YUNQI DOC

Release 0.1

YUNQI TECH

Aug 24, 2017

Contents

欢迎使用ConnetOS用户手册。

作为史上最懂网络的开放交换机操作系统，ConnetOS在网络智能分析、数据挖掘和机器学习等方面拥有的核心经验和实践，是应对云计算和人工智能时代下数据中心网络挑战的最佳方案。

通过ConnetOS用户手册，您可以了解ConnetOS支持的丰富的二三层功能，强大的智能分析和诊断功能，以及如何使用这些功能。

CHAPTER 1

快速入门

认识ConnetOS

简介

ConnetOS系统（以下简称为ConnetOS）是云启科技开发的一款基于Debian GNU/Linux的开放交换机操作系统，适用于现代数据中心网络环境。

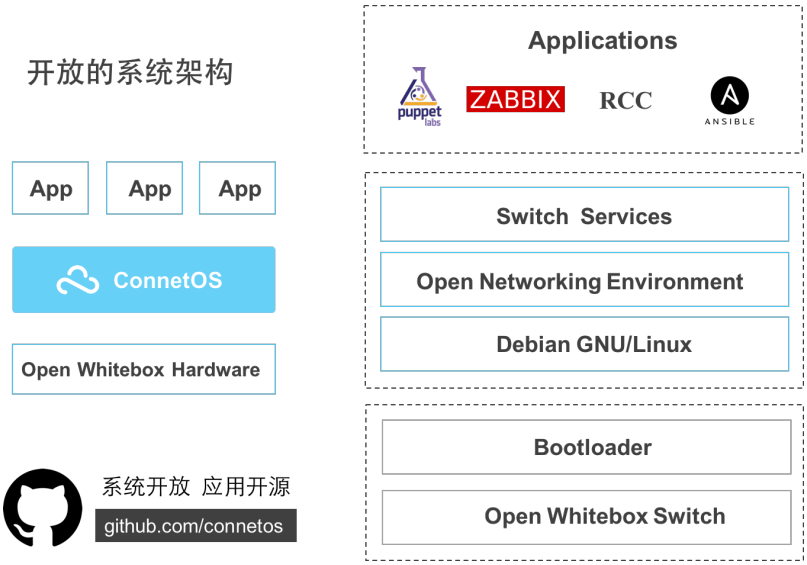
ConnetOS以数据中心网络的运维痛点和运营效率出发，前所未有地开放网络资源和自身能力，提供以数据中心网络自动化和智能化为目标的一系列功能，着眼于消除数据中心网络运维痛点，让网络不仅开放而且易运维。

通过ConnetOS系统，用户可以二次开发或完全编写自己的代码、部署第三方软件，甚至可以基于ConnetOS提供的资源化、服务化能力进行交换机转发逻辑的控制。

特点

更开放的架构

ConnetOS是基于开放网络理念构建的一个交换机操作系统，适用于现代数据中心网络架构。ConnetOS的Linux系统完全面向用户开放，让用户体验到“A Switch as A Server”，享受标准Linux发行版带来的便利性以及良好的软件生态。



ConnetOS：

- 基于白盒的操作系统，能够在运行在不同的设备厂商上，带来全新级别的自由度和灵活性。
- 系统完全开放给用户，提供用户在Linux系统上直接访问的能力。
- 应用程序开源，可安装更为丰富的应用软件，满足各类业务的部署扩展需求。

更丰富的接口

ConnetOS开放多种格式的接口，满足不同用户对网络的使用需求，提升自动化效率。

- Linux：Debian Linux系统的缺省shell环境。
- CLI：命令行接口视图，方便用户进行网络操作和管理。
- RCC：在本地实现对交换机的远程配置。
- API：开放的API接口提供配置、状态查询和故障定位，同时可以提供给基于API的二次开发。

更智能的网络

全方位关注网络运维：

- 聚焦网络运维痛点
  - 强大的网络质量感知能力：监听、分析转发平面所有丢弃和转发的数据包。
  - 丰富的网络工具集。
- 提升网络运维效率
  - 全自动网络部署：ATP提供高效零配置上线解决方案。
  - 三平面立体监控：全方位的监控体系Streaming Telemetry（sFlow、sDrop、sMetric），让数据统计和导出更精准高效。
  - 快速升级软件：Warmboot的升级方式，保证数据中心网络设备的快速迭代升级。
- 关注设备健康状态
  - 多维度建模：建立自主研发的健康评估管理体系。

- 大数据分析：实时进行信息搜集和分析。

亮点特性

能实现交换机任意上架的自动部署能力

ConnetOS支持的ATP功能，是目前数据中心领域最简单、高效、自动化程度最高的零配置上线方案。

- 真正意义上的全自动、零人工参与。
- 网络规划视角驱动，无额外运维负担增加。
- 无缝整机替换，即插即用。
- 强大的容错能力。

远程配置能力

RCC（Remote Config Client）远程CLI调用，通过此功能，用户在本地就可以实现对交换机的远程配置。RCC的使用极其简单，用户在使用时不需要进行任何配置，在本地编译之后，即可直接使用。

独一无二的丢包感知能力

能够识别静默丢包：

- 对转发平面所有数据包（转发的或丢弃的）进行监视和分析。
- 实时捕获被设备丢弃的数据包，并记录丢包原因。
- 根据五元组实时计算网络转发路径。
- 具备端口拥塞感知和实时上报能力。
- 高精度端口统计，最高精度1秒。

网络可视化能力

ConnetOS提供的abbix模版，可以实时监控交换机的系统、网络和服务状况。
在Zabbix Server上导入云启提供的监控模板和shell脚本后可以监控ConnetOS交换机。

网络排障工具集

ConnetOS自身提供丰富的分析诊断工具集。

- Ifconfig
- Mirror
- Navmesh
- Netstat
- Ping
- sDrop
- sFlow
- Tcpdump
- Traceroute

创新应用

ConnetOS提供如下的解决方案：

- 数据中心分流具有：
  - 同源同宿的流量分配行为。
  - 灵活的流量分配方式（M：N）。
  - 强大的会话过滤机制。
- 内容分发网络负载均衡：
  - 服务器故障后不影响其他机器的原有连接分布。
  - 故障机器修复只会恢复原本属于该机器上的连接，不影响其他机器上原有业务分布。
  - 不用部署专门的LB，运维管理方便，节省成本。

ConnetOS的使用

操作方式

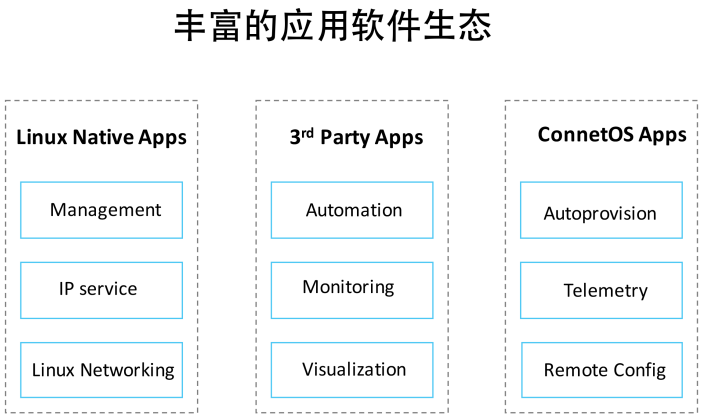
ConnetOS为用户提供三种方式，对设备进行操作。

- Linux shell：Linux界面视图，用户初始登录的视图，亦即Debian Linux系统的缺省shell环境。
- CLI shell：通过命令行接口（Command-line Interface），对交换机的网络操作和管理。
  用户登录到设备后，输入账户和密码，进入到Linux shell，输入cli，即可进入CLI shell。
- RESTful API：用API来灵活的管理设备和查询设备运行状态。

除了本地可以对设备进行配置等操作，ConnetOS还提供RCC远程配置功能。用户通过RCC功能，可以在本地对交换机进行远程配置。RCC的使用极其简单，用户在使用时不需要进行任何配置，在本地编译之后，即可直接使用。

ConnetOS支持的第三方软件

开源的交换机软件，可以直接安装在ConnetOS上，满足各类业务的部署扩展需求。



所有第三方软件都可在云启科技官网的软件商店进行下载：

http://www.connetos.com/app-store

业务功能

特性列表

ConnetOS支持如下的功能特性：

分类	特性
自动部署	Auto provision：ATP Zero Touching Provision：ZTP
管理	Telnet、SSH、TFTP 聚合网段管理 带内管理（loopback、vlan-interface） 带外管理 SNMP、DHCP、DHCP relay 基于TACACS+的认证、授权、计费 基于事务机制的命令行 多达49级的配置回滚
交换与路由	VLAN、LAG、LLDP、LACP 静态路由、OSPF、BGP、ECMP LAG、ECMP 哈希：一致性哈希、对称哈希、二次哈希 路由策略
QoS	基于优先级的映射：DSCP、IEEE 802.1p、Trust-port 队列调度模式：SP、WDRR、SP+WDRR
策略	访问控制列表：L2、L3、L4 fields 转发策略：分类、镜像、策略、交换、路由
流式遥测	sFlow：监控接口转发数据包级别 sDrop：分析丢弃的数据包 sMetric：监控管理面和控制面状态、配置和事件
分析和诊断	Tcpdump：Linux系统直接抓包 Mirror：业务端口数据包镜像 Navmesh：网络转发路径计算
DEVOPS	Remote CLI Call（RCC） Python SDK RESTful API agt-get/dpkg软件包：Puppet、Ansible、Zabbix
虚拟化	VXLAN bridging
可靠性	高智能堆叠系统：ISS

解决方案

ConnetOS提供如下的创新解决方案：

方案名称	描述
自动化	极简高效零配置上线方案
TAP分流	数据中心分流
CDN负载均衡	新型网络架构
诊断与分析	先进的流式遥测功能
SDN	软件定义网络

What's New

第二次发布

发布时间

版本号

新增特性

第一次发布

发布时间

版本号

新增特性

CHAPTER 2

系统安装

ConnetOS在正常运行前，需要进行系统软件和配置文件的安装。

ConnetOS安装

首次安装

简介

交换机在出厂时一般已经附带了完整的ConnetOS，如果没有安装，用户可以通过ONIE（Open Network Install Environment）安装完整的ConnetOS。

ConnetOS安装完成后，设备启动后直接载入ConnetOS运行，ONIE不再运行。当然在需要时，仍然可以激活ONIE用于OS的升级、重装等安装部署操作。

Note: 如果设备没有预装ONIE，也可以通过安装在U盘的ONIE进行安装。

安装步骤

1. 查看设备盘:

```
fdisk -l
```

2. 对设备盘进行分区:

```
fdisk /dev/sda
```

3. 创建分区:

```
1

p +22G

2 p

2 t 82
```

4. 设置分区标签:

```
mkfs.ext4 /dev/sda1 -L /;mkswap /dev/sda2 -L SWAP;e2label /dev/sda1;swaplabel /
↪dev/sda2;
```

5. 挂载分区到目录，并拷贝bin文件到目录下，并进行解压:

```
mount /dev/sda1 /mnt;cd /mnt;cp /connetos_c1020_2.1.5-30k17_amd64.bin .;sed -i 1d␣
↪ConnetOS_C1020_2.0.1_43D18.bin;
```

6. 设置设备时间，防止解压出错:

```
date -s 20170101;tar zxf connetos_c1020_2.1.5-30k17_amd64.bin;ls;
```

7. 安装引导，进行重启:

```
grub-install --boot-directory=/mnt/boot /dev/sda;rm -rf connetos_c1020_2.1.5-
↪30k17_amd64.bin;sync;reboot;
```

8. 重启成功，软件安装完成。

   软件安装之后，请继续进行配置文件的安装。

升级

ConnetOS是基于Debian GNU/Linux的操作系统，ConnetOS的升级分为linux升级和switch升级两部分。
云启提供如下的文件，以供升级:

- linux：connetos_*.bin，如connetos_c1020_2.1.5-30k17_amd64.bin。
- switch OS：switch_*.deb，如switch_c1020_2.1.5-30z19_amd64.deb。

switch可以单独升级，但是linux必须和switch一起升级。

Note:

- 执行 **request system reboot** 时，需要输入“yes“进行确认。
- 升级时，不会覆盖原有配置文件。

如果升级失败，重启设备进行ConnetOS的安装即可。

升级linux和switch

以U盘升级方式为例，升级步骤如下:

1. 登录到交换机上:

```
root@ConnetOS:~$
```

2. 查看设备分区:

```
root@ConnetOS:~$ fdisk -l
```

3. 将U盘挂载到mnt下:

```
root@ConnetOS:~$ mount /dev/sdc4 /mnt
```

4. 查看bin文件是否成功挂载到mnt下:

```
root@ConnetOS:~$ cd /mnt/

root@ConnetOS:/mnt$ ls

connetos_c1020_2.1.5-30k17_amd64.bin  IxNetwork_8.20_EA.exe  System Volume␣
↪Information

connetos_c1020_2.1.5-30w18_amd64.bin  setup.exe
```

5. 将connetos_*.bin拷贝到/var/upgrade目录下，例如:

```
root@ConnetOS:/mnt$ cp connetos_c1020_2.1.5-30k17_amd64.bin /var/upgrade/
```

6. 重启设备，系统将自动完成升级。两种重启方式选择一种即可。
   - 在linux shell下执行:

     ```
     root@ConnetOS $ reboot
     ```

   - 在CLI shell下，执行:

     ```
     root@ConnetOS $ cli

     ConnetOS> request system reboot
     ```

升级switch

switch升级的步骤如下:

1. 登录到交换机上:

```
root@ConnetOS:~$
```

2. 安装升级文件:

```
root@ConnetOS:~$ dpkg -i switch_c1020_2.1.5-30z19_amd64.deb
```

3. 重启设备，系统将自动完成升级。两种重启方式选择一种即可。
   - 在linux shell下执行:

     ```
     root@ConnetOS $ reboot
     ```

   - 在CLI shell下，执行:

```
ConnetOS> request system reboot
```

配置文件安装

ConnetOS的配置文件有两种安装方式:

- ATP自动部署
- 手动安装

Note: 自动部署功能为ConnetOS交换机出厂自启动功能，在ConnetOS交换机上线及工作过程中，不需要再进行额外的配置。

ATP自动部署

简介

概述

自动部署ATP（Auto Provision）功能，是指新出厂或空配置的设备，上电启动时采用的一种自动加载配置文件的功能。

ConnetOS交换机支持两种方式实现设备的自动部署:

- 极简部署方式

  只需要在核心交换机的端口描述上新增扩展信息，通过LLDP协议传递如下信息，就能实现零配置和自动化:

  - 管理网口IP地址
  - TFTP服务器IP的第4个字节

  例如，端口描述“TO-XXXXXXX@192.168.1.10@100”或“TO-XXXXXXX@192.168.1.10@192.168.1.100”表示，TOR管理IP地址为192.168.1.10，TFTP服务器IP地址为192.168.1.100，配置文件的名字为192.168.1.10_yunqi.cfg。

- ZTP（Zero Touch Provisioning）方式

  该方式需要部署DHCP服务器，并在DHCP服务器上配置必要的参数。

  - 管理网口IP地址
  - TFTP服务器IP地址

基于LLDP的极简部署方式是云启开放专利的最简单高效的零配置自动部署方案，推荐使用这种方式来部署，与传统的基于DHCP的ZTP相比，具有如下优点:

- 真正意义上的全自动、零人工参与。
- 网络规划视角驱动，无额外运维负担增加。
- 无缝整机替换，即插即用。
- 强大的容错能力。

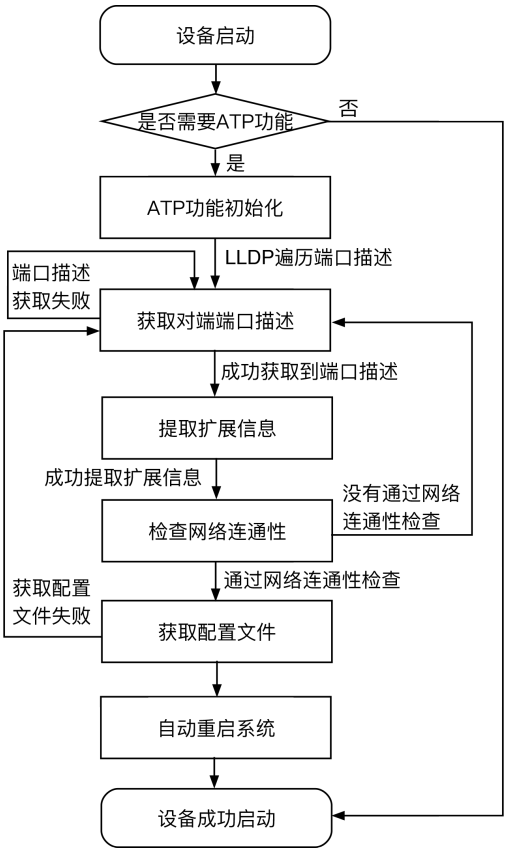
两种部署方式的上线流程比较如下表所示。

	极简部署流程	ZTP部署流程
上线设备IP地址	通过LLDP自动配置IP地址	通过DHCP获取IP地址
维护设备MAC和配置关系	不需要维护，只需准备配置文件即可	需要手动维护
DHCP服务器	不需要专门维护，无需扩展	需要扩展option来传递配置信息
配置过程记录	完整纪录配置细节，可远程跟踪异常	不记录

ATP工作原理

ATP的两种工作模式是同时运行的，根据设备外界的配置环境自动选取工作模式。这里主要介绍基于LLDP的极简部署方案的工作原理，其流程如下所示:

ATP自动部署流程



极简部署的实现流程为:

1. 设备上电启动

   设备上电启动后，如果设备有配置文件，则以该配置文件正常启动；如果是空配置设备，则进入ATP流程。

2. ATP功能进行初始化，通过LLDP遍历对端端口，获取端口描述。

   如果不能成功获取，60s之后，会重新获取。

3. 成功获取到端口描述后，会对端口描述进行合法性检查。

   - 如果检查通过，将提取扩展信息。
   - 检查检查不通过，将重新获取端口描述，重新执行步骤2。

4. 提取扩展信息后，进行网络连通性检查。
    - 如果检查通过，将获取配置文件。
    - 如果检查不通过，将重新获取端口描述，重新执行步骤2。
5. 获取配置文件后，将自动重启系统。

   如果获取配置文件失败，将将重新获取端口描述，重新执行步骤2。
6. 设备重新启动，ATP完成，设备正常工作。

ATP操作指导

首次上线

在部署ATP之前，首先确定选取哪种部署方式，然后根据不同的部署方式进行如下准备：

- 采用LLDP的极简部署方式
    - 在TOR上联的核心交换机中，选择一台配置aaa@x.x.x.x@y或aaa@x.x.x.x@y.y.y.y扩展信息并开启LLDP功能；
    - 在TFTP服务器上，按照x.x.x.x_yunqi.cfg的形式放置配置文件。
- 采用DHCP的ZTP部署方式
    - 在DHCP服务器上静态配置设备MAC与IP地址的分配关系；
    - 扩展DHCP option支持获取TFTP服务器的IP地址。比如在dhcpd.conf中定义"option tftp-server-address code 150 = ip-address;"，" option tftp-server-address x.x.x.x;"
    - 在TFTP服务器上，按照x.x.x.x_yunqi.cfg放置配置文件。

完成上述准备后，将TOR的上联光纤插好再加电，ATP就会立即开始工作。整体流程为：

1. 自动配置设备IP地址
2. 自动获取初始配置
3. 自动加载配置
4. 重启，设备自动部署成功。

整机替换

交换机进行整机替换时:

- 采用LLDP的极简部署方式
    - 如果是使用同型号机型替换，则不需要更改上述准备，只需要将新的TOR放到架上，插上光纤加电即可.
    - 如果是不同机型，则可能需要更改TFTP server中的配置文件以适配新的型号，然后插上光纤加电即可自动部署。
- 采用DHCP的ZTP部署方式
    - 如果是使用同型号机型替换，需要 **更改DHCP服务器上新设备MAC与IP地址的分配关系** ，然后将新的TOR放到架上，插上光纤加电即可；
    - 如果是不同机型，则还可能需要更改TFTP server中的配置文件以适配新的型号，然后插上光纤加电即可自动部署。

完成上述操作后，TOR会自动获取到最新的配置文件并完成加载。整体流程为：

1. 自动配置设备IP地址
2. 自动获取历史最新配置
3. 自动加载配置
4. 重启，设备自动部署成功。

查看部署状态

在运维模式下执行 **show atp** 命令，查看当前自动部署的状态以及如果没有部署成功时的出错信息：

```
ConnetOS> show atp
State: TFTP_FIN
The configure file is invalid
```

根据ATP的状态，对照下表中的状态以及提示信息帮助定位部署失败的原因。

状态	描述
ADDRESS_COLLECTING	正在搜集管理口IP和TFTP服务器IP。如果地址获取失败： • LLDP方式要检查上联口的端口描述符，配置的格式是否为： – XXXXX@192.168.1.10@100 – XXXXX@192.168.1.10@192.168.1.100 • DHCP的方式则检查： – DHCP服务器是否正确分配IP给管理口。 – 定义option150对应的TFTP服务器地址是否正确。
ADDRESS_VALID	管理口IP和TFTP服务器IP地址校验合法。 如果管理口和TFTP服务器网络不通，检查网络的连通性。
NETWORK_VALID	管理口和TFTP服务器网络连通。下载文件失败，可能是： • TFTP服务器地址错误 • TFTP服务器不存在或存在错误的配置文件名。 文件名要以管理口IP地址加厂商命名，如“192.168.1.33_yunqi.cfg”。
TFTP_FIN	完成配置文件的下载。配置文件不合法，有两个原因： • 下载了空的配置文件 • 配置文件中管理口IP不等于当前设备已配置的管理口IP。
SUCCESS	自动部署成功。 配置文件校验成功并重命名为启机配置文件。重启之后，部署完成。

手动升级配置文件

手动升级配置文件主要指通过TFTP方式将配置文件下载到交换机加载生效。

升级的步骤如下：

1. tftp下载配置文件:

```
ConnetOS> file tftp get remote-file xxx local-file connetos.boot ip-address 192.
↪168.2.2
```

2. 重启系统
3. 启动后，配置文件自动加载:

```
ConnetOS> request system reboot
```

Caution:

- 上述升级命令中xxx为待升级的配置文件。
- 上述升级命令中local-file的名字必须为connetos.boot。
- 执行 **request system reboot** 命令时，需要输入yes进行确认。
- 配置文件升级时，原有配置文件会被覆盖。

配置文件回退

配置回滚

ConnetOS可以基于历史活动配置，通过执行命令 **rollback version-number** 进行回滚操作。ConnetOS最多可以保持49个历史配置，因此可以实现最多49级回滚。

历史活动配置的保存是按照倒序进行的，即序号1保存的是当前配置。

- 用户执行 **rollback 1** 命令可以恢复到上一个活动配置。
- 当用户没有保存当前配置而设备重启时，使用 **rollback 1** 命令可以恢复到重启前的配置:

```
ConnetOS# rollback 1
Rolling back to config: /switch/config/connetos.conf.01
ConnetOS# Waiting for merging configuration.
Load done.
```

Note: rollback后使用“? ”命令可以列出所有的历史存档配置。

CHAPTER 3

配置指南

ConnetOS提供丰富的交换机二三层功能，本章介绍ConnetOS支持的功能特性及如何配置。

Shell介绍

概述

Shell视图简介

ConnetOS系统（以下简称为ConnetOS）为用户提供两个shell视图：

- Linux shell：Linux界面视图，用户初始登录的视图，亦即Debian Linux系统的缺省shell环境。
- CLI shell：命令行接口（Command-line Interface）视图，用于对交换机的网络操作和管理。

用户登录到设备后，输入账户和密码，进入到Linux shell，输入cli，即可进入CLI shell:

```
ConnetOS login: admin
Password:
Last login: Wed Mar 15 16:55:27 CST 2017 from 192.168.1.132 on pts/0
Linux ConnetOS 3.16.7-ckt25+ #1 SMP Thu Mar 2 10:28:57 CST 2017 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
    _______                                __   ____  _____
   / ____/____   ____   ____   ___   / /_ / __ \/ ___/
  / /    / __ \ / __ \ / __ \ / _ \ / __// / / /\__ \
 / /___ / /_/ // / / // / / //  __// /_ / /_/ /___/ /
 \____/ \____//_/ /_//_/ /_/ \___/ \__/ \____//____/
```

```
admin@ConnetOS:~$ cli
Welcome to switch CLI shell.
ConnetOS>
```

命令行视图特点

ConnetOS CLI的命令行有如下特点：

- 简洁的命令行模式 视图只有运维模式（operational mode）和配置模式（configuration mode）两种，命令行操作过程中避免频繁的视图跳转。
- 逻辑清晰的树形结构 命令行按照操作类型，划分为show、set、delete等种类，作为根节点；各类功能（如OSPF、LLDP）作为下一层节点，此功能的相关配置都在此节点下。

在使用命令行时，请注意：

- ConnetOS CLI只有在输入完整命令行时，才能执行命令。
- 命令自动补齐时，如果符合条件的命令行不唯一，会出现无法自动补齐的情况。您可以输入“？”或按Tab查看，手动进行选择、输入。

命令模式

运维模式

登录ConnetOS CLI后会自动进入运维模式，运维模式用于管理和监控设备操作。例如：查看配置信息、查看设备状态、设置系统时间。

运维模式的命令提示符为“>”:

```
ConnetOS>
```

Note: 运维模式下的操作，不需要commit，执行后立即生效。

配置模式

配置模式用于对设备进行各项配置，例如：管理用户、控制设备访问权限、配置各类协议、配置设备安全功能、设置系统硬件属性。

配置模式的命令提示符为“#”，例如:

```
ConnetOS#
```

Note:

- 运维模式下的命令前输入关键字 **run**，即可在配置模式下执行（不包括在运维模式和配置模式下都可以执行的命令）。例如:

  ```
  ConnetOS#run ping 192.168.1.1
  ```

- 配置模式下的任何配置（**set** 或 **delete**）操作，必须执行 **commit** 才会生效。而运维模式下的set操作执行后立即生效。

模式切换

进入配置模式

在运维模式下执行 **configure** 命令，进入配置模式。例如:

```
ConnetOS> configure
Entering configuration mode.
There are no other users in configuration mode.
ConnetOS#
```

如果用户只希望自己在设备上进行配置，可以通过执行 **congfigure exclusive** 命令，将配置模式锁定，避免其他用户的配置干扰。例如:

```
ConnetOS> configure exclusive
Entering configuration mode.
There are no other users in configuration mode.
ConnetOS#
```

Note:

- 如果设备当前有其他用户正在登录，不允许锁定当前配置模式。例如:

```
ConnetOS> configure exclusive
ERROR: Exclusive config mode requested,  but there are already other␣
↪configuration mode
users: admin.
ConnetOS>
```

- 如果要解除锁定，直接退出配置模式即可。

退出配置模式

从配置模式退回到运维模式，可以执行 **exit** 命令或 **quit** 命令。例如:

```
ConnetOS# exit
ConnetOS>
```

或:

```
ConnetOS# quit
ConnetOS>
```

从配置模式退回运维模式时，如果想丢弃当前尚未 **commit** 的配置，有两种方式:

- 执行 **exit** 命令时，使用 **discard** 参数:

```
ConnetOS# exit discard
ConnetOS>
```

- 直接执行 **discard** 命令，再执行 **exit** 命令或 **quit** 命令退出:

```
ConnetOS# discard
ConnetOS# exit
```

```
Leave configuration mode.
ConnetOS>
```

Note: 如果有配置尚未commit，使用 **exit** 命令是无法退出的。

配置方式介绍

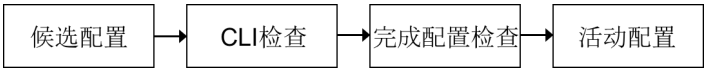
配置类型介绍

ConnetOS中一共包含如下4种配置:

- 缺省配置：ConnetOS首次启动时的默认配置;
- 候选配置：ConnetOS系统只进行了配置，尚未提交的配置；候选配置没有提交前，使用show命令查看，会有“+”的提示。
- 活动配置：候选配置进行提交，系统检查验证通过后，候选配置会转为活动配置。
- 启动配置：ConnetOS系统下次重启时加载的配置。ConnetOS支持对当前的活动配置进行保存。

配置转换介绍

候选配置转变为活动配置过程



设备上电后：

1. 首先加载缺省配置，再比较启动配置和缺省配置的区别，如果检查无误后，将有差异的配置进行加载。
2. 如果用户对配置进行了修改，配置其实并没有立即生效，而是作为候选配置存在于设备上。
3. 当用户对修改内容执行commit命令，ConnetOS系统将验证用户修改的内容，验证通过后，候选配置变为活动配置。
4. 如果用户想下次设备启动时采用当前配置，可以将当前配置保存为启动配置。

保存启动配置

如果要将当前活动配置保存为启动配置，可以使用 **save running-to-startup** 命令保存。当ConnetOS系统重启后，将恢复到重启前的活动配置。

配置回滚

ConnetOS可以基于历史活动配置，通过执行命令 **rollback version-number** 进行回滚操作。ConnetOS最多可以保持49个历史配置，因此可以实现最多49级回滚。

历史活动配置的保存是按照倒序进行的，即序号1保存的是当前配置。

- 用户执行 **rollback 1** 命令可以恢复到上一个活动配置。
- 当用户没有保存当前配置而设备重启时，使用 **rollback 1** 命令可以恢复到重启前的配置:

```
ConnetOS# rollback 1
Rolling back to config: /switch/config/connetos.conf.01
ConnetOS# Waiting for merging configuration.
Load done.
```

Note: rollback后使用“? ”命令可以列出所有的历史存档配置。

配置删除

在ConnetOS中，通过执行 **delete** 命令删除配置或将配置恢复到默认值。例如:

当前系统的hostname为test，删除系统的hostname，使其恢复为缺省的ConnetOS:

```
test# delete system hostname
Deleting:
    hostname: "test"
OK
test# commit
Waiting for merging configuration.
Commit OK.
ConnetOS#
```

Note: 如果要恢复成出厂空配置，可以先后执行 **save default-to-startup** 命令和 **request system reboot** 命令。

配置查询

使用命令行查询配置结果

采用 **show** 命令查看设备上的配置结果。例如查看service节点的配置情况:

```
ConnetOS# show system services
Waiting for building configuration.
    telnet {
        connection-limit: 15
        enable: true
    }
    ssh {
        connection-limit: 15
        enable: true
    }
```

过滤显示信息

ConnetOS CLI支持管道符“|”来过滤显示信息，并支持多级管道。

管道符“|”用来过滤命令行显示信息，帮助用户快速找到所需要的信息。管道符左边的命令输出将作为管道符右边的命令或文件的输入内容。

可以通过“? ”或Tab键查询当前管道支持的参数，例如:

```
ConnetOS# show protocols | ?
Possible completions:
  count                          Count occurrences
  except                         Show only text that does not match a pattern
  find                           Search for the first occurrence of a pattern
  hold                           Hold text without exiting the --More-- prompt
  match                          Show only text that matches a pattern
  no-more                        Don't paginate output
```

more提示信息

ConnetOS CLI下执行 **show** 命令输出的结果超过一屏时，命令行界面有提示“–More–”。 在“–More–”提示符后：

- 输入空格，可以自动下翻一屏。
- 输入回车，可以自动下翻一行。
- 输入“h”，查看显示选项列表。

例如，**show interface brief** 命令显示信息的“–More–”后输入h，可以看到如下选项列表:

```
                  SUMMARY OF MORE COMMANDS

    -- Get Help --
 h                 *  Display this help.

    -- Scroll Down --
 Enter   Return  j   *  Scroll down one line.
 ^M  ^N  DownArrow
 Tab d   ^D  ^X    *  Scroll down one-half screen.
 Space   ^F          *  Scroll down one whole screen.
 ^E  G             *  Scroll down to the bottom of the output.
 N                 *  Display the output all at once instead of one
                      screen at a time. (Same as specifying the
                      | no-more command.)

    -- Scroll Up --
 k   ^H  ^P        *  Display the previous line of output.
 UpArrow
 u   ^U            *  Scroll up one-half screen.
 b   ^B            *  Scroll up one whole screen.
 ^A  g             *  Scroll up to the top of the output.

    -- Misc Commands --
 ^L                *  Redraw the output on the screen.
--More--
```

快捷键

在使用命令行接口时，ConnetOS CLI提供了许多快捷键，用于快速输入命令行，简化操作。

ConnetOS CLI下支持的快捷方式有：

内容	快捷方式
转到下一条命令记录	Down arrow或Ctrl + n
转到上一条命令记录	Up arrow或Ctrl + p
转到行首	Ctrl + a
转到行尾	Ctrl + e
向左移动一个字符	Ctrl + b
向右移动一个字符	Ctrl + f
向前移动一个字	Esc + f
向后移动一个字	Esc + b
删除光标位置上的字符	Ctrl + d
删除光标后的字	Esc + d
删除光标前面的字	Esc + backspace
删除从光标开始至行尾的文本	Ctrl + k
删除行	Ctrl + u
将删除的文本粘贴到光标处	Ctrl + y

获取帮助

命令行自动补齐功能

在设备上进行命令行操作时，必须输入完整的命令行，否则命令行不能执行。

ConnetOS提供命令行自动补齐功能，输入命令时，只需要输入前几位字符，按空格或Tab键，系统会自动进行命令行的补齐。

- 空格键用于补齐大部分CLI命令。
- Tab键既可以用于补齐CLI命令，还可以用于补齐用户自定义的变量，例如ACL的名字，IP地址等。

当待补齐的命令或参数意义模糊时，请在列表中选择可能的补齐内容。例如:

```
ConnetOS> show v?
Possible completions:
  version                        Display system version
  vlan-interface                 Show vlan interface information
  vlans                          Show vlan information
```

系统预设帮助

ConnetOS提供了命令功能解释，在CLI上输入“help？”及具体命令，可以查看命令功能的详细说明。

ConnetOS CLI提供了两种方式的命令行帮助：

- 完整帮助 直接输入“？”，查看当前模式支持的命令及命令的完整格式。
- 部分帮助 输入命令行的部分字符或输入部分命令行，通过不断的“？”获取完整的命令行及命令行释义。

命令行错误信息提示

ConnetOS CLI在命令行输入时会逐字检查语法。当在CLI中输入一个字符串并按下空格键时，如果输入的内容不是命令的有效组成部分，CLI会进行信息提示，同时命令行将无法执行。

常见的提示错误的信息有：

常见提示信息	错误原因
syntax error， command “ci” is not recognized.	输入错误的命令，按“？”寻求帮助。
unknown command.	输入错误的命令，按“Tab”进行命令行补全。
ERROR: path “interface traceoptions fo” is not valid.	输入错误的命令，直接执行。
ERROR: There are uncommitted changes.	退出配置模式时，有未提交的配置。

常用命令行介绍

配置命令关键字

set

set 命令用来进行各项功能的配置。

配置模式下，修改配置后必须commit，修改才能生效。

运维模式下，**set** 命令直接生效。

delete

delete 命令用来删除各类配置。如果要将当前的活动配置全部删除，即清空所有配置恢复成出厂空配置，可以使用 **save default-to-startup** 命令并执行 **request system reboot** 命令进行系统重启。

注意：**delete** 命令会删除配置节点，相关命令下会不显示此功能配置项，需要set后才能显示。

edit

edit 命令用来进入不同层级的视图，只显示本功能相关的命令，同时简化命令行。如果要退出各类edit视图，执行 **top**、**up**、**quit**、**exit** 等命令即可。

运维命令关键字

show

show 命令用来查看设备上的各种配置，输入show后，键入“？”，可以方便地查阅各类设备信息。同时可以运用管道符“|”的过滤功能，进行查看。在显示信息过多，想要中断显示信息的输出时，可以通过输入“q”退回到配置模式。

clear

clear 命令用来清除各项统计信息。

run

运维模式下的命令，增加 **run** 前缀可以用在配置模式下。

Note: 在运维模式和配置模式下都能执行的命令，不支持运行**run**命令。

DevOps

ConnetOS提供原生Linux服务器上的体验，开放的技术架构和运行环境，是DevOps实践的首选。

- 支持丰富的远程控制设备的方式，提供RCC、XRL、RESTful API、Python SDK等能力；
- 支持apt/dpkg软件包管理方式，支持第三方软件（Puppet、Zabbix、Ansible等）的直接部署。

RCC远程配置

RCC（Remote Config Client）远程CLI调用，通过此功能，用户在本地就可以实现对交换机的远程配置。
RCC的使用极其简单，用户在使用时不需要进行任何配置，在本地编译之后，即可直接使用。
RCC有如下两种方式执行命令:

- 直接输入命令行。

  采用“-c”，输入命令行字符串，命令以“;“隔开。例如:

```
[yunqi@host-a remote_cli_client]$./RCC -s 192.168.1.31 -u admin -p admin -c
↪"configure;
show vlans;exit;exit"
```

- 输入命令行文件名。

  将要执行的命令保存在文件中，以回车隔开。输入”-f 命令行文件名”，例如:

```
[yunqi@host-a remote_cli_client]$./RCC -s 192.168.1.31 -u admin -p admin -f a.txt
```

RCC可执行如下操作:

```
[yunqi@host-a remote_cli_client]$ ./RCC
Usage: RCC -s < device addr > -u < username> -p < password > [-c < command >] [-f <␣
↪commands
file >] [-h]

           -s < device addr >            : device addr
           -u < username >               : the user name to login in
           -p < password >               : the password for the given user
           -c < commands >               : the command string containing multiple␣
↪commands to be executed
           -f < commands file >          : the file name of the excecuting commands
           -h                            : usage (this message)
```

项目	含义
-s	需要远程控制的设备IP地址
-u	可登录的用户名
-p	用户密码
-c	命令行。以“;”隔开
-f	文件名。要执行的命令列表，在文件中以回车隔开。

操作举例:

```
[yunqi@host-a remote_cli_client]$ ./RCC -s 192.168.1.31 -u admin -p admin -c "show␣
↪version"
Try to connect remote CLI 192.168.1.31:8080...
Connected successfully.
```

```
Doing user authentication...
Sending username...
From remote CLI: Begin to authenticate

Sending password...
From remote CLI: User authentication OK

Sending command: show version
...
From remote CLI:
Copyright (C) 2015-2017 YUNQI TECH, Inc.
PN              : C1020
OS              : ConnetOS GENERAL
Version ID      : 2.1.2
Build Code      : r2146 (13V21)
Switch MAC      : 00:03:0f:64:da:5f
Management MAC : 00:03:0f:64:da:60
Release Time    : 2017-03-21 11:41:52
Operational mode CLI OK.
```

RESTful API

用户登录管理

初次登录设备

登录方式简介

用户可以通过Console口、Telnet、SSH等方式登录交换机，但是当用户需要为第一次上电的设备进行配置时，必须使用Console口登录。

- Console口进行本地登录是最基本的登录方式，也是配置其他登录方式的基础。
- 用户通过SSH／Telnet远程登录到交换机上，对交换机进行远程管理和维护。

通过Console口登录设备

在配置通过Console口登录设备之前，需要准备好：

- Console口通信线缆
- PC端仿真软件

通过Console口登录时：

1. 首先用通信线缆把PC机和交换机连接起来；
2. 然后在PC机上运行终端仿真程序（如Windows的超级终端），选择与交换机相连的串口，设置终端通信参数。

   用户终端的参数配置必须和交换机Console口的缺省配置保持一致，用户才能通过Console口登录到交换机上。

ConnetOS上，Console口的缺省配置如下：

参数	缺省值
传输速率	115200 bit/s
流控方式	不进行流控
奇偶校验	不进行校验
停止位	1
数据位	8

管理口配置

配置管理口

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置管理口的IP地址。

   ConnetOS# **set interface management-ethernet eth0 address** *ip-address*

   缺省情况下，管理口的IP地址为0.0.0.0/0。

3. （可选）开启管理口的DHCP功能。

   ConnetOS# **set interface management-ethernet eth0 dhcp enable** { **false** | **true** }

   缺省情况下，管理口的DHCP功能是开启的。

4. 提交配置。

   ConnetOS# **commit**

检查配置结果

执行 **show interface management-ethernet eth0** 命令查看管理口的详细信息:

```
ConnetOS> show interface management-ethernet eth0
eth0     Inet addr: 192.168.30.24/24,  Gateway: 0.0.0.0,  HW address: 08:9e:01:88:51:ba
```

带内访问配置

ConnetOS系统除了支持从管理口登录外，还支持从三层接口进行登录，使用三层接口的IP地址当作管理口IP使用。

该功能在ConnetOS系统下缺省是开启的。

配置带内访问

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置带内访问功能是否开启。

   ConnetOS# **set system inband enable** { **false** | **true** }

3. 提交配置。

   ConnetOS# **commit**

配置用户登录方式

通过Console口登录

直接用Console口通信线缆连接，进行登录。

通过SSH登录

SSH（Secure Shell）安全外壳，是一个网络安全协议，标准协议端口号22。当用户通过一个不能保证安全的网络环境远程登录到交换机时，SSH通过对网络数据的的加密和认证，保证交换机不受IP地址欺骗、明文密码截取等恶意攻击。

用户通过SSH登录到交换机上，对交换机进行远程管理和维护。

1. 进入配置模式。

   ConnetOS> **configure**

2. （可选）使能SSH服务功能。

   ConnetOS# set system services ssh enable { false | true }

   缺省情况下，SSH服务功能是开启的。

3. （可选）配置允许SSH登录的最大连接数。

   ConnetOS# set system services ssh connection-limit limit-number

   缺省情况下，允许SSH登录的最大连接数是15。

4. 提交配置。

   ConnetOS# **commit**

通过Telnet登录

1. 进入配置模式。

   ConnetOS> **configure**

2. （可选）使能Telnet服务功能。

   ConnetOS# **set system services telnet enable** { **false** | **true** }

   缺省情况下，Telnet服务功能是开启的。

3. （可选）配置Telnet登录的最大连接数。

   ConnetOS# **set system services telnet connection-limit** *limit-number*

   缺省情况下，允许Telnet登录的最大连接数是15。

4. 提交配置

   ConnetOS# **commit**

用户账户管理

用户账号介绍

用户在Linux shell和CLI shell下都可以创建账户，账户类型分别为：

- Linux shell账户：
  - 特权账户：在设备上可以进行任何操作。缺省账户：root。
  - 非特权账户：具有有限的操作权限。缺省并没有创建此类账户，用户可以根据需要自行创建。
- CLI shell账户：
  - read-only：只能对设备进行查询操作。
  - super-user：可以对设备进行查询和配置操作。缺省可使用账号root。

如果想要Linux shell下创建的账号也能够正常使用CLI，有两种方式：

- 手工配置：首先在Linux shell下创建账号，其次根据想赋予的权限，决定是否将账号加入xorp组（加入xorp具有super-user权限）。
- 自动配置：直接在CLI shell下对账号进行权限设置。

配置local账户

ConnetOS的local账户要可用，必须同时满足如下两个条件：

- local账号启用。
- AAA可用，或开启了AAA但AAA服务器不可达。

在local账号启用的情况下，如果配置了AAA且可用，local账号会自动禁用，如果AAA失效，那local账号会自动恢复为可用。

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置是否禁用local帐号。

   ConnetOS# **set system aaa local enable** { **false** | **true** }

   缺省情况下，local帐号是开启的。

3. 创建local账户。

   ConnetOS# **set system login user** *user-name* **authentication plain-text-password** *plain-text-password*

   缺省情况下，创建local账户时，账户类型是super-user。

4. 设置账户类型

   ConnetOS# **set system login user** *user-name* **class** { **read-only** | **super-user** }

5. 提交配置。

   ConnetOS# **commit**

配置用户登录权限

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置允许登录设备的网段。

   ConnetOS# **set system login-acl network** *network-ip-address*

   缺省情况下，允许所有的网段登录设备。

3. 提交配置。

   ConnetOS# **commit**

配置用户认证方式

AAA介绍

AAA是Authentication（认证）、Authorization（授权）和Accounting（计费）的简称它提供对用户进行认证、授权和计费三种安全功能:

- 认证（Authentication）：验证用户是否可以获得访问权，确定哪些用户可以访问网络。
- 授权（Authorization）：授权用户可以使用哪些服务。
- 计费（Accounting）：记录用户使用网络资源的情况。

ConnetOS支持完整的认证、授权和计费功能。

配置TACACS方式进行认证、授权和计费

1. 进入配置模式。

   ConnetOS> **configure**

2. 指定TACACS服务器的地址。

   ConnetOS# **set system aaa tacacs-plus host server-ip** *ip-address*

3. 使能TACACS功能

   ConnetOS# **set system aaa tacacs-plus enable** { **false** | **true** }

   缺省情况下，TACACS功能没有使能。使能后，TACACS功能直接生效。

4. （可选）配置TACACS服务器共享密钥

   ConnetOS# **set system aaa tacacs-plus key** *shared-key*

   缺省情况下，TACACS服务器的共享密钥是keystring。

5. （可选）配置TACACS的认证类型

   ConnetOS# **set system aaa tacacs-plus auth-type** { **ascii** | **chap** | **pap** }

   缺省情况下，TACACS的认证类型是ASCII。

6. （可选）使能TACACS授权功能

   ConnetOS# **set system aaa tacacs-plus authorization enable** { **false** | **true** }

   缺省情况下，TACACS授权功能是使能的。

7. （可选）使能TACACS计费功能

   ConnetOS# **set system aaa tacacs-plus accounting enable** { **false** | **true** }

   缺省情况下，TACACS计费功能是使能的。

8. 提交配置。

   ConnetOS# **commit**

设备管理配置

配置设备的基本信息

1. 进入配置模式。

   ConnetOS> **configure**

2. 设置设备设备名称。

   - 单机模式：**set system hostname** *hostname*
   - 堆叠模式：**set iss member** *member-id* **hostname** *hostname*

3. 设置设备欢迎语。

   ConnetOS# **set system login announcement** *announcement-message*

4. 提交配置

   ConnetOS# **commit**

配置系统时间

系统时间概述

NTP（Network Time Protocol）网络时间协议是由RFC 1305定义的时间同步协议。NTP用于分布式时间服务器与客户端之间的时间同步，使网络内所有设备的时钟保持一致，从而使设备能够提供基于统一时间的多种应用。NTP报文通过UDP传输，端口号是123。

对于运行NTP的本地系统，既可以接受来自其他时钟源的同步，又可以作为时钟源同步其他的时钟，并且可以和其他设备互相同步。

配置时区

1. 进入配置模式。

   ConnetOS> **configure**

2. 修改系统时区

   ConnetOS# **set system timezone** *timezone*

   缺省情况下，系统时区为UTC。

3. 提交配置。

   ConnetOS# **commit**

配置NTP

1. 进入配置模式。

   ConnetOS> **configure**

2. 指定NTP服务器的地址。

   ConnetOS# **set system ntp-server-ip** *ip-address*

3. 提交配置。

   ConnetOS# **commit**

修改本地时间

1. 修改系统时间。

   ConnetOS> **set date time**

系统信息查询

查询版本信息

查看当前版本信息:

```
ConnetOS 1> show version
Copyright (C) 2015-2017 YUNQI TECH, Inc.
PN              : C1020
OS              : ConnetOS GENERAL
Version ID      : 2.1.1
Build Code      : r2078 (12Y17)
Switch MAC      : 00:03:0f:64:da:4d
Management MAC : 00:03:0f:64:da:4e
Release Time    : 2017-03-17 11:35:05
```

查询电源信息

ConnetOS支持对电源模块的SN号、温度、风扇转速、电压、电流和功率进行查询，并能对电源模块的状态进行监控展示。当前，定义了4种状态:

- Unpresent，表示电源模块不在位。
- No Power，表示电源模块在位，但是没有AC/DC输入。
- Error，表示电源模块在位，有AC/DC输入，但模块工作异常。
- OK，表示电源模块正常工作。

查看当前的电源信息:

```
ConnetOS> show system rpsu
RPSU 1:
    Module Status      : OK
    Serial Number      : SA020T051623000134
    Temperature        : 36     Centigrade
    IIN                : 0.44   A
    VIN                : 221.00 V
```

```
    PIN                 : 95.00  W
    IOUT                : 7.59   A
    VOUT                : 11.94  V
    POUT                : 75.00  W
RPSU 2:
    Module Status       : No Power
    Serial Number       : SA020T051623000142
    Temperature         : 42     Centigrade
    IIN                 : 0.00   A
    VIN                 : 0.00   V
    PIN                 : 0.00   W
    IOUT                : 0.00   A
    VOUT                : 0.00   V
    POUT                : 0.00   W
```

Note: 由于rpsu report精度原因，查询出来的参数可能和实际有些出入。

查询风扇信息

ConnetOS支持对风扇的转速和PWM值进行查询:

```
ConnetOS> show system fan
Fan Status:
    Fan 1 : speed =  8850 RPM, PWM =  40%
    Fan 2 : speed =  8700 RPM, PWM =  40%
    Fan 3 : speed =  8850 RPM, PWM =  40%
    Fan 4 : speed =  8775 RPM, PWM =  40%
    Fan 5 : speed =  8850 RPM, PWM =  40%
```

查询系统序列号

ConnetOS支持对系统的SN进行查询，包括主板SN、电源模块SN以及光模块信息。

查询产品序列号:

```
ConnetOS> show system serial-number
MotherBoard Serial Number : 1626000404
RPSU 1 Serial Number      : SA020T051623000173
RPSU 2 Serial Number      : SA020T051623000176
SFP+ te-1/1/2
    Vendor Name           : FINISAR CORP.
    Serial Number         : MUD1KHT
    Product Number        : FTLX8571D3BCL
    Module Type           : SR/850nm
    Cable Length          : 300.0m
SFP+ te-1/1/5
    Vendor Name           : FINISAR CORP.
    Serial Number         : MUG11SG
    Product Number        : FTLX8571D3BCL
    Module Type           : SR/850nm
    Cable Length          : 300.0m
SFP+ te-1/1/6
    Vendor Name           : YUNQI
```

```
    Serial Number          : RD161100070098
    Product Number         : RTXM228-551
    Module Type            : SR/850nm
    Cable Length           : 300.0m
SFP+ te-1/1/48
    Vendor Name            : YUNQI
    Serial Number          : BP162201790061
    Product Number         : RTXM228-551
    Module Type            : SR/850nm
    Cable Length           : 300.0m
QSFP+ qe-1/1/49
    Vendor Name            : FINISAR CORP
    Serial Number          : XUC065G
    Product Number         : FTL410QE2C
    Module Type            : SR4/850nm
    Cable Length           : 100.0m
```

查询光模块DDM信息

DDM（Digital Diagnostic Monitoring）数字诊断监控功能，可以监测模块温度、电压、偏置电流、收发光功率等，上述监控参数经过A/D转换后，会被写入模块内部的EEPROM，此部分内容由SFF-8472进行定义。

ConnetOS支持对光模块的DDM信息进行查询展示:

```
ConnetOS> show system ddm
Interface   Temp(C)  Voltage(V)  Bias(mA)  Tx Power(dBm)  Rx Power(dBm)  Module Type
----------  -------  ----------  --------  -------------  -------------  -----------
te-1/1/6    36.24    3.39        8.10      -1.94          -2.80          SR/850nm
te-1/1/7    28.45    3.35        5.14      -2.32          -2.75          SR/850nm
te-1/1/34   32.33    3.37        8.09      -2.15          -2.24          SR/850nm
qe-1/1/52   35.78    3.29        5.83      -3.61          -2.52          SR4/850nm
qe-1/1/54   33.78    3.33        NA        NA             NA             SR4/850nm
```

NMS管理设备

NMS管理设备

ConnetOS支持用户通过NMS（Network Management Station，网管工作站）登录到交换机，对交换机进行配置、管理。

如果要使用NMS登录设备，需要先在设备上配置好SNMP功能。配置完成后，即可通过登录。

SNMP概述

SNMP（Simple Network Management Protocol）简单网络管理协议，是网络中管理设备和被管理设备之间的通信规则，它定义了一系列消息、方法和语法，用于实现管理设备对被管理设备的访问和管理。

SNMP具有以下优势:

- 自动化网络管理。 网络管理员可以利用SNMP平台在网络上的节点检索信息、修改信息、发现故障、完成故障诊断、进行容量规划和生成报告。

- 屏蔽不同设备的物理差异，实现对不同厂商产品的自动化管理。 SNMP只提供最基本的功能集，使得管理任务分别与被管设备的物理特性和下层的联网技术相对独立，从而实现对不同厂商设备的管理，特别适合在小型、快速和低成本的环境中使用。

配置SNMP

1. 进入配置模式。

   ConnetOS> **configure**

2. 启动SNMP功能。

   ConnetOS# **set protocols snmp community** *community-info* [**authorization read-only** | **clients** *ip-address*]

   缺省情况下，SNMP功能是关闭的。

3. 配置SNMP访问控制

   ConnetOS# **set system snmp-acl network** *ip-address*

   缺省情况下，允许所有网段查询。

4. 提交配置

   ConnetOS# **commit**

ConnetOS常用OID

ConnetOS常用OID如下表所示。

名称	OID	描述
设备主机名	1.3.6.1.2.1.1.5	sysname
系统基本信息	1.3.6.1.2.1.1.1	sysdesc
系统启动时间	1.3.6.1.2.1.1.3	sysuptime
端口名称	1.3.6.1.2.1.31.1.1.1.18	ifalias
端口状态	1.3.6.1.2.1.2.2.1.8	ifoperstatus
端口速率	1.3.6.1.2.1.31.1.1.1.15	ifhighspeed
端口描述	1.3.6.1.2.1.2.2.1.2	ifDescr
端口in流量	1.3.6.1.2.1.31.1.1.1.6	ifHCInOctets
端口out流量	1.3.6.1.2.1.31.1.1.1.10	ifHCOutOctets
端口in error包	1.3.6.1.2.1.2.2.1.14	ifInErrors
端口out error包	1.3.6.1.2.1.2.2.1.20	ifOutErrors
MAC	1.3.6.1.2.1.17.7.1.2	
VLAN	1.3.6.1.2.1.17.7.1.4.2.1.3	
VLAN内成员端口	1.3.6.1.2.1.17.7.1.4.2.1.4	
arp信息	1.3.6.1.2.1.4.22.1.2	ipNetToMediaPhysAddress
三层接口IP	1.3.6.1.2.1.4.20.1.2	ipAdEntIfIndex
三层接口掩码	1.3.6.1.2.1.4.20.1.3	ipAdEntNetMask
ECMP路由	1.3.6.1.2.1.4.24.4.1.1	
指定IP的ECMP路由	1.3.6.1.2.1.4.24.4.1.1 x.x.x.x	
OSPF router id	1.3.6.1.2.1.14.1.1.0 1.3.6.1.2.1.14.8(查看总的信息)	ospfRouterId
OSPF Area	1.3.6.1.2.1.14.7.1.3 1.3.6.1.2.1.14.2.1.1	ospfifAreaId
OSPF	cost	1.3.6.1.2.1.14.8.1.4

Continued on next page

Table 3.1 – continued from previous page

名称	OID	描述
BGP标识	1.3.6.1.2.1.15.4	
BGP对端状态	1.3.6.1.2.1.15.3.1.2	
BGP设备所在AS	1.3.6.1.2.1.15.2	
BGP连接的本地IP	1.3.6.1.2.1.15.3.1.5	
CPU占用率	1.3.6.1.4.1.44781.1.1	
总内存	1.3.6.1.4.1.44781.1.2	
已使用内存	1.3.6.1.4.1.44781.1.3	
剩余可用内存	1.3.6.1.4.1.44781.1.4	
温度	1.3.6.1.4.1.44781.1.5	
光模块厂家	1.3.6.1.4.1.44781.1.6.1.2	
光模块SN	1.3.6.1.4.1.44781.1.6.1.3	
光模块类型	1.3.6.1.4.1.44781.1.6.1.9	
端口收光	1.3.6.1.4.1.44781.1.6.1.8	
端口发光	1.3.6.1.4.1.44781.1.6.1.7	
全部SN	1.3.6.1.2.1.47.1.1.1.1.11	
主板SN	1.3.6.1.2.1.47.1.1.1.1.11.256	
电源SN	1.3.6.1.2.1.47.1.1.1.1.11.257 1.3.6.1.2.1.47.1.1.1.1.11.258	
电源状态	1.3.6.1.4.1.44781.1.7.1.2	
AAA server IP	1.3.6.1.4.1.44781.1.8.1.2	
DHCP relay接口	1.3.6.1.4.1.44781.1.9.1.1	
DHCP relay server IP	1.3.6.1.4.1.44781.1.9.1.2	
Syslog server IP	1.3.6.1.4.1.44781.1.10.1.2	
NTP Server IP	1.3.6.1.4.1.44781.1.11.1.0	
NTP 同步状态	1.3.6.1.4.1.44781.1.11.2.0	

接口配置

接口简介

接口是设备与网络中的其它设备交换数据并相互作用的部件，分为管理接口、物理业务接口和逻辑接口三类，其中：

- 管理接口

  管理接口主要为用户提供配置管理支持，也就是用户通过此类接口可以登录到设备，并进行配置和管理操作。管理接口不承担业务传输。

- 物理接口

  物理接口是真实存在、有器件支持的接口，承担业务传输。

- 逻辑接口

  逻辑接口是指能够实现数据交换功能但物理上不存在、需要通过配置建立的接口。逻辑接口需要承担业务传输。

配置以太接口

缺省情况下，接口的基本参数都有缺省值，请根据具体网络情况进行选择。

1. 进入配置模式。

   ConnetOS> **configure**

2. （可选）配置接口功能是否使能。

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **enable** { **false** | **true** }

   缺省情况下，接口功能是使能的。

3. 可选）配置接口描述。

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **description** *description*

   缺省情况下，接口下没有接口描述。

4. 配置接口模式

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **family ethernet-switching port-mode** { **access** | **trunk** }

   - Trunk类型的接口主要用来连接其它交换机设备，一般用于干道链路。Trunk接口允许多个VLAN的帧通过。
   - Access类型的接口主要用来连接用户主机，一般用于接入链路，且接入链路上通过的帧为不带Tag的以太网帧。如果Access接口配置了缺省VLAN，则在该报文上加上Tag标记，并将Tag中的VID字段的值设置为接口所属的缺省VLAN编号，此时接入链路上允许与缺省VLAN Tag匹配的以太网帧通过。

5. 配置接口MTU

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **mtu** *mtu-value*

6. 配置接口限速

   ConnetOS# **set interface** *interface-name* **rate-limiting egress kilobits** *rate-limit*

7. 配置接口速率

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **speed** *speed-vlaue*

8. 配置端口风暴控制

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **storm-control** { **broadcast** | **multicast** | **unicast** } **kilobits** *suppress*

   ConnetOS支持物理接口单播，组播和广播的风暴抑制，单位是bps，范围1～40000000。

9. 提交配置

   ConnetOS# **commit**

loopback接口配置

简介

Loopback是一个应用广泛的逻辑接口。逻辑接口是指能够实现数据交换功能，但是物理上不存在、需要通过配置建立的接口。Loopback接口一旦被创建，其物理状态和链路协议状态永远是Up，即使该接口上没有配置IP地址。

Loopback接口可以提高设备的可靠性：

- 作为设备的管理地址。

  系统管理员完成网络规划之后，为了方便管理，会为每一台设备创建一个loopback 接口，并在该接口上单独指定一个IP地址作为管理地址，管理员会使用该地址进行远程登录（telnet），该地址实际上起到了类似设备名称一类的功能。

- 使用该接口地址作为动态路由协议OSPF 、BGP的router id。
- 使用该接口地址作为BGP 建立TCP连接的源地址。

配置loopback接口

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置Loopback接口的IP地址和掩码

   ConnetOS# **set loopback-interface interface** *lo-interface-name* **address** *ip-address* [**member** *member-id*]

   ConnetOS支持配置多个loopback接口。

3. 配置Loopback接口的接口描述

   ConnetOS# **set loopback-interface interface** *lo-interface-name* **description** *description*

4. 提交配置

   ConnetOS# **commit**

配置汇聚接口

接口汇聚是将多个接口聚合在一起形成1个汇聚组，以实现流量在成员接口中的分担，同时提供更高的连接可靠性。

ConnetOS支持通过手动、静态LACP、动态LACP配置汇聚接口。同一个汇聚组中端口的基本配置应该保持一致，即：

- 如果某端口为trunk端口，则其他端口也配置为trunk端口；
- 如该端口的链路类型改为access端口，则其他端口的链路类型也改为access端口。

汇聚接口的配置，请参见”以太网交换配置－链路聚合配置”。

配置VLAN接口

1. 进入配置模式。

   ConnetOS> **configure**

2. 创建VLAN

   ConnetOS# **set vlans vlan-id** *vlan-id*

3. 创建VLAN对应的接口vlan-interface

   ConnetOS# **set vlans vlan-id** *vlan-id* **l3-interface** *l3-interface-name*

4. 配置vlan-interface的地址及掩码

   ConnetOS# **set vlan-interface interface** *l3-interface-name* **address** *ip-address* **prefix-length** *prefix-length*

   ConnetOS支持一个VLAN配置多个IP地址。

5. 为vlan-interface指定DHCP Relay服务器。

   ConnetOS# **set vlan-interface interface** *l3-interface-name* **dhcp-relay server-ip** *ip-address*

6. 配置vlan-interface的MTU

   ConnetOS# **set vlan-interface interface** *l3-interface-name* **mtu** *mtu-vlaue*

7. （可选）配置vlan-interface的接口描述

   ConnetOS# **set vlan-interface interface** *l3-interface-name* **description** *description*

8. 提交配置

   ConnetOS# **commit**

以太网交换配置

转发模式配置

转发模式简介

ConnetOS支持两种二层转发模式：

- 直通模式：ConnetOS收完基本的用于二层转发的头部信息后就进行转发，不对数据包做帧校验，时延小。
- 存储转发模式：ConnetOS把完整的数据包收完才进行转发，会对数据包做帧校验。

因此，直通模式主要应用于网络环境较好的情况下，可以做到比存储转发的时延小。但我们线上的应用业务对时延的敏感度没有达到这个级别，因此建议使用存储转发模式。

配置转发模式

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置转发模式。

   ConnetOS# **set forwarding-options forwarding-mode** { **cut-through** | **store-and-forwarding** }

   缺省情况下，ConnetOS使用存储转发。

3. 提交配置

   ConnetOS# **commit**

MAC地址表配置

MAC地址表简介

MAC地址

MAC（Media Access Control）地址用来定义网络设备的位置。MAC地址由48比特长、12位的16进制数字组成，其中从左到右开始，0到23bit是厂商向IETF等机构申请用来标识厂商的代码，24到47bit由厂商自行分派，是各个厂商制造的所有网卡的一个唯一编号。

MAC地址可以分为3种类型：

- 物理MAC地址：这种类型的MAC地址唯一的标识了以太网上的一个终端，该地址为全球唯一的硬件地址；
- 广播MAC地址：全1的MAC地址为广播地址（FF-FF-FF-FF-FF-FF），用来表示LAN上的所有终端设备；
- 组播MAC地址：除广播地址外，第8bit为1的MAC地址为组播MAC地址（例如01-00-00-00-00-00），用来代表LAN上的一组终端。

ConnetOS支持的MAC地址格式是12位的十六进制数，用“:”分隔。比如：00:22:22:22:22:22。

MAC地址表

MAC地址表记录了目的MAC 地址、MAC地址对应的出接口以及所属的VLAN ID，用于指导报文进行单播转发。在转发数据时， 设备根据报文中的目的 MAC 地址查询 MAC 地址表，快速定位出接口，从而减少广播。

MAC 地址表项的生成方式有两种：

- 自动生成：设备根据收到的数据帧里的源MAC地址自动学习而建立。

  如果MAC地址表中不存在该MAC地址表项，设备则将这个新MAC地址以及该MAC地址对应的PortA和VLAN ID作为一个新的表项加入到MAC地址表中。

- 手工配置：手工在MAC地址表中加入特定MAC地址表项，将设备与接口绑定。

  MAC格式为12位的16进制数，用“：”分隔，同时需要设置MAC对应的VLAN。例如：配置VLAN100下的te-1/1/1口静态MAC地址为00:22:22:22:22:22。

为适应网络的变化，MAC地址表需要不断更新。MAC表中自动生成的表项（即动态表项）并非永远有效，每一条表项都有一个生存周期，到达生存周期仍得不到更新的表项将被删除，这个生存周期被称作老化时间。如果在到达生存周期前记录被更新，则该表项的老化时间重新计算。

配置静态MAC地址表

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置静态MAC地址表

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **static-mac-address** *mac-address* **vlan** *vlan-id*

3. 提交配置

   ConnetOS# **commit**

配置MAC地址表老化时间

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置MAC地址的老化时间。

   ConnetOS# **set forwarding-options mac-aging-time** *aging-time*

   缺省情况下，MAC地址表老化时间是300s。

3. 提交配置

   ConnetOS# **commit**

清空MAC地址表

1. 清空指定接口MAC表。

   ConnetOS# **run clear ethernet-switching table** *interface-name*

2. 清空全部MAC表

   ConnetOS# **run clear ethernet-switching table all**

VLAN配置

简介

VLAN概述

VLAN（Virtual Local Area Network）虚拟局域网，是将一个物理的LAN在逻辑上划分成多个广播域的通信技术。VLAN内的主机间可以直接通信，而VLAN间不能直接互通，从而将广播报文限制在一个VLAN内。

VLAN有如下优势：

- 限制广播域：广播域被限制在一个VLAN内，节省了带宽，提高了网络处理能力。
- 增强局域网的安全性：不同VLAN内的报文在传输时是相互隔离的。
- 提高了网络的健壮性：故障被限制在一个VLAN内，本VLAN内的故障不会影响其他VLAN的正常工作。
- 灵活构建虚拟工作组：用VLAN可以划分不同的用户到不同的工作组，同一工作组的用户也不必局限于某一固定的物理范围，网络构建和维护更方便灵活。

常见概念

VLAN的帧格式

IEEE 802.1Q是虚拟桥接局域网的正式标准，对Ethernet帧格式进行了修改，在源MAC地址字段和协议类型字段之间加入4字节的802.1Q Tag。每台支持802.1Q协议的交换机发送的数据包都会包含VLAN ID，以指明交换机属于哪一个VLAN。

在一个VLAN交换网络中，以太网帧有以下两种形式：

- 有标记帧（tagged frame）：加入了4字节802.1Q Tag的帧
- 无标记帧（untagged frame）：原始的、未加入4字节802.1Q Tag的帧

链路类型

VLAN中有以下两种链路类型：

- 接入链路（Access Link）：用于连接用户主机和交换机的链路。通常情况下，主机并不需要知道自己属于哪个VLAN，主机硬件通常也不能识别带有VLAN标记的帧。因此，主机发送和接收的帧都是untagged帧。

- 干道链路（Trunk Link）：用于交换机间的互连或交换机与路由器之间的连接。干道链路可以承载多个不同VLAN数据，数据帧在干道链路传输时，干道链路的两端设备需要能够识别数据帧属于哪个VLAN，所以在干道链路上传输的帧都是Tagged帧。

接口类型

ConnetOS支持两种类型的接口：

- Access接口：交换机上用来连接用户主机的接口，它只能连接接入链路。仅仅允许唯一的VLAN ID通过本接口，这个VLAN ID与接口的缺省VLAN ID相同，Access接口发往对端设备的以太网帧永远是不带标签的帧。
- Trunk接口：是交换机上用来和其他交换机连接的接口，它只能连接干道链路，允许多个VLAN的帧（带Tag标记）通过。

VLAN接口

不同VLAN间的主机不能直接通信，通过在设备上配置VLAN接口，可以实现 VLAN 间的三层互通。VLANIF 接口是一种三层逻辑接口，每个VLAN 对应一个 VLANIF。在为VLANIF接口配置了 IP 地址后，该IP 地址即可作为本 VLAN 内网络设备的网关地址，对需要跨网段的报文进行基于IP地址的三层转发。

缺省VLAN

ConnetOS除了可以设置接口允许通过的VLAN，还可以设置接口的缺省VLAN，即 PVID（Port VLAN ID，native-vlan-id）。

在缺省情况下，所有端口的缺省VLAN均为1。用户可以根据需要进行配置。

- Access类型接口的缺省VLAN就是它所在的VLAN。
- Trunk类型接口可以允许多个 VLAN 通过，能够配置缺省 VLAN。

配置VLAN的基本功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 创建VLAN。

   ConnetOS# **set vlans vlan-id** *vlan-id*

3. （可选）配置VLAN的名称。

   ConnetOS# **set vlans vlan-id** *vlan-id* **vlan-name** *vlan-name*

4. （可选）为指定VLAN配置描述信息

   ConnetOS# **set vlans vlan-id** *vlan-id* **description** *description*

5. 提交配置。

   ConnetOS# **commit**

配置基于接口的VLAN接口

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置接口类型。

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **family ethernet-switching port-mode** { **access** | **trunk** }

   Access模式下，一个接口只能属于一个VLAN，即Native VLAN。

   trunk模式下，可以设置一个接口属于多个VLAN。多个VLAN包括缺省VLAN和其他VLAN。

3. 配置接口的缺省VLAN 。

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **family ethernet-switching native-vlan-id** *vlan-id*

   缺省情况下，所有接口的native-vlan-id都为1。

4. 将接口加入VLAN ID。

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **family ethernet-switching vlan members** *vlan-member&<1-n>*

5. 提交配置。

   ConnetOS# **commit**

查看VLAN

在配置模式下，执行 **show vlans** 命令，查看VLAN的配置信息:

```
ConnetOS # show vlans
Waiting for building configuration.
    vlan-id 1 {
        description: ""
        vlan-name: "default"
        l3-interface: ""
    }
    vlan-id 10 {
        description: ""
        vlan-name: "default"
        l3-interface: "vlan10"
    }
```

在运维模式下，执行 **show vlans** 命令，查看VLAN信息:

```
ConnetOS > show vlans
Vlan ID    Tag         Interfaces
-------    --------    ----------------------------------------------------
1          tagged
           untagged    te-1/1/1,   te-1/1/2,   te-1/1/3,   te-1/1/4,   te-1/1/5,
                       te-1/1/6,   te-1/1/7,   te-1/1/8,   te-1/1/9,   te-1/1/10,
                       te-1/1/11,  te-1/1/12,  te-1/1/13,  te-1/1/14,  te-1/1/16,
                       te-1/1/17,  te-1/1/18,  te-1/1/19,  te-1/1/20,  te-1/1/21,
                       te-1/1/22,  te-1/1/23,  te-1/1/24,  te-1/1/25,  te-1/1/26,
                       te-1/1/27,  te-1/1/28,  te-1/1/29,  te-1/1/30,  te-1/1/31,
                       te-1/1/32,  te-1/1/33,  te-1/1/34,  te-1/1/35,  te-1/1/36,
```

```
                    te-1/1/37,  te-1/1/38,  te-1/1/39,  te-1/1/40,  te-1/1/41,
                    te-1/1/42,  te-1/1/43,  te-1/1/44,  te-1/1/45,  te-1/1/46,
                    te-1/1/47,  te-1/1/48,  qe-1/1/49,  qe-1/1/50,  qe-1/1/51,
                    qe-1/1/52,  qe-1/1/53,  qe-1/1/54,  te-2/1/1,   te-2/1/2,
                    te-2/1/3,   te-2/1/4,   te-2/1/5,   te-2/1/6,   te-2/1/7,
                    te-2/1/8,   te-2/1/9,   te-2/1/10,  te-2/1/11,  te-2/1/12,
                    te-2/1/13,  te-2/1/14,  te-2/1/15,  te-2/1/16,  te-2/1/17,
                    te-2/1/18,  te-2/1/19,  te-2/1/20,  te-2/1/21,  te-2/1/22,
                    te-2/1/23,  te-2/1/24,  te-2/1/25,  te-2/1/26,  te-2/1/27,
                    te-2/1/28,  te-2/1/29,  te-2/1/30,  te-2/1/32,  te-2/1/33,
                    te-2/1/34,  te-2/1/35,  te-2/1/36,  te-2/1/37,  te-2/1/38,
                    te-2/1/39,  te-2/1/40,  te-2/1/41,  te-2/1/42,  te-2/1/43,
                    te-2/1/44,  te-2/1/45,  te-2/1/46,  te-2/1/47,  te-2/1/48,
                    qe-2/1/49,  qe-2/1/50,  qe-2/1/51,  qe-2/1/52,  qe-2/1/53,
                    qe-2/1/54,
10          tagged
            untagged    ae1,
```

LLDP配置

简介

LLDP概述

目前，网络设备的种类日益繁多而且各自的配置错综复杂，为了使不同厂商的设备能够在网络中互通，就需要一个标准的信息交流平台，用于交互各自的系统及配置信息。

LLDP（Link Layer Discovery Protocol）链路层发现协议，是IEEE 802.1ab定义的二层发现协议。LLDP提供了一种标准的链路层发现方式：将本端设备的主要能力、管理地址、设备标识、接口标识等信息组织成不同的TLV（Type/Length/Value，类型/长度/值），并封装在LLDPDU（Link Layer Discovery Protocol Data Unit，链路层发现协议数据单元）中发布给自己直连的邻居，邻居设备收到这些信息后将其以标准的管理信息库MIB（Management Information Base）的形式保存起来，以供网络管理系统查询及判断链路的通信状况。

LLDP报文收发机制

LLDP的工作模式

LLDP有以下四种工作模式：

- Tx/Rx模式：既可以接收又可以发送LLDP报文。
- Rx模式：只接收LLDP报文。
- Tx模式：只发送LLDP报文。
- Disabled：既不发送也不接收LLDP报文。

当端口的LLDP工作模式发生变化时，端口将对协议状态机进行初始化操作。为了避免端口模式频繁改变导致端口不断初始化，可以配置端口的初始化延迟时间，即当端口工作模式改变时延迟一段时间再执行初始化操作。

LLDP报文的发送

当端口工作在Tx/Rx或Tx模式时，设备会周期性地向邻居设备发送LLDP报文。如果设备的本地配置变化则会立即发送LLDP报文，通知邻居设备本地信息的变化。为了避免由于本地信息的频繁变化大量发送LLDP报文，每发送一个LLDP报文都需要延迟一段时间后再继续发送下一个报文。

当发现新的邻居设备（即收到一个新的LLDP报文且本地尚未保存发送该报文设备信息），或者设备的LLDP功能由去使能状态变为使能，或者设备的接口状态由Down变为Up的时候，该设备将自动启用快速发送机制。即将LLDP报文的发送周期缩短为1秒，并连续发送指定数量的LLDP报文后再恢复为正常的发送周期。

LLDP报文的接收

当端口工作在Tx/Rx或Rx模式时，设备会对收到的LLDP报文及其携带的TLV进行有效性检查，通过检查后再将邻居信息保存到本地。并根据LLDPDU报文中TLV携带的TTL值设置邻居信息在本地设备的老化时间。如果接收到的LLDPDU中的TTL值等于零，将立刻老化掉该邻居信息。

配置LLDP

缺省情况下，LLDP功能都是使能的。当需要对LLDP功能参数进行调整时，可以按照如下步骤进行。

1. 进入配置模式。

   ConnetOS> **configure**

2. 使能全局LLDP功能。

   ConnetOS# **set protocols lldp enable** { **false** | **true** }

   缺省情况下，全局的LLDP功能已经使能。

3. 配置接口下LLDP的工作模式。

   ConnetOS# **set protocols lldp interface** *interface-name* **status** { **disabled** | **rx-only** | **tx-only** | **tx-rx** }

   缺省情况下，接口下LLDP的工作模式为Tx/Rx。只有全局和接口下的LLDP都使能，LLDP功能才会生效。

4. 配置接口初始化延迟时间（接口下LLDP工作模式变化时，需要配置）。

   ConnetOS# **set protocols lldp reinit-delay** *reinit-delay*

5. 配置本设备允许发布的TLV类型。

   ConnetOS# **set protocols lldp tlv-select** { **mac-phy-cfg** | **management-address** | **port-description** | **port-vlan** | **system-capabilities** | **system-description** | **system-name** } **enable** { **false** | **true** }

   缺省情况下，本设备支持的TLV类型都发布。

6. 调整LLDP的相关参数

   - 配置邻居设备信息在本设备中保存的时间倍数

     ConnetOS# **set protocols lldp hold-time-multiplier** *hold-time-multiplier*

     缺省情况下，邻居设备信息在本设备中保持的时间倍数是4。

   - 配置LLDP报文的发送间隔

     ConnetOS# **set protocols lldp advertisement-interval** *advertisement-interval*

     缺省情况下，发送LLDP报文的时间间隔是30秒。

- 配置LLDP报文的发送延迟

  ConnetOS# set protocols lldp transmit-delay transmit-delay

  缺省情况下，发送LLDP报文的延迟时间为2秒。

7. 提交配置

   ConnetOS# **commit**

链路聚合配置

链路聚合简介

链路聚合通过将多条以太网物理链路捆绑在一起成为一条逻辑链路，从而实现增加链路带宽的目的。同时，通过相互间的动态备份，有效地提高链路的可靠性。

ConnetOS支持通过手动、动态LACP配置汇聚接口。同一个汇聚组中端口的基本配置应该保持一致，即如果某端口为trunk端口，则其他端口也配置为trunk端口；如该端口的链路类型改为access端口，则其他端口的链路类型也改为access端口。

链路聚合技术主要有以下三个优势：

- 增加带宽

  链路聚合接口的最大带宽可以达到各成员接口带宽之和。

- 提高可靠性

  当某条活动链路出现故障时，流量可以切换到其他可用的成员链路上，从而提高链路聚合接口的可靠性。

- 负载分担

在一个链路聚合组内，可以实现在各成员活动链路上的负载分担。

常见概念

LAG

LAG（Link Aggregation Group）链路汇聚组，是指将若干条以太链路捆绑在一起所形成的逻辑链路。

每个LAG唯一对应着一个逻辑接口，这个逻辑接口称之为汇聚接口。

组成汇聚接口接口的各个物理接口称为成员接口。成员接口对应的链路称为成员链路。

汇聚接口可以作为普通的以太网接口来使用，实现各种路由协议以及其它业务。与普通以太网接口的差别在于：转发的时候链路聚合组需要从成员接口中选择一个或多个接口来进行数据转发。

成员接口

汇聚组接口的成员接口存在：

- 活动接口：转发数据的接口

  对应的链路称为活动链路。

- 非活动接口两种：不转发数据的接口。

  对应的链路称为非活动链路。

活动接口数上限阈值

设置活动接口数上限阈值的目的是在保证带宽的情况下提高网络的可靠性。当前活动链路数目达到上限阈值时，再向汇聚组接口中添加成员接口，不会增加活动接口的数目，超过上限阈值的链路状态将被置为Down，作为备份链路。

活动接口数下限阈值

设置活动接口数下限阈值是为了保证最小带宽，当前活动链路数目小于下限阈值时，Eth-Trunk接口的状态转为Down。

LACP

LACP（Link Aggregation Control Protocol，链路聚合控制协议）基于IEEE 802.3ad标准，是一种实现链路动态聚合与解聚合的协议。

接口使能LACP协议后，接口通过LACPDU（Link Aggregation Control Protocol Data Unit，链路聚合控制协议数据单元）与对端交互信息，通告自己的系统优先级、系统mac、端口优先级、端口号和操作key。对端接收到这些信息后，将这些信息与端口所保存的信息比较，选择能够聚合的端口，双方可以对端口加入或退出某个动态聚合组达成一致。

负载分担

操作Key是在链路聚合时用来表明成员端口汇聚能力的一个数值，它是根据成员端口上的一些信息（包括该端口的速率、双工模式等）的组合自动计算生成的，这个信息组合中任何一项的变化都会引起操作Key的重新计算。在同一汇聚组中，所有的选中端口都必须具有相同的操作Key。

其中，动态聚合端口在使能lacp协议后，其管理key缺省为零。静态聚合端口在使能lacp后，端口的管理key与聚合组id相同。对于动态聚合组而言，同组成员一定有相同的操作key，而手工和静态聚合组中，selected的端口有相同的操作key。

手动配置汇聚接口

1. 进入配置模式。

   ConnetOS> **configure**

2. 创建汇聚接口

   ConnetOS# **set interface aggregate-ethernet** *ae-number*

   ConnetOS支持最多配置32个汇聚接口。

3. （可选）使能汇聚接口

   ConnetOS# **set interface aggregate-ethernet** *ae-number* **enable true**

   缺省情况下，汇聚接口组创建后就使能

4. （可选）配置汇聚接口的描述

   ConnetOS# **set interface aggregate-ethernet** *ae-number* **description** *description*

5. 将物理接口加入汇聚接口

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **ether-options 802.3ad** *ae-number*

   实际使用时，建议每个汇聚组的成员接口数不要超过8个。

6. 提交配置

   ConnetOS# **commit**

配置汇聚组负载均衡

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置LAG哈希算法

   ConnetOS# **set forwarding-options load-balance lag algorithm** *algorithm-num*

   缺省情况下，ConnetOS采用的hash算法是0。

3. 配置哈希因子

   ConnetOS# **set forwarding-options load-balance lag key** { **dest-ip** | **dest-mac** | **ether-type** | **ingress-interface** | **ip-protocol** | **l4-dest-port** | **l4-source-port** | **source-ip** | **source-mac** | **vlan-id** } **enable** { **false** | **true** }

4. 提交配置。

   ConnetOS# **commit**

配置LACP汇聚接口

1. 进入配置模式。

   ConnetOS> **configure**

2. 使能接口的LACP协议。

   ConnetOS# **set interface aggregate-ethernet** *interface-name* **ether-options lacp enable** { **false** | **true** }

3. 设置最小选中接口。

   ConnetOS# **set interface aggregate-ethernet** *interface-name* **ether-options min-selected-port** *port-number*

   缺省情况下，最小选中接口数是1。

   当汇聚组中的活动接口数少于设置的最小数值时，ConnetOS会自动Down掉整个汇聚组。

4. 配置汇聚接口组MTU

   ConnetOS# **set interface aggregate-ethernet** *ae-number* **mtu** *mtu-value*

5. 配置汇聚接口组的风暴控制

   ConnetOS# **set interface aggregate-ethernet** *interface-name* **storm-control** { **broadcast** | **multicast** | **unicast** } **kilobits** *suppress*

6. 提交配置

   ConnetOS# **commit**

可以通过:

- 执行命令 **show lacp neighbor**，查看LACP邻居:

```
ConnetOS> show lacp neighbor
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
        D -- Synchronization, E -- Collecting, F -- Distributing,
        G -- Defaulted, H -- Expired
Aggregated interface: ae1
Port Number   Partner System ID         Partner Port Num   Port Priority   Admin␣
↪Key   Oper Key   Flag
-----------   ----------------------   ----------------   -------------   -------
↪--   --------   ------------
te-1/1/33     0,00:00:00:00:00:00       0                  0               0x00  ␣
↪     0x00       {}
te-2/1/36     32768,2C:60:0C:84:61:49   28                 32768           0x00  ␣
↪     0x46       {ACDEF}
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
        D -- Synchronization, E -- Collecting, F -- Distributing,
        G -- Defaulted, H -- Expired
Aggregated interface: ae2
Port Number   Partner System ID         Partner Port Num   Port Priority   Admin␣
↪Key   Oper Key   Flag
-----------   ----------------------   ----------------   -------------   -------
↪--   --------   ------------
te-1/1/6      32768,2C:60:0C:84:61:49   4                  32768           0x00  ␣
↪     0x45       {ACDEF}
te-2/1/6      32768,2C:60:0C:84:61:49   6                  32768           0x00  ␣
↪     0x45       {ACDEF}
```

- 执行命令 **show lacp internal**，查看汇聚接口组的LACP状态:

```
ConnetOS> show lacp internal
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
        D -- Synchronization, E -- Collecting, F -- Distributing,
        G -- Defaulted, H -- Expired
LACP System ID: 32768,00:03:0F:64:DA:5F
Aggregated interface: ae1
Port Number   Priority   Admin Key   Oper Key   Flag
-----------   --------   ---------   --------   -------------
te-1/1/33     32768      0x4F        0x4F       {ACG}
te-2/1/36     32768      0x4F        0x4F       {ACDEF}
Aggregated interface: ae2
Port Number   Priority   Admin Key   Oper Key   Flag
-----------   --------   ---------   --------   -------------
te-1/1/6      32768      0x50        0x50       {ACDEF}
te-2/1/6      32768      0x50        0x50       {ACDEF}
```

- 执行命令 **show lacp statistics**，查看LACP协议包的状态:

```
ConnetOS> show lacp statistics
Port          LACP PDUs     LACP PDUs     Marker        Marker        Marker Resp  ␣
↪Marker Resp    LACP PDUs     LACP PDUs
Number        Sent          Received      Sent          Received      Sent         ␣
↪Received       Error         Dropped
----------    ----------    ----------    ----------    ----------    -----------   ---
↪---------    ----------    ----------
te-1/1/33     16865         0             0             0             0             0 ␣
↪             0             0
te-1/1/6      16869         16837         0             0             0             0 ␣
↪             0             0
te-2/1/36     16865         16865         0             0             0             0 ␣
↪             0             0
```

```
te-2/1/6      16865         16865         0             0             0             0 ␣
↪             0             0
```

三层转发配置

ARP配置

ARP简介

在局域网中，当主机或其它网络设备有数据要发送给另一个主机或设备时，它必须知道对方的网络层地址（即IP地址）。因为IP数据报文必须封装成帧才能通过物理网络发送，所以发送方还必须有接收方的物理地址（MAC地址），所以需要一个从IP地址到物理地址的映射。

ARP（Address Resolution Protocol）地址解析协议，就是用来将IP地址解析为MAC地址的协议。 设备通过ARP解析到目的MAC地址后，将会在自己的ARP表中增加IP地址到MAC地址的映射表项，用于后续到同一目的地报文的转发。

ARP 表项分为:

- 静态ARP表项

  由用户手动配置和维护，不会被老化，不会被动态ARP覆盖。

- 动态ARP表项

  由ARP协议自动生成和维护，可以被老化，被新的ARP报文更新，被静态ARP覆盖。

静态ARP

静态ARP表项可以限制和指定IP地址的设备通信时只使用指定的MAC地址，此时攻击报文无法修改此表项的IP地址和MAC地址的映射关系，从而保护了本设备和指定设备间的正常通信。

动态ARP

ARP表项动态学习是交换机本身就具有的功能，并且设备的缺省状态即为ARP表项动态学习，不需要使用命令启动此功能。

为适应网络的变化，ARP表需要不断更新。在达到老化时间时，如果仍不得到刷新的ARP表项将被从ARP表中删除。如果在到达老化时间前记录被刷新，则重新计算老化时间。用户可以根据网络实际情况调整老化时间。

ARP代理

如果ARP请求是从一个网络的主机发往另一个网络上的主机，连接这两个网络的设备就可以回答该请求，这个过程称作ARP代理（Proxy ARP）。ConnetOS支持ARP代理功能。

配置静态ARP

配置静态ARP表项虽然可以保护ARP表不被改写，但是配置工作量大，不适用于主机IP地址可能发生更改的网络环境，建议在比较小的网络里使用。

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置静态ARP表项。

   ConnetOS# **set protocols arp interface** *vlan-interface* **address** *ip-address* **mac-address** *mac-address*

   配置静态ARP表项时，ARP所在的vlan-interface必须已经创建好。

3. 提交配置。

   ConnetOS# **commit**

（可选）配置动态ARP老化时间

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置动态ARP老化时间

   ConnetOS# **set protocols arp aging-time** *aging-time*

   缺省情况下，ARP老化时间为1200s。

3. 提交配置。

   ConnetOS# **commit**

配置ARP代理

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置ARP代理

   ConnetOS# **set protocols arp interface** *vlan-interface* **proxy enable** { **false** | **true** }

   缺省情况下，ARP代理没有使能。

3. 提交配置。

   ConnetOS# **commit**

查看ARP的配置信息

执行 **show arp** 命令，可以查看ARP表项信息:

```
ConnetOS> show arp
Aging-time(seconds): 1200
Total count         : 1
Address          HW Address                    Type                  Interface
↪    Age
---------------  ----------------------------  --------------------  ------------------
↪-   ------------------
5.5.5.5          00:E0:EC:38:E1:BD             Dynamic               vlan5
↪    550
```

通过 **show route forward-host ipv4 all** 命令，查看与ARP表对应的主机路由表:

```
ConnetOS> show route forward-host ipv4 all
Address                 HWaddress                 Port
--------------------    ------------------------  ----------------------
5.5.5.5                 00:E0:EC:38:E1:BD         te-1/1/5
Total host count:1
```

DHCP Relay配置

简介

DHCP（Dynamic Host Configuration Protocol）动态主机配置协议，用于对用户IP地址进行动态分配和管理。

DHCP采用客户端/服务器模式，DHCP客户端向DHCP服务器动态地请求网络配置信息，DHCP服务器根据策略返回相应的配置信息（IP地址、子网掩码、缺省网关等网络参数）。

DHCP Relay，负责转发来自客户端方向或服务器方向的DHCP报文，协助DHCP客户端和DHCP服务器完成地址配置功能。它实现了不同网段间的DHCP服务器和客户端之间的报文交互，避免在每个网段范围内都部署DHCP服务器，既节省成本，又便于进行集中管理。

配置DHCP Relay功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 创建VLAN。

   ConnetOS# **set vlans vlan-id** *vlan-id*

3. 创建VLAN对应的接口vlan-interface。

   ConnetOS# **set vlans vlan-id** *vlan-id* **l3-interface** *l3-interface-name*

4. 指定DHCP Relay服务器。

   ConnetOS# **set vlan-interface interface** *l3-interface-name* **dhcp-relay server-ip** *ip-address*

5. 提交配置。

   ConnetOS# **commit**

查看DHCP Relay的配置信息

执行 **show vlan-interface interface** 命令，查看DHCP Relay的配置情况:

```
ConnetOS# show vlan-interface interface vlan100
Waiting for building configuration.
    description: ""
    mtu: 1500
    dhcp-relay {
        server-ip 1.1.1.1
    }
```

静态路由配置

IP路由简介

路由就是报文在转发过程中的路径信息，用来指导报文转发。

根据路由目的地的不同，路由划分为：

- 网段路由：目的地址为网段，子网掩码长度小于32位。
- 主机路由：目的地为主机，子网掩码长度为32位。

据来源的不同，路由可以分为三类：

- 直连路由：通过链路层协议发现的路由。
- 静态路由：通过网络管理员手动配置的路由。
- 动态路由：通过动态路由协议发现的路由。

ConnetOS支持配置静态路由和OSPF动态路由协议。

静态路由配置

静态路由适用于结构简单并且稳定的小型网络，能改进网络性能，并保证重要应用的带宽。但是当网络发生故障或者拓扑发生变化后，静态路由不会自动改变，必须手动修改。

配置静态路由必须配置：目的地址和掩码、下一跳。

配置静态路由

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置静态路由。

   ConnetOS# **set protocols static route** *ip-address* **next-hop** *ip-address*

3. （可选）设置路由metric。

   ConnetOS# **set protocols static route** *ip-address* **metric** *metric-value*

4. 提交配置。

   ConnetOS# **commit**

检查配置结果

- 配置完成后，查看软件路由表:

```
ConnetOS> show route table ipv4 unicast final

1.0.0.0/24          [connected(0)/0]
                  > via vlan100/vlan100
2.0.0.0/24          [connected(0)/0]
                  > via vlan200/vlan200
10.0.0.0/24         [static(1)/1]
                  > to 20.0.0.2 via vlan200/vlan200
```

```
20.0.0.0/24         [static(1)/1]
                  > to 2.0.0.2 via vlan200/vlan200
```

- 查看硬件路由表:

```
ConnetOS> show route forward-route ipv4 all
Destination       NetMask            NextHopMac          Port
---------------   ---------------                 ------------------
↪            ---------
1.0.0.0              255.255.255.0          CC:37:AB:F4:82:F3       connected
2.0.0.0           255.255.255.0      CC:37:AB:F4:82:F3       connected
20.0.0.0          255.255.255.0      00:10:94:00:00:04       te-1/1/4
Total route count:3
```

OSPF配置

OSPF简介

OSPF（Open Shortest Path First）开放式最短路径优先协议，是基于链路状态的内部网关协议。

OSPF采用最短路径SPF（Shortest Path First）算法，通过链路状态通告LSA（Link State Advertisement）描述网络拓扑，依据网络拓扑生成一棵最短路径树SPT（Shortest Path Tree），计算出到网络中所有目的地的最短路径，进行路由信息的交换。

目前针对IPv4协议使用的是OSPFv2。如果没有特别说明，文中的OSPF都是指OSPFv2（RFC 2328）。

OSPF建立过程

OSPF建立过程如下：

1. 建立邻接关系。
   - 本端设备通过接口向外发送Hello报文与对端设备建立邻居关系。
   - 两端设备进行主/从关系协商和DD报文交换。
   - 两端设备通过更新LSA完成链路数据库LSDB（Link State Database）的同步。

   邻接关系建立成功。
2. OSPF采用SPF算法计算路由。

   OSPF协议路由的计算过程如下：

   (a) 每台OSPF设备根据自己周围的网络拓扑结构生成链路状态通告LSA，并通过更新报文将LSA发送给网络中的其它OSPF设备。

   (b) 每台OSPF设备都会收集其它设备发来的LSA，所有的LSA放在一起便组成了链路状态数据库LSDB。LSA是对设备周围网络拓扑结构的描述，LSDB则是对整个自治系统的网络拓扑结构的描述。

   (c) OSPF设备将LSDB转换成一张带权的有向图，这张图便是对整个网络拓扑结构的真实反映。同一区域内各个设备得到的有向图是完全相同的。

OSPF优点

OSPF协议具有以下优点：

- 快速收敛：在网络的拓扑结构发生变化后立即发送更新报文，使这一变化在自治系统中同步。
- 无自环：根据收集到的LSA用最短路径树算法计算路由，从算法本身保证了不会生成自环路由。
- 区域划分：把自治系统AS（Autonomous System）划分成逻辑意义上的一个或多个区域来管理。链路状态数据库只需要和区域内的设备保持一致，降低网络带宽。
- 等价路由：支持到同一目的地址的多条等价路由。
- 支持验证：支持基于区域和接口的报文验证，以保证报文交互的安全性。

常见概念

AS（Autonomous System）自治系统

一组使用相同路由协议交换路由信息的路由设备。

Router ID

Router ID是一个32比特无符号整数，用于在AS中唯一标识一台路由设备。如果要运行OSPF协议，必须存在Router ID。建议将Router ID配置为与该设备某个接口的IP地址一致。

Router ID通过两种方式获得：

- 命令行手动配置。
- 自动选取。

如果没有手动配置Router ID，ConnetOS会从当前接口的IP地址中自动选取一个最大值作为Router ID。

只有重新配置系统的Router ID或OSPF的Router ID，并且重新启动OSPF进程后，才会进行Router ID的重新选取。

路由设备类型

OSPF路由设备根据在AS中的不同位置，可以分为以下四类:

- 区域内路由器（Internal Router）

  该类路由器的所有接口都属于同一个OSPF区域。

- ABR（Area Border Router）区域边界路由器

  该类路由设备可以同时属于两个以上的区域，但其中一个必须是骨干区域。ABR用来连接骨干区域和非骨干区域，它与骨干区域之间既可以是物理连接，也可以是逻辑上的连接。

- 骨干路由器（Backbone Router）

  该类路由设备至少有一个接口属于骨干区域。因此，所有的 ABR 和位于 Area0 的内部路由设备都是骨干路由器。

- ASBR（Autonomous System Boundary Router）自治系统边界路由器

  与其他AS交换路由信息的路由设备称为ASBR。ASBR并不一定位于AS的边界，它有可能是区域内路由设备，也有可能是ABR。只要一台OSPF路由设备引入了外部路由的信息，它就成为 ASBR。

路由类型

OSPF 将路由分为四类，按照优先级从高到低的顺序依次为：

- 区域内路由（Intra Area）
- 区域间路由（Inter Area）
- 第一类外部路由（Type1 External）
- 第二类外部路由（Type2 External)

区域内和区域间路由描述的是AS内部的网络结构，外部路由则描述了应该如何选择到AS以外目的地址的路由。

OSPF将引入的AS外部路由分为两类:Type1和Type2。

- Type1是指接收的是IGP（Interio rGateway Protocol，内部网关协议）路由（例如静态路由）。由于这类路由的可信程度较高，并且和OSPF自身路由的开销具有可比性，所以到第一类外部路由的开销等于本路由器到相应的ASBR的开销与ASBR到该路由目的地址的开销之和。
- Type2是指接收的是EGP（Exterior Gateway Protocol，外部网关协议）路由。由于这类路由的可信度比较低，所以OSPF协议认为从ASBR到自治系统之外的开销远远大于在自治系统之内到达ASBR的开销。所以计算路由开销时将主要考虑前者，即到第二类外部路由的开销等于ASBR到该路由目的地址的开销。如果计算出开销值相等的两条路由，再考虑本路由器到相应的ASBR的开销。

OSPF的认证方式

OSPF支持报文验证功能，只有通过验证的OSPF报文才能接收，否则将不能正常建立邻居。交换机采用接口验证的方式验证报文。

路由引入

当OSPF网络中的设备需要访问运行其他协议的网络中的设备时，需要将其他协议的路由引入到OSPF网络中。

OSPF可以引入其它路由协议学习到的路由。在引入时通过配置路由策略来过滤路由，只引入满足条件的路由。由于只有ASBR才能引入路由，因此该过滤规则只在ASBR上配置才有效。

OSPF报文类型

OSPF有五种类型的协议报文:

- Hello报文

  周期性发送，用来发现和维持OSPF邻居关系。包括：定时器的数值、DR（Designated Router，指定路由器）、BDR（Backup Designated Router，备份指定路由器）以及已知的邻居。

- DD（Database Description，数据库描述）报文

  描述本地LSDB中每一条LSA的摘要信息，用于两台路由器进行数据库同步。

- LSR（Link State Request，链路状态请求）报文

  向对方请求所需的LSA。两台路由器互相交换DD报文之后，得知对端的路由器有哪些LSA是本地的LSDB所缺少的之后，发送LSR报文向对方请求所需的LSA。内容包括所需要的LSA的摘要。

- LSU（Link State Update，链路状态更新）报文

  向对方发送其所需要的LSA。

- LSAck（Link State Acknowledgment，链路状态确认）报文

  用来对收到的LSA进行确认。内容是需要确认的LSA的Header（一个报文可对多个LSA 进行确认）。

LSA类型

OSPF中对链路状态信息的描述都是封装在LSA中发布出去，常用的LSA有以下几种类型：

- Router LSA（Type1）：由每个路由设备产生，描述路由设备的链路状态和开销，在其始发的区域内传播。
- Network LSA（Type2）：由DR产生，描述本网段所有路由设备的链路状态，在其始发的区域内传播。
- Network Summary LSA（Type3）：由ABR（Area Border Router，区域边界路由器）产生，描述区域内某个网段的路由，并通告给其他区域。
- ASBR Summary LSA（Type4）：由ABR产生，描述到ASBR的路由，通告给相关区域。
- AS External LSA（Type5）：由ASBR产生，描述到AS外部的路由，通告到所有的区域（除了Stub区域和NSSA区域）。
- NSSA External LSA（Type7）：由NSSA（Not-So-Stubby Area）区域内的ASBR产生，描述到AS外部的路由，仅在NSSA区域内传播。
- Opaque LSA：是一个被提议的LSA类别，由标准的LSA头部后面跟随特殊应用的信息组成，可以直接由OSPF协议使用，或者由其它应用分发信息到整个OSPF域间接使用。

  Opaque LSA分为Type9、Type10、Type11三种类型，泛洪区域不同。其中，Type9的OpaqueLSA仅在本地链路范围进行泛洪，Type10的Opaque LSA仅在本地区域范围进行泛洪，Type11的LSA可以在一个自治系统范围进行泛洪。

选路规则

OSPF协议有RFC2328和RFC1583两种不同的选路规则。在如何选择最优路由的问题上，RFC1583和RFC2328所定义的优先规则是不相同的：

- 当RFC1583选路规则被使能时，设备会根据开销值选择发布到相同目的地址的路由。
- 当RFC1583选路规则被关闭时，设备会先根据路由类型来选择发布到相同目的地址的路由，其次才是路由的开销值。

OSPF路由域中的所有设备应统一配置同一种选路规则。目前大部分OSPF路由域都配置成RFC2328规定的选路规则。

区域Area

区域分类

随着网络规模的扩大，当网络中运行OSPF的路由设备较多时，会导致LSDB非常庞大，占用大量的存储空间。而随着SPF算法复杂度的增加，CPU负担也变得很重。而网络规模的增大，拓扑结构发生变化的概率也增大，造成大量的OSPF传递及路由重新计算，网路经常处于“动荡”之中。

OSPF协议通过将AS划分成不同的区域（Area）来解决这个问题。区域是从逻辑上将路由器划分为不同的组，每个组用区域号（Area ID）来标识。区域的边界不是链路，而是路由设备。一个路由设备可以属于不同的区域，但是一个网段（链路）只能属于一个区域，即每个运行OSPF的接口必须指明属于哪一个区域。

划分区域后，可以在区域边界路由设备上进行路由聚合，以减少通告到其他其他区域的LSA 数量，还可以将网络拓扑变化带来的影响最小化。

OSPF的区域类型分为：

- Normal：普通区域，分为标准区域和骨干区域。
- Stub区域：不传播它们接收到的自治系统外部路由，只允许发布区域内路由。
- NSSA区域：能够将自治域外部路由引入并传播到整个OSPF自治域中，同时又不会学习来自OSPF网络其它区域的外部路由。

Normal区域

普通区域，分为标准区域和骨干区域：

- 标准区域是最通用的区域，它传输区域内路由，区域间路由和外部路由。
- 骨干区域是连接所有其他OSPF区域的中央区域，用Area 0表示。骨干区域负责区域之间的路由，非骨干区域之间的路由信息必须通过骨干区域来转发。

区域必须满足：

- 所有非骨干区域必须与骨干区域保持连通。
- 骨干区域自身也必须保持连通。

在实际应用中，如果因为各方面条件的限制，无法满足所有非骨干区域与骨干区域保持连通的要求，此时可以通过配置OSPF虚连接（Virtual Link）来解决这个问题。

虚连接是指在两台ABR之间通过一个非骨干区域而建立的一条逻辑上的连接通道。虚连接相当于在两个ABR之间形成了一个点到点的连接，两端接口上配置的参数必须一致，如Hello报文间隔。为虚连接两端提供一条非骨干区域内部路由的区域称为传输区（Transit Area）。

Stub区域

Stub区域是一些特定的区域，Stub区域的ABR不允许注入Type5 LSA。对于位于AS边缘的非骨干区域，可以将区域配置为Stub区域，能避免Type5 LSA在Stub区域的泛洪，减少路由表的规模。

为保证到本自治系统的其他区域或者自治系统外的路由依旧可达，该区域的ABR将生成一条缺省路由，并发布给本区域中的其他非ABR路由器。

Totally Stub（完全Stub）区域的ABR不会将区域间的路由信息和外部路由信息传递到本区域，Stub区域中的路由表规模以及路由信息传递的数量进一步减少。

配置（Totally）Stub区域时需要注意：

- 骨干区域不能配置成Stub区域。
- 如果要将一个区域配置成Stub区域，则该区域中的所有路由设备必须都要配置stub命令。

- （Totally）Stub区域内不能存在ASBR，即自治系统外部的路由不能在本区域内传播。
- 虚连接不能穿过（Totally）Stub区域。

NSSA区域

NSSA（Not-So-Stubby Area）区域是Stub区域的变形。NSSA区域也不允许Type5 LSA注入，但可以允许Type7 LSA注入。

Type7 LSA由NSSA区域的ASBR产生，在NSSA区域内传播。当Type7 LSA到达NSSA的ABR时。由ABR将Type7 LSA转换成Type5 LSA。传播到其他区域。

网络类型

链路两端的OSPF接口的网络类型必须一致，否则双方不能建立起邻居关系。根据链路层协议类型，OSPF支持以下四种类型的网络:

- 广播（Broadcast）

  链路层协议是Ethernet、FDDI时，OSPF缺省认为网络类型是Broadcast。

  - 通常以组播形式（224.0.0.5）发送Hello报文和LSAck报文。
  - 对于LSU报文，通常以组播形式首次发送，以单播形式进行重传。
  - 以单播形式发送DD报文和LSR报文。
- NBMA（Non-Broadcast Multi-Access）

  当链路层协议是ATM时，OSPF缺省认为网络类型是NBMA。

  - NBMA网络必须是全连通的，即网络中任意两台交换机之间都必须直接可达。
  - 以单播形式发送协议报文（Hello报文、DD报文、LSR报文、LSU报文、LSAck报文）。
- 点到多点P2M（point-to-multipoint）

  没有一种链路层协议缺省是P2M，P2M类型是由其他的网络类型强制更改的。通常将NBMA网络改为P2M网络。

  - P2M网络中的掩码长度必须一致。
  - 以组播形式（224.0.0.5）发送Hello报文，以单播形式发送DD报文、LSR报文、LSU报文、LSAck报文。
- 点到点P2P（point-to-point）

  NBMA网络必须是全连通的，即网络中任意两台路由器之间都必须有一条虚电路直接可达。

  - 如果部分路由器之间没有直接可达的链路时，应将接口配置成P2M类型。
  - 如果路由器在NBMA网络中只有一个对端，也可将接口类型配置为P2M类型。

配置OSPF基本功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置Router ID。

   ConnetOS# **set protocols ospf4 router-id** *router-id*

   缺省情况下，Router ID是0.0.0.0。

   修改router-id后必须重启系统或者在修改router-id之前先删除所有OSPF配置。

3. 创建OSPF区域。

   ConnetOS# **set protocols ospf4 area** *area-id*

   缺省情况下，创建OSPF区域后，区域类型为normal。

   骨干区域的Area ID为0。

4. 配置OSPF区域所包含的网段。

   ConnetOS# **set protocols ospf4 area** *area-id* **area-range** *network-address* **advertise enable** { **false** | **true** }

   一个网段只能属于一个区域。

5. 使能接口的OSPF功能

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* [**vif** *l3-interface-name.1*] **addres** *ip-address* **enable** { **false** | **true** }

   如果 **vlan-interface** 下配置了两个或多个IP地址，则选择参数 *vif l3-interface-name.1* ，发布除第一个以外的IP地址。

   一个网段只能属于一个区域，每个运行OSPF协议的接口必须指明所属的区域。

6. 提交配置。

   ConnetOS# **commit**

配置OSPF区域

当对整个网络划分区域完毕后，可以根据组网需要进一步将区域配置成Stub区域或者NSSA区域。

当非骨干区域不能与非骨干区域保持连通，或者骨干区域因为各方面的限制无法保持连通时，可以通过配置OSPF虚连接解决。

1. 配置区域类型。

   ConnetOS# **set protocols ospf4 area** *area-id* **area-type** { **normal** | **nssa** | **stub** }

   对于位于AS边缘的非骨干区域，可以将区域配置为Stub区域。

2. 使能在stub区域中生成缺省路由的功能。

   ConnetOS# **set protocols ospf4 area default-lsa enable** { **false** | **true** }

   缺省情况下，此功能已经使能。

3. 配置OSPF发送到Stub区域的缺省路由开销。

   ConnetOS# **set protocols ospf4 area** *area-id* **default-lsa metric** *metric*

   本命令只能在Stub区域的ABR上配置才能生效。

4. 配置虚连接。

   ConnetOS# **set protocols ospf4 area** *area-id* **virtual-link** *ip-address* **authentication** { **md5** *key-id* | **simple-password** *password* } | **hello-interval** *hello-interval* | **retransmit-interval** *retransmit-interval* | **router-dead-interval** *router-dead-interval* | **transmit-area** *transmit-area-id* | **transmit-delay** *transmit-delay* }

为使虚连接生效，虚连接的另一端也需要配置此命令，并且 **hello-interval** 和 **router-dead-interval** 的值必须一致。

5. 提交配置。

   ConnetOS# **commit**

配置OSPF的网络类型

1. 配置接口的网络类型。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **link-type** { **broadcast** | **p2m** | **p2p** }

2. （可选）配置OSPF选举时的DR优先级。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **priority** *priority-value*

   只有在接口的网络类型是广播时，才会选举DR，其他网络类型不需要。

3. 提交配置。

   ConnetOS# **commit**

配置OSPF选路信息

1. 配置OSPF接口的开销值。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **interface-cost** *interface-cost*

2. 配置兼容RFC1583的外部路由选择规则。

   ConnetOS# **set protocols ospf4 rfc1583-compatibility enable** { **false** | **true** }

   OSPF路由域中的所有设备应统一配置同一种选路规则。

3. 提交配置。

   ConnetOS# **commit**

配置OSPF的路由信息控制

1. 配置应用路由策略

   ConnetOS# **set protocols ospf4** { **export** *import-policy* | **mport** *import-policy* }

2. 配置路由聚合。

   ConnetOS# **set protocols ospf4 area** *area-id* **summaries enable** { **false** | **true** }

3. 配置只通告，但是不运行OSPF协议。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **passive** [**host**] **enable** { **false** | **true** }

4. 配置邻居OSPF路由设备。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **neighbor** *ip-address* [**router-id** *router-id*]

5. 提交配置。

   ConnetOS# **commit**

调整和优化OSPF网络

1. 配置接口发送hello报文的时间间隔。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **hello-interval** *hello-interval*

   缺省情况下，接口发送hello报文的时间间隔为10秒。

2. 配置相邻邻居失效时间间隔。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **router-dead-interval** *router-dead-interval*

   缺省情况下，OSPF邻居失效时间间隔为40秒。

3. 配置接口的LSA传送延迟时间。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **transmit-delay** *transmit-delay*

   缺省情况下，LSA传送的延迟时间为1秒。

4. 配置相邻交换机重传LSA的时间间隔。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **retransmit-interval** *retransmit-interval*

   缺省情况下，LSA重传的时间间隔为5秒。

5. 配置OSPF接口的验证方式。

   ConnetOS# **set protocols ospf4 area** *area-id* **interface** *l3-interface-name* **address** *ip-address* **authentication** { **md5** *md5* | **simple-password** }

6. 提交配置。

   ConnetOS# **commit**

BGP配置

BGP简介

概述

为方便管理规模不断扩大的网络，网络被分成了不同的自治系统。

BGP（Border Gateway Protocol，边界网关协议）是一种实现自治系统AS（Autonomous System）之间的路由可达，并选择最佳路由的距离矢量路由协议。

BGP具有以下优点：

- BGP采用认证和GTSM的方式，保证了网络的安全性。
- BGP提供了丰富的路由策略，能够灵活的进行路由选路。
- BGP提供了路由聚合和路由衰减功能用于防止路由振荡，有效提高了网络的稳定性。

- BGP使用TCP作为其传输层协议（端口号为179），并支持BGP与BFD联动、BGP Auto FRR和BGP GR和NSR，提高了网络的可靠性。

常用概念

自治系统AS

AS(Autonomous System，自治系统)是指在一个实体管辖下的拥有相同选路策略的IP网络。

BGP网络中的每个AS都被分配一个唯一的AS号，用于区分不同的AS。AS号分为2字节AS号和4字节AS号，其中2字节AS号的范围为1至65535，4字节AS号的范围为1至4294967295。支持4字节AS号的设备能够与支持2字节AS号的设备兼容。

报文交互中的角色

BGP报文交互中分为Speaker和Peer两种角色。

- Speaker：发送BGP报文的设备称为BGP发言者（Speaker），它接收或产生新的报文信息，并发布（Advertise）给其它BGP Speaker。
- Peer：相互交换报文的Speaker之间互称对等体（Peer）。若干相关的对等体可以构成对等体组（Peer Group）。

BGP的Router ID

BGP的Router ID是一个用于标识BGP设备的32位值，通常是IPv4地址的形式，在BGP会话建立时发送的Open报文中携带。对等体之间建立BGP会话时，每个BGP设备都必须有唯一的Router ID，否则对等体之间不能建立BGP连接。

BGP的Router ID在BGP网络中必须是唯一的，可以采用手工配置，也可以让设备自动选取。缺省情况下，BGP选择设备上的Loopback接口的地址作为BGP的Router ID。如果设备上没有配置Loopback接口，系统会选择接口中最大的IPv4地址作为BGP的Router ID。

BGP分类

BGP按照运行方式分为EBGP（External/Exterior BGP）和IBGP（Internal/Interior BGP）

- EBGP：运行于不同AS之间的BGP称为EBGP。为了防止AS间产生环路，当BGP设备接收EBGP对等体发送的路由时，会将带有本地AS号的路由丢弃。
- IBGP：运行于同一AS内部的BGP称为IBGP。为了防止AS内产生环路，BGP设备不将从IBGP对等体学到的路由通告给其他IBGP对等体，并与所有IBGP对等体建立全连接。IBGP对等体的连接数量太多时，可以用路由反射器和BGP联盟来减少连接的数量。

路由反射器RR

为保证IBGP对等体之间的连通性，需要在IBGP对等体之间建立全连接关系。当设备数目很多时，设备配置将十分复杂，而且配置后网络资源和CPU资源的消耗都很大。在IBGP对等体间使用路由反射器可以解决以上问题。

路由反射器相关角色为

- 路由反射器RR（Route Reflector）：允许把从IBGP对等体学到的路由反射到其他IBGP对等体的BGP设备，类似OSPF网络中的DR。
- 客户机（Client）：与RR形成反射邻居关系的IBGP设备。在AS内部客户机只需要与RR直连。
- 非客户机（Non-Client）：既不是RR也不是客户机的IBGP设备。在AS内部非客户机与RR之间，以及所有的非客户机之间仍然必须建立全连接关系。
- 始发者（Originator）：在AS内部始发路由的设备。Originator_ID属性用于防止集群内产生路由环路。
- 集群（Cluster）：路由反射器及其客户机的集合。Cluster_List属性用于防止集群间产生路由环路。

同一cluster内的客户机只需要与该集群的RR建立IBGP连接，不需要与其他客户机建立IBGP连接，从而减少了IBGP连接数量。

RR突破了"从IBGP对等体获得的BGP路由，BGP设备只发布给它的EBGP对等体。"的限制，并采用独有的Cluster_List属性和Originator_ID属性防止路由环路。RR向IBGP邻居发布路由规则如下：#这段话需要再理一下，放到最前的介绍去

- 从非客户机学到的路由，发布给所有客户机。
- 从客户机学到的路由，发布给所有非客户机和客户机（发起此路由的客户机除外）。
- 从EBGP对等体学到的路由，发布给所有的非客户机和客户机。

BGP联盟

和RR一样，BGP联盟也是为了解决AS内部的IBGP网络连接激增问题。

联盟将一个AS划分为若干个子AS。每个子AS内部建立IBGP全连接关系，子AS之间建立联盟EBGP连接关系，但联盟外部AS仍认为联盟是一个AS。配置联盟后，原AS号将作为每个路由器的联盟ID。

这样的好处是：

- 可以保留原有的IBGP属性，包括Local Preference属性、MED属性和NEXT_HOP属性等
- 联盟相关的属性在传出联盟时会自动被删除，即管理员无需在联盟的出口处配置过滤子AS号等信息的操作。

路由衰减

路由振荡指路由表中添加一条路由后，该路由又被撤销的过程。当发生路由振荡时，设备就会向邻居发布路由更新，收到更新报文的设备需要重新计算路由并修改路由表。当BGP应用于复杂的网络环境时，路由振荡就会十分频繁。为了避免频繁的路由振荡，BGP使用路由衰减来抑制不稳定的路由。

路由衰减只对EBGP路由起作用，对IBGP路由不起作用。

路由衰减使用惩罚值（Penalty value）来衡量一条路由的稳定性，惩罚值越高说明路由越不稳定。

1. 路由每发生一次振荡，BGP便会给此路由增加1000的惩罚值，其余时间惩罚值会慢慢下降。
2. 当惩罚值超过抑制阈值（suppress value）时，此路由被抑制，不加入到路由表中，也不再向其他BGP对等体发布更新报文。
3. 被抑制的路由每经过一段时间，惩罚值便会减少一半，这个时间称为半衰期（half-life）。
4. 当惩罚值降到再使用阈值（reuse value）时，此路由变为可用并被加入到路由表中，同时向其他BGP对等体发布更新报文。

从路由被抑制到路由恢复可用的时间称为抑制时间（suppress time）。

工作原理

报文交互

BGP的报文类型

BGP对等体间通过以下5种报文进行交互，其中Keepalive报文为周期性发送，其余报文为触发式发送：

- Open报文：用于建立BGP对等体连接。
- Update报文：用于在对等体之间交换路由信息。
- Notification报文：用于中断BGP连接。
- Keepalive报文：用于保持BGP连接。
- Route-refresh报文：用于在改变路由策略后请求对等体重新发送路由信息。只有支持路由刷新（Route-refresh）能力的BGP设备会发送和响应此报文。

BGP对等体交互过程

BGP对等体的交互过程中存在6种状态机：

- 空闲（Idle）
- 连接（Connect）
- 活跃（Active）
- Open报文已发送（OpenSent）
- Open报文已确认（OpenConfirm）
- 连接已建立（Established）。

在BGP对等体建立的过程中，通常可见的3个状态是：Idle、Active和Established。

BGP对等体交互过程如下：

#.Idle状态是BGP初始状态。在Idle状态下，BGP拒绝邻居发送的连接请求。只有在收到本设备的Start事件后，BGP才开始尝试和其它BGP对等体进行TCP连接，并转至Connect状态。

..note::

- Start事件是由一个操作者配置一个BGP过程，或者重置一个已经存在的过程或者路由器软件重置BGP过程引起的。
- 任何状态中收到Notification报文或TCP拆链通知等Error事件后，BGP都会转至Idle状态。

1. 在Connect状态下，BGP启动连接重传定时器（Connect Retry），等待TCP完成连接。
   - 如果TCP连接成功，那么BGP向对等体发送Open报文，并转至OpenSent状态。
   - 如果TCP连接失败，那么BGP转至Active状态。
   - 如果连接重传定时器超时，BGP仍没有收到BGP对等体的响应，那么BGP继续尝试和其它BGP对等体进行TCP连接，停留在Connect状态。
2. 在Active状态下，BGP总是在试图建立TCP连接。
   - 如果TCP连接成功，那么BGP向对等体发送Open报文，关闭连接重传定时器，并转至OpenSent状态。

- 如果TCP连接失败，那么BGP停留在Active状态。
- 如果连接重传定时器超时，BGP仍没有收到BGP对等体的响应，那么BGP转至Connect状态。

3. 在OpenSent状态下，BGP等待对等体的Open报文，并对收到的Open报文中的AS号、版本号、认证码等进行检查。
   - 如果收到的Open报文正确，那么BGP发送Keepalive报文，并转至OpenConfirm状态。
   - 如果发现收到的Open报文有错误，那么BGP发送Notification报文给对等体，并转至Idle状态。
4. 在OpenConfirm状态下，BGP等待Keepalive或Notification报文。如果收到Keepalive报文，则转至Established状态，如果收到Notification报文，则转至Idle状态。
5. 在Established状态下，BGP可以和对等体交换Update、Keepalive、Route-refresh报文和Notification报文。
   - 如果收到正确的Update或Keepalive报文，那么BGP就认为对端处于正常运行状态，将保持BGP连接。
   - 如果收到错误的Update或Keepalive报文，那么BGP发送Notification报文通知对端，并转至Idle状态。
   - Route-refresh报文不会改变BGP状态。
   - 如果收到Notification报文，那么BGP转至Idle状态。
   - 如果收到TCP拆链通知，那么BGP断开连接，转至Idle状态。

BGP对等体之间的交互原则

BGP设备将最优路由加入BGP路由表，形成BGP路由。BGP设备与对等体建立邻居关系后，采取以下交互原则： 从IBGP对等体获得的BGP路由，BGP设备只发布给它的EBGP对等体。 从EBGP对等体获得的BGP路由，BGP设备发布给它所有EBGP和IBGP对等体。 当存在多条到达同一目的地址的有效路由时，BGP设备只将最优路由发布给对等体。 路由更新时，BGP设备只发送更新的BGP路由。 所有对等体发送的路由，BGP设备都会接收。

BGP路由选路和负载分担

在BGP路由表中，到达同一目的地可能存在多条路由。此时BGP会选择其中一条路由作为最佳路由，并只把此路由发送给其对等体。BGP为了选出最佳路由，会根据BGP的路由优选规则依次比较这些路由的BGP属性。

BGP属性

路由属性是对路由的特定描述，所有的BGP路由属性都可以分为以下4类，常见BGP属性类型如下表所示。

属性类型	特点	常见属性
公认必须遵循	所有BGP设备都可以识别 而且必须存在于Update报文中。 缺少这类属性，路由信息就会出错。	Origin属性 AS_Path属性 Next_Hop属性
公认任意	所有BGP设备都可以识别 缺少这类属性，路由信息也不会出错。	Local_Pref属性
可选过渡	可以不识别此类属性 会接收这类属性，并通告给其他对等体。	团体属性
可选非过渡	可以不识别此类属性 忽略该属性，且不会通告给其他对等体。	MED属性 Originator_ID属性 Cluster_List属性

- Origin属性

  Origin属性用来定义路径信息的来源，标记一条路由是怎么成为BGP路由的。它有以下3种类型：

  - IGP：具有最高的优先级。通过network命令注入到BGP路由表的路由，其Origin属性为IGP。
  - EGP：优先级次之。通过EGP得到的路由信息，其Origin属性为EGP。
  - Incomplete：优先级最低。通过其他方式学习到的路由信息。比如BGP通过import-route命令引入的路由，其Origin属性为Incomplete。

- AS_Path属性

  AS_Path属性按矢量顺序记录了某条路由从本地到目的地址所要经过的所有AS编号。在接收路由时，设备如果发现AS_Path列表中有本AS号，则不接收该路由，从而避免了AS间的路由环路。

- Next_Hop属性

  Next_Hop属性记录了路由的下一跳信息。BGP的下一跳属性和IGP的有所不同，不一定就是邻居设备的IP地址。

- Local_Pref属性

  Local_Pref属性表明路由器的BGP优先级，用于判断流量离开AS时的最佳路由。

- 团体属性

  团体属性（Community）用于标识具有相同特征的BGP路由，使路由策略的应用更加灵活，同时降低了维护管理的难度。

- MED属性

  MED（Multi-Exit Discriminator）属性用于判断流量进入AS时的最佳路由，当一个运行BGP的设备通过不同的EBGP对等体得到目的地址相同但下一跳不同的多条路由时，在其它条件相同的情况下，将优先选择MED值较小者作为最佳路由。

  MED属性仅在相邻两个AS之间传递，收到此属性的AS一方不会再将其通告给任何其他第三方AS。MED属性可以手动配置，如果路由没有配置MED属性，BGP选路时将该路由的MED值按缺省值0来处理。

- Originator_ID属性

  Originator ID由RR产生，使用的Router ID的值标识路由的始发者，用于防止集群内产生路由环路。

- Cluster_List属性

  路由反射器和它的客户机组成一个集群（Cluster），使用AS内唯一的Cluster ID作为标识。为了防止集群间产生路由环路，路由反射器使用Cluster_List属性，记录路由经过的所有集群的Cluster ID。

配置BGP基本功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置BGP使用的AS编号。

   ConnetOS# **set protocols bgp local-as** *as-id*

3. （可选）配置BGP使用的4字节的AS编号。

   ConnetOS# **set protocols bgp 4byte-as-numbers enable** { **false** | **true** }

   缺省情况下，BGP使用2字节的AS编号。

4. 设置BGP设备的Router ID。

   ConnetOS# **set protocols bgp bgp-id** *bgp-id*

   缺省情况下，设备没有配置Router ID。

5. 将指定设备配置为BGP对等体。

   ConnetOS# **set protocols bgp peer** *peer-id* **enable** { **false** | **true** }

6. 配置指定BGP对等体的AS编号。

   ConnetOS# **set protocols bgp peer** *peer-id* **as** *as-id*

7. 配置建立BGP对等体时的空闲时间。

   ConnetOS# **set protocols bgp peer** *peer-id* **holdtime** *holdtime*

8. 配置建立BGP对等体连接的延迟时间。

   ConnetOS# **set protocols bgp peer** *peer-id* **delay-open-time*** *delay-open-time*

9. 提交配置。

   ConnetOS# **commit**

配置IBGP网络连接简化

配置BGP路由反射器

1. 进入配置模式。

   ConnetOS> **configure**

2. 使能BGP路由反射器。

   ConnetOS# **set protocols bgp route-reflector enable** { **false** | **true** }

   缺省情况下，路由反射器功能是使能的。

3. 配置BGP对等体为路由反射器的客户。

   ConnetOS# **set protocols bgp peer** *peer-id* **client enable** { **false** | **true** }

4. 设置路由反射器的集群ID。

   ConnetOS# **set protocols bgp route-reflector cluster-id** *cluster-id*

5. 提交配置。

   ConnetOS# **commit**

配置BGP联盟

1. 进入配置模式。

   ConnetOS> **configure**

2. 使能BGP联盟。

   ConnetOS# **set protocols bgp confederation enable** { **false** | **true** }

3. 设置联盟ID。

   ConnetOS# **set protocols bgp confederation identifier** *confederation-id*

4. 将BGP对等体加入BGP联盟。

   ConnetOS# **set protocols bgp peer** *peer-id* confederation-member enable** { **false** | **true** }

5. 提交配置。

   ConnetOS# **commit**

配置BGP网络的安全性

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置BGP对等体在建立TCP连接时对BGP消息进行MD5认证。

   ConnetOS# **set protocols bgp peer** *peer-id* **md5-password** *md5-password*

3. 提交配置。

   ConnetOS# **commit**

配置BGP网络的收敛速度

1. 进入配置模式。

   ConnetOS> **configure**

2. 使能BGP路由衰减功能。

   ConnetOS# **set protocols bgp damping enable** { **false** | **true** }

   缺省情况下，路由衰减功能是使能的。

3. 设置路由进入抑制状态的抑制阈值。

   ConnetOS# **set protocols bgp damping suppress** *suppress-value*

   缺省情况下，抑制阈值是3000。

4. 设置路由的抑制时间。

   ConnetOS# **set protocols bgp damping max-suppress** *max-suppress-time*

   缺省情况下，路由抑制时间是60分钟。

5. 设置可达路由的半衰期。

   ConnetOS# **set protocols bgp damping half-life** *half-life*

   缺省情况下，半衰期是15分钟。

6. 设置路由解除抑制状态时的再使用阈值。

   ConnetOS# **set protocols bgp damping reuse** *reuse-value*

   缺省情况下，再使用阈值是750。

7. 提交配置。

   ConnetOS# **commit**

路由策略配置

简介

概述

路由策略主要实现了路由过滤和路由属性设置等功能，它通过改变路由属性（包括可达性）来改变网络流量所经过的路径。

路由协议在发布、接收和引入路由信息时，如果根据实际组网需求实施一些策略，那么就可以对路由信息进行过滤和改变路由信息的属性，如：

- 控制路由的接收和发布

  只发布和接收必要、合法的路由信息，以控制路由表的容量，提高网络的安全性。

- 控制路由的引入

  在一种路由协议在引入其它路由协议发现的路由信息丰富自己的路由信息时，只引入一部分满足条件的路由信息。

- 设置特定路由的属性

  修改通过路由策略过滤的路由的属性，满足自身需要。

路由策略具有以下：

- 通过控制路由器的路由表规模，节约系统资源。
- 通过控制路由的接收、发布和引入，提高网络安全性。
- 通过修改路由属性，对网络数据流量进行合理规划，提高网络性能。

路由策略和策略路由

策略路由（Policy-based Routing），本质上是路由。是一种根据用户指定的策略进行路由选择的机制，比基于目标网络进行路由更加灵活。

路由策略（Routing Policy），本质上是一种策略。是为了改变网络流量所经过的途径而修改路由信息的技术，主要通过改变路由属性（包括可达性）来实现。

差异	策略路由	路由策略
作用对象	数据包	路由信息
是否改变转发流程	改变	不改变
实现主体	转发平面，保证数据包按照策略转发	控制平面，实现路由过滤和属性修改
过滤机制	匹配五元组： • 源IP地址 • 目的IP地址 • 协议 • 源端口 • 目的端口	匹配路由： • 目标网段 • 下一跳 • 度量值 • Tag • Community
应用主体	• 本地数据包 • 转发数据包	• 直连路由 • 静态路由 • OSPF • BGP

路由策略的配置逻辑

路由策略的视线，分为两步：

1. 定义路由策略。即定义一组匹配规则，标识出将要实施路由策略的路由信息。可以采用路由信息中的不同属性作为匹配依据进行设置。

   ConnetOS采用策略名＋策略内容名＋动作＋策略规则的方式，来实现路由策略：

   set policy policy-statement *policy-name* **term** *term-name* ｛ **from** ｜ **to** ｜ **then** ｝ ＋ 策略规则

   set policy policy-statement *policy-name* **term** *term-name* **then** ＋ 策略动作

   set policy policy-statement then ＋ 策略动作

   其中：

   - *policy-name*：策略名称。
   - *term-name*：策略内容名称。一个策略策略下可以配置多个 *term-name*。
   - **from**：应用于源地址路由
   - **to**：应用于目的地址路由。
   - **then**：匹配策略后的后的动作。

2. 应用路由策略。

   将匹配规则应用于路由的发布、接收和引入。

   路由策略设置好后，直接在路由协议中应用即可生效。

配置路由策略

1. 进入配置模式。

   ConnetOS> **configure**

2. 定义路由策略的应用对象。

- 定义应用于源地址路由的路由策略。

  ConnetOS# **set policy policy-statement** *policy-name* **term** *term-name* **from** { **as-path** *as-path-name* | **as-path-list** *as-path-list-name* | **community** *community-name* | **community-list** *community-list-name* | **external-type** *type-number* | **localpref** *preference-number* | **med** *med* | **metric** *metric* | **neighbor** *neighbor-router* | **network4** *network4-address* | **network4-list** *network4-list-name* | **nexthop4** *nexthop4-address* | **origin** *origin-attribute* | **prefix-length4** *prefix-length4* | **protocol** { **connected** | **ospf4** | **static** } | **tag** *tag-value* }

- 定义应用于目的地址路由的路由策略。

  ConnetOS# **set policy policy-statement** *policy-name* **term** *term-name* **to** { **as-path** *as-path-name* | **as-path-list** *as-path-list-name* | **community** *community-name* | **community-list** *community-list-name* | **external-type** *type-number* | **localpref** *preference-number* | **med** *med* | **metric** *metric* | **neighbor** *neighbor-router* | **network4** *network4-address* | **network4-list** *network4-list-name* | **nexthop4** *nexthop4-address* | **origin** *origin-attribute* | **prefix-length4** *prefix-length4* | **protocol** { **connected** | **ospf4** | **static** } | **tag** *tag-value* | **was-aggregated** { **false** | **was-aggregated** } }

3. 设置路由策略的动作。

   - 设置路由策略的整体动作。

     set policy policy-statement *policy-name* **then** { **accept** | **reject** }

   - 设置指定策略内容的动作。

     ConnetOS# **set policy policy-statement** *policy-name* **term** *term-name* **then** { **accept** | **aggregate-brief-mode** { **false** | **true** } | **aggregate-prefix-len** *aggregate-prefix-len* | **as-path-expand** *as-path-expand* | **as-path-prepend** *as-path-prepend* | **community** *community-name* | **community-add** *community-add-name* | **community-del** *community-del* | **external-type** *external-type-number* | **localpref** *localpref* | **med** *med* | **med-remove** { **false** | **true** } | **metric** *metric* | **nexthop4** *nexthop4-address* | **nexthop4-var** { **peer-address** | **self**} | **origin** *origin-attribute* | **reject** | **tag** *tag-value* }

4. 应用路由策略。

   ConnetOS# **set protocols** { **bgp** | **ospf4** } { **export** | **import** } *policy-name*

5. 提交配置。

   ConnetOS# **commit**

流量策略配置

ConnetOS支持如下的完善的流量策略，对用户流量进行管理。

策略	主要功能
QoS	Priority mapping（DSCP、IEEE 802.1p、Trust-port） Queue scheduling（SP、WDRR、SP+WDRR）
Firewall	Filter（L2、L3、L4 fields） Forwarder（classifying、mirroring、policing、switching、routing）

QoS配置

简介

QoS概述

随着网络技术的发展，互联网中的业务种类越来越多。传统的E-mail、WWW、文件传输等业务，新兴的多媒体游戏、电话会议、在线视频、网络直播等业务，对网络带宽、时延、丢包率等的要求各不相同。同时随着网络的普及，流量激增，更容易产生网络拥塞、增大转发时延，甚至会产生丢包，导致业务质量下降甚至不可用。如何针对不同的业务提供不同层次的服务，同时解决网络拥塞提供更完善的服务迫在眉睫。

QoS技术就是在这种背景下发展起来的。QoS是Quality of Service（服务质量）的简称，根据用户的要求分配和调度资源，对不同的数据流提供不同的服务质量：对实时性强且重要的数据报优先处理；对于实时性不强的普通数据报文，提供较低的处理优先级，网络拥塞时甚至丢弃。

QoS的应用可以通过保证传输的带宽，降低转发时延、丢包率和时延抖动等措施，来提高网络服务质量。

QoS服务模型

网络应用是端到端的通信，两个主机进行通信，中间可能要跨越多个物理网络，经过多台设备，要实现端到端的QoS，就必须从全局考虑。QoS服务模型就是研究采用什么模式实现全局的服务质量保证。

目前QoS有三种服务模型：

- Best-Effort Service：尽力而为服务模型
- Integrated Service：综合服务模型，简称Int-Serv
- Differentiated Service：差分服务模型，简称Diff-Serv

Best-Effort服务模型

Best-Effort是最简单的QoS服务模型。应用程序可以在任何时候，发出任意数量的报文，而不需要通知网络。而网络则尽最大的可能来发送报文，但对时延、可靠性等性能不提供任何保证。

Best-Effort服务模型

Int-Serv模型是指应用程序在发送报文前，首先通过信令向网络描述流量参数，申请带宽。在确定网络已经为其预留资源后，再发送报文。Int-Serv模型下，可以保证报文的丢包率、延迟等要求。

Diff-Serv服务模型

Diff-Serv是一种基于报文流的QoS模型。在Diff-Serv模型中，应用程序发出报文前，通过设置报文的QoS参数信息，来告知网络节点它的QoS需求。网络根据每个报文流指定的QoS参数信息来提供差分服务，即对报文的服务等级划分，有差别地进行流量控制和转发，提供端到端的QoS保证。

目前，应用最广泛的是Diff-Serv模型。ConnetOS的QoS功能就是基于Diff-Serv模型实现的。

基于Diff-Serv模型的QoS实现

在Diff-Serv模型中，进入网络中的流量被分成不同的类，同一类的流量在网络中被聚合起来统一管理发送，保证相同的QoS服务指标。不同的类享受不同的处理，尤其是当网络出现拥塞时不同的类会享受不同的优先处理，从而得到不同的丢弃率、时延以及时延抖动。

在Diff-Serv模型中，常用到的技术包括：

- 流分类和流标记（Classification and Marking）

  将数据包分为不同的类别，并设置为不同的优先级。

  - 流分类：采用一定的规则识别符合某类特征的报文，将数据包分为不同的类别（并不修改原来的数据包优先级信息）。
  - 流标记：将数据包设置为不同的优先级（会修改原来的数据包优先级信息），通过优先级映射和重标记优先级实现。

- 流量监管和流量整形（Policing and Shaping）和接口限速

  将业务流量限制在特定的带宽。当业务流量超过额定带宽时，超过的流量将被丢弃或缓存。

  - 将超过的流量丢弃的技术称为流量监管。
  - 将超过的流量缓存的技术称为流量整形（流量整形效果需视其缓存的大小而定）。

- 拥塞管理（Congestion management）

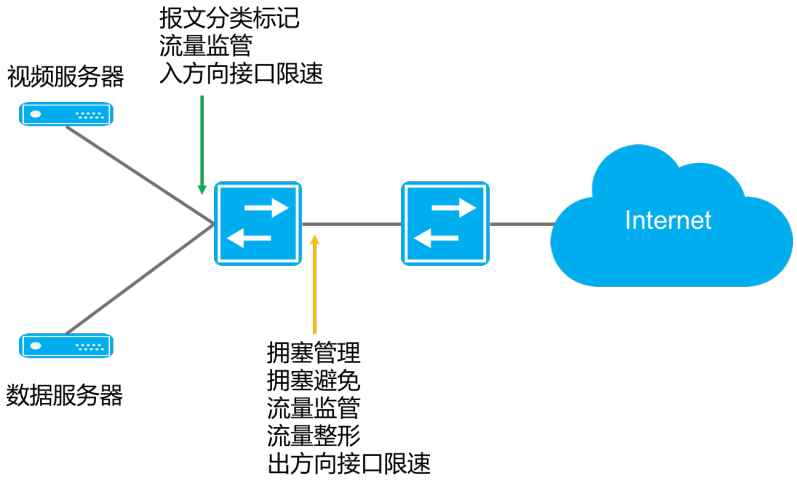
  在网络发生拥塞时，使用合适的队列调度机制，优先保证某种报文的QoS服务指标。

- 拥塞避免（Congestion avoidance）

  监督网络资源的使用情况，当发现拥塞有加剧的趋势时采取主动丢弃报文的策略，通过调整流量来解除网络的过载。通常作用在接口出方向。

流分类和流标记是实现差分服务的前提和基础；流量监管、流量整形、拥塞管理和拥塞避免从不同方面对网络流量及其分配的资源实施控制，是提供差分服务的具体体现。

QoS技术在网络中的应用位置。



ConnetOS支持的QoS功能

QoS在ConnetOS中的实现流程

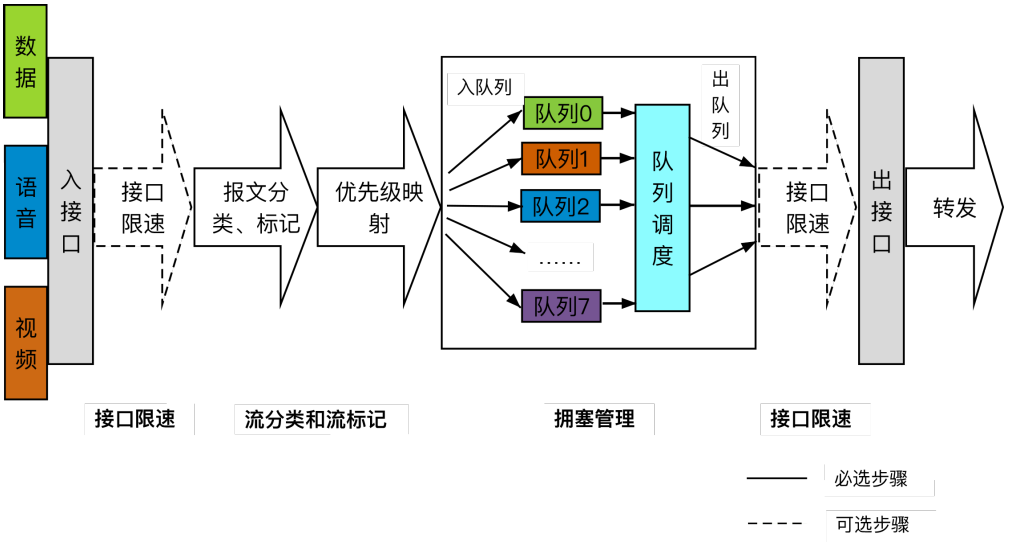
ConnetOS在入接口完成流量分类和标记工作，在出接口进行队列调度来分配资源和和控制流量。

具体实现过程如下：

1. （可选）在入接口对报文进行速度限制，保证网络流量在一个合理的范围内。
2. 在入接口将报文携带的优先级或者接口优先级映射为内部优先级，按照优先级映射表确定报文进入哪个转发队列。

3. 根据队列调度算法发送各个转发队列中的报文。
4. （可选）根据网络流量情况，对出接口的报文进行限速。

QoS处理流程



缺省情况下，ConnetOS上的QoS功能是关闭的，即不区分优先级，设备对所有的报文等同处理。

接口限速

接口限速，对通过接口的全部报文流量速率进行限制，以保证带宽不超过规定大小。入方向与出方向的接口限速属于并列关系，用户可以根据需要同时配置，也可以单独配置。

如果不限制用户发送的流量，大量用户不断突发的数据会使网络更拥挤。通过配置入方向的接口限速，可以将通过某个接口进入网络的流量限制在一个合理的范围内。

若需要对接口出方向所有流量进行控制时，可以配置出方向的接口限速。

优先级映射

优先级用于标识报文传输时的优先程度，可以分为：

- 报文携带优先级：根据公有标准和协议生成，是报文自身的优先等级。
- 设备调度优先级：又称为本地/内部优先级，是设备内部区分报文服务等级的优先级。

报文进入设备后，设备会根据相应规则选择报文的内部优先级，为队列调度和拥塞控制服务。

优先级映射用来实现报文携带优先级与内部优先级之间的转换。对于进入设备的报文，设备将报文携带的优先级或者接口优先级映射为内部优先级，然后根据内部优先级与队列之间的映射关系确定报文进入的队列，并可以根据配置修改报文发送时所携带的优先级，以便其他设备根据报文的优先级提供相应的QoS服务。

用户可以在不同的网络中使用不同的优先级字段，例如：VLAN网络中使用802.1p，IP网络中使用DSCP等。当报文经过不同网络时，为了保持报文的优先级，需要在连接不同网络的设备上配置优先级字段的映射关系。当设备接收报文时，优先级字段（如802.1p、DSCP）被映射为内部优先级；设备发出报文时，将内部优先级映射为某种外部优先级字段。 设备提供了优先级映射表，分别对应相应的优先级映射关系。

通常情况下，可以通过查找缺省优先级映射表来为报文分配相应的优先级。如果缺省优先级映射表无法满足用户需求，可以根据实际情况对映射表进行修改。

入口报文携带dscp	入口报文携带8021.p	出口队列
0～7	0	0
8～15	1	1
16～23	2	2
24～31	3	3
32～39	4	4
40～47	5	5
48～55	6	6
58～63	7	7

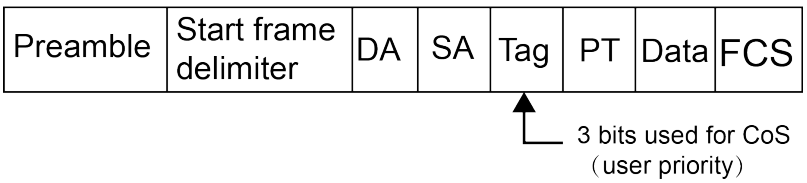
流分类

流分类就是根据报文的优先级字段（如802.1p、DSCP），对报文进行分类，以识别出具有不同优先级或服务等级特征的流量，实现外部优先级和内部优先级之间的映射。

流分类过程实际上就是信任接口的上行报文携带的优先级标记，并进行优先级映射（即根据优先级映射表，将上行报文携带的QoS优先级统一映射到设备内部的服务等级。

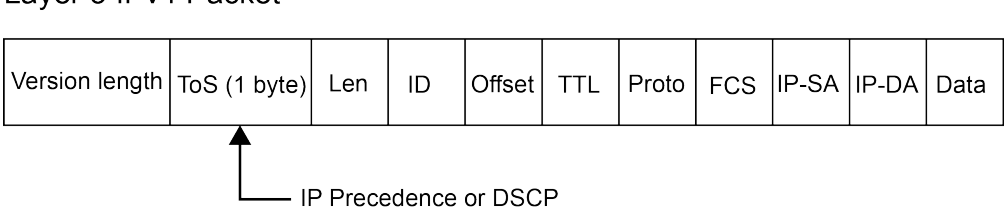
CoS：Class of Service 服务级别，L2 802.1Q帧携带的分类信息，在帧头的Tag字段中占3bits，称为用户优先级，范围为0～7。

CoS优先级



ToS：Type of Service 服务类型，L3 IPv4包头携带的一个字节的字段，标记IP包的服务类型， ToS字段内可以是IP Precedence值，也可以是DSCP值。

ToS优先级



IP Precedence： IP 优先级， L3 IP包头携带的分类信息，共占3bits，范围为0～7。

DSCP： Differentiated Services Code Point 差别化业务编码点， L3 IP包头携带的分类信息，共占6bits，范围为0～63，向下兼容IP Precedence。

流标记

为保证被分类报文对应的DSCP值能够传递给网络上的下一跳设备，需要通过流标记动作为报文写入QoS信息，或使用Trust方式直接保留报文中QoS信息。

缺省情况下，流标记总是将对应的DSCP值转化成QoS信息，然后写入到报文CoS字段（对于非IP 报文）、DSCP字段或者IP-precedence 字段（对于IP 报文）中。

接口信任模式

下游设备可以选择使用上游设备的分类结果，也可以按照自己的分类标准对报文重新进行分类。如果选择使用上游设备的分类结果，则表示该设备信任上游设备的分类结果，即信任（trust）从连接上游设备的接口接收的报文所携带的QoS标记。因此，设备在实现QoS优先级映射时，可以选择信任接口的上行报文携带的优先级标记（如DSCP、IP Precedence、802.1p），这种模式就称为接口信任模式。

入接口上，既可以设置流分类，也可以配置为信任接口。

配置成信任接口，信任报文cos中携带的标记，根据code-point map映射到出口队列。

目前，ConnetOS支持的如下的信任模式:

- dscp：IP报文按照报文dscp以及配置的code-point映射到出口队列。非IP报文按照“trust ieee-802.1”方式处理。
- ieee-802.1：tagged 根据报文的cos以及配置的code-point map映射到出口队列。untagged 根据接口默认cos以及配置的code-point map映射到出口队列。
- inet-precedence：IP报文按照报文ip-pre以及配置的code-point映射到出口队列。非IP报文按照“trust ieee-802.1”方式处理。
- trust port：任何报文都按照接口默认cos以及配置的code-point map映射到出口队列。

队列

Queueing即队列，负责将流量送往接口的某个转发队列中，送往接口的不同转发队列的报文将获得不同等级的服务。ConnetOS上，每个接口出方向支持8个队列，以队列索引号进行标识，分别为：0、1、2、3、4、5、6、7。编号越大，优先级越高。

ConnetOS根据本地优先级和队列之间的映射关系，自动将分类后的报文流送入各个队列，然后根据配置的队列调度算法进行调度。通过绑定流分类和转发队列，来确定不同的流量被送往不同的转发队列。

队列指的是在缓存中对报文进行排序的逻辑。当流量的速率超过接口带宽或超过为该流量设置的带宽时，报文就以队列的形式暂存在缓存中。报文离开队列的时间、顺序，以及各个队列之间报文离开的相互关系则由队列调度算法决定。

队列调度技术

拥塞管理是通过调整报文的转发次序，来满足时延敏感业务高QoS服务的一种流量控制机制。对于拥塞管理，一般采用队列技术来处理。

ConnetOS支持以下的队列调度算法:

- SP（Strict Priority）：严格优先级调度
- WDRR（Weighted Deficit Round Robin）：带赤字的加权轮询调度
- SP+WDRR

SP

SP调度就是严格按照队列优先级的高低顺序进行调度。只有高优先级队列中的报文全部调度完毕后，低优先级队列才有调度机会。在SP调度中，加权值为0。

在报文出队列的时候，首先让高优先队列中的报文出队并发送，直到高优先队列中的报文发送完；然后发送低优先队列中的报文。在调度低优先级队列时，如果高优先级队列又有报文到来，则会优先调度高优先级队列。

SP调度的缺点是：拥塞发生时，如果高优先级队列中长时间有报文存在，那么低优先级队列中的报文就会一直得不到调度机会。

WDRR

WDRR是以报文字节数为权重的调度算法，可以避免队列的平均报文长度变化时，用户无法通过WRR获取想要带宽的问题。当为某一个队列分配的权重值低于0时，该队列的权重就变为赤字，同时也影响下一次调度对队列赋予的权重。即该队列的新的权重会减去前一次产生的赤字，这样可以避免由于报文长度不等而产生的非预期的调度。

WDRR为每个队列设置一个计数器Deficit，Deficit初始化为一次调度允许的最大字节数，一般为Weight*MTU。每次轮询到一个队列时，该队列输出一个报文且计数器Deficit减去报文长度，如果报文长度超过了队列的调度能力，WDRR调度允许Deficit出现负值，以保证长报文也能够得到调度，但下次轮循调度时该队列将不会被调度。上一轮调度后的Deficit值作为下一轮调度的Deficit。直到计数器为0或负数时停止调度该队列，但继续调度其他计数器不为0的队列。当所有队列的计数器都为0或负数时，所有计数器的Deficit都加上Weight*MTU，开始新一轮调度。

WDRR调度避免了采用SP调度时低优先级队列中的报文可能长时间得不到服务的缺点，也避免了各队列报文长度不等或变化较大时，WRR调度不能按配置比例分配带宽资源的缺点。

但是，WDRR调度也具有低延时需求业务（如语音）得不到及时调度的缺点。

SP+WDRR

SP调度、WDRR调度各有优缺点。单纯采用SP调度时，低优先级队列中的报文长期得不到带宽，而单纯采用WDRR调度时低延时需求业务得不到优先调度。

SP+WDRR调度是指在队列调度中选择SP和WDRR算法共同参与运算的方法，即在高优先级队列的所有报文被调度完成后，才根据选择WDRR算法对其他队列进行调度。这样既保证最高优先级的队列能得到优先调度，又避免了其他低优先级队列长时间得不到调度的问题。

配置QoS功能

定义流分类

1. 进入配置模式。

   ConnetOS> **configure**

2. 定义流分类模版，用于对进入设备的流量进行分类。

   ConnetOS支持定义任意数量的流分类模版，但是必须绑定到接口才会生效。

   ConnetOS# **set class-of-service classifier** *classifier-name*

3. 配置指定流分类模板的优先级信任模式。

   ConnetOS# **set class-of-service classifier** *classifier-name* **trust-mode** { **dscp** | **ieee-802.1** | **trust-port** }

4. (可选）配置接口优先级。只有当优先级映射模式为信任接口时，才需要配置接口优先级。

   ConnetOS# **set interface gigabit-ethernet** *interface-number* **cos priority** *priority-value*

   配置接口优先级后，从该接口流入的流量将以接口优先级查找优先级映射表得到出口队列。

5. 配置转发队列，设置指定流分类在出接口进行报文转发时的转发队列。

   ConnetOS# **set class-of-service forwarding-class** *forwarding-class* **queue-num** *queue-numer*

   ConnetOS支持8个队列（0～7），数字越大，优先级越高。

6. 配置指定流分类模版的优先级映射表。

   ConnetOS# **set class-of-service classifier** *classifier-name* **forwarding-class** *forwarding-class* [**code-point** *code-point*]

   code-point 用于标识优先级，不同的信任模式下标识不同的优先级。比如：**trust-mode** 为dscp时，**code-point** 表示DSCP值。

7. 将流分类模版绑定到报文的入接口。

   ConnetOS# **set class-of-service interface** *interface-name* **classifie** *classifier-name*

   流分类模版需要绑定到报文的入接口，这样报文才会在出接口按照优先级到队列映射表映射到相应的出口队列。

8. 提交配置

   ConnetOS# **commit**

配置拥塞管理

配置拥塞管理后，当网络发生拥塞时，设备将按照指定的队列调度算法决定转发报文时的处理顺序，从而达到高优先级报文被先调度的目的。

拥塞管理需要配置在报文的出接口。

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置指定接口的队列调度算法

   ConnetOS# **set interface gigabit-ethernet** *interface-number* **cos schedule mode** { **sp** | **sp+wdrr** | **wdrr** }

   如果不配置调度算法，队列将按照设备默认调度算法（WDRR调度）进行调度。

3. 配置队列的调度权重。

   ConnetOS# **set interface gigabit-ethernet** *interface-number* **cos schedule queue** *queue-number* [**weight** *weight-value*]

   如果不配置权重，权重值就是0。在SP+WDRR调度中，执行SP算法的队列，都不需要配置权重。

4. 提交配置

   ConnetOS# **commit**

检查配置结果

查看QoS的相关配置信息:

```
ConnetOS# show class-of-service
Waiting for building configuration.
   forwarding-class fd1 {
       queue-num: 1
   }
   classifier c1 {
       trust-mode: "trust-port"
       forwarding-class fd1 {
            code-point 7
        }
    }
    interface "te-1/1/13" {
        classifier: "c1"
```

查看报文入接口的QoS相关配置信息:

```
ConnetOS# show interface gigabit-ethernet te-1/1/13
Waiting for building configuration.
    cos {
        priority: 4
    }
```

查看报文出接口的QoS相关配置信息:

```
ConnetOS# show interface gigabit-ethernet te-1/1/15
Waiting for building configuration.
   ether-options {
   }
   cos-schedule {
       mode: "sp+wdrr"
       queue 1 {
           weight: 30
       }
       queue 2 {
           weight: 50
       }
   }
```

Firewall配置

Firewall简介

通过配置firewall功能，可以实现对特定报文进行过滤、重定向、策略路由、流镜像功能。

过滤规则组term，是一条或者多条规则的集合，用于识别报文流，对流量进行分类。规则是指描述报文匹配条件的判断语句，这些条件可以是报文的源地址、目的地址、端口号等。设备依照这些规则识别出特定的报文，并根据预先设定的策略对其进行处理。

报文过滤就是对符合过滤规则的流量配置过滤动作，从而达到过滤的目的。

流量重定向就是将符合流分类的流重定向到其他地方进行处理。目前支持将二层转发报文重定向到接口，即对于接收到的需要某个接口处理的报文，可以通过配置重定向到此接口。

策略路由通过识别不同的网络数据包，然后按照预先设定好的策略进行转发，从而可以有效的控制网络用户数据包的流向和行为。策略路由位于IP层，在做IP转发前，如果报文命中某个策略路由对应的规则，则要进行相应的策略路由的动作，动作包括重定向到指定下一跳，以及remark标记（如TOS、IP优先级或DSCP），然后根据重定向的下一跳代替报文的目的IP去查FIB表，做IP转发。

流镜像就是将符合过滤条件的报文流量镜像到指定端口。通过制定不同的过滤条件，对报文类型进行了精确区分，获得更精确的统计信息。

配置报文过滤功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 设置过滤规则组。

   ConnetOS# **set firewall term** *term-name*

3. 配置过滤规则组，对需要过滤的流量进行分类。请根据网络环境需要，选择执行以下命令，进行流量的分类

   - 按照报文携带的COS值，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **cos value** *priority-value*

   - 按照目的IP地址，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **dest-ipv4 network** *ipv4-network-address*

   - 按照目的MAC地址，对报文行分类

     ConnetOS# **set firewall term** *term-name* **dest-mac hwaddr** *dest-mac-address*

   - 按照报文的DSCP值，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **dscp value** *dscp-value*

   - 按照以太网报文类型，对流量进行分类

     ConnetOS# **set firewall term** *term-name* **ether-type** { **name** { **arp** | **ipv4** | **rarp** } | **number** *ether-type-number* }

   - 按照目的端口号，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **l4-dest-port** { **name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *dest-port-numbe* | **range** *port-range* }

   - 按照源端口号，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **l4-source-port** { **name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *source-port-number* | **range** *port-range* }

   - 按照协议类型，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **protocol** { **name** { **ah** | **dstopts** | **egp** | **esp** | **fragment** | **gre** | **hop-by-hop** | **icmp** | **igmp** | **ipip** | **no-next-header** | **ospf** | **pim** | **routing** | **rsvp** | **sctp** | **tcp** | **udp** } | **number** *protocol-number* }

   - 按照源IP地址，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **source-ipv4 network** *ipv4-network-address*

- 按照源MAC地址，对报文行分类

  ConnetOS# **set firewall term** *term-name* **source-mac hwaddr** *source-mac-address*

- 按照报文所属VLAN，对报文进行分类

  ConnetOS# **set firewall term** *term-name* **vlan number** *vlan-id*

  同一个过滤规则组，可以配置多种过滤规则。报文在进行规则匹配时，只要命中了一个过滤规则就不再进行匹配。

4. 配置对报文的过滤策略。关联过滤规则和动作，对报文进行丢弃或转发处理。

   一个过滤策略，可以同时绑定多个过滤规则组。

   ConnetOS# **set firewall filter** *filter-name* **matching term** *term-name* [**action** { **discard** | **forward** }]

5. 应用过滤策略，对接口或VLAN上的报文进行过滤。

   ConnetOS# **set firewall filter** *filter-name* { **input** | **output** } { **gigabit-ethernet** | **vlan-interface** } *interface-name*

6. 提交配置。

   ConnetOS# **commit**

配置转发策略

1. 进入配置模式。

   ConnetOS> **configure**

2. 设置过滤规则组。

   ConnetOS# **set firewall term** *term-name*

3. 配置过滤规则组，对需要过滤的流量进行分类。请根据网络环境需要，选择执行以下命令，进行流量的分类

   - 按照报文携带的COS值，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **cos value** *priority-value*

   - 按照目的IP地址，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **dest-ipv4 network** *ipv4-network-address*

   - 按照目的MAC地址，对报文行分类

     ConnetOS# **set firewall term** *term-name* **dest-mac hwaddr** *dest-mac-address*

   - 按照报文的DSCP值，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **dscp value** *dscp-value*

   - 按照以太网报文类型，对流量进行分类

     ConnetOS# **set firewall term** *term-name* **ether-type** { **name** { **arp** | **ipv4** | **rarp** } | **number** *ether-type-number* }

   - 按照目的端口号，对报文进行分类

     ConnetOS# **set firewall term** *term-name* **l4-dest-port** { **name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *dest-port-number* | **range** *port-range* }

- 按照源端口号，对报文进行分类

  ConnetOS# **set firewall term** *term-name* **l4- source-port** { **name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *source-port-number* | **range** *port-range* }

- 按照协议类型，对报文进行分类

  ConnetOS# **set firewall term** *term-name* **protocol** { **name** { **ah** | **dstopts** | **egp** | **esp** | **fragment** | **gre** | **hop-by-hop** | **icmp** | **igmp** | **ipip** | **no-next-header** | **ospf** | **pim** | **routing** | **rsvp** | **sctp** | **tcp** | **udp** } | **number** *protocol-number* }

- 按照源IP地址，对报文进行分类

  ConnetOS# **set firewall term** *term-name* **source-ipv4 network** *ipv4-network-address*

- 按照源MAC地址，对报文行分类

  ConnetOS# **set firewall term** *term-name* **source-mac hwaddr** *source-mac-address*

- 按照报文所属VLAN，对报文进行分类

  ConnetOS# **set firewall term** *term-name* **vlan** number vlan-id

  同一个过滤规则组，可以配置多种过滤规则。报文在进行规则匹配时，只要命中了一个过滤规则就不再进行匹配。

4. 设置转发策略。

   ConnetOS# **set firewall forwarder** *forwarder-name*

5. 关联转发策略和过滤规则组，符合过滤规则的报文按照指定转发策略进行转发。

   ConnetOS# **set firewall forwarder** *forwarder-name* **matching term** *term-name*

6. 设置报文的匹配模式。

   matched：对符合过滤规则的报文按照转发行为进行转发。

   unmatched：对不符合过滤规则的报文按照转发行为进行转发。

   ConnetOS# **set firewall forwarder** *forwarder-name* **match-mode** { **matched** | **unmatched** }

7. 根据网络需要，选择执行以下命令，设置报文转发行为。

   - 设置报文分类动作，修改报文的设备优先级。

     ConnetOS# **set firewall forwarder** *forwarder-name* **action classifying** { **new-cos** *cos-modify-value* | **new-dscp** *dscp-modify-value* }

   - 设置对报文进行镜像。

     ConnetOS# **set firewall forwarder** *forwarder-name* **action mirroring interface** { **aggregate-ethernet** *ae-interface-name* | **gigabit-ethernet** *ge-interface-name* }

   - 设置对报文进行三层转发。

     ConnetOS# **set firewall forwarder** *forwarder-name* **action routing** { **mode** { **load-balance** | **redundancy** } | **nexthopv4 address** *ipv4-address* | **vlan-interface** *vlan-interface* }

   - 设置对报文进行二层转发。

     ConnetOS# **set firewall forwarder** *forwarder-name* **action switching interface** { **aggregate-ethernet** *ae-interface-name* | **gigabit-ethernet** *ge-interface-name* }

8. 执行命令，将转发策略应用到入接口上。接口上接收的报文将按照转发策略进行转发。

   ConnetOS# **set firewall forwarder** *forwarder-name* **input** { **gigabit-ethernet** *ge-interface-name* | **vlan-interface** *vlan-interface-name* }

9. 提交配置。

   ConnetOS# **commit**

查看配置结果

查看过滤规则组的相关配置信息:

```
ConnetOS# show firewall term term1
Waiting for building configuration.
    protocol {
        name gre
}
```

查看过滤策略的相关配置信息:

```
ConnetOS# show firewall filter f1
Waiting for building configuration.
    term t1 {
        action: "discard"
    }
    input {
        gigabit-ethernet "te-1/1/1"
        gigabit-ethernet "te-1/1/40"
    }
```

查看转发策略的相关配置信息:

```
ConnetOS# show firewall forwarder fd1
Waiting for building configuration.
    match-mode: "matched"
    matching {
        term t1
    }
    action {
        routing {
            mode: "load-balance"
        }
    }
    input {
        gigabit-ethernet "te-1/1/1"
    }
```

ISS堆叠配置

ISS（Intelligent Stacking System），是云启针对云业务环境研发的堆叠功能。具有如下特点:

- Linux shell界面独立。
- 分布式链路聚合/路由转发。
- 业务特性堆叠。

- 协议热备份。
- 本地转发优先。
- 堆叠分裂检测。

简介

概述

随着数据中心数据访问量的逐渐增大，对交换机提出了高密度端口、高可靠性、高性能的要求，而单台交换机由于存在单点异常、端口数量等问题已经无法满足数据中心的需求，交换机虚拟化技术（如堆叠）应运而生。

ConnetOS支持的ISS（Intelligent Stacking System）堆叠功能，是指将两台及以上的交换机组合在一起，从逻辑上组成一台交换机。用户通过对这台“逻辑交换机”的管理，实现对堆叠中所有交换机的管理。

通过堆叠，可以实现网络高可靠性和网络大数据量转发，同时简化网络管理。ISS堆叠主要有以下优点：

- 简化运维。整个堆叠系统被作为一台交换机进行管理，配置命令直接在堆叠内的所有交换机上生效；用户可以通过登录堆叠内的任意一台交换机，对堆叠内的所有交换机进行统一配置和管理。
- 高可靠性。堆叠内的交换机互为备份，同时，利用跨设备的Eth-Trunk实现跨设备的链路冗余备份。
- 强大的网络扩展能力。通过组建堆叠，在不改变网络拓扑的情况下，扩展端口数量、带宽和处理能力。

常见概念

角色

ISS堆叠中的单台设备称为成员设备。成员设备按照功能不同，分为两种角色：

- Master：负责整个堆叠的运行、维护和管理，由角色选举产生。一个堆叠系统中只能有一台Master设备。
- Slave：作为Master设备的备份设备运行。当Master重启时，Slave会被选举为Master。

Member ID

成员设备ID，用来标识和管理成员设备。堆叠系统中成员ID是唯一的。

ISS链路

成员设备之间用于互连形成ISS的链路。

堆叠接口

被配置为堆叠模式的物理接口，用于堆叠成员交换机之间的连接。

成员设备间可以配置多个堆叠接口，进行负载分担，但是控制报文只能从控制接口转发。

优先级

参与堆叠角色选举，用来确定成员交换机的角色。

优先级值越大表示优先级越高，当选为Master的可能性越大。

ISS MAC

堆叠MAC地址，即堆叠系统的对外体现的MAC地址。

堆叠系统刚刚建立时，ISS MAC和Master设备的MAC地址一致。当Master设备重启，Slave被选举为Master时，ISS MAC不会重新选举，保持不变。

堆叠报文

ISS用到的主要报文有：

- Hello报文：点对点报文，在相邻设备间交互，携带本设备所收集到的所有的设备信息、优先级信息和其它上下文信息。
- Elect选举报文：点对点报文，设备仅仅携带用于选举的相关信息，如设备MAC地址，优先级，设备运行时间等。
- ElectAck：Elect选举回应报文。
- Anno通告报文：竞选结果通告报文，Master发送宣布竞选结果，Slave回复ACK进行确认。
- AnnoAck：Anno通告回应报文。
- Urgent报文：广播报文，用于ISS系统紧急事件的通告，如堆叠口DOWN。

ConnetOS支持的堆叠特性

在为了防止堆叠建立后，大量的报文跨越中间有限带宽的堆叠链路进行转发导致链路满载现象出现，ISS采取报文转发本地优先机制。对于等价路由或聚合端口，如果本地有出口时，报文直接由本地转发，不会经过堆叠链路，只有当本地出口down时，才经堆叠链路由另一台设备进行转发。

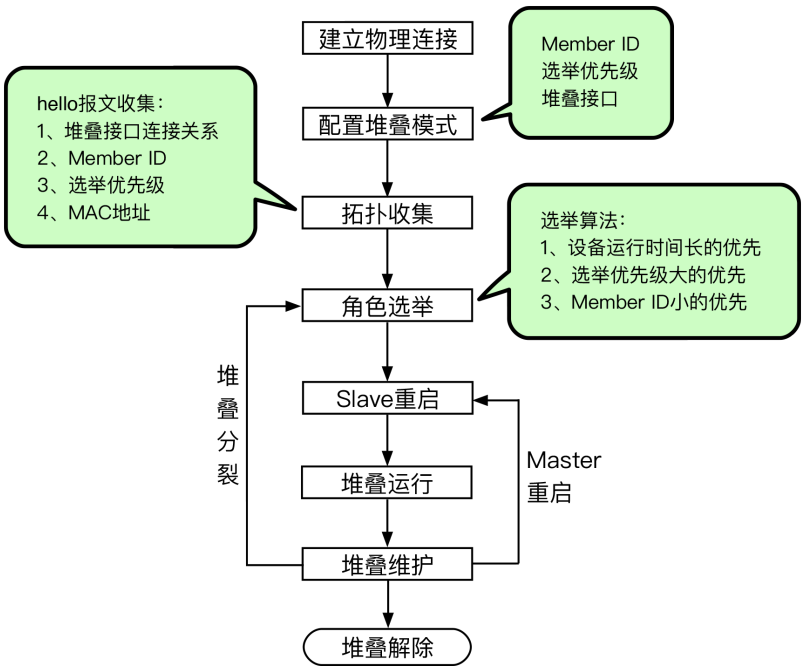
在ISS堆叠开始运行后：

- 在maste设备上的配置，在堆叠系统内的所有交换机上同时生效。
- 远程登录界面都能进行配置，但是对管理网络接入有要求：单线接入时必须接入到Master。
- 串口界面只有Master能做配置，Slave只能做查询。
- 切换ISS模式后，以下特性暂时不能使用：LLDP、SNMP、ATP。

ISS堆叠的工作原理

ISS堆叠系统中的成员设备通过Hello报文传递本设备的信息到相邻设备，选举出ISS堆叠系统中各个设备的角色（Master或者Slave），建立ISS堆叠拓扑数据库，并管理整个拓扑关系.

整个ISS堆叠的生命周期分为如下几个阶段：



堆叠的建立过程

堆叠建立的主要过程为：

1. 保证用于堆叠的设备，正常通信。
2. 将设备配置为堆叠模式，配置Member ID、选举优先级、堆叠接口。
3. 堆叠成员设备通过Hello报文交换信息，收集拓扑关系。

   Hello报文会携带：堆叠接口连接关系、Member ID、选举优先级、设备的MAC。

   成员设备在收到邻居的拓扑信息之后，更新本地的拓扑信息。
4. 当所有成员设备上都收集到完整的拓扑信息后，进行角色选举。

   角色选举时，从第一条规则开始判断。满足第一条规则，Master直接会被选出。否则会继续下一条规则的比较。

   (a) 系统运行时间长的优先。

   (b) 选举优先级高的优先。

   (c) Member ID小的优先。

   只有完成角色选举，选出Master设备后，堆叠系统才能正常运行。
5. 完成Master和Slave的选举后，Slave进行重启。重启后自动同步Master的配置信息。
6. 完成重启后，堆叠系统正式开始运行。

   堆叠运行的过程中，进行业务的转发、实时同步数据和配置。

堆叠维护

堆叠运行过程中，成员设备之间会定期交换Hello报文。如果持续一定周期未收到直接邻居的Hello报文，则认为该成员设备的Hello报文超时，堆叠会将超时设备从拓扑中隔离出来，并更新拓扑数据库。

如果Master重启，Slave会直接变为Master。

如果成员设备之间的链路断开等问题导致堆叠分裂，会重新进行成员角色的选举。

堆叠检测

堆叠分裂是指，堆叠系统形成后，由于ISS链路故障，导致ISS中两相邻成员设备上不连通，一个ISS变成两个ISS的过程称为ISS分裂（split）。

MAD（Multi-Active Detection，多Active检测），当ISS分裂后会产生多个ISS Master，MAD是用来检测ISS分裂的机制。

配置堆叠

配置堆叠基本功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置堆叠设备的成员ID。

   ConnetOS# **set member-id** *member-id*

   一个堆叠系统内，成员设备的Member ID不能相同。

3. 配置堆叠设备的选举优先级。

   ConnetOS# **set member-id** *member-id* **priority** *priority-number*

   缺省情况下，选举优先级为1。数值越大，优先级越高

4. 配置堆叠接口。

   ConnetOS# **set interface gigabit-ethernet** *interface-name* **iss-port enable** { **false** | **true** }

   缺省情况下，堆叠接口没有使能。

5. 提交配置。

   ConnetOS# **commit**

配置MAD检测功能

1. 进入配置模式。

   ConnetOS> **configure**

2. 使能ISS MAD的检测功能。

   ConnetOS# **set member-id** *member-id* **mad enable** { **false** | **true** }

   缺省情况下，MAD的检测功能没有使能

3. 配置用于ISS MAD的检测的接口。

   ConnetOS# **set member-id** *member-id* **mad interface** *interface-name*

4. 配置用于ISS MAD检测时不进行shutdown的接口。

   ConnetOS# **set member-id** *member-id* **mad excluded-interface** *interface-name*

5. 提交配置。

   ConnetOS# **commit**

查看堆叠系统情况

配置完成后，可以查看堆叠系统的信息及配置情况。

查看堆叠系统中的成员设备信息:

```
ConnetOS 1> show iss
Member ID   Role     Priority   Device MAC          ISS MAC             Hostname
---------   ------   --------   -----------------   -----------------   --------------
↪--
1           Master   1          00:03:0f:64:da:5f   00:03:0f:64:da:5f   BJ-YUNQI-
↪C1020-31.Int
2           Slave    1          00:03:0f:64:da:53   00:03:0f:64:da:5f   BJ-YUNQI-
↪C1020-32.Int
```

查看堆叠系统中成员设备的配置信息:

```
ConnetOS 1> show iss configuration
Member ID   ISS Link Status   Interface        Interface Status   Neighbour
---------   ---------------   --------------   ----------------   --------------
1           Up                qe-1/1/49        Up                 qe-2/1/49
                              qe-1/1/52(*)     Up                 qe-2/1/52

2           Up                qe-2/1/49        Up                 qe-1/1/49
                              qe-2/1/52(*)     Up                 qe-1/1/52

-----------------------------------------
 * indicates the control interface of ISS.
```

查看MAD的检测和处理信息:

```
ConnetOS 1> show iss mad
Member ID   Management   MAD State            MAD interface         Neighbor
↪      Excluded interfaces
---------   ----------   ------------------   -------------------   ------------------
↪--    -------------------
1           Enabled      Detect               qe-1/1/54             qe-2/1/54
↪      N/A
2           Enabled      Detect               qe-2/1/54             qe-1/1/54
↪      N/A
```

网络分析和诊断

ConnetOS针对网络运维的常见问题，提供如下的功能：

- 对转发平面所有数据包（转发的或丢弃的）进行监视和分析；
- 实时捕获被设备丢弃的数据包，并记录丢包原因；
- 根据五元组实时计算网络转发路径；
- 具备端口拥塞感知和实时上报能力；

- 高精度端口统计，最高精度1秒。

Ping和Traceroute

在日常的系统维护中，用户可以使用Ping功能和Tracert功能来检查当前网络的连接情况。

Ping功能

Ping功能是基于ICMP协议实现的：

源端向目的端发送ICMP回显请求(ECHO-REQUEST)报文后，根据是否收到目的端的ICMP回显应答(ECHO-REPLY)报文来判断目的端是否可达。对于可达的目的端，再根据发送报文个数、接收到响应报文个数来判断链路的质量，根据ping报文的往返时间来判断源端与目的端之间的“距离”。

ping 命令是最常见的用于检测网络设备可访问性的调试工具，它使用ICMP报文信息可以来检测：

- 远程设备是否可用。
- 与远程主机通信的来回旅程（round-trip）的延迟。
- 报文（packet）的丢失情况。

Traceroute功能

运维模式下支持traceroute操作。

tracert 命令用来测试数据包从发送主机到目的地所经过的网关，主要用于检查网络连接是否可达，以及分析网络什么地方发生了故障。

系统日志

简介

设备运行过程中，日志模块会对设备运行中的各种情况进行记录，从而形成Syslog系统日志。

当设备出现异常或故障时，通过系统日志，用户可以查看设备的运行状态、分析网络的状况以及定位问题发生的原因，为系统诊断和维护提供依据。

日志的格式

系统日志的各个字段之间用空格隔开，标准格式为：

事件发生的时间 哪台主机的日志 产生日志信息的系统 系统发生的事件

如:

```
Apr 12 2017 17:53:57 ConnetOS local0.debug : [1][rtrmgr] user admin requested non-
↪exclusive config
```

日志facility

facility，是用来指定产生日志的程序模块。syslog服务器上根据facility的值来对日志进行区分。
本地设备的facility的标识为local0～local7，缺省情况下facility的标识为local0。

日志级别

日志级别是用来表示日志的严重程度。

级别	含义
info	一般提示信息
trace	调试信息
warning	告警信息
error	错误事件
fatal	致命错误，必须马上采取行动

日志的存储

生成的系统日志可以在本设备查看，也可以输出到日志服务器进行查看。

配置系统日志

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置日志级别。

   ConnetOS# **set system syslog log-level** { **error** | **fatal** | **info** | **trace** | **warning** }

   缺省情况下，日志级别是warning。

3. 配置日志的facility。

   ConnetOS# **set system syslog log-facility** *facility-number*

   缺省情况下，日志的facility级别是0，即local0。

4. （可选）指定远端日志服务器，用于接收ConnetOS发送过来的日志。

   ConnetOS# **set system syslog host server-ip** *ip-address*

5. 提交配置

   ConnetOS# **commit**

6. 打开日志监控功能

   ConnetOS> **syslog monitor** { **off** | **on** }

查看本地系统日志

ConnetOS系统支持按行数和日期查看系统日志。

- 按行数查看日志:

```
ConnetOS> show log last-rows 4
Apr 12 2017 11:36:50 ConnetOS local2.debug : [1][cli_sh] Parsing configuration
Apr 12 2017 11:36:57 ConnetOS local2.debug : [1][cli_sh] Starting CLI
Apr 12 2017 11:36:57 ConnetOS local0.warning : admin logined the switch cli
Apr 12 2017 11:37:10 ConnetOS local0.warning : [1][cli_sh] Executing command by␣
↪admin:
"show log last-rows 4"
```

- 按日期查看日志:

```
ConnetOS> show log date 2017.4.18
Apr 18 2017 15:41:38 ConnetOS local0.info : START: telnet pid=29107 from=192.168.
↪1.141
Apr 18 2017 15:41:58 ConnetOS local0.debug : [1][rtrmgr] registering interest in
cli-29134-ConnetOS
Apr 18 2017 15:41:58 ConnetOS local2.debug : [1][cli_sh] Waiting for␣
↪configuration from
rtrmgr
Apr 18 2017 15:41:58 ConnetOS local0.debug : [1][rtrmgr] XRL Birth: class
cli-29134-ConnetOS instance cli-29134-ConnetOS-
↪dfe1a22adc1cbd2ebbdae103226aee58@127.0.0.1
Apr 18 2017 15:41:59 ConnetOS local2.debug : [1][cli_sh] Parsing configuration
Apr 18 2017 15:42:05 ConnetOS local2.debug : [1][cli_sh] Starting CLI
Apr 18 2017 15:42:05 ConnetOS local0.warning : admin logined the switch cli
Apr 18 2017 15:42:22 ConnetOS local0.warning : [1][cli_sh] Executing command by␣
↪admin:
"show version "
Apr 18 2017 15:42:38 ConnetOS local0.warning : [1][cli_sh] Executing command by␣
↪admin:
"show log date 2017.4.18"
```

端口镜像配置

端口镜像概述

在网络维护的过程中会遇到需要对报文进行获取和分析的情况，比如怀疑有攻击报文，此时需要在不影响报文转发的情况下，对报文进行获取和分析。

端口镜像是指在不影响报文正常处理流程的情况下，将镜像端口的报文复制一份到观测端口。用户可以利用数据监控设备来分析复制到观测端口的报文，进行网络监控和故障排除。

- 镜像端口：又叫源端口，是被监控的端口，从镜像端口流经的报文将被复制到观测端口。
- 观测端口：又叫目的端口，是连接监控设备的端口，用于输出从镜像端口复制过来的报文。

配置端口镜像

端口镜像支持对镜像端口入方向、出方向或双向的流量进行镜像。如果要对双向的流量进行镜像，分别配置出、入方向的端口镜像即可。input表示配置镜像端口，output表示配置观测端口。

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置镜像端口。

   ConnetOS# **set analyzer instance instance-name* **input** { **egress** | **ingress** } *interface-name*

1. 配置观测端口，观测端口不可以是LAG成员。

   ConnetOS# **set analyzer instance** *instance-name* **output** *interface-name*

2. 提交配置

   ConnetOS# **commit**

检查配置结果

查看端口镜像的相关配置信息:

```
ConnetOS# show analyzer
     instance mirror1 {
        input {
            egress "te-1/1/10"
        }
        output: "te-1/1/20"
    }
```

查看镜像信息:

```
ConnetOS# show analyzer
Analyzer name: mirror1
Output interface: <te-1/1/20>
Ingress monitored interfaces:
Egress monitored interfaces: <te-1/1/10>
```

sFlow配置

sFlow简介

概述

sFlow（Sampled Flow）采样流，是一种基于报文采样的网络流量监控技术，主要用于对网络流量进行统计分析。

企业网络规模较小、组网灵活，容易出现由于组网或者遭受攻击导致的流量业务异常。所以企业用户需要实时监控接口的流量状况，以便及时采取措施来保证企业网络的正常稳定运行。

sFlow关注的是接口的流量情况、转发情况以及设备整体运行状况，适合于网络异常监控以及网络异常定位，为企业用户的日常巡检维护提供了极大的方便。特别适合于企业网用户。

原理介绍

sFlow系统组成

sFlow系统由sFlow Agent和sFlow Collector组成。sFlow Agent嵌入到交换机中，对接口上的流量进行采样，然后将采样结果发送给sFlow collector。sFlow collector对sFlow报文进行分析，并显示分析结果。

采样机制

sFlow Agent将接口上获取的将信息封装成sFlow报文，当sFlow报文缓冲区满或达到sFlow报文老化时间后，将sFlow报文发送到指定的sFlow Collector。

sFlow Agent提供了两种采样方式，供用户从不同的角度分析网络流量状况：

- Flow采样： 基于数据包的流采样。在指定接口上按照特定的采样方向和采样比对报文进行采样分析。
- Counter采样：基于时间的统计信息采样。在指定接口上周期性地获取流量统计信息、内存使用信息。

Flow采样和Counter采样是两种相互独立的采样，两者互相没有影响。但是由于采样的方式不一样，获取的信息维度也不一样。Flow方式关注流量的详细信息，更聚焦于具体的流的分析，可以监控和分析网络上的流行为。而Counter采样只关注接口上流量的数量，不关注流量的详细信息。

优势

和其他网络流量监控技术相比，sFlow 具有如下优点:

- 能够在线实时的监控每个接口。
- 可以针对不同接口设置不同的采样方式及采样参数，采样灵活。
- 采样时不需要镜像流量，对网络性能影响小。
- sFlow Agent内嵌在设备中，不需要额外的监测设备和接口，节省成本。

配置sFlow

在配置sFlow时：

1. 首先需要指定进行sFlow采样的接口，即使能接口下的sFlow使能，并设置采样比或采样时间等采样参数。
2. 如果不配置接口下的采样参数，就按照全局的采样参数进行采样。如果配置了接口下的采样参数，就按照接口下的采样参数进行采样。
3. 必须指定Collector，否则无法进行采样分析。

配置Flow采样

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置全局sFlow的采样比。

   ConnetOS# **set protocols sflow sampling-rate** { **ingress** | **egress** } *sampling-rate*

   缺省情况下，全局sFlow的采样比为2000。

3. 配置全局sFlow的采样报文最大长度。

   ConnetOS# **set protocols sflow sampling-rate** { **ingress** | **egress** } *header-length*

4. 使能指定接口的sFlow的功能

   ConnetOS# **set protocols sflow interface** *interface-name* **enable** { **false** | **ture** }

   sFlow采样时必须指定接口进行采样。

5. （可选）配置指定接口下的采样比。

   ConnetOS# **set protocols sflow interface** *interface-name* **sampling-rate** { **ingress** | **egress** } *sampling-rate*

   如果接口下不配置采样比，就按照全局下的配置生效。

6. （可选）配置指定接口下的采样报文最大长度。

   ConnetOS# **set protocols sflow interface** *interface-name* **header-length** *header-length*

   如果接口下不配置采样报文最大长度，就按照全局下的配置生效。

7. 指定Collector的IP地址。

   ConnetOS# **set protocols sflow collector address** *ip-address*

8. 指定Collector的端口号。

   ConnetOS# **set protocols sflow collector port** *port-number*

9. 提交配置

   ConnetOS# **commit**

配置Counter采样

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置全局Counter采样的时间间隔报文。

   ConnetOS# **set protocols sflow polling-interval** *polling-interval*

   如果接口下不配置采样时间间隔，就按照全局下的配置生效。

3. 使能指定接口的sFlow的功能

   ConnetOS# **set protocols sflow interface** *interface-name* **enable** { **false** | **ture** }

   sFlow采样时必须指定接口进行采样。

4. （可选）配置接口下的Counter采样的时间间隔报文。

   ConnetOS# **set protocols sflow interface** *interface-name* **polling-interval** *polling-interval*

   如果接口下不配置采样时间间隔，就按照全局下的配置生效。

5. 指定Collector的IP地址。

   ConnetOS# **set protocols sflow collector address** *ip-address*

6. 指定Collector的端口号。

   ConnetOS# **set protocols sflow collector port** *port-number*

7. 提交配置

   ConnetOS# **commit**

检查配置结果

执行 **show sflow collector** 命令，查看collector的配置信息:

```
ConnetOS> show sflow collector
Collector IP Address  UDP Port   Flow Sample Counters   Counter Sample Counters
↪Datagram Counters
--------------------    --------     --------------------        --------------------
↪--       ------------------
2.2.2.2              6400      0                    4                    4
```

执行 **show sflow** 命令，查看全局的sFlow配置信息:

```
ConnetOS> show sflow
Status   Agent IP Address   Ingress Sample Rate   Egress Sample Rate   Polling
↪Interval   Header Length
--------   ----------------     -------------------       ------------------       ---
↪-------------     -------------
Enabled  192.168.1.35     1:2000              1:2000              60s             64B
```

执行 **show sflow interface** 命令，查看接口上的sFlow配置信息:

```
ConnetOS> show sflow interface
Interface  Status    Ingress Sample Rate  Egress Sample Rate  Polling interval
↪Header length
---------   ---------   -------------------      ------------------------   -----------
↪-----     -------------
te-1/1/1   Enabled  1:2000              1:2000              60s              100B
te-1/1/2   Enabled  1:600               1:2000              0s               64B
↪64B
```

sDrop配置

概述

sDrop简介

现代网络中出现的大部分的网络问题都跟丢包有扯不开的关系，而如何对丢包进行诊断一直是繁杂且专业的问题，针对这一现象ConnetOS对现今的数据中的网络环境中的丢包进行了详细的分析，并设计了sDrop（Streaming Dropped packet）解决用户所关注的问题。

不同于sFlow主要分析接口转发的数据流量，sDrop主要用于分析丢弃的数据流量。sDrop可以解决如下情况下的丢包:

- 发现瞬间突发流量丢包。
- 协助解决业务配置错误丢包。
- 发现网络中存在攻击流量丢包。
- 发现网络中存在的错误的报文。

并且提供用户全网的丢包信息的收集及分析能力。

sDrop对数据中心网络中存在的20多种常见的丢包情况进行了分析后，将设备丢弃的报文分为两种:

- 可以获取丢弃的原始报文，将报文送往CPU进行分析，得到丢包原因。
- 无法获取丢弃的原始报文，通过获取统计信息，得到丢包的原因。

sDrop可以进行计数的比较和查看，经过一定的轮询时间，可以查询丢包内容和统计。

sDrop功能嵌入到交换机中，对设备上丢弃的报文进行采样，并将采样结果发送给collector。collector对sFlow报文进行分析、显示分析结果。

Note: collector可以是安装了分析软件的用户终端，或专门的分析设备。

丢包查看方式

对于丢包的感知，ConnetOS提供了设备上和设备外两种展示方式。

- 设备支持将报文的丢包信息导出，通过wireshark的lua插件进行展示。
- 对于设备上的报文，通过 **show sdrop** 命令查看报文的实际内容，定位基本的丢包原因。

配置sDrop

1. 进入配置模式。

   ConnetOS> **configure**

2. 配置sDrop采样的轮询时间。

   ConnetOS# **set protocols sdrop polling-interval** *polling-interval*

   缺省情况下，轮询时间是60s。

3. 指定Collector的IP地址。

   ConnetOS# **set protocols sdrop collector address** *ip-address*

4. 指定Collector的UDP端口号。

   ConnetOS# **set protocols sdrop collector port** *port-number*

   缺省情况下，UDP端口号的端口号是32768。

5. 提交配置

   ConnetOS# **commit**

检查配置结果

执行 **show sdrop** 命令，查看丢弃报文的信息:

```
ConnetOS 1> show sdrop
Info of Dropped Packets in last 1 min.
Input Physical Port     Output Physical Port    Drop Reason
↪  Last Detectted Time
--------------------    --------------------    ----------------------------------
↪  ---------------------
NA                      te-1/1/2                Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
NA                      te-1/1/5                Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
NA                      te-1/1/6                Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
NA                      te-1/1/48               Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
```

Navmesh

Navmesh概述

在网络出现故障时，ConnetOS提供了命令行诊断方式：Navmesh。Navmesh命令用于查看指定流量的路径。

通过报文流量的五元组，Navmesh可以查询到流量唯一的出端口和入端口信息。并且，即使此流量已经停止发送，只要表项还在，仍然可以查询到出入端口信息。

但是如果流量是通过ECMP的方式发送到设备，那么通过Navmesh查询到的可能是入端口和出端口列表。此时，Navmesh将发起二次查询，二次查询会精确的找到唯一的一对出端口和入端口。

Note: 二次查询可以查到的前提是流量一直在发送。

查询流量的出入端口

Navmesh通过五元组来查询流量的出端口和入端口:

```
ConnetOS> show navmesh source-ipv4 10.10.10.10 dest-ipv4 40.40.40.10 l4-source-port 55
l4-dest-port 11 protocol 17
The Flow Information :
  Source ipv4 address          : 10.10.10.10
  Destination ipv4 address     : 40.40.40.10
  L4 source port               : 55
  L4 destination port          : 11
  IP protocol                  : 17
The forwarding path :
   Ingress Port   Egress Port
   ------------   -------------
   te-1/1/5       te-2/1/17
```

TcpDump

简介

TcpDump，即Dump the traffic on a network，是根据使用者的定义对网络上的数据包进行截获的包分析工具，也是Linux中强大的网络数据采集分析工具之一。

ConnetOS基于Debian Linux，天然支持TcpDump。

TcpDump可以将网络中传送的数据包的“头”完全截获下来提供分析。它支持针对网络层、协议、主机、网络或端口的过滤，并提供and、or、not等逻辑语句来帮助你去掉无用的信息。

TcpDump的总的输出格式为：系统时间 来源主机.端口 > 目标主机.端口 数据包参数 如下图所示:

```
admin@ConnetOS:~ 1$ tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
16:00:54.674781 IP 192.168.1.31.telnet > 192.168.1.127.57388: Flags [P.], seq␣
↪1750927828:1750927979, ack 93009793, win 57, options [nop,nop,TS val 248778 ecr␣
↪1026263780], length 151
16:00:54.675973 IP 192.168.1.127.57388 > 192.168.1.31.telnet: Flags [.], ack 151, win␣
↪4091, options [nop,nop,TS val 1026264589 ecr 248778], length 0
......
```

TcpDump使用

直接启动 **tcpdump**，将监视第一个网络接口上所有流过的数据包。虽然简单，但是数据信息太多，不利于数据包的截取和后续分析。可以使用各种参数进行数据包过滤。

TcpDump使用参数指定要监视数据包的类型、地址、端口等，根据具体的网络问题，充分利用这些过滤规则就能达到迅速定位故障的目的。

如果忘记了这个软件的用法，可以使用 **tcpdump –help** 查看使用方法:

```
admin@ConnetOS:~ 1$ tcpdump --help
tcpdump version 4.6.2
libpcap version 1.6.2
OpenSSL 1.0.1t  3 May 2016
Usage: tcpdump [-aAbdDefhHIJKlLnNOpqRStuUvxX#] [ -B size ] [ -c count ]
                [ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
                [ -i interface ] [ -j tstamptype ] [ -M secret ] [ --number ]
                [ -Q in|out|inout ]
                [ -r file ] [ -s snaplen ] [ --time-stamp-precision precision ]
                [ -T type ] [ --version ] [ -V file ]
                [ -w file ] [ -W filecount ] [ -y datalinktype ] [ -z command ]
                [ -Z user ] [ expression ]
```

Note: 使用 **man tcpdump**，可以查看这些过滤参数的具体用法。

由于TcpDump对截获的数据没有进行彻底解码，为了网络故障分析方便，通常的解决办法是先使用带“-w”参数的tcpdump截获数据并保存到文件中，然后再使用其他程序进行解码分析。

TcpDump常见命令介绍

启动

tcpdump

直接启动tcpdump将监视第一个网络接口上所有流过的数据包。

监视指定网络接口的数据包

tcpdump -i eth1

如果不指定网卡，默认tcpdump只会监视第一个网络接口，一般是eth0，下面的例子都没有指定网络接口。

监听指定协议的数据

监听ICMP协议的数据:

```
tcpdump -i eth0 -nn 'icmp'
```

如果要监听TCP或UDP协议，只需要修改上例的icmp就可以了。

监听指定的主机

抓取192.168.1.231接收到的包和发送的包:

```
tcpdump -i eth0 -nn 'host 192.168.1.231'
```

抓取192.168.1.231发送的包:

```
tcpdump -i eth0 -nn 'src host 192.168.1.231'
```

抓取192.168.1.231接收到的包:

```
tcpdump -i eth0 -nn 'dst host 192.168.1.231'
```

监听指定端口

监听主机的80端口收到和发送的所有数据包:

```
tcpdump -i eth0 -nnA 'port 80'
```

监听指定主机和端口

监听192.168.1.231主机通过80端口发送的数据包。多个条件可以用and，or连接:

```
tcpdump -i eth0 -nnA 'port 80 and src host 192.168.1.231'
```

监听除某个端口外的其它端口

监听非22端口的数据包:

```
tcpdump -i eth0 -nnA '!port 22'
```

VXLAN配置

VXLAN简介

概述

在云计算中，大量的采用和部署虚拟化是一个基本的技术模式。而服务器虚拟化技术的广泛部署，增加了数据中心的计算密度。同时为了实现业务的灵活变更，虚拟机需要在网络中不受限制的迁移。

虚拟机数量的快速增长与虚拟机迁移业务的日趋频繁，给传统的“二层+三层”数据中心网络带来了新的挑战:

- 虚拟机规模受网络设备MAC地址表项规格的限制
- 传统网络的VLAN隔离能力有限，
- 在传统的二层域中，虚拟机迁移范围受限。

VXLAN（Virtual eXtensible Local Area Network）虚拟扩展本地网络，是NVO3（Network Virtulization over Layer3）中的一种网络虚拟化技术。VXLAN采用MAC in UDP的报文封装方式，将二层报文用三层协议进行封装，可实现二层网络在三层范围内进行扩展，同时满足数据中心大二层虚拟迁移和多租户的需求。

VXLAN具有如下优点：

- 降低对MAC地址规格的需求。

  虚拟机发出的数据包封装在UDP，使用物理网络的IP地址、MAC地址作为外层头进行封装，对网络只表现为封装后的参数，降低了对MAC地址规格的需求。

- 满足用户的大量隔离需求。

  VXLAN采用VNI（VXLAN Network Identifier）进行用户标识。VNI由24比特组成，支持多达16M的VXLAN段，从而满足了大量的用户标识。

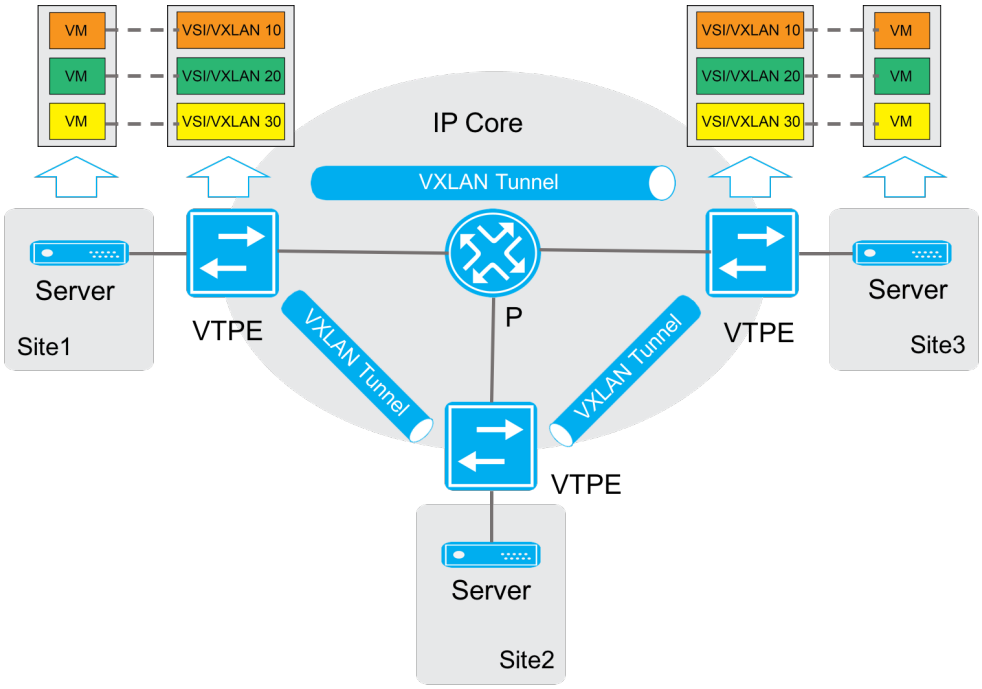
- 虚拟机迁移不受网络架构限制。

  VXLAN通过采用MAC in UDP封装来延伸二层网络，将以太报文封装在IP报文之上，通过路由在网络中传输，无需关注虚拟机的MAC地址。且路由网络无网络结构限制，通过路由网络，虚拟机迁移不受网络架构限制。

VXLAN网络模型

角色组成

VXLAN网络模型示意图



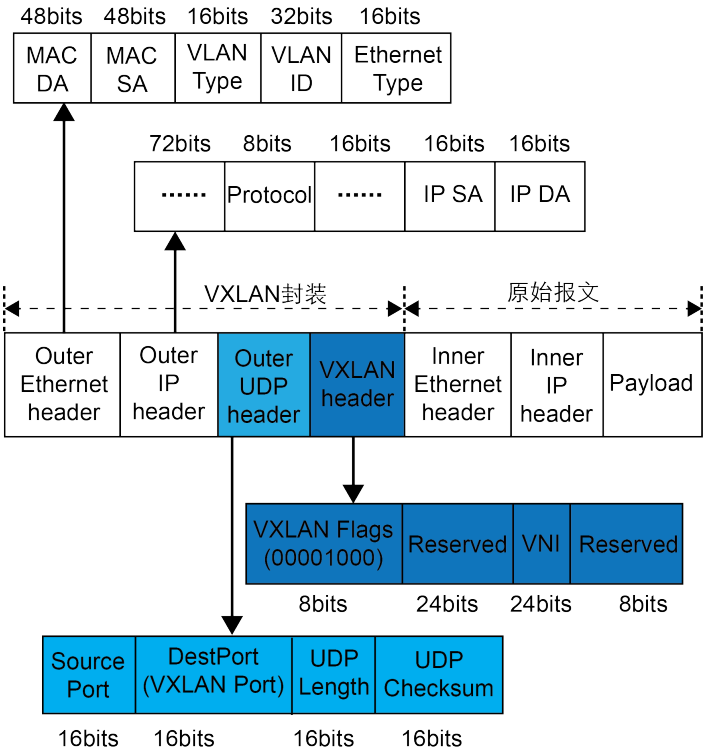
VXLAN网络中有以下角色组成：

- VM（Virtual Machine）虚拟机 在一台服务器上可以创建多台虚拟机，不同的虚拟机可以属于不同的VXLAN。属于相同 VXLAN的虚拟机处于同一个逻辑二层网络，彼此之间二层互通。属于不同VXLAN的虚拟机之间二层隔离。

- VTEP（VXLAN Tunnel Endpoints）VXLAN隧道端点 VXLAN网络的边缘设备，是VXLAN隧道的起点和终点，VXLAN报文的相关处理均在这上面进行。VTEP既可以是一台独立的网络设备，也可以是虚拟机所在的服务器。
- VNI（VXLAN Network Identifier）VXLAN 网络标识符 VNI是类似于VLAN ID。一个VNI代表了一个VXLAN段，属于不同VXLAN的虚拟机之间不能直接进行二层通信。
- VXLAN隧道 VXLAN隧道是建立在两个VTEP之间的一条虚拟通道，用来传输经过VXLAN封装的报文。VXLAN报文中源IP地址为本节点的VTEP地址，VXLAN报文中目的IP地址为对端节点的VTEP地址，一对VTEP地址就对应着一个VXLAN隧道。
- VSI（Virtual Switching Instance）虚拟交换实例 VTEP上为一个VXLAN提供二层交换服务的虚拟交换实例。VSI与VXLAN一一对应，可以看作是VTEP上的一台基于VXLAN 进行二层转发的虚拟交换机, 它具有传统以太网交换机的所有功能，包括源 MAC 地址学习、MAC 地址老化、泛洪等。

报文格式

VXLAN报文格式



外层Ethernet头封装。

- SA：发送报文的虚拟机所属的VTEP的MAC地址。
- DA：目的虚拟机所属VTEP上路由表中智联的下一跳MAC地址。
- VLAN Type：可选字段，当报文中携带VLAN Tag时，该字段取值为0x8100。
- Ethernet Type：以太报文类型，IP协议报文该字段取值为0x0800。

外层IP头封装

- 源IP地址为发送报文的虚拟机所属VTEP的IP地址；目的IP地址是目的虚拟机所属的VTEP的IP地址。

外层UDP封装

- 目的UDP端口号是4789。源端口号是内层以太报文头通过哈希算法计算后的值。

VXLAN头封装

- Flags：8比特，取值为00001000
- VNI：VXLAN网络标识，24比特，用于区分VXLAN段
- Reserved：24比特和8比特，必须设置为0。

Connetos支持的VXLAN功能

VXLAN可以为分散的物理站点提供二层互联实现VXLAN Bridging，即实现相同VXLAN中的相同网段之间的通信。当部署VXLAN Gateway时，可以为VXLAN站点内或站点间的虚拟机提供三层通信。

当前Connetos只支持VXLAN Bridging。

VXLAN运行机制

VXLAN隧道支持如下两种工作模式：

- 二层转发模式：VTEP通过查找MAC地址表项对流量进行转发。
- 三层转发模式：VTEP设备通过查找ARP表项对流量进行转发。

VXLAN Bridging工作在二层转发模式下，可以使相同网段的虚拟机通过VXLAN通信。当前Connetos只支持二层转发模式，VXLAN Bridging运行机制可以概括为：

1. 识别接收到的报文所属的VXLAN，以便将报文的源MAC地址学习到VXLAN对应的VSI（虚拟交换实例），并在该VSI内转发该报文。
2. 学习虚拟机的MAC地址。
3. 根据报文的目的MAC地址表项转发报文。

识别报文所属的VXLAN

1. 本地站点内接收到数据帧的识别

   VTEP将连接本地站点的端口绑定匹配规则后与VSI关联。VTEP从端口接收到数据帧后，根据匹配规则查找与其关联的VSI，VSI关联的VXLAN即为该数据帧所属的VXLAN。

   在VXLAN中，与VSI关联的端口统称为AC（Attachment Circuit，接入电路）。其中，AC在二层以太网接口上创建，它定义了一系列匹配规则，用来匹配从该二层以太网接口上接收到的数据帧。

2. VXLAN隧道上接收报文的识别

   对于从VXLAN隧道上接收到的VXLAN报文，VTEP根据报文中携带的VXLAN ID判断该报文所属的VXLAN。

学习MAC地址

MAC地址学习分为：

- 本地MAC地址学习

  本地MAC地址学习是指VTEP对本地站点内虚拟机MAC地址的学习。VTEP接收到本地虚拟机发送的数据帧后，判断该数据帧所属的VSI（VXLAN），并将数据帧中的源MAC地址（本地虚拟机的MAC地址）添加到该VSI的MAC地址表中，该MAC地址对应的接口为接收到数据帧的接口。

VXLAN不支持静态配置本地MAC地址。

- 远端MAC地址学习

  远端MAC地址学习是指VTEP对远端站点内虚拟机MAC地址的学习。远端MAC地址的学习方式有如下两种:

  - 静态配置：手工指定远端MAC地址所属的VSI（VXLAN），及其对应的VXLAN隧道接口。
  - 通过内层报文中的源MAC地址动态学习： VTEP从VXLAN隧道上接收到远端VTEP发送的VXLAN报文后，根据VXLAN ID判断报文所属的VXLAN，对报文进行解封装，还原二层数据帧，并将数据帧中的源MAC地址（远端虚拟机的MAC地址）添加到所属VXLAN对应VSI的MAC地址表中，该MAC地址对应的接口为VXLAN隧道接口。

  静态配置的远端MAC地址表项优先级高于源MAC地址动态学习的表项，后生成的表项可以覆盖已经存在的表项。

接入模式

接入模式分为以下两种:

- VLAN接入模式

  从本地站点接收到的、发送给本地站点的以太网帧可以携带VLAN tag，也可以不携带VLAN tag。VTEP从本地站点接收到以太网帧后，根据报文的tag映射到相应的VNI，删除该帧的最外一层VLAN tag，再转发该数据帧。

  VTEP发送以太网帧到本地站点时，根据VNI映射出VLAN，如果该VLAN为其tagged VLAN则添加VLAN tag，否则不需要添加。采用该模式时，VTEP不会传递VLAN tag信息，不同站点可以独立地规划自己的VLAN，不同站点的不同VLAN之间可以互通。

- Ethernet接入模式

  从本地站点接收到的、发送给本地站点的以太网帧可以携带VLAN tag，也可以不携带VLAN tag。VTEP从本地站点接收到以太网帧后，所有报文都映射到对应的VNI，删除该帧的最外一层VLAN tag，再转发该数据帧；VTEP发送以太网帧到本地站点时，不会为其添加VLAN tag。

转发已知单播流量

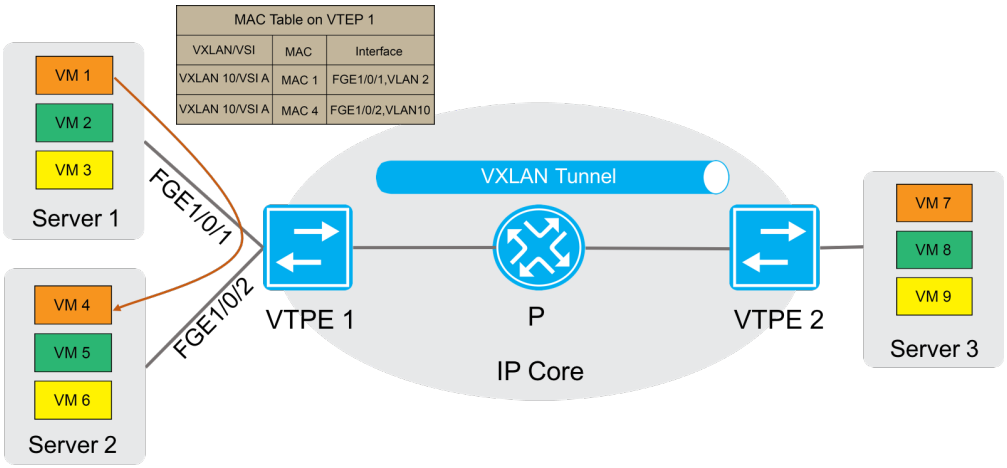
完成本地和远端MAC地址学习后，VTEP在VXLAN内转发已知单播流量分为:

- 站点内转发
- 站点间转发

站点内流量

对于站点内流量，VTEP判断出报文所属的VSI后，根据目的MAC地址查找该VSI的MAC地址表，从相应的本地接口转发给目的VM。

站点内单播流量转发



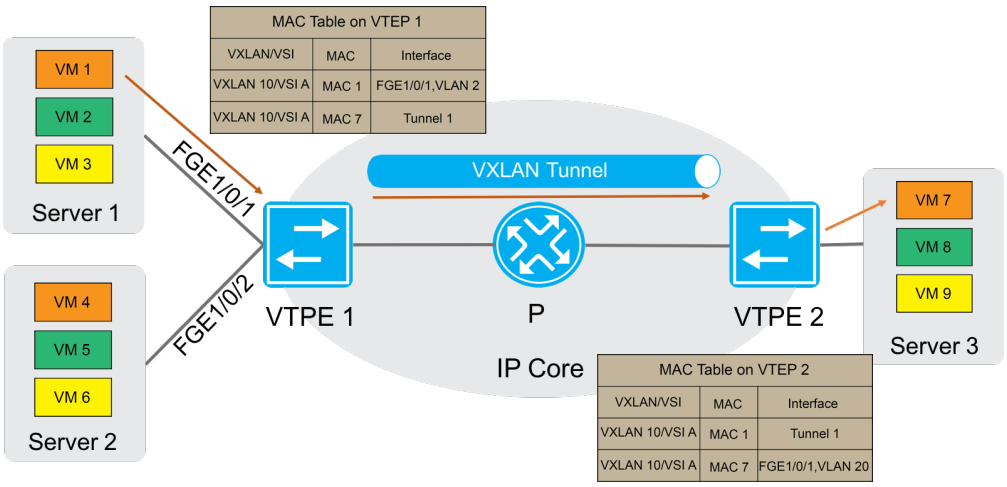
如上图所示，以VM 1（MAC地址为MAC 1）发送以太网帧到VM 4（MAC地址为MAC 4）为例，站点内单播流量的转发过程为：

1. VM 1发送以太网数据帧给VM 4，数据帧的源MAC地址为MAC 1，目的MAC为MAC 4，VLAN tag为2。
2. VTEP 1从接口GigabitEthernet1/0/1收到该数据帧后，判断该数据帧属于VSI A（VXLAN 10），查找VSI A的MAC地址表，得到MAC 7的出端口为FortyGigE1/0/2，坐在VLAN为VLAN10。
3. VTEP 2从接口GigabitEthernet1/0/2的VLAN 10内将数据帧发送给VM 4。

站点间流量

对于站点间流量，VTEP判断出报文所属的VSI后，根据目的MAC地址查找该VSI的MAC地址表，从相应的隧道将封装后的VXLAN报文给对端VTEP。

站点间单播流量转发



如上图所示，以VM 1（MAC地址为MAC 1）发送以太网帧给VM 7（MAC地址为MAC 7）为例，站点间单播流量的转发过程为：

1. VM 1发送以太网数据帧给VM 7，数据帧的源MAC地址为MAC 1，目的MAC为MAC 7，VLAN tag为2。

2. VTEP 1从接口GigabitEthernet1/0/1收到该数据帧后，判断该数据帧属于VSI A（VXLAN 10），查找VSI A的MAC地址表，得到MAC 7的出端口为Tunnel 1。
3. VTEP 1为数据帧封装VXLAN头、UDP头和IP头后，将封装好的报文通过VXLAN隧道Tunnel 1、经由IP设备发送给VTEP 2。
4. VTEP 2接收到报文后，根据报文中的VXLAN ID判断该报文属于VXLAN 10，并剥离VXLAN头、UDP头和IP头，还原出原始的数据帧。
5. VTEP 2查找与VXLAN 10对应的VSI A的MAC地址表，得到MAC 7的出端口为GigabitEthernet1/0/1，所在VLAN为VLAN 20。
6. VTEP 2从接口GigabitEthernet1/0/1的VLAN 20内将数据帧发送给VM 7。

BUM流量

BUM流量包括组播、广播和未知单播流量。

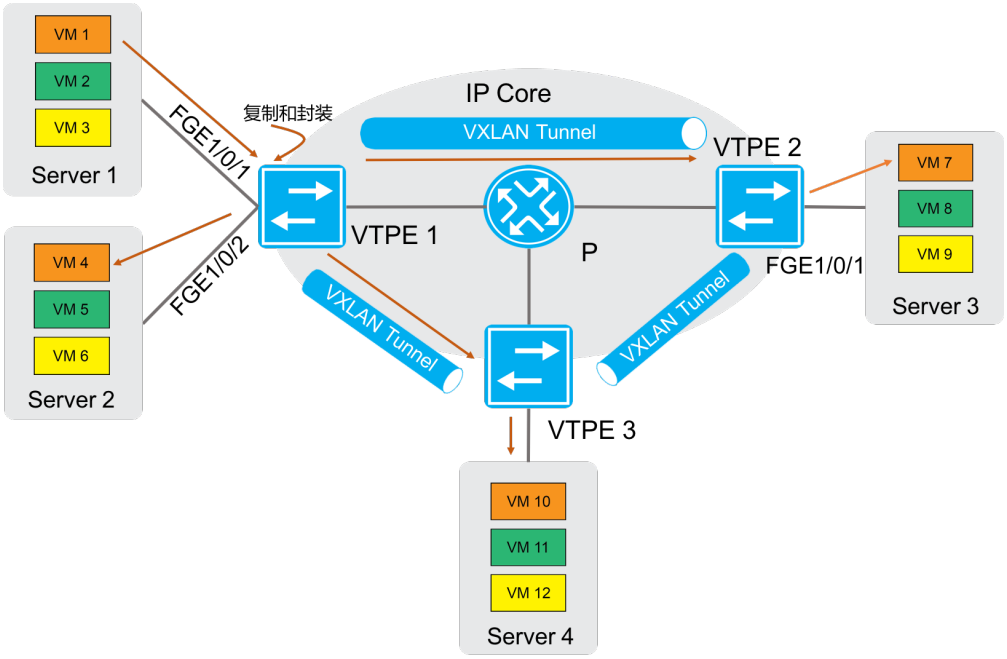
根据复制方式的不同，流量泛洪方式分为:

- 单播路由方式（头端复制）
- 组播路由方式（核心复制）

转发BUM流量

单播路由的方式又称为头端复制。接口收到BUM报文后，本地VTEP通过控制平面获取属于同一个VNI的VTEP列表，将收到的BUM报文通过本地接口发送给本地站点，根据VTEP列表进行复制并发送给属于同一个VNI的所有VTEP。通过头端复制完成BUM报文的泛洪，不需要依赖组播路由协议。

头端复制方式转发示意图



如上图所示，头端复制方式的BUM流量转发过程为:

1. VTEP 1接收到本地虚拟机发送的组播、广播和未知单播数据帧后，判断数据帧所属的VXLAN，通过该VXLAN内除接收接口外的所有本地接口和VXLAN隧道转发该数据帧。通过VXLAN隧道转发

数据帧时，需要为其封装VXLAN头、UDP头和IP头，将BUM数据帧封装在多个单播报文中，发送到VXLAN内的所有远端VTEP。

2. 远端VTEP（VTEP 2和VTEP 3）接收到VXLAN报文后，解封装报文，将原始的数据帧在本地站点的指定VXLAN内泛洪。为了避免环路，远端VTEP从VXLAN隧道上接收到报文后，不会再将其泛洪到其他的VXLAN隧道。

BUM流量抑制

缺省情况下，VTEP从本地站点内接收到BUM数据帧后，会在该VXLAN内除接收接口外的所有本地接口和VXLAN隧道上泛洪该数据帧，将该数据帧发送给VXLAN内的所有站点。如果用户希望把该类数据帧限制在本地站点内，不通过VXLAN隧道将其转发到远端站点，则可以通过配置命令手工禁止VXLAN对应VSI的泛洪功能。

负载均衡

Connetos交换机作为VTEP封装VXLAN时，如果tunnel的路由有多个下一跳或出接口是LAG口，封装后的VXLAN报文负载均衡发送出去。 Connetos交换机作为中间设备，三层转发VXLAN报文时，可以配置指定外层或内层原始报文做Hash后负载均衡。

配置VXLAN

配置VSI

1. 进入配置模式。

   ConnetOS> **configure**

2. 创建VSI。

   ConnetOS# **set vsi vsi-id** *vsi-id*

3. （可选）配置VSI描述。

   ConnetOS# **set vsi vsi-id** *vsi-id* **description** *description*

4. 将VSI和VNI关联。

   ConnetOS# **set vsi vsi-id** *vsi-id* **vni** *vni-id*

5. 关联VXLAN隧道与VSI。

   ConnetOS# **set vsi vsi-id** *vsi-id* **tunnel-ethernet** *tunnel-name*

6. 配置BUM流量抑制

   ConnetOS# **set vsi vsi-id** *vsi-id* **flooding enable** { **false** | **true** }

7. 提交配置。

   ConnetOS# **commit**

配置VXLAN隧道

1. 创建隧道。

   ConnetOS# **set interface tunnel-ethernet** *tunnel-name*

2. 配置隧道模式为VXLAN。

   ConnetOS# **set interface tunnel-ethernet** *tunnel-name* **mode vxlan**

3. 配置VXLAN隧道源端IP地址。

   ConnetOS# **set interface tunnel-ethernet** *tunnel-name* **source address** *ip-address*

4. 配置VXLAN隧道目的端IP地址。

   ConnetOS# **set interface tunnel-ethernet** *tunnel-name* **destination address** *ip-address*

5. （可选）配置VXLAN隧道描述。

   set interface tunnel-ethernet *tunnel-name* **description** *description*

6. 配置静态远端MAC地址。

   ConnetOS# **set interface tunnel-ethernet** *tunnel-name* **static-mac-address** *mac-address* [**vsi** *vsi-id*]

7. 提交配置。

   ConnetOS# **commit**

配置VXLAN业务接入点

1. 配置VXLAN业务接入点。

   ConnetOS# **set interface** { **gigabit-ethernet** | **aggregate-ethernet** } *interface-name* **family ethernet-switching vsi** *vsi-id* { **ethernet-mode enable true** | **vlan-mode dot1q** *vlan-id* }

2. 提交配置。

   ConnetOS# **commit**

命令行格式约定

符号	说明
粗体	命令行关键字，必须原样输入。
斜体	命令行参数，需要按照要求输入实际值。
[ ]	命令行配置过程中，括号中的部分是可选的
[a | b | .....]	从括号中的值选取一个或不选
{ a | b | ..... }	从括号中的值选取一个
[a | b | .....] *	从括号中的值选取多个或不选
{ a | b | ..... } *	从括号中的值选取一个或多个

CHAPTER 4

命令参考

本章介绍了设备中各特性的配置命令，包括每条命令的功能、格式、参数、视图、缺省级别、使用指南和举例。

基础命令

基础命令

clear system commit

命令功能

clear system commit 命令用来清除尚未提交的配置，即候选配置。

命令格式

clear system commit

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

清除候选配置:

```
ConnetOS> clear system commit
```

clear system reboot

命令功能

clear system reboot 命令用来清除设备reboot的定时器。

命令格式

clear system reboot

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

清除设备reboot的定时器:

```
ConnetOS> clear system reboot
```

cls

命令功能

cls 命令用来清除当前终端屏幕上显示的信息。

命令格式

cls

参数说明

无

命令模式

运维模式、配置模式

使用指南

无。

配置举例

清除当前终端屏幕上的显示信息:

```
ConnetOS# cls
```

commit

命令功能

commit 命令用来提交当前用户更改的配置，使配置生效。
缺省情况下，用户配置并不会自动提交。

命令格式

commit [**at** *active-time* | **comment** *comment* | **confirmed** *auto-confirm-time*]

参数说明

active-time：提交的配置开始生效的时间。

comment：增加对本次提交内容的描述记录。字符串形式，不支持空格。

auto-confirm-time：设置如果配置没有确认，自动回滚的时间。整数形式，取值范围是1～30000，单位是秒。

命令模式

配置模式

使用指南

配置模式下，修改配置的命令，只有 **commit** 之后才能真正生效。即：配置由候选配置变为活动配置。

配置举例

提交当前配置:

```
ConnetOS# set system hostname switcha
ConnetOS# commit
```

configure

命令功能

configure 命令用来从运维模式进入配置模式提交当前用户配置，使配置生效。
缺省情况下，用户进入的命令行视图是运维模式。

命令格式

configure [**exclusive**]

参数说明

exclusive：锁定当前配置模式，不允许其他用户进入。

命令模式

运维模式

使用指南

如果设备当前有其他用户正在登录，不允许锁定当前配置模式。
如果想解除配置模式的锁定，执行 **quit** 命令或 **exit** 命令退出当前配置模式即可。

配置举例

进入配置模式:

```
ConnetOS> configure
Entering configuration mode.
Users root and admin are also in configuration mode.
ConnetOS#
```

discard

命令功能

discard 命令用来丢弃当前没有提交的配置。

命令格式

discard

参数说明

无

命令模式

配置模式

使用指南

无。

配置举例

丢弃当前未提交的配置:

```
ConnetOS# discard
```

exit

命令功能

exit 命令用来退出当前模式。

命令格式

exit [**configuration-mode** | **discard**]

参数说明

configuration-mode：退出配置模式，返回运维模式。
discard：丢弃当前尚未 **commit** 的配置，并返回运维模式。

命令模式

运维模式、配置模式

使用指南

运维模式下，只能执行 **exit** 命令。

在edit视图下，执行 **exit configuration-mode** 或者 **exit discard** 命令，将直接返回到运维模式。

配置举例

从配置模式，返回到运维模式:

```
ConnetOS# exit
Leave configuration mode.
ConnetOS>
```

help

命令功能

运维模式下，**help** 命令用来介绍如何利用ConnetOS的help功能完成命令行的输入。

配置模式下，**help** 命令用来介绍各类基本命令的功能。

命令格式

help （运维模式）

help { **commit** | **delete** | **edit** | **exit** | **help** | **load** | **quit** | **rollback** | **run** | **save** | **set** | **show** | **status** | **top** | **up** }（配置模式）

参数说明

commit：提交配置。

delete：删除配置。

edit：进入各级edit视图。

exit：从当前配置模式中退出。

help：命令行帮助信息。

load：从配置文件中加载配置信息。

quit：退出当前模式。

rollback：回退到上一次提交的配置。

run：执行运维模式下的命令。

save：把配置信息保存到文件中。

set：设置参数值或者创建新的配置项。

show：配置信息。

status：用户当前的配置信息。

top：退回到最上级配置视图。

up：退回到上一级配置视图。

命令模式

配置模式

使用指南

无。

配置举例

了解run命令的基本功能:

```
ConnetOS# help run
The "run" command allows operational-mode commands to be executed withoutleaving␣
↪configuration-mode.  This is particularly important if there areuncommitted changes␣
↪to the configuration.

For example, the operational-mode command "show bgp peers" can be run from␣
↪configuration-mode as "run show bgp peers".

Navigation commands such as the operational-mode "configure" command are not␣
↪available using the run command.
```

load

命令功能

load 命令用来加载指定配置文件中的配置。

命令格式

load *local-file-name*

参数说明

local-file-name：本地文件名。

命令模式

配置模式

使用指南

无。

配置举例

加载配置文件:

```
ConnetOS# load /switch/config/connetos.conf.03
Loading config: /switch/config/connetos.conf.03
ConnetOS# Waiting for merging configuration.
Load done.
```

quit

命令功能

quit 命令用来退出配置模式，回到运维模式。

命令格式

quit

参数说明

无

命令模式

配置模式

使用指南

如果当前有没有提交的配置，无法通过 **quit** 命令退出。可以先提交配置、或通过 **exit discard** 命令丢弃当前命令退出、或执行 **discard** 命令丢弃当前配置再执行 **quit** 命令退出。

配置举例

退回到运维模式:

```
ConnetOS# quit
Leave configuration mode.
ConnetOS>
```

rollback

命令功能

rollback 命令用来基于历史版本进行版本回退。

命令格式

rollback *version-number*

参数说明

version-number：回退版本数。整数形式，取值范围是1～49。

命令模式

配置模式

使用指南

如果在系统运行过程中发现配置错误、业务运行不正常或者配置对网络产生了超出预期的结果，用户可以通过rollback命令将系统回退到指定的版本。

历史活动配置的保存是按照倒序进行的，即序号1保存的是当前配置。

配置举例

将版本回退到上一个版本:

```
ConnetOS# rollback 1
Rolling back to config: /switch/config/connetos.conf.01
ConnetOS# Waiting for merging configuration.
Load done.
ConnetOS#
```

save

命令功能

save local-file-name 命令用来将当前配置文件保存到本地。

save default-to-startup 命令用来将缺省配置保存为启动配置。

save running-to-startup 命令用来将活动配置保存为启动配置。

命令格式

save { *local-file-name* | **default-to-startup** | **running-to-startup** }

参数说明

loacal-file-name：本地文件名称。字符串形式，长度范围是1～63。支持区分大小写不支持空格。
default-to-startup：设置缺省配置为启动配置。
running-to-startup：设置当前的活动配置为启动配置。

命令模式

配置模式

使用指南

无

配置举例

设置当前的活动配置为启动配置:

```
ConnetOS# save running-to-startup
Save done.
```

set system hostname (stack模式)

命令功能

set iss member hostname 命令用来设置设备名称。
delete iss member hostname 命令用来删除配置的设备名称，恢复到缺省值。
缺省情况下，设备名称为ConnetOS。

命令格式

set iss member *member-id* **hostname** *hostname*
delete iss member *member-id* **hostname**

参数说明

member-id：堆叠设备的member ID。整数形式，取值范围是0～2。

- 0：表示设备处于单机模式。
- 1：表示设备处于堆叠模式，且member ID为1。
- 2：表示设备处于堆叠模式，且member ID为2。

hostname：本地文件名称。字符串形式，取值范围是1～63。支持区分大小写不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

设置设备名称为switcha:

```
ConnetOS 1# set iss member hostname switcha
```

set system hostname (standalone模式)

命令功能

set system hostname 命令用来设置设备名称。

delete system hostname 命令用来删除配置的设备名称，恢复到缺省值。

缺省情况下，设备名称为ConnetOS。

命令格式

set system hostname *hostname*

delete system hostname

参数说明

hostname：本地文件名称。字符串形式，取值范围是1～63。支持区分大小写不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

设置设备名称为switcha:

```
ConnetOS# set system hostname switcha
```

show cli

命令功能

show cli 命令用来查看当前CLI接口的设置。

命令格式

show cli { **history** | **information** }

参数说明

history：查看CLI接口上命令行操作历史纪录。
information：查看当前CLI接口的设置信息。

命令模式

运维模式

使用指南

无。

配置举例

查看当前CLI接口的设置信息:

```
ConnetOS> show cli information
Idle timeout       : 900 seconds
Screen size(w x h) : 118 x 29
Terminal type      : xterm-256color
```

show host

命令功能

show host 命令用来查看当前设备信息。

命令格式

show host { **date** | **os** }

参数说明

date：查看当前日期及时间。
os：查看当前设备的操作系统。

命令模式

运维模式

使用指南

无。

配置举例

查看当前设备的操作系统:

```
ConnetOS> show host os
Linux ConnetOS 3.16.7-ckt25+ #3 SMP Sun Jan 22 19:44:21 CST 2017 x86_64 GNU/Linux
```

show host

命令功能

show host 命令用来查看当前设备信息。

命令格式

show host { **date** | **os** }

参数说明

date：查看当前日期及时间。
os：查看当前设备的操作系统。

命令模式

运维模式

使用指南

无。

配置举例

查看当前设备的操作系统:

```
ConnetOS> show host os
Linux ConnetOS 3.16.7-ckt25+ #3 SMP Sun Jan 22 19:44:21 CST 2017 x86_64 GNU/Linux
```

show running-config

命令功能

show running-config 命令用来查看当前运行的配置信息。

命令格式

show running-config

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看当前运行的配置信息:

```
ConnetOS> running-config
```

show tech-support

命令功能

show tech-support 命令用来查看并保存设备故障时的日志。

命令格式

show tech-support

参数说明

无

命令模式

运维模式

使用指南

在发生无法处理的故障后，请收集故障日志，发送到tech@connetos.com。

配置举例

查看当前版本信息:

```
ConnetOS> show tech-support
Start to collect information ......
/switch/bin//shell/show_tech_support.sh: line 12: /switch/ConnetOS.Int-201704061939-
↪techSupport.log: Permission denied
/switch/bin//shell/show_tech_support.sh: line 13: /switch/ConnetOS.Int-201704061939-
↪techSupport.log: Permission denied
/switch/bin//shell/show_tech_support.sh: line 14: /switch/ConnetOS.Int-201704061939-
↪techSupport.log: Permission denied
/switch/bin//shell/show_tech_support.sh: line 19: /switch/ConnetOS.Int-201704061939-
↪techSupport.log: Permission denied

The information has been stored in /switch/ConnetOS.Int-201704061939-techSupport.log,␣
↪please forward to tech@connetos.com
```

show version

命令功能

show version 命令用来查看当前版本信息。

命令格式

show version

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看当前版本信息:

```
ConnetOS> show version
Copyright (C) 2015-2017 YUNQI TECH, Inc.
PN              : C1020
OS              : ConnetOS GENERAL
Version ID      : 2.1.2
Build Code      : r2179 (14P28)
Switch MAC      : 00:03:0f:64:da:5f
Management MAC : 00:03:0f:64:da:60
Release Time    : 2017-03-28 19:05:44
```

status

命令功能

status 命令用来查看当前进行配置操作的用户。

命令格式

status

参数说明

无

命令模式

配置模式

使用指南

无。

配置举例

查看当前进行配置操作的用户:

```
ConnetOS# status
User root is also in configuration mode.
```

top

命令功能

top 命令用来退回到顶级视图，即配置视图下。

命令格式

top

参数说明

无

命令模式

配置模式

使用指南

top 命令是直接退回到顶级视图，如果需要逐步退回到上一级视图，执行命令 **quit**、**exit**、**up** 即可。

配置举例

从ospf4视图退回到配置模式:

```
ConnetOS# edit protocols ospf4
[edit protocols ospf4]
ConnetOS# top
ConnetOS#
```

up

命令功能

up 命令用来逐步退出配置模式下的各类edit视图。

命令格式

up

参数说明

up 命令是逐步退出各级edit视图，如果想要直接退回到顶级视图，执行 **top** 命令。
up 命令只能逐步退出视图，不能退出模式。**quit** 命令、**exit** 命令既可以退出视图，又可以退出模式。

命令模式

配置模式

使用指南

无。

配置举例

从ospf4视图退回到配置模式:

```
ConnetOS# edit protocols ospf4
[edit protocols ospf4]
ConnetOS# up
[edit protocols]
ConnetOS# up
```

远程登录命令

ssh

命令功能

ssh 命令用来登录远程设备。

命令格式

ssh *ip-address*

参数说明

ip-address：设备的IP地址。

命令模式

运维模式

使用指南

无

配置举例

登录IP地址为1.1.1.1的远程设备:

```
ConnetOS> ssh 1.1.1.1
```

telnet

命令功能

telnet 命令用来登录远程设备。

命令格式

telnet *ip-address*

参数说明

ip-address：设备的IP地址。

命令模式

运维模式

使用指南

无

配置举例

登录IP地址为1.1.1.1的远程设备:

```
ConnetOS> telnet 1.1.1.1
```

用户登录管理命令

管理口命令参考

set interface management-ethernet eth0 address

命令功能

set interface management-ethernet eth0 address 命令用来配置管理口的IP地址。

delete interface management-ethernet eth0 address 命令用来删除配置的管理口IP地址，恢复为缺省值。

缺省情况下，管理口的IP地址为0.0.0.0/0。

命令格式

set interface management-ethernet eth0 [**member** *member-id*] **address** *ip-address*

delete interface management-ethernet eth0 [**member** *member-id*] **address**

参数说明

member-id：堆叠系统的成员设备编号，本参数只在stack模式下显示。整数形式，取值为1、2。

- 1：表示设备为master设备；
- 2：表示设备为slave设备。

ip-address：管理口IP地址。点分十进制格式，取值形式为：目的IP地址/掩码长度。

命令模式

配置模式

使用指南

无。

配置举例

设置设备的管理口IP地址为1.1.1.1/24:

```
ConnetOS# set interface management-ethernet eth0 member 1 address 1.1.1.1/24
```

set interface management-ethernet eth0 dhcp

命令功能

set interface management-ethernet eth0 dhcp enable 命令用来配置是否使能管理口的DHCP功能。

delete interface management-ethernet eth0 dhcp enable 命令用来删除配置的管理口DHCP功能，恢复为缺省值。

缺省情况下，管理口的DHCP功能已经使能。

命令格式

set interface management-ethernet eth0 [**member** *member-id*] **dhcp enable** { **false** | **true** }

delete interface management-ethernet eth0 [**member** *member-id*] **dhcp enable**

参数说明

member-id：堆叠系统的成员设备编号，本参数只在stack模式下显示。整数形式，取值为1、2。

- 1：表示设备为master设备；
- 2：表示设备为slave设备。

false：不使能管理口的DHCP功能。

true：使能管理口的DHCP功能。

命令模式

配置模式

使用指南

无。

配置举例

使能管理口的DHCP功能:

```
ConnetOS# set interface management-ethernet eth0 dhcp enable true
```

set interface management-ethernet eth0 gateway

命令功能

set interface management-ethernet eth0 gateway 命令用来配置管理口的网关地址。

delete interface management-ethernet eth0 gateway 命令用来删除配置的管理口网关地址，恢复为缺省值。缺省情况下，管理口的网关地址为0.0.0.0。

命令格式

set interface management-ethernet eth0 [**member** *member-id*] **gateway** *ip-address*

delete interface management-ethernet eth0 [**member** *member-id*] **gateway**

参数说明

member-id：堆叠系统的成员设备编号，本参数只在stack模式下显示。整数形式，取值为1、2。

- 1：表示设备为master设备；
- 2：表示设备为slave设备。

ip-address：管理口网关的IP地址。点分十进制格式。

命令模式

配置模式

使用指南

无。

配置举例

设置设备的管理口网关的IP地址为1.1.1.1:

```
ConnetOS# set interface management-ethernet eth0 member 1 gateway 1.1.1.1
```

show interface management-ethernet eth0

命令功能

show interface management-ethernet eth0 命令用来查看管理网口的信息。

命令格式

show interface management-ethernet eth0

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看管理网口的信息:

```
ConnetOS 1> show interface management-ethernet eth0
Member ID  Interface   Address             Gateway          MAC
---------  ----------  ------------------  ---------------  -----------------
1          eth0        192.168.1.34/24     0.0.0.0          cc:37:ab:f4:82:f2
2          eth0        192.168.1.35/24     0.0.0.0          00:03:0f:64:da:9f
```

登录命令

set system services ssh enable

命令功能

set system services ssh enable 命令用来配置是否使能SSH功能。

delete system services ssh enable 命令用来删除配置的SSH功能。

缺省情况下，SSH功能是使能的。

命令格式

set system services ssh enable { **false** | **true** }

delete system services ssh enable

参数说明

false：不使能SSH功能。

true：使能SSH功能。

命令模式

配置模式

使用指南

无。

配置举例

使能SSH功能:

```
ConnetOS# set system services ssh enable true
```

set system services ssh connection-limit

命令功能

set system services ssh connection-limit 命令用来配置允许通过SSH方式登录的用户数。
delete system services ssh connection-limit 命令用来删除配置的SSH登录用户数。
缺省情况下，允许15个用户通过SSH方式登录设备。

命令格式

set system services ssh connection-limit *limit-number*
delete system services ssh connection-limit

参数说明

limit-number：可以登录的用户数目。整数形式，取值范围是1～250。

命令模式

配置模式

使用指南

无。

配置举例

设置可以通过SSH方式登录设备的用户数是25:

```
ConnetOS# set system services ssh connection-limit 25
```

set system services telnet enable

命令功能

set system services telnet enable 命令用来配置是否使能Telnet功能。

delete system services telnet enable 命令用来删除配置的Telnet功能。

缺省情况下，Telnet没有使能。

命令格式

set system services telnet enable { **false** | **true** }

delete system services telnet enable

参数说明

false：不使能Telnet功能。

true：使能Telnet功能。

命令模式

配置模式

使用指南

无。

配置举例

使能Telnet功能:

```
ConnetOS# set system services telnet enable true
```

set system services telnet connection-limit

命令功能

set system services telnet connection-limit 命令用来配置允许通过Telnet方式登录的用户数。

delete system services telnet connection-limit 命令用来删除配置的Telnet登录用户数。
缺省情况下，允许15个用户通过Telnet方式登录设备。

命令格式

set system services telnet connection-limit *limit-number*
delete system services telnet connection-limit

参数说明

limit-number：可以通过Telnet登录的用户数目。整数形式，取值范围是1～250。

命令模式

配置模式

使用指南

无。

配置举例

设置可以通过Telnet方式登录设备的用户数是25:

```
ConnetOS# set system services telnet connection-limit 25
```

set system inband enable

命令功能

set system inband enable 命令用来配置是否使能带内访问功能。
delete system inband enable 命令用来删除配置的带内访问功能，恢复为缺省值。
缺省情况下，带内访问功能是开启的。

命令格式

set system inband enable { **false** | **true** }
delete system inband enable

参数说明

false：不使能带内访问功能.
true：使能带内访问功能。

命令模式

配置模式

使用指南

无。

配置举例

使能带内访问功能:

```
ConnetOS# set system inband enable true
```

set system login announcement

命令功能

set system login announcement 命令用来设置用户登录设备时的通知信息。
delete system login announcement 命令用来删除配置的通知信息。
缺省情况下，用户登录设备时的通知信息为ConnetOS。

命令格式

set system login announcement *announcement-message*
delete system login announcement

参数说明

announcement-message：通知信息。

命令模式

配置模式

使用指南

无。

配置举例

设置用户登录设备时的通知信息为Hello:

```
ConnetOS# set system login announcement Hello
```

set system login user authentication

命令功能

set system login user authentication 命令用来创建用户账户。
delete system login user authentication 命令用来删除配置的用户账户。

命令格式

set system login user *user-name* **authentication plain-text-password** *plain-password*
delete system login user *user-name* **authentication** [*plain-text-password*]

参数说明

user-name：用户名。
plain-password：用户密码

命令模式

配置模式

使用指南

无。

配置举例

创建用户名为usera，密码为test的用户账户:

```
ConnetOS# set system login user usera authentication plain-text-password test
```

set system login user class

命令功能

set system login user user-name class 命令用来配置账户类型。
delete system login user authentication 命令用来删除配置的账户类型，恢复为缺省值。

缺省情况下，用户账户创建后，账户类型为 **super-user**。

命令格式

set system login user *user-name* **class** { **read-only** | **super-user** }
delete system login user *user-name* **class**

参数说明

read-only：只能对设备进行查询操作。
super-user：可以对设备进行查询和配置操作。

命令模式

配置模式

使用指南

无。

配置举例

设置用户usera的账户类型为read-only:

```
ConnetOS# set system login user usera class read-only
```

set system login-acl network

命令功能

set system login-acl network 命令用来配置允许登录设备的网段。
delete system login-acl network 命令用来删除配置的允许登录网段。
缺省情况下，允许所有网段登录。

命令格式

set system login-acl network *acl-network*
delete system login-acl network

参数说明

acl-network：允许登录的网段。

命令模式

配置模式

使用指南

无。

配置举例

设置只允许1.1.1.0/24网段的用户登录:

```
ConnetOS# set system login-acl network 1.1.1.0/24
```

show system users

命令功能

show system users 命令用来查看当前登录的用户信息。

命令格式

show system users

参数说明

无

命令模式

运维模式

使用指南

无

配置举例

查看当前登录的用户信息:

```
ConnetOS 1> show system users
root     pts/0        2017-05-15 14:23 (192.168.1.102)
admin    pts/2        2017-05-15 14:45 (192.168.1.128)
```

AAA命令

set system aaa local enable

命令功能

set system aaa local enable 命令用来配置是否使能本地认证功能。
delete system aaa local enable 命令用来删除本地认证功能。
缺省情况下，本地认证功能能没有使能。

命令格式

set system aaa local enable { **false** | **true** }
delete system aaa local enable

参数说明

false：不使能本地认证功能。
true：使能本地认证功能。

命令模式

配置模式

使用指南

无

配置举例

使能本地认证功能:

```
ConnetOS# set system aaa local enable true
```

set system aaa tacacs-plus accounting enable

命令功能

set system aaa tacacs-plus accounting enable 命令用来配置是否使能TACACS+计费功能。
delete system aaa tacacs-plus accounting enable 命令用来删除TACACS＋计费功能使能。
缺省情况下，TACACS+计费功能并没有使能。

命令格式

set system aaa tacacs-plus accounting enable { **false** | **true** }

delete system aaa tacacs-plus accounting enable

参数说明

false：不使能TACACS＋计费功能。

true：使能TACACS＋计费功能。

命令模式

配置模式

使用指南

无。

配置举例

使能TACACS＋计费功能:

```
ConnetOS# set system aaa tacacs-plus accounting enable true
```

set system aaa tacacs-plus auth-type

命令功能

set system aaa tacacs-plus auth-type 命令用来配置TACACS+的认证类型。

delete system aaa tacacs-plus auth-type 命令用来删除TACACS＋计费功能使能。

缺省情况下，TACACS+的认证类型是ASCII。

命令格式

set system aaa tacacs-plus auth-type { **ascii** | **chap** | **pap** }

delete system aaa tacacs-plus auth-type

参数说明

ascii：ASCII认证类型。

chap：CHAP认证类型。

pap：PAP认证类型。

命令模式

配置模式

使用指南

无

配置举例

配置TACACS+的认证类型为CHAP:

```
ConnetOS# set system aaa tacacs-plus auth-type chap
```

set system aaa tacacs-plus authorization enable

命令功能

set system aaa tacacs-plus authorization enable 命令用来配置是否使能TACACS+授权功能。
delete system aaa tacacs-plus authorization enable 命令用来删除TACACS＋授权功能使能。
缺省情况下，TACACS+授权功能是使能的。

命令格式

set system aaa tacacs-plus authorization enable { **false** | **true** }
delete system aaa tacacs-plus authorization enable

参数说明

false：不使能TACACS＋授权功能。
true：使能TACACS＋授权功能。

命令模式

配置模式

使用指南

无。

配置举例

使能TACACS+授权功能:

```
ConnetOS# set system aaa tacacs-plus authorization enable true
```

set system aaa tacacs-plus enable

命令功能

set system aaa tacacs-plus enable 命令用来配置是否使能TACACS+功能。

delete system aaa tacacs-plus enable 命令用来删除TACACS＋功能。

缺省情况下，TACACS+功能没有使能。

命令格式

set system aaa tacacs-plus accounting enable { false | true }

delete system aaa tacacs-plus accounting enable

参数说明

false：不使能TACACS＋功能。

true：使能TACACS＋功能。

命令模式

配置模式

使用指南

使能后，TACACS功能直接生效。

配置举例

使能TACACS功能直接生效:

```
ConnetOS# set system aaa tacacs-plus enable true
```

system aaa tacacs-plus host server-ip

命令功能

set system aaa tacacs-plus host server-ip 命令用来指定TACACS＋服务器。

delete system aaa tacacs-plus host server-ip 命令用来删除指定的TACACS＋服务器。
缺省情况下，并没有指定TACACS＋服务器。

命令格式

set system aaa tacacs-plus host server-ip *ip-address*
delete system aaa tacacs-plus host server-ip *ip-address*

参数说明

ip-address：TACACS＋服务器的IP地址。

命令模式

配置模式

使用指南

TACACS＋服务器可以指定多个。

配置举例

指定IP地址为1.1.1.1的TACACS＋服务器作为本设备的TACACS＋服务器:

```
ConnetOS# set system aaa tacacs-plus host server-ip 1.1.1.1
```

set system aaa tacacs-plus key

命令功能

set system aaa tacacs-plus key 命令用来配置是TACACS+服务器的共享密钥。
delete system aaa tacacs-plus key 命令用来删除TTACACS+服务器的共享密钥。
缺省情况下，TACACS+服务器没有共享密钥。

命令格式

set system aaa tacacs-plus key *shared-key*
delete system aaa tacacs-plus key

参数说明

shared-key：共享密钥。

命令模式

配置模式

使用指南

无。

配置举例

配置TACACS+服务器的共享密钥为test:

```
ConnetOS# set system aaa tacacs-plus key test
```

系统时间设置命令

set date

命令功能

set date 命令用来设置系统时间。

命令格式

set date *date*

参数说明

date：系统时间。可以设置的系统时间包括：hh:mm[:ss]; [YYYY.]MM.DD-hh:mm[:ss]; MMDDhhmm[YYYY][.ss]

命令模式

运维模式

使用指南

无

配置举例

设置当前时间为14:52:00:

```
ConnetOS> set date 14:52:00
```

set system ntp-server-ip

命令功能

set system ntp-server-ip 命令用来指定NTP服务器的地址。

delete system ntp-server-ip 命令用来删除配置的NTP服务器地址。

缺省情况下，没有配置NTP服务器。

命令格式

set system ntp-server-ip *ip-address*

delete system ntp-server-ip

参数说明

ip-address：NTP服务器的地址

命令模式

配置模式

使用指南

无

配置举例

指定NTP服务器的地址为1.1.1.1:

```
ConnetOS# set system ntp-server-ip 1.1.1.1
```

set system timezone

命令功能

set system timezone 命令用来配置系统时区。

delete system timezone 命令用来删除配置的系统时区，恢复为缺省值。

缺省情况下，系统时区为UTC。

命令格式

set system timezone *timezone*

delete system timezone

参数说明

timezone：系统时区。系统支持的时区，请在设备上查看。

命令模式

配置模式

使用指南

无

配置举例

设置系统时区为Africa/Malabo:

```
ConnetOS# set system timezone Africa/Malabo
```

show system ntp-status

命令功能

show system ntp-status 命令用来查看NTP服务器的状态。

命令格式

show system ntp-status

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看NTP服务器的状态:

```
ConnetOS> show system ntp-status
synchronised to NTP server (192.168.1.102) at stratum 3
  time correct to within 1247 ms
   polling server every 64 s
```

日志命令

clear log

命令功能

clear log 命令用来清除日志文件。

命令格式

clear log { *file-name* | **all** }

参数说明

file-name：日志文件名称。
all：清除所有的日志文件。

命令模式

运维模式

使用指南

无。

配置举例

清除所有的日志文件:

```
ConnetOS> clear log all
```

set system syslog host server-ip

命令功能

set system syslog host server-ip 命令用来指定远端日志服务器。
delete system syslog host server-ip 命令用来删除指定的日志服务器。
缺省情况下，没有指定远端日志服务器。

命令格式

set system syslog host server-ip *ip-address*
delete system syslog host [**server-ip** *ip-address*]

参数说明

ip-address：远端日志服务器的地址。ConnetOS可以指定多个日志服务器。

命令模式

配置模式

使用指南

如果不指定具体的IP地址，将删除所有的指定远端日志服务器。

配置举例

ConnetOS将日志发送到地址2.2.2.2的日志服务器:

```
ConnetOS# set system syslog host server-ip 2.2.2.2
```

set system syslog log-facility

命令功能

set system syslog log-facility 命令用来日志的facility级别。
delete system syslog log-facility 命令用来删除用户配置的facility级别，恢复缺省值。
缺省情况下，日志的facility级别是0。

命令格式

set system syslog log-facility *facility-number*
delete system syslog log-facility

参数说明

facility-number：facility的编号。整数形式，取值范围是0～7。缺省值是0。

命令模式

配置模式

使用指南

日志服务器根据facility的值对日志进行存储，facility值相同的日志会被存储在同一个.log文件中。

配置举例

设置日志的facility的级别是3:

```
ConnetOS# set system aaa tacacs-plus key connetos
```

set system syslog log-level

命令功能

set system syslog log-level 命令用来配置日志的级别。
delete system syslog log-facility 命令用来删除配置的日志级别，恢复到缺省值。
缺省情况下，日志的级别是 **warning**。

命令格式

set system syslog log-level { **error** | **fatal** | **info** | **trace** | **warning** }
delete system syslog log-level

参数说明

error：错误。
fatal：致命。
info：提示。
trace：跟踪。
warning：警告。

命令模式

配置模式

使用指南

无。

配置举例

配置日志的级别是error:

```
ConnetOS# set system syslog log-level error
```

show syslog

命令功能

show syslog 命令用来查看设备上的日志信息。

命令格式

show syslog { **date** *date* | **last-rows** *row-number* }

参数说明

date：查看指定日期的日志信息，取值形式为：YEAR.MM.DD
row-number：查看指定行数的日志。整数形式，取值范围是1～。

命令模式

运维模式

使用指南

无。

配置举例

查看最新存储的4条日志:

```
ConnetOS> show syslog last-rows 4
Apr 12 2017 11:36:50 ConnetOS local2.debug : [1][cli_sh] Parsing configuration
Apr 12 2017 11:36:57 ConnetOS local2.debug : [1][cli_sh] Starting CLI
Apr 12 2017 11:36:57 ConnetOS local0.warning : admin logined the switch cli
Apr 12 2017 11:37:10 ConnetOS local0.warning : [1][cli_sh] Executing command by␣
↪admin: "show log last-rows 4"
```

syslog monitor

命令功能

syslog monitor 命令用来设置是否打开日志监控功能。

命令格式

syslog monitor { **off** | **on** }

参数说明

off：关闭日志监控功能。
on：打开日志监控功能。

命令模式

运维模式

使用指南

无。

配置举例

打开日志监控功能:

```
ConnetOS> syslog monitor on
```

系统信息查询命令参考

show system boot-messages

命令功能

show system boot-messages 命令用来查看启动信息。

命令格式

show system boot-messages

参数说明

无

命令模式

运维模式

使用指南

无

配置举例

查看系统的启动信息:

```
ConnetOS 1> show system boot-messages
[    0.000000] Initializing cgroup subsys cpuset
[    0.000000] Initializing cgroup subsys cpu
[    0.000000] Initializing cgroup subsys cpuacct
[    0.000000] Linux version 3.16.7-ckt25+ (root@host-b) (gcc version 4.9.2 (Debian 4.
↪9.2-10) ) #1 SMP Thu Mar 2 10:28:57 CST 2017
[    0.000000] Command line: BOOT_IMAGE=/boot/vmlinuz-3.16.7-ckt25+ root=LABEL=/ ro␣
↪console=tty0 console=ttyS1,115200n8 quiet
[    0.000000] e820: BIOS-provided physical RAM map:
[    0.000000] BIOS-e820: [mem 0x0000000000000000-0x000000000009abff] usable
[    0.000000] BIOS-e820: [mem 0x000000000009ac00-0x000000000009ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000000e0000-0x00000000000fffff] reserved
[    0.000000] BIOS-e820: [mem 0x0000000000100000-0x000000007f15bfff] usable
[    0.000000] BIOS-e820: [mem 0x000000007f15c000-0x000000007f18bfff] reserved
[    0.000000] BIOS-e820: [mem 0x000000007f18c000-0x000000007f234fff] usable
[    0.000000] BIOS-e820: [mem 0x000000007f235000-0x000000007f4a2fff] ACPI NVS
[    0.000000] BIOS-e820: [mem 0x000000007f4a3000-0x000000007f639fff] reserved
[    0.000000] BIOS-e820: [mem 0x000000007f63a000-0x000000007f7fffff] usable
[    0.000000] BIOS-e820: [mem 0x00000000e0000000-0x00000000e3ffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fed01000-0x00000000fed03fff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fed08000-0x00000000fed08fff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fed0c000-0x00000000fed0ffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fed1c000-0x00000000fed1cfff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000fef00000-0x00000000feffffff] reserved
[    0.000000] BIOS-e820: [mem 0x00000000ff800000-0x00000000ffffffff] reserved
[    0.000000] NX (Execute Disable) protection: active
[    0.000000] SMBIOS 0.8 present.
[    0.000000] DMI: Accton AS5512-54X/AS5512-54X, BIOS 5.6.5 08/20/2015
[    0.000000] e820: update [mem 0x00000000-0x00000fff] usable ==> reserved
[    0.000000] e820: remove [mem 0x000a0000-0x000fffff] usable
[    0.000000] AGP: No AGP bridge found
[    0.000000] e820: last_pfn = 0x7f800 max_arch_pfn = 0x400000000
[    0.000000] MTRR default type: write-back
[    0.000000] MTRR fixed ranges enabled:
[    0.000000]   00000-9FFFF write-back
[    0.000000]   A0000-BFFFF uncachable
[    0.000000]   C0000-E7FFF write-through
[    0.000000]   E8000-FFFFF write-protect
[    0.000000] MTRR variable ranges enabled:
[    0.000000]   0 base 080000000 mask F80000000 uncachable
[    0.000000]   1 base 100000000 mask F00000000 uncachable
[    0.000000]   2 base 200000000 mask E00000000 uncachable
[    0.000000]   3 base 400000000 mask C00000000 uncachable
```

```
[    0.000000]   4 base 800000000 mask 800000000 uncachable
[    0.000000]   5 base 07F800000 mask FFF800000 uncachable
[    0.000000]   6 disabled
[    0.000000]   7 disabled
[    0.000000] x86 PAT enabled: cpu 0, old 0x7040600070406, new 0x7010600070106
[    0.000000] found SMP MP-table at [mem 0x000fd6b0-0x000fd6bf] mapped at␣
↪[ffff8800000fd6b0]
[    0.000000] Base memory trampoline at [ffff880000094000] 94000 size 24576
[    0.000000] init_memory_mapping: [mem 0x00000000-0x000fffff]
[    0.000000]  [mem 0x00000000-0x000fffff] page 4k
[    0.000000] BRK [0x01b09000, 0x01b09fff] PGTABLE
[    0.000000] BRK [0x01b0a000, 0x01b0afff] PGTABLE
[    0.000000] BRK [0x01b0b000, 0x01b0bfff] PGTABLE
[    0.000000] init_memory_mapping: [mem 0x7ee00000-0x7effffff]
[    0.000000]  [mem 0x7ee00000-0x7effffff] page 2M
[    0.000000] BRK [0x01b0c000, 0x01b0cfff] PGTABLE
[    0.000000] init_memory_mapping: [mem 0x7c000000-0x7edfffff]
[    0.000000]  [mem 0x7c000000-0x7edfffff] page 2M
[    0.000000] init_memory_mapping: [mem 0x00100000-0x7bffffff]
[    0.000000]  [mem 0x00100000-0x001fffff] page 4k
--More--
```

show system core-dumps

命令功能

sshow system core-dumps 命令用来查看CPU的利用率。

命令格式

show system core-dumps

参数说明

无

命令模式

运维模式

使用指南

无

配置举例

查看本设备的CPU的利用率:

```
ConnetOS> show system cpu-usage
Cpu usage: 2%
```

show system cpu-usage

命令功能

show system cpu-usage 命令用来查看CPU的利用率。

命令格式

单机模式：**show system cpu-usage**

堆叠模式：**show system cpu-usage** [**all** | **member** *member-id*]

参数说明

all：查看堆叠系统内所有设备的CPU的利用率。

member-id：查看指定 *member-id* 设备的CPU的利用率。

命令模式

运维模式

使用指南

无

配置举例

查看本设备的CPU的利用率:

```
ConnetOS> show system cpu-usage
Cpu usage: 2%
```

show system ddm

命令功能

show system ddm 命令用来查看设备上光模块DDM信息。

命令格式

单机模式：**show system ddm** [*interface-name*]

堆叠模式：**show system ddm** [**all** | **member** *member-id* | *interface-name*]

参数说明

all：查看堆叠系统内所有设备的光模块DDM信息。

member-id：查看指定 *member-id* 设备的光模块DDM信息。

interface-name：接口名称。查看指定接口的光模块DDM信息。

命令模式

运维模式

使用指南

无。

配置举例

查看设备上所有光模块DDM信息:

```
ConnetOS> show system ddm
Interface   Temp(C)  Voltage(V)  Bias(mA)  Tx Power(dBm)  Rx Power(dBm)  Module Type
----------  -------  ----------  --------  -------------  -------------  -----------
te-1/1/6    36.24    3.39        8.10      -1.94          -2.80          SR/850nm
te-1/1/7    28.45    3.35        5.14      -2.32          -2.75          SR/850nm
te-1/1/34   32.33    3.37        8.09      -2.15          -2.24          SR/850nm
qe-1/1/52   35.78    3.29        5.83      -3.61          -2.52          SR4/850nm
qe-1/1/54   33.78    3.33        NA        NA             NA             SR4/850nm
```

show system fan

命令功能

show system fan 命令用来查询风扇的转速和PWM值。

命令格式

单机模式：**show system fan**

堆叠模式：**show system fan** [**all** | **member** *member-id*]

参数说明

all：查看堆叠系统内所有设备的风扇信息。

member-id：查看指定 *member-id* 设备的风扇信息。

命令模式

运维模式

使用指南

无

配置举例

查询风扇的转速和PWM值:

```
ConnetOS> show system fan
Fan Status:
    Fan 1 : speed =  8925 RPM, PWM =  40%
    Fan 2 : speed =  8850 RPM, PWM =  40%
    Fan 3 : speed =  8850 RPM, PWM =  40%
    Fan 4 : speed =  8850 RPM, PWM =  40%
    Fan 5 : speed =  8850 RPM, PWM =  40%
```

show system memory-usage

命令功能

show system memory-usage 命令用来查询内存的使用率。

命令格式

单机模式：**show system memory-usage**

堆叠模式：**show system memory-usage** [**all** | **member** *member-id*]

参数说明

all：查看堆叠系统内所有设备的内存使用率。

member-id：查看指定 *member-id* 设备的内存使用率。

命令模式

运维模式

使用指南

无

配置举例

查询内存使用率:

```
ConnetOS> show system memory-usage
             total       used       free     shared    buffers     cached
Mem:       2045380     550172    1495208      10640      16400     130444
-/+ buffers/cache:     403328    1642052
Swap:      7680300          0    7680300
```

show system processes

命令功能

show system processes 命令用来查看设备上对所有操作的系统处理过程。

命令格式

单机模式：**show system processes**
堆叠模式：**show system processes** [**all** | **member** *member-id*]

参数说明

all：查看堆叠系统内所有设备的系统处理过程
member-id：查看指定 *member-id* 设备的系统处理过程。

命令模式

运维模式

使用指南

无

配置举例

查看设备上对所有操作的系统处理过程:

```
ConnetOS> show system processes
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.2  28672  4648 ?        Ss   May12   0:13 /sbin/init
root         2  0.0  0.0      0     0 ?        S    May12   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    May12   0:01 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    May12   0:07 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    May12   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    May12   0:00 [migration/0]
root        10  0.0  0.0      0     0 ?        S    May12   0:00 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    May12   0:00 [watchdog/1]
root        12  0.0  0.0      0     0 ?        S    May12   0:00 [migration/1]
root        13  0.0  0.0      0     0 ?        S    May12   0:00 [ksoftirqd/1]
root        15  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/1:0H]
root        16  0.0  0.0      0     0 ?        S    May12   0:00 [watchdog/2]
root        17  0.0  0.0      0     0 ?        S    May12   0:00 [migration/2]
root        18  0.0  0.0      0     0 ?        S    May12   0:00 [ksoftirqd/2]
root        20  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/2:0H]
root        21  0.0  0.0      0     0 ?        S    May12   0:00 [watchdog/3]
root        22  0.0  0.0      0     0 ?        S    May12   0:00 [migration/3]
root        23  0.0  0.0      0     0 ?        S    May12   0:00 [ksoftirqd/3]
root        25  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/3:0H]
root        26  0.0  0.0      0     0 ?        S<   May12   0:00 [khelper]
root        27  0.0  0.0      0     0 ?        S    May12   0:00 [kdevtmpfs]
root        28  0.0  0.0      0     0 ?        S<   May12   0:00 [netns]
root        29  0.0  0.0      0     0 ?        S    May12   0:00 [khungtaskd]
root        30  0.0  0.0      0     0 ?        S<   May12   0:00 [writeback]
root        31  0.0  0.0      0     0 ?        SN   May12   0:00 [ksmd]
root        32  0.0  0.0      0     0 ?        SN   May12   0:00 [khugepaged]
root        33  0.0  0.0      0     0 ?        S<   May12   0:00 [crypto]
root        34  0.0  0.0      0     0 ?        S<   May12   0:00 [kintegrityd]
root        35  0.0  0.0      0     0 ?        S<   May12   0:00 [bioset]
root        36  0.0  0.0      0     0 ?        S<   May12   0:00 [kblockd]
root        37  0.0  0.0      0     0 ?        S    May12   0:00 [khubd]
root        38  0.0  0.0      0     0 ?        S<   May12   0:00 [devfreq_wq]
root        39  0.0  0.0      0     0 ?        S    May12   0:04 [kworker/0:1]
root        40  0.0  0.0      0     0 ?        S    May12   0:00 [kswapd0]
root        41  0.0  0.0      0     0 ?        S<   May12   0:00 [vmstat]
root        42  0.0  0.0      0     0 ?        S    May12   0:00 [fsnotify_mark]
root        49  0.0  0.0      0     0 ?        S<   May12   0:00 [kthrotld]
root        51  0.0  0.0      0     0 ?        S<   May12   0:00 [ipv6_addrconf]
root        52  0.0  0.0      0     0 ?        S<   May12   0:00 [deferwq]
root        58  0.0  0.0      0     0 ?        S    May12   0:00 [kworker/3:1]
root        96  0.0  0.0      0     0 ?        S<   May12   0:00 [ata_sff]
root        98  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_0]
root        99  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_0]
root       100  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_1]
root       101  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_1]
root       102  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_2]
root       103  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_2]
root       104  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_3]
root       105  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_3]
root       110  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_4]
root       111  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_4]
root       112  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_5]
root       113  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_5]
root       116  0.0  0.0      0     0 ?        S    May12   0:00 [scsi_eh_6]
root       117  0.0  0.0      0     0 ?        S<   May12   0:00 [scsi_tmf_6]
```

```
root         118  0.0  0.0      0     0 ?        S    May12   0:06 [usb-storage]
root         119  0.0  0.0      0     0 ?        S    May12   0:06 [kworker/1:1]
root         121  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/3:1H]
root         122  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/2:1H]
root         123  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/1:1H]
root         130  0.0  0.0      0     0 ?        S<   May12   0:00 [kworker/0:1H]
root         142  0.0  0.0      0     0 ?        S    May12   0:00 [jbd2/sda1-8]
root         143  0.0  0.0      0     0 ?        S<   May12   0:00 [ext4-rsv-conver]
root         174  0.0  0.0      0     0 ?        S    May12   0:00 [kworker/1:2]
root         177  0.0  0.0      0     0 ?        S    May12   0:00 [kauditd]
root         186  0.0  0.3  38456  6364 ?        Ss   May12   0:06 /lib/systemd/systemd-
↪journald
root         190  0.0  0.1  40668  3340 ?        Ss   May12   0:00 /lib/systemd/systemd-
↪udevd
root         257  0.0  0.0      0     0 ?        S<   May12   0:00 [kvm-irqfd-clean]
root         405  0.0  0.1  19856  2540 ?        Ss   May12   0:00 /lib/systemd/systemd-
↪logind
message+     411  0.0  0.1  42124  3360 ?        Ss   May12   0:00 /usr/bin/dbus-daemon␣
↪--system --address=systemd: --nofork --nopidfile --systemd-activation
root         437  0.1  1.8 115756 37476 ?        S    May12   4:28 /switch/bin/rtrmgr -
↪L local0.info
root         490  0.0  0.1  62852  3360 ttyS1    Ss   May12   0:00 /bin/login --
root         506  0.0  0.6  92760 14052 ?        S    May12   0:25 atp
root         507  0.0  0.7  73828 15840 ?        S    May12   0:14 cardmgr
root         508  0.0  2.5 134088 51636 ?        S    May12   2:16 sif
root         585  0.0  0.4  14716  9044 ?        Ss   May12   0:00 dhclient eth0
root         592  0.0  1.3  92636 26680 ?        S    May12   0:20 login
root         605  0.0  0.1 258676  3208 ?        Ssl  May12   0:01 /usr/sbin/rsyslogd -n
root         713  0.0  0.1  20216  2088 ?        Ss   May12   0:00 /usr/sbin/xinetd -
↪pidfile /run/xinetd.pid -stayalive -inetd_compat -inetd_ipv6
root         864  0.0  0.1  27504  2816 ?        Ss   May12   0:00 /usr/sbin/cron -f
root         866  0.0  0.5  90336 11768 ?        S    May12   0:46 pcmgr
root         867  0.0  0.9  98148 20220 ?        S    May12   4:00 stat
root         870  0.0  0.7  87564 15684 ?        S    May12   0:20 lacp
root         871  5.5  6.1 552332 125840 ?       Sl   May12 236:23 lcmgr
root         875  0.0  0.8  95592 17600 ?        S    May12   1:25 lldp
ntp          887  0.0  0.2  29168  4232 ?        Ss   May12   0:07 /usr/sbin/ntpd -p /
↪var/run/ntpd.pid -g -u 106:111
root         891  0.0  0.6  69032 13992 ?        S    May12   0:13 policy
root         892  0.0  0.7  67408 15332 ?        S    May12   0:23 static_routes
root         893  0.0  0.8  93424 18284 ?        S    May12   0:26 ospfv2
root         986  0.0  0.1  21940  3736 ttyS1    S    May12   0:00 -bash
root        1005  0.0  1.4 2538224 30524 ttyS1   Sl+  May12   0:13 ./rest
root        1960  0.0  0.0      0     0 ?        S    May12   0:05 [kworker/3:0]
root        7663  0.0  0.1  18976  2120 ?        Ss   16:00   0:00 in.telnetd: 192.168.
↪1.128
root        7664  0.0  0.1  61252  2912 pts/0    Ss   16:00   0:00 login -h 192.168.1.
↪128 -p
admin       7737  0.0  0.1  21876  3780 pts/0    S    16:01   0:00 -bash
root        7758  0.0  0.0      0     0 ?        S    May13   0:13 [kworker/2:0]
root        8430  0.0  0.0      0     0 ?        S    16:10   0:00 [kworker/2:2]
root        9001  0.0  0.0      0     0 ?        S    08:46   0:00 [kworker/u8:1]
root        9635  0.0  0.0      0     0 ?        S    16:26   0:00 [kworker/u8:0]
root       12794  0.0  0.0      0     0 ?        S    17:10   0:00 [kworker/u8:2]
admin      13082  0.0  0.1  13244  2808 pts/0    S+   17:15   0:00 /bin/sh /usr/bin/cli
admin      13088  1.5  0.8 176648 16972 pts/0    S+   17:15   0:00 /switch/bin/cli_sh
root       13582  0.0  0.0      0     0 ?        S    May14   0:02 [kworker/0:2]
```

show system rpsu

命令功能

show system rpsu 命令用来查询电源信息。

命令格式

show system rpsu

show system rpsu [**all** | **member** *member-id*]

参数说明

all：查看堆叠模式下所有设备的电源信息。

member-id：查看指定 *member-id* 设备的电源信息。

命令模式

运维模式

使用指南

本命令可以查询电源模块的SN号、温度、风扇转速、电压、电流和功率，并能对电源模块的状态进行监控展示。

配置举例

查询电源信息:

```
ConnetOS> show system rpsu
RPSU 1:
    Module Status    : OK
    Serial Number    : SA020T051623000218
    Temperature      : 29      Centigrade
    IIN              : 0.43    A
    VIN              : 215.00  V
    PIN              : 89.00   W
    IOUT             : 6.16    A
    VOUT             : 12.02   V
    POUT             : 74.00   W
RPSU 2:
    Module Status    : Unpresent
```

show system running-mode

命令功能

show system running-mode 查看系统的运行状态，单机还是堆叠。

命令格式

show system running-mode

参数说明

无

命令模式

运维模式

使用指南

无

配置举例

查看设备的运行状态:

```
ConnetOS> show system running-mode
Current mode      :  Standalone
```

show system serial-number

命令功能

show system serial-number 命令用来查询系统的SN信息，包括主板SN、电源模块SN以及光模块信息。

命令格式

单机模式：**show system serial-number**

堆叠模式：**show system serial-number** [**all** | **member** *member-id*]

参数说明

all：查看堆叠模式下所有设备的SN信息。

member-id：查看指定 *member-id* 设备的SN信息。

命令模式

运维模式

使用指南

无。

配置举例

查询产品序列号信息:

```
ConnetOS> show system serial-number
MotherBoard Serial Number : SW047110G826000010
RPSU 1 Serial Number      : SA020T051623000218
RPSU 2 is not ready.
SFP+ te-1/1/6
    Vendor Name           : FINISAR CORP.
    Serial Number         : MUG0ZRH
    Product Number        : FTLX8571D3BCL
    Module Type           : SR/850nm
    Cable Length          : 300.0m
SFP+ te-1/1/7
    Vendor Name           : CISCO-SUMITOMO
    Serial Number         : SPC150704J2
    Product Number        : SPP5100SR-C5
    Module Type           : SR/850nm
    Cable Length          : 300.0m
SFP+ te-1/1/34
    Vendor Name           : FINISAR CORP.
    Serial Number         : MUG1702
    Product Number        : FTLX8571D3BCL
    Module Type           : SR/850nm
    Cable Length          : 300.0m
QSFP+ qe-1/1/52
    Vendor Name           : Yunqi
    Serial Number         : RD160900420010
    Product Number        : RTXM320-571
    Module Type           : SR4/850nm
    Cable Length          : 300.0m
QSFP+ qe-1/1/54
    Vendor Name           : FINISAR CORP
    Serial Number         : XUC06GP
    Product Number        : FTL410QE2C
    Module Type           : SR4/850nm
    Cable Length          : 100.0m
```

show system temperature

命令功能

show system temperature 命令用来查看设备的温度信息。

命令格式

show system temperature

参数说明

无

命令模式

运维模式

使用指南

无

配置举例

查询设备温度信息:

```
ConnetOS> show system temperature
Temperature: 31 Centigrade
    Sensor 1 Temperature :  32 Centigrade
    Sensor 2 Temperature :  33 Centigrade
    Sensor 3 Temperature :  30 Centigrade
```

show system uptime

命令功能

show system uptime 命令用来查看系统启动时间。

命令格式

单机模式：**show system uptime**

堆叠模式：**show system uptime** [**all** | **member** *member-id*]

参数说明

all：查看堆叠模式下所有系统的启动时间。

member-id：查看指定 *member-id* 设备的系统启动时间。

命令模式

运维模式

使用指南

无

配置举例

查看系统启动时间:

```
ConnetOS> show system uptime
 15:57:49 up 3 days, 18:39,  1 user,  load average: 0.29, 0.11, 0.07
```

SNMP命令参考

命令功能

命令格式

参数说明

命令模式

配置模式

使用指南

配置举例

#

```
ConnetOS#
```

命令功能

命令格式

参数说明

命令模式

配置模式

使用指南

配置举例

#

```
ConnetOS#
```

命令功能

命令格式

参数说明

命令模式

配置模式

使用指南

配置举例

#

```
ConnetOS#
```

命令功能

命令格式

参数说明

命令模式

配置模式

使用指南

配置举例

#

```
ConnetOS#
```

set system snmp-acl

命令功能

set system snmp-acl network 命令用来配置允许通过SNMP协议管理设备的用户网段。

delete system snmp-acl network 命令用来删除配置的SNMP管理用户网段，恢复为缺省值。

缺省情况下，允许所有网段的用户通过SNMP管理设备。

命令格式

set system snmp-acl network *network-address*

delete system snmp-acl [**network** *network-address*]

参数说明

network-address：允许通过SNMP管理设备的用户网段。取值形式是网段地址／掩码。

命令模式

配置模式

使用指南

无。

配置举例

设置网段地址是1.1.1.0/24的用户可以通过SNMP协议管理设备:

```
ConnetOS# set system snmp-acl network 1.1.1.0/24
```

show snmp statistics

命令功能

show snmp statistics 命令用来查看

命令格式

show snmp statistics

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看SNMP协议的统计信息:

```
ConnetOS> show snmp statistics
SNMP statistics:
Input:
Packets: 2431, Bad versions: 0, Bad community names: 0,
Bad community uses: 0, ASN parse errors: 0,
Too bigs: 0, No such names: 0, Bad values: 0,
Read onlys: 0, General errors: 0,
Total request varbinds: 79849, Total set varbinds: 0,
Get requests: 187, Get nexts: 2244, Set requests: 0,
Get responses: 0, Traps: 0,
Silent drops: 0, Proxy drops 0
Output:
Packets: 2431, Too bigs: 0, No such names: 0,
Bad values: 0, General errors: 0,
Get requests: 0, Get nexts: 0, Set requests: 0,
Get responses: 2431, Traps: 0
```

接口命令

物理接口命令

set interface gigabit-ethernet description

命令功能

set interface gigabit-ethernet description 命令用来配置接口描述。

delete interface gigabit-ethernet description 命令用来删除配置的接口描述

缺省情况下，接口下没有配置接口描述。

命令格式

set interface gigabit-ethernet *interface-name* **description** *description*

delete interface gigabit-ethernet *interface-name* **description**

参数说明

interface-name：接口名称。
description：接口描述。字符串形式，不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

增加对接口te-2/2/1的描述:

```
ConnetOS# set interface gigabit-ethernet te-2/1/1 description test
```

set interface gigabit-ethernet edge-port enable

命令功能

set interface gigabit-ethernet edge-port enable 命令用来配置是否使能指定接口为边缘接口。
delete interface gigabit-ethernet edge-port enable 命令用来删除接口使能功能。
缺省情况下，所有的接口都不是边缘接口。

命令格式

set interface gigabit-ethernet *interface-name* **edge-port enable** { **false** | **true** }
delete interface gigabit-ethernet *interface-name* **edge-port** [**enable**]

参数说明

interface-name：接口名称。
false：不使能边缘接口功能。
true：使能边缘接口功能。

命令模式

配置模式

使用指南

无。

配置举例

使能接口te-1/1/1为边缘接口:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 edge-port enable true
```

set interface gigabit-ethernet enable

命令功能

set interface gigabit-ethernet enable 命令用来配置接口功能是否使能。
delete interface gigabit-ethernet enable 命令用来删除接口使能功能。
缺省情况下，接口功能是使能的。

命令格式

set interface gigabit-ethernet *interface-name* **enable** { **false** | **true** }
delete interface gigabit-ethernet *interface-name* **enable**

参数说明

interface-name：接口名称。
false：去使能接口功能。
true：使能接口功能。

命令模式

配置模式

使用指南

无。

配置举例

使能接口te-2/2/1接口功能:

```
ConnetOS# set interface gigabit-ethernet te-2/1/1 enable true
```

set interface gigabit-ethernet ether-options 802.3ad

命令功能

set interface gigabit-ethernet ether-options 802.3ad 命令用来将指定接口加入到汇聚接口。

delete interface gigabit-ethernet ether-options 802.3ad 命令用来将指定接口从汇聚接口删除。

缺省情况下，接口没有加入任何汇聚接口。

命令格式

set interface gigabit-ethernet *interface-name* **ether-options 802.3ad** *ae-number*

delete interface gigabit-ethernet *interface-name* **ether-options 802.3ad**

参数说明

interface-name：接口名称。

ae-number：聚合接口编号，取值范围为ae1～ae46。

命令模式

配置模式

使用指南

实际使用时每个汇聚接口的成员端口数建议不要超过8个。

配置举例

将接口te-1/1/1加入到汇聚接口ae1中:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 ether-options 802.3ad ae1
```

set interface gigabit-ethernet ether-options flow-control enable

命令功能

set interface gigabit-ethernet ether-options flow-control enable 命令用来配置是否使能指定接口的流量控制功能。

delete interface gigabit-ethernet ether-options flow-control enable 命令用来删除聚合接口的流量控制功能。

缺省情况下，接口功能流量控制功能是不使能的。

命令格式

set interface gigabit-ethernet *interface-name* **ether-options flow-control enable** { **false** | **true** }

delete interface gigabit-ethernet *interface-name* **ether-options flow-control** [**enable**]

参数说明

interface-name：接口名称。

false：去使能接口的流量控制功能。

true：使能接口的流量控制功能。

命令模式

配置模式

使用指南

无。

配置举例

使能接口te-1/1/1的流量控制功能:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 ether-options flow-control enable␣
↪true
```

set interface gigabit-ethernet ether-options mac-learning enable

命令功能

set interface gigabit-ethernet ether-options mac-learning enable 命令用来配置是否使能指定接口的MAC地址学习功能。

delete interface gigabit-ethernet ether-options mac-learning enable 命令用来删除配置的接口MAC地址学习功能，恢复为缺省值。

缺省情况下，接口的MAC地址学习功能是使能的。

命令格式

set interface gigabit-ethernet *interface-name* **ether-options mac-learning enable** { **false** | **true** }

delete interface gigabit-ethernet *interface-name* **ether-options lacp mac-learning** [**enable**]

参数说明

interface-name：接口名称。

false：去使能接口的MAC地址学习功能。

true：使能接口的MAC地址学习功能。

命令模式

配置模式

使用指南

无。

配置举例

使能接口te-1/1/1的MAC地址学习功能:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 ether-options mac-learning enable␣
↪true
```

set interface gigabit-ethernet family ethernet-switching native-vlan-id

命令功能

set interface gigabit-ethernet family ethernet-switching native-vlan-id 命令用来配置指定接口的Native VLAN。

delete interface gigabit-ethernet family ethernet-switching vlan members 命令用来将指定接口从Native VLAN中删除。

缺省情况下，接口的Native VLAN为VLAN 1。

命令格式

set interface gigabit-ethernet *interface-name* **family ethernet-switching native-vlan-id** *vlan-id*

delete interface gigabit-ethernet *interface-name* **family ethernet-switching** [*native-vlan-id*]

参数说明

interface-name：接口名称。

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

无论端口模式为Access还是Trunk，都有Native VLAN ID。可以通过命令对所属的VLAN ID进行修改。

配置举例

将接口te-1/1/1的Native VLAN修改为VLAN 2:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 family ethernet-switching native-
↪vlan-id 2
```

set interface gigabit-ethernet family ethernet-switching port-mode

命令功能

set interface gigabit-ethernet family ethernet-switching port-mode 命令用来配置接口的链路类型。

delete interface gigabit-ethernet family ethernet-switching port-mode 命令用来删除用户配置的链路类型，恢复为缺省值。

缺省情况下，接口的链路类型为access。

命令格式

set interface gigabit-ethernet *interface-name* **family ethernet-switching port-mode** { **access** | **trunk** }

delete interface gigabit-ethernet *interface-name* **family** [**ethernet-switching** [**port-mode**]]

参数说明

interface-name：接口名称。

access：此类型的接口主要用来连接用户主机，用于连接接入链路，且接入链路上通过的帧为不带Tag的以太网帧。仅仅允许唯一的VLAN ID通过本接口，这个VLAN ID与接口的缺省VLAN ID相同，Access接口发往对端设备的以太网帧永远是不带标签的帧。

trunk：此类型的接口主要用来和其他交换机进行连接，用于连接干道链路，允许多个VLAN的帧（带Tag标记）通过。

命令模式

配置模式

使用指南

无。

配置举例

配置接口te-2/2/1的链路类型为trunk:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 family ethernet-switching port-mode␣
↪trunk
```

set interface gigabit-ethernet family ethernet-switching vlan members

命令功能

set interface gigabit-ethernet family ethernet-switching vlan members 命令用来将指定接口加入到多个VLAN中。

delete interface gigabit-ethernet family ethernet-switching vlan members 命令用来讲指定接口从指定VLAN中删除。

缺省情况下，聚合接口已经加入到Native VLAN1中。

命令格式

set interface gigabit-ethernet *interface-name* **family ethernet-switching vlan members** *vlan-id*

delete interface gigabit-ethernet *interface-name* **family ethernet-switching** [**vlan members** *vlan-id*]

参数说明

interface-name：接口名称。

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

如果要让一个接口属于多个VLAN，该接口的接口模式必须是Trunk。

Access模式下，一个端口只能属于一个VLAN，即Native VLAN。在Trunk模式下，可以设置一个端口属于多个VLAN。多个VLAN包括Native VLAN和其他VLAN。

配置举例

将接口te-1/1/1加入到VLAN2、VLAN3、VLAN4、VLAN5、VLAN7中:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 family ethernet-switching vlan␣
↪members 2:5,7
```

set interface gigabit-ethernet mtu

命令功能

set interface gigabit-ethernet mtu 命令用来配置接口MTU值。

delete interface gigabit-ethernet mtu 命令用来删除接口配置的MTU值，恢复到缺省值。

缺省情况下，接口的MTU值为1518。

命令格式

set interface gigabit-ethernet *interface-name* **mtu** *mtu-value*

delete interface gigabit-ethernet *interface-name* **mtu**

参数说明

interface-name：接口名称。

mtu-value：接口MTU值。整数形式，取值范围是64～9216，单位是字节。

命令模式

配置模式

使用指南

无。

配置举例

配置接口te-1/1/1的MTU为1200:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 mtu 1200
```

set interface gigabit-ethernet rate-limiting

命令功能

set interface gigabit-ethernet rate-limiting 命令用来配置指定接口上的报文限速。

delete interface gigabit-ethernet rate-limiting 命令用来删除配置的接口限速。

缺省情况下，没有配置报文限速。

命令格式

set interface gigabit-ethernet *interface-name* **rate-limiting** { **egress** | **ingress** } **kilobits** *rate-limit*

delete interface gigabit-ethernet *interface-name* **rate-limiting** [**egress** | **ingress** [**kilobits**]]

参数说明

egress：出接口方向。

ingress：入接口方向。

rate-limit：报文速率。整数形式，取值范围是1～40000000，单位是Kbit/s。

命令模式

配置模式

使用指南

无。

配置举例

配置接口te-1/1/1的入接口的报文限速为10000000 Kbit/s:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 rate-limiting ingress kilobits␣
↪10000000
```

set interface gigabit-ethernet speed

命令功能

set interface gigabit-ethernet speed 命令用来配置指定接口的接口速率。 缺省情况下，接口速率为10G。

命令格式

set interface gigabit-ethernet *interface-name* **speed** { **1000** | **10000** | **40000** }

参数说明

1000：配置接口速率为1G。
10000：配置接口速率为10G。
40000：配置接口速率为40G

命令模式

配置模式

使用指南

无。

配置举例

配置接口te-1/1/1的接口速率为40G:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 speed 40000
```

set interface gigabit-ethernet static-mac-address

命令功能

set interface gigabit-ethernet static-mac-address 命令用来配置指定接口的静态MAC地址。
delete interface gigabit-ethernet static-mac-address 命令用来删除指定接口的静态MAC地址。
缺省情况下，接口下没有配置静态MAC地址。

命令格式

set interface gigabit-ethernet *interface-name* **static-mac-address** *static-mac-address* [**vlan** *vlan-id*]
delete interface gigabit-ethernet *interface-name* **static-mac-address** *static-mac-address* [**vlan** *vlan-id*]

参数说明

interface-name：接口名称。
static-mac-address：静态MAC地址。取值形式为00:11:22:33:44:55。
vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

无。

配置举例

配置接口te-1/1/1的静态MAC地址为00:11:22:33:44:50:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 static-mac-address 00:11:22:33:44:50
```

set interface gigabit-ethernet storm-control

命令功能

set interface gigabit-ethernet storm-control 命令用来配置指定接口下的风暴控制功能。
delete interface gigabit-ethernet storm-control 命令用来删除配置的风暴控制功能，恢复到缺省值。
缺省情况下，聚合接口下的风暴控制功能。

命令格式

set interface gigabit-ethernet *interface-name* **storm-control** { **broadcast** | **multicast** | **unicast** } **kilobits** *suppress*
delete interface gigabit-ethernet *interface-name* **storm-control** { **broadcast** | **multicast** | **unicast** } [**kilobits**]

参数说明

interface-name：接口名称。
suppress：对流量的限制速率。整数形式，取值范围是1～40000000，单位是Kbit/s。

命令模式

配置模式

使用指南

无。

配置举例

配置接口te-1/1/1的对广播报文的抑制速率为10000000Kbit/s:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 storm-control broadcast kilobits␣
↪10000000
```

clear ethernet-switching table

命令功能

clear ethernet-switching table 命令用来清除以太网交换的信息。

命令格式

clear ethernet-switching table { *interface-name* | **tunnel-ethernet** *tunnel-interface* }

参数说明

interface-name：接口名称。
tunnel-interface：tunnel接口名称。

命令模式

运维模式

使用指南

无。

配置举例

清除接口te-1/1/1的信息:

```
ConnetOS> clear ethernet-switching table te-1/1/1
```

clear interface statistics

命令功能

clear interface statistics 命令用来清除接口的统计信息。

命令格式

clear interface statistics { *interface-name* | **tunnel-ethernet** *tunnel-interface* }

参数说明

interface-name：接口名称。
tunnel-interface：tunnel接口名称。

命令模式

运维模式

使用指南

无。

配置举例

清除接口te-1/1/1的统计信息:

```
ConnetOS>  clear interface statistics te-1/1/1
```

汇聚接口命令

clear lacp statistics

命令功能

clear lacp statistics 命令用来清除LACP的统计信息。

命令格式

clear lacp statistics [**aggregate-ethernet** *ae-number* | **gigabit-ethernet** *interface-name*]

参数说明

ae-number：清除指定汇聚接口的LACP信息。
interface-name：清除指定GE接口的LACP信息。

命令模式

运维模式

使用指南

无

配置举例

清除设备上所有接口的LACP的统计信息:

```
ConnetOS> clear lacp statistics
```

set interface aggregate-ethernet description

命令功能

set interface aggregate-ethernet description 命令用来配置汇聚接口的描述。
delete interface aggregate-ethernet description 命令用来删除配置的汇聚接口描述
缺省情况下，接口下没有配置汇聚接口的描述。

命令格式

set interface aggregate-ethernet *ae-number* **description** *description*
delete interface aggregate-ethernet *ae-number* **description**

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。
description：接口描述。字符串形式，不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

增加对汇聚接口ae1的描述:

```
ConnetOS# set interface aggregate-ethernet ae1 description test
```

set interface aggregate-ethernet enable

命令功能

set interface aggregate-ethernet enable 命令用来配置汇聚接口功能是否使能。
delete interface aggregate-ethernet enable 命令用来删除汇聚接口使能功能。
缺省情况下，接口功能是使能的。

命令格式

set interface aggregate-ethernet *ae-number* **enable** { **false** | **true** }

delete interface aggregate-ethernet *ae-number* **enable**

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

false：去使能接口功能。

true：使能接口功能。

命令模式

配置模式

使用指南

无。

配置举例

使能ae1的汇聚接口功能:

```
ConnetOS# set interface aggregate-ethernet ae1 enable true
```

set interface aggregate-ethernet ether-options flow-control enable

命令功能

set interface aggregate-ethernet ether-options flow-control enable 命令用来配置是否使能汇聚接口的流量控制功能。

delete interface aggregate-ethernet ether-options flow-control enable 命令用来删除汇聚接口的流量控制功能。

缺省情况下，汇聚接口的流量控制功能没有使能。

命令格式

set interface aggregate-ethernet *ae-number* **ether-options flow-control enable** { **false** | **true** }

delete interface aggregate-ethernet *ae-number* **ether-options flow-control** [**enable**]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。
false：去使能接口的流量控制功能。
true：使能接口的流量控制功能。

命令模式

配置模式

使用指南

无。

配置举例

使能ae1接口的流量控制功能:

```
ConnetOS# set interface aggregate-ethernet ae1 ether-options flow-control enable true
```

set interface aggregate-ethernet ether-options lacp enable

命令功能

set interface aggregate-ethernet ether-options lacp enable 命令用来配置汇聚接口的LACP功能是否使能。
delete interface aggregate-ethernet ether-options lacp enable 命令用来删除汇聚接口的LACP功能。
缺省情况下，汇聚接口的LACP功能是不使能的。

命令格式

set interface aggregate-ethernet *ae-number* **ether-options lacp enable** { **false** | **true** }
delete interface aggregate-ethernet *ae-number* **ether-options lacp** [**enable**]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。
false：去使能接口的LACP功能。
true：使能接口的LACP功能。

命令模式

配置模式

使用指南

无。

配置举例

使能ae1接口的LACP功能:

```
ConnetOS# set interface aggregate-ethernet ae1 ether-options lacp enable true
```

set interface aggregate-ethernet ether-options mac-learning enable

命令功能

set interface aggregate-ethernet ether-options mac-learning enable 命令用来配置汇聚接口的MAC地址学习功能是否使能。

delete interface aggregate-ethernet ether-options mac-learning enable 命令用来删除配置的汇聚接口MAC地址学习功能，恢复为缺省值。

缺省情况下，汇聚接口的MAC地址学习功能是使能的。

命令格式

set interface aggregate-ethernet *ae-number* **ether-options mac-learning enable** { **false** | **true** }

delete interface aggregate-ethernet *ae-number* **ether-options lacp mac-learning** [**enable**]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

false：去使能接口的MAC地址学习功能。

true：使能接口的MAC地址学习功能。

命令模式

配置模式

使用指南

无

配置举例

使能ae1接口的MAC地址学习功能:

```
ConnetOS# set interface aggregate-ethernet ae1 ether-options mac-learning enable true
```

set interface aggregate-ethernet ether-options min-selected-port

命令功能

set interface aggregate-ethernet ether-options min-selected-port 命令用来配置LACP的最小选中接口数量。

delete interface aggregate-ethernet ether-options min-selected-port 命令用来删除配置的最小选中接口数量，恢复到缺省值。

缺省情况下，端口选举时的最小选中接口数量为1。

命令格式

set interface aggregate-ethernet *ae-number* **ether-options min-selected-port** *port-number*

delete interface aggregate-ethernet *ae-number* **ether-options min-selected-port**

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

port-number：最小选中接口数量。整数形式，取值范围是1～72。

命令模式

配置模式

使用指南

无。

配置举例

配置LACP选举端口时最小的选中接口数量为5:

```
ConnetOS# set interface aggregate-ethernet ae1 ether-options min-selected-port 5
```

set interface aggregate-ethernet family ethernet-switching native-vlan-id

命令功能

set interface aggregate-ethernet family ethernet-switching native-vlan-id 命令用来修改汇聚接口的Native VLAN。

delete interface aggregate-ethernet family ethernet-switching vlan members 命令用来将汇聚接口从Native VLAN中删除。

缺省情况下，汇聚接口的Native VLAN为VLAN1

命令格式

set interface aggregate-ethernet *ae-number* **family ethernet-switching native-vlan-id** *vlan-id*

delete interface aggregate-ethernet *ae-number* **family ethernet-switching** [**native-vlan-id**]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

无论端口模式为Access还是Trunk，都有Native VLAN ID。可以通过命令对所属的VLAN ID进行修改。

配置举例

将汇聚接口ae1的Native VLAN修改为VLAN 2:

```
ConnetOS# set interface aggregate-ethernet ae1 family ethernet-switching native-vlan-
↪id 2
```

set interface aggregate-ethernet family ethernet-switching port-mode

命令功能

set interface aggregate-ethernet family ethernet-switching port-mode 命令用来配置汇聚接口的链路类型。

delete interface aggregate-ethernet family ethernet-switching port-mode 命令用来删除用户配置的链路类型，恢复为缺省值。

缺省情况下，接口的链路类型为access。

命令格式

set interface aggregate-ethernet *ae-number* **family ethernet-switching port-mode** { **access** | **trunk** }

delete interface aggregate-ethernet *ae-number* **family** [**ethernet-switching** [**port-mode**]]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

access：此类型的接口主要用来连接用户主机，用于连接接入链路，且接入链路上通过的帧为不带Tag的以太网帧。仅仅允许唯一的VLAN ID通过本接口，这个VLAN ID与接口的缺省VLAN ID相同，Access接口发往对端设备的以太网帧永远是不带标签的帧。

trunk：此类型的接口主要用来和其他交换机进行连接，用于连接干道链路，允许多个VLAN的帧（带Tag标记）通过。

命令模式

配置模式

使用指南

无。

配置举例

配置接口ae1的链路类型为access:

```
ConnetOS# ConnetOS# set interface aggregate-ethernet ae1 family ethernet-switching␣
↪port-mode access
```

set interface aggregate-ethernet family ethernet-switching vlan members

命令功能

set interface aggregate-ethernet family ethernet-switching vlan members 命令用来将汇聚接口加入到多个VLAN中。

delete interface aggregate-ethernet family ethernet-switching vlan members 命令用来将汇聚接口从指定VLAN中删除。

缺省情况下，汇聚接口已经加入到Native VLAN1中。

命令格式

set interface aggregate-ethernet *ae-number* **family ethernet-switching vlan members** *vlan-id*

delete interface aggregate-ethernet *ae-number* **family ethernet-switching** [**vlan members** *vlan-id*]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

如果要让一个接口属于多个VLAN，该接口的接口模式必须是Trunk。 Access模式下，一个端口只能属于一个VLAN，即Native VLAN。在Trunk模式下，可以设置一个端口属于多个VLAN。多个VLAN包括Native VLAN和其他VLAN。

配置举例

将汇聚接口ae1加入到VLAN2、VLAN3、VLAN4、VLAN5、VLAN7中:

```
ConnetOS#  set interface aggregate-ethernet ae1 family ethernet-switching vlan␣
↪members 2:5,7
```

set interface aggregate-ethernet mtu

命令功能

set interface aggregate-ethernet mtu 命令用来配置汇聚接口的MTU值。

delete interface aggregate-ethernet mtu 命令用来删除汇聚接口配置的MTU值，恢复到缺省值。

缺省情况下，接口的MTU值为1518。

命令格式

set interface aggregate-ethernet *ae-number* **mtu** *mtu-value*

delete interface aggregate-ethernet *ae-number* **mtu**

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。

mtu-value：接口MTU值。整数形式，取值范围是64～9216，单位是字节。

命令模式

配置模式

使用指南

无。

配置举例

配置接口ae1的MTU值为1200:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 mtu 1200
```

set interface aggregate-ethernet static-mac-address

命令功能

set interface aggregate-ethernet static-mac-address 命令用来配置汇聚接口的静态MAC地址。
delete interface aggregate-ethernet static-mac-address 命令用来删除汇聚接口配置的静态MAC地址。
缺省情况下，汇聚接口下没有配置静态MAC地址。

命令格式

set interface aggregate-ethernet *ae-number* **static-mac-address** *static-mac-address* [**vlan** *vlan-id*]
delete interface aggregate-ethernet *ae-number* **static-mac-address** *static-mac-address* [**vlan** *vlan-id*]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。
static-mac-address：静态MAC地址。取值形式为00:11:22:33:44:55。
vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

无。

配置举例

配置汇聚接口ae1的静态MAC地址为00:11:22:33:44:55:

```
ConnetOS# set interface aggregate-ethernet ae1 static-mac-address 00:11:22:33:44:55
```

set interface aggregate-ethernet storm-control

命令功能

set interface aggregate-ethernet storm-control 命令用来配置汇聚接口下的风暴控制功能。
delete interface aggregate-ethernet storm-control 命令用来删除配置的风暴控制功能，恢复到缺省值。
缺省情况下，汇聚接口下的风暴控制功能。

命令格式

set interface aggregate-ethernet *ae-number* **storm-control** { **broadcast** | **multicast** | **unicast** } **kilobits** *suppress*
delete interface aggregate-ethernet *ae-number* **storm-control** { **broadcast** | **multicast** | **unicast** } [**kilobits**]

参数说明

ae-number：汇聚接口编号，取值范围为ae1～ae46。
suppress：对流量的限制速率。整数形式，取值范围是1～40000000，单位是Kbit/s。

命令模式

配置模式

使用指南

无。

配置举例

配置接口ae1的对广播报文的抑制速率为10000000Kbit/s:

```
ConnetOS# set interface aggregate-ethernet ae1 storm-control broadcast kilobits␣
↪10000000
```

show lacp internal

命令功能

show lacp internal 命令用来查看汇聚接口组的LACP状态。

命令格式

show lacp internal [**aggregate-ethernet** *ae-number* | **gigabit-ethernet** *interface-name*]

参数说明

ae-number：查看指定汇聚接口的成员接口LACP状态。

interface-name：查看指定GE接口的LACP状态。

命令模式

运维模式

使用指南

无。

配置举例

查看汇聚接口组的LACP状态:

```
ConnetOS> show lacp internal
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
            D -- Synchronization, E -- Collecting, F -- Distributing,
            G -- Defaulted, H -- Expired
LACP System ID: 32768,00:03:0F:64:DA:5F
Aggregated interface: ae1
Port Number   Priority   Admin Key   Oper Key   Flag
-----------   --------   ---------   --------   -------------
te-1/1/33     32768      0x4F        0x4F       {ACG}
te-2/1/36     32768      0x4F        0x4F       {ACDEF}
Aggregated interface: ae2
Port Number   Priority   Admin Key   Oper Key   Flag
-----------   --------   ---------   --------   -------------
te-1/1/6      32768      0x50        0x50       {ACDEF}
te-2/1/6      32768      0x50        0x50       {ACDEF}
```

show lacp neighbor

命令功能

show lacp neighbor 命令用来清除LACP的统计信息。

命令格式

show lacp neighbor [**aggregate-ethernet** *ae-number* | **gigabit-ethernet** *interface-name*]

参数说明

ae-number：查看指定汇聚接口的LACP邻居。

interface-name：查看指定GE接口的LACP邻居。

命令模式

运维模式

使用指南

无

配置举例

查看LACP邻居:

```
ConnetOS> show lacp neighbor
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
        D -- Synchronization, E -- Collecting, F -- Distributing,
        G -- Defaulted, H -- Expired
Aggregated interface: ae1
Port Number   Partner System ID         Partner Port Num   Port Priority   Admin Key
↪ Oper Key   Flag
------------   ----------------------   ----------------   -------------   ---------
↪ --------   -----------
te-1/1/33     0,00:00:00:00:00:00       0                  0               0x00
↪ 0x00       {}
te-2/1/36     32768,2C:60:0C:84:61:49   28                 32768           0x00
↪ 0x46       {ACDEF}
Flags:  A -- LACP_Activity, B -- LACP_Timeout, C -- Aggregation,
        D -- Synchronization, E -- Collecting, F -- Distributing,
        G -- Defaulted, H -- Expired
Aggregated interface: ae2
Port Number   Partner System ID         Partner Port Num   Port Priority   Admin Key
↪ Oper Key   Flag
------------   ----------------------   ----------------   -------------   ---------
↪ --------   -----------
te-1/1/6      32768,2C:60:0C:84:61:49   4                  32768           0x00
↪ 0x45       {ACDEF}
te-2/1/6      32768,2C:60:0C:84:61:49   6                  32768           0x00
↪ 0x45       {ACDEF}
```

show lacp statistics

命令功能

show lacp statistics 命令用来查看LACP协议的状态。

命令格式

clear lacp statistics [**aggregate-ethernet** *ae-number* | **gigabit-ethernet** *interface-name*]

参数说明

ae-number：查看指定汇聚接口的LACP协议状态。
interface-name：查看指定GE接口的LACP协议状态。

命令模式

运维模式

使用指南

无

配置举例

查看LACP协议的状态:

```
ConnetOS> show lacp statistics
Port          LACP PDUs     LACP PDUs     Marker        Marker        Marker Resp   Marker␣
↪Resp    LACP PDUs     LACP PDUs
Number        Sent          Received      Sent          Received      Sent          ␣
↪Received       Error         Dropped
----------    ----------    ----------    ----------    ----------    ----------    -------
↪----    ----------    ----------
te-1/1/33     16865         0             0             0             0             0      ␣
↪        0             0
te-1/1/6      16869         16837         0             0             0             0      ␣
↪        0             0
te-2/1/36     16865         16865         0             0             0             0      ␣
↪        0             0
te-2/1/6      16865         16865         0             0             0             0      ␣
↪        0             0
```

环回接口命令

set loopback-interface interface address

命令功能

set loopback-interface interface address 命令用来配置Loopback接口的IP地址。
delete loopback-interface interface address 命令用来删除配置的Loopback接口的IP地址。
缺省情况下，没有配置Loopback的IP地址。

命令格式

set loopback-interface interface *lo-interface-name* **address** *ip-address* [**member** *member-id*]
delete loopback-interface interface *lo-interface-name* **address** [member**]

参数说明

lo-interface-name：Loopback接口名称，取值形式为lo100。

ip-address：Loopback接口的IP地址，取值形式为：目的IP地址/掩码长度，点分十进制格式。

member-id：堆叠系统的成员设备编号，本参数只在stack模式下显示：表示此Loopback接口所属的成员设备。所有发送到此Loopback接口的报文都将发送到所属设备上。整数形式，取值为1、2。

- 1：表示设备为master设备。
- 2：表示设备为slave设备。

命令模式

配置模式

使用指南

无。

配置举例

配置Loopback接口lo100的IP地址:

```
ConnetOS# set loopback-interface interface lo100 address 1.1.1.1/24
```

set loopback-interface interface description

命令功能

set loopback-interface interface description 命令用来配置对Loopback接口的描述。

delete loopback-interface interface description 命令用来删除配置的Loopback接口描述。

缺省情况下，没有配置对Loopback接口的描述。

命令格式

set loopback-interface interface *lo-interface-name* **description** *description*

delete loopback-interface interface *lo-interface-name* **description**

参数说明

lo-interface-name：Loopback接口名称，取值形式为lo100。

description：Loopback接口的接口描述。字符串形式，不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

配置Loopback接口lo100的接口描述:

```
ConnetOS# set loopback-interface interface lo100 description test
```

show loopback-interface

命令功能

show loopback-interface 命令用来查看Loopback接口信息。

命令格式

show loopback-interface [*lo-interface-name*]

参数说明

lo-interface-name：当前已经配置好的Loopback接口名称。

命令模式

运维模式

使用指南

无。

配置举例

查看Loopback接口信息:

```
ConnetOS> show loopback-interface
lo100: Index:101  Flags:<ENABLED,LOOPBACK>
        member 1 inet 1.1.1.1 subnet 1.1.1.0/24 broadcast 1.1.1.255
```

以太网交换命令

转发模式配置命令

set forwarding-options forwarding-mode

命令功能

set forwarding-options forwarding-mode 命令用来设置设备的转发模式。

delete forwarding-options forwarding-mode 命令用来删除配置的转发模式，恢复为缺省值。

缺省情况下，设备采用存储转发模式。

命令格式

set forwarding-options forwarding-mode { cut-through | store-and-forwarding }

delete forwarding-options forwarding-mode

参数说明

cut-through：直通模式。设备接收到基本的用于二层转发的头部信息后就进行转发，不对数据包做帧校验，时延小。

store-and-forwarding：存储转发模式。设备把完整的数据包收完才进行转发，会对数据包做帧校验。

命令模式

配置模式

使用指南

直通模式主要应用于网络环境较好的情况下，可以做到比存储转发的时延小。如果线上的应用业务对时延的敏感度没有达到这个级别，建议使用存储转发模式。

配置举例

配置设备的转发模式为直通模式:

```
ConnetOS# set forwarding-options forwarding-mode cut-through
```

MAC地址命令

set interface gigabit-ethernet static-mac-address

命令功能

set interface gigabit-ethernet static-mac-address 命令用来配置指定接口的静态MAC地址。

delete interface gigabit-ethernet static-mac-address 命令用来删除配置的指定接口静态MAC地址。

缺省情况下，没有配置静态MAC地址。

命令格式

set interface gigabit-ethernet *interface-name* **static-mac-address** *mac-address* [**vlan** *vlan-id*]

delete interface gigabit-ethernet *interface-name* **static-mac-address** *mac-address* [**vlan** *vlan-id*]

参数说明

interface-name：接口名称。

mac-address：静态MAC地址。取值形式为00:11:22:33:44:55。

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

无。

配置举例

配置vlan100下的te-1/1/1口静态MAC地址为00:22:22:22:22:22:

```
ConnetOS# set interface gigabit-ethernet te-1/1/10 static-mac-address␣
↪00:22:22:22:22:22 vlan 100
```

set forwarding-options mac-aging-time

命令功能

set forwarding-options mac-aging-time 命令用来配置MAC地址的老化时间。

delete forwarding-options mac-aging-time 命令用来删除配置的MAC地址老化时间，恢复为缺省值。

缺省情况下，MAC地址的老化时间为300s。

命令格式

set forwarding-options mac-aging-time *aging-time*

delete forwarding-options mac-aging-time

参数说明

aging-time：MAC地址的老化时间。整数形式，取值范围是60～1000000，单位是秒。

命令模式

配置模式

使用指南

无

配置举例

设置MAC地址的老化时间为100s:

```
ConnetOS# set forwarding-options mac-aging-time 100
```

clear ethernet-switching table

命令功能

clear ethernet-switching table 命令用来清除MAC地址表。

命令格式

clear ethernet-switching table { **all** | *interface-name* | **tunnel-ethernet** *tunnel-name* }

参数说明

all：清除全部MAC地址表。

interface-name：接口名称。

tunnel-name：隧道名称。

命令模式

运维模式

使用指南

无

配置举例

清空全部MAC表:

```
ConnetOS> clear ethernet-switching table all
```

VLAN命令

clear vlan-interface statistics

命令功能

clear vlan-interface statistics

命令格式

clear vlan-interface statistics *vlan-interface*

参数说明

vlan-interface：接口名称。

命令模式

运维模式

使用指南

无

配置举例

清除接口vlan5上的统计信息:

```
ConnetOS> clear vlan-interface statistics vlan5
```

set vlans vlan-id

命令功能

set vlans vlan-id 命令用来创建VLAN。
delete vlans vlan-id 命令用来删除已经创建的VLAN。
缺省情况下，已经创建了VLAN 1。

命令格式

set vlans vlan-id *vlan-id* [**description** *description* | **vlan-name** *vlan-name*]
delete vlans vlan-id *vlan-id* [**description** *description* | **vlan-name** *vlan-name*]

参数说明

vlan-id：VLAN ID，整数形式，取值范围是1～4094。
description：创建对此VLAN的描述。
vlan-name：创建此VLAN的名字。字符串形式，取值范围是1～32。缺省名称为“default”。

命令模式

配置模式

使用指南

无。

配置举例

创建VLAN 100:

```
ConnetOS# set vlans vlan-id 100
```

set vlans vlan-id l3-interface

命令功能

set vlans vlan-id l3-interface 命令用来将VLAN和三层接口绑定。
delete vlans vlan-id l3-interface 命令用来删除VLAN上绑定的三层接口。
缺省情况下，VLAN下没有绑定三层接口。

命令格式

set vlans vlan-id *vlan-id* **l3-interface** *l3-interface-name*

delete vlans vlan-id *vlan-id* **l3-interface**

参数说明

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

l3-interface-name：三层接口的名称。取值形式，如vlan100。

命令模式

配置模式

使用指南

无。

配置举例

将VLAN 100和三层接口vlan100绑定:

```
ConnetOS# set vlans vlan-id 100 l3-interface vlan100
```

set vlan-interface interface address

命令功能

set vlan-interface interface address 命令用来配置三层接口的IP地址。

delete vlan-interface interface address 命令用来删除配置的三层接口IP地址。

缺省情况下，没有为三层接口配置IP地址。

命令格式

set vlan-interface interface *l3-interface-name* **address** *vif-ip-address* **prefix-length** *prefix-length*

delete vlan-interface interface *l3-interface-name* [**address** *vif-ip-address* [**prefix-length**]]

参数说明

l3-interface-name：三层接口名称。

vif-ip-address：三层接口的IP地址。

prefix-length：地址前缀长度。整数形式，取值范围4～32。

命令模式

配置模式

使用指南

无。

配置举例

配置三层接口vlan100的IP地址及地址前缀:

```
ConnetOS# set vlan-interface interface vlan100 address 1.1.1.1 prefix-length 24
```

set vlan-interface interface description

命令功能

set vlan-interface interface description 命令用来配置对三层接口的描述。
delete vlan-interface interface description 命令用来删除配置的三层接口的描述。
缺省情况下，接口下没有配置接口描述。

命令格式

set vlan-interface interface *l3-interface-name* **description** *description*
delete vlan-interface interface *l3-interface-name* [*description*]

参数说明

l3-interface-name：三层接口名称。
description：创建对此三层接口的描述。不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

配置三层接口vlan100的接口描述:

```
ConnetOS# set vlan-interface interface vlan100 description connecttoa
```

set vlan-interface interface mtu

命令功能

set vlan-interface interface mtu 命令用来配置三层接口的MTU值。

delete vlan-interface interface mtu 命令用来删除配置的三层接口MTU值。

缺省情况下，三层接口的MTU值为1500。

命令格式

set vlan-interface interface *l3-interface-name* **mtu** *mtu-value*

delete vlan-interface interface *l3-interface-name* [**mtu**]

参数说明

l3-interface-name：三层接口名称。

mtu-value：三层接口的MTU值。整数形式，取值范围是64～9198。

命令模式

配置模式

使用指南

无。

配置举例

配置三层接口vlan100的MTU值为100:

```
ConnetOS#  set vlan-interface interface vlanq100 mtu 100
```

set vlan-interface interface dhcp-relay

命令功能

set vlan-interface interface dhcp-relay 命令用来指定DHCP Relay服务器的地址。

delete vlan-interface interface dhcp-relay 命令用来删除指定的DHCP Relay服务器。
缺省情况下，没有指定DHCP Relay服务器。

命令格式

set vlan-interface interface *l3-interface-name* **dhcp-relay server-ip** *ip-address*
delete vlan-interface interface *l3-interface-name* [**dhcp-relay** [**server-ip** *ip-address*]]

参数说明

l3-interface-name：三层接口名称。
ip-address：DHCP Relay服务器的地址。

命令模式

配置模式

使用指南

无。

配置举例

指定DHCP Relay服务器:

```
ConnetOS# set vlan-interface interface vlan100 dhcp-relay server-ip 1.1.1.1
```

show vlan-interface

命令功能

show vlan-interface

命令格式

show vlan-interface [*vlan-interface*]

参数说明

vlan-interface：接口名称。

命令模式

运维模式

使用指南

无

配置举例

查看设备上的vlan接口信息:

```
ConnetOS> show vlan-interface
Interface  Status  VLAN ID  MTU    MAC                Address
---------  ------  -------  -----  -----------------  ------------------
vlan5      Down    5        1500   00:03:0F:64:DA:5F  5.5.5.1/24
vlan7      Up      7        1500   00:03:0F:64:DA:5F  7.7.7.1/24
vlan9      Up      9        1500   00:03:0F:64:DA:5F  9.9.9.1/24
vlan20     Up      20       1500   00:03:0F:64:DA:5F  22.22.22.10/24
vlan30     Up      30       1500   00:03:0F:64:DA:5F  33.33.33.10/24
vlan100    Up      100      1500   00:03:0F:64:DA:5F  11.11.11.1/24
vlan444    Down    444      1500   00:03:0F:64:DA:5F
vlan555    Down    555      1500   00:03:0F:64:DA:5F
vlan666    Down    666      1500   00:03:0F:64:DA:5F
```

show vlans

命令功能

show vlans 命令用来查看设备上的VLAN信息。

命令格式

show vlans [**brief** | **detail** | *vlan-id*]

参数说明

brief：查看VLAN的概要信息。

detail：查看VLAN的详细信息。

vlan-id：VLAN ID。当前已经创建的VLAN。

命令模式

运维模式

使用指南

无

配置举例

查看VLAN 100信息:

```
ConnetOS> show vlans vlan-id 100
VLAN ID: 100
VLAN Name: default
Description:
vlan-interface: vlan100
Number of member ports: 3
Tagged port: None
Untagged port: te-2/1/2,  ae1,       ae2,
```

LLDP命令

set protocols lldp advertisement-interval

命令功能

set protocols lldp advertisement-interval 命令用来配置发送LLDP报文的时间间隔。

set protocols lldp advertisement-interval 命令用来删除配置的发送LLDP报文的时间间隔，恢复到缺省值。

缺省情况下，发送LLDP报文的时间间隔是30秒。

命令格式

set protocols lldp advertisement-interval *advertisement-interval*

delete set protocols lldp advertisement-interval

参数说明

advertisement-interval：发送LLDP报文的时间间隔。整数形式，取值范围为5～32768，单位是秒。

命令模式

配置模式

使用指南

发送LLDP报文的时间间隔是指设备LLDP状态一直没有发生变化或没有发现新的邻居的情况下，接口周期性的向邻居设备发送LLDP报文的时间间隔。配置该时间间隔后，每一个使能LLDP功能的接口都以该间隔为周期向邻居设备发送LLDP报文，但是各个接口发送报文的时间点可以不一致。

当调整网络拓扑的发现频率时，需要使用该命令修改发送LLDP报文的时间间隔。

配置举例

配置发送LLDP报文的时间间隔为40秒:

```
ConnetOS# set protocols lldp advertisement-interval 40
```

set protocols lldp enable

命令功能

set protocols lldp enable 命令用来配置是否使能全局的LLDP功能。
delete protocols lldp enable 命令用来删除LLDP功能。
缺省情况下，全局LLDP功能已经使能。

命令格式

set protocols lldp enable { **false** | **true** }
delete protocols lldp enable

参数说明

false：去使能全局LLDP功能。
true：使能全局LLDP功能。

命令模式

配置模式

使用指南

使能全局的LLDP功能后，缺省情况下所有接口的LLDP功能都处于使能状态。

配置举例

去使能全局的LLDP功能:

```
ConnetOS# set protocols lldp enable false
```

set protocols lldp hold-time-multiplier

命令功能

set protocols lldp hold-time-multiplier 命令用来配置邻居信息在本设备上的老化时间的时间倍数。

delete protocols lldp hold-time-multiplier 命令用来恢复邻居设备信息在本设备中保持的时间倍数的缺省值。

缺省情况下，邻居设备信息在本设备中保持的时间倍数是4。

命令格式

set protocols lldp hold-time-multiplier *hold-time-multiplier* **delete protocols lldp hold-time-multiplier**

参数说明

hold-time-multiplier：邻居信息在本设备保存的时间倍数。整数形式，取值范围为1～300。

命令模式

配置模式

使用指南

邻居信息在本地设备上的老化时间由Time to Live TLV中TTL的值来确定， 老化时间的计算公式是：TTL = Min (65535, (interval × hold))

TTL代表Time to Live，表示设备信息在邻居设备中保持的时间，取65535和interval × hold中的最小值。interval代表设备向邻居设备发送LLDP报文的时间周期，hold代表设备信息在邻居设备中保持的时间倍数。

当原来使能LLDP功能的设备去使能LLDP功能后，它的邻居设备不会马上老化掉该设备的信息，而是在等待TTL时间后再进行老化，以防止网络拓扑频繁变更。

配置举例

配置邻居设备信息在本设备中中保持的时间倍数为10:

```
ConnetOS# set protocols lldp hold-time-multiplier 10
```

set protocols lldp interface status

命令功能

set protocols lldp interface status 命令用来配置接口的LLDP工作模式。

delete protocols lldp interface status 命令用来将LLDP工作模式恢复到缺省值。

缺省情况下，接口下LLDP的工作模式为Tx/Rx模式。

命令格式

set protocols lldp interface *interface-name* **status** { **disabled** | **rx-only** | **tx-only** | **tx-rx** }

delete protocols lldp interface *interface-name* **status**

参数说明

interface-name：接口名称。此接口既可以是单接口，也可以是汇聚接口组。

disabled：既不发送也不接收LLDP报文。

rx-only：Rx模式，即只接收LLDP报文。

tx-only：Tx模式，即只发送LLDP报文。

tx-rx：Tx/Rx模式，即既可以接收又可以发送LLDP报文。

命令模式

配置模式

使用指南

通过配置LLDP的工作模式，使得指定接口只能工作在指定的工作模式，可以有效减少网络中LLDP报文的数量，降低系统负担，保证用户其他业务的正常运行。

配置举例

配置接口te-1/1/1上的LLDP功能工作在Rx模式:

```
ConnetOS# set protocols lldp interface te-1/1/1 status rx-only
```

set protocols lldp reinit-delay

命令功能

set protocols lldp reinit-delay 命令用来配置当LLDP工作模式变化时，接口初始化延迟时间。

delete protocols lldp reinit-delay 命令用来删除配置的接口初始化延迟时间，恢复到缺省值。

缺省情况下，LLDP工作模式变化，接口初始化的延迟时间为2秒。

命令格式

set protocols lldp reinit-delay *reinit-delay*

delete protocols lldp reinit-delay

参数说明

reinit-delay：接口初始化的延迟时间。整数形式，取值范围为2～5，单位是秒。

命令模式

配置模式

使用指南

当端口的LLDP工作模式发生变化时，端口将对协议状态机进行初始化操作。为了避免端口模式频繁改变导致端口不断初始化，可以配置端口的初始化延迟时间，即当端口工作模式改变时延迟一段时间再执行初始化操作。

配置举例

设置LLDP重新使能的延迟时间为3秒:

```
ConnetOS# set protocols lldp reinit-delay 3
```

set protocols lldp tlv-select enable

命令功能

set protocols lldp tlv-select enable 命令用来配置是否使能发布指定类型的TLV。

delete protocols lldp tlv-select enable 命令用来删除发布指定类型的TLV。

缺省情况下，发布所有类型的TLV。

命令格式

set protocols lldp tlv-select { **mac-phy-cfg** | **management-address** | **port-description** | **port-vlan** | **system-capabilities** | **system-description** | **system-name** } **enable** { **false** | t**rue** }

delete protocols lldp tlv-select { **mac-phy-cfg** | **management-address** | **port-description** | **port-vlan** | **system-capabilities** | **system-description** | **system-name** } **enable**

参数说明

mac-phy-cfg：MAC/PHY Configuration/Status TLV，用于标识端口的速率和双工状态、是否支持端口速率自动协商、是否已使能自动协商功能以及当前的速率和双工状态。

management-address：Management Address TLV，管理地址的TLV。管理地址是供网络管理系统标识网络设备并进行管理的地址。管理地址可以明确地标识一台设备，有利于网络拓扑的绘制，便于网络管理。

port-description：Port Description TLV，用于描述端口。

port-vlan：Port VLAN TLV，一个LLDPDU中只允许携带一个此类型的TLV。

system-capabilities：System Capabilities TLV，用于描述系统的主要功能及已经使能的功能。

system-description：System Description TLV，用于对系统进行描述。

system-name：System Name TLV，用户描述系统名称。

false：不使能发布TLV。

true：使能发布TLV。

命令模式

配置模式

使用指南

无。

配置举例

不使能发布MAC/PHY Configuration/Status TLV:

```
ConnetOS# set protocols lldp tlv-select mac-phy-cfg enable false
```

set protocols lldp transmit-delay

命令功能

set protocols lldp transmit-delay 命令用来配置本设备向邻居设备发送LLDP报文的延迟时间。

delete protocols lldp transmit-delay 命令用来删除用户配置的发送LLDP报文的延迟时间，恢复到缺省值。

缺省情况下，发送LLDP报文的延迟时间为2秒。

命令格式

set protocols lldp transmit-delay *transmit-delay* **delete protocols lldp transmit-delay**

参数说明

transmit-delay：发送LLDP报文的延迟时间。整数形式，取值范围是1～8192，单位是秒。

命令模式

配置模式

使用指南

发送LLDP报文的延迟时间是指设备状态频繁发生变化的时候，接口模块向邻居设备发送LLDP报文的最小延迟时间。设备配置该延迟时间之后，每一个使能LLDP功能的接口都以该值为最小延迟时间向邻居节点发送LLDP报文，但是各个接口发送报文的时间点可以不一致。

当设备的状态信息频繁发生变化的时候可以通过增大该延迟时间来减少设备频繁向邻居设备发送信息，以达到抑制拓扑振荡目的。

配置举例

设置设备LLDP报文的延迟时间为10秒:

```
ConnetOS# set protocols lldp transmit-delay 10
```

show lldp detail

命令功能

show lldp detail 命令用来查看LLDP功能的详细信息。

命令格式

show lldp detail

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看LLDP功能的详细信息:

```
ConnetOS> show lldp detail
LLDP: Enable
Advertisement interval: 30
Re-initialization Delay: 2
Transmit Delay: 2
Hold timer: 120
```

```
Selected TLVs:
  port_description
  system_name
  system_description
  system_capabilities
  management_address
  port_vlan_id
  mac_phy
```

show lldp local-info

命令功能

show lldp local-info 命令用来查看本设备的LLDP信息。

命令格式

show lldp local-info

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看本设备的LLDP信息:

```
ConnetOS> show lldp local-info
LLDP Local configuration details
Chassis ID: 00:03:0f:64:da:5f
System name: ConnetOS
System description: ConnetOS switch software, Release version 2.1.2, Release time␣
↪2017-03-27
15:41:46, Product name C1020
Interface     LLDP         State
----------    ---------    ---------
te-1/1/1      Enable       tx_rx
te-1/1/2      Enable       tx_rx
te-1/1/3      Enable       tx_rx
te-1/1/4      Enable       tx_rx
```

```
te-1/1/5       Enable       tx_rx
te-1/1/6       Enable       tx_rx
te-1/1/7       Enable       tx_rx
te-1/1/8       Enable       tx_rx
te-1/1/9       Enable       tx_rx
te-1/1/10      Enable       tx_rx
te-1/1/11      Enable       tx_rx
te-1/1/12      Enable       tx_rx
te-1/1/13      Enable       tx_rx
te-1/1/14      Enable       tx_rx
te-1/1/15      Enable       tx_rx
te-1/1/16      Enable       tx_rx
te-1/1/17      Enable       tx_rx
te-1/1/18      Enable       tx_rx
te-1/1/19      Enable       tx_rx
te-1/1/20      Enable       tx_rx
te-1/1/21      Enable       tx_rx
te-1/1/22      Enable       tx_rx
te-1/1/23      Enable       tx_rx
te-1/1/24      Enable       tx_rx
te-1/1/25      Enable       tx_rx
te-1/1/26      Enable       tx_rx
te-1/1/27      Enable       tx_rx
te-1/1/28      Enable       tx_rx
te-1/1/29      Enable       tx_rx
te-1/1/30      Enable       tx_rx
te-1/1/31      Enable       tx_rx
te-1/1/32      Enable       tx_rx
te-1/1/33      Enable       tx_rx
te-1/1/34      Enable       tx_rx
te-1/1/35      Enable       tx_rx
te-1/1/36      Enable       tx_rx
te-1/1/37      Enable       tx_rx
te-1/1/38      Enable       tx_rx
te-1/1/39      Enable       tx_rx
te-1/1/40      Enable       tx_rx
te-1/1/41      Enable       tx_rx
te-1/1/42      Enable       tx_rx
te-1/1/43      Enable       tx_rx
te-1/1/44      Enable       tx_rx
te-1/1/45      Enable       tx_rx
te-1/1/46      Enable       tx_rx
te-1/1/47      Enable       tx_rx
te-1/1/48      Enable       tx_rx
qe-1/1/49      Enable       tx_rx
qe-1/1/50      Enable       tx_rx
qe-1/1/51      Enable       tx_rx
qe-1/1/52      Enable       tx_rx
qe-1/1/53      Enable       tx_rx
qe-1/1/54      Enable       tx_rx
```

show lldp neighbor

命令功能

show lldp neighbor 命令用来查看LLDP的邻居信息。

命令格式

show lldp neighbor [*interface-name* [**detail**]]

参数说明

interface-name：接口名称。
detail：指定接口的LLDP邻居的详细信息。

命令模式

运维模式

使用指南

无。

配置举例

查看接口te-1/1/6邻居的LLDP详细信息:

```
ConnetOS> show lldp neighbor te-1/1/6 detail
Local Port: te-1/1/6
  Time To Live: 105
  Chassis ID: 2C:60:0C:84:61:49
  Port ID: te-1/1/4
  Port Description:
  System Name: ConnetOS
  System Description: ConnetOS switch software, Release version 1.3.2, Release time␣
↪2017-03-08
  11:27:57, Product name C1010B
  System Capability: Bridge, Router,
  Management Address: 0.0.0.0
  Default VLAN ID: 1
  Auto Negotiation: Supported, Enabled
  Physical media capabilities: Others, 10base_T, 100base_TX, 100base_TXFD, 1000base_T,
↪ 1000base
  _TFD,
  Media Attachment Unit type: 10Gbase_LX4
```

show lldp statistics

命令功能

show lldp statistics 命令用来查看LLDP的邻居信息。

命令格式

show lldp statistics [*interface-name*]

参数说明

interface-name：接口名称。

命令模式

运维模式

使用指南

无。

配置举例

查看接口te-1/1/6的LLDP统计信息:

```
ConnetOS> show lldp statistics te-1/1/6
Interface   Received             Transmitted           Unknown-TLVs          With-
↪Errors          Discarded
----------  -------------------  --------------------  --------------------  --------
↪------------  --------------------
te-1/1/6    8428                 8434                  0                     0
↪              0
```

clear lldp

命令功能

clear lldp { **entry** | **statistics** } *interface-name*

命令格式

clear lldp

参数说明

entry：清除LLDP信息表。

statistics：清除LLDP统计信息。

interface-name：清除指定接口的LLDP信息。

命令模式

运维模式

使用指南

无。

配置举例

清除LLDP统计信息:

```
ConnetOS> clear lldp statistics
```

负载分担命令

set forwarding-options load-balance ecmp algorithm

命令功能

set forwarding-options load-balance ecmp algorithm 命令用来配置等值负载均衡时的哈希（hash）算法。

delete forwarding-options load-balance ecmp algorithm 用来删除等值负载均衡的哈希算法。

缺省情况下，等值负载均衡采用的哈希算法为0，即可以进行二次哈希。

命令格式

set forwarding-options load-balance ecmp algorithm *algorithm-num*

delete forwarding-options load-balance ecmp algorithm

参数说明

algorithm-num：哈希算法，整数形式，取值范围是0～8。

命令模式

配置模式

使用指南

本命令为覆盖式命令，最后一次配置生效。

配置hash算法为0表示可以进行二次哈希（进行二次哈希的设备必须为相同设备型号）。1～8为设备内部算法，哈希算法设置为非0后，请根据具体的哈希后的效果进行调整哈希算法的值。

配置举例

设置等值负载均衡哈希算法值为4:

```
ConnetOS# set forwarding-options load-balance ecmp algorithm 4
```

set forwarding-options load-balance ecmp key

命令功能

set forwarding-options load-balance ecmp key 命令用来配置用于等值负载均衡的哈希因子。

delete forwarding-options load-balance ecmp key 用来删除配置的哈希因子。

缺省情况下，哈希因子都是使能的，即全部用于进行哈希算法。

命令格式

set forwarding-options load-balance ecmp key { dest-ip | dest-mac | ether-type | ingress-interface | ip-protocol | l4-dest-port | 4-source-port | source-ip | source-mac | vlan-id } enable { false | true }

delete forwarding-options load-balance ecmp key { dest-ip | dest-mac | ether-type | ingress-interface | ip-protocol | l4-dest-port | l4-source-port | source-ip | source-mac | vlan-id } enable

参数说明

dest-ip：目的IP地址。

dest-mac：目的MAC地址。

ether-type：以太类型。

ingress-interface：报文流入接口。

ip-protocol：IP协议。

l4-dest-port：目的端口号。

l4-source-port：源端口号。

source-ip：源IP地址。

source-mac：源MAC地址。

vlan-id：VLAN ID。

命令模式

配置模式

使用指南

负载均衡依靠哈希运算实现，哈希算法的输入是各报文的特征值，称之为哈希因子。如果哈希因子散列性较好，则哈希算法得出的负载分担将会更均匀。可作为哈希因子的特征值有报文的源目的MAC地址，源目的IP地址，源目的端口号和协议号。

配置举例

设置ip-protocol不参与哈希运算:

```
ConnetOS# set forwarding-options load-balance ecmp key ip-protocol enable false
```

set forwarding-options load-balance ecmp max-path

命令功能

set forwarding-options load-balance ecmp max-path 命令用来配置等值负载均衡时的最大链路数，即参与负载均衡的链路数。

delete forwarding-options load-balance ecmp max-path 用来删除等值负载均衡时的最大链路数。

缺省情况下，等值负载均衡最大链路数是32。

命令格式

set forwarding-options load-balance ecmp max-path *max-path*

delete forwarding-options load-balance ecmp max-path

参数说明

max-path：整数形式，取值范围是1～32。

命令模式

配置模式

使用指南

本命令为覆盖式命令，最后一次配置生效。

配置举例

设置等值负载均衡时最大链路为8:

```
ConnetOS# set forwarding-options load-balance ecmp max-path 8
```

set forwarding-options load-balance ecmp tunnel

命令功能

set forwarding-options load-balance ecmp tunnel 命令用来配置Tunnel报文用于等值负载均衡的哈希因子。

delete forwarding-options load-balance ecmp tunnel 用来删除配置的Tunnel报文的哈希因子。

缺省情况下，使用Tunnel报文外层报文头内容做哈希运算。

命令格式

set forwarding-options load-balance ecmp tunnel { **inner** | **outer** }

delete forwarding-options load-balance ecmp tunnel

参数说明

inner：使用Tunnel报文内层报文头内容做哈希运算。

outer：使用Tunnel报文外层报文头内容做哈希运算。

命令模式

配置模式

使用指南

本命令为覆盖式命令，最后一次的配置生效。

配置举例

设置使用Tunnel报文内层报文头内容做哈希运算:

```
ConnetOS# set forwarding-options load-balance ecmp tunnel inner
```

show forwarding-options load-balance ecmp

命令功能

show forwarding-options load-balance ecmp 命令用来查看等值负载均衡时哈希算法的配置信息。

命令格式

show forwarding-options load-balance ecmp [**key** [**dest-ip** | **dest-mac** | **ether-type** | **ingress-interface** | **ip-protocol** | **l4-dest-port** | **l4-source-port** | **source-ip** | **source-mac** | **vlan-id**]]

参数说明

dest-ip：目的IP地址。

dest-mac：目的MAC地址。

ether-type：以太类型。

ingress-interface：报文流入接口。

ip-protocol：IP协议。

l4-dest-port：目的端口号。

l4-source-port：源端口号。

source-ip：源IP地址。

source-mac：源MAC地址。

vlan-id：VLAN ID。

命令模式

配置模式

使用指南

无。

配置举例

查看等值负载均衡时哈希算法的配置信息:

```
ConnetOS# show forwarding-options load-balance ecmp
Waiting for building configuration.
   algorithm: "0"
   max-path: 32
   key {
       ingress-interface {
           enable: true
       }
       source-mac {
           enable: true
       }
       dest-mac {
           enable: true
       }
       ether-type {
           enable: true
       }
       vlan-id {
           enable: true
       }
       ip-protocol {
           enable: false
       }
```

```
        source-ip {
            enable: true
        }
        dest-ip {
            enable: true
        }
        l4-source-port {
            enable: true
        }
        l4-dest-port {
            enable: true
        }
    }
    tunnel: "outer"
```

set forwarding-options load-balance lag algorithm

命令功能

set forwarding-options load-balance lag algorithm 命令用来配置汇聚组负载均衡的哈希算法。

delete forwarding-options load-balance lag algorithm 用来删除负载均衡的哈希算法。

缺省情况下，负载均衡采用的哈希算法为0，即可以进行二次哈希。

命令格式

set forwarding-options load-balance lag algorithm *algorithm-num*

delete forwarding-options load-balance lag algorithm

参数说明

algorithm-num：哈希算法，整数形式，取值范围是0～8。

命令模式

配置模式

使用指南

本命令为覆盖式命令，最后一次配置生效。

配置hash算法为0表示可以进行二次哈希（进行二次哈希的设备必须为相同设备型号）。1～8为设备内部算法，哈希算法设置为非0后，请根据具体的哈希后的效果进行调整哈希算法的值。

配置举例

设置汇聚组负载均衡的哈希算法值为4:

```
ConnetOS# set forwarding-options load-balance lag algorithm 4
```

set forwarding-options load-balance lag key

命令功能

set forwarding-options load-balance lag key 命令用来配置用于汇聚组负载均衡的哈希因子。

delete forwarding-options load-balance lag key 用来删除配置的哈希因子。

缺省情况下，哈希因子都是使能的，即全部用于进行哈希算法。

命令格式

set forwarding-options load-balance lag key { dest-ip | dest-mac | ether-type | ingress-interface | ip-protocol | l4-dest-port | l4-source-port | source-ip | source-mac | vlan-id } enable { false | true }

delete forwarding-options load-balance lag key { dest-ip | dest-mac | ether-type | ingress-interface | ip-protocol | l4-dest-port | l4-source-port | source-ip | source-mac | vlan-id } enable

参数说明

dest-ip：目的IP地址。

dest-mac：目的MAC地址。

ether-type：以太类型。

ingress-interface：报文流入接口。

ip-protocol：IP协议。

l4-dest-port：目的端口号。

l4-source-port：源端口号。

source-ip：源IP地址。

source-mac：源MAC地址。

vlan-id：VLAN ID。

命令模式

配置模式

使用指南

负载均衡依靠哈希运算实现，哈希算法的输入是各报文的特征值，称之为哈希因子。如果哈希因子散列性较好，则哈希算法得出的负载分担将会更均匀。可作为哈希因子的特征值有报文的源目的MAC地址，源目的IP地址，源目的端口号和协议号。

配置举例

设置ether-type不参与哈希运算:

```
ConnetOS# set forwarding-options load-balance lag key ether-type enable false
```

set forwarding-options load-balance lag symmetric

命令功能

set forwarding-options load-balance lag symmetric 命令用来配置对称哈希。

delete forwarding-options load-balance lag symmetric 用来删除配置的对称哈希。

缺省情况下，对称哈希没有使能。

命令格式

set forwarding-options load-balance lag symmetric enable [false | true]

delete forwarding-options load-balance lag symmetric

参数说明

false：不使能对称哈希。

true：使能对称哈希。

命令模式

配置模式

使用指南

本命令为覆盖式命令，最后一次配置生效。

配置举例

设置使能对称哈希:

```
ConnetOS# set forwarding-options load-balance lag symmetric enable true
```

set forwarding-options load-balance lag tunnel

命令功能

set forwarding-options load-balance lag tunnel 命令用来配置Tunnel报文用于等值负载均衡的哈希因子。

delete forwarding-options load-balance lag tunnel 用来删除配置的Tunnel报文的哈希因子。
缺省情况下，使用Tunnel报文外层报文头内容做哈希运算。

命令格式

set forwarding-options load-balance lag tunnel { **inner** | **outer** }
delete forwarding-options load-balance lag tunnel

参数说明

inner：使用Tunnel报文内层报文头内容做哈希运算。
outer：使用Tunnel报文外层报文头内容做哈希运算。

命令模式

配置模式

使用指南

本命令为覆盖式命令，最后一次的配置生效。

配置举例

设置使用Tunnel报文内层报文头内容做哈希运算:

```
ConnetOS# set forwarding-options load-balance lag tunnel inner
```

三层转发命令

ARP命令

clear arp

命令功能

clear arp all 命令用来清除所有的ARP表项信息。 **clear arp ip-address** 命令用来清除指定的ARP表项信息。

命令格式

clear arp { **all** | **ip-address** *ip-address* }

参数说明

ip-address：IP地址。

命令模式

运维模式

使用指南

无

配置举例

清除所有的ARP表项信息:

```
ConnetOS 1> clear arp all
```

set protocols arp aging-time

命令功能

set protocols arp aging-time 命令用来配置ARP的老化时间。

delete protocols arp aging-time 命令用来删除配置的ARP的老化时间，恢复到缺省值。

缺省情况下，ARP的老化时间是1200秒。

命令格式

set protocols arp aging-time *aging-time*

delete protocols arp aging-time

参数说明

aging-time：ARP的老化时间。整数形式，取值范围是300～14400，单位是秒。

命令模式

配置模式

使用指南

为适应网络的变化，ARP表需要不断更新。在达到老化时间时，如果仍不得到刷新的ARP表项将被从ARP表中删除。如果在到达老化时间前记录被刷新，则重新计算老化时间。用户可以根据网络实际情况调整老化时间。

配置举例

配置ARP的老化时间为600s:

```
ConnetOS# set protocols arp aging-time 600
```

set protocols arp interface proxy enable

命令功能

set protocols arp interface proxy enable 命令用来配置是否使能指定三层接口的ARP代理功能。
delete protocols arp interface proxy enable 命令用来删除接口下的ARP代理功能。
缺省情况下，ARP代理功能没有使能。

命令格式

set protocols arp interface *vlan-interface* **proxy enable** { **false** | **true** }
delete protocols arp interface *vlan-interface* [**proxy** [**enable**]]

参数说明

vlan-interface：已经配置好的三层接口。接口形式为vlan100。
false：去使能ARP代理功能。
true：使能ARP代理功能。

命令模式

配置模式

使用指南

无。

配置举例

使能三层接口vlan100上的ARP代理功能:

```
ConnetOS# set protocols arp interface vlan100 proxy enable true
```

set protocols arp interface proxy address

命令功能

set protocols arp interface proxy address 命令用来配置静态ARP表项。
delete protocols arp interface proxy address 命令用来删除配置的静态ARP表项。
缺省情况下，没有配置静态ARP表项。

命令格式

set protocols arp interface *vlan-interface* **address** *ip-address* **mac-address** *mac-address*
delete protocols arp interface *vlan-interface* [**address** *ip-address* [*mac-address*]]

参数说明

vlan-interface：已经配置好的三层接口。接口形式为vlan100。
mac-address：MAC地址。形式是hh:hh:hh:hh:hh:hh，比如：00:10:94:00:00:01。

命令模式

配置模式

使用指南

无。

配置举例

配置静态ARP表项:

```
ConnetOS# set protocols arp interface vlan100 address 1.1.1.1 mac-address␣
↪cc:37:ab:f4:82:f3
```

show arp

命令功能

show arp 命令用来查看设备上的ARP信息。

命令格式

show arp

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看设备上的的ARP信息:

```
ConnetOS# ConnetOS> show arp
Aging-time(seconds): 1200
Total count         : 1
Address                HW Address        Type            Interface       Age
---------------                 -----------------               -------
↪ ---------        -----
5.5.5.5                00:00:00:12:78:19       Dynamic         vlan5       10
```

DHCP Relay命令

set vlan-interface interface dhcp-relay server-ip

命令功能

set vlan-interface interface dhcp-relay server-ip 命令用来指定DHCP Relay服务器。

set vlan-interface interface dhcp-relay server-ip 命令用来指定DHCP Relay服务器。

缺省情况下，没有指定HCP Relay服务器。

命令格式

set vlan-interface interface *l3-interface-name* **dhcp-relay server-ip** *ip-address*

delete vlan-interface interface *l3-interface-name* **dhcp-relay server-ip** *ip-address*

参数说明

l3-interface-name：vlan－interface的名称。

ip-address：DHCP Relay服务器的IP地址。

命令模式

配置模式

使用指南

无

配置举例

指定IP地址为1.1.1.1的DHCP Relay服务器作为本设备的服务器:

```
ConnetOS# set vlan-interface interface vlan3 dhcp-relay server-ip 1.1.1.1
```

路由表查看命令参考

show route admin distance ipv4 unicast

命令功能

show route admin distance ipv4 unicast 命令用来查看不同种类路由的优先级。

命令格式

show route admin distance ipv4 unicast

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看不同种类路由的优先级:

```
ConnetOS> show route admin distance ipv4 unicast
Protocol                  Administrative distance
connected                 0
static                    1
eigrp-summary             5
ebgp                      20
eigrp-internal            90
igrp                      100
ospf                      110
is-is                     115
rip                       120
eigrp-external            170
ibgp                      200
fib2mrib                  254
unknown                   255
```

show route forward-host

命令功能

show route forward-host 命令用来查看硬件路由表中本机的路由信息。

命令格式

show route forward-host { **brief** | **ipv4** { *ip-address* | **all** } }

参数说明

brief：查看表中的概要信息，即路由条目数。

ip-address：本机IP地址。

all：查看所有的路由信息。

命令模式

运维模式

使用指南

无。

配置举例

查看硬件路由表中本机的所有路由信息:

```
ConnetOS> show route forward-host ipv4 all
Address           HWaddress           Port
---------------   -----------------   ---------
9.9.9.9           00:00:00:11:22:99   te-2/1/9
Total host count:1
```

show route forward-route

命令功能

show route forward-route 命令用来查看FIB转发表中的路由条目。

命令格式

show route forward-route { **brief** | **ipv4** { *network-address* | **all** } }

参数说明

brief：查看转发表中的概要信息，即路由条目数。
network-address：目的网络地址。取值形式是网段地址／掩码。
all：查看转发表中所有的IPv4路由信息。

命令模式

运维模式

使用指南

无

配置举例

查看转发表中所有的IPv4路由信息:

```
ConnetOS> show route forward-route ipv4 all
Destination       NetMask           NextHopMac          Port
---------------   ---------------   -----------------   ---------
9.9.9.0           255.255.255.0     00:03:0F:64:DA:53   connected
11.11.11.0        255.255.255.0     00:03:0F:64:DA:53   connected
33.33.33.0        255.255.255.0     00:03:0F:64:DA:53   connected
Total route count:3
```

show route table ipv4 unicast

命令功能

show route table ipv4 unicast 命令用来查看RIB路由表中的路由条目。

命令格式

show route table ipv4 unicast { **connected** | **final** | **ospf** [**winners**]| **static** } [**brief** | **detail**]

参数说明

connected：查看直连网络的路由条目。
final：查看最终选择进行转发的路由条目。
ospf：查看通过OSPF协议学到的路由条目。
winners：查看通过OSPF协议学到的路由条目
static：查看静态路由条目。
brief：查看路由表的概要信息。
detail：查看路由表的详细信息。

命令模式

运维模式

使用指南

无

配置举例

查看路由表中最终选择进行转发的路由条目的详细信息:

```
ConnetOS> show route table ipv4 unicast final detail
Network 9.9.9.0/24
    Nexthop        := 9.9.9.1
    Interface      := vlan9
    Vif            := vlan9
    Metric         := 0
    Protocol       := connected
    Admin distance := 0
Network 11.11.11.0/24
    Nexthop        := 11.11.11.1
    Interface      := vlan100
    Vif            := vlan100
    Metric         := 0
    Protocol       := connected
```

```
    Admin distance := 0
Network 33.33.33.0/24
    Nexthop        := 33.33.33.10
    Interface      := vlan30
    Vif            := vlan30
    Metric         := 0
    Protocol       := connected
    Admin distance := 0
Network 55.55.55.0/24
    Nexthop        := 33.33.33.20
    Interface      := vlan30
    Vif            := vlan30
    Metric         := 30
    Protocol       := ospf
    Admin distance := 110
```

静态路由命令

set protocols static route

命令功能

set protocols static route 命令用来配置静态路由。

命令格式

set protocols static route *ip-address*

参数说明

ip-address：目的IP地址/掩码长度，点分十进制格式。

命令模式

配置模式

使用指南

无。

配置举例

配置到目的地址10.10.1.0/24的静态路由:

```
ConnetOS# set protocols static route 10.10.1.0/24
```

set protocols static route metric

命令功能

set protocols static route metric 命令用来配置静态路由的度量值metric。

命令格式

set protocols static route *ip-address* **metric** *metric-value*

参数说明

ip-address：目的IP地址/掩码长度，点分十进制格式。
metric-value：整数形式，取值范围是1～65535。

命令模式

配置模式

使用指南

无。

配置举例

配置到目的地址10.10.1.0/24的静态路由的metric是5:

```
ConnetOS# set protocols static route 10.10.1.0/24 metric 5
```

set protocols static route

命令功能

set protocols static route 命令用来配置静态路由的等价。

命令格式

set protocols static route *ip-address* **qualified-next-hop** *ip-address* [**metric** *metric-value*]

参数说明

ip-address：目的IP地址/掩码长度。点分十进制格式。

命令模式

配置模式

使用指南

无。

配置举例

配置到目的地址10.10.1.0/24的静态路由:

```
ConnetOS# set protocols static route 10.10.1.0/24
```

OSPF命令

set protocols ospf4 area

命令功能

set protocols ospf4 area 命令用来配置OSPF区域。

delete protocols ospf4 area 命令用来删除配置好的OSPF区域。

缺省情况下，没有配置OSPF区域。

命令格式

set protocols ospf4 area *area-id*

delete protocols ospf4 area *area-id*

参数说明

area-id：OSPF区域ID。IP地址形式，取值为点分十进制。建议和某个接口的IP地址保持一致。0.0.0.0表示骨干区域。

命令模式

配置模式

使用指南

创建好区域后，区域的缺省类型是normal。

配置举例

创建区域ID为1.1.1.1的OSPF区域:

```
ConnetOS# set protocols ospf4 area 1.1.1.1
```

set protocols ospf4 area area-range advertise enable

命令功能

set protocols ospf4 area area-range advertise enable 命令用来配置是否使能聚合路由通告功能。

delete protocols ospf4 area area-range 命令用来删除配置的聚合路由通告功能。

缺省情况下，没有使能聚合路由通告功能。

命令格式

set protocols ospf4 area *area-id* **area-range** *network-address* **advertise enable** { **false** | **true** }

delete protocols ospf4 area *area-id* **area-range** *network-address* [**advertise** [**enable**]]

参数说明

area-id：OSPF区域标识，点分十进制形式。

network-address：区域所包含的网段。取值形式是网段地址／掩码。

false：去使能聚合路由通告功能。

true：使能聚合路由通告功能。

命令模式

配置模式

使用指南

一个网段只能属于一个区域，或者说每个运行OSPF协议的接口必须指明属于某一个特定的区域。该处的网段是指运行OSPF协议接口的IP地址所在的网段。

OSPF需要对接收到的Hello报文做网络掩码检查，当接收到的Hello报文中携带的网络掩码和本设备不一致时，则丢弃这个Hello报文，即不能建立邻居关系。

配置举例

在OSPF区域1.1.1.1上使能路由聚合通告功能:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 area-range 2.2.2.0/24 advertise enable
```

set protocols ospf4 area area-type

命令功能

set protocols ospf4 area-type 命令用来配置OSPF的区域类型。

delete protocols ospf4 area area-type 用来删除配置的OSPF区域类型。

缺省情况下，OSPF的网络类型是normal。

命令格式

set protocols ospf4 area *area-id* **area-type** { **normal** | **nssa** | **stub** }

delete protocols ospf4 area *area-id* **area-type**

参数说明

area-id：区域标识，点分十进制形式。

normal：标准区域。

nssa：NSSA区域。

stub：Stub区域。

命令模式

配置模式

使用指南

对于位于AS边缘的非骨干区域，可以将区域配置为Stub区域，避免Type5 LSA在Stub区域的泛洪，减少路由表的规模。

在配置位于区域时需要注意：

- 骨干区域不能被配置成Stub区域。
- 如果要将一个区域配置成Stub区域，那么需要把区域中的所有路由设备的区域类型都配置成stub区域。
- 虚连接不能穿过Stub区域和NSSA区域。

配置举例

配置OSPF的区域类型为Stub:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 area-type stub
```

set protocols ospf4 area default-lsa enable

命令功能

set protocols ospf4 area default-lsa enable 命令用来配置是否使能在stub区域中生成缺省路由的功能。
delete protocols ospf4 area default-lsa enable 用来删除配置的在stub区域中生成缺省路由功能。
缺省情况下，没有使能stub区域生成缺省路由功能。

命令格式

set protocols ospf4 area *area-id* **default-lsa enable** { **false** | **true** }
delete protocols ospf4 area *area-id* **default-lsa enable**

参数说明

false：不使能。
true：使能

命令模式

配置模式

使用指南

无。

配置举例

在stub区域1.1.1.1中使能生成缺省路由的功能:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 default-lsa enable true
```

set protocols ospf4 area default-lsa metric

命令功能

set protocols ospf4 default-lsa metric 命令用来指定OSPF发送到Stub区域的Type3缺省路由的开销。
delete set protocols ospf4 default-lsa metric 命令用来删除配置的缺省路由开销。
缺省情况下，发送到STUB区域的Type3缺省路由的开销为0。

命令格式

set protocols ospf4 area *area-id* **default-lsa metric** *metric*

delete protocols ospf4 area *area-id* **default-lsa metric**

参数说明

metric：发送到STUB区域的Type3缺省路由的开销。

命令模式

配置模式

使用指南

本命令只能配置到连接到Stub区域的ABR上。

配置举例

设置Stub区域1.1.1.1到缺省路由的开销为32:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 default-lsa metric 32
```

set protocols ospf4 area interface address authentication

命令功能

set protocols ospf4 area interface address authentication 命令用来配置OSPF区域的接口认证方式。

delete rotocols ospf4 area interface address authentication 命令用来删除配置的接口认证方式。

缺省情况下，接口不对OSPF报文进行认证。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **authentication** { **md5** *key-id* | **simple-password** *password* }

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**authentication** [**md5** | **simple-password**]]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

key-id：MD5验证字标识符，必须和对端的验证字标识符一致。整数形式，取值范围是0～255。

password：简单密码。

命令模式

配置模式

使用指南

接口验证方式可以提高OSPF网络的安全性。用于在相邻的设备之间设置验证模式和口令，优先级高于区域验证方式。

配置举例

在接口vlan100上配置OSPF的接口认证方式为MD5:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 3.3.3.3␣
↪authentication md5 5
```

set protocols ospf4 area interface address enable

命令功能

set protocols ospf4 area interface address enable 命令用来配置是否使能接口的OSPF功能。

delete rotocols ospf4 area interface address enable 命令用来删除配置OSPF。

缺省情况下，接口下的OSPF功能没有使能。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **enable** { **false** | **true** }

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**enable**]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

false：不使能OSPF功能。

true：使能OSPF功能。

命令模式

配置模式

使用指南

区域的边界是设备，而不是链路。必须为每一个运行OSPF的接口指明所属的区域。

当此接口使能了OSPF功能之后，OSPF将把这个接口的直连路由宣告出去。

配置举例

使能三层接口vlan100的OSPF功能:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 6.6.6.6 enable␣
↪true
```

set protocols ospf4 area interface address hello-interval

命令功能

set protocols ospf4 area interface address hello-interval 命令用来配置接口发送Hello报文的时间间隔。

delete rotocols ospf4 area interface address hello-interval 命令用来删除配置的接口发送Hello报文的时间间隔，恢复为缺省值。

缺省情况下，接口发送Hello报文的时间间隔为10秒。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **hello-interval** *hello-interval*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**hello-interval**]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。

vif-ip-address：三层接口的IP地址。

hello-interval：发送Hello报文的时间间隔。整数形式，取值范围是1～65535，单位是秒。

命令模式

配置模式

使用指南

Hello报文周期性的发送给邻居路由设备，用于维持邻居关系以及DR/BDR的选举。

hello-interval 的值越小，发现网络拓扑改变的速度越快，路由开销也就越大。本接口和邻接设备的 **hello-interval** 要保持一致。

配置举例

设置Hello报文发送的时间间隔是30s:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 7.7.7.7 hello-
↪interval 30
```

set protocols ospf4 area interface address interface-cost

命令功能

set protocols ospf4 area interface address interface-cost 命令用来配置接口上运行OSPF协议所需要的开销值。

delete rotocols ospf4 area interface address interface-cost 命令用来删除配置的开销值，恢复为缺省值。

缺省情况下，OSPF接口的开销值为1。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **interface-cost** *interface-cost*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**interface-cost**]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

interface-cost：整数形式，取值范围是1～65535。

命令模式

配置模式

使用指南

当有多条发现协议、开销值、目的地址都相同的路由时，这几条路由就满足负载分担的条件。请根据实际组网情况，通过修改接口开销值来选择是否需要进行负载分担。

配置举例

配置接口vlan100的开销值为10:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 7.7.7.7␣
↪interface-cost 10
```

set protocols ospf4 area interface address neighbor

命令功能

set protocols ospf4 area interface address neighbor 命令用来指定邻居路由设备。

delete rotocols ospf4 area interface address neighbor 命令用来删除指定的邻居路由设备。

缺省情况下，没有指定邻居路由设备。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **neighbor** *ip-address* **router-id** *router-id*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**neighbor** *ip-address* [**router-id**]]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

ip-address：邻居路由设备的IP地址。

router-id：邻居路由设备的Router ID。

命令模式

配置模式

使用指南

无。

配置举例

指定邻居OSPF为2.2.2.2:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 7.7.7.7 neighbor␣
↪2.2.2.2 router-id 2.2.2.2
```

set protocols ospf4 area interface address passive enable

命令功能

set protocols ospf4 area interface address passive enable 命令用来配置是否使能只广播不运行OSPF协议功能。

delete rotocols ospf4 area interface address passive enable 命令用来恢复为缺省值。

缺省情况下，既不运行也不广播OSPF协议。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **passive** [**host**] **enable** { **false** | **true** }

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**passive** [**host**] [**enable**]]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

host：只通告本机的OSPF路由。

false：不使能。

true：使能。

命令模式

配置模式

使用指南

无。

配置举例

使能三层接口vlan100只广播不运行OSPF协议:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 7.7.7.7 passive␣
↪enable true
```

set protocols ospf4 area interface address priority

命令功能

set protocols ospf4 area interface address priority 命令用来配置广播网络中接口的DR选举优先级。

delete rotocols ospf4 area interface address priority 命令用来删除配置的DR选举优先级，恢复为缺省值。

缺省情况下，DR选举优先级为128。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **priority** *priority*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**priority**]]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

priority：本设备在DR选举时的优先级。整数形式，取值范围是0～255。值越大，优先级越高。

命令模式

配置模式

使用指南

接口的优先级决定了该接口在选举DR时所具有的资格，优先级高的接口在DR选举时被首先考虑。

如果一台设备的接口优先级为0，则它不会被选举为DR或BDR。在广播网络中，可以通过配置接口的DR优先级来影响网络中DR或BDR的选择。

当网段上选举出DR和BDR之后，它们就会向所有的邻居发送DD报文，建立邻接关系。

配置举例

配置接口vlan100的DR优先级是20:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 3.3.3.3 priority␣
↪20
```

set protocols ospf4 area interface address retransmit-interval

命令功能

set protocols ospf4 area interface address retransmit-interval 命令用来配置LSA重传时间间隔。

delete rotocols ospf4 area interface address retransmit-interval 命令用来删除配置的LSA重传时间间隔，恢复为缺省值。

缺省情况下，LSA重传的时间间隔为5秒。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **retransmit-interval** *retransmit-interval*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**retransmit-interval**]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

retransmit-interval：LSA重传的时间间隔。整数形式，取值范围是1～65535，单位是秒。

命令模式

配置模式

使用指南

在网络相对稳定、对路由收敛时间要求较高的组网环境中，可以指定LSA的更新时间间隔为0来取消LSA的更新时间间隔，使得拓扑或者路由的变化可以立即通过LSA发布到网络中，从而加快网络中路由的收敛速度。

如果对网络没有特殊要求，建议使用命令的缺省值。

配置举例

配置LSA重传的时间间隔为3秒:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 3.3.3.3␣
↪retransmit-interval 3
```

set protocols ospf4 area interface address router-dead-interval

命令功能

set protocols ospf4 area interface address router-dead-interval 命令用来配置OSPF的邻居失效时间间隔。

delete rotocols ospf4 area interface address retransmit-interval 命令用来删除配置的OSPF邻居失效时间间隔，恢复为缺省值。

缺省情况下，OSPF的邻居失效时间间隔是40秒。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **router-dead-interval** *router-dead-interval*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**router-dead-interval**]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

router-dead-interval：OSPF的邻居失效时间间隔。整数形式，取值范围是1～4294967295，单位是秒。

命令模式

配置模式

使用指南

OSPF邻居的失效时间间隔是指：在该时间间隔内，若未收到邻居的Hello报文，就认为该邻居已失效。运行OSPF接口上的邻居失效时间dead interval必须大于发送Hello报文的时间间隔hello interval，且同一网段上的设备的dead interval值也必须相同。

缺省情况下，邻居失效时间为发送Hello报文时间间隔的4倍。

配置举例

配置接口vlan100上的OSPF的邻居失效时间间隔:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 2.2.2.2 router-
↪dead-interval 250
```

set protocols ospf4 area interface address transmit-delay

命令功能

set protocols ospf4 area interface address transmit-delay 命令用来配置接口上发送LSA过程中的传输延迟时间。

delete rotocols ospf4 area interface address transmit-delay 命令用来删除配置的LSA传输延迟时间，恢复为缺省值。

缺省情况下，LSA过程中的传输延迟时间为1秒。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **address** *vif-ip-address* **transmit-delay** *transmit-delay*

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**address** *vif-ip-address* [**transmit-delay**]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。必须为每个运行OSPF的接口指明所属的区域。

vif-ip-address：三层接口的IP地址。

transmit-delay：LSA过程中的传输延迟时间。整数形式，取值范围是1～3600，单位是秒。

命令模式

配置模式

使用指南

LSA在本设备的链路状态数据库（LSDB）中会随时间老化，但在网络的传输过程中却不会，所以有必要在发送之前在LSA的老化时间上增加本命令所设置的一段时间。此配置对低速率的网络尤其重要。

配置举例

配置接口vlan100上的LSA传输延迟时间为2秒:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 address 2.2.2.2 transmit-
↪delay 2
```

set protocols ospf4 area interface link-type

命令功能

set protocols ospf4 area interface link-type 命令用来配置OSPF接口的网络类型。

delete rotocols ospf4 area interface link-type 用来删除配置的OSPF接口网络类型，恢复为缺省值。

缺省情况下，接口的网络类型根据物理接口而定。以太网接口的网络类型为Broadcast，串口的网络类型为P2P。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **link-type** { **broadcast** | **p2m** | **p2p** }

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* [**link-type**]

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。

l3-interface-name：三层接口的名称，比如vlan100。

broadcast：将接口的网络类型修改为广播。

p2m：将接口的网络类型修改为点到多点。

p2p：将接口的网络类型修改为点到点。

命令模式

配置模式

使用指南

一般情况下，链路两端的OSPF接口的网络类型必须一致，否则不能正确的计算路由。 根据实际情况配置接口的网络类型，例如：

- 如果接口的网络类型是NBMA，但网络不是全连通的，必须将接口的网络类型改为P2M。这样，两台不能直接可达的交换机就可以通过一台与两者都直接可达的交换机来交换路由信息。
- 如果同一网段内只有两台路由器运行OSPF协议，建议将接口的网络类型改为P2MP。

当接口配置成NBMA类型，由于无法通过广播Hello报文的形式动态的发现相邻路由设备，必须手动为接口指定相邻接口的IP地址、是否有选举权等。

配置举例

设置接口vlan100的OSPF网络类型是P2MP:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 link-type p2m
```

set protocols ospf4 area interface vif

命令功能

set protocols ospf4 area interface address enable 命令用来配置OSPF的虚接口。

delete rotocols ospf4 area interface address enable 用来删除配置的虚接口。

缺省情况下，没有配置OSPF虚接口。

命令格式

set protocols ospf4 area *area-id* **interface** *l3-interface-name* **vif** *virtual-interface* [**address** *ip-address*]

delete protocols ospf4 area *area-id* **interface** *l3-interface-name* **vif** *virtual-interface*

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。
l3-interface-name：三层接口的名称，比如vlan100。
virtual-interface：OSPF的虚接口。

命令模式

配置模式

使用指南

无。

配置举例

设置虚接口:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 interface vlan100 vif vlan100.1 address 7.
↪7.7.
```

set protocols ospf4 area summaries enable

命令功能

set protocols ospf4 area summaries enable 命令用来配置是否使能向Stub区域发送聚合LSA功能。
delete protocols ospf4 area summaries enable 用来删除配置的向Stub区域发送聚合LSA功能。
缺省情况下，ABR会向Stub区域发送聚合LSA。

命令格式

set protocols ospf4 area *area-id* **summaries enable** { **false** | **true** }
delete protocols ospf4 area *area-id* **summaries enable**

参数说明

area-id：区域标识。IP地址形式，取值为点分十进制。
false：去使能向Stub区域发送聚合LSA功能。
true：使能向Stub区域发送聚合LSA功能。

命令模式

配置模式

使用指南

该命令需要在ABR上配置。

配置举例

使能向Stub区域发送聚合LSA功能:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 summaries enable true
```

set protocols ospf4 area virtual-link

命令功能

set protocols ospf4 area virtual-link 命令用来创建并配置虚连接。

delete protocols ospf4 area virtual-link 用来删除虚连接或恢复虚连接的参数为缺省值。

缺省情况下，OSPF没有创建虚连接。

命令格式

set protocols ospf4 area *area-id* **virtual-link** *ip-address* **authentication** { **md5** *key-id* | **simple-password** *password* } | **hello-interval** *hello-interval* | **retransmit-interval** *retransmit-interval* | **router-dead-interval** *router-dead-interval** | ***transmit-area** transmit-area-id* | **transmit-delay** *transmit-delay* }

delete protocols ospf4 area *area-id* **virtual-link** *ip-address* **authentication** { **md5** | **simple-password** } | **hello-interval** | **retransmit-interval** | **router-dead-interval** | **transmit-area** | **transmit-delay** }

参数说明

ip-address：指定建立虚连接的对端交换机IP地址。

key-id：MD5验证字标识符，必须与对端的标识符一致。整数形式，取值范围是0～255。

password：简单密码。

hello-interval：接口发送hello报文的时间间隔，该值必须与建立虚连接设备上的hello-interval值相同。整数形式，取值范围是1～65535，单位是秒。缺省值是10秒。

retransmit-interval：接口重传LSA的时间间隔。整数形式，取值范围是1～65535，单位是秒。缺省值是5秒。

router-dead-interval：失效时间间隔，该值必须与建立虚连接设备上的router-dead-interval值相同。取值范围1～4294967295。单位是秒。缺省值是40秒。

transmit-area-id：虚连接传输经过的区域ID。

transmit-delay：接口延迟发送LSA的时间间隔。整数形式，取值范围是1～3600，单位是秒。缺省值是1秒。

命令模式

配置模式

使用指南

在划分OSPF区域之后，非骨干区域之间的OSPF路由更新是通过骨干区域来交换完成的。因此，OSPF要求所有非骨干区域必须与骨干区域保持连通，并且骨干区域之间也要保持连通。但在实际应用中，因为各方面条件的限制，可能无法满足这个要求，这时可以通过配置OSPF虚连接解决。

配置参数值时有以下几点建议：

- hello参数值越小，交换机感知网络变化的速度越快，消耗的网络资源也会越多。
- retransmit参数值设置的太小会引起不必要的LSA重传，建议在网络速度较慢的网络中，该值可以设置得大一些。
- 虚连接的验证模式必须与骨干区域的验证方式一致。

配置举例

创建虚连接，对端设备ID为2.2.2.2:

```
ConnetOS# set protocols ospf4 area 1.1.1.1 virtual-link 2.2.2.2
```

set protocols ospf4 export

命令功能

set protocols ospf4 export 命令用来路由发布时的应用策略。

delete protocols ospf4 export 用来删除配置的路由发布策略。

缺省情况下，发布时没有应用路由策略。

命令格式

set protocols ospf4 export *export-policy*

delete protocols ospf4 export

参数说明

exort-policy：策略名称。

命令模式

配置模式

使用指南

无。

配置举例

设置路由发布应用策略:

```
ConnetOS# set protocols ospf4 export p1
```

set protocols ospf4 import

命令功能

set protocols ospf4 import 命令用来配置路由接收时的应用策略，控制引入的路由信息。
delete protocols ospf4 import 用来删除配置的路由策略。
缺省情况下，接收路由时没有应用发布策略。

命令格式

set protocols ospf4 import *import-policy*
delete protocols ospf4 import

参数说明

import-policy：路由策略。

命令模式

配置模式

使用指南

无。

配置举例

设置路由接收时的应用策略为p2:

```
ConnetOS# set protocols ospf4 import p2
```

set protocols ospf4 ip-router-alert enable

命令功能

set protocols ospf4 ip-router-alert enable 命令用来配置是否识别IP报文中携带的Router-Alert选项。
delete set protocols ospf4 ip-router-alert enable 用来删除设置的识别IP报文中携带的Router-Alert选项功能。

缺省情况下，设备不识别报文中携带的Router-Alert选项。

命令格式

set protocols ospf4 ip-router-alert enable { **false** | **true** }

delete protocols ospf4 ip-router-alert enable

参数说明

false：不识别IP报文中携带的Router-Alert选项。只有目的地址属于本设备的接口地址时，报文才会上送给路由协议层处理。

true：识别IP报文中携带的Router-Alert选项。带有Router-Alert选项的IP报文才会被上送到路由协议层处理。

命令模式

配置模式

使用指南

Router-Alert是一种标识协议报文的特殊机制。通常情况下，只有目的地址属于本设备的接口地址时，报文才会上送给路由协议层处理。如果一个报文中带有Router-alert选项，则表示该报文需要被上送到路由协议层去处理。

配置举例

设置设备识别IP报文中携带的Router-Alert选项:

```
ConnetOS# set protocols ospf4 ip-router-alert enable true
```

set protocols ospf4 rfc1583-compatibility enable

命令功能

set protocols ospf4 ip-router-alert enable 命令用来配置是否兼容RFC1583选路规则。

delete set protocols ospf4 ip-router-alert enable 用来删除配置的是否兼容RFC1583选路规则。

缺省情况下，不兼容RFC1583选路规则。

命令格式

set protocols ospf4 rfc1583-compatibility enable { **false** | **true** }

set protocols ospf4 rfc1583-compatibility enable

参数说明

false：不兼容RFC1583选路规则。

true：兼容RFC1583选路规则。

命令模式

配置模式

使用指南

为了避免路由环路，对于是否兼容RFC1583的选路规则，同一路由域内的交换机建议配置相同，即要么配置所有交换机都兼容RFC1583的选路规则，要么配置所有交换机都不兼容RFC1583的选路规则。

配置举例

设置ConnetOS兼容RFC1583的选路规则:

```
ConnetOS# set protocols ospf4 rfc1583-compatibility enable true
```

set protocols ospf4 router-id

命令功能

set protocols ospf4 router-id 命令用来配置运行OSPF协议设备的Router ID。

缺省情况下，Router ID为0.0.0.0。

命令格式

set protocols ospf4 router-id *route-id*

参数说明

route-id：是一个32比特无符号整数，是一台交换机在自治系统中的唯一标识。自治系统中任意两台Router ID都不能相同。

命令模式

配置模式

使用指南

通常将Router ID配置为与交换机某个接口的IP地址一致。

修改Router ID后必须重启系统或者在修改Router ID之前先删除所有OSPF配置。

配置举例

设置设备的Router ID为1.1.1.1:

```
ConnetOS# set protocols ospf4 router-id 1.1.1.1
```

show ospf4 database

命令功能

show ospf4 database 命令用来查看LSA数据库的信息。

命令格式

show ospf4 database [**area** *area-id*] [**asbrsummary** | **external** | **netsummary** | **network** | **nssa** | **router**] [**brief** | **detail**]

参数说明

area-id：OSPF区域ID。IP地址形式，取值为点分十进制。建议和某个接口的IP地址保持一致。0.0.0.0表示骨干区域。

*asbrsummary**：查看Summary-LSA的信息。此信息为AS边界路由器上发送的聚合LSA。

external：查看External-LSA的信息。

netsummary：查看Summary-LSA的信息。此信息为网络上的聚合LSA。

network：查看Network-LSA的信息。

nssa：查看NSSA-LSA的信息。

router ：查看Router-LSA的信息。

brief：查看概要信息。

detail：查看详细信息。

命令模式

运维模式

使用指南

如果不指定**brief**或**detail**，缺省情况下查看的是**brief**信息。

配置举例

查看设备上的LSA数据库信息:

```
ConnetOS> show ospf4 database
  OSPF link state database, Area 0.0.0.0
Type      ID                Adv Rtr           Seq         Age   Opt   Cksum   Len
-------   ---------------   ---------------   ---------   ----  ---   ------  ---
Router    *192.168.1.31     192.168.1.31      0x800001cd  439   0x2   0x9ccc  84
Router     192.168.1.22     192.168.1.22      0x80003992  434   0x2   0x7721  72
```

show ospf4 interface

命令功能

show ospf4 interface 命令用来查看OSPF接口的信息。

命令格式

show ospf4 interface [brief | detail]

参数说明

brief：查看概要信息。

detail：查看详细信息。

命令模式

运维模式

使用指南

无

配置举例

查看OSPF接口的信息:

```
ConnetOS> show ospf4 interface
Interface  State     Area              DR ID             BDR ID            Nbrs
---------  --------  ----------------  ----------------  ----------------  ----
vlan100    DR        0.0.0.0           192.168.1.31      0.0.0.0           0
vlan20     PtToPt    0.0.0.0           0.0.0.0           0.0.0.0           1
vlan30     PtToPt    0.0.0.0           0.0.0.0           0.0.0.0           1
```

show ospf4 neighbor

命令功能

show ospf4 neighbor 命令用来查看OSPF的邻居信息。

命令格式

show ospf4 neighbor [*neighbor-name* | **all**] [**brief** | **detail**]

参数说明

neighbor-name：查看指定邻居信息。
all：查看所有邻居信息。
brief：查看概要信息。
detail：查看详细信息。

命令模式

运维模式

使用指南

无

配置举例

查看OSPF的邻居信息:

```
ConnetOS> show ospf4 neighbor
Address          Interface              State     Router ID        Pri    Dead
---------------  ---------------------  --------  ---------------  -----  ----
22.22.22.20      vlan20/vlan20          Full      192.168.1.22     128    31
33.33.33.20      vlan30/vlan30          Full      192.168.1.22     128    30
```

clear ospf4 database

命令功能

clear ospf4 database 命令用来清除LSA数据库信息。

命令格式

clear ospf4 database

参数说明

无

命令模式

运维模式

使用指南

无

配置举例

清除LSA数据库信息:

```
ConnetOS> clear ospf4 database
```

BGP命令

set protocols bgp 4byte-as-numbers enable

命令功能

set protocols bgp 4byte-as-numbers enable 命令用来设置BGP使用4字节的AS编号。
delete protocols bgp 4byte-as-numbers enable 命令用来删除设置的AS编号，恢复为缺省值。
缺省情况下，不使用4字节的AS编号，使用2字节的AS编号。

命令格式

set protocols bgp 4byte-as-numbers enable { **false** | **true** }
delete protocols bgp 4byte-as-numbers enable

参数说明

false：不使用4字节的AS编号，即使用2字节的AS编号。
true：使用4字节的AS编号。

命令模式

配置模式

使用指南

无

配置举例

使用4字节的AS编号:

```
ConnetOS# set protocols bgp 4byte-as-numbers enable true
```

set protocols bgp bgp-id

命令功能

set protocols bgp bgp-id 命令用来设置BGP设备的Router ID。

delete protocols bgp bgp-id 命令用来删除配置的BGP设备的Router ID，恢复为缺省值。

缺省情况下，BGP设备没有设置Router ID。

命令格式

set protocols bgp bgp-id *bgp-id*

delete protocols bgp bgp-id

参数说明

bgp-id：本BGP设备的Router ID，必须是IPv4地址的形式。

命令模式

配置模式

使用指南

无

配置举例

设备本设备的Router ID为1.1.1.1:

```
ConnetOS# set protocols bgp bgp-id 1.1.1.1
```

set protocols bgp confederation enable

命令功能

set protocols bgp confederation enable 命令用来配置是否使能BGP联盟。

delete protocols bgp confederation enable 命令用来删除配置的BGP联盟，恢复为缺省值。

缺省情况下，BGP联盟是使能的。

命令格式

set protocols bgp confederation enable { **false** | **true** }

delete protocols bgp confederation enable

参数说明

false：不使能BGP联盟。

true：使能BGP联盟。

命令模式

配置模式

使用指南

无

配置举例

使能BGP联盟:

```
ConnetOS# set protocols bgp confederation enable true
```

set protocols bgp confederation identifier

命令功能

set protocols bgp confederation identifier 命令用来设置联盟ID。

delete protocols bgp confederation identifier 命令用来删除设置的联盟ID。

缺省情况下，没有设置联盟ID。

命令格式

set protocols bgp confederation identifier *confederation-id*

delete protocols bgp confederation identifier

参数说明

confederation-id：联盟ID。整数形式，取值范围1～4294967295。

命令模式

配置模式

使用指南

联盟ID相当于整个自治系统的编号，其他相关外部AS在指定对等体所在的AS号时，要指定这个联盟ID。属于同一个联盟的所有子自治系统都必须指定相同的联盟ID。

配置举例

设置联盟ID为90:

```
ConnetOS# set protocols bgp confederation identifier 90
```

set protocols bgp damping enable

命令功能

set protocols bgp damping enable 命令用来配置是否使能路由衰减功能。

delete protocols bgp damping enable 命令用来删除配置的路由衰减功能，恢复为缺省值。

缺省情况下，路由衰减功能是使能的。

命令格式

set protocols bgp damping enable { **false** | **true** }

delete protocols bgp damping enable

参数说明

false：不使能路由衰减功能。

true：使能路由衰减功能。

命令模式

配置模式

使用指南

无。

配置举例

使能路由衰减功能:

```
ConnetOS# set protocols bgp damping enable
```

set protocols bgp damping half-life

命令功能

set protocols bgp damping half-life 命令用来配置可达路由的半衰期。

delete protocols bgp damping half-life 命令用来取消配置的可达路由的半衰期，恢复为缺省值。

缺省情况下，可达路由的半衰期为15分钟。

命令格式

set protocols bgp damping half-life *half-life*

delete protocols bgp damping half-life

参数说明

half-life：可达路由的半衰期。整数形式，取值范围是1～4294967295。单位是分钟。

命令模式

配置模式

使用指南

无。

配置举例

配置路由衰减的半衰期为10分钟:

```
ConnetOS# set protocols bgp damping half-life 10
```

set protocols bgp damping max-suppress

命令功能

set protocols bgp damping max-suppress 命令用来配置路由衰减是的抑制时间。

delete protocols bgp damping max-suppress 命令用来删除配置的抑制时间，恢复为缺省值。

缺省情况下，抑制时间为60分钟。

命令格式

set protocols bgp damping max-suppress *max-suppress-time*

delete protocols bgp damping max-suppress

参数说明

max-suppress-time：抑制时间，即路由从被抑制到恢复可用的时间。整数形式，取值范围是1～4294967295。单位是分钟。

命令模式

配置模式

使用指南

无

配置举例

配置抑制时间为40分钟:

```
ConnetOS# set protocols bgp damping max-suppress 40
```

set protocols bgp damping reuse

命令功能

set protocols bgp damping reuse 命令用来设置路由衰减时路由解除抑制状态时的再使用阈值。

delete protocols bgp damping reuse 命令用来删除配置的再使用阈值，恢复为缺省值。

缺省情况下，再使用阈值是750。

命令格式

set protocols bgp damping reuse *reuse-value*

delete protocols bgp damping reuse

参数说明

reuse-value：指定路由解除抑制状态的阈值。当惩罚降低到该值以下，路由就被再使用。整数形式，取值范围是1～4294967295。

命令模式

配置模式

使用指南

无

配置举例

配置路由解除抑制状态时的再使用阈值为1000:

```
ConnetOS# set protocols bgp damping reuse 10000
```

set protocols bgp damping suppress

命令功能

set protocols bgp damping suppress 命令用来配置路由进入抑制状态的抑制阈值。
delete protocols bgp damping suppress 命令用来删除配置的抑制阈值，恢复为缺省值。
缺省情况下，抑制阈值是3000。

命令格式

set protocols bgp damping suppress *suppress-value*
delete protocols bgp damping suppress

参数说明

suppress-value：指定路由进入抑制状态的阈值。整数形式，取值范围是1～4294967295。

命令模式

配置模式

使用指南

当惩罚值超过抑制阈值时，路由受到抑制，不再加入到路由表中，也不再向其他BGP对等体发布更新报文。
抑制阈值必须比再使用阈值大。

配置举例

设置抑制阈值为5000:

```
ConnetOS# set protocols bgp damping suppress 50000
```

set protocols bgp local-as

命令功能

set protocols bgp local-as 命令过用来设置AS编号。

delete protocols bgp local-as 命令用来删除配置的AS编号。

缺省情况下，没有设置AS编号。

命令格式

set protocols bgp local-as *as-id*

delete protocols bgp local-as

参数说明

as-id：AS编号。整数形式，取值范围是1～4294967295。

命令模式

配置模式

使用指南

无。

配置举例

设置设备的AS编号为1:

```
ConnetOS# set protocols bgp local-as 1
```

set protocols bgp route-reflector enable

命令功能

set protocols bgp route-reflector enable 命令用来配置是否使能路由反射器。

delete protocols bgp route-reflector enable 命令用来删除配置的路由反射器功能，恢复为缺省值。

缺省情况下，路由反射器是使能的。

命令格式

set protocols bgp route-reflector enable { **false** | **true** }

delete protocols bgp route-reflector enable

参数说明

false：不使能路由反射器。

true：使能路由反射器。

命令模式

配置模式

使用指南

无

配置举例

使能路由反射器:

```
ConnetOS# set protocols bgp route-reflector enable true
```

set protocols bgp route-reflector cluster-id

命令功能

set protocols bgp route-reflector cluster-id 命令用来配置路由反射器的集群ID。

delete protocols bgp route-reflector cluster-id 命令用来删除配置的集群ID，恢复为缺省值。

缺省情况下，每个路由反射器使用自己的Router ID作为集群ID。

命令格式

set protocols bgp route-reflector cluster-id *cluster-id*

delete protocols bgp route-reflector cluster-id

参数说明

cluster-id：集群ID。IPv4地址形式。

命令模式

配置模式

使用指南

同一集群内所有的路由反射器配置相同的集群ID，以便标识这个集群，避免路由环路。

配置举例

设置集群ID为1.1.1.1:

```
ConnetOS# set protocols bgp route-reflector cluster-id 1.1.1.1
```

set protocols bgp peer as

命令功能

set protocols bgp peer as 命令用来配置指定BGP对等体的AS编号。

delete protocols bgp peer as 命令用来删除配置的BGP对等体的AS编号。

缺省情况下，没有配置BGP对等体的AS编号。

命令格式

set protocols bgp peer *peer-id* **as** *as-id*

delete protocols bgp peer *peer-id* **as**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

as-id：AS编号。整数形式，取值范围是1～4294967295。

命令模式

配置模式

使用指南

配置BGP对等体时，如果指定对等体所属的AS编号与本地AS编号相同，表示配置IBGP对等体。如果指定对等体所属的AS编号与本地AS编号不同，表示配置EBGP对等体。

配置举例

设置BGP对等体的AS编号为1:

```
ConnetOS# set protocols bgp peer 1.1.1.1 as 1
```

set protocols bgp peer client enable

命令功能

set protocols bgp peer client enable 命令用来配置是否将BGP对等体设置为路由反射器的客户。

delete protocols bgp peer client enable 命令用来取消配置的路由反射器客户。

缺省情况下，BGP对等体并不是路由反射器的客户。

命令格式

set protocols bgp peer *peer-id* **client enable** { **false** | **true** }

delete protocols bgp peer *peer-id* **client enable**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

false：不使能BGP对等体是路由反射器的客户。

true：使能BGP对等体是路由反射器的客户。

命令模式

配置模式

使用指南

在一个AS内，其中一台交换机作为路由反射器RR（Route Reflector），其它交换机做为客户机（Client）。客户机与路由反射器之间建立IBGP连接。

RR的优点在于配置方便，因为只需要在反射器上配置，客户机不需要知道自己是客户机。

配置举例

将BGP对等体1.1.1.1 设置为路由反射器的客户:

```
ConnetOS# set protocols bgp peer 1.1.1.1 client enable true
```

set protocols bgp peer confederation-member enable

命令功能

set protocols bgp peer confederation-member enable 命令用来配置是否将BGP对等体加入BGP联盟，即成为BGP联盟成员。

delete protocols bgp peer confederation-member enable 命令用来删除配置的加入BGP联盟。

缺省情况下，BGP对等体没有加入BGP联盟。

命令格式

set protocols bgp peer *peer-id* confederation-member enable** { **false** | **true** }

delete protocols bgp peer *peer-id* confederation-member enable**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

false：不使能BGP对等体加入BGP联盟。

true：使能BGP对等体加入BGP联盟。

命令模式

配置模式

使用指南

无

配置举例

将IP地址为1.1.1.1的BGP对等体加入BGP联盟:

```
ConnetOS# set protocols bgp peer 1.1.1.1 confederation-member enable true
```

set protocols bgp peer delay-open-time

命令功能

set protocols bgp peer delay-open-time 命令用来配置建立BGP对等体连接的延迟时间。

delete protocols bgp peer delay-open-time 命令用来删除配置的建立BGP对等体连接的延迟时间，恢复为缺省值。

缺省情况下，延迟时间是0秒，即马上建立BGP对等体连接。

命令格式

set protocols bgp peer *peer-id* **delay-open-time** *delay-open-time*
delete protocols bgp peer *peer-id* **delay-open-time**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。
delay-open-time：建立BGP对等体连接的延迟时间。整数形式，取值范围是0～4294967295，单位是秒。

命令模式

配置模式

使用指南

无。

配置举例

配置建立BGP对等体连接的延迟时间为120秒:

```
ConnetOS# set protocols bgp peer 1.1.1.1 delay-open-time 120
```

set protocols bgp peer enable

命令功能

set protocols bgp peer enable 命令用来配置是否将指定的交换机使能为BGP对等体。
delete protocols bgp peer enable 命令用来删除使能的BGP对等体。
缺省情况下，交换机已经使能为BGP对等体。

命令格式

set protocols bgp peer *peer-id* **enable** { **false** | **true** }
delete protocols bgp peer *peer-id* **enable**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。
false：不使能BGP对等体。
true：使能BGP对等体。

命令模式

配置模式

使用指南

无。

配置举例

使能BGP对等体:

```
ConnetOS# set protocols bgp peer 1.1.1.1 enable true
```

set protocols bgp peer holdtime

命令功能

set protocols bgp peer holdtime 命令用来配置建立BGP对等体时的空闲时间。

delete protocols bgp peer holdtime 命令用来删除配置的空闲时间，恢复为缺省值。

缺省情况下，建立BGP对等体时的空闲时间是90秒。

命令格式

set protocols bgp peer *peer-id* **holdtime** *holdtime*

delete protocols bgp peer *peer-id* **holdtime**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

holdtime：建立BGP对等体时的空闲时间。整数形式，取值范围是：0，3～65535。0表示不再等待。单位是秒。

命令模式

配置模式

使用指南

Idle状态是BGP初始状态。在Idle状态下，BGP拒绝邻居发送的连接请求。只有在收到本设备的Start事件后，BGP才开始尝试和其它BGP对等体进行TCP连接，并转至Connect状态。

配置举例

设置建立BGP对等体时的空闲时间:

```
ConnetOS# set protocols bgp peer 1.1.1.1 holdtime 60
```

set protocols bgp peer ipv4-multicast enable

命令功能

set protocols bgp peer ipv4-multicast enable 命令用来配置BGP对等体是否使能IPv4组播功能。
delete protocols bgp peer ipv4-multicast enable 命令用来删除配置的IPv4组播功能，恢复为缺省值。
缺省情况下，BGP对等体没有使能IPv4组播功能。

命令格式

set protocols bgp peer *peer-id* **ipv4-multicast enable** { **false** | **true** }
delete protocols bgp peer *peer-id* **ipv4-multicast enable**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。
false：BGP对等体不使能IPv4组播功能。
true：BGP对等体使能IPv4组播功能。

命令模式

配置模式

使用指南

无。

配置举例

BGP对等体使能IPv4组播功能:

```
ConnetOS# set protocols bgp peer 1.1.1.1 ipv4-multicast enable true
```

set protocols bgp peer ipv4-unicast enable

命令功能

set protocols bgp peer ipv4-unicast enable 命令用来配置BGP对等体是否使能IPv4单播功能。
delete protocols bgp peer ipv4-unicast enable 命令用来删除配置的IPv4单播功能，恢复为缺省值。
缺省情况下，BGP对等体的IPv4单播功能是使能的。

命令格式

set protocols bgp peer *peer-id* **ipv4-unicast enable** { **false** | **true** }
delete protocols bgp peer *peer-id* **ipv4-unicast enable**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。
false：BGP对等体不使能IPv4单播功能。
true：BGP对等体使能IPv4单播功能。

命令模式

配置模式

使用指南

无。

配置举例

BGP对等体使能IPv4单播功能:

```
ConnetOS# set protocols bgp peer 1.1.1.1 ipv4-unicast enable true
```

set protocols bgp peer local-ip

命令功能

set protocols bgp peer local-ip 命令用来配置BGP对等体在建立对等体连接时使用的IP地址。
delete protocols bgp peer local-ip 命令用来删除配置的BGP对等体IP地址。
缺省情况下，没有配置BGP对等体的IP地址。

命令格式

set protocols bgp peer *peer-id* **local-ip** *ip-address*

delete protocols bgp peer *peer-id* **local-ip**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

命令模式

配置模式

使用指南

无。

配置举例

配置BGP对等体的IP地址:

```
ConnetOS# set protocols bgp peer 1.1.1.1 local-ip 2.2.2.2
```

set protocols bgp peer local-port

命令功能

set protocols bgp peer local-port 命令用来配置本地BGP对等体进行TCP认证时端口号。

delete protocols bgp peer local-port 命令用来删除配置的本地TCP认证端口号。

缺省情况下，本地BGP对等体进行TCP认证时端口号是179。

命令格式

set protocols bgp peer *peer-id* **local-port** *local-port*

delete protocols bgp peer *peer-id* **local-port**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

local-port：本地BGP对等体进行TCP认证时端口号。整数形式，取值范围是1～4294967295。

命令模式

配置模式

使用指南

无。

配置举例

配置本地BGP对等体进行TCP认证时端口号:

```
ConnetOS# set protocols bgp peer 1.1.1.1 local-port 1010
```

set protocols bgp peer md5-password

命令功能

set protocols bgp peer md5-password 命令用来配置BGP对等体在建立TCP连接时对BGP消息进行MD5认证。

delete protocols bgp peer md5-password 命令用来删除配置的MD5认证，恢复为缺省值。

缺省情况下，对BGP消息不进行MD5认证。

命令格式

set protocols bgp peer *peer-id* **md5-password** *md5-password*

delete protocols bgp peer *peer-id* **md5-password**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

md5-password：进行MD5认证的密码。字符串形式。

命令模式

配置模式

使用指南

BGP使用TCP作为传输层协议，为提高BGP的安全性，可以在建立TCP连接时进行MD5认证。BGP的MD5认证只是为TCP连接设置MD5认证密码，由TCP完成认证。

如果MD5认证失败，则不建立TCP连接。

配置举例

配置BGP对等体在建立TCP连接时采用MD5认证:

```
ConnetOS# set protocols bgp peer 1.1.1.1 md5-password test
```

set protocols bgp peer next-hop

命令功能

set protocols bgp peer next-hop 命令配置指定BGP对等体的下一跳IP地址。
delete protocols bgp peer next-hop 命令用来删除配置的下一跳IP地址。
缺省情况下，BGP对等体没有配置下一跳IP地址。

命令格式

set protocols bgp peer *peer-id* **next-hop** *next-hop-address*
delete protocols bgp peer *peer-id* **next-hop**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。
next-hop-address：下一跳IP地址。点分十进制形式。

命令模式

配置模式

使用指南

无

配置举例

设置对等体的下一跳地址为1.1.1.2:

```
ConnetOS# set protocols bgp peer 1.1.1.1 next-hop 1.1.1.2
```

set protocols bgp peer peer-port

命令功能

set protocols bgp peer peer-port 命令用来配置远端设备进行TCP认证时端口号。

delete protocols bgp peer peer-port 命令用来删除配置的TCP认证端口号。

缺省情况下，进行TCP认证时端口号端口号是179。

命令格式

set protocols bgp peer *peer-id* **peer-port** *peer-port-number*

delete protocols bgp peer *peer-id* **peer-port**

参数说明

peer-id：BGP对等体的IP地址。点分十进制形式。

peer-port-number：端口号。整数形式，取值范围是1～4294967295。

命令模式

配置模式

使用指南

无。

配置举例

配置TCP认证端口号:

```
ConnetOS# set protocols bgp peer 1.1.1.1 peer-port 1010
```

路由策略命令

set policy policy-statement term from

命令功能

set policy policy-statement term from 命令用来定义应用于源地址路由的路由策略。

delete policy policy-statement term from 命令用来删除配置的定义应用于源地址路由的路由策略。

命令格式

set policy policy-statement *policy-name* **term** *term-name* **from** { **as-path** *as-path-name* | **as-path-list** *as-path-list-name* | **community** *community-name* | **community-list** *community-list-name* | **external-type** *type-number* | **localpref** *preference-number* | **med** *med* | **metric** *metric* | **neighbor** *neighbor-router* | **network4** *network4-address* | **network4-list** *network4-list-name* | **nexthop4** *nexthop4-address* | **origin** *origin-attribute* | **prefix-length4** *prefix-length4* | **protocol** { **connected** | **ospf4** | **static** } | **tag** *tag-value* }

delete policy policy-statement *policy-name* **term** *term-name* **from** { **as-path** | **as-path-list** *type-number* | **localpref** *preference-number* | **med** | **metric** | **neighbor** | **network4** | **network4-list** | **nexthop4** | **origin** | **prefix-length4** | **protocol** | **tag** }

参数说明

as-path-name：BGP路由所经过的所有AS名称。

as-path-list-name：BGP路由所经过的所有AS列表名称。

community-name：BGP团体属性名称。

community-list-name：BGP团体属性列表名称。

type-number：引入OSPF外部路由的类型。取值为1、2。

preference-number：本地路由的优先级，整数形式，取值范围是0～4294967295。

med：BGP MED属性值，整数形式，取值范围是0～4294967295。

metric：路由信息开销，整数形式，取值范围是0～4294967295。

neighbor-router：邻居路由的IP地址，取值为点分十进制。

network4-address：区域所包含的网段。取值形式是网段地址／掩码。

network4-list-name：网段列表的名称。

nexthop4-address：下一跳的IP地址，取值为点分十进制。

origin-attribute：路由的原始属性。整数形式，取值范围是0～2。

- 0：IGP
- 1：EGP
- 2：不完全路由

prefix-length4：地址前缀列表长度，整数形式，取值范围是0～4294967295。

connected：子网的直连路由。

ospf4：OSPFv4路由。

static：静态路由。

tag-value：OSPF路由信息的标记值。整数形式，取值范围是0～4294967295。

命令模式

配置模式

使用指南

无

配置举例

定义应用于源地址路由的路由策略policy1:

```
ConnetOS# set policy policy-statement policy1 term t1 from protocol ospf4
```

set policy policy-statement term then

命令功能

set policy policy-statement term then 用来设置对符合路由策略中策略内容定义的路由的处理方式。

delete policy policy-statement term then 命令用来删除定义的路由的处理方式。

命令格式

set policy policy-statement *policy-name* **term** *term-name* **then** { **accept** | **aggregate-brief-mode** { **false** | **true** } | **aggregate-prefix-len** *aggregate-prefix-len* | **as-path-expand** *as-path-expand* | **as-path-prepend** *as-path-prepend* | **community** *community-name* | **community-add** *community-add-name* | **community-del** *community-del* | **external-type** *external-type-number* | **localpref** *localpref* | **med** *med* | **med-remove** { **false** | **true** } | **metric** *metric* | **nexthop4** *nexthop4-address* | **nexthop4-var** { **peer-address** | **self**} | **origin** *origin-attribute* | **reject** | **tag** *tag-value* }

delete policy policy-statement *policy-name* **term** *term-name* **then** { **accept** | **aggregate-brief-mode** | **aggregate-prefix-len** | **as-path-expand** | **as-path-prepend** | **community** | **community-add** | **community-del** | **external-type** | **localpref** | **med** | **med-remove** | **metric** | **nexthop4** | **nexthop4-var** | **origin** | **reject** | **tag** }

参数说明

accept：接受符合路由策略及过滤条件的路由。

aggregate-brief-mode：不生成聚合路由。

aggregate-prefix-len *aggregate-prefix-len*：生成指定前缀长度的聚合路由。*aggregate-prefix-len* 为前缀长度，整数形式，取值范围是0～4294967295。

as-path-expand *as-path-expand*：在增加本地AS域时的预留AS编号优先级。整数形式，取值范围是0～4294967295。

as-path-prepend *as-path-prepend*：AS路径的预留AS编号优先级。整数形式，取值范围是0～4294967295。

community-name：BGP团体的名称。

community-add-name：在路由中引入的BGP团体的名称。

community-del：在路由中删除的BGP团体的名称。

external-type-number：OSPF的外部路由类型。取值为1、2，分别代表type1和type2类型的外部OSPF路由。

localpref：本地优先级。整数形式，取值范围是0～4294967295。

med：BGP MED属性值，整数形式，取值范围是0～4294967295。

med-remove：是否移除MED属性。

metric：路由信息开销。整数形式，取值范围是0～4294967295。

nexthop4-address：下一跳的IP地址，取值为点分十进制。

peer-address：对端IP地址。

self：

origin-attribute：路由的原始属性。整数形式，取值范围是0～2。

- 0：IGP
- 1：EGP
- 2：不完全路由

reject：拒绝接受符合路由策略及过滤条件的路由。

tag-value：OSPF路由信息的标记值。整数形式，取值范围是0～4294967295。

命令模式

配置模式

使用指南

无。

配置举例

接受符合policy1中term1定义的源路由:

```
ConnetOS# set policy policy-statement policy1 term term1 then accept
```

set policy policy-statement term to

命令功能

set policy policy-statement term to 命令用来定义应用于目的地址路由的路由策略。

set policy policy-statement term to 命令用来删除定义的应用于目的地址路由的路由策略。

命令格式

set policy policy-statement *policy-name* **term** *term-name* **to** { **as-path** *as-path-name* | **as-path-list** *as-path-list-name* | **community** *community-name* | **community-list** *community-list-name* | **external-type** *type-number* | **localpref** *preference-number* | **med** *med* | **metric** *metric* | **neighbor** *neighbor-router* | **network4** *network4-address* | **network4-list** *network4-list-name* | **nexthop4** *nexthop4-address* | **origin** *origin-attribute* | **prefix-length4** *prefix-length4* | **protocol** { **connected** | **ospf4** | **static** } | **tag** *tag-value* | **was-aggregated** { **false** | **was-aggregated** } }

delete policy policy-statement *policy-name* **term** *term-name* **to** { **as-path** | **as-path-list** *type-number* | **localpref** *preference-number* | **med** | **metric** | **neighbor** | **network4** | **network4-list** | **nexthop4** | **origin** | **prefix-length4** | **protocol** | **tag** | **was-aggregated** }

参数说明

as-path-name：BGP路由所经过的所有AS名称。

as-path-list-name：BGP路由所经过的所有AS列表名称。

community-name：BGP团体属性名称。

community-list-name：BGP团体属性列表名称。

type-number：引入OSPF外部路由的类型。取值为1、2。

preference-number：路由协议的优先级，整数形式，取值范围是0～4294967295。

med：BGP MED属性值，整数形式，取值范围是0～4294967295。

metric：路由信息的路由开销。整数形式，取值范围是0～4294967295。

neighbor-router：邻居路由的IP地址，取值为点分十进制。

network4-address：区域所包含的网段。取值形式是网段地址／掩码。

network4-list-name：网段列表的名称。

nexthop4-address：下一跳的IP地址，取值为点分十进制。

origin-attribute：路由的原始属性。整数形式，取值范围是0～2。

- 0：IGP
- 1：EGP
- 2：不完全路由

prefix-length4：地址前缀列表长度，整数形式，取值范围是0～4294967295。

connected：子网的直连路由。

ospf4：OSPFv4路由。

static：静态路由。

tag-value：OSPF路由信息的标记值。整数形式，取值范围是0～4294967295。

命令模式

配置模式

使用指南

无

配置举例

定义应用于目的地址路由的路由策略policy1:

```
ConnetOS# set policy policy-statement policy1 term term2 to nexthop4 1.1.1.1
```

set policy policy-statement then

命令功能

set policy policy-statement then 命令用来设置匹配路由策略后的执行动作。
delete policy policy-statement then 命令用来删除配置的执行动作。

命令格式

set policy policy-statement *policy-name* **then** { **accept** | **reject**}
delete policy policy-statement *policy-name* **then** [**accept** | **reject**]

参数说明

accept：接受符合 *policy-name* 定义的路由策略的路由。
reject：拒绝符合 *policy-name* 定义的路由策略的路由。

命令模式

配置模式

使用指南

无

配置举例

接受policy1定义的路由:

```
ConnetOS# set policy policy-statement policy1 then accept
```

流量策略命令

QoS命令

set class-of-service

命令功能

set class-of-service 命令用来QoS服务等级节点。
缺省情况下，没有设置QoS服务等级节点。

命令格式

set class-of-service

参数说明

无

命令模式

配置模式

使用指南

无。

配置举例

设置QoS服务等级节点:

```
ConnetOS# set class-of-service
```

set class-of-service classifier

命令功能

set class-of-service classifier 命令用来定义流分类。

delete class-of-service classifier 命令用来删除指定的流分类及流分类的相关配置。 缺省情况下，ConnetOS上没有定义流分类。

命令格式

set class-of-service classifier *classifier-name*

delete class-of-service classifier *classifier-name*

参数说明

classifier-name：流分类的名字。字符串形式，支持区分大小写及特殊字符，如果首字母是空格，能配置但是空格在名称中不体现。

命令模式

配置模式

使用指南

ConnetOS上可以定义任意个数的流分类，但是不提交流分类就不生效，并不消耗多少系统资源。

Note:

- 流分类必须配置优先级信任模式（trust-mode）后才能提交。
- 如果流分类已经被绑定，必须先去除绑定后才能删除。

配置举例

创建一个名字为c1的类:

```
ConnetOS# set class-of-service classifier c1
```

set class-of-service classifier forwarding-class

命令功能

set class-of-service classifier forwarding-class 命令用来设置指定流分类在出端口进行报文转发时的转发队列。

delete class-of-service classifier forwarding-class 命令用来删除流分类和转发队列的绑定。

缺省情况下，如果不指定转发队列，那报文进0队列进行转发。

命令格式

set class-of-service classifier *classifier-name* **forwarding-class** *forwarding-class* [**code-point** *code-poin*]

delete class-of-service classifier *classifier-name* **forwarding-class** *forwarding-class* [**code-point** *code-point*]

参数说明

classifier-name：流分类的名字。字符串形式，支持区分大小写及特殊字符，如果首字母是空格，能配置但是空格在名字中不体现。

forwarding-class：转发队列的名称。字符串形式，支持区分大小写及特殊字符，如果首字母是空格，能配置但是空格在名字中不体现。

code-point：ieee-802.1和inet-precedence的取值范围是0～7，dscp的取值范围是0～63。

命令模式

配置模式

使用指南

在入端口上为流分类绑定转发队列时，决定了报文发送时走哪个队列。 配置此命令后，流分类将进入不同的转发队列。

配置举例

设置流分类c1中的DSCP流量，在转发时按照转发队列fd1的队列编号进行转发:

```
ConnetOS# set class-of-service classifier c1 forwarding-class fd1 code-point 14
```

set class-of-service classifier trust-mode

命令功能

set class-of-service classifier trust-mode 命令用来配置优先级信任模式，即设备根据哪种优先级进行映射。

delete class-of-service classifier trust-mode 用来删除配置的优先级信任模式。

命令格式

set class-of-service classifier *classifier-name* **trust-mode** { **dscp** | **ieee-802.1** | **trust-port** }

delete class-of-service classifier *classifier-name* **trust-mode** { **dscp** | **ieee-802.1** | **trust-port** }

参数说明

dscp：指定对报文按照DSCP优先级进行映射。

ieee-802.1：指定对报文按照802.1p优先级进行映射。

trust-port：指定对报文按照端口信任模式进行映射。

命令模式

配置模式

使用指南

无。

配置举例

设置流分类c1中的IP报文按照DSCP优先级优先级进行映射:

```
ConnetOS# set class-of-service classifier c1 trust-mode dscp
```

set class-of-service forwarding-class

命令功能

set class-of-service forwarding-class 命令用来设置出端口的转发队列。

delete class-of-service forwarding-class 命令用来删除创建的转发队列。

缺省情况下，ConnetOS上没有创建转发队列。

命令格式

set class-of-service forwarding-class *forwarding-class* **queue-num** *queue-numer*

delete class-of-service forwarding-class *forwarding-class* [**queue-num**]

参数说明

forwarding-class：转发队列的名称。字符串形式，支持区分大小写及特殊字符，如果首字母是空格，能配置但是空格在名字中不体现。

queue-numer：队列编号。整数形式，取值范围是0～7。数字越大，优先级越高。

命令模式

配置模式

使用指南

ConnetOS按照配置的队列优先级在出端口对报文进行转发，不同的转发队列，将获得不同的服务等级。

配置举例

设置转发队列fd1的队列编号为1:

```
ConnetOS# set class-of-service forwarding-class fd1 queue-num 1
```

set class-of-service interface

命令功能

set class-of-service interface 命令用来设置将指定的流分类绑定到指定接口。

delete class-of-service interface 命令用来删除接口上绑定的流分类。

缺省情况下，接口上没有绑定任何流分类。

命令格式

set class-of-service interface *interface-name* [**classifier** *classifier-name*]
delete class-of-service interface *interface-name* [**classifier**]

参数说明

interface-name：接口名称。
classifier-name：流分类名称。此流分类必须是ConnetOS上已经存在的流分类。

命令模式

配置模式

使用指南

将流分类绑定到流量的入口后，流量在出口会按照优先级到队列映射表映射到相应的出口队列。

配置举例

将流分类c1绑定到接口te－1/1/15:

```
ConnetOS# set class-of-service interface te-1/1/15 classifier c1
```

set interface gigabit-ethernet cos

命令功能

set interface gigabit-ethernet cos 命令用来在报文映射模式为信任端口时，配置端口优先级。
delete interface gigabit-ethernet cos 命令用来删除端口优先级。

命令格式

set interface gigabit-ethernet *interface-number* **cos priority** *priority-value*
delete interface gigabit-ethernet *interface-number* **cos priority**

参数说明

interface-number：接口编号。
priority-value：优先级。整数形式，取值范围是0～7。

命令模式

配置模式

使用指南

配置端口优先级后，此端口流入的流量将以端口优先级查找优先级映射表得到出口队列。

配置举例

设置接口te－1/1/15的端口优先级为3:

```
ConnetOS# set class-of-service interface gigabit-ethernet te-1/1/15 cos priority 3
```

set interface gigabit-ethernet cos schedule mode

命令功能

set interface gigabit-ethernet cos schedule mode 命令用来设置指定接口在拥塞管理时的调度算法。
delete interface gigabit-ethernet cos schedule mode 命令用来删除指定接口的拥塞管理调度算法。
缺省情况下，接口采用WDRR调度算法。

命令格式

set interface gigabit-ethernet *interface-number* **cos schedule mode** { **sp** | **sp+wdrr** | **sp+wrr** | **wdrr** | **wrr** }
delete interface gigabit-ethernet *interface-number* **cos schedule mode**

参数说明

interface-number：接口编号。
sp：严格优先级调度。
sp+wrr：SP和WRR结合的调度算法
wdrr：带赤字的加权轮询调度。
wrr：加权轮询调度。

命令模式

配置模式

使用指南

WRR和WDRR模式需要配置全部8个队列的权重。

WRR+SP和WDRR+SP模式下没有配置权重的队列按照SP调度，SP模式无需配置队列权重。

配置举例

设置接口gigabit-ethernet te-1/1/16采用sp+wdrr的调度算法:

```
ConnetOS# set interface gigabit-ethernet te-1/1/16 cos schedule mode sp+wdrr
```

set interface gigabit-ethernet cos schedule queue

命令功能

set interface gigabit-ethernet cos schedule queue 命令用来设置在拥塞管理时指定接口进入的调度队列编号和加权值。

delete interface gigabit-ethernet cos schedule 命令用来删除指定端口的队列编号和加权值。

缺省情况下，端口上不同报文入指定编号的队列。

命令格式

set interface gigabit-ethernet *interface-number* **cos schedule queue** *queue-number* [**weight** *weight-value*]

delete interface gigabit-ethernet *interface-number* **cos schedule queue** *queue-number* [**weight**]

参数说明

interface-number：接口编号。

queue-number：队列编号。整数形式，取值范围是0～7。

weight-value：WRR和WDRR的权重值。整数形式，取值范围是0～127。0属于SP队列。

命令模式

配置模式

使用指南

缺省的队列映射如下：

入口报文携带dscp	入口报文携带8021.p	出口队列
0～7	0	0
8～15	1	1
16～23	2	2
24～31	3	3
32～39	4	4
40～47	5	5
48～55	6	6
58～63	7	7

配置举例

设置接口gigabit-ethernet te-1/1/16在发生拥塞时进入2号队列:

```
ConnetOS# set interface gigabit-ethernet te-1/1/17 cos schedule queue 2 weight 34
```

Firewall命令

set firewall term

命令功能

set firewall term 命令用来设置过滤规则组的各项过滤规则。

delete firewall term 命令用来删除设置的过滤规则组。

缺省情况下，设备上没有设置好的过滤规则组。

命令格式

set firewall term *term-name* [**cos value** *priority-value* | **dest-ipv4 network** *ipv4-network-address* | **dest-mac hwaddr** *mac-address* | **dscp value** *dscp-value* | **ether-type** { **name** { **arp** | **ipv4** | **rarp** } | **number** *ether-type-number* } | **l4-dest-port** { **name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *dest-port-number* | **range** *port-range* } | **l4- source-port** { **name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *source-number* | **range** *port-range* } | **protocol** { **name** { **ah** | **dstopts** | **egp** | **esp** | **fragment** | **gre** | **hop-by-hop** | **icmp** | **igmp** | **ipip** | **no-next-header** | **ospf** | **pim** | **routing** | **rsvp** | **sctp** | **tcp** | **udp** } | **number** *protocol-number* } | **source-ipv4 network** *ipv4-network-address* | **source-mac hwaddr** *mac-address* | **vlan number** *vlan-id*]

delete firewall term *term-name* [**cos value** *priority-value* | **dest-ipv4 network** *ipv4-network-address* | **dest-mac hwaddr** *mac-address* | **dscp value** *dscp-value* | **ether-type** [**name** { **arp** | **ipv4** | **rarp** } | **number** *ether-type-number*] | **l4-dest-port** [**name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *dest-port-number* | **range** *port-range*] | **l4- source-port** [**name** { **bgp** | **bootpc** | **bootps** | **dns** | **finger** | **ftp** | **ftp-data** | **http** | **https** | **msdp** | **ntp** | **oob-ws-http** | **pop3** | **radius** | **rip** | **smtp** | **snmp** | **telnet** | **tftp** } | **number** *source-number* | **range** *port-range*] | **protocol** [**name** { **ah** | **dstopts** | **egp** | **esp** | **fragment** | **gre** | **hop-by-hop** | **icmp** | **igmp** | **ipip** | **no-next-header** | **ospf** | **pim** | **routing** | **rsvp** | **sctp** | **tcp** | **udp** } | **number** *protocol-number*] | **source-ipv4 network** *ipv4-network-address* | **source-mac hwaddr** *mac-address* | **vlan number** *vlan-id*]

参数说明

term-name：过滤规则组名称。

priority-value：CoS优先级。整数形式，取值范围是0～7。

ipv4-network-address：目的IPv4地址，IPv4地址/掩码的形式。

mac-address：目的MAC地址。

dscp-value：DSCP值。整数形式，取值范围是0～63。

ether-type-number：整数形式，取值范围是0～65535。

dest-port-number：目的端口编号。整数形式，取值范围是0～65535。

port-range：目的端口范围，取值形式为1:10。

protocol-number：协议编号。整数形式，取值范围是0～255。

vlan-id：VLAN ID，整数形式，取值范围是1～4094。

命令模式

配置模式

使用指南

配置过滤规则组的各项功能前，需要先创建过滤规则组。即必须先执行 **set firewall term** *term-name* 后，才能进行规则的设置。

只有配置了的过滤规则组，执行 **delete** 命令时才会显示。

在同一个term下，可以同时配置多种过滤规则。报文在进行规则匹配时，只要命中了一个过滤规则就不再进行匹配。

配置举例

设置过滤规则组term1，用于过滤VLAN100的报文:

```
ConnetOS# set firewall term term1 vlan number 100
```

set firewall filter input/output interface

命令功能

set firewall filter input/output interface 命令用来将创建好的过滤策略应用到接口上。

delete firewall filter input/output interface 命令用来删除接口上应用的过滤策略。

缺省情况下，接口上没有应用任何过滤策略。

命令格式

set firewall filter *filter-name* { **input | output** } { **gigabit-ethernet | vlan-interface** } *interface-name*

delete firewall filter *filter-name* [**input | output**] [[**gigabit-ethernet | vlan-interface**] *interface-name*]

参数说明

filter-name：过滤策略名称。

input：将过滤策略应用在接口的入方向。

output：将过滤策略应用在接口的出方向。

interface-name：应用过滤规则的接口名称。

命令模式

配置模式

使用指南

同一个过滤策略可以应用在不同接口的入方向或者出方向，但是不能配置在同一个接口的入方向和出方向。

删除过滤规则时，设备上只显示已经配置的过滤内容。

配置举例

将过滤规则应用在接口te-1/1/1的入方向上:

```
ConnetOS# set firewall filter filter-dest input gigabit-ethernet te-1/1/1
```

set firewall filter matching term action

命令功能

set firewall filter matching term action 命令用来创建过滤策略，对符合过滤规则组的报文采取指定的动作。

delete firewall filter matching term action 命令用来删除建立的过滤策略。

缺省情况下，设备上没有建立过滤策略。

命令格式

set firewall filter *filter-name* **matching term** *term-name* [**action** { **discard | forward** }]

delete firewall filter *filter-name* [**matching term** *term-name* [**action**]]

参数说明

filter-name：过滤策略名称。
term-name：过滤规则组名称。
discard：丢弃符合过滤规则组的报文。
forward：转发符合过滤规则组的报文。

命令模式

配置模式

使用指南

过滤规则组需要先配置，才能关联成功。一个过滤策略，可以关联多个过滤规则组。

配置举例

对符合过滤规则组term1的报文进行丢弃:

```
ConnetOS# set firewall filter f1 matching term t1 action discard
```

set firewall forwarder

命令功能

set firewall forwarder 命令用来创建转发策略。
delete firewall forwarder 命令用来删除已经配置的转发策略。
缺省情况下，没有设置好的转发策略。

命令格式

set firewall forwarder *forwarder-name*
delete firewall forwarder *forwarder-name*

参数说明

forwarder-name：转发策略名称。

命令模式

配置模式

使用指南

配置转发策略的各项功能前，需要先创建转发策略。

配置举例

设置转发策略fd1:

```
ConnetOS# set firewall forwarder fd1 action classifying new-cos 5
```

set firewall forwarder action classifying

命令功能

set firewall forwarder action classifying 命令用来设置转发策略的报文分类动作，对符合过滤规则的报文进行分类。

delete firewall forwarder action classifying 命令用来删除转发策略的报文分类动作。

缺省情况下，转发策略的报文分类动作没有设置。

命令格式

set firewall forwarder *forwarder-name* **action classifying** { **new-cos** *cos-modify-value* | **new-dscp** *dscp-modify-value* }

delete firewall forwarde *forwarder-name* **action classifying** [**new-cos** | **new-dscp**]

参数说明

forwarder-name：转发策略名称。

cos-modify-value：CoS的修改值，用于修改报文优先级映射到设备优先级的值。

dscp-modify-value：DSCP的修改值。

命令模式

配置模式

使用指南

无。

配置举例

设置对符合过滤规则组的报文进行转发时，修改cos值为5:

```
ConnetOS# set firewall forwarder fd1 action classifying new-cos 5
```

set firewall forwarder action mirroring

命令功能

set firewall forwarder action mirroring 命令用来设置转发策略的镜像动作，用于将符合指定过滤规则组的报文镜像到指定端口。

delete firewall forwarder action mirroring 命令用来删除转发策略的镜像动作。 缺省情况下，转发策略的镜像动作没有设置。

命令格式

set firewall forwarder *forwarder-name* **action mirroring interface gigabit-ethernet** *ge-interface-name*

delete firewall forwarder *forwarder-name* **action** [**mirroring** [**interface** [**gigabit-ethernet**]]]

参数说明

forwarder-name：转发策略名称。

ge-interface-name：GE接口名称。

命令模式

配置模式

使用指南

无。

配置举例

将符合过滤规则组的报文镜像到接口te-1/1/1上:

```
ConnetOS# set firewall forwarder fd1 action mirroring interface gigabit-ethernet te-1/
↪1/1
```

set firewall forwarder action routing

命令功能

set firewall forwarder action routing 命令用来设置转发策略的三层转发行为，用于将符合指定过滤规则组的报文进行三层转发。

delete firewall forwarder action routing 命令用来删除转发策略的三层转发。

缺省情况下，转发策略的三层转发行为没有设置。

命令格式

set firewall forwarder *forwarder-name* **action routing** { **mode** { **load-balance** | **redundancy** } | **nexthopv4 address** *ipv4-address* | **vlan-interface** *vlan-interfac* }

delete firewall forwarder *forwarder-name* **action routing** { **mode** | **nexthopv4 address** *ipv4-address* | **vlan-interface** *vlan-interface* }

参数说明

forwarder-name：转发策略名称。

mode：设置三层转发的模式。

load-balance：进行负载分担转发，从多条活动链路转发。

redundancy：进行冗余备份转发，只从一条活动链路转发。

ipv4-address：目的IPv4地址，点分十进制格式。

vlan-interface：VLAN接口编号。

命令模式

配置模式

使用指南

无

配置举例

对符合过滤规则组的报文，转发时进行负载分担转发:

```
ConnetOS# set firewall forwarder fd1 action routing mode load-balance
```

set firewall forwarder action switching

命令功能

set firewall forwarder action switching 命令用来设置转发策略的二层转发行为，用于将符合指定过滤规则组的报文按照指定接口进行转发。

delete firewall forwarder action switching 命令用来删除设置的转发策略的二层转发行为。

缺省情况下，转发策略的二层转发行为没有设置。

命令格式

set firewall forwarder *forwarder-name* **action switching interface** { **aggregate-ethernet** *ae-interface-name* | **gigabit-ethernet** *ge-interface-name* }

delete firewall forwarder *forwarder-name* **action** [**switching** [**interface** [**aggregate-ethernet** | **gigabit-ethernet**]]]

参数说明

forwarder-name：转发策略名称。

ae-interface-name：汇聚组接口名称。

ge-interface-name：GE接口名称。

命令模式

配置模式

使用指南

此命令为覆盖式命令，即最后一次配置的接口生效。同时，同一个转发策略，只能指定一个接口进行转发，汇聚组接口和GE接口不能同时配置。

配置举例

对符合过滤规则组的报文，进行二层转发时从接口te-1/1/10转进行转发:

```
ConnetOS# set firewall forwarder fd1 action switching interface gigabit-ethernet te-1/
↪1/10
```

set firewall forwarder input

命令功能

set firewall forwarder input 命令用来将转发策略应用到接口上。

delete firewall forwarder input 命令用来删除接口上应用的转发策略。

缺省情况下，接口上没有应用转发策略。

命令格式

set firewall forwarder *forwarder-name* **input** { **gigabit-ethernet** *ge-interface-name* | **vlan-interface** *vlan-interface-name* }

delete firewall forwarder *forwarder-name* **input** [**gigabit-ethernet** *interface-name* | **vlan-interface** vlan-interface-name]

参数说明

forwarder-name：转发策略名称。

ge-interface-name：GE接口编号。

vlan-interface-name：VLAN接口编号。

命令模式

配置模式

使用指南

无

配置举例

在接口te-1/1/1上应用转发策略:

```
ConnetOS# ConnetOS# set firewall forwarder fd1 input gigabit-ethernet te-1/1/1
```

set firewall forwarder match-mode

命令功能

set firewall forwarder match-mode 命令用来设置进行转发的报文的匹配模式。

delete firewall forwarder match-mode 命令用来删除配置匹配模式。

缺省情况下，对符合过滤规则的报文进行转发。

命令格式

set firewall forwarder *forwarder-name* **match-mode** { **matched** | **unmatched** }

delete firewall forwarder *forwarder-name* [**match-mode**]

参数说明

forwarder-name：转发策略名称。
matched：对符合过滤规则的报文进行策略转发。
unmatched：对不符合过滤规则的报文进行转发。

命令模式

配置模式

使用指南

无

配置举例

对不符合过滤规则的报文进行重定向:

```
ConnetOS# set firewall forwarder fd1 match-mode unmatched
```

set firewall forwarder matching term

命令功能

set firewall forwarder matching term 命令用来将转发策略和过滤规则组关联。
delete firewall forwarder matching term 命令用来删除转发策略关联的过滤规则组。
缺省情况下，转发策略没有绑定过滤规则组。

命令格式

set firewall forwarder *forwarder-name* **matching term** *term-name*
delete firewall forwarder *forwarder-name* **matching** [**term** *term-name*]

参数说明

forwarder-name：转发策略名称。
term-name：过滤规则组名称。

命令模式

配置模式

使用指南

无

配置举例

将重定向策略fd1和过滤规则组term1进行关联:

```
ConnetOS# set firewall forwarder fd1 matching term term1
```

show firewall forwarder

命令功能

show firewall forwarder 命令用来查看转发策略的信息。

命令格式

show firewall forwarder [*forwarder-name*]

参数说明

forwarder-name：转发策略名称。

命令模式

运维模式

使用指南

必须先进行转发策略的设置，才能用show查看到相关配置。

配置举例

查看转发策略fd1的配置信息:

```
ConnetOS 1> show firewall forwarder
Total ingress HW Entries    :   5120
Consumed ingress HW Entries :   1
Total egress HW Entries     :   1280
Consumed egress HW Entries  :   0
======================== Policy Based Forwarding ====================
Forwarder : f1
```

show firewall filter

命令功能

show firewall filter 命令用来查看过滤信息。

命令格式

show firewall filter [*filter-name*]

参数说明

filter-name：已经配置的过滤策略名称。

命令模式

运维模式

使用指南

必须先配置过滤策略，才能用show查看。

配置举例

查看设备的过滤情况:

```
ConnetOS 1> show firewall filter
Total ingress HW Entries    :   5120
Consumed ingress HW Entries :   1
Total egress HW Entries     :   1280
Consumed egress HW Entries  :   0
======================== Policy Based Filtering ====================
Filter : f1
    Term : t1
        Term-size       : 1
        Action          : Forward
        Matched packets : 0
        Match-condition :
            dscp :                    value 2
    Input interface     : te-1/1/10
```

clear firewall

命令功能

clear firewall 命令用来清除firewall的统计计数。

命令格式

clear firewall [**filter** *filter-name* | **forwarder** *forwarder-name*]

参数说明

filter-name：过滤策略名称。
forwarder-name：转发策略名称。

命令模式

运维模式

使用指南

无

配置举例

清除过滤策略f1的统计计数:

```
ConnetOS> clear firewall filter f1
```

网络监控和诊断命令

镜像命令

set analyzer instance input

命令功能

set analyzer instance input 命令用来配置镜像端口。
delete analyzer instance input 命令用来删除配置的镜像端口。
缺省情况下，没有配置镜像端口。

命令格式

set analyzer instance instance-name **input** { **egress** | **ingress** } *interface-name*
delete analyzer instance instance-name **input** { **egress** | **ingress** } *interface-name*

参数说明

instance-name：镜像实例的名称。
egress：对出方向的流量进行镜像。
ingress：对入方向的流量进行镜像。
interface-name：接口名称。此接口是用于做镜像的接口。

命令模式

配置模式

使用指南

如果要对接口双向的流量进行镜像，分别配置出、入方向的端口镜像即可。

配置举例

对接口te-1/1/1上ingress方向的报文进行镜像:

```
ConnetOS#  set analyzer instance mirror1 input ingress te-1/1/1
```

set analyzer instance output

命令功能

set analyzer instance output 命令用来配置镜像时的观测端口。
delete analyzer instance output 命令用来删除配置的镜像观测端口。
缺省情况下，没有配置观测端口。

命令格式

set analyzer instance *instance-name* **output** *interface-name*
delete analyzer instance *instance-name* **output**

参数说明

instance-name：镜像实例的名称。
interface-name：接口名称。此接口是将镜像的报文复制到的接口。

命令模式

配置模式

使用指南

无

配置举例

设置接口te-1/1/2为观测端口:

```
ConnetOS# set analyzer instance mirror1 output te-1/1/2
```

show analyzer

命令功能

show analyzer 命令用来查看端口镜像的配置信息。

命令格式

show analyzer [*instance-name*]

参数说明

instance-name：镜像实例的名称。

命令模式

运维模式

使用指南

如果指定 *instance-name* ， 查看的是指定镜像的配置信息；否则查看的是设备上所有镜像的配置信息。

配置举例

查看设备上所有镜像的配置信息:

```
ConnetOS> show analyzer
Analyzer name: mirror1
Output interface: <te-1/1/2>
Ingress monitored interfaces: <te-1/1/1>
Egress monitored interfaces:
```

sFlow命令

clear sflow statistics

命令功能

clear sflow statistics 命令用来清除设备上的sFlow的统计信息。

命令格式

clear sflow statistics

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

清除sFlow的统计信息:

```
ConnetOS> clear sflow statistics
```

set protocols sflow collector

命令功能

set protocols sflow collector 命令用来指定sFlow Collector
delete protocols sflow collector 命令用来删除指定的sFlow Collector。
缺省情况下，没有指定sFlow Collector。

命令格式

set protocols sflow collector { **address** *ip-address* | **port** *port-number* }
delete protocols sflow collector { **address** | **port** }

参数说明

ip-address：sFlow Collector的IP地址。只支持配置一个sFlow Collector。
port-number：sFlow Collector使用的UDP端口号。整数形式，取值范围是1～65535。
缺省情况下，UDP端口号为6343。

命令模式

配置模式

使用指南

只有配置了sFlow Collector之后，sFlow才开始进行采样。

配置举例

指定地址为1.1.1.1的sFlow Collector作为采样输出的Collector:

```
ConnetOS# set protocols sflow collector 1.1.1.1 udp-port 6500
```

set protocols sflow header-length

命令功能

set protocols sflow header-length 命令用来配置全局的sFlow采样报文长度。
delete protocols sflow header-length 命令用来删除配置的全局采样报文长度，恢复为缺省值。
缺省情况下，采样报文长度是128字节。

命令格式

set protocols sflow header-length *header-length*
delete protocols sflow header-length

参数说明

header-length：采样报文长度。整数形式，取值范围是64～512，单位是字节。

命令模式

配置模式

使用指南

采样时，如果被采样的报文的长度小于配置的采样长度，那么采样全部报文。

配置举例

配置全局的sFlow采样报文长度为512byte:

```
ConnetOS# set protocols sflow header-length 512
```

set protocols sflow interface enable

命令功能

set protocols sflow interface enable 命令用来配置是否使能接口下的sFlow功能。
delete protocols sflow interface enable 命令用来删除接口下的sFlow功能。
缺省情况下，接口下的sFlow功能并没有使能。

命令格式

set protocols sflow interface *interface-name* **enable** { **falses** | **true** }
delete protocols sflow interface *interface-name* **enable**

参数说明

interface-name：接口名称。
false：去使能接口下的sFlow功能。
true：使能接口下的sFlow功能。

命令模式

配置模式

使用指南

如果sFlow功能要生效，必须使能具体接口的sFlow功能。

配置举例

使能接口te-1/1/1的sFlow功能:

```
ConnetOS# set protocols sflow interface te-1/1/1 enable true
```

set protocols sflow interface header-length

命令功能

set protocols sflow interface header-length 命令用来配置全局的sFlow采样报文长度。

delete protocols sflow interface header-length 命令用来删除配置的全局采样报文长度，恢复为缺省值。

缺省情况下，采样报文长度是128字节。

命令格式

set protocols sflow interface *interface-name* **header-length** *header-length*

delete protocols sflow interface *interface-name* **header-length**

参数说明

header-length：采样报文长度。整数形式，取值范围是64～512，单位是字节。

命令模式

配置模式

使用指南

如果接口下配置了报文采样长度，就按照接口下的采样长度进行采样。如果接口下没有配置，就按照全局的采样长度进行采样。

采样时，如果被采样的报文的长度小于配置的采样长度，那么采样全部报文。

配置举例

配置接口te-1/1/1的sFlow采样报文长度为512 byte:

```
ConnetOS# set protocols sflow interface te-1/1/1 header-length 512
```

set protocols sflow interface polling-interval

命令功能

set protocols sflow interface polling-interval 命令用来配置指定接口的sFlow采样时间间隔。

delete protocols sflow interface polling-interval polling 命令用来删除配置的sFlow采样时间间隔，恢复为缺省值。

缺省情况下，接口的采样间隔是0，即不采样。

命令格式

set protocols sflow interface *interface-name* **polling-interval** *polling-interval*

delete set protocols sflow interface *interface-name* **polling-interval**

参数说明

polling-interval：报文采样时间间隔。整数形式，取值范围是0～86400，单位是秒。

命令模式

配置模式

使用指南

如果接口下配置了采样间隔，就按照接口下的采样间隔进行采样。如果接口下没有配置，就按照全局的采样间隔进行采样。

配置举例

配置接口下的sFlow报文采样时间间隔是100秒:

```
ConnetOS# set protocols sflow interface te-1/1/1 polling-interval 100
```

set protocols sflow interface sampling-rate

命令功能

set protocols sflow interface sampling-rate 命令用来配置指定接口的sFlow采样比。

delete protocols sflow interface sampling-rate 命令用来删除配置的指定接口的采样比，恢复为缺省值。

缺省情况下，出入方向的接口sFlow采样比都是2000，即每2000个报文采样一个报文。

命令格式

set protocols sflow interface *interface-name* **sampling-rate** { **egress** | **ingress** } *sampling-rate*

delete protocols sflow interface *interface-name* **sampling-rate** { **egress** | **ingress** }

参数说明

egress：对出方向的报文进行采样。

ingress：对入方向的报文进行采样。

sampling-rate：sFlow采样比。整数形式，取值范围是1000～1000000。

命令模式

配置模式

使用指南

无

配置举例

设置sFlow对接口te-1/1/1的出方向采样比为1500:

```
ConnetOS# set protocols sflow interface te-1/1/1 sampling-rate egress 1500
```

set protocols sflow polling-interval

命令功能

set protocols sflow polling-interval 命令用来配置全局的sFlow采样时间间隔。
delete protocols sflow polling-interval 命令用来删除配置的sFlow采样时间间隔，恢复为缺省值。
缺省情况下，全局的采样间隔是0，即不采样。

命令格式

set protocols sflow polling-interval *polling-interval*
delete set protocols sflow polling-interval

参数说明

polling-interval：报文采样时间间隔。整数形式，取值范围是0～86400，单位是秒。

命令模式

配置模式

使用指南

无

配置举例

配置全局的sFlow报文采样时间间隔是100秒:

```
ConnetOS# set protocols sflow polling-interval 100
```

set protocols sflow sampling-rate

命令功能

set protocols sflow sampling-rate 命令用来配置指定全局的sFlow采样比。

delete protocols sflow interface sampling-rate 命令用来删除配置的全局采样比，恢复为缺省值。

缺省情况下，出入方向的全局sFlow采样比都是2000，即每2000个报文采样一个报文。

命令格式

set protocols sflow sampling-rate { **egress** | **ingress** } *sampling-rate*

delete protocols sflow sampling-rate { **egress** | **ingress** }

参数说明

egress：对出方向的报文进行采样。

ingress：对入方向的报文进行采样。

sampling-rate：sFlow采样比。整数形式，取值范围是1000～1000000。

命令模式

配置模式

使用指南

无

配置举例

设置sFlow全局出方向采样比为1500:

```
ConnetOS# set protocols sflow sampling-rate egress 1500
```

show sflow

命令功能

show sflow 命令用来查看sFlow的配置信息。

命令格式

show sflow [**collector** | **interface**]

参数说明

collector：查看sFlow collector的配置信息。
interface：查看接口的sFlow配置信息。

命令模式

运维模式

使用指南

所有配置都必须 **commit** 之后，才能查看到配置信息。

配置举例

查看全局的sFlow的信息:

```
ConnetOS> show sflow
Status          Agent IP Address  Ingress Sample Rate  Egress Sample Rate  Polling␣
↪Interval  Header Length
--------        -----------------        --------------------        ------------------- ␣
↪           -----------------    --------------
Enabled         192.168.1.35    1:2000              1:2000              60s            ␣
↪64B
```

sDrop命令

set protocols sdrop collector address

命令功能

set protocols sdrop collector address 命令用来指定sDrop Collector。
delete protocols sdrop collector address 命令用来删除指定的sDrop Collector。
缺省情况下，没有指定sDrop Collector。

命令格式

set protocols sdrop collector address *ip-address*

delete protocols sdrop collector address

参数说明

ip-address：sDrop Collector的IP地址。

命令模式

配置模式

使用指南

只能指定一个sDrop Collector。

配置举例

指定IP地址是1.1.1.1的服务器作为本设备的sDrop Collector。：

```
ConnetOS# set protocols sdrop collector address 1.1.1.1
```

set protocols sdrop collector port

命令功能

set protocols sdrop collector port 命令用来配置sDrop Collector的UDP端口号。

delete protocols sdrop collector port 命令用来删除配置的轮UDP端口号，恢复为缺省值。

缺省情况下， sDrop Collector的UDP端口号为32768。

命令格式

set protocols sdrop collector port *port-number*

delete protocols sdrop collector port

参数说明

port-number：UDP端口号。整数形式，取值范围是32768～65535。

命令模式

配置模式

使用指南

无。

配置举例

配置sDrop Collector的UDP端口号为65535:

```
ConnetOS# set protocols sdrop collector port 65535
```

set protocols sdrop polling-interval

命令功能

set protocols sdrop polling-interval 命令用来配置sDrop采样的轮询时间。
delete protocols sdrop polling-interval 命令用来删除配置的轮询时间，恢复为缺省值。
缺省情况下，采样时间是60s。

命令格式

set protocols sdrop polling-interval *polling-interval*
delete protocols sdrop polling-interval

参数说明

polling-interval：报文采样的轮询时间。整数形式，取值范围是1～60，单位是秒。

命令模式

配置模式

使用指南

无。

配置举例

配置采样时间间隔是30秒:

```
ConnetOS# set protocols sdrop collector polling-interval 30
```

show sdrop

命令功能

show sdrop 命令用来查看设备上报文的丢弃信息。

命令格式

show sdrop [**detail**]

参数说明

detail：查看被丢弃报文的详细信息。

命令模式

运维模式

使用指南

无。

配置举例

查看设备上报文丢弃的信息:

```
ConnetOS 1> show sdrop
Info of Dropped Packets in last 1 min.
Input Physical Port     Output Physical Port    Drop Reason
↪  Last Detectted Time
--------------------    --------------------    -----------------------------------
↪  --------------------
NA                      te-1/1/2                Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
NA                      te-1/1/5                Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
NA                      te-1/1/6                Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
NA                      te-1/1/48               Exceed Egress Buffer Threshold
↪  2017-03-23 14:56:55
```

navmesh流量查询

show navmesh

命令功能

show navmesh 命令用来查看指定五元组流量的出入端口信息。

命令格式

show navmesh source-ipv4 *source-ipv4-address* **dest-ipv4** *dest-ipv4-address* **l4-source-port** *source-port-number* **l4-dest-port** *dest-port-number* **protocol** *protocol-number*

参数说明

source-ipv4-address：源IP地址。

dest-ipv4-address：目的IP地址。

source-port-number：源端口号。整数形式，取值范围是1～65535。

dest-port-number：目的端口号。整数形式，取值范围是1～65535。

protocol-number：IP协议号。

命令模式

运维模式

使用指南

无。

配置举例

查看源地址为10.10.10.10，目的地址40.40.40.10，源端口号为55，目的端口为11，协议号为17的流量的出入端口：

```
ConnetOS> show navmesh source-ipv4 10.10.10.10 dest-ipv4 40.40.40.10 l4-source-port 55
l4-dest-port 11 protocol 17
The Flow Information :
  Source ipv4 address          : 10.10.10.10
  Destination ipv4 address     : 40.40.40.10
  L4 source port               : 55
  L4 destination port          : 11
  IP protocol                  : 17
The forwarding path :
   Ingress Port   Egress Port
   ------------   ------------
   te-1/1/5       te-2/1/17
```

堆叠命令

堆叠命令

set member-id

命令功能

set member-id 命令用来切换设备模式及member ID。
缺省情况下，设备的member ID是0，即处于单机模式。

命令格式

set member-id *member-id*

参数说明

member-id：堆叠设备的member ID。整数形式，取值范围是0～2。

- 0：表示设备处于单机模式。
- 1：表示设备处于堆叠模式，且成员ID为1。
- 2：表示设备处于堆叠模式，且成员ID为2。

命令模式

运维模式

使用指南

如果想要使用堆叠功能，必须首先配置member ID，将设备设置成堆叠模式。
重启设备后，即可进行堆叠功能的配置。

Note: 成员设备的member ID不能设备为相同的。

配置举例

设置本设备为堆叠模式，member ID为1:

```
ConnetOS> set member-id 1
Are you sure you want to change the member-id(yes/no)?
Now you need to reboot to enable stack mode or the new member-id.
```

set interface gigabit-ethernet iss-port enable

命令功能

set interface gigabit-ethernet iss-port enable 命令用来配置是否将指定接口配置成堆叠接口。
delete interface gigabit-ethernet iss-port enable 命令用来删除堆叠接口。
缺省情况下，设备上没有堆叠接口。

命令格式

set interface gigabit-ethernet *interface-name* **iss-port enable** { **false** | **true**}

delete interface gigabit-ethernet *interface-name* **iss-port enable**

参数说明

interface-name：堆叠接口。

false：不使能指定接口的堆叠功能。

true：使能指定接口的堆叠功能。

命令模式

配置模式

使用指南

堆叠接口是堆叠设备连接的接口， 一般在进行ISS配置的时候指定，配置完成重启后，成员设备进行初始化时完成端口模式的转换及进行相关配置。在运行的过程中也可以动态添加或者删除堆叠口成员。

如果指定多个物理端口为堆叠接口，那么这些堆叠接口将形成汇聚接口组，进行流量负载分担。 跨设备通信的报文需要通过堆叠接口转发。

配置举例

设置接口te-1/1/13为堆叠接口:

```
ConnetOS# set interface gigabit-ethernet te-1/1/13 iss-port enable true
```

set iss member mad enable

命令功能

set iss member mad enable 命令用来配置是否使能ISS MAD检测功能。

delete member mad enable 命令用来删除配置的ISS MAD检测功能，恢复为缺省值。

缺省情况下，ISS MAD检测功能没有使能。

命令格式

set iss member *member-id* **mad enable** { **false** | **true** }

delete iss member *member-id* **mad** [**enable**]

参数说明

member-id：堆叠设备的member ID。

false：不使能ISS MAD的检测功能。

true：使能ISS MAD的检测功能。

命令模式

配置模式

使用指南

当一个堆叠好的系统发生分裂时，会成为两个独立的ISS系统，各自成为Master，这种现象的出现是会影响网络的可用性的。这时，可以使用MAD检测ISS系统是否有发生分裂，以便进行及时处理。

配置举例

使能ISS MAD的检测功能:

```
ConnetOS# set iss member 1 mad enable true
```

set iss member mad excluded-interface

命令功能

set iss member mad excluded-interface 命令用来配置ISS MAD检测时的预留接口。

delete iss member mad excluded-interface 命令用来删除配置的ISS MAD检测预留接口。

缺省情况下，没有配置ISS MAD的检测预留接口。

命令格式

set iss member *member-id* **mad excluded-interface** *interface-name*

delete iss member *member-id* **mad excluded-interface**

参数说明

member-id：堆叠设备的member ID。

interface-name：进行ISS MAD检测的预留接口，可配置多个。

命令模式

配置模式

使用指南

堆叠分裂时，除了预留接口，其他接口都会shutdown。

配置举例

配置进行ISS MAD检测的预留接口为te-1/1/2:

```
ConnetOS# set iss member 1 mad excluded-interface te-1/1/2
```

set iss member mad interface

命令功能

set iss member mad interface 命令用来配置ISS MAD检测的接口。
delete iss memberme mad interface 命令用来删除配置的ISS MAD检测接口。
缺省情况下，没有配置ISS MAD的检测接口。

命令格式

set iss member *member-id* **mad interface** *interface-name*
delete iss member *member-id* **mad interface**

参数说明

member-id：堆叠设备的member ID。
interface-name：进行ISS MAD检测的接口。

命令模式

配置模式

使用指南

无。

配置举例

配置进行ISS MAD检测的接口为te-1/1/1:

```
ConnetOS# set iss member 1 mad interface te-1/1/1
```

set iss member priority

命令功能

set iss member priority 命令用来设置堆叠系统成员设备的选举优先级。
delete iss member priority 命令用来删除配置的堆叠系统成员设备的选举优先级。
缺省情况下，设备成员的选举优先级是1。

命令格式

set iss member *member-id* **priority** *priority-number*
delete iss member *member-id* **priority**

参数说明

member-id：堆叠设备的member ID。
priority-number：堆叠系统成员设备的优先级。整数形式，取值范围是0～32。数值越大的，优先级越高。

命令模式

配置模式

使用指南

无。

配置举例。

设置成员编号为1的成员优先级为4:

```
ConnetOS# set iss member 1 priority 4
```

set system hostname (stack模式)

命令功能

set iss member hostname 命令用来设置设备名称。
delete iss member hostname 命令用来删除配置的设备名称，恢复到缺省值。
缺省情况下，设备名称为ConnetOS。

命令格式

set iss member *member-id* **hostname** *hostname*

delete iss member *member-id* **hostname**

参数说明

member-id：堆叠设备的member ID。整数形式，取值范围是0～2。

- 0：表示设备处于单机模式。
- 1：表示设备处于堆叠模式，且member ID为1。
- 2：表示设备处于堆叠模式，且member ID为2。

hostname：本地文件名称。字符串形式，取值范围是1～63。支持区分大小写不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

设置设备名称为switcha:

```
ConnetOS 1# set iss member hostname switcha
```

show iss

命令功能

show iss 命令用来查看堆叠系统中的成员设备信息。

命令格式

show iss

参数说明

无

命令模式

运维模式

使用指南

此命令可以查看成员设备的Member ID、角色、选举优先级、设备MAC、堆叠MAC、设备名称信息。

配置举例

查看堆叠系统中的成员设备信息:

```
ConnetOS 1> show iss
Member ID   Role     Priority   Device MAC          ISS MAC             Hostname
---------   ------   --------   -----------------   -----------------   --------------
↪--
1           Master   1          00:03:0f:64:da:5f   00:03:0f:64:da:5f   BJ-YUNQI-
↪C1020-31.Int
2           Slave    1          00:03:0f:64:da:53   00:03:0f:64:da:5f   BJ-YUNQI-
↪C1020-32.Int
```

项目	含义
Member ID	成员ID。
Role	角色。
Priority	选举优先级。
Device MAC	设备MAC地址。
ISS MAC	堆叠系统对外的MAC地址。
Hostname	设备名称。

show iss configuration

命令功能

show iss configuration 命令用来查看堆叠系统中成员设备的配置信息。

命令格式

show iss configuration

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看堆叠系统中成员设备的配置信息:

```
ConnetOS 1> show iss configuration
Member ID   ISS Link Status   Interface        Interface Status   Neighbour
---------   ---------------   --------------   ----------------   --------------
1           Up                qe-1/1/49        Up                 qe-2/1/49
                              qe-1/1/52(*)     Up                 qe-2/1/52

2           Up                qe-2/1/49        Up                 qe-1/1/49
                              qe-2/1/52(*)     Up                 qe-1/1/52

-----------------------------------------
 * indicates the control interface of ISS.
```

项目	含义
Member ID	成员ID。
ISS Link Status	堆叠链路状态。
Interface	堆叠接口，用于连接邻居成员设备。
Interface Status	接口状态。
Neighbour	邻居成员设备的接口。
•	传输控制报文的接口，为自动选举接口。

show iss mad

命令功能

show iss mad 用来查看MAD的检测和处理情况。

命令格式

show iss mad

参数说明

无

命令模式

运维模式

使用指南

MAD：Multi-Active Detection，多Active检测。是一种检测和处理堆叠分裂后产生的多个Master的机制。

配置举例

查看MAD的检测和处理信息:

```
ConnetOS 1> show iss mad
Member ID   Management   MAD State           MAD interface         Neighbor
↪     Excluded interfaces
---------   ----------   ------------------   -------------------   ------------------
↪--   --------------------
1           Enabled      Detect              qe-1/1/54             qe-2/1/54
↪     N/A
2           Enabled      Detect              qe-2/1/54             qe-1/1/54
↪     N/A
```

show iss statistics

命令功能

show iss statistics 用来查看堆叠接口上报文的统计信息。

命令格式

show iss statistics

参数说明

无

命令模式

运维模式

使用指南

ISS的主要报文有：

- Hello报文：点对点报文，在相邻设备间交互，携带本设备所收集到的所有的设备信息、优先级信息和其它上下文信息。
- Elect选举报文：点对点报文，设备仅仅携带用于选举的相关信息，如设备MAC地址，优先级，设备运行时间等。
- ElectAck：Elect选举回应报文。
- Anno通告报文：竞选结果通告报文，Master发送宣布竞选结果，Slave回复ACK进行确认。

- AnnoAck：Anno通告回应报文。
- Urgent报文：广播报文，用于ISS系统紧急事件的通告，如堆叠口DOWN。

当堆叠正常运行时，只会收发hello报文。

配置举例

来查看堆叠接口上各个类型的报文收发计数统计信息:

```
ConnetOS 1> show iss statistics
Interface           Packet Type         Input               Output
----------          -----------         ----------          ----------
qe-1/1/52           Hello               166748              166748
                    Elect               0                   1
                    ElectAck            4                   0
                    Anno                0                   1
                    AnnoAck             2                   0

qe-1/1/49           Hello               166747              166748
                    Elect               0                   0
                    ElectAck            0                   0
                    Anno                0                   0
                    AnnoAck             0                   0
```

项目	含义
Interface	堆叠接口名称
Packet Type	报文类型
Input	接收的报文
Output	发送的报文

show iss sync-status

命令功能

show iss sync-status 命令用来查看堆叠系统内成员设备的配置同步状态。

命令格式

show iss sync-status

参数说明

无

命令模式

运维模式

使用指南

堆叠建立时，当Slave重启后，会自动同步Master的配置信息。**show iss sync-status** 命令用来查看设备同步的时间和状态。

配置举例

查看堆叠系统内成员设备的配置同步状态:

```
ConnetOS 1> show iss sync-status
Member ID  Role    State    Last Sync Time
---------  ------  -------  -------------------
1          Master  Full     2017-03-27 20:32:10
2          Slave   Full     2017-03-27 20:32:10
```

项目	含义
Member ID	成员ID
Role	成员角色
State	设备状态
Last Sync Time	最后一次同步时间

VXLAN命令

VXLAN命令

set vsi vsi-id

命令功能

set vsi vsi-id 命令用来创建VSI。

delete vsi vsi-id 命令用来删除创建的VSI。

缺省情况下，设备上没有创建好的VSI。

命令格式

set vsi vsi-id *vsi-id*

delete vsi vsi-id *vsi-id*

参数说明

vsi：VSI ID，用于标识VSI。整数形式，取值范围是1～8192。

命令模式

配置模式

使用指南

报文是否进入VXLAN隧道是根据报文是否匹配到VSI（Virtual Switching Instance，虚拟交换实例）来决定的，VSI可以看做是VTEP上的一台基于VXLAN进行二层转发的虚拟交换机，它具有传统以太网交换机的所有功能，包括源MAC地址学习、MAC地址老化、泛洪等。VSI与VXLAN一一对应。可以用VSI来标识VXLAN。

配置举例

配置VSI ID为1的VSI:

```
ConnetOS# set vsi vsi-id 1
```

set vsi vsi-id description

命令功能

set vsi vsi-id description 命令用来增加对VSI的描述。

delete vsi vsi-id description 命令用来删除指定VSI的描述。

缺省情况下，VSI创建后没有描述。

命令格式

set vsi vsi-id *vsi-id* **description** *description*

delete vsi vsi-id *vsi-id* **description**

参数说明

vsi-id：当前已经创建好的VSI ID。

description：对VSI的描述。字符串形式，不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

增加对VSI 1的描述:

```
ConnetOS# set vsi vsi-id 1 description toswtichb
```

set vsi vsi-id flooding enable

命令功能

set vsi vsi-id flooding enable 命令用来配置是否使能VSI向tunnel方向的泛洪抑制功能，向本地的业务接入点泛洪不受抑制。

delete vsi vsi-id 命令用来删除VSI向tunnel方向的泛洪抑制功能。

缺省情况下，VSI向tunnel方向的泛洪抑制功能没有使能。

命令格式

set vsi vsi-id *vsi-id* **flooding enable** { **false** | **true** }

delete vsi vsi-id *vsi-id* **flooding** [**enable**]

参数说明

vsi-id：VSI ID。整数形式，取值范围是1～8192。

false：不使能VSI泛洪抑制功能。

true：使能VSI泛洪抑制功能。

命令模式

配置模式

使用指南

缺省情况下，VTEP从本地站点内接收到BUM数据帧后，会在该VXLAN内除接收接口外的所有本地接口和VXLAN隧道上泛洪该数据帧，将该数据帧发送给VXLAN内的所有站点。如果用户希望把该类数据帧限制在本地站点内，不通过VXLAN隧道将其转发到远端站点，则可以通过配置命令手工禁止VXLAN对应VSI的泛洪功能。

配置举例

使能VSI泛洪抑制功能:

```
ConnetOS# set vsi vsi-id 1 flooding enable true
```

set vsi vsi-id tunnel-ethernet

命令功能

set vsi vsi-id tunnel-ethernet 命令用来关联VSI和VXLAN隧道。

delete vsi vsi-id tunnel-ethernet 命令用来删除VSI关联的VXLAN隧道。

缺省情况下，VSI没有关联VXLAN隧道。

命令格式

set vsi vsi-id *vsi-id* **tunnel-ethernet** *tunnel-name*

delete vsi vsi-id *vsi-id* **tunnel-ethernet** *tunnel-name*

参数说明

vsi-id：VSI ID。整数形式，取值范围是1～8192。

tunnel-name：VXLAN隧道名称，取值范围是tunnel1~tunnel4094。

命令模式

配置模式

使用指南

VTEP必须与相同VNI内的其它VTEP建立VXLAN隧道，ConnetOS通过将VNI和VSI绑定，再将VSI和VXLAN隧道绑定实现。

配置举例

配置VSI 1关联隧道tunnel1:

```
ConnetOS# set vsi vsi-id 1 tunnel-ethernet tunnel1
```

set vsi vsi-id vni

命令功能

set vsi vsi-id vni 命令用来给指定的VSI关联VNI。

delete vsi vsi-id vni 命令用来删除VSI关联的VNI。

缺省情况下，VSI没有关联VNI。

命令格式

set vsi vsi-id *vsi-id* **vni** *vni-id*

delete vsi vsi-id *vsi-id* **vni**

参数说明

vsi-id：VSI ID。整数形式，取值范围是1～8192。

vni-id：VNI ID。整数形式，取值范围是1～16777215。

命令模式

配置模式

使用指南

VSI与VNI一一对应，一个VSI只能关联一个VNI，一个VNI只能被一个VSI绑定。

配置举例

将vsi 1和vni 1关联:

```
ConnetOS#  set vsi vsi-id 1 vni 1
```

set interface tunnel-ethernet

命令功能

set interface tunnel-ethernet 命令用来新创建一条隧道。

delete interface tunnel-ethernet 命令用来删除已经配置好的隧道。

缺省情况下，没有创建隧道。

命令格式

set interface tunnel-ethernet *tunnel-name*

delete interface tunnel-ethernet *tunnel-name*

参数说明

tunnel-name：隧道名称。字符串形式，取值形式为tunnelx。x为整数，取值范围是1～4094。

命令模式

配置模式

使用指南

无。

配置举例

创建名字为tunnel1的隧道:

```
ConnetOS# set interface tunnel-ethernet tunnel1
```

set interface tunnel-ethernet description

命令功能

set interface tunnel-ethernet description 命令用来增加对隧道的描述。

delete interface tunnel-ethernet description 命令用来删除已经配置的隧道描述。

缺省情况下，隧道创建后没有配置描述。

命令格式

set interface tunnel-ethernet *tunnel-name* **description** *description*

delete interface tunnel-ethernet *tunnel-name* **description**

参数说明

tunnel-name：隧道名称。字符串形式，取值形式为tunnelx。x为整数，取值范围是1～4094。

description： 对隧道的描述。字符串形式，取值范围是1～32。区分大小写，支持特殊字符，但不支持空格。

命令模式

配置模式

使用指南

无。

配置举例

增加对隧道的描述:

```
ConnetOS# set interface tunnel-ethernet tunnel1 description toswitcha
```

set interface tunnel-ethernet destination address

命令功能

set interface tunnel-ethernet destination address 命令用来配置隧道的目的端IP地址。
delete interface tunnel-ethernet destination 命令用来删除已经配置的隧道目的端IP地址。
缺省情况下，隧道创建后没有配置目的端IP地址。

命令格式

set interface tunnel-ethernet *tunnel-name* **destination address** *ip-address*
delete interface tunnel-ethernet *tunnel-name* **destination**

参数说明

tunnel-name：隧道名称。字符串形式，取值形式为tunnelx。x为整数，取值范围是1～4094。
ip-address：目的端IP地址。点分十进制形式。

命令模式

配置模式

使用指南

无。

配置举例

配置隧道tunnel1的目的端IP地址为2.2.2.2:

```
ConnetOS# set interface tunnel-ethernet tunnel1 destination address 2.2.2.2
```

set interface tunnel-ethernet mode vxlan

命令功能

set interface tunnel-ethernet mode vxlan 命令用来将隧道模式配置成VXLAN模式。

delete interface tunnel-ethernet mode 命令用来删除隧道的模式。
缺省情况下，隧道的模式是VXLAN。

命令格式

set interface tunnel-ethernet *tunnel-name* **mode vxlan**
delete interface tunnel-ethernet *tunnel-name* **mode**

参数说明

tunnel-name：隧道名称。字符串形式，取值形式为tunnelx。x为整数，取值范围是1～4094。

命令模式

配置模式

使用指南

无。

配置举例

将隧道tunnel1配置成为VXLAN隧道:

```
ConnetOS# set interface tunnel-ethernet tunnel1 mode vxlan
```

set interface tunnel-ethernet source address

命令功能

set interface tunnel-ethernet source address 命令用来配置隧道的源端IP地址。
delete interface tunnel-ethernet source address 命令用来删除用户配置的隧道源端IP地址。
缺省情况下，隧道创建后没有配置源端IP地址。

命令格式

set interface tunnel-ethernet *tunnel-name* **source address** *ip-address*
delete interface tunnel-ethernet *tunnel-name* **source** [**address**]

参数说明

tunnel-name：隧道名称。字符串形式，取值形式为tunnelx。x为整数，取值范围是1～4094。
ip-address：源端IP地址。点分十进制形式。用户最多可以配置510个不同的源IP地址。

命令模式

配置模式

使用指南

隧道的源端IP地址将作为封装后报文的源IP地址。

配置举例

配置隧道tunnel1的源端IP地址为1.1.1.1:

```
ConnetOS# set interface tunnel-ethernet tunnel1 source address 1.1.1.1
```

set interface tunnel-ethernet static-mac-address

命令功能

set interface tunnel-ethernet static-mac-address 命令用来为隧道配置远端的静态MAC地址表项。
delete interface tunnel-ethernet static-mac-address 命令用来删除隧道配置远端的MAC地址表项。
缺省情况下，隧道上不存在任何远端的静态MAC地址表项。

命令格式

set interface tunnel-ethernet *tunnel-name* **static-mac-address** *mac-address* [**vsi** *vsi-id*]
delete interface tunnel-ethernet *tunnel-name* [**static-mac-address** *mac-address* [**vsi** *vsi-id*]]

参数说明

tunnel-name：隧道名称。字符串形式，取值形式为tunnelx。x为整数，取值范围是1～4094。
mac-address：MAC地址。
vsi：VSI ID。整数形式，取值范围是1～8192。

命令模式

配置模式

使用指南

VXLAN隧道必须与vsi参数指定的VSI关联，且该VSI必须已经创建，否则配置将失败。
VSI与tunnel的关联之前需要先删除对应的静态MAC。

配置举例

配置VSI 1中VXLAN隧道1的远端MAC地址为00:00:00:00:00:22:

```
ConnetOS# set interface tunnel-ethernet tunnel1 static-mac-address 00:11:22:33:44:55␣
↪vsi 1
```

set interface family ethernet-switching vsi

命令功能

set interface family ethernet-switching vsi 命令用来配置VTEP上端口的VXLAN接入点。

delete interface family ethernet-switching vsi 命令用来删除配置的接口的VXLAN接入点。

缺省情况下，没有创建接口的VXLAN接入点。

命令格式

set interface { **gigabit-ethernet** | **aggregate-ethernet** } *interface-name* **family ethernet-switching vsi** *vsi-id* { **ethernet-mode enable true** | **vlan-mode dot1q** *vlan-id* }

delete interface { **gigabit-ethernet** | **aggregate-ethernet** } *interface-name* **family ethernet-switching vsi** { **ethernet-mode enable** | **vlan-mode dot1q** }

参数说明

interface-name：接口名称。

vsi-id：VSI ID。整数形式，取值范围是1～8192。

vlan-id：VLAN ID，允许进入VXLAN隧道的指定VLAN Tag。整数形式，取值范围为1～4094。

命令模式

配置模式

使用指南

缺省情况下，端口未配置VXLAN业务接入点。业务接入点有两种模式：

- VLAN模式
- Ethernet模式

同一个端口只能配置一种模式。

VLAN模式时一个端口可以配置不同的匹配规则关联不同的VSI，一个端口只能有一个VLAN关联到某个VSI，如果一个VSI已经关联了某个VLAN，该端口再配置VSI关联其它VLAN时，需要先会先删掉前面的关联之后再配置后面的关联，如果这个VLAN已经关联到了其它的VSI，则下发失败，需要先手动删除之前关联之后才能下发。

端口一旦配置成了VXLAN业务接入点之后就与传统的VLAN隔离了，不能在原有的VLAN内做二层转发，不能再配置VLAN的静态MAC，同样配置了VLAN静态MAC的端口需要先删除端口静态MAC配置之后再配置VXLAN业务接入点。

物理接口和汇聚接口配置VXLAN接入点互斥，即已经配置了业务接入点的物理接口不能加入汇聚接口，已经在汇聚接口中的物理接口不能配置业务接入点。

配置举例

配置端口te-1/1/1收到的VLAN 1的报文映射到VSI 1中:

```
ConnetOS# set interface gigabit-ethernet te-1/1/1 family ethernet-switching vsi 1␣
↪vlan-mode dot1q 1
```

show ethernet-switching table vxlan

命令功能

show ethernet-switching table vxlan 命令用来查看VXLAN的FDB信息。

命令格式

show ethernet-switching table vxlan

参数说明

无

命令模式

运维模式

使用指南

无。

配置举例

查看VXLAN的FDB信息:

```
ConnetOS> show ethernet-switching table vxlan
VSI       MAC address              Type         Age       Interfaces
----     ------------------        -------      ----      ----------
1         00:00:00:00:00:23        Dynamic      0         ae1
```

show interface tunnel-ethernet

命令功能

show interface tunnel-ethernet 命令用来查看隧道的信息。

命令格式

show interface tunnel-ethernet { *tunne-name* | **detail** }

参数说明

tunnel-name：隧道名称。
detail：查看隧道的详细信息。

命令模式

运维模式

使用指南

如果不指定detail，就显示所有tunnel的概要信息。

配置举例

查看VXLAN隧道tunnel1的信息:

```
ConnetOS> show interface tunnel-ethernet tunnel1 detail
Tunnel: tunnel1,   State: Up,   Description:
Source IP: 2.2.2.1,   Destination IP: 3.3.3.3
Mode: VXLAN
VNI ID:
  100,
Traffic statistics:
  5 sec input rate 0 packets/sec
  5 sec output rate 0 packets/sec
  Input Packets.....................0
  Output Packets....................8133931727
```

show vsi

命令功能

show vsi 命令用来查看VSI的信息。

命令格式

show vsi [**vsi-id** *vsi-id* | **detail**]

参数说明

vsi-id：当前已经创建好的VSI ID。

detail：查看VSI的详细信息。

命令模式

运维模式

使用指南

如果不指定 **detail**，就显示所有VSI的概要信息。

配置举例

查看VSI的详细信息:

```
BConnetOS> show vsi detail
VSI ID: 1
VNI: 100
Description:
Tunnel Flooding: true
VxLAN access ports:
  Interface   Dot1q   Mode
  ---------   -----  --------
  te-1/1/2    2       Vlan
  te-2/1/2    3       Vlan
Tunnel interface:
  tunnel1
  tunnel10
```

命令行格式约定

符号	说明
粗体	命令行关键字，必须原样输入。
斜体	命令行参数，需要按照要求输入实际值。
[ ]	命令行配置过程中，括号中的部分是可选的
[a | b | .....]	从括号中的值选取一个或不选
{ a | b | ..... }	从括号中的值选取一个
[a | b | .....] *	从括号中的值选取多个或不选
{ a | b | ..... } *	从括号中的值选取一个或多个

CHAPTER 5

解决方案

分流器解决方案

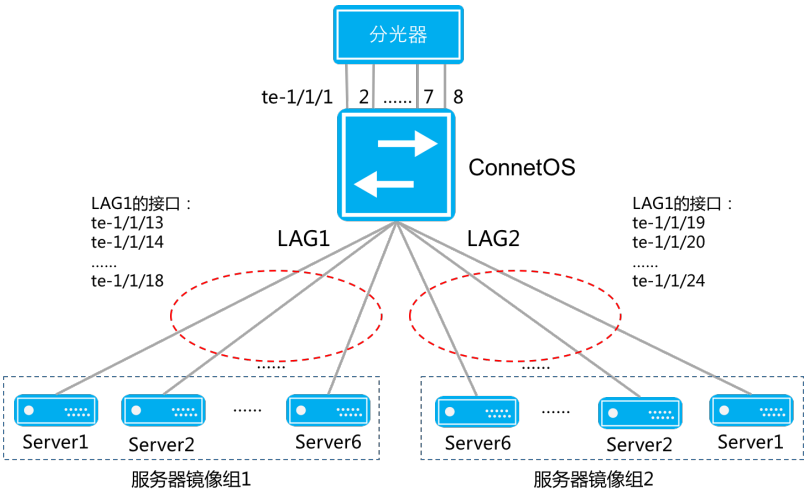
网络需求

在数据中心网络场景下，需要采用TAP（Test Access Point）方式监视所有数据中心的网络流量，要求分光器的流量进入指定端口。TAP流量镜像到两组服务器，每组服务器都得到完整的分光流量。

同时要求:

1. 分光器进入的流量不带VLAN Tag。
2. 服务器组内的流量进行负载分担，流量均匀并且同源同宿。

ConnetOS提供的负载均衡和对称哈希特性，可以支持数据中心网络场景下的TAP（Test Access Point）分流应用。



技术优势

配置思路简单

1. 首先将两组服务器组分别进行汇聚，然后将两个汇聚组和多个入口全部划入一个VLAN。
2. 利用广播特性让每个汇聚组都得到一份相同的流量。
3. 在入端口关闭MAC地址学习功能。

   考虑到流量镜像场景下多个入口进入的报文可能存在源目的MAC地址颠倒情况。为了避免交换机MAC地址学习后这 类报文无法转发，因此需要在入端口关闭MAC地址学习功能。
4. 为了确保同源同宿，使能LAG的对称哈希。

配置举例

举例如下:

```
ConnetOS> configure
ConnetOS# set vlans vlan-id 100
ConnetOS# set interface gigabit-ethernet te-1/1/13:18 ether-options 802.3ad ae1
ConnetOS# set interface gigabit-ethernet te-1/1/19:24 ether-options 802.3ad ae2
ConnetOS# set interface aggregate-ethernet ae1 family ethernet-switching native-vlan-
↪id 100
ConnetOS# set interface aggregate-ethernet ae2 family ethernet-switching native-vlan-
↪id 100
ConnetOS# set interface gigabit-ethernet te-1/1/1:8 family ethernet-switching native-
↪vlan-id 100
ConnetOS# set interface gigabit-ethernet te-1/1/1:8 ether-options mac-learning enable␣
↪false
ConnetOS# set forwarding-options load-balance lag symmetric enable true
ConnetOS# commit
```

网络质量感知解决方案

网络需求

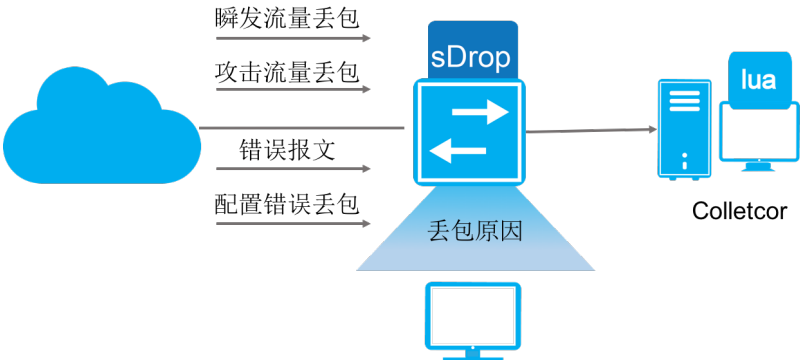
现代网络中，出现的网络问题大多和丢包有关，如何进行有效的丢包诊断，一直是一个难题。

为了解决这个问题，ConnetOS对网络环境中的丢包进行了详细的分析，设计了sDrop（Streaming Dropped packet）功能。

sDrop能够:

- 发现瞬间突发流量丢包。
- 协助解决业务配置错误丢包。
- 发现网络中存在攻击流量丢包。
- 发现网络中存在的错误的报文。

并且提供用户全网的丢包信息的收集及分析能力。



技术优势

sDrop配置简单，展示结果清晰，能够帮用户快速定位丢包原因。

配置简单

ConnetOS通过分析数据中心存在的20多种常见丢包，将报文分成两类处理。

- 可以获取丢弃的原始报文，将报文送往cpu进行分析，得到丢包原因。
- 无法获取丢弃的原始报文，通过获取统计信息，得到丢包的原因。

sDrop缺省情况下是使能的，用户只需要指定Collector即可。

展示方便

对于丢包的感知我们提供了两种的展示方式：设备上和设备外。

设备上

设备上我们提供了命令行，查看基本信息，可以定位基本的丢包原因:

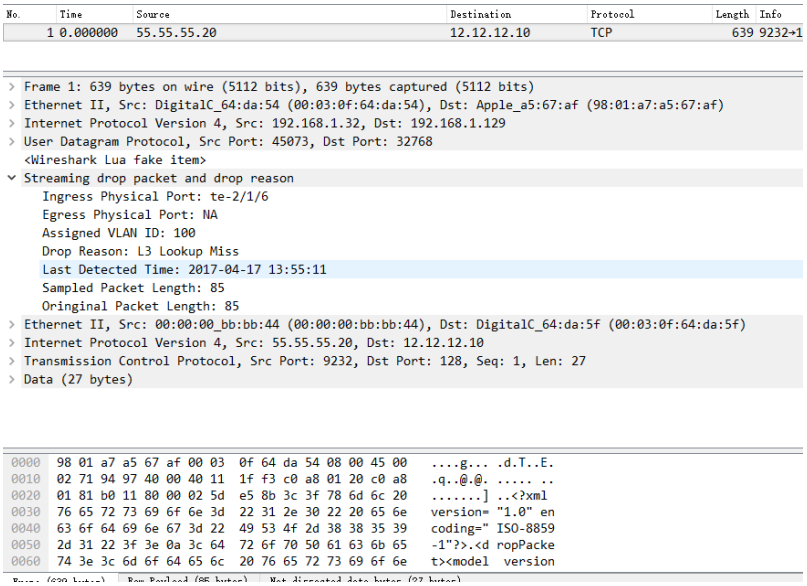
```
Info of Dropped Packets in last x min.
Input Physical Port     Out Physical Port       Drop Reason                 ␣
↪Last Detectted time
--------------------    -----------------       ---------------------------   --
↪------------------
te-1/1/1                NA                      Tag Vlan not exist          ␣
↪2001-11-23 11:22:23
te-1/1/1                NA                      Port not in Tag Vlan Member ␣
↪2001-11-23 11:22:23
te-1/1/1                NA                      Ingress MTU check fail      ␣
↪2001-11-23 11:22:23
te-1/1/1                te-1/1/2                Egress MTU check fail       ␣
↪2001-11-23 11:22:23
```

同时还提供了丢包的报文详细信息的查看，以分析实际的根本丢包原因:

```
Detail of Dropped packet in last x min.
Dropped packet detial entry 1:
    Meta Data:
        Ingress Physical Port: te-1/1/6      Egress Physical Port: NA
        Drop Reason: L3 Lookup Miss
        Last Detectted Time: 2001-11-23 11:22:33
    Packet Data:
        Dmac: 00:11:11:22:22:22  Smac: 00:22:22:33:33:33  Ethertype: 0x800  Length:153
        Ip Version: Ipv4    Tos: 12     Ip Length: 135  TTL: 32     Protocol:6
        Source Ip: 52.52.52.52    Dest Ip: 12.12.12.12
        L4 Source Port: 2048     L4 Dest Port: 1024
```

设备外

设备支持将报文的丢包信息导出，通过wireshark的lua插件进行展示，展示的结果如下所示。



CDN负载均衡

网络需求

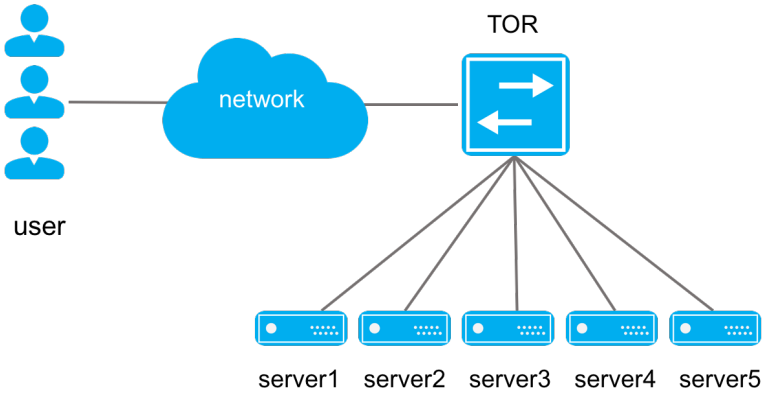
CDN（Content Delivery Network）内容分发网络，目的是通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的网络”边缘”，使用户可以就近取得所需的内容，解决Internet网络拥塞状况，提高用户访问网站的响应速度。

实现CDN的关键技术是服务器负载均衡，一般通过专用的硬件负载均衡器或软件负载均衡器实现。但是硬件负载均衡器价格高昂，而软件负载均衡性能差。

技术优势

ConnetOS支持的一致性Hash的负载均衡，在CDN场景下，不但可以作为缓存服务器的大密度高带宽的接入TOR，同时还可以作为高性能的负载均衡器。相较于专用的硬件和软件的负载均衡器，采用TOR实现负载

均衡可以降低成本、提高性能、提高稳定性、简化网络拓扑及减小运维难度等优势。



实现方案

实现TOR对多台服务器负载均衡方案的核心在于TOR上建立到服务器VIP的ECMP路由，通过配置TOR和服务器即可实现。有两种配置方案:

- 静态路由配置
- OSPF动态路由配置

静态路由和OSPF动态路由两种配置方案的比较如下所示:

比较	静态路由配置	OSPF动态路由
优点	易于配置无需安装Quagge	扩展服务器时不需要配置TOR，仅配服务器即可
缺点	扩展服务器时需要配置TOR	需要熟悉OSPF及Quagga的使用，学习和运维成本高

从表中可以看出两种方案各有优缺点，需要根据实际的应用场景比如网络的规模做权衡而采用适合的方案。

网络可视化

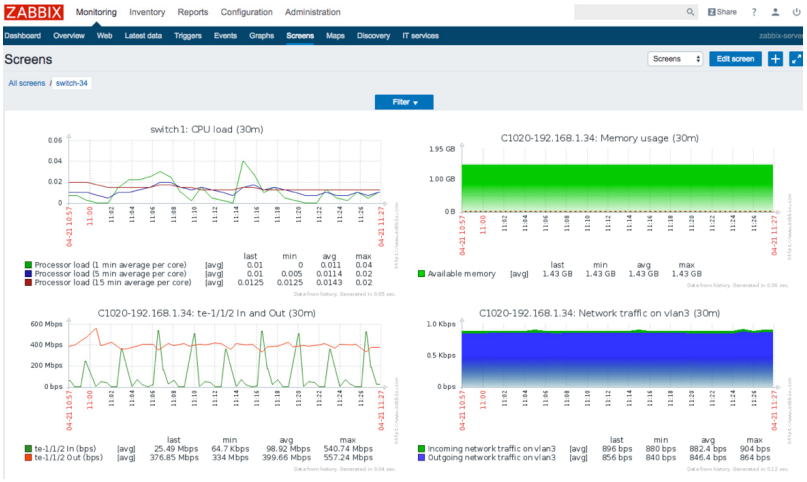
网络需求

在运维中，网络专家需要从大量的数据中，寻找异常数据或探测出网络趋势。网络可视化以生动形式呈现隐藏的庞大数据，有效帮助用户做业务的数据洞察

技术优势

Zabbix是一个企业级的、开源的、分布式的监控套件。可以实时监控交换机的系统、网络和服务状况。在ZabbixServer上导入ConnetOS的Zabbix监控模板后便可以支持网络的可视化监控。

ConnetOS能够提供给用户生动、形象的监控效果。



实现方案

导入监控模板

通过Web浏览器登录Zabbix的控制平台，进入导入模板的界面，Configuration->Templates->Import。

选中本地的模板文件ConnetOS_Zabbix_Template.xml，点击Import。

设置SNMP Community宏，Administration->General->Macros 此处的配置要与交换机中的community保持一样。

导入ConnetOS模板

Zabbix需要外部shell脚本实现对交换机内存的监控，登录安装Zabbix Server的服务器，进入Zabbix执行外部脚本的目录/usr/lib/zabbix/externalscripts，该目录可以通过Zabbix Server的配置文件zabbix_server.conf找到。

将附件中的ConnetOS_Zabbix_Script.sh复制到usr/lib/zabbix/externalscripts/中，并更改权限为777:

```
$ sudo cp ConnetOS_Zabbix_Script.sh /usr/lib/zabbix/externalscripts/
$ sudo chmod 777 /usr/lib/zabbix/externalscripts/ConnetOS_Zabbix_Script.sh
```

创建Hosts

打开创建主机界面，Configuration->Hosts->Create host。

1. Host name 主机名称，最好与交换机名称一致，可再带上交换机的IP。
2. Group 选择交换机所属的群组Switch。
3. Agent interfaces 填写交换机的IP。
4. SNMP interfaces 填写交换机的IP。
5. 选择Host选择旁边的Templates选项为主机选择所属的模板。
6. 点击Select进入到模板选择界面。
7. Group 这里选择之前建立的Switch Templates。
8. 选择主机所属的模板，点击Select返回Host的Templates界面。

9. 再次执行第6步的操作添加Linux模板，这次选择Templates的Group。选择Template OS Linux的模板之后点击Select。
10. 点击Add，确定增加这个模板给到这个Host，这一步很重要。
11. 确定Linked templates里面有我们刚才选择的模板之后，点击Add，创建完成。

创建完成后主机会自动建立模板中创建好的Item。Availablity这一栏中的SNMP是绿色的，表示通过SNMP监控成功，不成功则为红色。

配置ConnetOS

进入ConnetOS CLI之后做如下配置:

```
ConnetOS>
ConnetOS> configure
ConnetOS# set protocols snmp community public
ConnetOS# commit
```

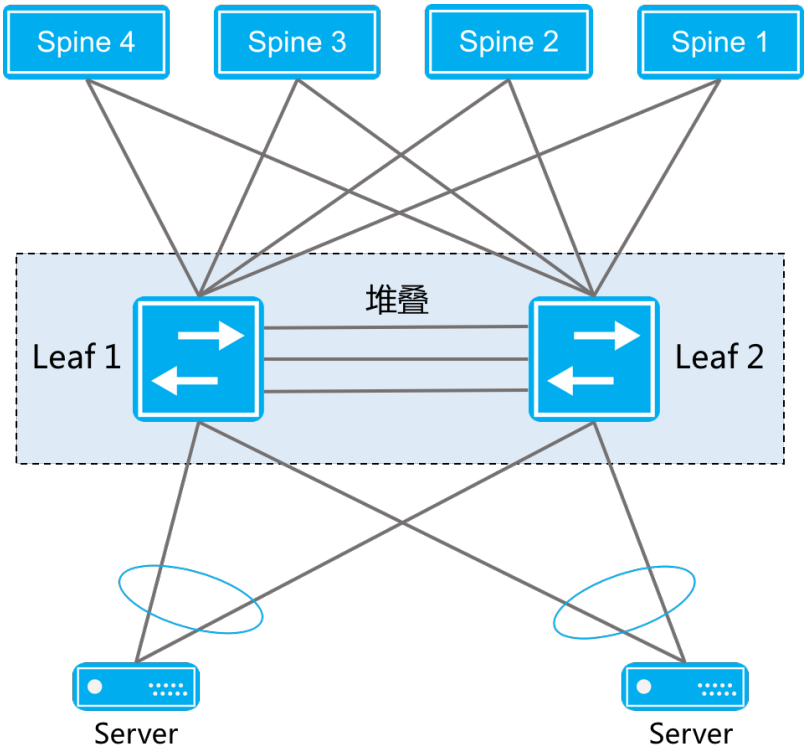
高可靠网络

网络需求

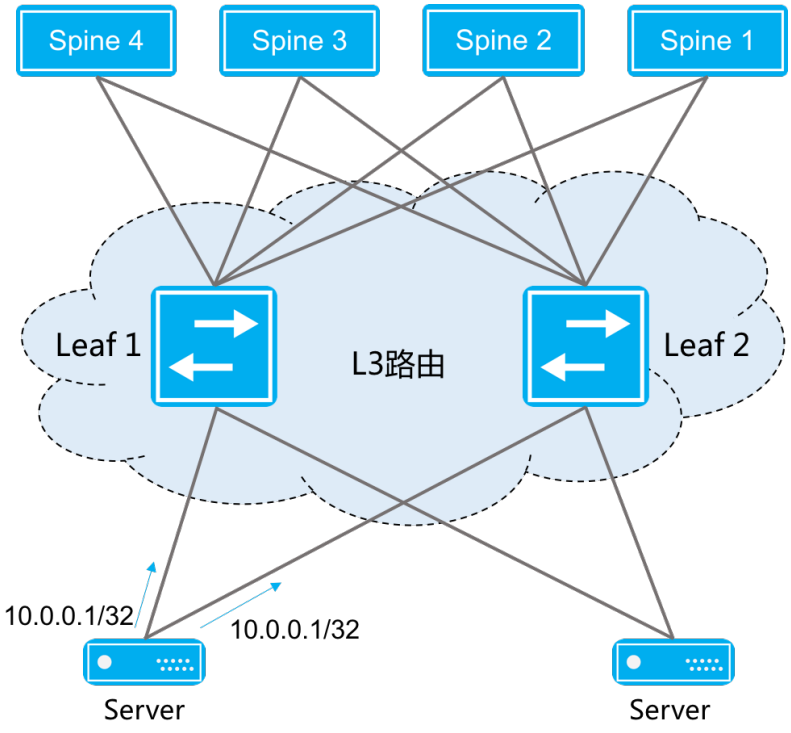
云计算的发展，对网络的稳定性提出了更高的要求。ConnetOS操作系统提供多种解决方案，以应对云计算用户对基础网络稳定性的需求。

技术优势

传统堆叠网络方案



现代数据中心网络方案



方案对比

对比	传统堆叠方案	现代数据中心网络方案
维护	管理方便，维护简单	网络架构一致性好，配置相对复杂
稳定性	稳定性差，升级难度高	扩展性强，稳定性高
兼容性	跨厂商不兼容，问题定位困难	跨厂商兼容性好

CHAPTER 6

应用中心

网络监控

管理工具

故障诊断

CHAPTER 7

ONE精选

白盒交换机硬件架构

背景

第三方交换机软件

Cumulus

Snaproute

SONiC

简介

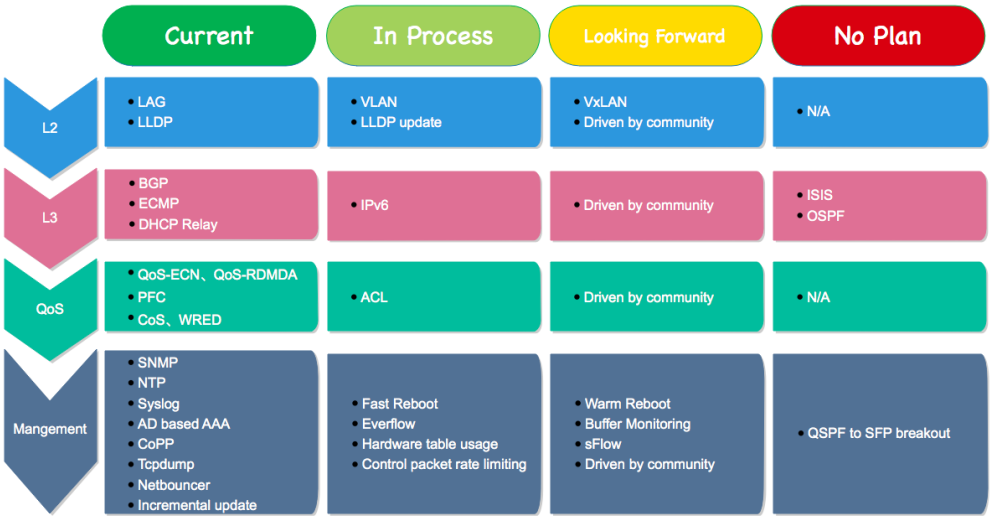
SONiC（Software for Open Networking in the Cloud）是微软开源的一套，据称能应用于云数据中心环境下的交换机三层软件功能组件集，该软件缺省运行在Debian GNU/Linux 8上。

功能介绍

基本功能

SONiC原本服务于微软数据中心网络，它实现的是微软数据中心里所需要使用到的一些功能。微软将SONiC开源出来后，虽然后续有计划再增加一些新的特性，但是从功能的完备性来讲，可覆盖的场景比较有限。虽然微软号称在云数据中心场景下得到了使用，但这并不能代表能示范性地被毫无代价平移到其他用户，微软能够成功驾驭很大程度上受益于微软网络架构方面的精简和强大控制能力。

下面我们来看看SONiC开源出来的特性，以及未来计划再增加的特性，甚至一些微软明确表示不会支持的特性。



亮点

微软的数据中心网络遍布全球，规模首屈一指，在数据中心网络的运营和管理上微软积累了丰富的经验。比如广为大家所熟悉的pingmesh、everflow、netbouncer等用于分析诊断方面的功能。

而微软开源SONiC的同时，将自己在数据中心网络运营实践方面的经验也一并开放给了外界，everflow、netbouncer等功能就是SONiC软件里要支持的功能。这些功能的开放，是广大网络运维工程师的福音，让网络排障再也不仅仅只是依靠ping、traceroute或者纯人工经验，而是走向智能化和科学化。

应用：SONiC和云数据中心

从目前云数据中心主流架构来看，为了提升可靠性，大部分的用户都选择了MLAG或堆叠的技术方案。MLAG或堆叠功能几乎成为云环境下网络架构的标配功能。

SONiC的Roadmap里并没有体现出对这方面的功能进行支持的计划，如果想依靠社区的力量补充该功能的话，风险和稳定性等都是需要谨慎考虑的问题。

也就是说，SONiC虽然提供了交换机的一些主要业务功能，但是目前对于绝大多数主流的云数据中心环境而言，是无法胜任的。

常见问题

SONiC是全开源的么?

答：SONiC不是全开源的，交换芯片适配部分，仅提供二进制格式。这主要受限于芯片厂商的License保护，微软也不能直接开源。

SONiC有统一的命令行入口么?

答：SONiC有命令行，但伴随模块的功能分散存在，没有统一的入口，并且有些模块的命令行缺少交互界面。

SONiC支持API接口么?

答：暂不支持。

SONiC支持堆叠或MLAG么?

答：暂不支持。

SONiC支持Openflow么?

答：暂不支持。

SONiC适合什么样的用户?

答：具备较强的交换机软件研发能力，并且数据中心网络架构为三层架构且功能要求相对比较简单，满足这些条件的用户可以尝试直接使用SONiC。

更多SONiC问题，请扫码探讨：

CHAPTER 8

硬件列表

软件版本	产品型号	端口规格	交换芯片	硬件厂商
ConnetOS 1	C1010A C1010B	48x10G + 4x40G 48x10G + 4x40G	BCM56846 BCM56846	广达 天弘
ConnetOS 2	C1020	48x10G + 6x40G	MT3257	智邦
ConnetOS 3(*)	C2010 C2020	48x25G + 6x100G 32x100G	NP836X NP836X	智邦 智邦

注: (*)表示研发中，2017年下半年release。

CHAPTER 9

资料下载

ONS 2017

幻灯片合集下载链接：https://pan.baidu.com/s/1eRAuBS2

CHAPTER 10

常见问题

ConnetOS的特点是什么?

ConnetOS是基于开放网络理念构建的一个交换机操作系统，适用于现代数据中心网络架构。

ConnetOS以数据中心网络的运维痛点和运营效率出发，前所未有地开放网络资源和自身能力，提供以数据中心网络自动化和智能化为目标的一系列功能，让网络不仅开放而且易运维。

该操作系统的特色是:**「懂网络」**。

- 关注网络运维痛点

  针对网络故障诊断、业务超时、流量瞬发等网络运维痛点，提供如下使用功能:

  - 对转发平面所有数据包（转发的或丢弃的）进行监视和分析;
  - 实时捕获被设备丢弃的数据包，并记录丢包原因;
  - 根据五元组实时计算网络转发路径;
  - 具备端口拥塞感知和实时上报能力;
  - 高精度端口统计，最高精度1秒。

- 关注网络运营效率

  ConnetOS追求极致的网络效率，包括网络部署效率、网络监控效率、软件升级效率等。

  - 全自动部署功能，让网络部署做到真正完全自动化，无须任何人工干预，首次上线和整机替换时均实现自由上架和即插即用;
  - 基于PUSH模型和PUB/SUB机制的流式遥测（Streaming Telemetry）功能，让数据统计和导出更精准高效;
  - 基于warmboot技术的软件升级方式，业务透明无感知。

- 关注设备健康状态

ConnetOS内建立了一套自主研发的健康评估管理体系。体系抽象出网络设备运行过程中的几十项状态指标，实时进行信息搜集和分析，用于评估当前设备的健康状态（该状态信息既支持在设备上查看，也支持以订阅的方式进行远程导出）。

ConnetOS提供的健康状态评级机制可以供用户进行网络风险评估时的决策参考，避免因为软件潜在bug或者硬件缺陷，导致系统陷入“半死不活”的状态。

ConnetOS相比业界传统封闭OS有何优势?

ConnetOS在开放架构的基础上，结合数据中心网络的特点进行了诸多实践，提供了一系列自动化、智能分析诊断的服务和能力，最大化地将网络和芯片的能力开放给用户。

具体优势如下：

- 将业界一线互联网公司数据中心网络运营实践和经验进行沉淀和转化，提供一系列智能分析诊断功能，提升排障效率；
- 关注和跟进业界数据中心网络最新发展趋势和热点，打造下一代数据中心网络中的三平面立体监控技术；
- 提供原生Linux服务器上的体验，开放的技术架构和运行环境是DevOps实践的首选；
- 支持丰富的远程控制设备的方式，提供RCC、XRL、RESTful API、Python SDK等能力；
- 把握用户需求和痛点，攻克若干数据中心网络中固有问题和常见痛点，例如真正的全自动部署、静默丢包发现；
- 支持apt/dpkg软件包管理方式，支持第三方软件（Puppet、Zabbix、Ansible等）的直接部署；

ConnetOS相比业界其他开放OS有何优势?

业界目前其他的开放OS方案主要提供传统网络设备中已支持的一些通用特性，采用不断迭代的方式进行功能完善。ConnetOS相比这些OS，不仅提供基本的交换机业务功能，还立足网络的本质和用户的需求，提供满足从上线部署到后期运维全生命周期过程中的辅助功能。

具体优势如下：

- 业界唯一支持堆叠功能的开放OS方案；
- 提供先进的流式遥测功能，用于取代效率低下、功能有限的传统SNMP功能；
- 提供一系列网络分析诊断工具集，例如tcpdump/navmesh/sflow/sdrop（定位静默丢包问题）/smetric；
- 提供业界最高效且真正全自动化的网络零配置部署方案，比ZTP的自动化程度更高；
- 提供warmboot功能，确保软件升级过程中数据包转发不中断；
- 已有诸多大规模商用部署案例，稳定性有保障；

ConnetOS稳定性如何，有商用部署案例么?

ConnetOS完全具备在数据中心大规模生产环境下稳定运行的能力，目前已有客户包括百度、金山云、滴滴、美团云、小米、巨人网络、高升控股、云端智度等。搭载ConnetOS的设备在客户生产环境中稳定运行时间最长的已超1年。

ConnetOS能给用户带来什么价值?

ConnetOS让用户以更低的成本获得BAT级别的数据中心网络实践能力和运营经验。包括：

- 拥有一流互联网公司数据中心网络运维和实践经验转化而来的产品功能；
- 获得开放网络系统提供的传统基础功能之外的网络资源和服务能力；
- 解耦软件和硬件，提供白盒硬件选择上的更多灵活性；
- 同时降低企业的CAPEX和OPEX;

ConnetOS支持的交换芯片有哪些?

ConnetOS除了支持业界最流行的Broadcom芯片外，还支持Nephos的芯片。未来会有更多交换芯片支持。

ConnetOS开源么?

目前ConnetOS采用的是系统开放、应用开源，并可以根据客户需要进行策略性开源。

可以免费体验或试用ConnetOS么?

当然。

请发送邮件至tech@connetos.com，申请信息包括：姓名、公司、职位。我们会安排专人对接并支持资源。

CHAPTER 11

关于我们

公司介绍

云启科技是一家专业从事开放网络操作系统以及SDN控制器产品研发的公司，核心创始团队来自百度，对数据中心网络有着深刻的见解和丰富的实战经验。云启科技为公有云服务运营商和私有云大型行业客户，提供扩展性更强、成本更低、部署更简单的数据中心网络整体解决方案。

云启科技致力于打造网络设备领域的开放式业务生态系统，构建具有完全中国自主知识产权的超大型数据中心整体解决方案。云启科技倡导的开放模式，正在改变网络设备领域的传统封闭格局：云启科技通过与行业上下游合作伙伴建立战略合作关系，完善和规范交换机硬件标准和方案，给用户带来前所未有的硬件选择灵活性。

云启的愿景：让网络运营变得更简单。 **云启的使命**：打造开放而智能的网络操作系统。

联系我们

电话：（+86）10 - 5940 3172

传真：（+86）10 - 5940 3173

信息获取：info@connetos.com

技术支持：tech@connetos.com

公司网站：http://yunqi.ai

公司地址：北京市海淀区中关村软件园云基地C座一楼120室

欢迎关注

关注公众号，获取业界前沿的深度报道。

访问官网，了解更多ConnetOS更多信息。