policy is created, apply it so that it can take effect.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the attack defense policy view.

```
host-security policys policy name p_name
```

Step 3 Apply the attack defense policy globally.

```
applied-policys applied-policy applied-type all
```

Step 4 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display host-security/policys/policy[name=*p_name*]/applied-policys/** command to check the configuration of the applied attack defense policy.

11.1.5.4 Verifying the Attack Source Tracing Configuration

Procedure

- Run the **display host-security/policys/policy[name=*p_name*]** command to check information about the attack defense policy.
- Run the **display host-security/policys/policy[name=*p_name*]/auto-defend/** command to check the attack source tracing configuration.

----End

11.1.6 Configuration Examples for Local Attack Defense

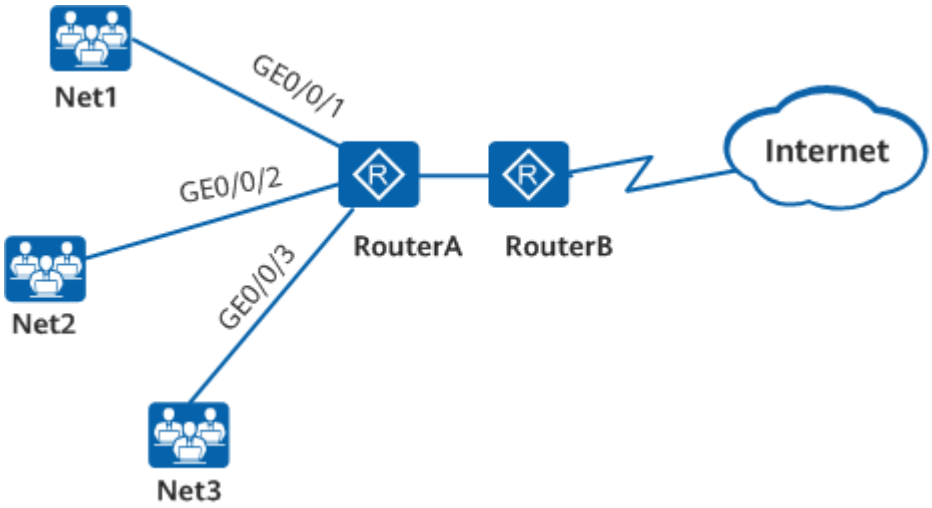
Networking Requirements

In **Figure 11-2**, users on different LANs access the Internet through DeviceA. To locate attacks on DeviceA, attack source tracing needs to be configured to trace the attack source. The network administrator finds that:

- A user on network segment Net1 frequently initiates attacks.
- The attacker sends a large number of ARP Request packets, degrading CPU performance.

The administrator needs to perform configurations on DeviceA to solve the preceding problems.

Figure 11-2 Network diagram of local attack defense



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure a blacklist and add the attacker on Net1 to the blacklist to prevent the attacker from accessing the network.
2. Configure the rate limit for ARP Request packets sent to the CPU. This limits the rate of ARP Request packets within a small range, and thereby reduces the impact on CPU processing of normal services.
3. Disable the Telnet server function on DeviceA so that DeviceA discards received Telnet packets.

Procedure

Step 1 Configure an ACL to be referenced by the blacklist.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> acl groups group identity 2001

[*(gl)user@HUAWEI]/acl/groups/group[identity="2001"]
MDCLI> rule-basics rule-basic name r1

[*(gl)user@HUAWEI]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="r1"]
MDCLI> source-ipaddr 192.168.1.5 source-wild 0.0.0.255 action permit

[*(gl)user@HUAWEI]/acl/groups/group[identity="2001"]/rule-basics/rule-basic[name="r1"]
MDCLI> commit
```

Step 2 Create an attack defense policy.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security policys policy name p_name
```

Step 3 Configure CPCAR rate limiting.

```
[user@HUAWEI]
MDCLI> edit-config
```



```
[(gl)user@HUAWEI]
MDCLI> host-security policys policy name p_name

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]
MDCLI> cpcars cpcar protocol-type arp-request

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/cpcars/cpcar[protocol-type="arp-request"]
MDCLI> cir 1000

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/cpcars/cpcar[protocol-type="arp-request"]
MDCLI> commit
```

Step 4 Configure attack source tracing.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security policys policy name p_name

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]
MDCLI> auto-defend

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/auto-defend
MDCLI> enable true alarm-enable true alarm-threshold 128 penalty-threshold 512

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/auto-defend
MDCLI> defend-type source-ip source-mac

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/auto-defend
MDCLI> commit
```

Step 5 Configure a blacklist.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security policys policy name p_name

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]
MDCLI> filters filter filter-id 1

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/filters/filter[filter-id="1"]
MDCLI> ipv4-acl 2001

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/filters/filter[filter-id="1"]
MDCLI> commit
```

Step 6 Apply the attack defense policy.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security policys policy name p_name

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]
MDCLI> applied-policys applied-policy applied-type all

[*(gl)user@HUAWEI]/host-security/policys/policy[name="p_name"]/applied-policys/applied-policy[applied-type="all"]
MDCLI> commit
```

Step 7 Check the configured attack defense policy.

```
[user@HUAWEI]
MDCLI> display host-security/policys/policy[name=p_name]
{
```



```
 "name": "p_name",
 "filters": {
  "filter": [
   {
    "filter-id": 1,
    "ipv4-acl": "2001"
   }
  ]
 },
 "cpcars": {
  "cpcar": [
   {
    "protocol-type": "arp-request"",
    "cir": 1000
   }
  ]
 },
 "auto-defend": {
  "enable": true,
  "alarm-enable": true,
  "alarm-threshold": 128,
  "penalty-threshold": 512,
  "defend-type": [
   "source-ip",
   "source-mac"
  ]
 },
 "applied-policys": {
  "applied-policy": [
   {
    "applied-type": "all"
   }
  ]
 }
}
```


----End

11.2 Attack Defense Configuration

This chapter describes how to configure attack defense to protect networks. Attack defense allows a device to identify various types of network attacks and protect itself and the connected network against malicious attacks to ensure device and network operation.

11.2.1 Overview of Attack Defense

Definition

Attack defense is a network security feature that enables a device to analyze the content and behavior of packets sent to the CPU, identify attack packets, and take measures to block attack packets.

Attack defense can help defend against malformed packet attacks, fragmentation attacks, and flood attacks.

Purpose

Attacks initiated by utilizing inherent bugs of communication protocols or improper network deployment have great impact on networks. In particular, attacks on a network device can cause the device or network to crash.

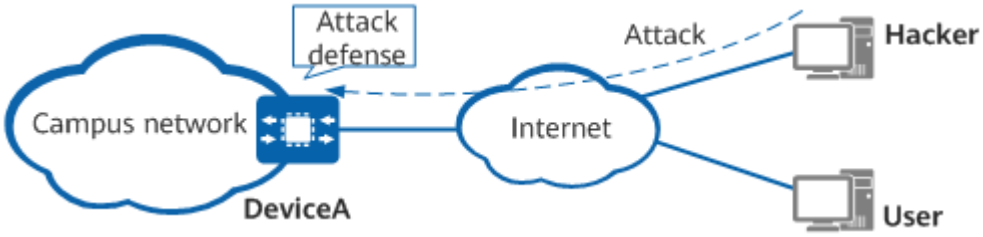
The attack defense feature discards or limits the rate of various attack packets sent to the CPU, protecting the device and ensuring normal services.

11.2.2 Understanding Attack Defense

As shown in **Figure 11-3**, when DeviceA undergoes attacks, its CPU usage surges and network services are affected. To safeguard network services, the following attack defense functions are configured on DeviceA:

- Defense against malformed packet attacks.
- Defense against fragmentation attacks, which rate-limits packet fragments to prevent them from overloading the CPU and device.
- Defense against flood attacks, including:
  - Defense against TCP SYN flood attacks, which rate-limits TCP SYN packets to protect the CPU.
  - Defense against UDP flood attacks, which enables the device to discard UDP packets sent from specified ports.
  - Defense against ICMP flood attacks, which rate-limits ICMP flood attack packets to protect the CPU.

Figure 11-3 Network diagram of attack defense



11.2.2.1 Defense Against Malformed Packet Attacks

A malformed packet attack is a type of attack in which malformed IP packets are sent to a target device, causing the device to encounter an error or even crash when processing such packets. Defense against malformed packet attacks enables a device to detect and discard malformed packets in real time to protect itself.

Malformed packet attacks are classified into the following types.

Flood Attack from IP Null Payload Packets

An IP null payload packet has only a 20-byte IP header, and no data section. Attackers often carry out flood attacks by sending a large number of packets with only IP headers, known as IP null payload packets. Processing such packets causes devices to experience faults or even crash.

After defense against malformed packet attacks is enabled, a device directly discards the received IP null payload packets.

Attack from IGMP Null Payload Packets

An IGMP packet consists of a 20-byte IP header and an 8-byte IGMP body. An IGMP null payload packet consists of less than 28 bytes. When a network device processes IGMP null payload packets, it may experience a fault or even crash.

After defense against malformed packet attacks is enabled, the device directly discards the received IGMP null payload packets.

LAND Attack

A Local Area Network Denial (LAND) attacker targets the defects in the three-way handshake mechanism of TCP, sending a SYN packet in which both the source and destination addresses are the target host's address and the source and destination ports are the same. After receiving the SYN packet, the target host creates a null TCP connection by using its own address as both the source and destination addresses, and retains this connection until it times out. The target host will create many null TCP connections after receiving a large number of such SYN packets, wasting network resources or even crashing the system.

After defense against malformed packet attacks is enabled, the device checks source and destination addresses in TCP SYN packets. The device considers TCP SYN packets with the same source and destination addresses as malformed packets and discards them.

Smurf Attack

An attacker sends an ICMP Request packet with the target host's address as the source address and the broadcast address of the target network as the destination address. This results in all hosts on the target network receiving the ICMP Request packet, after which they send ICMP Reply packets to the target host. This inevitably leads to the target host being flooded with packets and overloaded. In some cases, the entire device or network crashes.

With defense against malformed packet attacks enabled, the device checks whether the destination addresses in ICMP Request packets are the broadcast or subnet broadcast addresses and discards them if they are either one.

Attack from Packets with Invalid TCP Flag Bits

A TCP packet contains six flag bits: URG, ACK, PSH, RST, SYN, and FIN. Different systems respond differently based on the combination of these flag bits.

- If the six flag bits are all 1s, the attack is a Christmas tree attack. A device subject to a Christmas tree attack may crash.
- An attacker sends a TCP packet in which SYN and FIN are 1 to a target host. If the receiving interface is disabled, the receiver replies with an RST | ACK message. If the receiving interface is enabled, the receiver replies with an SYN | ACK message. Such an attack is used to detect whether a host is online or offline and whether an interface is enabled or disabled.
- An attacker sends a TCP packet in which the six flag bits are all 0s to a target host. If the receiving interface is disabled, the receiver replies with an RST | ACK message, which can be used by the attacker to detect whether the host is online or offline. If the receiving interface is enabled, the target host does not

respond if running Linux or UNIX but replies with an RST | ACK message if running Windows. This attack is used to detect the type of operating system on the target host.

After defense against malformed packet attacks is enabled, the device checks each flag bit in TCP packets to prevent attacks from packets with invalid TCP flag bits. If any of the following conditions is met, the device discards the TCP packets:

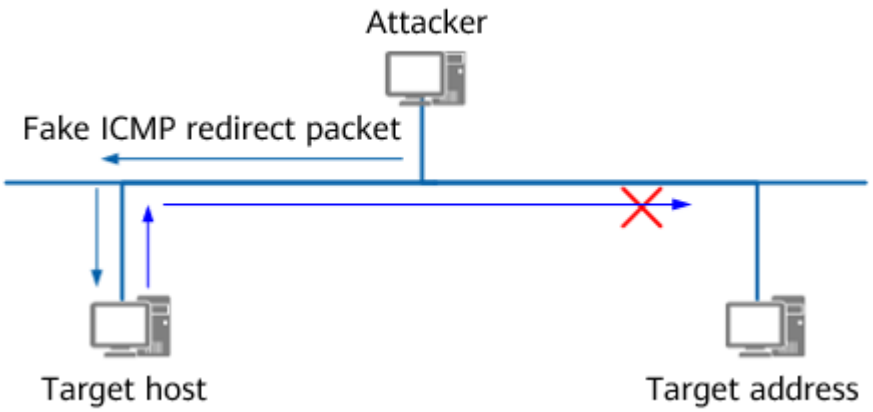
- The six flag bits are all 1s.
- Both SYN and FIN are 1.
- The six flag bits are all 0s.

ICMP Redirect Attack

An ICMP redirect attack is similar to an ICMP unreachable attack. ICMP redirect packets can be sent by network devices to a host in the same subnet, requesting the host to change its routes.

An attacker can utilize this by sending a fake redirect packet to the target host on another network segment, requesting the target host to modify the routing table. The attack changes routes on the target host and affects packet forwarding, as shown in **Figure 1**.

Figure 11-4 ICMP redirect attack



IP Header Length Attack

An attacker sends packets with abnormal IP header lengths.

11.2.2.2 Defense Against Fragmentation Attacks

A fragmentation attack is an attack in which error fragments are sent to a target device, which then crashes, restarts, or consumes a large amount of CPU resources when processing such fragments. Fragmentation attack defense enables a device to detect packet fragments in real time and discard or rate-limit them to protect the device.

Fragmentation attacks are classified into the following types.

Excess-Fragment Attack

The offset of IP packets is measured in units of 8-byte blocks. Normally, an IP header has 20 bytes and the maximum payload of an IP packet is 65515 bytes. Therefore, an IP packet can be fragmented into up to 8189 fragments, above which the device would consume a large amount of CPU resources when reassembling packets. Currently, the device can reassemble a maximum of 64 fragments; excess fragments are discarded.

With defense against fragmentation attacks enabled, the device considers a packet with over 8189 fragments malicious and discards all of its fragments.

The device defends against excess-fragment attacks regardless of whether defense against fragmentation attacks is enabled.

Excess-Offset Attack

An attacker sends a fragment with a large offset value to a target device, which then allocates a large memory space to store all fragments, overloading the device.

The maximum offset value is 65528. Generally, the offset value does not exceed 8190. If the offset value is 8189 multiplied by 8 and the IP header is 20, the last fragment can have only a 3-byte IP payload. As such, the maximum value of the offset is generally 8189. The device considers packets with an offset value larger than 8190 malicious and discards them.

After defense against fragmentation attacks is enabled, the device checks whether the offset value multiplied by 8 is greater than 65528. If so, the device considers the fragments malicious and discards them.

If defense against fragmentation attacks is enabled, the device directly discards the current fragment. If defense against fragmentation attacks is disabled, the device does not defend against excess-offset attacks.

Repeated Fragment Attack

A repeated fragment attack is an attack in which fragments are sent to a target host multiple times. Such an attack is carried out in one of two ways:

- The same fragments are sent multiple times, causing high CPU usage or a memory error on the target host.
- Different fragments with the same offset are sent. In this case, the target host cannot determine which fragment will be reserved and which fragment will be discarded, or whether all fragments need to be discarded. As a result, the target host may experience high CPU usage or a memory error.

With defense against fragmentation attacks enabled, the device applies the committed access rate (CAR) limit to packet fragments, reserves the first repeated fragment, and discards all the remaining repeated fragments to protect the CPU.

The device discards repeated fragments regardless of whether defense against fragmentation attacks is enabled.

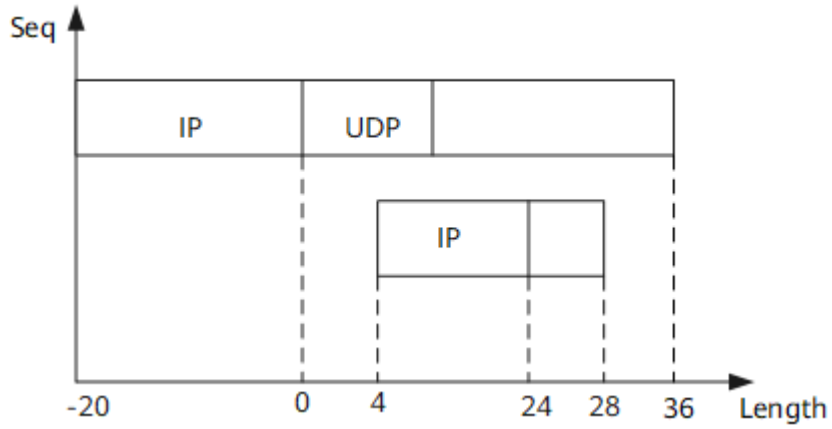
Teardrop Attack

A Teardrop attack is the most well-known IP fragmentation attack, which exploits the mechanism of IP fragmentation to put the second fragment into the first. The offset of the second fragment is smaller than that of the first fragment, and the offset plus the data field of the second fragment does not exceed the tail of the first fragment.

As shown in **Figure 11-5**:

- The IP payload in the first fragment is 36 bytes, the total length of the fragment is 56 bytes, the protocol is UDP, and the UDP checksum is 0 (unchecked).
- The IP payload in the second fragment is 4 bytes, the total length of the fragment is 24 bytes, the protocol is UDP, and the offset is 24 (should be 36).

Figure 11-5 Teardrop attack



Teardrop attacks cause the system to restart or even crash. With defense against fragmentation attacks enabled, the device discards all fragments in a Teardrop attack.

If defense against fragmentation attacks is enabled, the device discards all fragments in the case of fragment overlapping. If defense against fragmentation attacks is disabled, the device discards the overlapping fragments and retains the previous fragments.

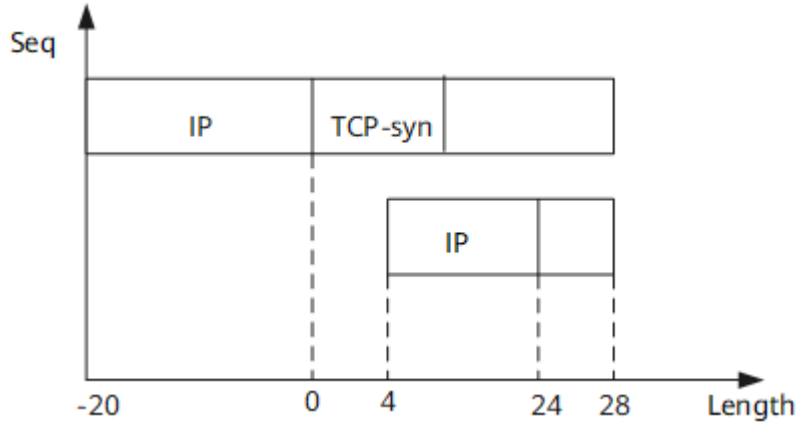
Syndrop Attack

A Syndrop attack is similar to a Teardrop attack, except for the fact that a Syndrop attack uses TCP packets that carry a SYN flag and also IP payload.

As shown in **Figure 11-6**:

- The IP payload in the first fragment is 28 bytes, and the IP header is 20 bytes.
- The IP payload in the second fragment is 4 bytes, the IP header is 20 bytes, and the offset is 24 (should be 28).

Figure 11-6 Syndrop attack



Syndrop attacks cause the system to restart or even crash. With defense against fragmentation attacks enabled, the device discards all fragments in a Syndrop attack.

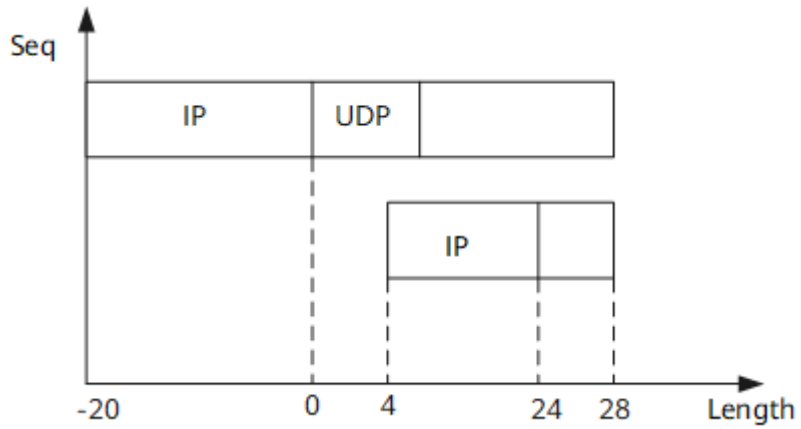
If defense against fragmentation attacks is enabled, the device discards all fragments in the case of fragment overlapping. If defense against fragmentation attacks is disabled, the device discards the overlapping fragments and retains the previous fragments.

NewTear Attack

A NewTear attack uses error fragments. As shown in **Figure 11-7**, the protocol is UDP.

- The IP payload in the first fragment is 28 bytes including the UDP header. The UDP checksum is 0.
- The IP payload in the second fragment is 4 bytes, and the offset is 24 (should be 28).

Figure 11-7 NewTear attack



NewTear attacks cause the system to restart or even crash. With defense against fragmentation attacks enabled, the device discards all fragments in a NewTear attack.

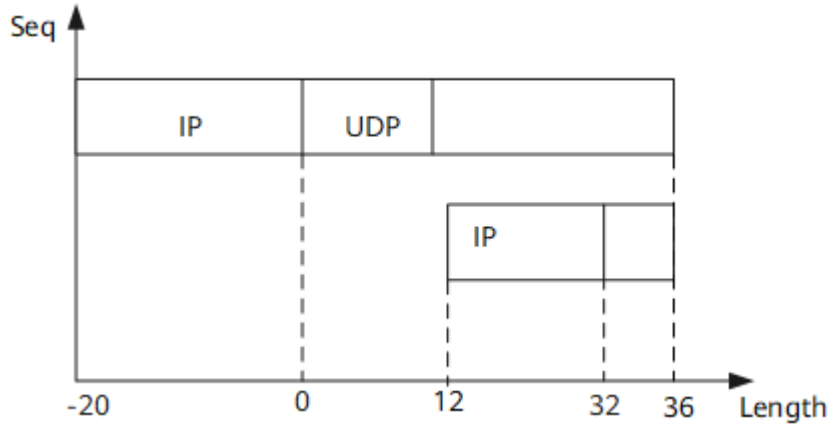
If defense against fragmentation attacks is enabled, the device discards all fragments in the case of fragment overlapping. If defense against fragmentation attacks is disabled, the device discards the overlapping fragments and retains the previous fragments.

Bonk Attack

A Bonk attack uses error fragments. As shown in **Figure 11-8**, the protocol is UDP.

- The IP payload in the first fragment is 36 bytes including the UDP header. The UDP checksum is 0.
- The IP payload in the second fragment is 4 bytes, and the offset is 32 (should be 36).

Figure 11-8 Bonk attack



Bonk attacks cause the system to restart or even crash. With defense against fragmentation attacks enabled, the device discards all fragments in a Bonk attack.

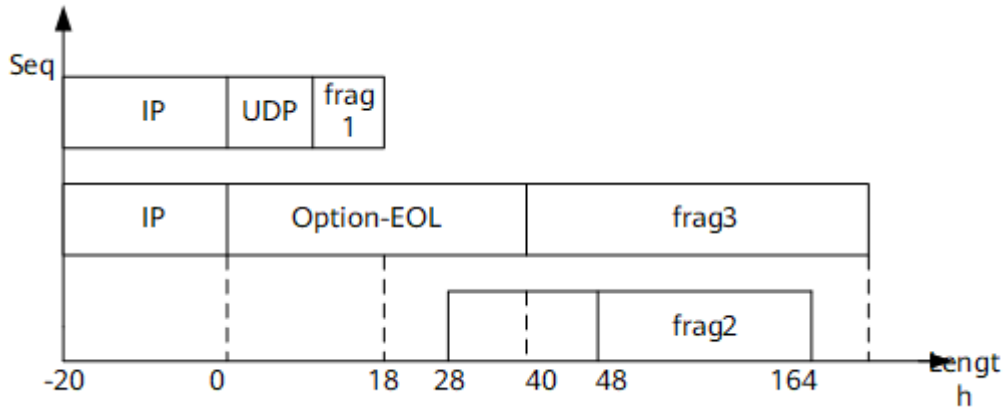
If defense against fragmentation attacks is enabled, the device discards all fragments in the case of fragment overlapping. If defense against fragmentation attacks is disabled, the device discards the overlapping fragments and retains the previous fragments.

Nesta Attack

A Nesta attack uses error fragments. As shown in **Figure 11-9**:

- The IP payload in the first fragment is 18 bytes, the protocol used is UDP, and the checksum is 0.
- The offset in the second fragment is 48 and the IP payload is 116 bytes.
- The offset in the third fragment is 0, the More Frag field is 1 (meaning there are more fragments), the IP option (all EOLs) is 40 bytes, and the IP payload is 224 bytes.

Figure 11-9 Nesta attack



Nesta attacks cause the system to restart or even crash. With defense against fragmentation attacks enabled, the device discards all fragments in a Nesta attack.

If defense against fragmentation attacks is enabled, the device discards all fragments in the case of fragment overlapping. If defense against fragmentation attacks is disabled, the device discards the overlapping fragments and retains the previous fragments.

Rose Attack

In a Rose attack, the protocol of IP packets may be UDP or TCP.

As shown in **Figure 11-10**:

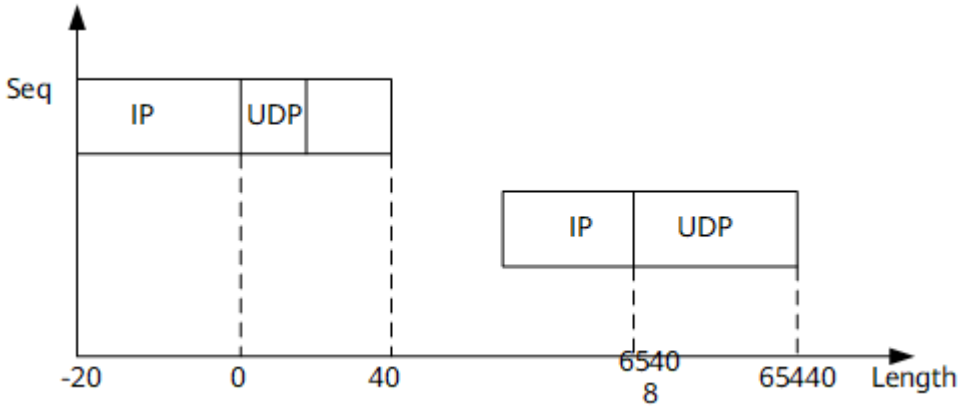
If the protocol is TCP:

- The IP payload in the first fragment is 48 bytes (including the TCP header) and the IP header is 20 bytes.
- The IP payload in the second fragment is 32 bytes, the offset is 65408, and the More Frag field is 0 (meaning the end fragment).

If the protocol is UDP:

- The IP payload in the first fragment is 40 bytes (including the UDP header, with UDP checksum 0), and the IP header is 20 bytes.
- The IP payload in the second fragment is 32 bytes, the offset is 65408, and the More Frag field is 0 (meaning the end fragment).

Figure 11-10 Rose attack



Rose attacks cause the system to restart or even crash. With defense against fragmentation attacks enabled, the device discards all fragments in a Rose attack.

If defense against fragmentation attacks is enabled, the device discards all the fragments.

Ping of Death Attack

A Ping of Death attack is an attack in which ICMP packets with the data field longer than 65507 bytes are sent to a target device. Such an attack leverages the fact that if a device processes ICMP packets that have more than 65507 bytes in the data field, the protocol stack may crash.

With defense against fragmentation attacks enabled, the device discards ICMP packets that have more than 65507 bytes in the data field.

If defense against fragmentation attacks is enabled, the device directly discards the current fragment. If defense against fragmentation attacks is disabled, the device does not defend against Ping of Death attacks.

Jolt Attack

A Jolt attack is an attack in which packets longer than 65535 bytes are sent to a target device. Jolt attack packets have 173 fragments, and the IP payload in each fragment is 380 bytes. That equates to a total length of 65760 (173 x 380 + 20) bytes, which is greater than 65535. Improperly processing such packets may cause the device to stop responding, restart, or even crash. Currently, the device can reassemble a maximum of 64 fragments; excess fragments are discarded.

With defense against fragmentation attacks enabled, the device discards Jolt attack packets.

The device defends against Jolt attacks regardless of whether defense against fragmentation attacks is enabled.

11.2.2.3 Defense Against Flood Attacks

If an attacker sends a large number of bogus packets to a target device in a short period of time, the target device becomes overloaded and cannot process normal services.

Defense against flood attacks enables a device to detect flood packets in real time and discard or rate-limit them to protect the device.

Flood attacks include TCP SYN flood attacks, UDP flood attacks, and ICMP flood attacks.

TCP SYN Flood Attack

A TCP SYN flood attack exploits the vulnerability of the TCP three-way handshake. During the TCP three-way handshake, when the receiver receives the initial SYN packet from the sender, it returns a SYN+ACK packet to the sender. The connection remains in a half-open state while the receiver waits for the final ACK packet from the sender. If the receiver does not receive the ACK packet, it retransmits a SYN +ACK packet to the sender. Finally, after several retransmission attempts, the receiver shuts down the session and then updates the session in memory. The period from the first SYN+ACK packet being sent to session teardown is approximately 30 seconds.

During this period, an attacker may send thousands of SYN packets to all open interfaces, and never responds to SYN+ACK packets from the receiver. This causes memory overloading on the receiver and prevents the receiver from accepting new connection requests. The receiver then disconnects all existing connections.

With defense against TCP SYN flood attacks enabled, the device rate-limits TCP SYN packets to ensure that system resources are not exhausted if an attack occurs.

UDP Flood Attack

A UDP flood attack is an attack in which a large number of UDP packets are sent to a target device within a short time, causing the target device to be busy with these UDP packets and fail to process normal services. After defense against flood attacks is enabled, the device considers UDP packets whose rate exceeds the threshold as UDP flood attack packets and discards them.

UDP flood attacks are classified into two types:

- Fraggle attack

  In a Fraggle attack, an attacker sends UDP packets where the source address is the target host's address, the destination address is the broadcast address of the target network, and the destination port is port 7. If multiple hosts use UDP echo services on the broadcast network, the target host receives excessive response packets from these hosts and becomes occupied while processing these packets.
- UDP diagnosis port attack

  In a UDP diagnosis port attack, an attacker sends a large number of packets to the target device's UDP diagnosis ports (such as 7-echo, 13-daytime, and 19-Chargen), causing flooding and affecting normal device running.

ICMP Flood Attack

A network administrator typically monitors a network and rectifies faults using the ping tool as follows:

1. The source host sends an ICMP Echo message to a target device.

2. Upon receiving the ICMP Echo message, the target device sends an ICMP Echo Reply message to the source host.

If the target device receives many ICMP Echo messages from an attacker, it becomes occupied and unable to process other data packets. As a result, normal services are affected.

A device can apply the committed access rate (CAR) limit to ICMP packets, thereby protecting the CPU.

11.2.3 Default Settings for Attack Defense

Table 11-6 describes the default settings for attack defense.

Table 11-6 Default settings for attack defense

Parameter	Default Setting
Defense against malformed packet attacks	Enabled
Defense against fragmentation attacks	Enabled
Defense against TCP SYN flood attacks	Enabled
Rate limit for TCP SYN flood packets	1000 pps
Defense against UDP flood attacks	Enabled
Rate limit for UDP flood packets	1000 pps
Defense against ICMP flood attacks	Enabled
Rate limit for ICMP flood packets	1000 pps

11.2.4 Configuring Attack Defense

Context

Attack defense enables a device to discard or rate-limit various attack packets sent to the CPU, protecting the device and ensuring normal service running.

For details about configuration parameters, see **huawei-host-security.yang**.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Specify packets requiring attack defense and enter the attack defense view of the packets.

```
host-security anti-attacks anti-attack anti-attack-type { abnormal | fragment | tcp-syn | udp-flood | icmp-flood }
```

Step 3 Enable or disable attack defense for the packets.

```
enable { true | false }
```

Step 4 Set the threshold for flood attack packets. Only thresholds for tcp-syn, udp-flood, and icmp-flood packets can be set.

```
para pps
```

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

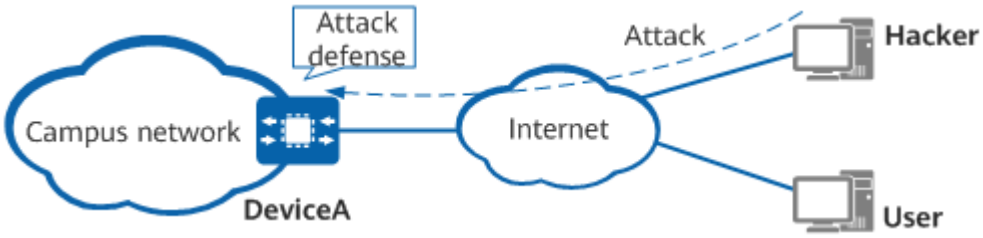
Run the **display host-security/anti-attacks/[anti-attack[anti-attack-type={ abnormal | fragment | tcp-syn | udp-flood | icmp-flood }]]** command to check the attack defense configuration.

11.2.5 Example for Configuring Attack Defense

Networking Requirements

As shown in **Figure 11-11**, if a hacker on a LAN launches a malformed packet attack, a fragmentation attack, or a flood attack on DeviceA, DeviceA may break down. To prevent such issues, the network administrator needs to take attack defense measures on DeviceA, so as to secure the network environment and ensure services run as normal.

Figure 11-11 Network diagram of attack defense



Configuration Roadmap

1. Configure defense against malformed packet attacks.
2. Configure defense against fragmentation attacks.
3. Configure defense against TCP SYN flood attacks.
4. Configure defense against UDP flood attacks.

5. Configure defense against ICMP flood attacks.

Procedure

Step 1 Configure defense against malformed packet attacks.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security anti-attacks anti-attack anti-attack-type abnormal

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="abnormal"]
MDCLI> enable true

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="abnormal"]
MDCLI> commit
```

Step 2 Configure defense against fragmentation attacks.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security anti-attacks anti-attack anti-attack-type fragment

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="fragment"]
MDCLI> enable true

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="fragment"]
MDCLI> commit
```

Step 3 Configure defense against TCP SYN flood attacks and set the rate limit for receiving TCP SYN packets to 1500 pps.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security anti-attacks anti-attack anti-attack-type tcp-syn

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="tcp-syn"]
MDCLI> enable true para 1500

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="tcp-syn"]
MDCLI> commit
```

Step 4 Configure defense against UDP flood attacks and set the rate limit for receiving UDP packets to 150 pps.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI> host-security anti-attacks anti-attack anti-attack-type udp-flood

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="udp-flood"]
MDCLI> enable true para 150

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="udp-flood"]
MDCLI> commit
```

Step 5 Configure defense against ICMP flood attacks and set the rate limit for receiving ICMP flood packets to 1500 pps.

```
[user@HUAWEI]
MDCLI> edit-config

[(gl)user@HUAWEI]
MDCLI>  host-security anti-attacks anti-attack anti-attack-type icmp-flood
```



```
[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="icmp-flood"]
MDCLI> enable true para 1500

[*(gl)user@HUAWEI]/host-security/anti-attacks/anti-attack[anti-attack-type="icmp-flood"]
MDCLI> commit
```

----End

Verifying the Configuration

Run the **display host-security/anti-attacks/** command to check the attack defense configuration.

```
[user@HUAWEI]
MDCLI> display host-security/anti-attacks/
{
  "anti-attack": [
    {
      "anti-attack-type": "abnormal",
      "enable": true
    },
    {
      "anti-attack-type": "fragment",
      "enable": true
    },
    {
      "anti-attack-type": "tcp-syn",
      "enable": true,
      "para": 1500
    },
    {
      "anti-attack-type": "udp-flood",
      "enable": true,
      "para": 150
    },
    {
      "anti-attack-type": "icmp-flood",
      "enable": true,
      "para": 1500
    }
  ]
}
```

11.3 Storm Suppression Configuration

11.3.1 Overview of Storm Suppression

Definition

Storm suppression is used to control broadcast packets, unknown multicast packets, and unknown unicast packets, thereby preventing broadcast storms that they may cause.

Storm suppression includes traffic suppression and storm control.

- Traffic suppression limits the rate of broadcast packets, unknown multicast packets, or unknown unicast packets by setting a threshold. When traffic exceeds the set threshold, the system discards excessive traffic, limiting the traffic rate to a reasonable range. Note that traffic suppression can also block outgoing traffic on interfaces.

- Storm control interrupts broadcast, unknown multicast, and unknown unicast packets by blocking these packets or disabling related interfaces. Storm control can also control the average rate of packets by suppressing them. When traffic exceeds the specified threshold, the system performs a predefined storm control action.

Table 1 compares traffic suppression and storm control.

Table 11-7 Comparison between traffic suppression and storm control

Item	Traffic Suppression	Storm Control
Traffic control	• If traffic suppression is configured in the outbound direction of an interface, the system blocks all traffic of the corresponding packet type. • In other cases, the system discards the traffic exceeding the threshold and allows the packets within the threshold to pass through.	• If the storm control action is configured to suppress packets, when the average rate of packets received on the interface exceeds the configured upper threshold, the system discards excess traffic until the average rate of packets drops below the threshold. • In other cases, when traffic exceeds the specified threshold, the system blocks the traffic received by the interface or shuts down the interface.
Traffic detection	• Chip-based detection • If traffic exceeds the threshold, traffic suppression takes effect immediately.	• Software-based detection • If the average packet rate exceeds the threshold within the detection interval, storm control takes effect.

NOTE

The device supports only traffic suppression.

Purpose

When a Layer 2 Ethernet interface on a device receives broadcast, unknown multicast, or unknown unicast packets, the device forwards these packets to other Layer 2 Ethernet interfaces in the same VLAN if the outbound interfaces cannot be determined based on the destination MAC addresses of these packets. As a result,

a broadcast storm may be generated, degrading forwarding performance of the device.

When an Ethernet interface on a device receives known unicast packets, heavy traffic of a certain type of packets may prevent the device from properly processing other services.

Traffic suppression and storm control can effectively control the traffic of these types of packets.

11.3.2 Configuration Precautions for Storm Suppression

Licensing Requirements

Storm Suppression is not under license control.

Hardware Requirements

Table 11-8 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

Table 11-9 Feature requirements

Feature	Feature Requirements	Series	Models
Supporting Traffic Suppression	Traffic suppression takes effect only for Layer 2 traffic.	S380-H series S380-L series S380-S series	S380-H8T3ST S380-L4P1T/ S380-L4T1T S380-S8P2T/ S380-S8T2T

Feature	Feature Requirements	Series	Models
Supporting Traffic Suppression	1. If the configured traffic rate limit is smaller than the lower limit of the device chip, the minimum rate limit value of the device chip takes effect. For example, if the minimum rate limit of a device chip is 32 pps, when the configured rate limit is smaller than 32 pps, the chip uses 32 pps for traffic suppression. 2. When the chip uses the two-bucket algorithm, a buffer PBS is calculated based on the configured rate, and rate limiting is triggered only when the rate is greater than the configured PPS and the PBS is reached. 3. There are some errors in the accuracy of chip-based rate limiting due to the inherent constraints of hardware chips. For details about the error rate, see the manuals of each chip vendor.	S380-H series S380-L series S380-S series	S380-H8T3ST S380-L4P1T/ S380-L4T1T S380-S8P2T/ S380-S8T2T

11.3.3 Default Settings for Storm Suppression

Table 11-10 describes the default settings for storm suppression.

Table 11-10 Default settings for storm suppression

Parameter	Default Setting
Traffic suppression in the inbound direction of an interface	Enabled
Traffic suppression mode in the inbound direction of an interface	Traffic suppression by limiting the packets per second (pps).
Traffic suppression statistics collection	Global and interface-based statistics collection. VLAN-based statistics collection is not supported.

11.3.4 Configuring Traffic Suppression

11.3.4.1 Understanding Traffic Suppression

Traffic suppression limits the rate of broadcast, unknown multicast, and unknown unicast packets through threshold setting. When traffic exceeds the set threshold, the system discards excessive traffic, limiting the traffic rate to a reasonable range.

Traffic suppression limits broadcast, unknown multicast, or unknown unicast packets based on the packet rate:

- In the inbound direction of all interfaces, the device supports traffic suppression for broadcast, unknown multicast, and unknown unicast packets.
- The device monitors the rate of all types of packets. When the rate of incoming packets exceeds the threshold, the device discards excess packets.

The device also supports the following traffic suppression function:

Traffic suppression for ICMP packets: The device rate-limits ICMP packets based on the specified threshold using the Control Plane Committed Access Rate (CPCAR) function.

Based on the statistics collection mode, traffic suppression can be classified into global traffic suppression and interface-based traffic suppression.

11.3.4.2 Configuring Traffic Suppression in the Inbound Direction

Context

To prevent broadcast storms, you can configure traffic suppression in the inbound direction of an interface. The device supports traffic suppression for broadcast, unknown multicast, and unknown unicast packets based on the packet rate. When the packet rate of any of these packet types exceeds the threshold, the system discards excess packets to ensure the packet rate is within an appropriate range.

For details about configuration parameters, see **huawei-storm-control.yang**.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 (Optional) Enter the view of the interface on which traffic suppression needs to be configured.

```
ifm interfaces interface name GE0/0/1
```

Step 3 Configure a type of packets for which traffic suppression needs to be configured and a rate range of the packets. The unit is pps.

```
storm-control storm-rates storm-rate packet-type { broadcast | unknown-multicast | unknown-unicast }
pps-min-rate min-rate-value pps-max-rate max-rate-value       // Configure the rate range for this type of packets, in pps.
```

Step 4 Return to the parent node view.

```
quit 2
```

Step 5 Configure a traffic suppression action for this type of packets.

```
storm-control-action
action suppress
```

Step 6 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

- Run the **display ifm/interfaces/interface[name=*"GE0/0/1"*]/storm-control/storm-rates** command to check the configuration of interface-based traffic suppression for all types of packets in the inbound direction of an interface.
- Run the **display ifm/interfaces/interface[name=*"GE0/0/1"*]/storm-control/storm-rates/storm-rate[packet-type={broadcast|unknown-multicast|unknown-unicast}]** command to check the configuration of interface-based traffic suppression for a certain type of packets in the inbound direction of an interface.

11.3.4.3 Example for Configuring Traffic Suppression in the Inbound Direction

Networking Requirements

On the network shown in **Figure 1**, DeviceA connects a Layer 2 network to a Layer 3 device. To prevent broadcast storms, traffic suppression needs to be configured on DeviceA to rate-limit incoming broadcast, unknown multicast, and unknown unicast packets forwarded at Layer 2.

Figure 11-12 Networking diagram of traffic suppression in the inbound direction

NOTE

In this example, interface 1 represents GE0/0/1.

Procedure

Step 1 Enter the edit-config view.

```
[user@HUAWEI]
MDCLI> edit-config
```

Step 2 (Optional) Enter the interface view.

```
[(gl)user@HUAWEI]
MDCLI> ifm interfaces interface name GE0/0/1
```

Step 3 Configure traffic suppression for broadcast traffic and set the maximum rate of broadcast packets to 100 pps. This maximum rate is the total rate of all interfaces on the device. (In this example, traffic suppression is based on the global total packet rate.)

```
[(gl)user@HUAWEI]
MDCLI> storm-control storm-rates storm-rate packet-type broadcast
[*(gl)user@HUAWEI]/storm-control/storm-rates/storm-rate[packet-type="broadcast"]
MDCLI> pps-min-rate 32 pps-max-rate 100
[*(gl)user@HUAWEI]/storm-control/storm-rates/storm-rate[packet-type="broadcast"]
MDCLI> commit
```

Step 4 Configure traffic suppression for unknown multicast traffic and set the maximum rate of unknown multicast packets to 80 pps.

```
[(gl)user@HUAWEI]
MDCLI> storm-control storm-rates storm-rate packet-type unknown-multicast
```



```
[*(gl)user@HUAWEI]/storm-control/storm-rates/storm-rate[packet-type="unknown-multicast"]
MDCLI> pps-min-rate 32 pps-max-rate 80
[*(gl)user@HUAWEI]/storm-control/storm-rates/storm-rate[packet-type="unknown-multicast"]
MDCLI> commit
```

Step 5 Configure traffic suppression for unknown unicast traffic and set the maximum rate of unknown unicast packets to 100 pps.

```
[(gl)user@HUAWEI]
MDCLI> storm-control storm-rates storm-rate packet-type unknown-unicast
[*(gl)user@HUAWEI]/storm-control/storm-rates/storm-rate[packet-type="unknown-unicast"]
MDCLI> pps-min-rate 32 pps-max-rate 100
[*(gl)user@HUAWEI]/storm-control/storm-rates/storm-rate[packet-type="unknown-unicast"]
MDCLI> commit
```

----End

Verifying the Configuration

Check the traffic suppression configuration in the inbound direction.

```
[(gl)user@HUAWEI]
MDCLI> display storm-control/storm-rates
{
  "storm-rate": [
    {
      "packet-type": "broadcast",
      "pps-min-rate": 32,
      "pps-max-rate": 100
    },
    {
      "packet-type": "unknown-multicast",
      "pps-min-rate": 32,
      "pps-max-rate": 80
    },
    {
      "packet-type": "unknown-unicast",
      "pps-min-rate": 32,
      "pps-max-rate": 100
    }
  ]
}
```

11.4 SSH Configuration

11.4.1 Overview of SSH

Definition

Secure Shell (SSH) is a cryptographic network protocol for transmitting network services (such as access and file transfer) securely over an unsecured network.

SSH versions are classified as SSH1.X and SSH2.0. SSH2.0 has an extended structure and features both security and function improvements over SSH1.X, including support for SFTP and more authentication and key exchange methods.

Purpose

Telnet lacks a secure authentication mode and uses TCP to transmit data in clear text, which brings great security risks. As a result, the system is vulnerable to

attacks such as denial of service (DoS), IP address spoofing, and route spoofing. Telnet alone is no longer viable with the increasing importance of network security. SSH addresses Telnet's shortcomings by providing secure login and other secure network services on an insecure network.

SSH provides a secure channel over TCP for data exchange using the well-known port 22 (this can be changed as required for security purposes).

11.4.2 Understanding SSH

This section uses SSH2.0 as an example to describe how SSH works.

Table 11-11 SSH working mechanism

Stage	Description
Connection setup	The SSH server listens to port 22 for SSH connections. After the client sends a connection request to the server, a TCP connection is set up between the client and server.
Version negotiation	The server and client determine which SSH version to use through version negotiation.
Algorithm negotiation	SSH supports multiple algorithms. Based on their supported algorithms, the server and client negotiate the following algorithms: key exchange algorithm for generating a session key, encryption algorithm for encrypting data, public key algorithm for digital signature and authentication, and hash-based message authentication code (HMAC) algorithm for data integrity protection.
Key exchange	The server and client dynamically generate a session key to protect data transmission and a session ID to identify the SSH connection through key exchange. The client also authenticates the server during this stage.
Client authentication	The client sends an authentication request to the server, and the server authenticates the client.
Session request	After the authentication succeeds, the client sends a session request to the server, requesting the server to provide a certain type of service (STelnet or SFTP). That is, the client requests to establish a session with the server.
Session interaction	After a session is established, the server and client exchange data.

11.4.3 Configuration Precautions for SSH

Licensing Requirements

SSH is not under license control.

Hardware Requirements

Table 11-12 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

11.4.4 Default Settings for SSH

11.4.4 Default Settings for SSH describes the default settings for SSH.

Table 11-13 Default settings for SSH

Parameter	Default Setting
STelnet server	Enabled
SSH server port number	22
Interval for updating the SSH server key pair	0 hours, indicating that the server key pair is never updated
SSH authentication timeout interval	60s
Maximum number of SSH authentication retries	3
Supported key exchange algorithms	curve25519-sha256,curve25519-sha256@libssh.org,diffie-hellman-group16-sha512,diffie-hellman-group18-sha512,ecdh-sha2-nistp256, ecdh-sha2-nistp384,ecdh-sha2-nistp521
Supported encryption algorithm	aes256-ctr
Supported HMAC algorithms	hmac-sha2-512, hmac-sha2-512-etm@openssh.com, hmac-sha2-256, hmac-sha2-256-etm@openssh.com
Supported public key algorithms	ssh-ed25519, ssh-ed25519-cert-v01@openssh.com, rsa-sha2-512, rsa-sha2-256, x509v3-rsa2048-sha256

11.4.5 Configuring an SSH Server

11.4.5.1 Configuring the SSH Server Function and Related Parameters

Context

This section describes how to configure the SSH server function and parameters, including generating the local key pair of the server, enabling the SSH server function, and setting server parameters, such as the port number, listening interface, and PKI realm.

For details about configuration parameters, see **huawei-sshs.yang**.

NOTE

To ensure that the SSH algorithm negotiation is successful, the SSH client must support the key exchange algorithm, encryption algorithm, public key algorithm, and HMAC algorithm configured on the SSH server.

The SSH server does not support SSH1.X.

After the SSH service is disabled, the established connections are not disconnected.

Procedure

Step 1 Enter the edit-config mode.

```
edit-config
```

Step 2 Enable the SSH server function.

1. Enter the server enabling view.
   ```
   sshs server-enable
   ```
2. Enable the SSH server function.
   ```
   stelnet-ipv4-enable enable
   ```
   By default, the STelnet service is enabled.

Step 3 Configure the port number of the SSH server.

1. Enter the port view.
   ```
   sshs server-port
   ```
2. Set the server port.
   ```
   ipv4-port-number number
   ```
   By default, the port number of an SSH server is 22.

Step 4 Set the source IP address of the SSH server.

```
sshs ipv4-server-sources ipv4-server-source src-interface src-interface
```

By default, no source interface is configured for an SSH server.

Step 5 Configure a PKI realm.

```
sshs server
pki-domain pki-domain
```

When the SSH server performs PKI certificate authentication with an SSH client, the identity certificate of the server is obtained from the PKI realm.

Step 6 Commit the configuration.


```
commit
```

----End

11.4.5.2 Configuring an SSH User

Context

Configuring an SSH user includes creating an SSH user and configuring a PKI realm for the SSH user.

Procedure

Step 1 Enter the editing view.

```
edit-config
```

Step 2 Create an SSH user.

```
sshs users user name user-name
```

Step 3 Configure a PKI realm for the SSH user.

```
pub-key-type PKI key-name key-name
```

pub-key-type specifies the public key type. Currently, only PKI is supported. **key-name** specifies the name of the PKI realm bound to the user.

Step 4 Commit the configuration.

```
commit
```

----End

11.5 ASPF/ALG Configuration

11.5.1 Overview of ASPF/ALG

ASPF Definition

Application specific packet filter (ASPF) is a packet filtering technology applied to the application layer, which is also called state-based packet filtering.

ASPF Purpose

ASPF enables the device to automatically detect the application layer information of certain packets and create access rules (generate a server mapping table) based on the application layer information. The server mapping table is used to permit subsequent packets in the data channel, which is equivalent to creating a refined security policy.

For example, multi-channel protocols (such as FTP and SIP) must negotiate the address and port of the subsequent data channel in the control channel, and then establish the data channel connection according to the negotiation result. The IP address and port of the data channel are dynamically negotiated, and cannot be predicted by the administrator. To ensure the proper establishment of the data

channel, all ports need to be opened, which raises the risk of attacks on the server or client. In this case, you can enable ASPF to prevent this risk.

ALG Definition

Application Level Gateway (ALG) is a NAT traversal technology.

ALG Purpose

In NAT scenarios, the Application Level Gateway (ALG) function can automatically detect application-layer information of certain packets, generate corresponding access rules based on the application-layer information (generate a server mapping table), and automatically translate the IP address and port information in packet payloads. NAT translates only the IP address and port in the packet header, but not application-layer data. In many application-layer protocols, the packet payload also contains the address or port information. If the data is not translated, the subsequent communication may be abnormal.

11.5.2 Understanding ASPF/ALG

The ASPF/ALG function is implemented through the server mapping table. This section describes only the server mapping table created by the ASPF/ALG function. For the server mapping tables of other types, see **"Server Mapping Entries Configuration"** in *CLI Configuration Guide > System Forwarding Configuration*.

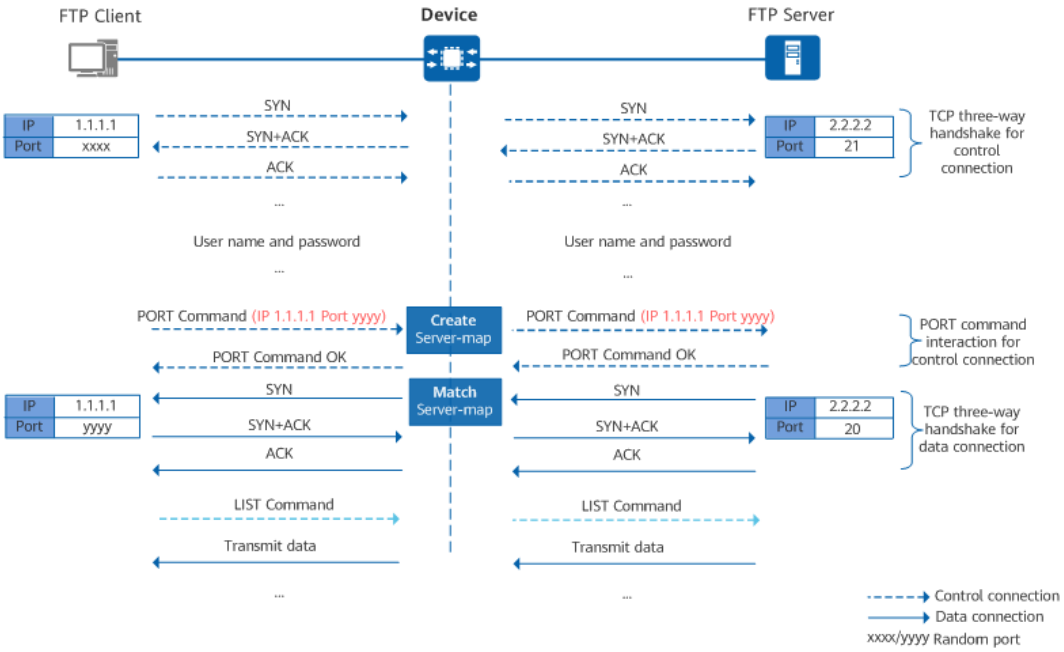
The server mapping table is one of the bases for implementing the ASPF/ALG function.

The ASPF/ALG function can detect the application layer information of certain packets and record the key data in the application layer information in the server mapping table. Subsequent packets that match server mapping entries are directly permitted or NAT is performed, and sessions are established.

For example, ASPF for multi-channel protocols (such as FTP and SIP) often require the establishment of two channels: control channel and data channel. After the control channel is established, the address and port of the data channel is negotiated through the control channel. The data channel is established according to the negotiation result. The device automatically generates a server mapping entry based on the address and port information carried in the application layer of the negotiation packet. Subsequent data packets that match server mapping entries are permitted. The data channel is then established successfully.

For example, in FTP active mode (the server proactively accesses the client) the device detects the application layer information of the **PORT** command packet and records the IP address and port carried in the application layer information in the server mapping table.

Figure 11-13 ASPF in FTP active mode



Port yyyy is the data port opened by the client to the server through the control channel. The data packet for the server (2.2.2.2) to proactively access port yyyy of the client (1.1.1.1) matches the server mapping entry and is permitted.

NOTE

A server mapping entry of the ASPF/ALG type is generated only when the corresponding traffic passes through the device.

11.5.3 Configuration Precautions for ASPF/ALG

Licensing Requirements

ASPF/ALG is not under license control.

Hardware Requirements

Table 11-14 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

11.5.4 Default Settings for ASPF/ALG

Table 11-15 Default settings for ASPF/ALG

Parameter	Default Setting
Global ASPF/ALG	Disabled

11.5.5 Configuring ASPF/ALG for Well-Known Protocols

11.5.5.1 List of Well-Known Protocols Supported by ASPF/ALG

The Device can perform ASPF/ALG for the following known protocols. For scenario-based support details, see **Table 11-16**. The following sections use FTP, DNS, PPTP, and SIP as examples to describe the ASPF/ALG process for each protocol.

Table 11-16 List of well-known protocols supported by ASPF/ALG

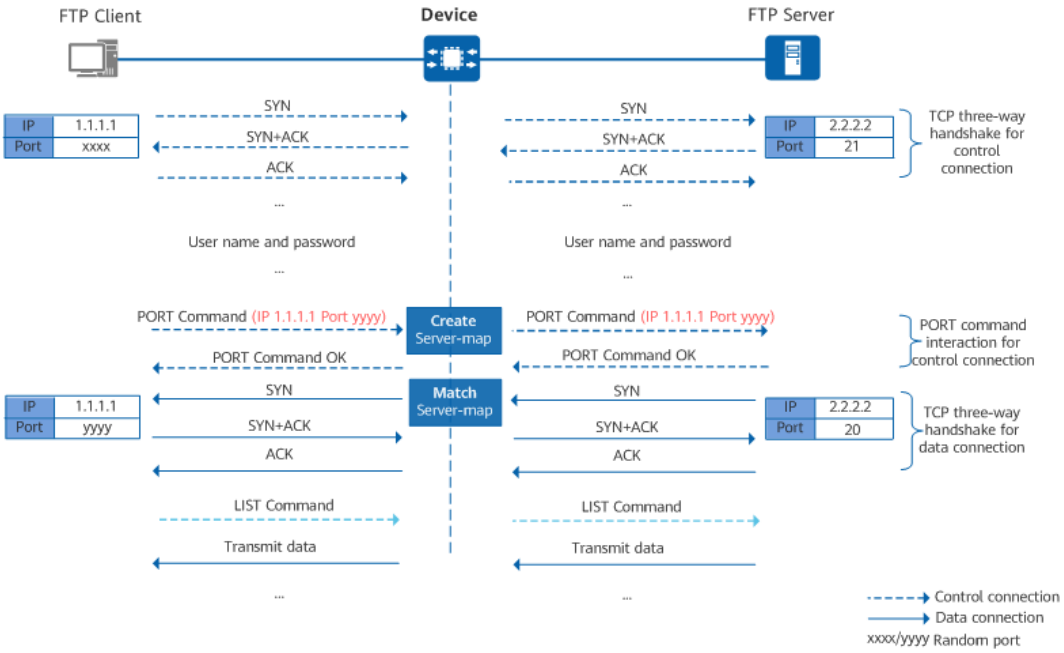
Protocol	ASPF (IPv4)	ALG (IPv4–IPv4)
DNS	Yes	Yes
FTP	Yes	Yes
PPTP	Yes	Yes
RTSP	Yes	Yes
SIP	Yes	Yes

11.5.5.2 Understanding FTP ASPF/ALG

ASPF in FTP active mode

In FTP active mode, the client uses the random port (xxxx) to send a connection request to port 21 on the server to set up a control channel. The client then uses the **PORT** command to negotiate the port number for the data channel. The negotiated port number is yyyy. After that, the server initiates a data channel request to port yyyy of the client. The server sends data to the client after the data channel is established.

Figure 11-14 ASPF in FTP active mode

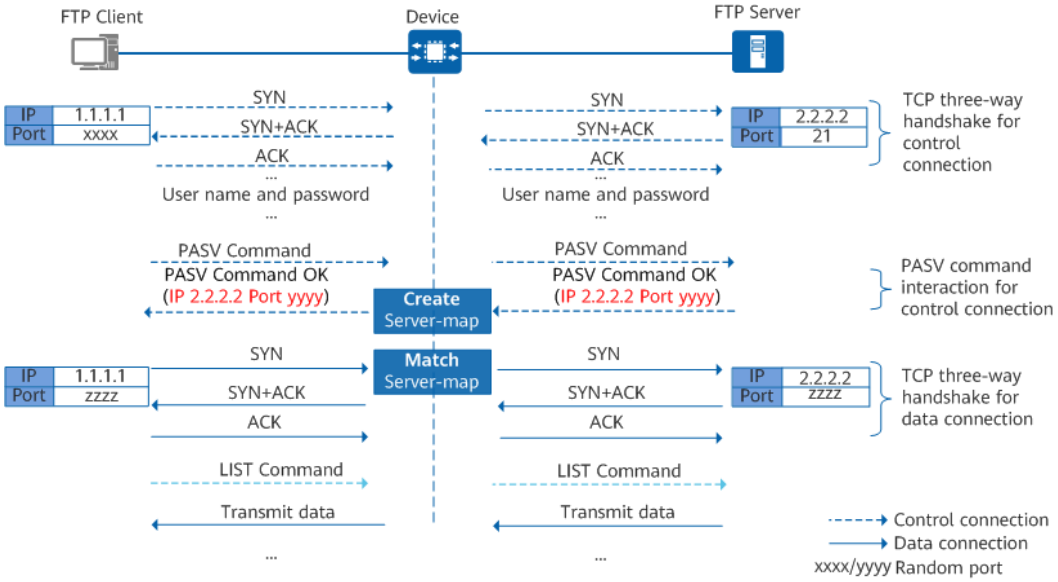


The application-layer information of the **PORT** command carries the IP address and randomly-opened port of the client. By analyzing the application-layer information of the **PORT** command, the **Device** can predict the behavior of subsequent packets and create a server mapping table based on the IP address and port in the application-layer information. After receiving a data connection packet from the server to the client, the device forwards the packet if the packet matches a server mapping entry.

ASPF in FTP passive mode

In FTP passive mode, the client uses the random port (xxxx) to send a connection request to port 21 on the server to set up a control channel. The client then uses the **PASV** command to negotiate the port number for the data channel. The negotiated port number is yyyy. After that, the client initiates a data channel request to port yyyy of the server. The server sends data to the client after the data channel is established.

Figure 11-15 ASPF in FTP passive mode

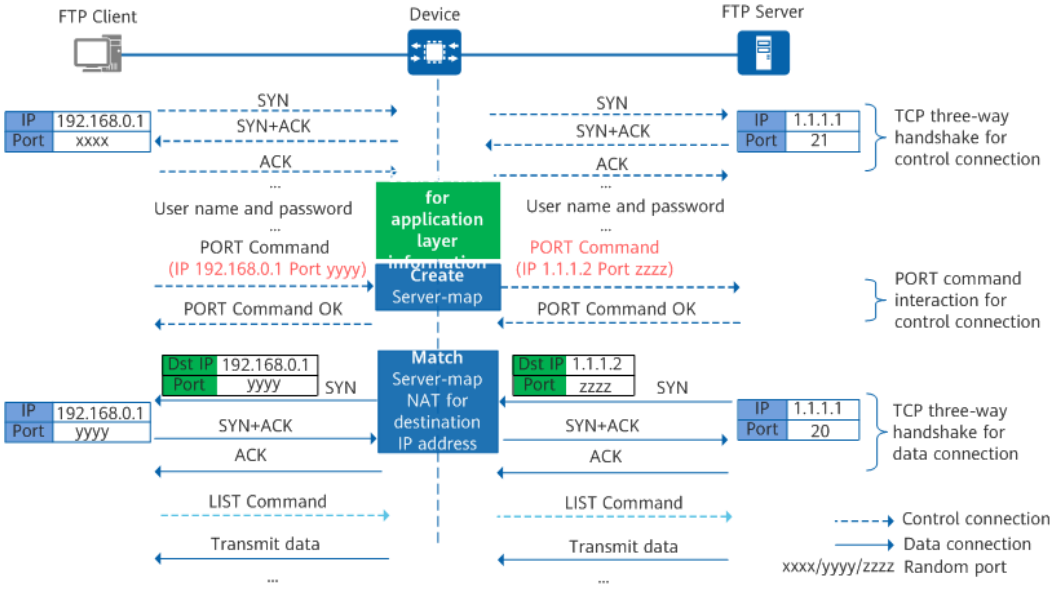


The application-layer information of the **PASV Command OK** command carries the IP address and randomly-opened port of the server. By analyzing the application-layer information of the **PASV Command OK** command, the device can predict the behavior of subsequent packets and create a server mapping table based on the IP address and port in the application-layer information. After receiving a data connection packet from the client to the server, the device forwards the packet if the packet matches a server mapping entry.

ALG in FTP active mode

As shown in **Figure 11-16**, a client resides on the intranet, and a server on the Internet. To enable the client to properly access the server, configure a source NAT policy on the Device to translate the private address of the client into a public address and allow port translation. After the client and server establish a control channel through TCP three-way handshake, the client sends a private address and an open private network port to the server through the **PORT** command to establish a data channel.

Figure 11-16 ALG in FTP active mode



Before ALG is configured, the device translates only the IP address and port information in the packet header when performing source NAT. However, the IP address and port information carried in the packet payload remain unchanged. If the IP addresses and port numbers of both sides are different, FTP cannot work properly.

After ALG is configured, the **Device** analyzes the application-layer information of the **PORT** command, translates the private address and private network port carried in the command into the public address and public network port, forwards them to the server, and creates a server mapping table. The server initiates a connection to the public IP address and public port. After the packet reaches the device, it matches a server mapping entry. The device then automatically translates the destination IP address and port to the actual private address.

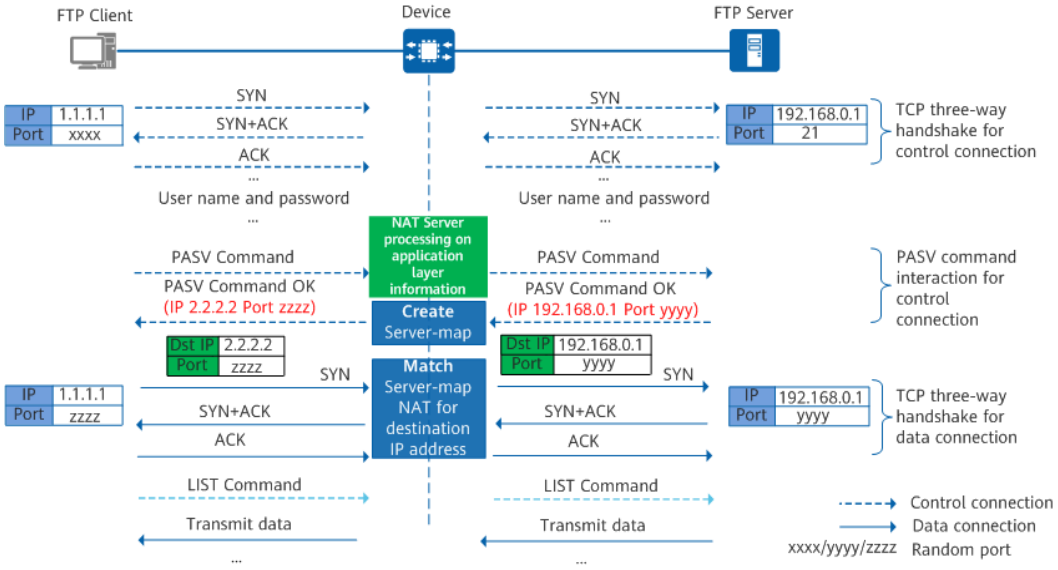
NOTE

Common NAT translates only transport-layer information of packets, and ALG can also translate application-layer information of packets.

ALG in FTP passive mode

As shown in **Figure 11-17**, a client resides on the Internet, and a server on the intranet. To enable the client to properly access the server, configure NAT Server on the Device to translate the public address of the server into a private address and allow port translation. After the client and server establish a control channel through TCP three-way handshake, the server sends a private address and an open private network port to the client through the **PASV Command OK** command to establish a data channel.

Figure 11-17 ALG in FTP passive mode



Before ALG is configured, the device translates only the IP address and port information in the packet header when performing destination NAT. However, the IP address and port information carried in the packet payload remain unchanged. If the IP addresses and port numbers of both sides are different, FTP cannot work properly.

After ALG is configured, the **Device** analyzes the application-layer information of the **PASV Command OK** command, translates the private address and private network port carried in the command into the public address and public network port, forwards them to the client, and creates a server mapping table. The client initiates a connection to the public IP address and public port. After the packet reaches the device, it matches a server mapping entry and automatically translates the destination IP address and port to the actual private address.

NOTE

Common NAT translates only transport-layer information of packets, and ALG can also translate application-layer information of packets.

11.5.5.3 Understanding DNS ALG

Understanding DNS

The Domain Name System (DNS) protocol is an application-layer protocol that is used to implement mapping between a network IP address and a host domain name.

The DNS adopts the client/server (C/S) structure. The DNS query process in UDP mode is used as an example to describe the interaction between the DNS client and the DNS server.

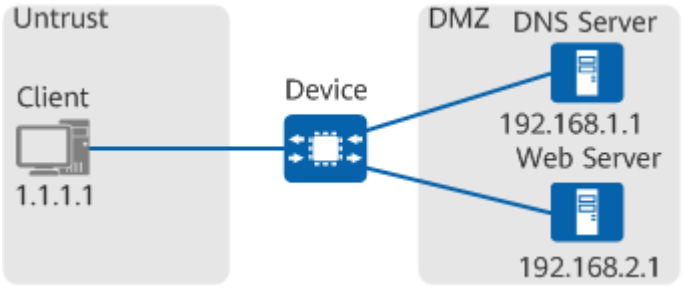
Figure 11-18 DNS packet exchange

1. The client requests the IP address corresponding to the domain name **www.huawei.com** from the DNS server.
2. The DNS server adds the IP address corresponding to the domain name to the application layer information of the DNS response packet and sends the packet to the client.
3. The client initiates an access request to the received IP address.

Understanding DNS ALG

As shown in **Figure 11-19**, an enterprise has deployed a device as a security gateway at the intranet border. To enable the DNS server and web server on the private network to provide services for external users, the NAT Server needs to be configured on the device to map the public IP addresses of the servers to the corresponding private IP addresses. In addition to the IP addresses of Internet interfaces, the enterprise applies for two public IP address (1.1.1.10, 1.1.1.11) from the ISP for the intranet DNS server and web server to provide services. The mapping between the domain name and private IP address of the web server exists on the DNS server.

Figure 11-19 DNS ALG



When DNS ALG is disabled, the following process ensures:

1. The client sends a domain name resolution request to the DNS server, requesting the IP address corresponding to the domain name of the web server.
2. The DNS server returns the private IP address of the web server to the client.
3. The client directly initiates an access request to the private IP address of the web server, but the access fails.

The ALG function can solve the access failure issue in NAT scenarios. After the ALG function is enabled, the packet exchange process is as follows:

1. The client sends a domain name resolution request to the DNS server, requesting the IP address corresponding to the domain name of the web server.
2. The DNS server returns the private IP address of the web server to the client.
3. The device uses the ALG function to translate the private IP address carried in the application layer of the DNS response packets into the corresponding public IP address.
4. The client initiates an access request to the public IP address of the web server. The public IP address is translated into a private IP address through the NAT Server. The access succeeds.

DNS ALG is used to translate the address in the application layer of DNS response packets. In non-NAT scenarios, the address does not need to be translated. Therefore, DNS ALG does not need to be enabled in non-NAT scenarios.

The ALG function for the DNS protocol does not involve monitoring dynamic ports or permitting special traffic. Therefore, the ALG function for the DNS protocol does not depend on the server mapping table.

11.5.5.4 Understanding PPTP ALG

The Point-to-Point Tunneling Protocol (PPTP) is a new enhanced security protocol developed based on the PPP protocol. It is a method for implementing Virtual Private Networks (VPNs).

PPTP is a multi-channel protocol. It uses TCP to create control channels to send control commands, and uses GRE to encapsulate data packets that will be transmitted in data channels.

PPTP adopts the client/server mode. The PPTP client establishes a TCP connection with the PPTP server, and then establishes a PPTP control-layer connection on the TCP connection. Then, the PPP data packets are encapsulated through GRE and then transmitted in the tunnel. **Figure 11-20** shows the key interaction process between the client and the server in the PPTP ALG scenario.

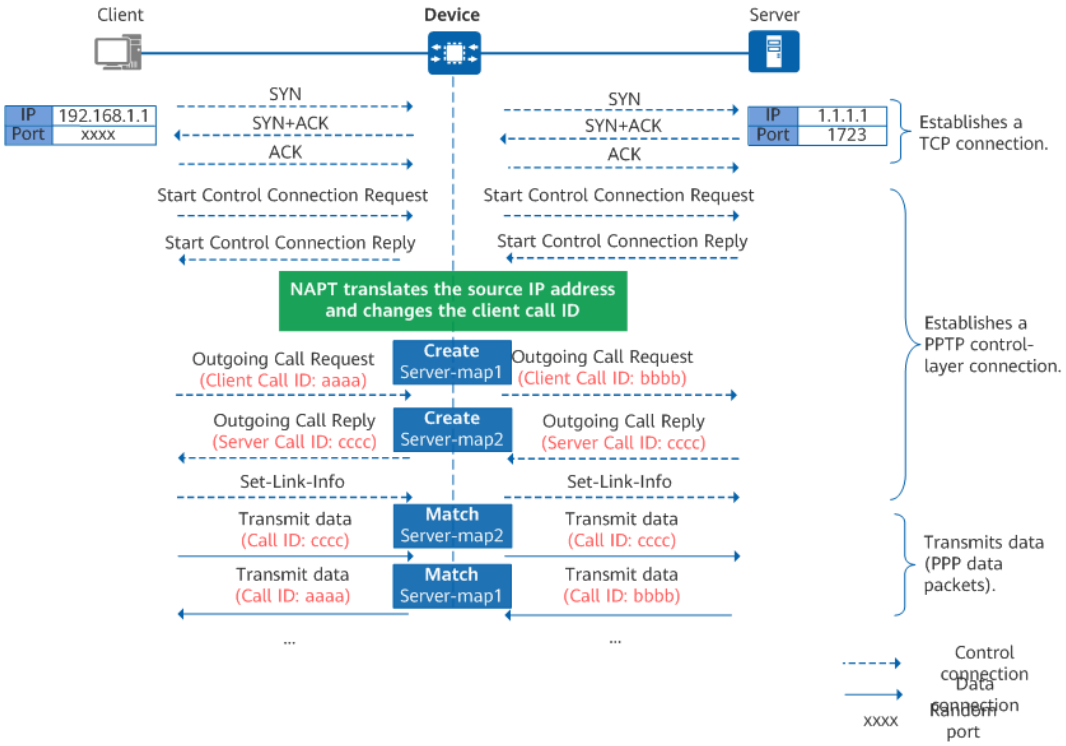
In the NAT scenario where the source IP address and source port are both translated, the communication between the client and the server may be abnormal. PPTP uses GRE to encapsulate data packets, but the GRE header does not contain port information. Common NAT can only identify and translate the source IP address of the client. When multiple clients on the intranet interact with the same server, the Device receiving a response packet from the server cannot determine to which client the response packet is sent based on only the destination public IP address. As a result, the Device discards the packet.

In addition, PPTP differentiates tunnels based on the source IP address and call ID in the GRE header. Because clients do not negotiate with each other, they may carry the same call ID. After the source IP address is translated using NAT, different clients correspond to the same public network address and call ID. As a result, tunnels fail to be established.

Therefore, in the NAT scenario where the source IP address and source port are both translated, you need to enable PPTP ALG on the **Device** to translate the

source IP address and convert the call ID, and create an invisible channel (server mapping table) for subsequent data transmission. If the invisible channel does not exist, data packets will be blocked.

Figure 11-20 PPTP ALG



1. The client and the server establish a TCP connection through three-way handshake. A control connection session is generated on the Device. The IP address of the client is translated from 192.168.1.1 to 1.1.1.2 through source NAT, and the port is translated from **xxxx** to **yyyy**.
2. The client and the server continue to establish a PPTP control-layer connection based on the TCP connection. The packets matching the control connection session are permitted.
   a. The client carries the local call ID in the Outgoing Call Request packet. The source IP address and call ID of the packet are changed on the **Device** and then sent to the server. **Server-map1** is also created on the **Device**.
   b. The server carries the local call ID in the Outgoing Call Reply packet. The destination IP address of the packet is translated on the **Device** and then sent to the client. **Server-map2** is also created on the **Device**. The call ID of the server is equivalent to the destination port. In the source NAT scenario, the destination port does not need to be translated. Therefore, the **Device** does not change the call ID of the server.
3. The client and the server transmit data through tunnels. Two tunnels are generated for each pair of client and server to exchange data packets. One tunnel is used for communication from the client to the server, and the other is used for communication from the server to the client.

   When a packet from the client to the server reaches the device, **Server-map2** is matched and the device translates the source address to the public address 1.1.1.2, permits the packet, and creates a session.

When a packet from the server to the client reaches the device, **Server-map1** is matched and the device translates the destination address to the client's actual IP address 192.168.1.1 and changes the call ID to the actual call ID. Then, the device permits the packet and creates a session.

11.5.5.5 Understanding SIP ASPF/ALG

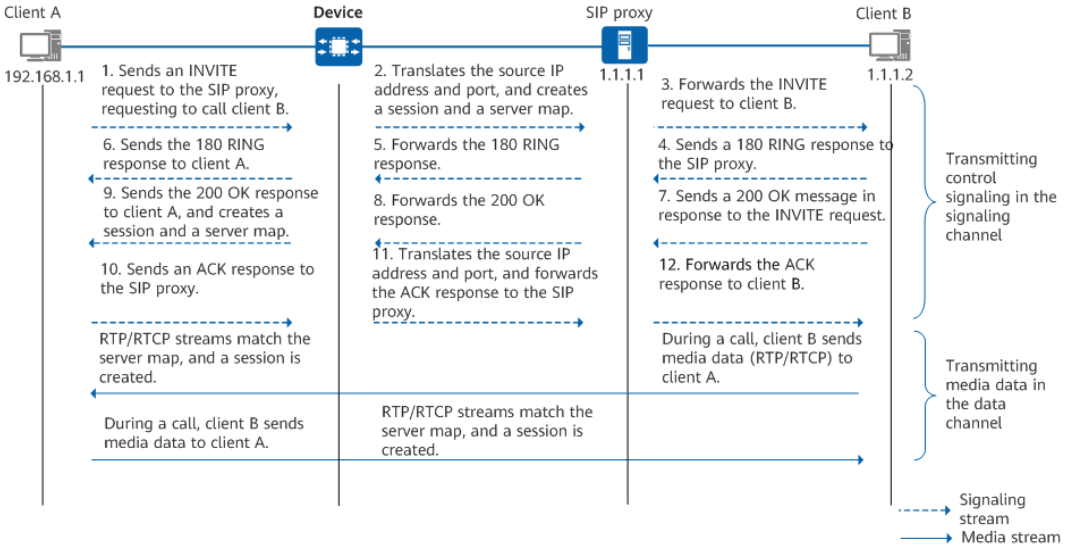
The Session Initiation Protocol (SIP) is an IETF standard protocol used to create, modify, and release sessions of one or more participants. These sessions can be interactive user sessions that involve multimedia elements such as voice calls, multimedia conferences, and virtual reality.

SIP is a multi-channel protocol. In addition to establishing signaling channels for transmitting signaling, the calling and called parties also establish data channels for transmitting media data such as voice and video data. Therefore, SIP traffic is classified into signaling streams and media streams. Signaling streams are transmitted through UDP or TCP, including requests and responses from both parties. Media streams such as voice and video data packets are transmitted through RTP and RTCP.

In the NAT scenario, NAT can only translate the network-layer IP address and transport-layer port carried in the packets and cannot translate the IP address and port carried in the application layer information. As a result, subsequent signaling and media streams cannot be exchanged. In this case, the SIP ASPF/ALG function needs to be enabled on the device for detecting and translating the IP address and port information carried in the application layer information of packets, and record the information in the server mapping entries. The media data packets match the server mapping entries when passing through the device and are permitted.

As shown in **Figure 11-21**, client A is located on the intranet, and client B and the SIP proxy are located on the Internet in the source NAT scenario. This section describes the key interaction process among client A, client B, and the SIP proxy and the processing of packets after the ASPF/ALG function is enabled on the **Device**.

Figure 11-21 SIP ASPF/ALG



1. Client A sends an INVITE request to port 5060 of the SIP proxy to call client B. The Via field in the header of the INVITE request contains the address information of the sender (for example, 192.168.1.1:2000). The message body contains the media control information (IP address and port indicated by the Connection Information and Media Description fields) described by the SDP, instructing the peer end to send media streams to the specified IP address and port number (for example, 192.168.1.1:3000).
2. After receiving the INVITE request, the **Device** translates the IP address and port number, forwards the INVITE request to the SIP proxy, creates a signaling channel session, and creates a server mapping table based on the IP address and port number in the header and body of the INVITE message. The server mapping table is used to permit the subsequent interaction messages.

   NOTE

   The **Device** creates two server mapping entries based on the media connection address in the message body. The two server mapping entries are used to permit RTP streams and RTCP streams respectively. The port number used by RTCP streams is equal to the port number for RTP streams plus 1.

   The **Device** creates a server mapping entry (the first server mapping entry) for signaling streams to permit subsequent signaling data sent from client B to client A. It also creates two server mapping entries (the second and third server mapping entries) for data streams to permit subsequent media data sent from client B to client A.
3. The SIP proxy forwards the INVITE request to client B, requesting client B to join the call. The INVITE message also carries the session description of client A.
4. Client B rings and sends a 180 RING response to the SIP proxy.
5. The SIP proxy forwards the 180 RING response.
6. After receiving the 180 RING response, the Device matches the signaling channel session, translates the destination address of the message to the actual IP address and port of client A, and forwards the message to client A. Then, client A hears the ringback tone.
7. Client B answers the call and then sends a 200 OK response to the SIP proxy, indicating that the INVITE request sent by the SIP proxy has been accepted and processed. The Via field in the header of the message contains the address information of the sender (for example, 1.1.1.2:3000). The message body contains the media control information (IP address and port indicated by the Connection Information and Media Description fields) described by the SDP, instructing the peer end to send media streams to the specified IP address and port number (for example, 1.1.1.2:4000).
8. The SIP proxy forwards the 200 OK response, indicating that the INVITE request has been accepted and processed. The 200 OK response also carries the session description of client B.
9. After receiving the 200 OK response, the Device translates the destination address of the message to the actual IP address and port of client A and forwards the message to client A. It also creates a server mapping table based on the IP address and port number in the header and body of the 200 OK response. The server mapping table is used to permit the packets exchanged subsequently.

NOTE

The **Device** creates two server mapping entries based on the media connection address in the message body. The two server mapping entries are used to permit RTP streams and RTCP streams respectively. The port number used by RTCP streams is equal to the port number for RTP streams plus 1.

The **Device** creates a server mapping entry (the first server mapping entry) for signaling streams to permit subsequent signaling data sent from client A to client B. It also creates two server mapping entries (the second and third server mapping entries) for data streams to permit subsequent media data sent from client A to client B.

10. Client A sends an ACK message to the SIP proxy, indicating that it has received the SIP proxy's final response to the INVITE request.
11. The Device receives the ACK message, translates the IP address and port number, and forwards the message to the SIP proxy.
12. The SIP proxy forwards the ACK message to client B, indicating that it has received client B's final response to the INVITE request. In this case, both the calling party and the called party know each other's media connection address, and the call can be set up.
13. During a call between client A and client B, media streams match the server mapping entries. The **Device** creates two sessions (RTP session and RTCP session) from client A to client B and from client B to client A respectively.

11.5.5.6 Configuring ASPF/ALG for Well-Known Protocols

Context

The device supports the global ASPF/ALG function.

For details about configuration parameters, see **huawei-aspf.yang**.

Procedure

Step 1 Enter the system view.

```
edit-config
```

Step 2 Configure the ASPF/ALG function.

```
aspf
protocol { dns | ftp | pptp | sip | rtsp }
```

NOTE

- If ASPF/ALG processing is required for traffic of multiple protocols, run this command for each involved protocol.
- The ASPF/ALG function for SIP configured using the command takes effect only on UDP-based SIP traffic.
- Enable the ASPF/ALG function for a specific protocol as required. For protocols that do not require the ASPF/ALG function, disable the function in a timely manner.

Step 3 Commit the configuration.

```
commit
```

----End

11.5.5.7 Example for Configuring SIP ALG

Networking Requirements

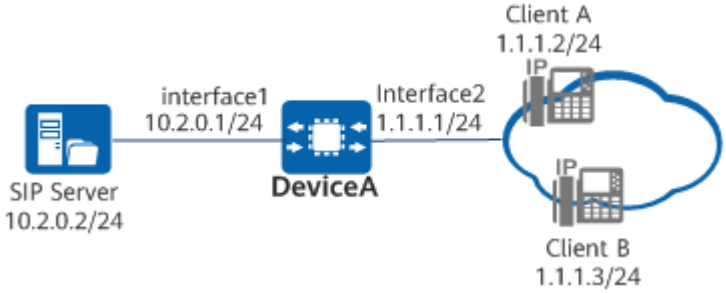
As shown in **Figure 11-22**, a SIP server is deployed on the enterprise intranet. When going online, each SIP client needs to send a register message to the SIP servers. The register messages are carried through SIP.

DeviceA is deployed between SIP clients and a SIP server to perform NAT ALG on transmitted SIP messages.

Figure 11-22 Configuring ALG for SIP messages

NOTE

In this example, interface 1 and interface 2 represent GE0/0/9 and GE0/0/10, respectively.



Item	Data	Description
GE0/0/9	IP address: 10.2.0.1	This interface is connected to the server and resides on the same network segment as the server.
GE0/0/10	IP address: 1.1.1.1	This interface is connected to the clients and resides on the same network segment as the clients.
SIP server	10.2.0.2/24	The server is deployed on the intranet.

Configuration Roadmap

1. Configure IP addresses for the interfaces and set basic network parameters.
2. Configure destination NAT so that the intranet SIP server can provide services externally (public address: 1.1.1.10).
3. Configure ALG to properly forward SIP messages.

Procedure

Step 1 Configure IP addresses for the interfaces.

```
MDCLI> edit-config
MDCLI> ifm interfaces interface name GE0/0/9
MDCLI> ipv4 addresses address ip 10.2.0.1
MDCLI> type main mask 255.255.255.0
MDCLI> quit 6
MDCLI> ifm interfaces interface name GE0/0/10
MDCLI> ipv4 addresses address ip 1.1.1.1
MDCLI> type main mask 255.255.255.0
MDCLI> quit 6
MDCLI> commit
```

Step 2 Configure destination NAT so that the intranet SIP server can provide services externally.

```
MDCLI> nat-server server-mappings server-mapping name tcp_sip
MDCLI> protocol tcp
MDCLI> global-port
MDCLI> start-port 5060
MDCLI> quit
MDCLI> inside-port
MDCLI> start-port 5060
MDCLI> quit
MDCLI> global
MDCLI> start-ip 1.1.1.10
MDCLI> quit
MDCLI> inside
MDCLI> start-ip 10.2.0.2
MDCLI> quit 4
MDCLI> nat-server server-mappings server-mapping name udp_sip
MDCLI> protocol udp
MDCLI> global-port
MDCLI> start-port 5060
MDCLI> quit
MDCLI> inside-port
MDCLI> start-port 5060
MDCLI> quit
MDCLI> global
MDCLI> start-ip 1.1.1.10
MDCLI> quit
MDCLI> inside
MDCLI> start-ip 10.2.0.2
MDCLI> quit 4
MDCLI> commit
```

Step 3 Configure ALG to properly forward SIP messages.

```
MDCLI> aspf
MDCLI> protocol sip
MDCLI> commit
```

----End

Verification

Client A and Client B register successfully on the server.

11.6 Service and Management Isolation Configuration

Context

The purpose of service and management isolation is to minimize the attack surface and ensure network security. The device supports such isolation.

Specifically, it complies with the three-layer and three-plane security isolation mechanism of X.805. The three planes refer to:

- Management plane (or O&M plane): carries O&M data flows of the device.
- Control plane (or signaling plane): carries protocol interaction data flows of the device.
- Service plane (or forwarding/user plane): carries information forwarding data flows of the device.

After the three planes are isolated, if one plane is attacked, the operations and security of other planes are not affected. For example, if the service plane is subject to a DoS attack, the management plane will be unaffected. The administrator can then log in to the management plane to eliminate the DoS attacks. If, on the other hand, the planes are not isolated, such an attack would cause the processing tasks on the data plane to further occupy resources such as CPU and memory resources until the resources are fully exhausted. In such cases, the administrator would be unable to manage the device.

Isolating the service plane from the management plane is to isolate service interface traffic from management interface traffic. Currently, such isolation is implemented logically as follows:

- Apply ACL policies to interfaces so that the service network cannot communicate with the management network. In this way, Layer 3 packets on the service network are logically isolated from those on the management network.
- Assign the service interface and management interface to different VLANs. As such, Layer 2 packets cannot be transmitted between the service interface and management interface, achieving logical isolation.

For details about the parameters, see **huawei-acl.yang** and **huawei-sacl.yang**.

Procedure

- Configure ACL policies and the **traffic-filter** command to isolate the service plane from the management plane.
  a. Configure an uplink ACL rule group for the management plane to allow only the packets destined for the management plane to pass through.

```
edit-config
acl groups group identity ACL_NAME1        // Create an ACL group.
type advance
rule-advances rule-advance name rule1    // Create an advanced ACL rule.
protocol 0 action permit dest-ipaddr M_IP dest-wild M_IP_Wild  // Configure an ACL rule.
commit
quit
rule-advance name rule2     // Create an advanced ACL rule.
protocol 0 action deny      // Configure an ACL rule.
commit
```

  b. Configure a downlink ACL rule group for the management plane to allow only packets originating from the management plane to pass through.

```
edit-config
acl groups group identity ACL_NAME2                // Create an ACL group.
type advance
rule-advances rule-advance name rule1            // Create an advanced ACL rule.
protocol 0 action permit source-ipaddr M_IP source-wild M_IP_Wild     // Configure an ACL
rule.
commit
quit
```



```
rule-advance name rule2                 // Create an advanced ACL rule.
type advance protocol 0 action deny     // Configure an ACL rule.
commit
```

c. Bind an ACL rule to the inbound direction of an interface.

```
edit-config
ifm interfaces interface name GE_Name   // Enter the interface view.
traffic-filter-applys traffic-filter-apply direction inbound   // Create a simplified traffic policy for packet filtering in the inbound direction of the interface.
acl-instances acl-instance acl ACL_NAME1 // Configure an ACL applied to the simplified traffic policy for packet filtering.
enable-statistic true                   // Enable statistics collection of simplified traffic policy for packet filtering.
commit
```

d. Bind an ACL rule to the outbound direction of an interface.

```
edit-config
ifm interfaces interface name GE_Name   // Enter the interface view.
traffic-filter-applys traffic-filter-apply direction outbound  // Create a simplified traffic policy for packet filtering in the outbound direction of the interface.
acl-instances acl-instance acl ACL_NAME2      // Configure an ACL applied to the simplified traffic policy for packet filtering.
enable-statistic true         // Enable statistics collection of simplified traffic policy for packet filtering.
commit
```

- Assign the service interface and management interface to different VLANs. As such, Layer 2 packets cannot be transmitted between the service interface and management interface, achieving logical isolation.

  a. Create and configure a VLAN instance for the management interface.

  ```
  edit-config
  vlan vlans vlan id VLAN_ID          // Configure a VLAN ID for the bridge.
  commit
  ```

  b. Assign VLANs by interface.

  ```
  edit-config
  ifm interfaces interface name interface_name    // Enter the interface view.
  ethernet main-interface l2-attribute            // Enter the Ethernet view.
  pvid VLAN_ID
  commit
  ```

----End

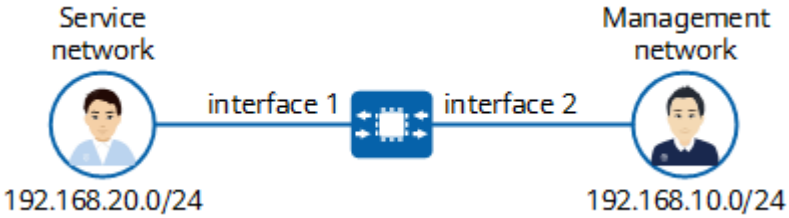
Example

In **Figure 11-23**, the service network on the 192.168.20.0/24 network segment is connected to interface 1 (service interface) of the device; the management network on the 192.168.10.0/24 network segment is connected to interface 2 (management interface) of the device. ACL-based policy control is configured on the device to isolate the service plane and management plane. This is to prevent 192.168.20.0/24 from communicating with 192.168.10.0/24, ultimately protecting the device against attacks caused by address leakage on the management interface.

Figure 11-23 Networking diagram of service and management isolation

NOTE

In this example, interface 1 and interface 2 stand for GE0/0/1 and GE0/0/0 respectively.



The configuration of the device in this example is as follows:

1. Create VLAN 10.
   ```
   edit-config
   vlans vlan id 10
   commit
   ```
2. Create VLAN 20.
   ```
   edit-config
   vlans vlan id 20
   commit
   ```
3. Bind GE0/0/0 to VLAN 10.
   ```
   edit-config
   ifm interfaces interface name GE0/0/0
   ethernet main-interface l2-attribute
   pvid 10
   commit
   ```
4. Configure an ACL policy in which destination address is the management plane address.
   ```
   edit-config
   acl groups group identity manager_dest
   rule-advances rule-advance name rule1
   type advance protocol 0 action permit dest-ipaddr 192.168.10.0 dest-wild 0.0.0.255
   commit
   quit
   rule-advance name rule2
   protocol 0 action deny
   commit
   ```
5. Configure an ACL policy in which the source IP address is the IP address of the management plane.
   ```
   edit-config
   acl groups group identity manager_source
   rule-advances rule-advance name rule1
   protocol 0 action permit source-ipaddr 192.168.10.0 source-wild 0.0.0.255
   commit
   quit
   rule-advance name rule2
   type advance protocol 0 action deny
   commit
   ```
6. Apply the ACL policy in which destination address is the management plane address to the inbound direction of GE0/0/0.
   ```
   edit-config
   ifm interfaces interface name GE0/0/0
   traffic-filter-applys traffic-filter-apply direction inbound
   acl-instances acl-instance acl manager_dest
   enable-statistic true
   commit
   ```
7. Apply the ACL policy in which source address is the management plane address to the outbound direction of GE0/0/0.
   ```
   edit-config
   ifm interfaces interface name GE0/0/0
   traffic-filter-applys traffic-filter-apply direction outbound
   ```



```
acl-instances acl-instance acl manager_source
enable-statistic true
commit
```

Verifying the Configuration

- Run the **display acl/groups/** command to check the ACL configuration.
- Run the **ifm interfaces interface name** *GE0/0/0* command to enter the interface view, and then run the **display traffic-filter-applys/** command to check the applied ACL traffic filtering policy.

11.7 Weak Password Dictionary Maintenance Configuration

Context

A weak password is a simple password that can be easily guessed or cracked within a short period of time. A device provides the weak password dictionary maintenance function to prevent security problems caused by simple passwords. You can configure a dictionary of prohibited passwords in advance and load it to the device. When a new user is added or a user password needs to be changed, the system will prevent the passwords in this dictionary from being used.

The weak password dictionary is stored as a text file and supports only the .txt format. Each line contains a password. The following is an example of the weak password dictionary **pwd_dict.txt**.

```
Abcd@123
Huawei@123
Aaabb@321
Raatr@321
```

After a weak password dictionary is loaded, password settings in CLI login, configuration file management, and AAA configuration are affected. For details, see the Configuration Notes and Command Reference. For details about configuration parameters, see huawei-system.yang.

Licensing Requirements

The weak password dictionary is not under license control.

Hardware Requirements

All products support the weak password dictionary maintenance function.

Feature Requirements

Limitations on the content of the weak password dictionary file are as follows:

- Each weak password can contain a maximum of 128 characters. If the length of a weak password exceeds 128 characters, the weak password does not take effect.

- When the weak password dictionary file is loaded, the system does not check whether the file is empty or damaged. In this case, no weak password is loaded or the loaded content is unpredictable.
- A maximum of 1000 weak passwords can be loaded to a device. If the number of valid weak passwords in a file exceeds 1000, the file fails to be loaded.
- After a weak password dictionary file is loaded, it is locked and cannot be modified or deleted. You can unload the weak password dictionary file to unlock it.

Procedure

- Load a weak password dictionary.
  load-weak-password-dictionary filename *filename*

  Before the loading, ensure that the weak password dictionary in .txt format has been uploaded to the device.
- Unload the weak password dictionary.
  unload-weak-password-dictionary

----End

Verifying the Configuration

Run the **display system/weak-passwords** command to check the weak passwords that are prohibited from being used on the device.

11.8 PKI Configuration

11.8.1 Overview of PKI

Definition

Public Key Infrastructure (PKI) provides certificate management in compliance with established standards. It uses public keys to provide security services for all network applications. PKI is the core of information security and basis of e-commerce.

Purpose

As the network and information technologies develop, e-commerce is widely used and accepted. However, e-commerce has the following problems:

- The transaction parties cannot verify the identities of each other.
- Data may be eavesdropped and tampered with during transmission. Information is not secure.
- No paper receipt is used in transaction, making arbitration difficult.

PKI uses public keys to implement identity verification, confidentiality, data integrity, and non-repudiation of transactions. Therefore, PKI is widely used in network communication and transactions, especially e-government and e-commerce.

Benefits

- Certificate authentication allows users to authenticate network devices to which they connect, ensuring that users connect to secure and legal networks.
- Cryptography protects data against tampering and eavesdropping so that data is securely transmitted.
- Digital signature ensures that data is accessible to only authorized devices and users, protecting data privacy.
- PKI prevents unauthorized users from connecting to enterprise networks.
- PKI establishes secure connections between enterprise branches to ensure data security.

11.8.2 Understanding PKI

11.8.2.1 Basic Concepts of PKI

11.8.2.1.1 Cryptography

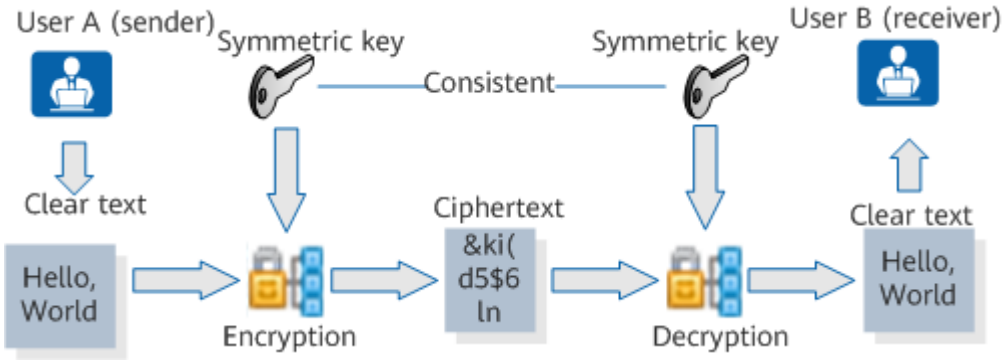
Cryptography is the basis for secure information transmission on networks. Cryptography is to convert plaintext (to be hidden) into ciphertext (unreadable data) using mathematics.

Symmetric Key Cryptography

Symmetric key cryptography, which is also called shared key cryptography, uses the same key to encrypt and decrypt data.

Figure 11-24 shows the symmetric key encryption and decryption process.

Figure 11-24 Symmetric key encryption and decryption process



Users A and B have negotiated the symmetric key. The encryption and decryption process is as follows:

1. User A uses the symmetric key to encrypt data and sends the encrypted data to user B.
2. User B decrypts the data using the symmetric key and gets the original data.

   Symmetric key cryptography features high efficiency, simple algorithm, and low cost. It is suitable for encrypting a large amount of data. However, it is

difficult to implement because the two parties must exchange their keys securely before communication. Besides, it is difficult to expand because each pair of communicating parties needs to negotiate keys, and n users needs to negotiate n*(n-1)/2 different keys.

The algorithms commonly used in symmetric key cryptography include Data Encryption Standard (DES), Triple Data Encryption Standard (3DES), and Advance Encrypt Standard (AES).

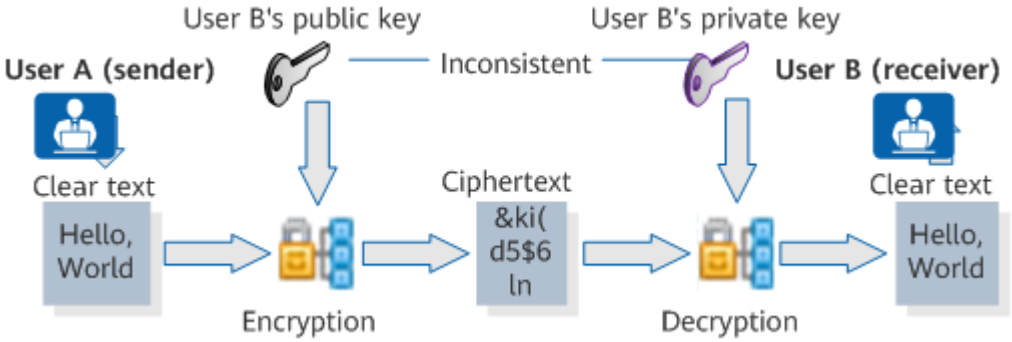
Public Key Cryptography

Public key cryptography, which is also called asymmetric key cryptography, uses different keys (public and private) for data encryption and decryption. The public key is open to public, and the private key is possessed by only the owner.

Public key cryptography prevents the security risks in the distribution and management of a symmetric key. In an asymmetric key pair, the public key is used to encrypt data and the private key is used to decrypt data. The two parties do not need to exchange keys before a secure communication session. The sender uses the public key of the receiver to encrypt the data, and the receiver uses its own private key to decrypt data. The receiver's private key is only known by the receiver, so the data is secure.

Figure 11-25 shows the public key encryption and decryption process.

Figure 11-25 Public key encryption and decryption process



Assume that user A has the public key of user B. The encryption and decryption process is as follows:

1. User A uses the public key of user B to encrypt data and sends the encrypted data to user B.
2. User B decrypts the data using its own private key and gets the original data.

Attackers cannot use one key in a key pair to figure out the other key. The data encrypted by a public key can only be decrypted by the private key of the same user. However, the public key cryptography requires a long time to encrypt a large amount of data, and the encrypted data is too long, consuming much bandwidth.

Public key cryptography is suitable for encrypting sensitive information such as keys and identities to provide higher security.

The algorithms commonly used in public key cryptography include Diffie-Hellman (DH), Ron Rivest, Adi Shamirh, LenAdleman (RSA), and Digital Signature Algorithm (DSA).

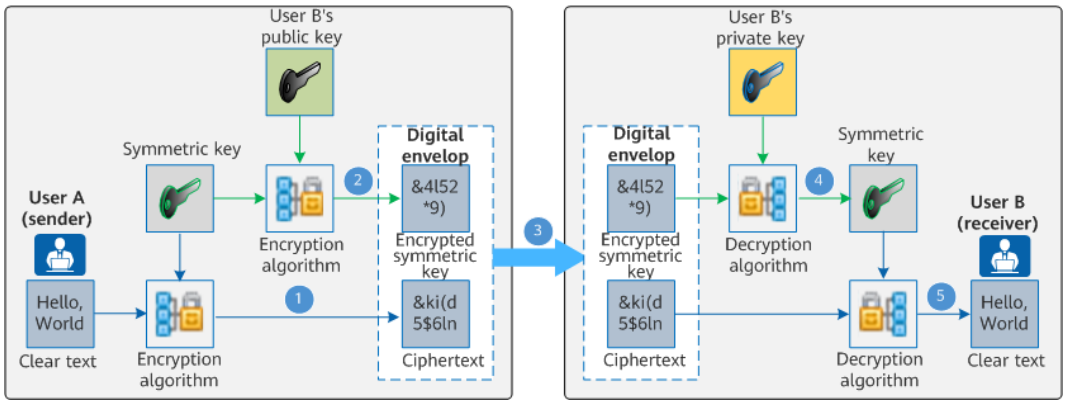
11.8.2.1.2 Digital Envelope and Digital Signature

Digital Envelope

A digital envelope contains the data that combines the symmetric key encrypted using the receiver's public key and the data encrypted using the symmetric key. When receiving a digital envelope, the receiver uses its own private key to decrypt the digital envelope and obtains the symmetric key. The digital envelope has the advantages of both symmetric key cryptography and public key cryptography. It speeds up symmetric key distribution and public key encryption, while improving key security, extensibility, and transmission efficiency.

Figure 11-26 shows the encryption and decryption process for a digital envelope.

Figure 11-26 Digital envelope encryption and decryption process



Assume that user A has the public key of user B. The encryption and decryption process is as follows:

1. User A uses a symmetric key to encrypt data.
2. User A uses the public key of user B to encrypt the symmetric key and generate a digital envelope.
3. User A sends the digital envelope and encrypted data to user B.
4. User B uses its own private key to decrypt the digital envelope and obtains the symmetric key.
5. User B uses the symmetric key to decrypt the data and obtains the original data.

However, the following vulnerability should be noted regarding the digital envelope: An attacker may intercept information from user A, use its own symmetric key to encrypt forged information, use the public key of user B to encrypt its own symmetric key, and send the information to user B. User B then decrypts and considers this information to have been sent from user A. To address this problem, the digital signature is used to ensure that the received information is sent from the correct sender.

Digital Signature

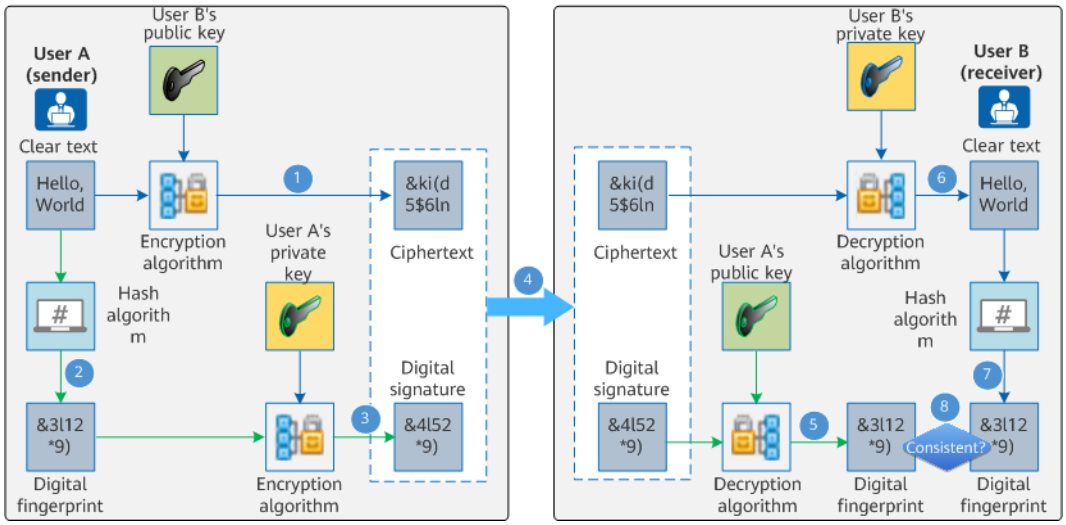
A digital envelope cannot determine if the information received actually originates from the sender. Instead, a digital signature can verify the identity of the sender as well as whether the information received has been tampered with.

A digital signature is generated by the sender by encrypting the digital fingerprint using its own private key. The receiver then uses the sender's public key to decrypt the digital signature and obtain the digital fingerprint.

A digital fingerprint, which is also called information digest, is generated by the sender using the hash algorithm on plaintext information. The sender sends both digital fingerprint and plaintext to the receiver, and the receiver uses the same hash algorithm to calculate the digital fingerprint on the plaintext. If the two fingerprints are the same, the receiver knows that the information has not been tampered with.

Figure 11-27 shows the encryption and decryption process for a digital signature.

Figure 11-27 Digital signature encryption and decryption process



Assume that user A has the public key of user B. The encryption and decryption process is as follows:

1. User A uses the public key of user B to encrypt data.
2. User A performs hash on the plaintext and generates a digital fingerprint.
3. User A uses its own private key to encrypt the digital fingerprint, generating the digital signature.
4. User A sends both the ciphertext and digital signature to user B.
5. User B uses the public key of user A to decrypt the digital signature, obtaining the digital fingerprint.
6. After receiving the ciphertext from user A, user B uses its own private key to decrypt the information, obtaining the plaintext information.
7. User B performs hash on the plaintext and generates a digital fingerprint.
8. User B compares the generated fingerprint with the received one. If the two fingerprints are the same, user B accepts the plaintext; otherwise, user B discards it.

The digital signature proves that information has not been tampered with and verifies the sender's identity, and can be used in conjunction with the digital envelope.

However, the digital signature still has a vulnerability. If the attacker modifies the public key of user B, then user A obtains the attacker's public key. The attacker can obtain information from user B to user A, sign the forged information using its own private key, and send the forged information encrypted using user A's public key to user A. After receiving the encrypted information, user A decrypts the information and verifies that the information has not been tampered with. In addition, user A considers that the information was sent by user B.

11.8.2.1.3 Digital Certificate

A digital signature cannot determine whether a public key belongs to a specific owner, as any entity can generate public and private keys. Therefore, a secure and reliable carrier is required to exchange public keys. This carrier is a digital certificate.

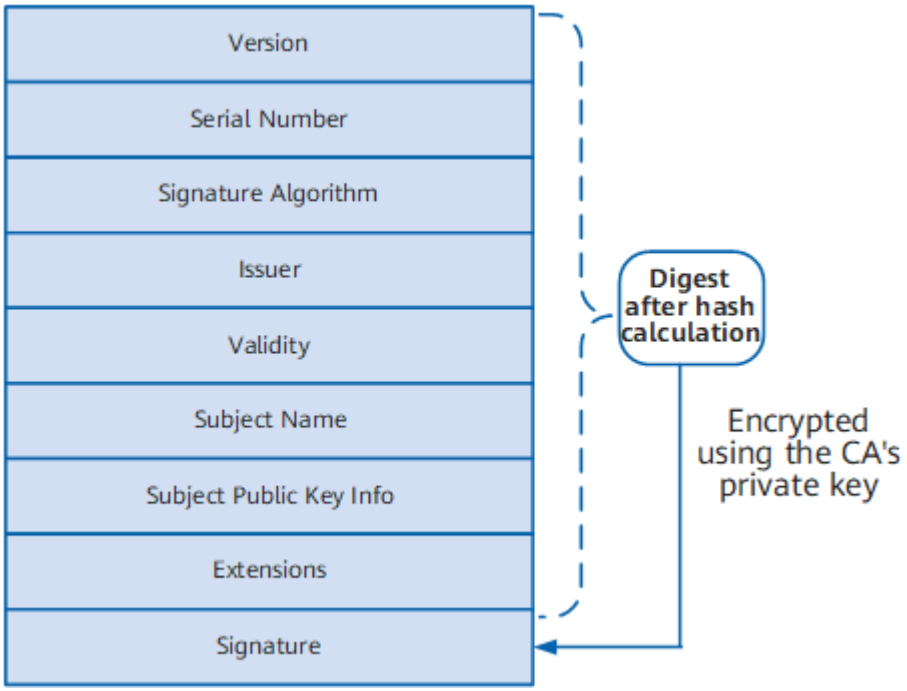
A digital certificate is a digitally signed file issued by a CA, containing the owner's public key and identity information.

A digital certificate similar to an electronic copy of a passport or an ID card is typically used for identity verification on the network. It ensures that one public key is possessed by only one owner.

Digital Certificate Structure

A simplest digital certificate contains mandatory information such as public key, name, and digital signature of the CA. In most cases, the certificate also includes information such as the public key validity period, issuer name (CA), and certificate serial number. The certificate structure complies with X.509 v3. **Figure 11-28** shows the typical structure of a digital certificate.

Figure 11-28 Digital certificate structure diagram



The meaning of each field in the digital certificate is as follows:

- Version: version of X.509. Generally, the v3 (0x2) is used.
- Serial Number: a positive and unique integer assigned by the issuer to the certificate. Each certificate is uniquely identified by the issuer name and the serial number.
- Signature Algorithm: signature algorithm used by the issuer to sign the certificate.
- Issuer: name of the device that has issued a certificate. It must be the same as the subject name in the digital certificate. Generally, the issuer name is the CA server's name.
- Validity: time interval during which a digital certificate is valid, including the start and end dates. The expired certificates are invalid.
- Subject: name of the entity that possesses a digital certificate. In a self-signed certificate, the issuer name is the same as the subject name.
- Subject Public Key Info: public key and the algorithm with which the key is generated.
- Extensions: a sequence of optional fields such as key usage and CRL address (URL).
- Signature: signature signed on a digital certificate by the issuer using the private key.

The process of generating a certificate signature is as follows: The CA first uses a cryptographic hash algorithm (such as SHA1) to generate digest information of the certificate, and then uses a public-key cryptography algorithm (such as RSA) and a private key of the CA to encrypt the digest information and finally generate a signature. These operations are performed on the CA before the certificate is issued.

Digital Certificate Types

There are three types of certificates, as described in **Table 11-17**.

Table 11-17 Certificate types

Type	Definition	Description
CA certificate	CA's own certificate. If a PKI system does not have a hierarchical CA structure, the CA certificate is the self-signed certificate. If a PKI system has a hierarchical CA structure, the top CA is the root CA, which owns a self-signed certificate.	An applicant trusts a CA by verifying its digital signature. Any applicant can obtain the CA's certificate (including the public key) to verify the local certificate issued by the CA.
Local certificate	A certificate issued by a CA to the applicant.	-
Self-signed certificate	• A self-signed certificate is issued by a device to itself and is signed by the CA predefined on the device. That is, the certificate issuer is the same as the certificate subject. This type of certificate contains signature information, and it does not require signature application. • An unsigned certificate, as its name implies, is not signed. It is issued by a device to itself. A signature needs to be obtained from the CA, and the certificate issuer is the CA.	A device can generate a self-signed or unsigned certificate for itself, which is a simple certificate issuing function. The device does not support lifecycle management (such as certificate update and revocation) of its self-signed certificate. To ensure security of the device and certificate, you are advised to replace the self-signed certificate with a local certificate.

Certificate Formats

Three certificate formats are supported, as described in **Table 11-18**.

Table 11-18 Certificate formats

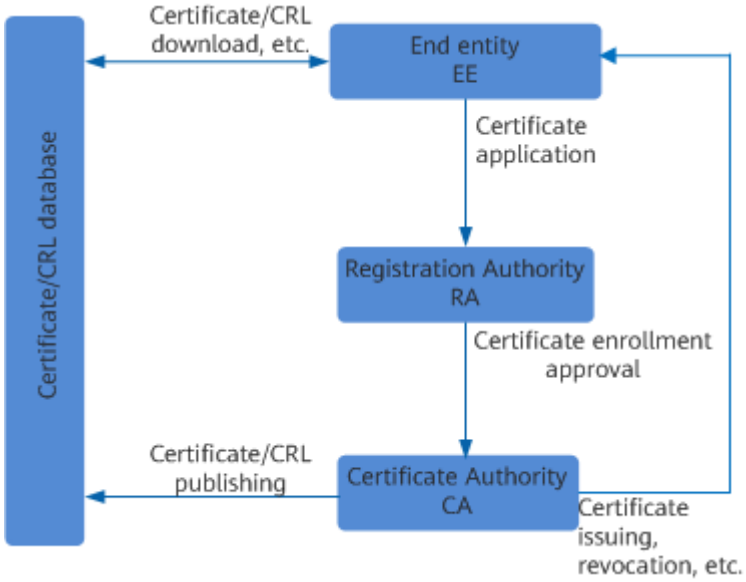
Format	Description	Description
PKCS#12	Saves certificate files in binary format, including or excluding the private key. Commonly used file name extensions include .P12 and .PFX.	If the file name extension of a certificate is .CER or .CRT, use Notepad to open this certificate and check its content to differentiate the certificate format. • If the certificate starts with "-----BEGIN CERTIFICATE-----" and ends with "-----END CERTIFICATE-----", the certificate format is PEM. • If the certificate content is displayed as garbled characters, the certificate format is DER.
DER	Saves certificate files in binary format, excluding the private key. Commonly used file name extensions include .DER, .CER, and .CRT.	
PEM	Saves certificate files in ASCII format, including or excluding the private key. Commonly used file name extensions include .PEM, .CER, and .CRT.	

11.8.2.2 PKI System Structure

PKI System Composition

As shown in **Figure 11-29**, a PKI system consists of the end entity, registration authority (RA), certification authority (CA), and certificate/certificate revocation list (CRL) database.

Figure 11-29 PKI system composition



End Entity

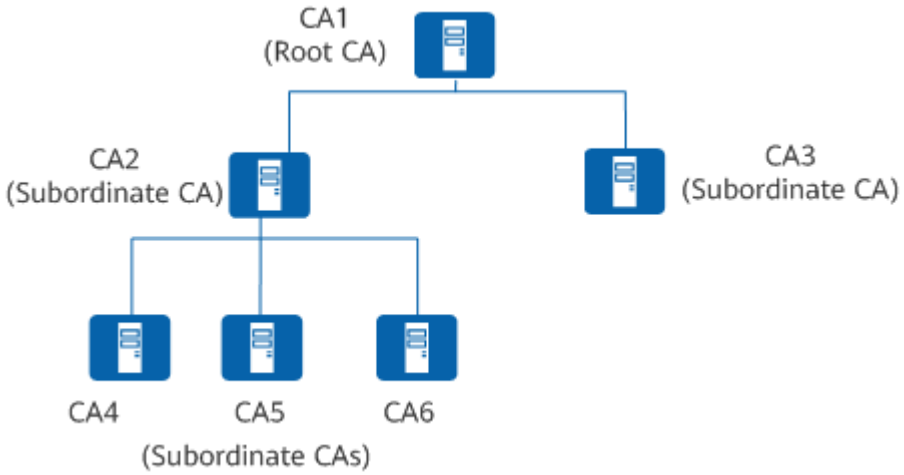
An end entity, or PKI entity, is the end user of PKI products or services. It can be an individual, an organization, a device (for example, a router or firewall), or process running on a computer.

CA

The CA is the trusted entity that issues and manages digital certificates. It is an authoritative, trusted, and impartial organization. Generally, a CA is a server.

Figure 11-30 shows a hierarchical CA. The CA on the top of the hierarchy is the root CA and the others are subordinate CAs.

Figure 11-30 Hierarchical CA



- The root CA is the first CA (trustpoint) in the PKI system. It issues certificates to subordinate CAs, computers, users, and services. In most certificate-based applications, the root CA can be traced through the certificate chain. The root CA holds a self-signed certificate.

- A subordinate CA can only obtain a certificate from its upper-level CA. The upper-level CA can be the root CA or another subordinate CA authorized by the root CA to issue certificates. The upper-level CA is responsible issuing and managing certificates of lower-level CAs, and the CAs at the bottom issue certificates to end entities. For example, CA 2 and CA 3 are subordinate CAs, holding the certificates issued by CA 1, and CA 4, CA 5, as well as CA 6 are also subordinate CAs, holding the certificates issued by CA 2.

When a PKI entity trusts a CA, the trust is derived along the certificate chain. A certificate chain is a set of certificates from the end entity to the root certificate. When a PKI entity in communication receives a certificate to be authenticated, the PKI entity verifies each issuer along the certificate chain.

Certificate management is the primary function of CAs, and includes issuing, revoking, querying, and archiving certificates, as well as publishing CRLs.

RA

An RA enrolls and approves digital certificates. It is a proxy for the CA and provides extended applications of certificate issuing and management. The RA processes the certification enrollment and revocation requests from users, verifies user identities, and decides whether to submit certificate issuing or revocation requests to the CA.

While an RA is combined with a CA in most cases, it can also be independent of a CA, sharing the CA's workload and enhancing CA system security.

Certificate/CRL Database

The certificate/CRL database stores and manages information about certificates and CRLs and allows information query.

A certificate may need to be revoked for reasons such as entity name changing, private key leaking, or service interruptions. Revoking a certificate is to unbind the public key from the PKI entity identity information. The PKI system uses a CRL to revoke a certificate.

After a certificate is revoked, the CA issues a CRL to declare that the certificate is invalid and lists the serial numbers of revoked certificates. The CRL provides a method to verify certificate validity.

Certificate-related Operations

A PKI manages the entire lifecycle of certificates, including applying for, issuing, storing, downloading, installing, authenticating, updating, and revoking certificates.

Certificate Application

Certificate application, also known as certificate enrollment, is a process in which a PKI entity introduces itself to a CA, which then issues it a certificate. Generally, a PKI entity generates a pair of public/private keys. The public key and the identity information (included in the certificate enrollment request) of the PKI entity are sent to the CA to generate a local certificate. The private key is stored by the PKI entity and is used to generate a digital signature and decrypt the ciphertext sent by the peer entity. Currently, the device supports offline certificate application and CMPv2-based online certificate application.

Certificate Issuing

If an RA is available, the RA verifies the PKI entity's identity information when the PKI entity applies for a local certificate from a CA. After the verification, the RA sends the request to the CA. The CA generates a local certificate based on the public key and identity information of the PKI entity, and then returns the local certificate information to the RA. If no RA is available, the CA verifies the PKI entity's identity information.

In addition, a PKI entity can issue a self-signed certificate or a certificate without a signature to itself, implementing simple certificate issuing.

Certificate Storage

After the CA generates a local certificate, the CA or RA distributes the certificate to the certificate/CRL database. Users can download or browse directory of the certificates in the database.

Certificate Download

A PKI entity downloads an issued certificate in LDAP or out-of-band mode. The certificate can be a local certificate of the PKI entity, a CA/RA certificate, or a local certificate of another PKI entity.

Certificate Installation

In order for a downloaded certificate to take effect, it must be installed on the device (specifically imported to the device memory). The certificate can be a local certificate of the PKI entity, a CA/RA certificate, or a local certificate of another PKI entity.

Certificate Authentication

Each certificate installed on the local device must be authenticated to ensure validity before it is used. The main point of this is to check the CA signature on the certificate and ensure that the certificate is valid and not revoked.

Certificate Update

When a certificate expires or if its private key is leaked, it must be replaced by the PKI entity. You can manually apply for a new certificate or configure CMPv2 to implement automatic certificate update.

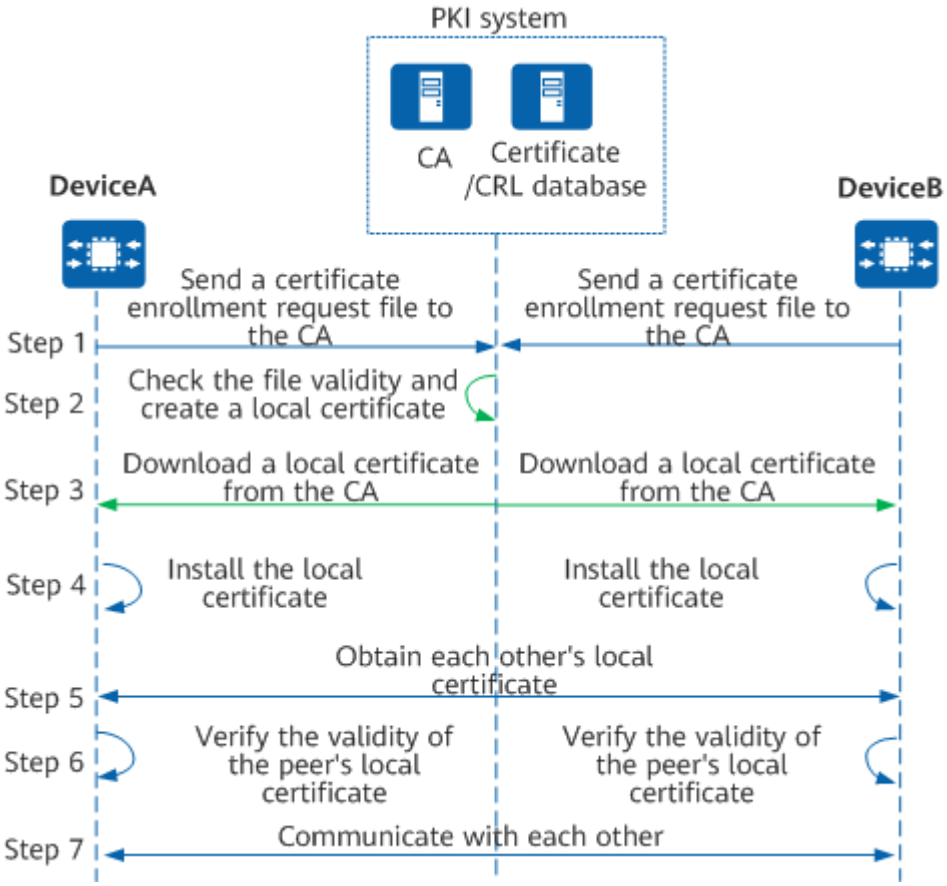
Certificate Revocation

In scenarios involving a change of user identity, user information, or public key; or due to user service suspension, the user must revoke the digital certificate, that is, unbind the public key from the user's identity information. A CA provides the certificate revocation function. When a PKI entity revokes its certificate in out-of-band mode, the CA stores the certificate in a CRL database.

11.8.2.3 PKI Working Mechanism

The following figure shows the process of applying for a certificate in offline mode.

Figure 11-31 PKI entity's offline application process



1. A PKI entity sends the certificate request file to the CA in out-of-band mode (for example, by disk or email), requesting the CA to create a certificate.
2. The CA checks the validity of the certificate request file. If the certificate request file is valid, the CA creates a certificate based on this file.
3. The PKI entity downloads the certificate to the local device in out-of-band mode (for example, by disk or email).
4. The PKI entity installs the local certificate to the device memory.
5. **Optional:**

   When PKI entities communicate with one another, they must obtain each other's local certificate and CA certificate.
6. The PKI entity uses the CRL to check whether the peer's local certificate is valid.
7. The PKI entity uses the public key in the peer's local certificate for encrypted communication only after confirming that the peer's local certificate is valid.

11.8.3 Configuration Precautions for PKI

Licensing Requirements

PKI is not under license control.

Hardware Requirements

Table 11-19 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

11.8.4 Default Settings for PKI

Table 11-20 describes the default settings for PKI.

Table 11-20 Default settings for PKI

Parameter	Default Setting
PKI realm	By default, the device has a PKI realm named **default**, which cannot be deleted or modified.
RSA key pair	RSA key pair file named **default**
Format of saved certificate request, certificate, and CRL	PEM
Certificate check method	CRL
Number of days in advance users are notified that the local or CA certificate is about to expire	90

11.8.5 Preconfiguration for Certificate Application

11.8.5.1 Configuring an RSA/SM2/ECC Key Pair

Context

A local certificate is digitally signed and issued by a CA. It is a bundle of a public key and a PKI entity's identity information. Before applying for a local certificate, you must configure an RSA/SM2/ECC key pair to generate public and private keys. The public key is sent by the PKI entity to the CA, and the peer uses this key to

encrypt clear text. In contrast, the private key is retained by the PKI entity, which uses it to digitally sign and decrypt the peer's ciphertext.

You can configure an RSA/SM2/ECC key pair using either of the following methods:

- Create an RSA/SM2/ECC key pair.

  You can directly create a key pair on the device without importing it to the device memory. When creating an RSA/SM2 key pair, you need to enter the number of bits of the public key, which ranges from 2048 to 4096. A longer public key indicates higher security but slower calculation. When creating an ECC key pair, you need to select the curve type of the public key. The number of bits of the public key is determined by the curve type.
- Import the RSA/SM2/ECC key pair.

  To use the RSA/SM2/ECC key pair generated by another PKI entity, upload the key pair to the device through FTP or SFTP and then import it to the device memory. Otherwise, the key pair does not take effect on the device.

For details about configuration parameters, see huawei-pki.yang.

Procedure

Step 1 Configure an RSA/SM2/ECC key pair as required.

- Creating an RSA/SM2/ECC key pair
  a. Go to the RPC node for creating a key pair.
  key-pair-create
  b. Access the list of key pairs to be created.
  key-pairs
  c. Configure the key pair name and type.
  key-pair name *name* **type** { **rsa** | **sm2** | **ecc** }
  d. **Optional:** When creating an RSA key pair, configure the number of bits of the public key, which ranges from 2048 to 4096. The default value is 3072.
  key-size *key-size*
  e. When creating an ECC key pair, configure the curve type.
  curve-type { **prime256v1** }
  f. **Optional:** To create more key pairs at the same time, exit to the upper-layer node, and repeat steps c to e.
  quit
- Importing the RSA/SM2/ECC key pair
  a. Go to the RPC node for importing a key pair.
  key-pair-import
  b. Access the list of key pairs to be imported.
  key-pairs
  c. Configure the key pair name and type.
  key-pair name *name* **type** { **rsa** | **sm2** | **ecc** }
  d. Configure the key pair file name and file format. RSA and ECC key pair files can be in PEM or PKCS12 format, and SM2 key pair files can be in PEM or DER format.
  file-name *file-name* **format** { **pem** | **der** | **pkcs12** }
  e. For a PKCS12 file, enter the password in interactive mode. If the file does not have a password, enter any content.


```
password
Enter password:password
Confirm password:password
```

f. **Optional:** To import more key pairs at the same time, exit to the upper-layer node, and repeat steps c to e.

```
quit
```

Step 2 Commit the configuration.

```
emit
```

----End

Verifying the Configuration

- In the **pki** directory, run the **display key-pair-infos** command to check information about all key pairs on the device.
- In the **pki** directory, run the **display key-pair-infos/key-pair-info[name=***name***][type=** { **rsa** | **sm2** | **ecc** } **]** command to check information about the key pair with the specified name and type.

Follow-up Procedure

- Destroying a specified RSA/SM2/ECC key pair

  a. Go to the RPC node for destroying a key pair.

  ```
  key-pair-destory
  ```

  b. Access the list of key pairs to be destroyed.

  ```
  key-pairs
  ```

  c. Configure the key pair name and type.

  ```
  key-pair name key-name type { rsa | sm2 | ecc }
  ```

  d. **Optional:** To destroy more key pairs at the same time, exit to the upper-layer node, and repeat step 3.

  ```
  quit
  ```

  e. Commit the configuration.

  ```
  emit
  ```

- Searching for the RSA/SM2/ECC key pair corresponding to the certificate

  a. In the **pki** directory, run the **display cert-key-infos** command to check information about the key pairs corresponding to all certificates on the device.

  b. In the **pki** directory, run the **display cert-key-infos/cert-key-info[cert-name=***cert-name***]** command to check information about the key pair corresponding to the certificate with a specified name on the device.

11.8.5.2 Configuring a PKI Entity

Context

A local certificate is digitally signed and issued by a CA. It is a bundle of a public key and a PKI entity's identity information. PKI entity information contains the identity information of a PKI entity, based on which the CA identifies a certificate applicant. As such, when applying for a local certificate, the PKI entity must send PKI entity information to the CA.

The entity information includes the common name, fully qualified domain name (FQDN), IP address, and email address. The common name is mandatory, while others are optional. The preceding information is contained in the certificate.

For details about configuration parameters, see huawei-pki.yang.

Procedure

Step 1 Access the configuration mode.

```
edit-config
```

Step 2 Go to the PKI directory.

```
pki
```

Step 3 Access the list of entities to be configured.

```
entitys
```

Step 4 Create a PKI entity and enter the PKI entity view, or enter the PKI entity view directly.

```
entity name entity-name
```

Step 5 Configure a common name for the PKI entity.

```
common-name common-name
```

Step 6 **Optional:** Set other parameters for the PKI entity.

To uniquely identify an applicant, you can run the following optional commands to configure the alias name for the PKI entity. If you do not configure alias names for the PKI entities that have the same common name, each of them will fail to apply for a certificate.

Operation	Command
Configure an IP address for the PKI entity.	**ip-address** { *ipv4-address* | *ipv6-address* }
Configure an FQDN for the PKI entity.	**fqdn** *fqdn-name*
Configure an email address for the PKI entity.	**email** *email-address*
Configure a country code for the PKI entity.	**country** *country-code*
Configure a locality name for the PKI entity.	**locality** *locality-name*
Configure a state name for the PKI entity.	**state** *state-name*
Configure an organization name for the PKI entity.	**organization** *organization-name*
Configure a department name for the PKI entity.	**department** *department-name*

Step 7 To configure multiple entities at the same time, exit to the upper-layer node, and repeat steps 4 to 6.

```
quit
```

Step 8 Commit the configuration.

```
commit
```

Step 9 Exit the configuration mode and return to the top directory.

```
return
```

----End

Verifying the Configuration

- In the **pki** directory, run the **display entitys** command to view information about all PKI entities.
- In the **pki** directory, run the **display entitys/entity[name=***entity-name***]** command to check information about the PKI entity with a specified name.

11.8.5.3 Downloading a CA Certificate

Context

When applying for a local certificate, the PKI entity sends the certificate enrollment request to the CA. To improve transmission security, the PKI entity must use the CA's public key to encrypt the certificate enrollment message. Therefore, the PKI entity must download and obtain the CA certificate and then obtain the CA's public key.

A CA certificate can be downloaded using the following method:

- Out-of-band mode: Obtain the CA certificate in out-of-band mode (for example, by disk or email) and then upload it to the device storage.

For details about configuration parameters, see huawei-pki.yang.

Procedure

- Download a CA certificate in out-of-band mode.

  After you obtain a CA certificate in out-of-band mode (for example, by disk or email), manually upload it to the device storage. You can also download the CA certificate through the administrator's PC and then upload it to the device storage through FTP or SFTP. SFTP is recommended because it is more secure than FTP.

----End

11.8.5.4 Installing a CA Certificate

Context

Before installing a CA certificate obtained in out-of-band mode (for example, by disk or email), upload it to the **flash:** directory on the device.

After the CA certificate is saved to the specified directory, you also need to import it to the device memory. After the device restarts, the system automatically loads the certificate.

For details about configuration parameters, see huawei-pki.yang.

NOTE

To prevent a failure to install the CA certificate, ensure that the CA certificate file size does not exceed 1 MB.

By default, a PKI realm named **default** exists. The default realm cannot be modified or deleted.

Procedure

Step 1 **Optional:** Download the CA certificate to the **flash:** directory.

To obtain a CA certificate in out-of-band mode and upload it to the storage of the device through FTP or SFTP, perform the following operations. SFTP is recommended because it is more secure than FTP.

1. Access the RPC node of FTP.
   ftpc-transfer-file
2. Configure the server port number (usually 21) and IP address.
   server-port *server-port* **server-ipv4-address** *server-ipv4-address*
3. Configure the command type.
   command-type get
4. Configure a user name.
   user-name *user-name*
5. Configure a password in interactive mode.
   password
   Enter password:*password*
   Confirm password:*password*
6. Specify the name of the certificate file to be downloaded on the server.
   remote-file-name *remote-file-name*
7. **Optional:** Specify the name of the downloaded certificate file on the local end.
   local-file-name *local-file-name*

Step 2 **Optional:** Import the predefined CA certificate to the default realm.

1. Go to the RPC node for importing a certificate.
   certificate-import
2. Access the list of certificates to be imported.
   certificates
3. Configure the name and type of the predefined CA certificate.
   certificate name default_ca.cer type default-ca
4. Configure the name of the PKI realm.
   domain-name default
5. Commit the configuration.
   emit

A predefined CA certificate is stored in the non-volatile random-access memory (NVRAM) before the device is delivered. To use the predefined CA certificate, run the preceding commands to load the certificate to the default realm. The predefined CA certificate can be deleted. After it is deleted, you can import other

CA certificates to the default realm. However, **default_ca.cer** is the name reserved for the predefined CA certificate. An imported certificate cannot be named **default_ca.cer**. To restore the deleted predefined CA certificate, run the preceding commands to load the certificate to the default realm.

Step 3 Create a PKI realm.

1. Access the configuration mode.
   edit-config
2. Go to the **pki** directory.
   pki
3. Access the list of realms to be configured.
   domains
4. Create a PKI realm and enter its view, or enter the view of an existing PKI realm.
   domain name *domain-name*
5. **Optional:** To configure more PKI realms at the same time, exit to the upper-layer node, and repeat step d.
   quit
6. Commit the configuration.
   commit
7. Exit the configuration mode and return to the top directory.
   return

Step 4 Import the CA certificate into the device memory.

1. Go to the RPC node for importing a certificate.
   certificate-import
2. Access the list of certificates to be imported.
   certificates
3. Configure the name and type of the predefined CA certificate.
   certificate name *name* **type ca**
4. Configure the name of the PKI realm.
   domain-name *domain-name*
5. **Optional:** Configure the certificate file format.
   format { **pem** | **der** | **pkcs12** }
6. For a PKCS12 file, enter the password in interactive mode. If the file does not have a password, enter any content.
   password
   Enter password:*password*
   Confirm password:*password*
7. **Optional:** To import more CA certificates at the same time, return to the upper-layer node, and repeat steps c to f.
   quit
8. Commit the configuration.
   emit

NOTE

If the format of the certificate to be imported is not specified, the system automatically identifies and imports the certificate. If the certificate file format is PKCS12, you must set the format to PKCS12 and set a password.

----End

Verifying the Configuration

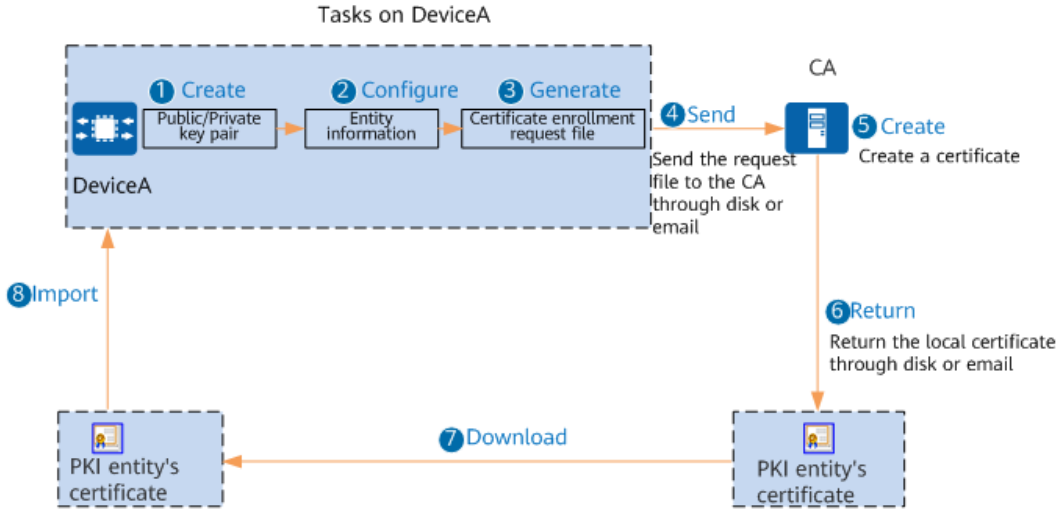
- In the **pki** directory, run the **display certificate-infos** command to check the information about all certificates loaded to the device.
- In the **pki** directory, run the **display certificate-infos/certificate-info[type=ca][name=***name***][domain-name=***domain-name***]** command to check the information about the specified CA certificate loaded to the specified realm on the device.

11.8.6 Applying for a Certificate in Offline Mode

11.8.6.1 Understanding Offline Certificate Application

Certificate application, also known as certificate enrollment, is a process in which a PKI entity introduces itself to a CA, which then issues it a certificate. To obtain a local certificate offline, you need to configure PKI entity information, configure an RSA key pair, apply for a local certificate, and install the local certificate. **Figure 11-32** shows the process of applying for a certificate in offline mode.

Figure 11-32 Process of applying for a certificate in offline mode



1. Create a public/private key pair on DeviceA. The public key information is required during certificate application.
2. Configure entity information. When applying for a certificate, DeviceA must provide the CA with information that can prove its identity. The entity information represents identity information, including the common name, FQDN, IP address, and email address. The common name is mandatory, while others are optional. After entity information is configured, reference the entity information in the PKI realm.
3. Generate a certificate enrollment request file. The generated certificate enrollment request file is named *PKI realm name*.**req** and saved in the storage of DeviceA.
4. After the certificate enrollment request file is generated, it can be sent to the CA in out-of-band mode (for example, by disk or email).

5. After approving the request, the CA creates a certificate based on the certificate enrollment request file.
6. After the certificate is generated, DeviceA's local certificate **DeviceA.cer** is obtained in out-of-band mode (for example, by disk or email).
7. Download **DeviceA.cer** to the **flash:** directory on DeviceA.
8. Import **DeviceA.cer** to the memory of DeviceA.

11.8.6.2 Applying for a Local Certificate in Offline Mode

Prerequisites

You have completed the preconfiguration for a local certificate application. For details, see **11.8.5 Preconfiguration for Certificate Application**.

Context

You can apply for a local certificate offline. To do this, you need to first generate a certificate enrollment request file on the device, and then send the file to the CA in out-of-band mode (for example, by disk or email).

For details about configuration parameters, see huawei-pki.yang.

Procedure

Step 1 Access the configuration mode.

```
edit-config
```

Step 2 Go to the **pki** directory.

```
pki
```

Step 3 Access the list of realms to be configured.

```
domains
```

Step 4 Create a PKI realm and enter its view, or enter the view of an existing PKI realm.

```
domain name domain-name
```

By default, a PKI realm named **default** exists in the system. This realm cannot be modified or deleted.

A PKI realm is locally available. It is unavailable to CAs or other devices. Each PKI realm has its own parameter settings.

Step 5 Specify the PKI entity that applies for a certificate.

```
entity entity-name
```

The PKI entity specified by *entity-name* must have been created using the **pki entitys entity name** *entity-name* command.

Step 6 Configure the key pair used to apply for a certificate in offline mode as required.

1. Go to the node for configuring a key pair.
   ```
   key-pair
   ```
2. Configure the key pair name and type.
   ```
   name key-name type { rsa | sm2 | ecc }
   ```
3. Return to the upper-layer node.


```
quit
```

Step 7 Configure the digest algorithm used to sign certificate enrollment requests.

```
digest-algorithm { md5 | sha1| sha-256 | sha-384 | sha-512 | sm3 }
```

By default, the digest algorithm used to sign certificate enrollment requests is **sha-256**.

The digest algorithm used on a PKI entity must be the same as that used on the CA server. Note that the MD5 and SHA1 algorithms are insecure, so you are advised to use the more secure SHA2 algorithms (SHA-256, SHA-384, and SHA-512).

In a PKI realm, if the SM2 key pair is used to apply for a certificate in offline mode, the digest algorithm used to sign certificate enrollment requests must be configured as SM3. If the RSA key pair is used to apply for a certificate in offline mode, the digest algorithm used to sign certificate enrollment requests cannot be configured as SM3. Otherwise, offline certificate application fails.

NOTE

For security purposes, you are advised to use SHA2 algorithms (SHA-256, SHA-384, and SHA-512) instead of the less secure algorithms MD5 and SHA1.

The **MD5** and **SHA1** parameters can be used only after the weak security algorithm/protocol feature package is installed.

Step 8 **Optional:** Configure the certificate public key usage attribute.

```
key-usage { encipherment | signature }
```

Step 9 To configure more PKI realms at the same time, exit to the upper-layer node, and repeat steps 4 to 8.

```
quit
```

Step 10 Commit the configuration.

```
commit
```

Step 11 Exit the configuration mode and return to the top directory.

```
return
```

Step 12 Set parameters to save certificate enrollment information into a file in PKCS#10 format.

1. Go to the RPC node for generating a certificate enrollment request file.
   ```
   csr-generate
   ```
2. Access the list of CSRs to be generated.
   ```
   csrs
   ```
3. Configure the PKI realm of the CSR to be generated.
   ```
   csr domain-name domain-name
   ```
4. Configure the CSR file name.
   ```
   file-name file-name
   ```
5. Configure the challenge password in interactive mode. The challenge password used by the PKI entity must be the same as that configured on the CA server. If the CA server does not require a challenge password, this challenge password does not need to be configured.
   ```
   password
   Enter password:password
   Confirm password:password
   ```
6. **Optional:** To generate more certificate enrollment request files at the same time, return to the upper-layer node, and repeat steps c to e.


```
quit
```

7. Commit the configuration.

```
emit
```

Step 13 Enable the device to send the certificate enrollment request file to the CA in out-of-band mode (for example, by disk or email) to apply for a local certificate.

----End

Verifying the Configuration

In the **pki** directory, run the **display domains/domain[name=***name***]** command to check information about the PKI realm.

11.8.6.3 Downloading a Local Certificate

Context

The device often obtains the local certificate using the following method depending on the service types provided by the CA server:

- Out-of-band mode: The device's local certificate is obtained in out-of-band mode (for example, by disk or email) and then uploaded to the device storage.

For details about configuration parameters, see huawei-pki.yang.

Procedure

- Download the local certificate in out-of-band mode.

  After you obtain a local certificate in out-of-band mode (for example, by disk or email), manually upload it to the device storage. You can also download the local certificate through the administrator's PC and then upload it to the device storage through FTP or SFTP. SFTP is recommended because it is more secure than FTP.

----End

Verifying the Configuration

- Run the **ls** command in the **flash:** directory of the device to check the local certificate file in the device storage.

11.8.6.4 Installing the Local Certificate

Context

Before installing a local certificate obtained in out-of-band mode (for example, by disk or email), upload it to the **flash:** directory on the device.

After the local certificate is stored in the preceding directory, you need to manually import it to the device memory. After the device restarts, the system automatically loads the certificate.

For details about configuration parameters, see huawei-pki.yang.

NOTE

To prevent a failure to install the local certificate, ensure that the certificate file size does not exceed 1 MB.

By default, the public system has a PKI realm named **default**; the local certificate predefined on the device is stored in the default realm, which can be modified but not deleted.

The predefined local certificate that identifies a Huawei device provides certificate authentication for user login services of the device by default.

Procedure

Step 1 **Optional:** Download the local certificate to the **flash:** directory.

To obtain a local certificate in out-of-band mode and upload it to the storage of the device through FTP or SFTP, perform the following operations. SFTP is recommended because it is more secure than FTP.

1. Access the RPC node of FTP.
   ftpc-transfer-file
2. Configure the server port number (usually 21) and IP address.
   server-port *server-port* **server-ipv4-address** *server-ipv4-address*
3. Configure the command type.
   command-type get
4. Configure a user name.
   user-name *user-name*
5. Configure a password in interactive mode.
   password
   Enter password:*password*
   Confirm password:*password*
6. Specify the name of the certificate file to be downloaded on the server.
   remote-file-name *remote-file-name*
7. **Optional:** Specify the name of the downloaded certificate file on the local end.
   local-file-name *local-file-name*

Step 2 **Optional:** Import the predefined local certificate to the default realm.

1. Go to the RPC node for importing a certificate.
   certificate-import
2. Access the list of certificates to be imported.
   certificates
3. Configure the name and type of the predefined CA certificate.
   certificate name default_local.cer type default-local
4. Configure the name of the PKI realm.
   domain-name default
5. Commit the configuration.
   emit

The predefined local certificate can be deleted, and only this certificate can be used in the default realm. In addition, **default_local.cer** is the name of the predefined local certificate reserved by the system. An imported certificate cannot be named **default_local.cer**. To restore the predefined local certificate after it is deleted, run the preceding commands to load the certificate from the NVRAM to the default realm.

Step 3 Create a PKI realm.

1. Access the configuration mode.
   edit-config
2. Go to the **pki** directory.
   pki
3. Access the list of realms to be configured.
   domains
4. Create a PKI realm and enter its view, or enter the view of an existing PKI realm.
   domain name *domain-name*
5. **Optional:** To configure more PKI realms at the same time, exit to the upper-layer node, and repeat step d.
   quit
6. Commit the configuration.
   commit
7. Exit the configuration mode and return to the top directory.
   return

Step 4 Import the local certificate into the device memory.

- When applying for a local certificate from the CA based on the PKI entity information and RSA key pair created in the local PKI entity, you only need to run the following commands to import the local certificate into the device memory. The RSA key pair is already imported into the memory by default during its creation.
  a. Go to the RPC node for importing a certificate.
     certificate-import
  b. Access the list of certificates to be imported.
     certificates
  c. Configure the name and type of the predefined CA certificate.
     certificate name *name* **type local**
  d. Configure the name of the PKI realm.
     domain-name *domain-name*
  e. **Optional:** Configure the certificate file format.
     format { **pem** | **der** | **pkcs12** }
  f. For a PKCS12 file, enter the password in interactive mode. If the file does not have a password, enter any content.
     password
     Enter password:*password*
     Confirm password:*password*
  g. **Optional:** To import more local certificates at the same time, return to the upper-layer node, and repeat steps c to f.
     quit
  h. Commit the configuration.
     emit
- To use the key pair generated by another PKI entity and the certificate of another PKI entity, you need to import the corresponding key pair file when importing the certificate.
  a. Go to the RPC node for importing a key pair.
     key-pair-import
  b. Access the list of key pairs to be imported.
     key-pairs

c. Configure the key pair name and type.
key-pair name *name* **type** { **rsa** | **sm2** | **ecc** }

d. Configure the key pair file name and file format. RSA and ECC key pair files can be in PEM or PKCS12 format, and SM2 key pair files can be in PEM or DER format.
file-name *file-name* **format** { **pem** | **der** | **pkcs12** }

e. For a PKCS12 file, enter the password in interactive mode. If the file does not have a password, enter any content.
password
Enter password:*password*
Confirm password:*password*

f. **Optional:** To import more key pairs at the same time, exit to the upper-layer node, and repeat steps c to e.
quit

g. Commit the configuration.
emit

NOTE

If the format of the certificate to be imported is not specified, the system automatically identifies and imports the certificate. If the certificate file format is PKCS12, you must set the format to PKCS12 and set a password.

----End

Verifying the Configuration

- In the **pki** directory, run the **display certificate-infos** command to check the information about all certificates loaded to the device.
- In the **pki** directory, run the **display certificate-infos/certificate-info[type=local][name=***name***][domain-name=***domain-name***]** command to check the information about the specified local certificate loaded to the specified realm on the device.

11.8.6.5 Configuring Certificate Revocation Status Check

Context

The CA sometimes needs to unbind a public key from related PKI entities due to factors such as user name change, private key leak, or service interruption. Therefore, before using each certificate, you must check whether the certificate is revoked to ensure its validity.

The device provides the following methods for checking the certificate revocation status: CRL and None.

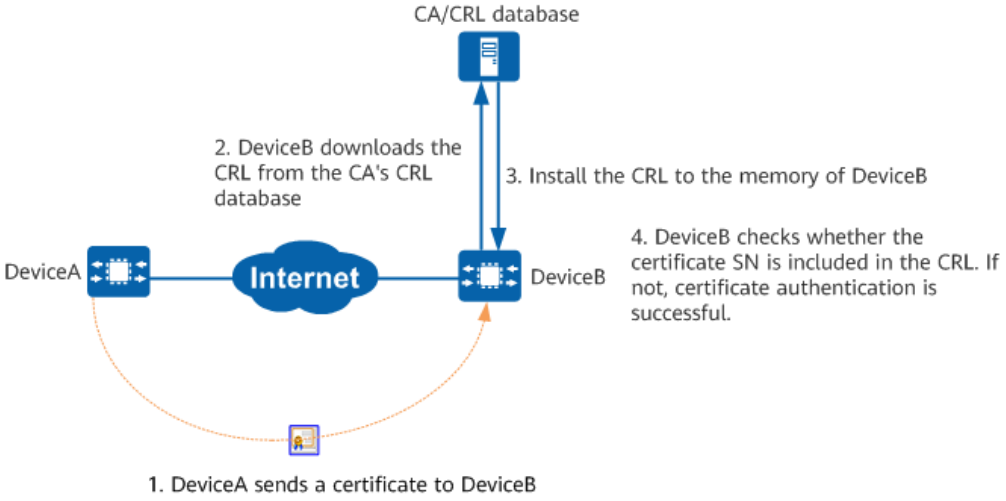
If multiple methods are configured, the system checks the certificate revocation status in the configured sequence. The latter method is used only when the current method is unavailable (for example, the CRL is missing). If CRL and None methods are configured in sequence and the former one is unavailable, the certificate is considered valid.

You can select a method for checking the certificate revocation status as required.

- CRL

The status of a certificate is determined by checking whether the certificate is included in the CRL that is saved in the CRL database. After a certificate is revoked, the SN of the certificate is recorded in the CRL. When the PKI entity authenticates the local certificate, the PKI entity searches for the certificate in the CRL stored in local memory. If the certificate is included in the CRL, it indicates that the certificate has been revoked. If no CRL is available in local memory, the CRL needs to be downloaded and installed.

Figure 11-33 Authenticating a certificate using the CRL



NOTE

A PKI entity can download a CRL using an LDAPv3 template. A PKI entity must frequently download the CRL to keep it up to date. By default, the device allocates about 5 KB memory space for processing and caching the CRL. If the reserved memory space is insufficient, new certificate revocation data cannot be imported. If you want to import new certificate revocation data, delete the old data first.

- None

  If no CRL server is available to the PKI entity or the PKI entity does not need to check the local certificate status, this mode can be used. In this mode, the PKI entity does not check the certificate revocation status.

For details about configuration parameters, see huawei-pki.yang.

Procedure

Step 1 Access the configuration mode.

```
edit-config
```

Step 2 Go to the PKI directory.

```
pki
```

Step 3 Access the list of realms to be configured.

```
domains
```

Step 4 Create a PKI realm and enter its view, or enter the view of an existing PKI realm.

```
domain name domain-name
```

By default, a PKI realm named **default** exists in the system. This realm cannot be modified or deleted.

Step 5 Configure the method of checking the certificate revocation status in a PKI realm.
validate-method { **crl-none** | **crl** | **none** }

By default, the method of checking certificate revocation status is not configured in a PKI realm.

Step 6 To configure the method of checking certificate revocation status for more PKI realms at the same time, return to the upper-layer node, and repeat steps 4 and 5.
quit

Step 7 Commit the configuration.
commit

Step 8 Exit the configuration mode and return to the top directory.
return

Step 9 Select a method to check peer certificate status according to the service types provided by the CA:

- Manual CRL update
  a. Go to the RPC node for importing a CRL.
  crl-import
  b. Access the list of CRLs to be imported.
  crls
  c. Configure the CRL file name.
  crl file-name *file-name*
  d. Configure the PKI realm to which the CRL is to be imported.
  domain-name *domain-name*
  e. **Optional:** To import multiple CRLs at the same time, return to the upper-layer node, and repeat steps d to e.
  quit
  f. Commit the configuration.
  emit

----End

Verifying the Configuration

- In the **pki** directory, run the **display global/certificate-check/validate-method** command to view the global certificate revocation status check mode.
- In the **pki** directory, run the **display domains/domain[name=***domain-name***]/validate-method** command to view the certificate revocation status check mode in a specified PKI realm.
- In the **pki** directory, run the **display crl-infos** command to check information about all CRLs on the device.
- In the **pki** directory, run the **display crl-infos/crl-info[name=***name***][domain-name=***domain-name***]** command to check information about the CRL with the specified PKI realm and file name on the device.

Follow-up Procedure

If a CRL expires or is not used, run the following commands to delete the CRL from the memory by PKI realm:

1. Access the RPC node for deleting a CRL by realm.
   crl-delete-by-domain
2. Access the list of CRLs to be deleted.
   crls
3. Configure the PKI realm of the CRL to be deleted.
   crl domain-name *domain-name*
4. **Optional:** To delete CRLs from more PKI realms at the same time, return to the upper-layer node, and repeat step 3.
   quit
5. Commit the configuration.
   emit

11.8.6.6 Checking the Validity of Certificates

Prerequisites

The CA certificate and local certificate have been installed on the device, and the certificate revocation status check mode has been configured.

Context

Each certificate installed on the local device must be authenticated to ensure validity before it is used. Certificate authentication verifies the issuing time, issuer information, and certificate validity. The main point of this is to check the CA signature on the certificate and ensure that the certificate is valid and not revoked.

To complete certificate authentication, the local device needs the following information: CA certificate, CRL, local certificate and its private key, and certificate authentication configuration.

The local device authenticates a local certificate as follows:

1. Uses the public key of the CA certificate to authenticate its signature.

   To authenticate a certificate, a PKI entity must obtain the public key of the CA that issued the certificate from the CA's certificate, so that the PKI entity can check the signature of the CA on the certificate. An upper-level CA authenticates the certificates of lower-level CAs. The authentication is performed along the certificate chain, and terminated at the trustpoint (the root CA holding a self-signed certificate or a subordinate CA trusted by the PKI entity).

   PKI entities sharing the same root or subordinate CA and having CA certificates can authenticate certificates of each other (peer certificates).

   In short, certificate chain authentication starts at the target certificate (PKI entity's certificate to be authenticated) and ends at a trustpoint. Authentication of a peer certificate chain ends at the first trusted certificate or CA.
2. Checks whether the certificate has expired.
3. Checks whether the certificate is valid based on the certificate revocation status. You can use either of the following methods to check the certificate revocation status:
   - CRL: The CRL imported to the memory is used for check.

- None: In this mode, the system does not check whether the certificate is revoked or whether the certificate chain is complete.

You are advised to use CRL to check the certificate revocation status.

For details about configuration parameters, see huawei-pki.yang.

Procedure

Step 1 Go to the RPC node for verifying a certificate.

cert-validate-by-domain

Step 2 Configure the PKI realm for which the certificate needs to be verified.

domain-name *domain-name*

Step 3 Commit the configuration.

emit

----End

11.8.6.7 Example for Applying for a Local Certificate for a PKI Entity in Offline Mode

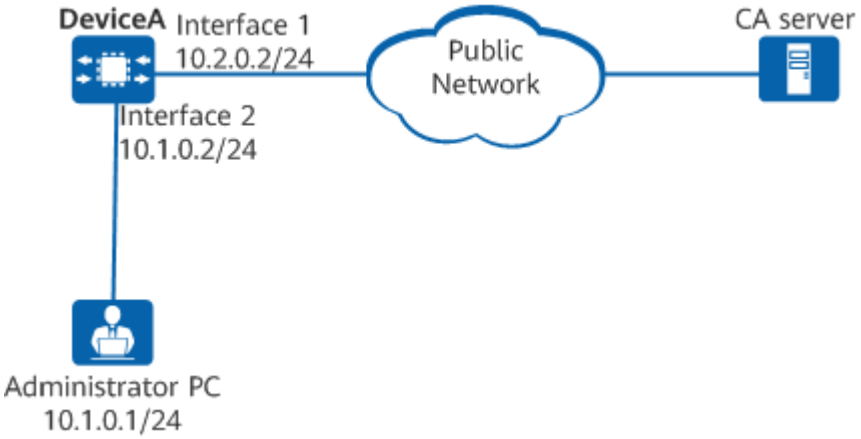
Network Requirements

As shown in **Figure 11-34**, the device applies for a local certificate from the CA server on the public network in offline mode.

NOTE

In this example, interface 1 and interface 2 represent GE1/0/1 and GE1/0/2, respectively.

Figure 11-34 Applying for a local certificate for a PKI entity in offline mode



Configuration Roadmap

The configuration roadmap is as follows:

1. Create an RSA key pair so that the local certificate application request contains the public key.
2. Configure a PKI entity and its related information to identify the PKI entity.

3. Configure local certificate application for the PKI entity in offline mode and generate a local certificate request file.
4. Send the local certificate request file in out-of-band mode and download the local certificate.
5. Install the local certificate so that the device can protect communication data.

Procedure

Step 1 Configure IP addresses for interfaces.

```
[ADMIN@HUAWEI]
MDCLI> edit-config
[(gl)ADMIN@HUAWEI]
MDCLI> ifm interfaces interface name GE0/0/0
[*(gl)ADMIN@HUAWEI]/ifm/interfaces/interface[name="GE0/0/0"]
MDCLI> ipv4 addresses address ip 10.2.0.2
[*(gl)ADMIN@HUAWEI]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="10.2.0.2"]
MDCLI> mask 255.255.255.0 type main
[*(gl)ADMIN@HUAWEI]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="10.2.0.2"]
MDCLI> commit
[(gl)ADMIN@HUAWEI]/ifm/interfaces/interface[name="GE0/0/0"]/ipv4/addresses/address[ip="10.2.0.2"]
MDCLI> return
[ADMIN@HUAWEI]
```

Step 2 Create an RSA key pair.

Create a 3072-bit RSA key pair named **rsakey**.

```
[ADMIN@HUAWEI]
MDCLI> key-pair-create
[(x)ADMIN@HUAWEI]/key-pair-create
MDCLI> key-pairs key-pair name rsakey type rsa
[*(x)ADMIN@HUAWEI]/key-pair-create/key-pairs/key-pair[name="rsakey"][type="rsa"]
MDCLI> key-size 3072
[*(x)ADMIN@HUAWEI]/key-pair-create/key-pairs/key-pair[name="rsakey"][type="rsa"]
MDCLI> emit
[ADMIN@HUAWEI]
```

Step 3 Configure a PKI entity to identify a certificate applicant.

Configure a PKI entity named **user01**.

```
[ADMIN@HUAWEI]
MDCLI> edit-config
[(gl)ADMIN@HUAWEI]
MDCLI> pki entitys entity name user01
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> common-name hello
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> country cn
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> email user@test.abc.com
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> fqdn test.abc.com
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> ip-address 10.2.0.2
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> state jiangsu
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> organization huawei
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> department info
[*(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> commit
[(gl)ADMIN@HUAWEI]/pki/entitys/entity[name="user01"]
MDCLI> return
[ADMIN@HUAWEI]
```


Step 4 Apply for a local certificate for the PKI entity in offline mode.

```
[ADMIN@HUAWEI]
MDCLI> edit-config
[(gl)ADMIN@HUAWEI]
MDCLI> pki domains domain name abc
[*(gl)ADMIN@HUAWEI]/pki/domains/domain[name="abc"]
MDCLI> entity user01
[*(gl)ADMIN@HUAWEI]/pki/domains/domain[name="abc"]
MDCLI> key-pair
[*(gl)ADMIN@HUAWEI]/pki/domains/domain[name="abc"]/key-pair
MDCLI> name rsakey type rsa
[*(gl)ADMIN@HUAWEI]/pki/domains/domain[name="abc"]/key-pair
MDCLI> commit
[(gl)ADMIN@HUAWEI]/pki/domains/domain[name="abc"]/key-pair
MDCLI> return
[ADMIN@HUAWEI]
MDCLI> csr-generate
[(x)ADMIN@HUAWEI]/csr-generate
MDCLI> csrs csr domain-name abc
[*(x)ADMIN@HUAWEI]/csr-generate/csrs/csr[domain-name="abc"]
MDCLI> file-name cer_req
[*(x)ADMIN@HUAWEI]/csr-generate/csrs/csr[domain-name="abc"]
MDCLI> emit
[ADMIN@HUAWEI]
```

Step 5 Transfer the certificate enrollment request file to the CA server in out-of-band mode (for example, by disk or email) to apply for a local certificate.

When the local certificate is successfully registered, download the local certificate **abc_local.cer** also in out-of-band mode. After the download is complete, you can import the file to the **flash:** directory of the device using a file transfer protocol.

Step 6 Install the local certificate.

After the certificate is imported successfully, the **abc_local.cer** file in the **flash:** directory is deleted by default.

```
[ADMIN@HUAWEI]
MDCLI> certificate-import
[(x)ADMIN@HUAWEI]/certificate-import
MDCLI> certificates certificate name abc_local.cer type local
[*(x)ADMIN@HUAWEI]/certificate-import/certificates/certificate[name="abc_local.cer"][type="local"]
MDCLI> domain-name abc
[*(x)ADMIN@HUAWEI]/certificate-import/certificates/certificate[name="abc_local.cer"][type="local"]
MDCLI> format pem
[*(x)ADMIN@HUAWEI]/certificate-import/certificates/certificate[name="abc_local.cer"][type="local"]
MDCLI> emit
[ADMIN@HUAWEI]
```

----End

Verifying the Configuration

After the local certificate is installed, the devices at both ends can use it to protect communication data.

11.8.7 Maintaining PKI

11.8.7.1 Deleting Certificates

Context

When a local certificate expires or a new certificate is required, delete the existing local certificate from the device memory.

For details about configuration parameters, see huawei-pki.yang.

Procedure

- Deleting a certificate by PKI realm
  a. Go to the RPC node for deleting a certificate by PKI realm.
     certificate-delete-by-domain
  b. Access the list of certificates to be deleted.
     domain-certificates
  c. Configure the PKI realm and type of the certificate to be deleted.
     domain-certificate domain-name *domain-name* **type** { **ca** | **local** }
  d. **Optional:** To delete certificates in more PKI realms at the same time, return to the upper-layer node, and repeat step 3.
     quit
  e. Commit the configuration.
     emit
- Deleting a specified certificate by file name
  a. Go to the RPC node for deleting a certificate by file name.
     certificate-delete
  b. Access the list of certificates to be deleted.
     certificates
  c. Configure the name and type of the certificate to be deleted.
     certificate name *name* **type** { **ca** | **local** }
  d. **Optional:** To delete more specified certificates at the same time, return to the upper-layer node, and repeat step 3.
     quit
  e. Commit the configuration.
     emit

----End

12 QoS Configuration

12.1 ACL-based Simplified Traffic Policy Configuration

12.1.1 Overview of an ACL-based Simplified Traffic Policy

Definition

An ACL-based simplified traffic policy enables the device to match packet information with ACL rules and provide the same QoS service for packets that match the same ACL rules, implementing differentiated services.

Purpose

To control traffic entering a network, you can configure ACL rules to match packets based on information such as the source IP address, fragment flag, destination IP address, source port number, and source MAC address in packets. You can then configure an ACL-based simplified traffic policy so that the device can filter, redirect, re-mark, or collect statistics on packets that match such ACL rules.

Compared with a common traffic policy, an ACL-based simplified traffic policy is easier to configure because no traffic classifier, traffic behavior, or traffic policy needs to be created independently. However, an ACL-based simplified traffic policy defines fewer matching rules than a common traffic policy because only ACL rules are used to match packets.

NOTE

- Currently, a simplified traffic policy can be applied to an interface, but not globally or in a VLAN.
- If **acl** *acl-name* is specified in a command for configuring an ACL-based simplified traffic policy, the command can be configured successfully only when the corresponding ACL has been created using the **acl** command. Otherwise, the command fails to be configured.
- If the ACL rule referenced by an ACL-based simplified traffic policy configured on an interface changes, the ACL-based simplified traffic policy temporarily becomes invalid.
- After a simplified traffic policy is configured, the configuration takes effect during traffic processing after a certain delay.
- Only one ACL instance can be configured for an ACL-based simplified traffic policy in a single direction.

12.1.2 Configuration Precautions for ACL-based Simplified Traffic Policy

Licensing Requirements

ACL-based Simplified Traffic Policy is not under license control.

Hardware Requirements

Table 12-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

12.1.3 Default Settings for an ACL-based Simplified Traffic Policy

Table 12-2 describes the default settings for an ACL-based simplified traffic policy.

Table 12-2 Default settings for an ACL-based simplified traffic policy

Parameter	Default Setting
Traffic statistics collection function	Disabled

12.1.4 Configuring ACL-based Packet Filtering

Context

You can configure ACL-based packet filtering to permit or deny packets matching ACL rules, thereby controlling network traffic.

The **traffic-filter** command is used to configure packet filtering.

- If the ACL associated with the **traffic-filter** command is also associated with another ACL-based simplified traffic policy:
  - When both the **traffic-filter** command and the ACL-based simplified traffic policy are configured and the deny action is configured in the ACL rule, if packets match the ACL rule and are filtered based on the **traffic-filter** command configuration, the traffic-remark and traffic-redirect policies are not performed on the packets.
  - When both the **traffic-filter** command and the ACL-based simplified traffic policy are configured and the permit action is configured in the ACL rule, if packets match the ACL rule and are permitted, the traffic-remark and traffic-redirect policies are performed on the packets.

For details about configuration parameters, see huawei-sacl.yang.

NOTE

When the **traffic-redirect** or **traffic-remark** command is configured, the device performs the corresponding action on packets based on the configured simplified traffic policy regardless of whether the packets match the permit or deny rule in the ACL. To configure the device to filter packets that match the deny rule, you must run the **traffic-filter** command.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Configure a packet filtering policy in the inbound or outbound direction of the interface.

```
traffic-filter-applys traffic-filter-apply direction { inbound | outbound }
```

Step 4 Create an ACL instance.

```
acl-instances acl-instance acl acl-name
```

Step 5 (Optional) Enable or disable the traffic statistics collection function.

```
enable-statistic { true | false }
```

By default, the traffic statistics collection function is disabled.

Step 6 Commit the configuration.

```
commit
```

----End

12.1.5 Configuring ACL-based Redirection

Context

You can configure ACL-based redirection to redirect packets matching ACL rules to the CPU, a specified interface, or a specified next-hop address. For details about configuration parameters, see huawei-sacl.yang.

NOTE

- In the current version, the device supports only redirection to a next-hop IP address, and the IP address cannot be a multicast address.
- The **traffic-redirect** command can be applied only in the inbound direction of an interface.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Configure a redirection policy in the inbound direction of the interface.

```
traffic-redirect-applys traffic-redirect-apply direction inbound
```

Step 4 Create an ACL instance.

```
acl-instances acl-instance acl acl-name
```

Step 5 Configure the next-hop IP address to which packets are redirected.

```
ip-nexthop ip-address
```

Step 6 Commit the configuration.

```
commit
```

----End

12.1.6 Configuring ACL-based Re-marking

Context

You can configure ACL-based re-marking to re-mark packets matching a specified ACL rule, for example, re-marking the DSCP priority. For details about configuration parameters, see huawei-sacl.yang.

Procedure

Step 1 Enter the edit-config view.

```
edit-config
```

Step 2 Enter the interface view.

```
ifm interfaces interface name interface-name
```

Step 3 Configure a re-marking policy in the inbound or outbound direction of the interface.

traffic-remark-applys traffic-remark-apply direction { **inbound** | **outbound** }

Step 4 Create an ACL instance.

acl-instances acl-instance acl *acl-name*

Step 5 Configure the DSCP value for re-marking the DSCP priority.

dscp-value *value*

Step 6 Commit the configuration.

commit

----End

12.1.7 Verifying the Configuration

Procedure

Operation	Command
Check the simplified traffic policy configured on an interface.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Check the simplified traffic policy configured on the interface. **display this**
Check the traffic-filter policy configured on an interface.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Check the traffic-filter policy configured on the interface. **display traffic-filter-applys/**
Check the traffic-redirect policy configured on an interface.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Check all traffic-redirect policies configured on the interface. **display traffic-redirect-applys**
Check the traffic-remark policy configured on an interface.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Check the traffic-remark policy configured on the interface. **display traffic-remark-applys**

12.1.8 Maintaining an ACL-based Simplified Traffic Policy

Context

- After ACL-based packet filtering is configured on the device, you can view statistics on forwarded and discarded packets.
- To check whether an ACL is applied successfully, you can view the status of the ACL applied to an ACL-based simplified traffic policy.
- To re-collect traffic statistics on ACL-based packet filtering, you can clear existing statistics.

For details about configuration parameters, see huawei-sacl.yang.

NOTICE

Traffic statistics on ACL-based packet filtering cannot be restored after they are cleared. Exercise caution when clearing them.

Procedure

The following table describes the operations for maintaining an ACL-based simplified traffic policy.

Table 12-3 Operations for maintaining an ACL-based simplified traffic policy

Operation	Command
Check traffic statistics on ACL-based packet filtering.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Check traffic statistics on ACL-based packet filtering. **display traffic-filter-applys/traffic-filter-apply[direction={inbound|outbound}]/acl-instances/acl-instance[acl=***acl_name***] all**

Operation	Command
Check the status of the ACL applied to an ACL-based simplified traffic policy.	1. Enter the edit-config view. **edit-config** 2. Enter the interface view. **ifm interfaces interface name** *interface-name* 3. Check the status of the ACL applied to a traffic-filter policy. **display traffic-filter-applys/traffic-filter-apply[direction={inbound|outbound}]/acl-instances/acl-instance[acl=***acl_name***] all** 4. Check the status of the ACL applied to a traffic-redirect policy. **display traffic-redirect-applys/traffic-redirect-apply[direction=inbound]/acl-instances/acl-instance[acl=***acl_name***] all** 5. Check the status of the ACL applied to a traffic-remark policy. **display traffic-remark-applys/traffic-remark-apply[direction={inbound|outbound}]/acl-instances/acl-instance[acl=***acl_name***] all**
Clear traffic statistics on ACL-based packet filtering.	1. Enter the view of clearing statistics. **clear-traffic-apply-statistic** 2. Configure the function of clearing traffic statistics on ACL-based packet filtering. **acl** *acl_name* **application-type filter direction {inbound|outbound} rule** *rule_name* **interface-name** *interface-name* 3. Perform the clear operation. **emit**

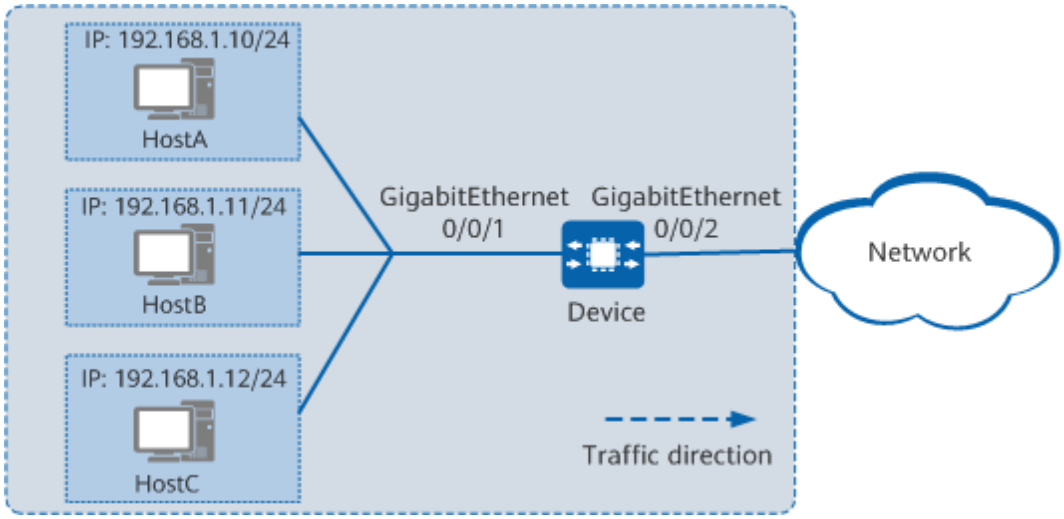
12.1.9 Configuration Examples for an ACL-based Simplified Traffic Policy

12.1.9.1 Example for Denying Packets from a Specified Host

Networking Requirements

During working hours from 08:00 to 18:00 every day, GE0/0/1 filters incoming packets and prevents the host at 192.168.1.10/24 from accessing the external network.

Figure 12-1 Networking diagram of preventing a specified host from accessing the network



Configuration Roadmap

You can use a traffic policy containing the deny action to filter packets. The configuration roadmap is as follows:

1. Configure the time range to be referenced in an ACL.
2. Configure an ACL rule to deny packets during working hours.
3. Configure packet filtering in the inbound direction of GE0/0/1.

Procedure

Step 1 Create the time range that defines working hours.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> time-range time-range-instances time-range-instance name work_time

[*(gl)user@localhost]/time-range/time-range-instances/time-range-instance[name="work_time"]
MDCLI> period-ranges period-range start-time 08:00 end-time 18:00 day-of-week "monday tuesday wednesday thursday friday"

[*(gl)user@localhost]/time-range/time-range-instances/time-range-instance[name="work_time"]/period-ranges/period-range[day-of-week="monday tuesday wednesday thursday friday"][start-time="08:00"][end-time="18:00"]
MDCLI> commit
```

Step 2 Create an ACL rule named **net-deny** that denies network access during working hours.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> acl groups group identity net-deny

[*(gl)user@localhost]/acl/groups/group[identity="net-deny"]
MDCLI> type basic

[*(gl)user@localhost]/acl/groups/group[identity="net-deny"]
MDCLI> rule-basics rule-basic name deny-rule
```



```
[*(gl)user@localhost]/acl/groups/group[identity="net-deny"]/rule-basics/rule-basic[name="deny-rule"]
MDCLI> time-range-name work_time

[*(gl)user@localhost]/acl/groups/group[identity="net-deny"]/rule-basics/rule-basic[name="deny-rule"]
MDCLI> source-ipaddr 192.168.1.10 source-wild 0.0.0.255

[*(gl)user@localhost]/acl/groups/group[identity="net-deny"]/rule-basics/rule-basic[name="deny-rule"]
MDCLI> action deny

[*(gl)user@localhost]/acl/groups/group[identity="net-deny"]/rule-basics/rule-basic[name="deny-rule"]
MDCLI> commit
```

Step 3 Configure ACL-based packet filtering in the inbound direction of GE0/0/1.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> traffic-filter-applys traffic-filter-apply direction inbound

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-filter-applys/traffic-filter-
apply[direction="inbound"]
MDCLI> acl-instances acl-instance acl net-deny

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-filter-applys/traffic-filter-
apply[direction="inbound"]/acl-instances/acl-instance[acl="net-deny"]
MDCLI> enable-statistic true

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-filter-applys/traffic-filter-
apply[direction="inbound"]/acl-instances/acl-instance[acl="net-deny"]
MDCLI> commit
```

Step 4 Verify the configuration.

Display the time range configured on the device.

```
[user@localhost]
MDCLI> time-range

[user@localhost]/time-range
MDCLI> display this
{
  "time-range-instances": {
    "time-range-instance": [
      {
        "name": "work_time",
        "period-ranges": {
          "period-range": [
            {
              "day-of-week": "monday tuesday wednesday thursday friday",
              "start-time": "08:00",
              "end-time": "18:00"
            }
          ]
        }
      }
    ]
  }
}
```

Display the ACL rule configured on the device.

```
[user@localhost]
MDCLI> acl

[user@localhost]/acl
MDCLI> display groups/
{
```



```
 "group": [
  {
   "identity": "net-deny",
   "type": "basic",
   "rule-basics": {
    "rule-basic": [
     {
      "name": "deny-rule",
      "action": "deny",
      "source-ipaddr": "192.168.1.10",
      "source-wild": "0.0.0.255",
      "time-range-name": "work_time"
     }
    ]
   }
  }
 ]
}
```


Display the ACL rule applied to the inbound direction of an interface and the corresponding statistics.


```
[user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1

[user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> display traffic-filter-applys/traffic-filter-apply[direction=inbound]/acl-instances/acl-instance[acl=net-deny] all
{
 "acl": "net-deny",
 "ipv6": false,
 "enable-statistic": true,
 "statistics": {
  "statistic": [
   {
    "rule": "deny-rule",
    "slot": "slot0",
    "apply-status": "success",
    "match-permit-packet": 0,
    "match-permit-byte": 0,
    "match-discarded-packet": 0,
    "match-discarded-byte": 0
   }
  ]
 }
}
```


----End

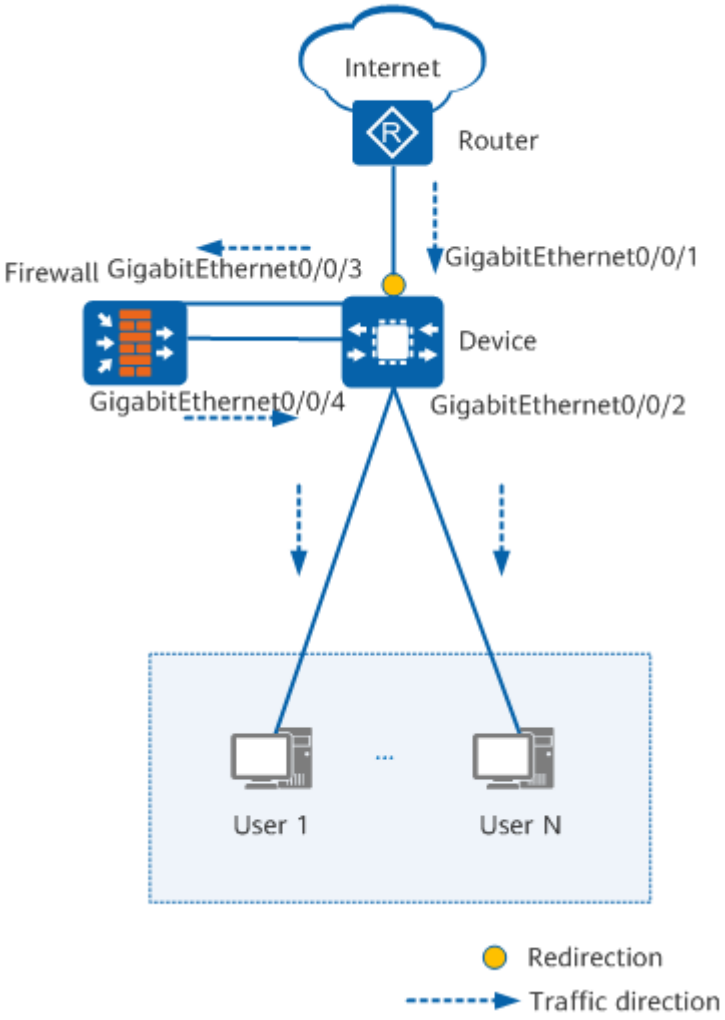
12.1.9.2 Example for Configuring ACL-based Redirection

Networking Requirements

Users are connected to Device and need to access the Internet.

For data and network security purposes, users want to ensure security of all traffic from the Internet to the internal network. Redirection can be configured so that all traffic from the external network to the internal network is sent to the firewall for security filtering.

Figure 12-2 Networking diagram of redirecting packets from the external network to the firewall for security filtering



Configuration Roadmap

1. Configure an ACL rule to match all traffic.
2. Configure the IP address 192.168.7.10 for GE0/0/3 that is connected to the firewall.
3. Configure a redirection policy on GE0/0/1 that receives traffic from the external network.
4. Configure redirection to the next-hop IP address 192.168.7.10 so that traffic is sent to the firewall for security filtering.

Procedure

Step 1 Configure ACL-based redirection to implement traffic filtering.

Configure a basic ACL to match all packets.

```
[user@localhost]
MDCLI> edit-config
```



```
[(gl)user@localhost]
MDCLI> acl groups group identity sec_redirct

[*(gl)user@localhost]/acl/groups/group[identity="sec_redirct"]
MDCLI> type basic

[*(gl)user@localhost]/acl/groups/group[identity="sec_redirct"]
MDCLI> rule-basics rule-basic name sec_rule

[*(gl)user@localhost]/acl/groups/group[identity="sec_redirct"]/rule-basics/rule-basic[name="sec_rule"]
MDCLI> action permit

[*(gl)user@localhost]/acl/groups/group[identity="sec_redirct"]/rule-basics/rule-basic[name="sec_rule"]
MDCLI> commit
```

Configure the IP address 192.168.7.10 for GE0/0/3.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> ifm interfaces interface name GE0/0/3

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/3"]
MDCLI> ipv4 addresses address ip 192.168.7.10

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/3"]/ipv4/addresses/address[ip="192.168.7.10"]
MDCLI> mask 255.255.255.0

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/3"]/ipv4/addresses/address[ip="192.168.7.10"]
MDCLI> type main

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/3"]/ipv4/addresses/address[ip="192.168.7.10"]
MDCLI> commit
```

Configure redirection to a specified next-hop address in the inbound direction of GE0/0/1.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> traffic-redirect-applys traffic-redirect-apply direction inbound

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-redirect-applys/traffic-redirect-apply[direction="inbound"]
MDCLI> acl-instances acl-instance acl sec_redirct

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-redirect-applys/traffic-redirect-apply[direction="inbound"]/acl-instances/acl-instance[acl="sec_redirct"]
MDCLI> ip-nexthop 192.168.7.10

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-redirect-applys/traffic-redirect-apply[direction="inbound"]/acl-instances/acl-instance[acl="sec_redirct"]
MDCLI> commit
```

Step 2 Verify the configuration.

Display the ACL rule configured on the device.

```
[user@localhost]
MDCLI> acl

[user@localhost]/acl
MDCLI> display groups/
{
  "group": [
    {
```



```
    "identity": "sec_redirct",
    "type": "basic",
    "rule-basics": {
      "rule-basic": [
        {
          "name": "sec_rule",
          "action": "permit"
        }
      ]
    }
  }
 ]
}
```

Display the ACL rule and traffic behavior applied to the inbound direction of an interface.

```
[user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1

[user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> display traffic-redirect-applys/traffic-redirect-apply[direction=inbound]/acl-instances/acl-instance[acl=sec_redirct] all
{
  "acl": "sec_redirct",
  "ipv6": false,
  "ip-nexthop": "192.168.7.10",
  "statuses": {
    "status": [
      {
        "rule": "sec_rule",
        "slot": "slot0",
        "apply-status": "success"
      }
    ]
  }
}
```

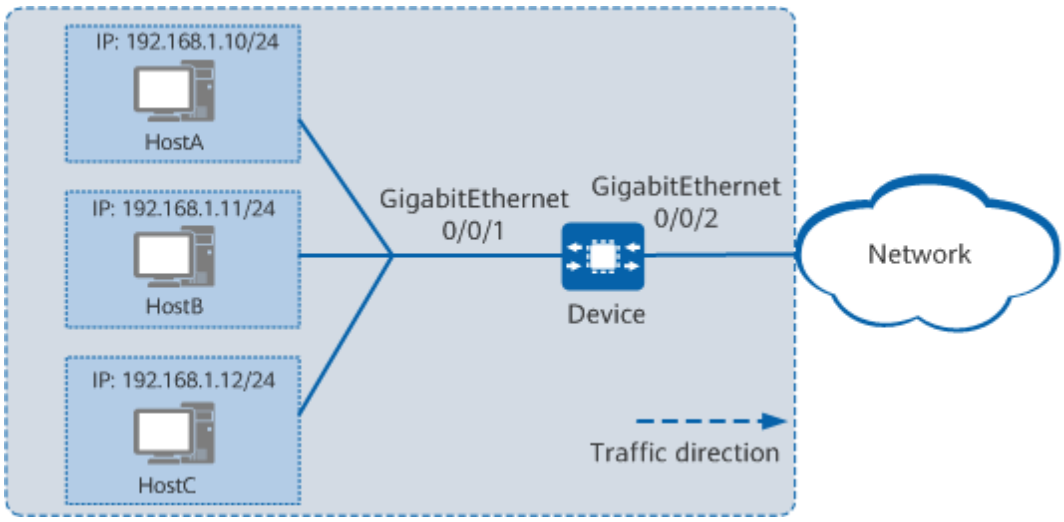
----End

12.1.9.3 Example for Configuring an ACL-based Simplified Traffic Policy to Implement Priority Mapping

Networking Requirements

Data packets from different users enter the device through GE0/0/1. It is required that differentiated services be provided for ICMP packets from different users based on DSCP priorities.

Figure 12-3 Networking diagram of configuring differentiated services for ICMP packets from different users



Configuration Roadmap

The configuration roadmap is as follows:

1. Configure an ACL rule to specify the source user network segment for which differentiated services are provided.
2. Configure priority mapping in the inbound direction of GE0/0/1 and re-mark the priority to 4.

Procedure

Step 1 Configure priority mapping.

Configure an ACL to identify ICMP packets from the source user network segment for which differentiated services are provided.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> discard

[(gl)user@localhost]
MDCLI> acl groups group identity icmp_remark

[*(gl)user@localhost]/acl/groups/group[identity="icmp_remark"]
MDCLI> type advance

[*(gl)user@localhost]/acl/groups/group[identity="icmp_remark"]
MDCLI> rule-advances rule-advance name user_rule

[*(gl)user@localhost]/acl/groups/group[identity="icmp_remark"]/rule-advances/rule-
advance[name="remark_rule"]
MDCLI> protocol 1 action deny

[*(gl)user@localhost]/acl/groups/group[identity="icmp_remark"]/rule-advances/rule-
advance[name="remark_rule"]
MDCLI> source-ipaddr 192.168.1.10 source-wild 0.0.0.255

[*(gl)user@localhost]/acl/groups/group[identity="icmp_remark"]/rule-advances/rule-
advance[name="remark_rule"]
MDCLI> commit
```


Step 2 Configure priority mapping in the inbound direction of GE0/0/1.

```
[user@localhost]
MDCLI> edit-config

[(gl)user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> traffic-remark-applys traffic-remark-apply direction inbound

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-remark-applys/traffic-remark-apply[direction="inbound"]
MDCLI> acl-instances acl-instance acl icmp_remark

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-remark-applys/traffic-remark-apply[direction="inbound"]/acl-instances/acl-instance[acl="icmp_remark"]
MDCLI> dscp-value 4

[*(gl)user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]/traffic-remark-applys/traffic-remark-apply[direction="inbound"]/acl-instances/acl-instance[acl="icmp_remark"]
MDCLI> commit
```

Step 3 Verify the configuration.

Display the ACL rule configured on the device.

```
[user@localhost]
MDCLI> acl

[user@localhost]/acl
MDCLI> display groups/
{
  "group": [
    {
      "identity": "icmp_remark",
      "type": "advance",
      "rule-advances": {
        "rule-advance": [
          {
            "name": "remark_rule",
            "action": "deny",
            "protocol": 1,
            "source-ipaddr": "192.168.1.10",
            "source-wild": "0.0.0.255"

          }
        ]
      }
    }
  ]
}
```

Display the ACL rule and traffic behavior applied to the inbound direction of an interface.

```
[user@localhost]
MDCLI> ifm interfaces interface name GE0/0/1

[user@localhost]/ifm/interfaces/interface[name="GE0/0/1"]
MDCLI> display traffic-remark-applys/traffic-remark-apply[direction=inbound]/acl-instances/acl-instance[acl=icmp_remark] all
{
  "acl": "icmp_remark",
  "ipv6": false,
  "dscp-value": 4,
  "statuses": {
    "status": [
      {
        "rule": "remark_rule",
        "slot": "slot0",
        "apply-status": "success"
```



```
    }
   ]
  }
}
```

----End

13 System Monitoring Configuration

13.1 Packet Capture Configuration

NOTE

The packet capture function aims at locating the faults and errors in data communication. Some personal information may need to be collected and stored. Huawei is unable to collect or store user communication information without permission. You are responsible for complying with applicable laws and regulations when enabling related functions used to collect or store user communication information. During user communication information collection and storage, proper measures must be taken to protect user communication information.

13.1.1 Overview of Packet Capture

As the Internet develops, devices need to transmit various services, and maintenance personnel often have to obtain packets to locate fault causes. The packet capture function enables a device to obtain received packets and provides an efficient approach for locating faults on a network without the need to deploy packet analysis devices and network monitoring devices.

After the packet capture function is enabled, the device obtains packets that match the specified filter rules. The maintenance personnel can save the packets to a *.cap file on the local storage media, and download the file to a local PC for fault analysis. This greatly improves maintenance efficiency and reduces maintenance costs.

13.1.2 Configuration Precautions for Packet capture

Licensing Requirements

Packet Capture is not under license control.

Hardware Requirements

Table 13-1 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

13.1.3 Configuring a Device to Obtain Packets

Context

If traffic on a device is abnormal, you can configure the device to obtain service packets for analysis. In this way, you can process packets promptly to ensure stable running of the device.

For details about configuration parameters, see **huawei-capture.yang**.

Procedure

Step 1 Enter the packet capture view.

```
capture-packet
```

Step 2 (Optional) Adjust parameters for obtaining packets.

1. Specify whether the device obtains only the packets sent to itself.

   ```
   local-host { true | false }
   ```

   By default, a device obtains the packets sent to and forwarded by itself.
2. Specify whether to clear the payload information (packet data) of obtained packets.

   ```
   clear-payload { true | false }
   ```

   By default, the payload information of obtained packets is not cleared.
3. Configure the direction of packets to be obtained. **inbound** indicates that the device obtains incoming packets. **outbound** indicates that the device obtains outgoing packets. **both** indicates that the device obtains both incoming and outgoing packets.

   ```
   direction { both | inbound | outbound }
   ```

   By default, the device obtains both incoming and outgoing packets.
4. Configure the length of obtained packets.

   ```
   packet-length packet-length
   ```

   By default, 64-byte packets are obtained.

5. Configure the number of packets to be obtained.
   packet-number *packet-number*

   By default, the device obtains 100 packets at a time.
6. Configure the period for obtaining packets at a time.
   timeout *timeout*

   By default, packets are obtained for a period of 60 seconds at a time.
7. Obtain packets from a specified VLAN.
   vlan-id *vlan-id*

   By default, the device obtains packets regardless of VLANs.
8. Obtain packets matching the specified ACL. The ACL must have been configured. For details, see "ACL Configuration" in **Configuration Guide** > **IP Addresses and Services**.
   acl-name *capture_acl*

   By default, the device obtains packets regardless of ACLs.
9. Obtain packets on a specified interface.
   interface-name { *interface-name* } &<1-n>

   By default, the device obtains packets regardless of interfaces.
10. Configure the name of the file for storing obtained packets.
    file-name *file-name*

    By default, obtained packets are stored in the **capture_id*index*.cap** file. For example, if packet obtaining instance 1 is used, the name of the file that stores obtained packets is **capture_id1.cap**.

    NOTE

    Only one file is generated for each packet obtaining instance. The file is stored in the **/opt/vrpv8/home** directory. The user-defined file name must end with **.cap**.

Step 3 Enable the device to obtain packets.
emit

----End

Example

Obtain all packets received by GE0/0/1.

```
[user@HUAWEI]
MDCLI> capture-packet
[(x)user@HUAWEI]/capture-packet
MDCLI> interface-name GE0/0/1
[*(x)user@HUAWEI]/capture-packet
MDCLI> file-name inbound.cap
[*(x)user@HUAWEI]/capture-packet
MDCLI> direction inbound
[*(x)user@HUAWEI]/capture-packet
MDCLI> emit
{
  "huawei-capture:capture-index": 1
}
```

Obtain only the packets sent to the device itself.


```
[user@HUAWEI]
MDCLI> capture-packet
[(x)user@HUAWEI]/capture-packet
MDCLI> local-host true
[*(x)user@HUAWEI]/capture-packet
MDCLI> emit
{
  "huawei-capture:capture-index": 2
}
```

13.1.4 Configuring a Device to Stop Obtaining Packets

Context

During packet obtaining, you can manually configure a device to stop a specified packet obtaining task or all packet obtaining tasks.

Procedure

Step 1 Enter the view of stopping packet obtaining.

```
stop-capture
```

Step 2 Specify the packet obtaining task to be stopped as required.

- Stop a specified packet obtaining task.
  ```
  capture-index index
  ```
- Stop all packet obtaining tasks.
  ```
  all [ null ]
  ```

Step 3 Stop the action of obtaining packets.

```
emit
```

----End

13.2 Ping and Tracert Configuration

13.2.1 Overview of Ping and Tracert

Definition

Ping

Packet Internet Groper (Ping) is a typical debugging tool used to test the reachability of network devices. You can perform a ping operation on a source to send an Internet Control Message Protocol (ICMP) Echo message to a destination. After receiving the message, the destination returns an ICMP Echo Reply message to the source. You can test the following items using ping:

- Reachability of a remote device
- Round-trip time (RTT) in communication with a remote device
- ICMP Echo message loss

Tracert

Tracert tests reachability of each hop on the path from a source to a destination. You can perform a tracert operation on a source to send UDP packets with

different TTL values to a destination. During packet forwarding, the TTL value of a packet is decremented by one each time the packet reaches a hop. If a hop receives a packet with the TTL value of 0, this hop sends an ICMP Time Exceeded message to the source. If the destination receives such a packet, it sends an ICMP Port Unreachable message to the source. You can test the following items using tracert:

- Reachability of each hop on the path from a source to a destination
- RTT in communication between the source and each hop

Purpose

If a network service is unavailable, the network provider needs to test reachability of network devices and locate the network fault. As most network devices support ping and tracert functions, network providers can use ping to test device reachability and use tracert to test reachability of each hop on a network path.

Benefits

The ping and tracert functions help users test network reachability and RTT, facilitating network fault locating.

13.2.2 Understanding Ping and Tracert

13.2.2.1 Ping Fundamentals

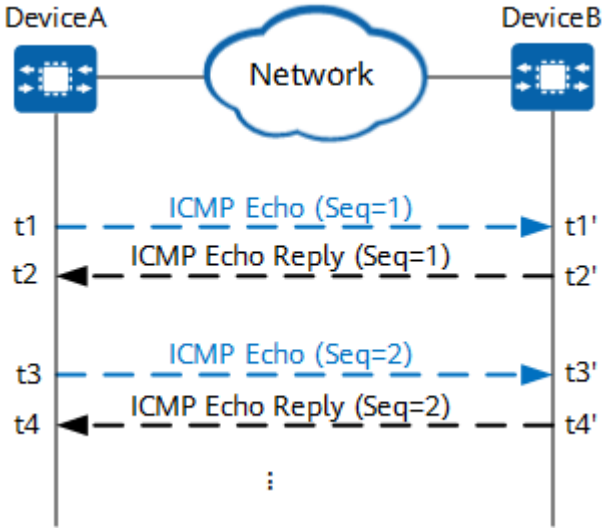
Ping is a common tool used to test network device reachability. Based on the ICMP mechanism, the ping operation enables the source to send ICMP Echo Request messages to the destination. After receiving these messages, the destination returns ICMP Echo Reply messages. The source determines the reachability of the destination based on whether the ICMP Echo Reply messages are received within the timeout interval. If the destination is reachable, the source can then determine the RTT.

Ping Process

The packet exchange during a ping operation is as follows:

1. A user performs a ping operation on the source (DeviceA), sending a batch of ICMP Echo Request messages to the specified destination. The sequence number of the ICMP Echo Request messages starts from 1 and is incremented by a step of 1, as shown in **Figure 13-1**. The number of ICMP Echo Request messages sent in a ping operation is configurable, which is 5 by default. After sending the ICMP Echo Request messages, the source starts the timeout timer and waits for the ICMP Echo Reply messages from the destination (DeviceB).
2. After receiving an ICMP Echo Request message, the destination sends an ICMP Echo Reply message with the same sequence number as the received message.
3. If the ICMP Echo Reply message reaches the source before the timer expires, the destination is considered reachable. The source can obtain the RTT by simply adding the time required for a request to be sent and that required for the corresponding response to be received.

Figure 13-1 Packet exchange during a ping operation



A request timeout message will be displayed on the source if either of the following problems occurs:

- The TTL value of an ICMP Echo Request message is decremented to 0 before the message reaches the destination. The intermediate device that receives this message sends an ICMP Time Exceeded message to the source, indicating that the destination is unreachable.
- The network between the source and destination is disconnected, or the ICMP Echo Reply messages sent by the destination do not reach the source before the timer expires.

A ping operation on the source is assigned a process ID, which is used as an identifier for ICMP Echo Request messages. As a result, the source can still successfully distinguish ICMP Echo Reply messages, even if multiple ping operations are performed concurrently.

Application Scenarios of Ping

The ping function can be used to detect the reachability and response performance of different networks. The following table describes its application scenarios.

Table 13-2 Application scenarios of ping

Network Type	Detection Item	Description
IP network	IPv4 network connectivity	Detect the connectivity between two devices on an IPv4 network.

13.2.2.2 Tracert Fundamentals

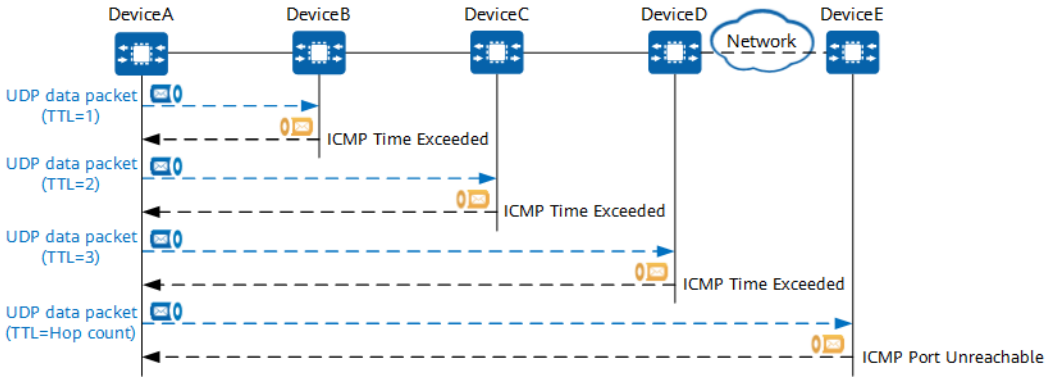
The tracert function utilizes the ICMP error-reporting mechanism. You can send UDP packets with different TTL values from the source to the destination so that the packets will reach different hops on the network path. Each hop then sends ICMP error-reporting messages to the source.

Tracert Process

The packet exchange during a tracert operation is as follows:

1. A user performs a tracert operation on the source (DeviceA) to send UDP packets (three packets by default) with the TTL value of 1 to the destination (DeviceE). The first packet uses port 33434 as the destination port by default, and the subsequent packets increase the port number by 1 in sequence. These specified ports are not the actual destination port.
2. The first hop (DeviceB) receives the UDP packets. As the TTL value in the received packets is reduced to 0, the first hop sends three ICMP Time Exceeded messages to the source.
3. After receiving the ICMP Time Exceeded messages sent by the first hop, the source determines that the first hop is reachable, and obtains the RTT (time difference between sending and receiving packets) in communication between the source and the first hop.
4. The source continues to send three UDP packets with a TTL value of 2 to the destination. Similarly, after receiving the packets with the TTL value reduced to 0, the second hop sends ICMP Time Exceeded messages to the source.
5. The source continues to send UDP packets by increasing the TTL value by 1 each time a packet is sent in order to test the reachability of each hop.
6. Finally, the destination receives the UDP packets from the source. As the destination port specified in the packets is incorrect, the destination sends three ICMP Port Unreachable messages to the source, indicating that the destination port is unreachable.
7. The source receives the ICMP Port Unreachable messages, indicating that the tracert process is completed. The command output on the source displays the reachability and RTT of each hop on the path from the source to the destination.

Figure 13-2 Packet exchange during a tracert operation



If no response packet is received within a specified period after a UDP packet is sent, a timeout prompt is displayed on the source. If a timeout prompt is displayed

after the source sends a UDP packet with the configured maximum TTL value, the destination cannot be reached and the test fails.

Application Scenarios of Tracert

The tracert function can be used to detect paths on different networks. The following table describes its application scenarios.

Table 13-3 Application scenarios of tracert

Network Type	Detection Item	Description
IP network	Path information of an IPv4 network	Detect the path between two devices on an IPv4 network.

13.2.3 Configuration Precautions for Ping

Hardware Requirements

Table 13-4 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

13.2.4 Configuration Precautions for Tracert

Hardware Requirements

Table 13-5 Hardware requirements

Series	Models
S380-H series	S380-H8T3ST
S380-L series	S380-L4P1T/S380-L4T1T
S380-S series	S380-S8P2T/S380-S8T2T

Feature Requirements

None

13.2.5 Default Settings for Ping and Tracert

The following tables describe the default settings for the ping function.

Table 13-6 Default settings for the ping operation on an IPv4 network

Parameter	Default Setting
Number of sent ICMP Echo messages	5
Length of an ICMP Echo message	64 bytes
Timeout interval for receiving an ICMP Echo Reply message	10 seconds
Interval for sending ICMP Echo messages	500 milliseconds
TTL value of an ICMP Echo message	255
ToS value of an ICMP Echo message	0

Table 13-7 Default settings for the tracert operation on an IPv4 network

Parameter	Default Setting
Initial TTL value of packets sent by the source	1
Maximum TTL value of packets sent by the source	30
UDP port number of the destination	33434
Number of probe data packets sent at a time	3
Packet length	46 bytes
Timeout interval for receiving a response packet	5 seconds

13.2.6 Using Ping and Tracert to Test an IP Network

13.2.6.1 Using Ping to Test IPv4 Network Connectivity

Context

The number of ICMP Echo Request messages sent in a ping operation is configurable. By default, which is 5 by default. The packet sequence number starts at 1. Each time an ICMP Echo Request message is sent, the packet sequence number is incremented by 1. If the destination is reachable, it sends ICMP Echo Reply messages with the same sequence number as the ICMP Echo Request messages that cause the reply.

For details about configuration parameters, see **huawei-diagnostic-tools-ipv4.yang**.

Procedure

- Configure ping to test IPv4 network connectivity.
  a. Enter the configuration path.
     ipv4-start-ip-ping
  b. Configure the test name.
     test-name *test-name*
  c. Configure the destination IP address or host name.
     dest-addr { *ip-address* | *host-name* }
  d. (Optional) Configure the interval at which packets are sent.
     interval *interval*
  e. (Optional) Configure the number of sent packets.
     packet-count *packet-count*
  f. (Optional) Configure the packet size.
     packet-size *packet-size*
  g. (Optional) Configure the IP address of the interface for sending packets.
     source-address *ipv4-address*
  h. (Optional) Configure the name of the interface for sending packets. (If the **source-address ipv4-address** command has been configured, you cannot run this command to configure the name of the interface for sending packets.)
     if-name *if-name*
  i. (Optional) Configure the TTL.
     ttl *ttl*
  j. Commit the configuration.
     emit

----End

Example

- Perform a ping operation to test IPv4 network connectivity.

```
[(x)user@localhost]
MDCLI> ipv4-start-ip-ping
[(x)user@localhost]/ipv4-start-ip-ping
MDCLI> test-name test
[*(x)user@localhost]/ipv4-start-ip-ping
MDCLI> dest-addr 127.0.0.1
[*(x)user@localhost]/ipv4-start-ip-ping
```



```
MDCLI> packet-count 5
[*(x)user@localhost]/ipv4-start-ip-ping
MDCLI> emit
  PING 127.0.0.1 (127.0.0.1): 56 data bytes
  64 bytes from 127.0.0.1: seq=0 ttl=255 time=7.270 ms
  64 bytes from 127.0.0.1: seq=1 ttl=255 time=2.069 ms
  64 bytes from 127.0.0.1: seq=2 ttl=255 time=2.823 ms
  64 bytes from 127.0.0.1: seq=3 ttl=255 time=2.770 ms
  64 bytes from 127.0.0.1: seq=4 ttl=255 time=2.856 ms

  --- 127.0.0.1 ping statistics ---
  5 packets transmitted, 5 packets received, 0% packet loss
  round-trip min/avg/max = 2.069/3.557/7.270 ms
```

The command output contains the following information:

- Response to each test packet: If no ICMP Echo Reply message is received before the timer on the source expires, the message "Request time out" is displayed. If an ICMP Echo Reply message is received, information such as the number of bytes in the payload, packet sequence number, TTL, and response time is displayed.
- Ping statistics: include the numbers of sent and received packets, the packet loss rate, and the minimum, maximum, and average response time durations.

13.2.6.2 Using Tracert to Test a Network Path on an IPv4 Network

Context

The tracert function is used to test the reachability of each hop on the path from the source to the destination.

You can perform a tracert operation on the source by sending three UDP packets at a time. The TTL value of the first batch of UDP packets is 1, and the TTL value of each further batch is incremented by 1.

If the source receives response packets within a specified period after sending packets, the address and RTT of the response node are displayed on the source. If no response packet is received, a response timeout prompt is displayed on the source. If a response timeout prompt is displayed after the source sends a UDP packet with the maximum TTL value, the destination cannot be reached and the test fails.

For details about configuration parameters, see **huawei-diagnostic-tools-ipv4.yang**.

Procedure

- Configure tracert to test an IPv4 network.
  a. Enter the configuration path.
     ipv4-start-ip-trace
  b. Configure the test name.
     test-name *test-name*
  c. Configure the destination IP address or host name.
     dest-addr { *ip-address* | *host-name* }
  d. (Optional) Configure the IP address of the interface for sending packets.
     source-address *ipv4-address*

e. (Optional) Configure the initial TTL.
first-ttl *first-ttl*

f. (Optional) Configure the maximum TTL.
max-ttl *max-ttl*

g. (Optional) Configure the port number.
udp-port *udp-port*

h. (Optional) Configure the timeout interval.
timeout *timeout*

i. (Optional) Configure the number of sent packets.
count *count*

j. (Optional) Configure an outbound interface.
if-name *if-name*

k. (Optional) Configure the packet length.
packet-size *packet-size*

l. Commit the configuration.
emit

----End

Example

- Perform a tracert operation to test an IPv4 network.

```
[(x)user@localhost]
MDCLI> ipv4-start-ip-trace
[(x)user@localhost]/ipv4-start-ip-trace
MDCLI> test-name test
[*(x)user@localhost]/ipv4-start-ip-trace
MDCLI> dest-ip-addr 10.136.152.27
[*(x)user@localhost]/ipv4-start-ip-trace
MDCLI> emit
traceroute to 10.136.152.27 (10.136.152.27), 30 hops max, 46 byte packets
 1  169.254.195.10 (169.254.195.10)  3.323 ms  3.627 ms  3.927 ms
 2  169.254.192.1 (169.254.192.1)  3.212 ms  2.082 ms  2.489 ms
 3  10.131.142.66 (10.131.142.66)  1.567 ms  1.664 ms  1.792 ms
 4  * * *
 5  10.136.152.27 (10.136.152.27)  33.744 ms  34.224 ms  33.693 ms
```

13.3 Telemetry Configuration

13.3.1 Overview of Telemetry

Definition

Telemetry is a technology that remotely collects data from physical or virtual devices at a high speed. These devices periodically send interface traffic statistics, CPU usage, and memory usage to collectors in push mode. Compared with the traditional pull mode (question-answer interaction), the push mode provides faster and real-time data collection.

Purpose

As the software-defined networking (SDN) scale increases, more and more services need to be carried, and users have placed higher requirements on

intelligent SDN O&M. Specifically, monitoring data requires a sampling interval with higher precision, so that microburst traffic can be efficiently detected and adjusted. The monitoring process should have little impact on device functions and performance in order to improve device and network utilization.

Conventional network monitoring methods, such as Simple Network Management Protocol (SNMP) Get and command line interface (CLI), offer low management efficiency and are unable to meet user requirements due to the following disadvantages:

- Pull mode is used to obtain monitoring data from devices. This mode cannot be used to monitor a large number of network nodes, which limits the network growth.
- Generally, a sampling interval is accurate to minute. To improve accuracy, data must be queried more frequently, which increases network node CPU usage, and affects normal device functions.
- Data obtained from monitored network nodes is inaccurate due to network transmission delays.

Therefore, telemetry technology has been developed to implement large-scale and high-performance monitoring on network devices. Through this technology, the intelligent O&M system can manage more devices, monitoring data can be obtained in real time with higher precision, and the monitoring process has little impact on device functions and performance. Telemetry also provides the most important big data basis for fast fault locating and network quality optimization. It converts network quality analysis into big data analytics, effectively supporting intelligent O&M. **Table 13-8** compares telemetry and conventional network monitoring modes.

Table 13-8 Comparison between telemetry and conventional network monitoring modes

Item	Telemetry	SNMP Get	SNMP Trap	CLI	Syslog
Working mode	Push	Pull	Push	Pull	Push
Precision	Subseconds Milliseconds	Minutes	Seconds	Minutes	Seconds
Data scope	All data	All data	**Traps only**	All data	Events only

Item	Telemetry	SNMP Get	SNMP Trap	CLI	Syslog
Structured	Structured using the YANG model	Structured using MIBs	Structured using MIBs	Non-structured	Non-structured

NOTE

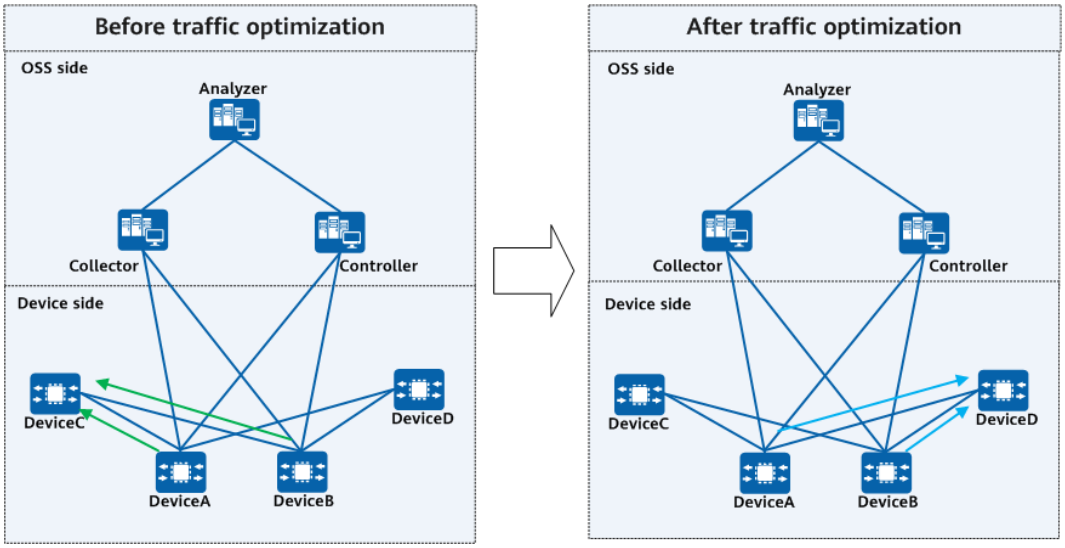
As described in **Table 13-8**, although SNMP trap and syslog use the push mode, only traps or events are pushed. Monitoring data such as interface traffic statistics cannot be collected or sent.

Benefits

Using telemetry technology, a collector can collect a large amount of device data and send the data to an analyzer for comprehensive analysis. The analyzer sends the results to a controller, which then adjusts device configurations accordingly, enabling you to determine whether the adjusted device status meets expectations in real time.

As shown in **Figure 13-3**, DeviceA and DeviceB support telemetry. Assume that traffic of DeviceA and DeviceB is transmitted to DeviceC along the paths in green. When the optimal path changes, telemetry-enabled devices can collect data indicators and push them to the collector. The analyzer reads the data indicators stored in the collector, analyzes them and makes appropriate decisions, and sends the results to the controller. The controller then sends the configurations to be adjusted to DeviceA and DeviceB, and the traffic of these devices is then transmitted to DeviceD through the paths in blue. The traffic optimization result can be quickly reported to the operations support system (OSS) and is used to determine whether traffic optimization meets the expectations.

Figure 13-3 Diagram of telemetry traffic optimization



Compared with traditional network monitoring modes, telemetry greatly reduces both the adjustment and feedback delays. It frees you from being affected by traffic path changes and facilitates management and maintenance.

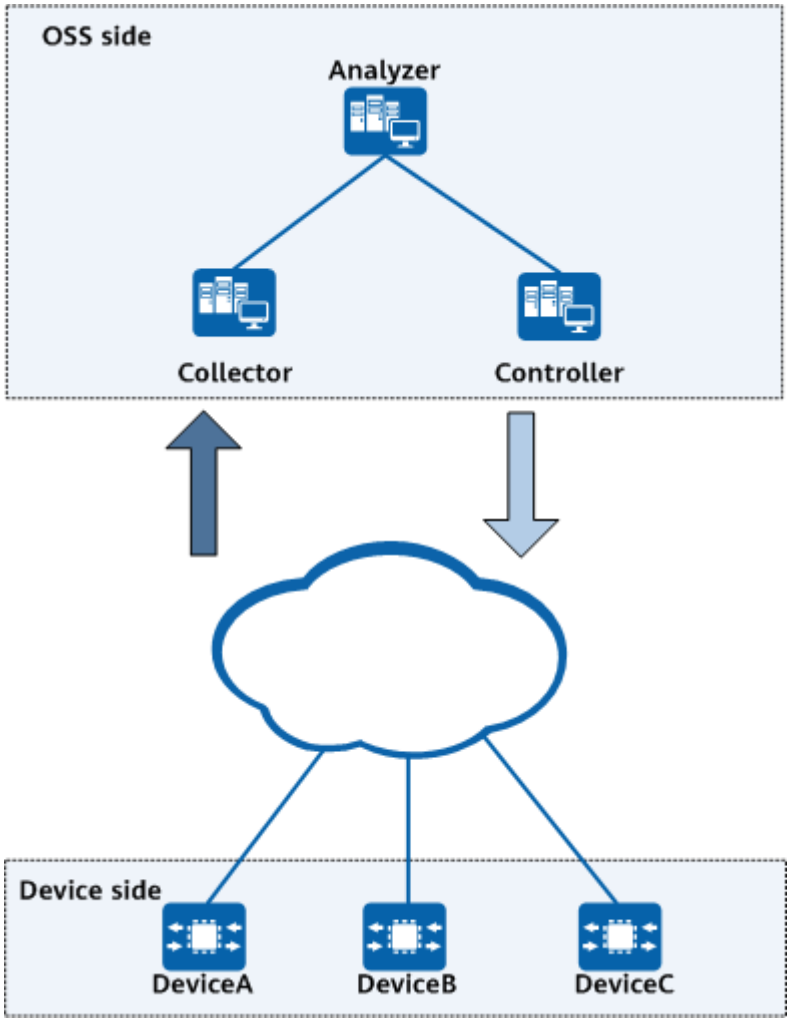
13.3.2 Understanding Telemetry

13.3.2.1 Telemetry System Architecture and Service Process

System Architecture

Telemetry is a closed-loop automatic O&M system that consists of the device side (network devices) and OSS side including the collector, analyzer, and controller, as shown in **Figure 13-4**.

Figure 13-4 Telemetry system architecture



Service Process

Telemetry's OSS and device sides collaborate to complete the telemetry service process shown in **Figure 13-5**.

1. Configuring a subscription: A subscription can be configured on network devices to subscribe to data sources for data collection. Currently, only static subscription is supported. You can run commands to configure subscription data sources for data collection.
2. Pushing sampled data: A network device reports sampled data to the collector based on controller configurations. The collector receives and stores the data.
3. Reading data: The analyzer reads the data stored in the collector.
4. Analyzing data: The analyzer analyzes the data and sends the results to the controller for network management and optimization.
5. Adjusting network parameters: The controller delivers network configurations to the network devices that need to be adjusted. After these configurations take effect, the devices report new sampled data to the collector. The telemetry OSS side analyzes whether the network optimization meets expectations. The service process is closed once optimization is complete.

NOTE

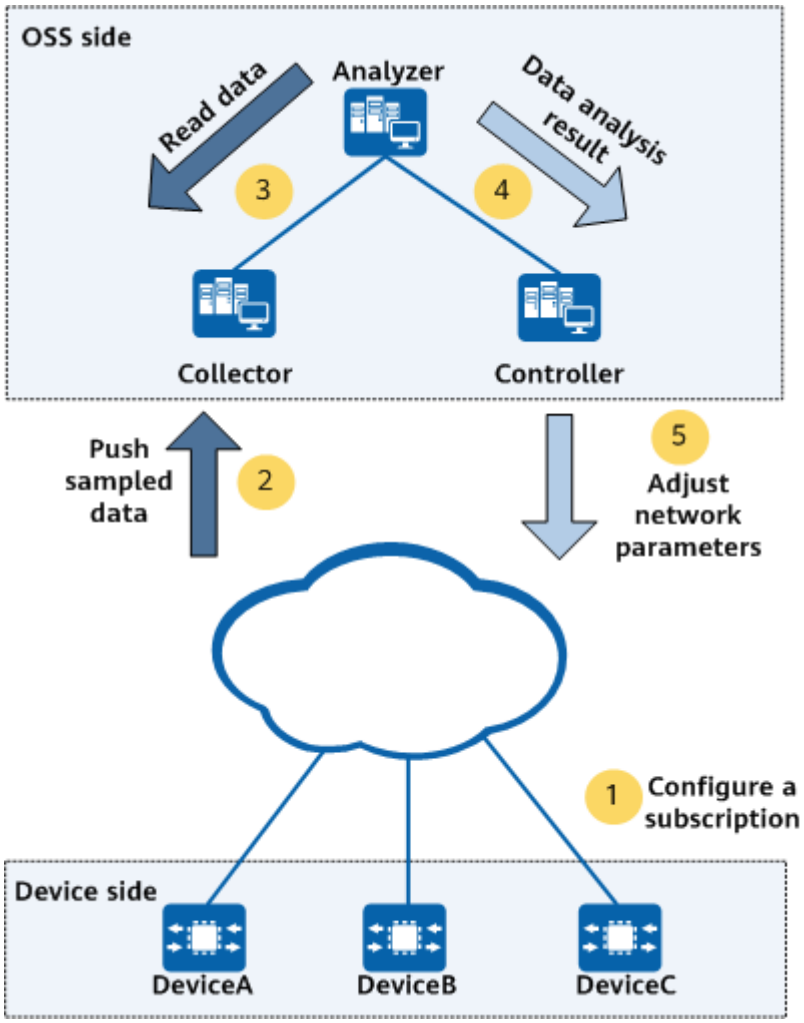
Telemetry mentioned in this document refers to the telemetry OSS side unless otherwise specified.

If the connection between a device and the collector is interrupted, the device reconnects to the collector and resends the data. However, any data sampled while the connection is being re-established is lost.

When the system restarts after saving the telemetry service configuration, the device reloads the telemetry service configuration so that the service can run properly. However, the data sampled during the restart is lost.

Only administrators can use the telemetry function.

Figure 13-5 Telemetry service process



13.3.2.2 Key Telemetry Components

A telemetry system consists of two parts:

- OSS side: consists of the collector, analyzer, and controller.
- Device side: consists of network devices, which encode the sampled data based on the encoding format and transmit the data using a transport protocol.

NOTE

This chapter describes key components on the device side only.

13.3.2.2.1 Sampled Data

Introduction to Sampled Data

The sampled data includes the following:

- Original data: Telemetry-enabled devices collect data from their forwarding, control, and management planes. Currently, the data that can be collected includes interface traffic statistics, CPU usage, and memory usage.

- Data model: Telemetry-enabled devices collect data based on YANG models.

  YANG is a data modeling language developed to define configuration data models, status data models, remote procedure call (RPC) models, and notification mechanisms for transport protocols.
- Performance indicators: Currently, telemetry can collect data specified by certain sampling paths.

Sampling Path

A sampling path defines the data to be sampled. Data on the device has been described in the YANG model. The YANG model and its subtree paths can form sampling paths.

Sampling Path Format

- Basic format

  Example: huawei-ifm:ifm/interfaces/interface/common-statistics

  huawei-ifm before the colon (:) indicates the YANG model name, and **ifm/interfaces/interface/mib-statistics** indicates the node names in the YANG model. The node names at different layers are separated by slashes (/).
- Model mounting format

  Example: huawei-ifm:ifm/interfaces/interface/mib-statistics/huawei-pic:eth-port-err-sts

  huawei-ifm and **huawei-pic** before the colons (:) indicate two YANG model names. The **eth-port-err-sts** node of **huawei-pic** is mounted to the **mib-statistics** node of **huawei-ifm** using the augment parameter.

Based on the preceding two formats, nodes in YANG models can be converted into sampling paths.

Sampling Interval

The sampling interval refers to the interval at which the device proactively sends information such as interface traffic statistics, CPU data, or memory data to the collector.

13.3.2.2.2 Encoding Formats

Currently, only the JavaScript Object Notation (JSON) encoding format is supported.

JSON Encoding

JSON is a lightweight data exchange format. Based on a subset of ECMAScript (a JavaScript standard developed by the ECMA), JSON uses a text format independent of programming languages to store and express data with a simple and clear structure, so that the data can be easily read or compiled by development personnel and parsed or generated by devices. **Table 13-9** shows the JSON encoding format.

Telemetry supports pure JSON encoding format. Both the telemetry layer and service data layer use JSON encoding, as described in **Table 13-9**.

Table 13-9 JSON encoding format

JSON Encoding

```
{
  "telemetry": [{
    "node_id_str": "HUAWEI",
    "subscription_id_str": "sub1",
    "sensor_path": "cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info",
    "proto_path": "",
    "collection_id": "3",
    "collection_start_time": "1666426823",
    "msg_timestamp": "1666426823",
    "collection_end_time": "1666426823",
    "current_period": 300000,
    "except_desc": "OK",
    "product_name": "S380",
    "encoding": "Encoding_JSON",
    "data_str": {
      "row": [{
        "timestamp": "1666426823",
        "content": {
          "cpu-memory": {
            "board-cpu-infos": {
              "board-cpu-info": {
                "slot-id": "0",
                "cpu-id": "0",
                "overload-threshold": 90,
                "system-cpu-usage": 26
              }
            }
          }
        }
      }],
      "delete": [],
      "generator": {
        "generator_id": "0",
        "generator_sn": "0",
        "generator_sync": false
      }
    },
    "software_version": "V600R022C00",
    "mac_address": "00-00-00-00-00-00"
  }]
}
```

NOTE

If telemetry uses JSON encoding, the data packets sent to the collector do not contain typesetting characters such as carriage return or space, reducing the size of JSON-encoded data.

13.3.2.2.3 Transport Protocol

Telemetry uses the HTTP protocol to report the data encapsulated in an encoding format to a collector.

Overview of HTTP

HTTP is a request/response protocol and therefore involves request and response messages.

Request message

An HTTP client sends a request message to an HTTP server. This request message consists of three parts: request line, request header, and request body, as shown in **Figure 13-6**.

Figure 13-6 Request format

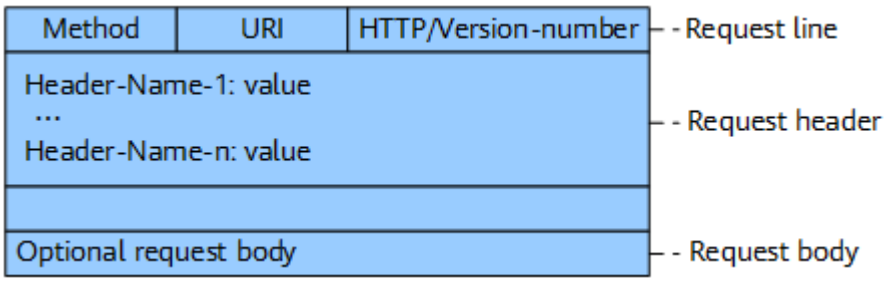


Table 13-10 Request fields

Field	Description
Method	HTTP method, performed on the resource identified by the request URI. It includes the following methods: GET, HEAD, PUT, POST, TRACE, OPTIONS, DELETE, and extension-method.
URI	URL.
HTTP/Version-number	HTTP version.
Header-Name-n:value	Header field name: value.
Optional request body	(Optional) Request body.

Interconnection When the Device Functions as an HTTP Client

When a device functions as an HTTP client and connects to the controller using HTTP/2, the **Method**, **URI**, and **content-type** fields in the packet header need to be set to **POST**, **huawei-telemetry/data/http2.0**, and **application/yang-data +json+telemetry**, respectively. After the connection is set up, it sends encoded telemetry data to the controller through data frames.

When the system restarts after saving the telemetry service configuration, the device reloads the telemetry service configuration so that the service can run properly. However, the data sampled during the restart is lost.

13.3.3 Configuring Telemetry Subscription on the Device

13.3.3.1 Configuring Static Subscription

13.3.3.1.1 Understanding Static Subscription

Definition

In static telemetry subscription, a device functioning as a client initiates a connection to a collector functioning as a server in order to report sampled data.

Fundamentals

You can use commands to configure telemetry-capable devices by subscribing to data sources in order to collect data.

If the connection between a device and the collector is interrupted, the device reconnects to the collector and resends the data. However, any data sampled while the connection is being re-established is lost.

When the system restarts after saving the telemetry service configuration, the device reloads the telemetry service configuration so that the service can run properly. However, the data sampled during the restart is lost. As this poses high pressure requirements on devices, telemetry static subscription is often used for coarse-grained data collection.

NOTE

Only administrators can use the static subscription function.

13.3.3.1.2 Configuring a Destination Collector for Receiving Sampled Data

Context

When configuring static telemetry subscription to collect sampled data, you need to create a destination group and specify its working mode, and configure a destination collector for receiving the sampled data (destination collector for short), including the IP address and port number.

For details about configuration parameters, see huawei-openconfig-telemetry.yang.

Procedure

Step 1 Enter the edit-config mode.

```
edit-config
```

Step 2 Create a destination group where a destination collector belongs and enter the destination group view.

```
telemetry-system destination-groups destination-group group-id destination-group-name
config
group-id destination-group-name
```

Step 3 Configure the IP address and port number of the destination collector.

```
destinations destination destination-address ip-address-ip4 destination-port port-value
config
destination-address ip-address-ip4 destination-port port-value
```

Step 4 (Optional) Configure an SSL policy.

```
ssl-policy-name name
```


By default, the SSL policy in the default PKI realm is used.

Step 5 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

Run the **display telemetry-system/destination-groups/destination-group[group-id="***destination-group-name***"]** command to check destination group information.

13.3.3.1.3 Configuring a Sampling Path

Context

When configuring static subscription to collect sampled data, you need to create a sampling sensor group and specify a sampling path.

For details about configuration parameters, see huawei-openconfig-telemetry.yang.

Procedure

Step 1 Enter the edit-config mode.

```
edit-config
```

Step 2 Create a sampling sensor group, and enter the sensor group view.

```
telemetry-system sensor-groups sensor-group sensor-group-id sensor-name
config
sensor-group-id sensor-name
```

Step 3 Configure a sampling path.

```
sensor-paths sensor-path path path
config
path path
```

NOTE

A maximum of 64 sampling paths can be configured for a sampling sensor group.

----End

Verifying the Configuration

Run the **display telemetry-system/sensor-groups/sensor-group[sensor-group-id="***sensor-name***"]** command to check sensor group information.

13.3.3.1.4 Creating a Static Subscription

Context

When configuring static subscription to collect sampled data, you need to create a sampling sensor group and specify a sampling path. When sampled data is collected through the sampling path, the device reports the sampled data to the collector for service policy determination.

For details about configuration parameters, see huawei-openconfig-telemetry.yang.

Procedure

Step 1 Create a subscription.

1. Enter the edit-config mode.

```
edit-config
```

2. Create a subscription for associating a destination group and a sampling sensor group, and enter the subscription view.

```
telemetry-system subscriptions persistent subscription subscription-name subscription-name
config
subscription-name subscription-nam
```

3. Associate the subscription with a destination group.

```
destination-groups destination-group group-id destination-group-name
config
group-id destination-name
```

4. Associate the subscription with a sampling sensor group, and configure a sampling interval for the sampling sensor group.

```
sensor-profiles sensor-profile sensor-group destination-name
config
sensor-group sensor-name
sample-interval sample-interval
group-id destination-name
```

NOTE

You need to specify telemetry sampling paths for different sampling sensor groups based on different sampling intervals.

Step 2 (Optional) Configure the source IP address for the packets to be reported.

```
telemetry-system subscriptions persistent subscription subscription-name subscription-name
config
local-source-address ip-address
```

By default, no source IP address is configured for the packets to be reported.

Step 3 Commit the configuration.

```
commit
```

----End

Verifying the Configuration

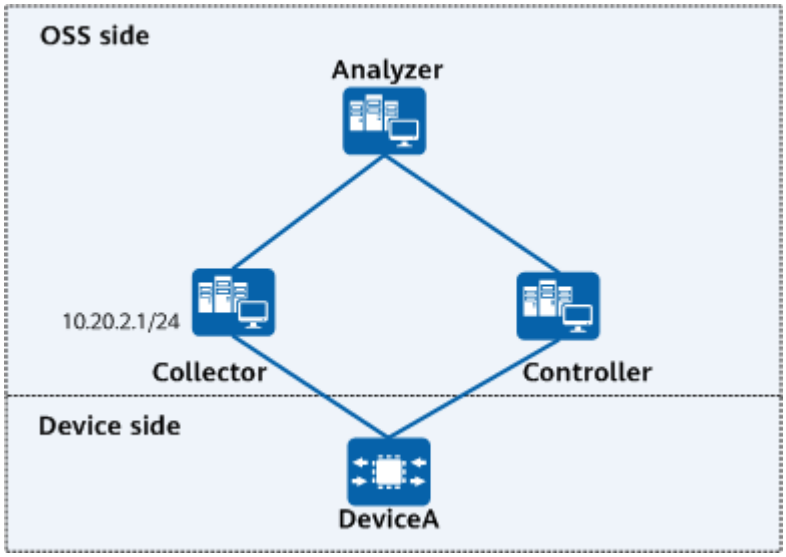
Run the **display telemetry-system/subscriptions/persistent/subscription[subscription-name="***subscription-name***"]** command to check subscription information.

13.3.3.1.5 Example for Configuring Static Subscription (IPv4 Collector)

Networking Requirements

As shown in **Figure 13-7**, DeviceA supports telemetry and establishes an HTTP/2 connection with the collector to send data to the collector.

Figure 13-7 Networking diagram for configuring static telemetry subscription



Procedure

Step 1 Configure a destination collector for receiving sampled data.

On DeviceA, create the destination group **destination1** to which a destination collector belongs, set the IP address and port number of the destination collector to 10.20.2.1 and 10001, respectively.

```
[user@HUAWEI]
MDCLI> edit-config
[(gl)user@HUAWEI]
MDCLI> telemetry-system destination-groups destination-group group-id destination1
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]
MDCLI> config
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/
config
MDCLI> group-id destination1
[*(gl)root@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/
config
MDCLI> quit
[*(gl)root@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]
MDCLI> destinations destination destination-address 10.20.2.1 destination-port 10001
[*(gl)root@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/
destinations/destination[destination-address="10.20.2.1"][destination-port="10001"]
MDCLI> config
[*(gl)root@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/
destinations/destination[destination-address="10.20.2.1"][destination-port="10001"]/config
MDCLI> destination-address 10.20.2.1 destination-port 10001
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/
config
MDCLI> commit
```

Step 2 Configure a sampling path.

Configure the sampling sensor group **sensor1** and set the sampling path to **huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-infohuawei-debug:debug/cpu-infos/cpu-info**.

```
[(gl)user@HUAWEI]
MDCLI> telemetry-system sensor-groups sensor-group sensor-group-id sensor1
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]
MDCLI> config
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]/config
```



```
MDCLI> sensor-group-id sensor1
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]/config
MDCLI> quit
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]
MDCLI> sensor-paths sensor-path path huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]/sensor-paths/sensor-path[path="huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info"]
MDCLI> config
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]/sensor-paths/sensor-path[path="huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info"]/config
MDCLI> path huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info
[*(gl)user@HUAWEI]/telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]/sensor-paths/sensor-path[path="huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info"]/config
MDCLI> commit
```

Step 3 Create a static subscription.

On DeviceA, create a subscription named **subscription1** and associate it with the sampling sensor group **sensor1** and destination group **destination1**.

```
[(gl)user@HUAWEI]
MDCLI> telemetry-system subscriptions persistent subscription subscription-name subscription1
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]
MDCLI> config
[*(gl)usr@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/config
MDCLI> subscription-name subscription1
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/config
MDCLI> quit
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]
MDCLI> sensor-profiles sensor-profile sensor-group sensor1
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/sensor-profiles/sensor-profile[sensor-group="sensor1"]
MDCLI> config
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/sensor-profiles/sensor-profile[sensor-group="sensor1"]/config
MDCLI> sensor-group sensor1
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/sensor-profiles/sensor-profile[sensor-group="sensor1"]/config
MDCLI> sample-interval 60000
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/sensor-profiles/sensor-profile[sensor-group="sensor1"]/config
MDCLI> quit 3
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]
MDCLI> destination-groups destination-group group-id destination1
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/destination-groups/destination-group[group-id="destination1"]
MDCLI> config
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-name="subscription1"]/destination-groups/destination-group[group-id="destination1"]/config
MDCLI> group-id destination1
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/config
MDCLI> quit
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]
MDCLI> destinations destination destination-address 10.20.2.1 destination-port 10001
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/destinations/destination[destination-address="10.20.2.1"][destination-port="10001"]
MDCLI> config
[*(gl)user@HUAWEI]/telemetry-system/destination-groups/destination-group[group-id="destination1"]/destinations/destination[destination-address="10.20.2.1"][destination-port="10001"]/config
MDCLI> destination-address 10.20.2.1 destination-port 10001
[*(gl)user@HUAWEI]/telemetry-system/subscriptions/persistent/subscription[subscription-
```



```
name="subscription1"]/destination-groups/destination-group[group-id="destination1"]/config
MDCLI> commit
```

----End

Verifying the Configuration

- Run the **display telemetry-system/destination-groups/destination-group[group-id="***destination-group-name***"]** command to check destination group information.

```
[user@HUAWEI]
MDCLI> display telemetry-system/destination-groups/destination-group[group-id="destination1"]
{
  "group-id": "destination1",
  "config": {
    "group-id": "destination1"
  },
  "destinations": {
    "destination": [
      {
        "destination-address": "10.20.2.1",
        "destination-port": "10001",
        "config": {
          "destination-address": "10.20.2.1",
          "destination-port": 10001
        }
      }
    ]
  }
}
```

- Run the **display telemetry-system/sensor-groups/sensor-group[sensor-group-id="***sensor-name***"]** command to check sensor group information.

```
[user@HUAWEI]
MDCLI> display telemetry-system/sensor-groups/sensor-group[sensor-group-id="sensor1"]
{
  "sensor-group-id": "sensor1",
  "config": {
    "sensor-group-id": "sensor1"
  },
  "sensor-paths": {
    "sensor-path": [
      {
        "path": "huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info",
        "config": {
          "path": "huawei-cpu-memory:cpu-memory/board-cpu-infos/board-cpu-info"
        }
      }
    ]
  }
}
```

- Run the **display telemetry-system/subscriptions/persistent/subscription[subscription-name="***subscription-name***"]** command to check subscription information.

```
[user@HUAWEI]
MDCLI> display telemetry-system/subscriptions/persistent/subscription[subscription-
name="subscription1"]
{
  "subscription-name": "subscription1",
  "config": {
    "subscription-name": "subscription1"
  },
  "sensor-profiles": {
    "sensor-profile": [
      {
        "sensor-group": "sensor1",
        "config": {
```



```
        "sensor-group": "sensor1",
        "sample-interval": 60000
      }
    }
  ]
 },
 "destination-groups": {
  "destination-group": [
   {
    "group-id": "destination1",
    "config": {
     "group-id": "destination1"
    }
   }
  ]
 }
}
```