Sun™ N2000 Series Release 2.0— Command Reference

Sun Microsystems, Inc.
www.sun.com
Part No. 817-7636-12
November 2004, Revision A

Submit comments about this document at: `http://www.sun.com/hwdocs/feedback`

Please
Recycle

Adobe PostScript

Regulatory Compliance Statements

Your Sun product is marked to indicate its compliance class:

- Federal Communications Commission (FCC) — USA
- Industry Canada Equipment Standard for Digital Equipment (ICES-003) — Canada
- Voluntary Control Council for Interference (VCCI) — Japan
- Bureau of Standards Metrology and Inspection (BSMI) — Taiwan

Please read the appropriate section that corresponds to the marking on your Sun product before attempting to install the product.

FCC Class A Notice

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received, including interference that may cause undesired operation.

Note: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy, and if it is not installed and used in accordance with the instruction manual, it may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case the user will be required to correct the interference at his own expense.

Modifications: Any modifications made to this device that are not approved by Sun Microsystems, Inc. may void the authority granted to the user by the FCC to operate this equipment.

FCC Class B Notice

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

1. This device may not cause harmful interference.
2. This device must accept any interference received, including interference that may cause undesired operation.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/television technician for help.

Modifications: Any modifications made to this device that are not approved by Sun Microsystems, Inc. may void the authority granted to the user by the FCC to operate this equipment.

ICES-003 Class A Notice - Avis NMB-003, Classe A

This Class A digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe A est conforme à la norme NMB-003 du Canada.

ICES-003 Class B Notice - Avis NMB-003, Classe B

This Class B digital apparatus complies with Canadian ICES-003.

Cet appareil numérique de la classe B est conforme à la norme NMB-003 du Canada.

VCCI 基準について

クラス A VCCI 基準について

クラス A VCCI の表示があるワークステーションおよびオプション製品は、クラス A 情報技術装置です。これらの製品には、下記の項目が該当します。

この装置は、情報処理装置等電波障害自主規制協議会 (VCCI) の基準に基づくクラス A 情報技術装置です。この装置を家庭環境で使用すると電波妨害を引き起こすことがあります。この場合には使用者が適切な対策を講ずるよう要求されることがあります。

クラス B VCCI 基準について

クラス B VCCI の表示  があるワークステーションおよびオプション製品は、クラス B 情報技術装置です。これらの製品には、下記の項目が該当します。

この装置は、情報処理装置等電波障害自主規制協議会 (VCCI) の基準に基づくクラス B 情報技術装置です。この装置は、家庭環境で使用することを目的としていますが、この装置がラジオやテレビジョン受信機に近接して使用されると、受信障害を引き起こすことがあります。取扱説明書に従って正しい取り扱いをしてください。

BSMI Class A Notice

The following statement is applicable to products shipped to Taiwan and marked as Class A on the product compliance label.

警告使用者：
這是甲類的資訊產品，在居住的環境中使用時，可能會造成射頻 干擾，在這種情況下，使用者 會被要求採取某些適當的對策。

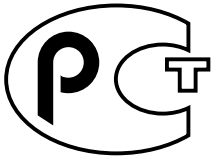
CCC Class A Notice

The following statement is applicable to products shipped to China and marked with “Class A” on the product’s compliance label.

以下声明适用于运往中国且其认证标志上注有 "Class A" 字样的产品。

声明
此为A级产品，在生活环境中，该产品可能会造成无线电干扰。
在这种情况下，可能需要用户 对其干扰采取切实可行的措施。

GOST-R Certification Mark



Contents

Chapter 2. Global commands

Chapter 5. Event commands

Chapter 6. SNMP and trap commands

Chapter 7. TFTP commands

Chapter 8. FTP client commands

Chapter 9. Telnet commands

Chapter 10. NTP and clock commands

Chapter 11. CLI and HTTP commands

Chapter 12. Software commands

Part III. Security

Chapter 13. User administration commands

Chapter 15. Certificate and Key Manager commands

Chapter 20. LAG commands

Chapter 21. VLAN commands

Chapter 22. IP interface commands

Chapter 32. Virtual Router Redundancy Protocol commands

Appendix A. Advanced CLI use

Appendix B. TCL usage

Appendix C. Object rule predicate statements

Appendix D. About authentication and authorization services

Command Index

Preface

About this manual

The *Nauticus N2000 Series – Command Reference* supports the Sun™ N2000 Series Release 2.0 hardware and software. The Sun N2000 Series system is an intelligent application switch that provides advanced Secure Sockets Layer (SSL) acceleration with reencryption and advanced Layer 4 to Layer 7 (L4 to L7) load balancing. The Sun N2000 Series system provides these services on a flexible, virtualized basis, within the convenience of a single enclosure, and with industry-leading speed, security, and availability. The N2000 Series comprises the N2040 switch and the N2120 switch. When it is necessary to differentiate between the two switches, the model numbers are used in this manual.

This manual may refer to the Sun N2000 Series system as the “N2000 Series,” the “application switch,” the “switch,” or the “system.”

This manual is intended for network and system administrators who are responsible for defining network parameters, including Ethernet, SNMP, Spanning Tree Protocol, load balancing, and connections/sessions. You should be familiar with Web and command-line interfaces to configure all features and functions of the Sun N2000 Series system.

What is in this manual?

This manual includes the following topics:

For information about:	See:
Using management interfaces	Chapter 1.
Global commands	Chapter 2.
Chassis commands	Chapter 3.
Remote monitoring commands	Chapter 4.
Event commands	Chapter 5.
SNMP and trap commands	Chapter 6.
TFTP commands	Chapter 7.
FTP client commands	Chapter 8.
Telnet commands	Chapter 9.
NTP and clock commands	Chapter 10.
CLI and HTTP commands	Chapter 11.
Software commands	Chapter 12.
User administration commands	Chapter 13.
SSH commands	Chapter 14.
Certificate and Key Manager commands	Chapter 15.
vSwitch and vRouter configuration commands	Chapter 16.
Resource commands	Chapter 17.
Ethernet management port commands	Chapter 18.
Ethernet data port commands	Chapter 19.
LAG commands	Chapter 20.
VLAN commands	Chapter 21.
IP interface commands	Chapter 22.
Virtual router interface commands	Chapter 23.
RIP and static routing commands	Chapter 24.
OSPF commands	Chapter 25.
ARP, ICMP, and IRDP commands	Chapter 26.
Ping and traceroute utility commands	Chapter 27.

For information about:	See:
ACL commands	Chapter 28.
Load balancing commands	Chapter 29.
TideRunner function card commands	Chapter 30.
Virtual Service Redundancy Protocol commands	Chapter 31.
Virtual Router Redundancy Protocol commands	Chapter 32.
Advanced CLI use	Appendix A.
TCL use	Appendix B.
Object rule predicate statements	Appendix C.
About authentication and authorization services	Appendix D.

Related documentation

For complete information about the Sun N2000 Series system, see the following documents.

Title	Document Number	Location
Sun N2000 Release 2.0 — Introduction Guide	817-7641-10	Documentation CD
Sun N2000 Series Release 2.0 — Quick Installation	817-7640-10	Printed, in ship kit
Sun N2000 Series Release 2.0 — Hardware Installation and Startup Guide	817-7638-10	Printed, in ship kit Documentation CD
Sun N2000 Series Release 2.0 — System Configuration Guide	817-7637-10	Documentation CD
Sun N2000 Series Release 2.0 — System Administration Guide	817-7635-10	Documentation CD
Sun N2000 Series Release 2.0 — Command Reference	817-7636-10	Documentation CD
Sun N2000 Series Release 2.0 — Release Notes	817-7639-10	Printed, in ship kit Documentation CD

Conventions

Typographical conventions

This manual uses the following typographical conventions:

Key Convention	Function	Example
Ctrl+*x*	Indicates a control key combination.	Press Ctrl+C
[*key name*]	Identifies the name of a key to press.	Type **`xyz`**, then press [Enter]
brackets [ ]	Indicates an optional argument.	`show protocol telnet sessions [ipAddress ipaddress]`

braces { }	Indicates a choice of argument values; choose one.
	
	Encloses an object rule predicate or a list within an object rule created with the CLI.
		`ckm import paste pairHalf {privateKey | certificate}`
		
		`objectRule rule1 predicate {URI_QUERY matches "information*"}`

vertical bar |	Separates parameter values. Means "or."	`format {pem | der | iis4 | pkcs12 | nauticus}`
Monospaced regular	Screen output, argument keywords, and defined argument values.	`protocol telnet adminState enabled`
Monospaced italic	Variable; generic text for which you supply a value.	`ntp id number`
Monospaced bold	User input.	`nauticus> show vSwitch`

CLI commands

Command-line interface (CLI) commands are not case sensitive. For example, `SWITCHSERVICES` is the same as `switchServices`. However, the text strings that you enter for argument values *are* case sensitive. For example, `ENGR` and `engr` represent two different values.

Data formats

Enter data in these formats unless the instructions say otherwise.

IP addresses

Use 4-byte dotted decimal notation, also called *dot address* or *dotted quad address* notation: `192.168.12.34`. You can omit leading 0s in a byte position.

Subnet masks and wildcard masks

Use 4-byte dotted decimal notation: `255.255.255.0` (1s in bit positions to match, 0s in bit positions to ignore). A *wildcard mask* is the reverse of a subnet mask: `0.0.0.255` (0s in bit positions to match, 1s in bit positions to ignore). You can omit leading 0s in a byte position.

In some functions, you might see a complete IP address and subnet mask in CIDR (Classless Interdomain Routing) notation: `192.168.12.34/24`. Here, the `/24` means that the first 24 bits of the address represent the network part of the address, and therefore the last 8 bits indicate the specific host on the network.

MAC addresses

Use 6-byte hexadecimal notation: `00:B0:D0:C9:99:1F.`

Text strings

Use alphanumeric characters, upper- and lower-case. Most text strings are case sensitive; for example, `Evan` and `evan` represent different user names.

Port numbers

Use `eth.1.x`, where `x` is an Ethernet port number from 1 through 44 on the N2040, and from 1 to 12 on the N2120.

Hexadecimal numbers

Use a `0x` prefix: `0x001732FF`.

Notes, cautions, warnings

Note: Pay special attention to the described feature or operation.

Caution: Damage to hardware, software, or data is possible.

Warning: Personal injury to yourself or others is possible.

Accessing Sun documentation

You can view, print, or purchase a broad selection of Sun documentation, including localized versions, at:

`http://www.sun.com/documentation`

Third-party Web sites

Sun is not responsible for the availability of third-party Web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Contacting Sun technical support

If you have technical questions about this product that are not answered in this document, go to:

`http://www.sun.com/service/contacting`

Sun welcomes your comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can submit your comments by going to:

`http://www.sun.com/hwdocs/feedback`

Please include the title and part number of your document with your feedback:

Sun N2000 Series Release 2.0 — Command Reference, part number 817-7636.

Abbreviations and acronyms

This manual contains the following industry-standard and product-specific abbreviations and acronyms:

AAA	authentication, authorization, and accounting
ACL	access control list
ARP	Address Resolution Protocol
BGP	Border Gateway Protocol
CA	certification authority
CAT	client address translation
CKM	Certificate and Key Manager
CLI	Command line interface
CSR	Certificate signing request
DER	Distinguished Encoding Rules format, ASN.1
DSA	Digital Signature Algorithm
DTE	data terminal equipment
ethMgmt.1	Ethernet management port on the N2000 Series

FQDN	Fully qualified domain name
GE	Gigabit Ethernet
HMAC	Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IIS4	Microsoft Internet Information Server (IIS)
IP	Internet Protocol
IRDP	Internet Router Discovery Protocol
ISP	Internet Service Provider
L2 ...L7	Layers in the OSI model that the N2000 Series supports
L4SLB	Layer 4 Server Load Balancing
L4SLB_SSL	Layer 4 Server Load Balancing with Secure Socket Layer
LAG	Link aggregation group
LB	Load balancer application on the N2000 Series
MD5	Message Digest 5
MIB	Management Information Base
N2000 Series	Nauticus N2000 Series Application Switch
N2040	N2000 Series model that provides 40 10/100 Mbps ports and 4 SFP Gigabit Ethernet ports
N2120	N2000 Series that provides 12 small form-factor pluggable (SFP) Gigabit Ethernet ports
NAT	Network Address Translation
NMON	Nauticus remote monitor
NTP	Network Time Protocol
OID	object identifier
OSPF	Open Shortest Path First
PEM	Privacy Enhanced Mail format
PKCS12	Public Key Cryptography Standard format
QoS	Quality of Service
RIP	Routing Information Protocol
SFF	Small Form Factor
SFP	Small Form Factor Plug

SFTP	Secure Shell File Transfer Protocol
SLB	server load balancing
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
STP	Spanning Tree Protocol
TACACS	Terminal Access Controller Access Control System
TCL	Tool command language
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
URL	uniform resource locator
USM	User Security Model (SNMPv3)
UTC	coordinated universal time
VIP	virtual IP address
VLAN	virtual LAN (Local Area Network)
VPN	virtual private network
VRRP	Virtual Router Redundancy Protocol
VSRP	Virtual Service Redundancy Protocol (Nauticus-proprietary)
vRouter	virtual router on the N2000 Series
vSwitch	virtual switch on the N2000 Series

Part I. Overview

The chapters in Part I describe how to use the management interfaces that are available for the N2000 Series systems and the global commands.

Chapter 1. Using the management interfaces

About this chapter

This chapter provides an overview of the management interfaces that the Sun™ N2000 Series application switch supports.

This chapter includes the following topics..

N2000 Series management interfaces

This section briefly describes each of the interfaces that the N2000 Series application switch supports for management and configuration. You can use any of the following interfaces:

- Command-line interface (CLI)
- Sun Application Switch Manager (HTTP interface)
- XML interface
- SNMP interface

Command-line interface

The command-line interface (CLI) is a flexible, text-based interface that allows you to manage and configure all aspects of the N2000 Series. You can access the CLI using a system console, a Telnet application, or Secure Shell (SSH). In addition, the CLI provides the following:

- An industry-standard command syntax, including `show` and `no` (delete) commands.
- Support for Tool Command Language (TCL) Version 8.3.3. The N2000 Series allows you to use TCL commands to program functions in the CLI. For details, see Appendix B, “TCL usage.”
- Support for command completion. You can type part of a command and press [Tab] to complete the command.
- Extensive online Help for commands, arguments, and command modes.

Sun Application Switch Manager

The Sun Application Switch Manager provides the Web interface for the Hypertext Transfer Protocol (HTTP)-based interface. As with the CLI, you can manage and configure all aspects of the N2000 Series with the Web interface. In addition, the Web interface provides:

- A graphing feature for statistical and operational data. You can generate pie charts, bar charts, and line charts.
- The ability to copy existing configuration entries and modify them to create new configuration entries more quickly.
- The ability to export configuration entries to eXtensible Markup Language (XML) files.
- Advanced configuration editors that help you complete complex tasks quickly.
- Detailed online Help describing all display fields and command arguments.

See the *Sun N2000 Series Release 2.0 – Introduction Guide* for details about the Web interface.

You can access the Web interface using Netscape™, Internet Explorer, or Opera Web browsers. See the product release notes for the supported Web browser releases.

SNMP interface

You can use any Simple Network Management Protocol (SNMP)-based network management application for fault and configuration management. The SNMP agent on the N2000 Series supports SNMP v1, v2c, and v3.

For detailed SNMP information about the SNMP commands, see Chapter 6, “SNMP and trap commands.” See the *Sun N2000 Series Release 2.0 – System Administration Guide* for instructions about configuring SNMP access.

Traps

The N2000 Series generates traps for SNMP authentication failures and system events that it logs in the system log file. In addition, the N2000 Series forwards traps defined in the standard MIBs that the switch supports. You can configure up to 10 remote trap destinations.

See the *Sun N2000 Series Release 2.0 – System Administration Guide* for details about configuring SNMP access and trap forwarding.

MIBs

The N2000 Series has a comprehensive set of standard and enterprise management information bases (MIBs) covering all aspects of the platform. If you plan on using SNMP to manage the N2000 Series, you can install these MIBs on a management station to make management tasks easier. To download the MIBs and installation instructions, go to:

`http://www.sun.com/service`

XML interface

The XML interface allows you to retrieve configuration information from the N2000 Series for use with other applications. The Web interface provides an XML export feature that allows you to obtain the XML data for configuration entries as well as the associated URL for the entries. You can use these URLs with any XML application to query the N2000 Series for specific configuration information and transform the XML data for use in other applications.

See the *Sun N2000 Series Release 2.0 – Introduction Guide* for details about the Web interface XML export feature.

Command-line interface overview

The N2000 Series command-line interface (CLI) has a multilevel structure that consists of command modes and executable commands for each command mode. Most command modes contain multiple levels of additional command modes.

Access modes

The CLI has access modes that determine which commands and command modes are available to you. Each access mode contains all of the privileges of the previous access modes as well as additional privileges. You can view all available commands with any privilege level, but you need the correct privilege level to create or modify configurations. Table 1-1 describes these access modes.

A user profile determines the access mode you enter when you log in to the CLI. See "CLI user profiles" on page 1-6 for a description of the user profiles.

Table 1-1. Access mode descriptions

Access mode	Description
User	The User access mode is the top-level access mode in the CLI hierarchy. From this mode, you can execute all `show` commands at the top command level, and `show` commands in lower-level command modes that you can access from this mode.
Enable	The Enable access mode provides access to all of the commands that do not affect forwarding of traffic. You can enter a context that requires configuration access, but you cannot modify or create records within that mode. **Accessing Enable mode:** From the User access mode, enter the `enable` command.
Configure (Config)	The Config access mode provides access to all of the command modes and commands that allow you to configure and manage all aspects of the system. **Accessing Config mode:** From the Enable access mode, enter the `config` command.

CLI user profiles

The N2000 Series includes predefined user profiles that determine which operations you can perform on the system and which access mode you enter when you log in to the CLI. The following table describes the profiles and the privileges associated with the profiles.

Table 1-2. CLI user profiles, privileges, and access modes

This profile:	Allows:	And places you in this access mode:
systemAdmin	Read and write access to all commands for the system, including all vSwitches. You can configure settings that affect the system and any vSwitch.	User
systemOperator	Read-only access to all commands for the system and all vSwitches. You cannot configure or delete any settings for the system or any vSwitch.	User
vSwitchAdmin	Read and write access to all commands that affect a specific vSwitch. You can configure settings for the specified vSwitch only.	User, in the specified vSwitch command mode
vSwitchOperator	Read-only access to all commands that affect a specific vSwitch. You cannot configure or delete settings for the specified vSwitch.	User, in the specified vSwitch command mode

Interpreting the CLI prompt

The CLI prompt always indicates where you are located in the CLI hierarchy. It also includes indicators for the access command mode in effect and the command mode (or modes) in which you are working. The following table describes the different CLI prompts.

Table 1-3. CLI prompts

This access mode:	Has this prompt:	Example
User	>	`sun (mode)>`
Enable	#	`sun (mode mode)#`
Config	(config) #	`sun (config mode)#`

Accessing the CLI

You can access the CLI using a system console (serial connection), Telnet, or Secure Shell (SSH). For complete instructions on installing and initially configuring the application switch and its network connections, see the *Sun N2000 Series Release 2.0 – Hardware Installation and Startup Guide*.

Using a system console

The following procedure describes how to access the CLI using a system console.

Step	Action
1.	Connect a PC to the N2000 Series. To do this, connect an EIA-232 (RS-232) straight-through serial cable between the Console port on the application switch front panel and the COM1 or serial port on the PC.
2.	Start a terminal emulator on the PC (such as Tera Term Pro or HyperTerminal).
3.	Configure the terminal emulator for: • VT100 emulation, or let it autodetect the terminal type • 9600 baud, 8 data bits, no parity bit, 1 stop bit (8/N/1), no flow control
4.	Respond to the prompts for username and password.
5.	Press [Enter] on the PC keyboard until you see the CLI User access mode prompt: `sun> _`

Using Telnet

The following procedure describes how to access the CLI using Telnet, the TCP/IP terminal emulation protocol.

Step	Action
1.	Connect the N2000 Series to a network that the Telnet client system can reach. To do this, connect a UTP/STP Category 5 network cable from the Management port on the application switch front panel to a network patch panel or device.
2.	Configure an IP address for the Management port according to instructions in the *Sun N2000 Series Release 2.0 – Hardware Installation and Startup Guide*.
3.	Start the Telnet client. At the Telnet prompt, enter `open<ipaddress>`, where `<ipaddress>` is the IP address of the Management port.

4.	Log in using the user name and password that the system administrator assigned to you.
	`username: user1`
	`password: <password>`
	`sun>`

Using Secure Shell

The following procedure describes how to access the CLI using SSH, a secure Telnet-like terminal emulator protocol. See "Supported SSH clients" on page 1-10 for a list of SSH clients that the N2000 Series supports.

Step	Action
1.	Connect the N2000 Series to a network that the SSH client system can reach. To do this, connect a UTP/STP Category 5 network cable from the Management port on the application switch front panel to a network patch panel or device.
2.	Configure an IP address for the Management port (ethMgmt.1) according to the instructions in the *Sun N2000 Series Release 2.0 – Hardware Installation and Startup Guide*.
3.	Does a Digital Signature Algorithm (DSA) key currently exist on the system? • If **yes**, go to Step 4. • If **no**, generate a DSA key for the system virtual switch using the Certificate and Key Manager (CKM) utility. Be certain to change the algorithm type to DSA when creating the key for SSH. **Example:** The following example generates a key pair with a length of 1024 bits, sets the algorithm to DSA, and assigns a key ID of 2. `sun(vswitch-system ckm)# generate keyId dsa-ssh-key 1024 dsa`
4.	Bind the key pair index with the system's server key ID. The system uses the index to identify the key pair to use during SSH communication. **Example:** The following example binds the key pair generated in Step 3 to the SSH daemon. `sun# switchServices sshd serverkeyid dsa-ssh-key`
5.	Configure the authentication method and the order of preference. The default authentication method is both `password` and `publickey`. **Example:** The following example sets the authentication method to use a public key first. If the system does not receive a valid public key from the client, it prompts the client for a password. `sun# switchServices sshd userauthentication publickey,password` **Note:** Specify the most restrictive authentication method as the first method to use, followed by less restrictive methods.

Step	Action

6.	If using public key authentication, add the client key to the N2000 Series database. After specifying the client IP address and user name, the system prompts you to enter the client key data. To do this, copy the key from the client system (where you generate the key) and paste it into the window where you are logged in to the CLI or in the Key Data field in the Web interface.
	
	Example: The following example adds a public key for user JohnDoe to the system database.
	`sun# switchServices sshd clientKey 1.1.1.1 JohnDoe active`
	`Please enter your data ctrl-z to accept ctrl-c to cancel:`
	
	`-----BEGIN DSA PRIVATE KEY----`
	
	`MIIBuwIBAAKBgQDdJKhnTqMg0QjI/RhQJIwtsVxO70JENbG`

7.	Start the SSH client and specify the host name or address for the system. **Result:** If using the password authentication method, the system prompts you for a password. Otherwise, if using a valid public key or no authentication is required, you connect to the system.

8.	Log in using the user name and password that the system administrator assigned to you:.
	`username: secureuser1`
	`password: <password>`
	`sun>_`

Supported SSH clients

The N2000 Series supports the following SSH clients:

- SecureCRT, for Microsoft Windows platforms
- PuTTY, for Microsoft Windows platform
- OpenSSH, for UNIX and Linux platforms

Other SSH clients may work; however, Sun has not tested other clients with the N2000 Series at this time.

Entering CLI commands

When you enter a command, the CLI checks whether a configuration entry with the same argument values already exists. If an entry does not exist, the CLI creates the entry. If the entry does exist, the CLI modifies the entry.

Moving through the CLI hierarchy

To navigate through the CLI hierarchy, you move from one command mode to another. To move to a command mode, you enter one or more commands that indicate which mode you want to use. In some cases, you need to specify arguments with the command for a command mode.

The following table describes the different methods you can use to move through the CLI hierarchy.

Table 1-4. CLI navigation methods

To do this:	Enter this:

Move down one command mode level at a time.	Enter a single command and command argument (if required) and press [Enter]. Repeat this step until you are in the desired mode.
	
	Example: In this example, you move down the CLI hierarchy to reach the `keypair` command mode in the system vSwitch command mode.
	`sun# **vSwitch system**`
	`sun(vSwitch-system)# **ckm**`
	`sun(vSwitch-system ckm)# **keypair**`
	`sun(vSwitch-system ckm keypair)#_`

Move down multiple command levels at one time.	Enter the appropriate commands and arguments (if required) for each mode and press [Enter]. You must enter the commands in the actual order of the CLI hierarchy.
	
	Example: In this example, you enter multiple commands to move from the top level of the CLI hierarchy down three levels to the `certificates` command mode.
	`sun# **vSwitch system ckm keypair**`
	`sun(vSwitch-system ckm keypair)#_`

Table 1-4. CLI navigation methods (continued)

To do this:	Enter this:

Move up one level in the CLI or exit the CLI from the top level of the CLI hierarchy.	Enter the `exit` command and press [Enter].
	
	Example: In this example, you use the `exit` command to move one level above the `certificates` command mode.
	`sun(vSwitch-system ckm keypair)# exit`
	`sun(vSwitch-system ckm)#_`

Move to the top level of the CLI hierarchy.	Enter the `end` command and press [Enter].
	
	Example: In this example, you enter the `end` command to move up three levels to reach the top level of the CLI hierarchy.
	`sun(vSwitch-system ckm keypair)# end`
	`sun# _`

CLI command syntax

The syntax for a command can consist of the following:

- A *command mode path* that indicates where in the CLI hierarchy a specific executable command is located, if required. If you are already at the command mode level for an executable command, you do not need to enter the command mode path.
- An *executable command* that creates or modifies the system settings.
- *Arguments* that set characteristics for a command. Commands can have optional and required arguments, optional arguments only, or no arguments. Any argument and value that are enclosed in brackets ([ ])are optional.
- *Argument values* that set characteristics for each argument. The `show` command has additional arguments called *filters*. For details about using filters, see Appendix A, “Advanced CLI use.”
- *Predicate operators*. In object rule predicate statements, operators determine how text strings and integers perform. For example, a question mark (?) and an asterisk (*) wildcard operate with the `matches` predicate operator only. Predicate statements are enclosed in braces ({}) and text strings are enclosed in double quotes (“ “). See Appendix A, “Advanced CLI use,” for more information about CLI usage and Appendix C, “Object rule predicate statements,” for more information about predicate operators.

Case sensitivity

CLI commands are not case sensitive. For example, `SWITCHSERVICES` is the same as `switchServices`. However, the text strings that you enter for argument values *are* case sensitive. For example, `ENGR` and `engr` represent two different values.

Entering executable commands

The following table describes the basic methods for entering executable commands. See Appendix A, "Advanced CLI use," for additional methods for entering commands..

Table 1-5. Command execution methods

Method	Example

Do the following:
 	
1. Enter a command for a mode and press [Enter]. Continue doing this until you reach the appropriate level for the command you want to use.
2. Enter the executable command and its arguments.
 	
Result: In each step, the CLI prompt displays the path of your current location in the CLI hierarchy.
	`sun# switchServices`
	`sun(switchServices)# sshd`
	`sun (switchServices sshd)# adminState enabled`
	`sun(switchServices sshd)#_`

Do the following: • Enter the complete command mode path, commands, and arguments from the top level of the CLI hierarchy and press [Enter]. With this method, you do not need to be in a specific command mode to enter a command. **Result:** After executing a command, you remain in the command mode that you were in before you entered the command.	`sun# switchServices sshd adminState enabled` `sun#_`

Global commands

The CLI includes commands that you can enter from any command mode. See Chapter 2, "Global commands" for a complete description of each command.

CLI editing functions

The CLI includes several editing functions to help you use the CLI. You can do the following:

- Use accelerator keys to edit command lines quickly.
- Customize the number of rows the system displays in the CLI window.
- Select commands from a list of unique commands that you previously entered and reexecute them.

Accelerator keys for navigation and editing

An accelerator key is a keyboard key or combination of keys that allow you to perform a commonly used function quickly. The CLI includes a comprehensive set of accelerator keys that help you move through a command line as well as edit it. These keys also work with command lines that span multiple lines.

The following table lists the valid accelerator keys.

Table 1-6. CLI cursor accelerator keys

Keys	Function
Ctrl-A	Move cursor to beginning of current command line.
Ctrl-B or ←	Move cursor backward one character position.
Ctrl-C	Clear current command line.
Ctrl-D	Delete character at current cursor position.
Ctrl-E	Move cursor to end of current command line.
Ctrl-F or →	Move cursor forward one character position.
Ctrl-H or Backspace	Delete one character to the left of cursor.
Ctrl-K	Delete all characters from current cursor position to the end of current command line.

Table 1-6. CLI cursor accelerator keys (continued)

Keys	Function
Ctrl-L	Clear the screen; that is, position current command line at the top of the terminal window (but do not erase the scroll buffer).
Ctrl-N or ↓	Display the next command in the history list. (This action is meaningful only if you have moved up the history list using **Ctrl-P** or ↑.)
Ctrl-P or ↑	Redisplay previous command line, but do not execute it.
Ctrl-U	Clear current command line.
Ctrl-V	Hide sensitive information being typed, such as passwords.
Ctrl-Y	Paste line or characters deleted with Ctrl-K.
Esc-B	Move cursor back one word (string separated by space or period) to the first character of the word.
Esc-D	Delete from cursor to the end of current word (string separated by space or period).
Esc-F	Move cursor forward one word (string separated by space or period).

Using the CLI Help

The CLI provides several kinds of Help at the command line.

Displaying a list of commands

Enter [?] at the CLI prompt to display the executable commands and command modes available for a given mode, plus a brief description of each.

For example:

```
GLOBAL COMMANDS                 (can be run from any access mode)
      cliLog                    Turn on|off the log for CLI output.
      commandModeEntry          Allow/Disallow the entry of a contextual
                                command mode on creation or
                                modification.
      config                    Enter configuration mode
      enable                    Turn on privileged commands
      end                       Go to the top-level directory
      exit                      Exit the session or terminate a
                                mode
      getField                  Return the field values of rows for a
                                given query.
      getKey                    Return a TCL list of keys
      getRow                    Return a TCL list of values
      getRowCount               Return the number of rows for a given
                                query.
      history                   Show the history of commands for this
                                CLI session
      import                    Import running configuration from a file
      install                   Update software
      interactive               Show/Set the interactive mode for this
                                CLI session
      mode                      Return the current session user mode
      monitor                   Set the monitoring on a given field.
      no                        Negate a command or set its defaults
      quit                      Quit the session or terminate a
                                mode
      redo                      Execute a previous command
      rows                      Show/Set the rows for this CLI session
      saveCfg                   Save configuration to flash.
      show                      Show running system information
  EXECUTABLE COMMANDS
      ethMgmt                   Configure characteristics of the
                                Ethernet management port
      event                     Configure the system to store events
                                in the event log
      lag                       Create new LAG configuration on N2000
```

```
      nmon                      Change the behaviour of the
                                configuration nmon
      port                      Configure characteristics of the
                                specified Ethernet port
      vSwitch                   Create a new vSwitch
  COMMAND MODES
      event                     Enter event command mode.
      lag                       Enter lag command mode.
      nmon                      Enter nmon command mode.
      port                      Enter port command mode.
      switchServices            Enter switchServices command mode
      vSwitch                   Enter vSwitch command mode.
sun(config)#
```

Displaying full syntax and arguments for a command

Press [Spacebar] and press [Enter] or [?] after you enter a command to display the full syntax and all arguments for a command, as well as short command descriptions.

If you use [?], the CLI rewrites the command name after displaying the Help output.

For example:

```
sun(config-event)# syslog [Enter]
  host <IP Address>           IP address of a syslog host
  [port (1..65535)]           UDP port that a syslog host uses for
                              events (default: 514)
  [filter <NamedIndex>]       Name of filter for events to this syslog
                              (default: defaultSyslog)
  [facility (local0|local1|local2|local3...)]  Facility code (default:
                              local0)

sun (config-event)# syslog ?
  host <IP Address>           IP address of a syslog host
  [port (1..65535)]           UDP port that a syslog host uses for
                              events (default: 514)
  [filter <NamedIndex>]       Name of filter for events to this syslog
                              (default: defaultSyslog)
  [facility (local0|local1|local2|local3...)]  Facility code (default:
                              local0)
```

Displaying valid values for an argument

Press [Tab] after entering an argument to view a compact display of all valid values.

For example:

```
sun(config-vSwitch-e-commerce)# vrouter default [Tab]
    mode       debug      interfaces ip         irdp       ospf
    ping       rip        traceroute vlan       vrrp
```

Using command completion

Enter the minimum unique characters of a command or argument and press [Tab]. The CLI completes the command or argument for you. Command completion is available only in the command mode for a specific command.

The following table shows examples of how command completion works.

Table 1-7. Command and argument completion examples

This command:	Becomes:
`sun(config-event)# fi[Tab]`	`sun(config-event)# filterProfile`
`sun(config-event)# filterProfile[Tab]`	`sun(config-event)# filterProfile default`

If more than one command or argument matches the character string that you enter, the CLI displays all items that match the character string. The following example shows how command completion works in this situation:

```
sun(config-event)# s[Tab]
    show   syslog
sun(config-event)# sh[Tab]
sun (config-event)# show
```

Working with CLI configuration entries

When you use the CLI commands to configure the N2000 Series, you add or modify entries in the current system configuration in volatile memory. You can also use the CLI to save the entries to flash memory (non-volatile memory), delete entries, or view entries.

Modifying configuration entries

When modifying a configuration, you use the same commands that you used when you built the configuration; however, all of the arguments used during an add operation may not be available during a modify operation. To determine the available arguments you can change, view the online Help for the command.

If you want to clear a value in a configuration entry during a modify operation, enter double quotes ("") as the argument value. The system interprets the double quotes as a null value.

Example 1

In this example, displaying Help for the vSwitch command for an existing vSwitch shows which arguments you can use to modify the configuration entry. During a modify operation, you cannot change the vSwitch name, but you can modify all other values.

```
sun> enable
sun# vSwitch e-commerce ?
[description <text>]              Description of the vSwitch
[adminState (enabled|disabled)]   Administrative state of the vSwitch
                                 (default:enabled)
```

Example 2

In this example, specifying double quotes for the vSwitch description in a modify operation removes the description value.

```
sun> enable
sun# show vSwitch e-commerce
Name:               e-commerce
ID:                 1
Description:        Ecommerce vSwitch   <-- Original description value
Admin State:        enabled
Operational Status: up
```

```
sun# vSwitch e-commerce description ""

sun# show vSwitch e-commerce
Name:                e-commerce
ID:                  1
Description:
Admin State:         enabled
Operational Status:  up

sun#
```

Description value after modify operation

Saving configuration entries

Until you save configurations, any changes you make apply to the running configuration only. If you exit the CLI, the running configuration is still in effect. However, if you reboot the system, the running configuration is lost.

Use the `saveCfg` command to save the current configuration entries to non-volatile flash memory (this command has no arguments). The N2000 Series reads the configuration file in flash memory the next time it boots.

Note: Using the `saveCfg` command saves all configuration changes from a CLI session, a Web interface session, or from SNMP operations that are stored in the current running configuration.

Deleting configuration entries

To delete a configuration entry, use the `no` command followed by an executable command and its appropriate arguments. You must enter any required arguments with the command to identify a specific entry. You can also use a wildcard symbol to delete multiple entries of a specific type. See Appendix A, "Advanced CLI use," for details.

Caution: Use this command carefully. There is no undo feature and, for some commands, there is no confirmation feature. In these situations, the `no` command immediately deletes the entry or entries.

Example

In this example, the no command deletes the configuration entry for the NTP 3 server.

```
sun(switchServices)# show ntp 3
Server ID:          3
Server IP address:  1.1.1.1
Preferred server:   false
Burst mode:         false
Min. poll interval: 256
Max. poll interval: 1024
NTP version:        4

sun(switchServices)# no ntp 3
sun(switchServices)# _

sun(switchServices)# show ntp 3
There are no entries matching your query.
sun(switchServices)# _
```

Displaying configuration entries

To display configuration entries, enter the show command for the entry whose configuration you want to view. To view a listing of the show commands that are available in a command mode, navigate to the command mode and enter:

```
sun# show ?
```

You can also use the asterisk (*) wildcard when entering filter values. See Appendix A, “Advanced CLI use,” for details.

Example

This example displays how to use the show command from the top level of the CLI hierarchy to view the configuration entries for the NTP servers.

```
sun# show switchServices ntp server 3
Server ID:          3
Server IP address:  1.1.1.1
Preferred server:   false
Burst mode:         false
Min. poll interval: 256
Max. poll interval: 1024
NTP version:        4
```

Common CLI errors

The CLI displays error messages if you enter invalid commands, arguments, or argument values. This section describes the most commonly seen errors and what action to take if you receive an error message.

Unrecognized command

The system displays this error message if you type a command that is not part of the CLI or if you enter a command that is not available in the current command mode.

```
sun# frob
ERROR: "frob" is not a recognized command
```

Solutions

Try any of the following:

- Enter [?] to display a list of valid commands and command modes.
- Enter a text string and press [Tab] to display all commands beginning with the string.
- Enter the complete command mode path for the command you want to use.
- Check that you typed the command correctly.

Invalid values

The system displays this type of error message if you enter an argument value that is not valid for a specific argument.

```
sun(switchServices)# ntp server 3 172.26.3.11 minpoll 500
ERROR: "500" is an invalid value for field "minpoll"
```

Solutions

Try any of the following:.

- Enter [Tab] or [?] to display a list of valid values. Correct a typing error.
- Check that you typed the command correctly.

Illegal actions or incomplete commands

The system displays this type of error message if you attempt an illegal action or do not enter a complete command.

You receive the following error message if you enter a `config` command when you are not in Enable access mode:

```
sun> configure
ERROR: Users must be enabled to access the configure mode
```

You receive the following error message if you enter an incomplete command:

```
sun(switchServices)# ntp address 172.26.3.10
ERROR: Either specify all keys, or set at least one key to "*".
```

Solutions

Do the following:

- Enter the `enable` command. You must be in the Enable mode before you can enter Configure mode.
- Enter values for all required arguments. Use the CLI Help to determine which arguments are required.

Chapter 2. Global commands

Global command description

The CLI includes commands that you can enter from any command mode. These commands are listed from the top level of the CLI hierarchy. They are not listed from other levels, even though they are available. Note that command completion for these global commands is not in effect from any level but the top level of the hierarchy.

Global command path

The commands in this chapter can be entered from any point in the CLI hierarchy.

Global command summary

Table 2-1 lists and briefly describes the global commands.

Table 2-1. Global command summary

Command name	Description
`cliLog`	Turn on or off CLI logging.
`commandModeEntry`	Allow or disallow the entry of a contextual command mode on creation or modification.
`configure`	Enter Configure access mode.
`enable`	Enter Enable access mode.
`end`	Move to the top level of the CLI hierarchy.
`exit`	Move up one level in the CLI hierarchy.
`getField`	Return the value of the requested field.
`getKey`	Return the identifier of a table row.
`getRow`	Return the table row matching the query.
`getRowCount`	Return the number of entries matching the query.
`history`	Display a list of commands executed this session.
`import runningConfig`	Import a running configuration file.
`install`	Install an N2000 Series software version from a specified directory path.
`interactive`	Toggle system prompting.
`mode`	Display the current command entry mode.
`monitor`	Monitor N2000 Series statistics and counters.
`no`	Delete configuration data.
`quit`	Move up one level in the CLI hierarchy.
`redo`	Reexecute a command.
`rows`	Set the number of rows displayed.
`saveCfg`	Save the running configuration file to flash memory.
`show`	Display configuration data.

Table 2-1. Global command summary (continued)

Command name	Description
`show redundantConfig`	Display the parts of the running configuration used to synchronize a redundant pair.
`show runningConfig`	Display the running configuration file.

cliLog

Purpose

Saves all input from the CLI to a log file. Logging is disabled by default. To enable logging, set the command to on and specify a file name. You must also specify a file size. When the file reaches the configured size, it closes automatically. If you reinitiate logging with the same file name, all previous logging data is lost. To save the data, specify a new file name and file size.

Access mode

enable

Syntax

```
cliLog
   [on filename]
   [off]
   size integer
```

Arguments

Argument	Description
on *filename*	**Optional.** Enables logging of CLI output to a specified file.
off	**Optional.** Disables CLI output logging.
size *integer*	Sets the maximum size for the log file, in kilobytes. The file size can be from 1 to 1024 kilobytes; the default size is 40 kilobytes.

Example

The following example turns logging on for file capture, sets the log file size to 100 Kbytes, and then disables logging again.

```
sun(config)# cliLog on clioutput.txt
sun(config)# cliLog size 100

sun(config)# cliLog off
Turning CLI logging off.
sun(config)#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

commandModeEntry

Purpose

Allows or disallows the entry of a contextual command mode when creating or modifying the configuration.

If specified without any arguments, the CLI returns the current command mode entry setting, either `on` or `off`.

Access mode

user

Syntax

```
commandModeEntry
  {on |off}
```

Arguments

Argument	Description
on	Enables command mode entry.
off	Disables command mode entry.

Example

```
sun(config)# commandModeEntry
Command mode entry: OFF
sun(config)# commandModeEntry on
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

configure

Purpose

Enters the Configure mode. You must be in Enable mode before you can enter Configure mode. The Configure mode provides access to all of the command modes and commands that configure and manage all aspects of the system.

You are required to be in Configure mode to execute all commands that impact the flow of traffic to and from the system.

Access mode

enable

Syntax

```
configure
```

Example

The following example illustrates entering Configure mode immediately after login.

```
sun> enable
sun# configure
sun(config)#
```

The following example illustrates entering Configure mode after completing a task that does not require the top-level access. In addition, it displays the difference in commands available from User and Configure modes.

```
sun> ?

GLOBAL COMMANDS              (can be run from anywhere)
      show                     Show running system information
EXECUTABLE COMMANDS
      clock
      tftp                     Transfers files using TFTP
```

```
COMMAND MODES
      chassis                 chassis command mode
      resource                resource command mode
      software                software command mode
      tideRunner              tideRunner command mode
      trap                    trap command mode
sun> switchServices
sun(switchServices)> clock timeZoneOffset -5.00
sun(switchServices)> enable
sun(switchServices)# configure
sun(config-switchServices)# ?

  GLOBAL COMMANDS             (can be run from anywhere)
      show                    Show running system information
  EXECUTABLE COMMANDS
      cli                   Modify the CLI current session and default
                              configuration
      clock
      httpd                   Modify the HTTP configuration
      ntp                     Modify the NTP configuration variables
      reset                   Reset the N2000
      snmp                    Modify the SNMP access configuration
      sshd                    Configure the main parameters of sshd
      telnetd                 Configure the main parameters of the
                              Telnet protocol
      tftp                    Transfers files using TFTP
      tftpd                   Modifies the TFTPd configuration
      userAdministration      Configures or modifies system-wide AAA
                              attributes
  COMMAND MODES
      chassis                 chassis command mode
      ntp                     enter ntp command mode
      resource                resource command mode
      snmp                    enter snmp command mode
      software                software command mode
      sshd                    enter sshd command mode
      telnetd                 enter telnetd command mode
      tftpd                   enter tftpd command mode
      tideRunner              tideRunner command mode
      trap                    trap command mode
      userAdministration      enter userAdministration command mode
```

Associated MIB

None

Web path

When you log in to the Web interface, the system automatically places you in the Configure mode. If you log in as a systemOperator or vSwitchOperator user, you cannot change the configuration. You must log in as a systemAdmin or vSwitchAdmin user to change the configuration.

enable

Purpose

Enters the Enable mode. When you are in the Enable mode, you can access all commands that do not affect traffic flow to and from the system.

Access mode

user

Syntax

```
enable
```

Example

The following example illustrates a command mode that requires the additional privileges of Enable mode.

```
sun> switchServices
sun(switchServices)> sshd
ERROR: command not found!

sun(switchServices)> enable
sun(switchServices)# sshd
sun(switchServices sshd)#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

end

Purpose

Exits you from your current position in the CLI hierarchy, and moves you to the top level. In addition, if you were previously in Configure mode, the end command moves you to Enable mode.

Access mode

user

Syntax

```
end
```

Example

The following example illustrates the action of the end command.

```
sun# configure
sun(config)# vswitch system vrouter management
sun(config-vSwitch-system vRouter-management)# ip
sun(config-vSwitch-system vRouter-management ip)# end
sun#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

exit

Purpose

The `exit` command does the following, depending on where you are in the CLI hierarchy:

- If entered at a command mode, moves up to the previous command mode level.
- If entered at the top of the command hierarchy, moves up to the previous access mode level.
- If entered at the top-level access mode, exits the CLI.

Depending on the user profile definition, the top-level access mode is either User or Enable. See "CLI user profiles" on page 1-6 for details.

This command has the same functionality as the `quit` command.

Access mode

user

Syntax

```
exit
```

Example

The following example illustrates all three uses of the `exit` command.

```
sun(config-vSwitch-system vRouter-management)# exit
sun(config-vSwitch-system)# exit
sun(config)# exit
sun# exit
sun> exit
Goodbye
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

getField

Purpose

Returns the value of the requested field from the system's configuration tables in the form of a TCL variable.

Access mode

enable

Syntax

```
getField
   commandPath
```

Arguments

Argument	Description
commandPath	Specifies the name of the field for which you want a value returned. By specifying more components at the command line, you further filter your resulting output. Enter a command and field, but not the field value.

Example

The following example displays the operational status for a specific vSwitch and for all vSwitches on the system.

```
sun# getField vswitch e-commerce operationalStatus
up
sun# getField vswitch operationalStatus
 {up up }
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

getKey

Purpose

Displays a list of index values (identifiers) from the system's configuration tables for the specified configuration component. You can use this command to obtain information for TCL scripts.

Access mode

enable

Syntax

```
getKey
   commandPath
```

Arguments

Argument	Description
commandPath	Specifies the name of the field for which you want a value returned. By specifying more components at the command line, you further filter your resulting output. Enter a command and optionally, the field, or field *and* field value.

Example

The following examples return a list of vSwitches on the system and the ports matching the filter criteria.

```
sun# getKey vswitch
 {vSwitchName {e-commerce} } {vSwitchName {system} }
sun# getKey port ifIndex eth.1.4*
 {ifIndex {eth.1.4} } {ifIndex {eth.1.40} } {ifIndex {eth.1.41} }
{ifIndex {eth1.42} } {ifIndex {eth.1.43} } {ifIndex {eth.1.44} }
sun#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

getRow

Purpose

Displays the current values for the specified configuration (similar to the output for a `show` command). The command returns output for rows from within the configuration table that matches your entire query. You can use this command to obtain information for TCL scripts.

Access mode

enable

Syntax

```
getRow
   commandPath
```

Arguments

Argument	Description
commandPath	Specifies the name of the field for which you want a value returned. By specifying more components at the command line, you further filter your resulting output. Enter a command and field, but not the field value.

Example

The following example displays the configuration for all configured vSwitches and a specific vSwitch.

```
sun# getRow vswitch
 {{vSwitchName {e-commerce}} {vSwitchId {1}} {description {E-commerce
vSwitch}}{adminState {enabled}} {operationalStatus {up}} }
{{vSwitchName {system}} {vSwitchId {0}} {description {System vSwitch}}
{adminState {enabled}} {operationalStatus {up}} }
sun# getRow vswitch e-commerce
{{vSwitchName {e-commerce}} {vSwitchId {1}} {description {E-commerce
vSwitch}} {adminState {enabled}} {operationalStatus {up}} }
sun#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

getRowCount

Purpose

Returns an integer indicating the number of entries matching the query.

Access mode

enable

Syntax

```
getRowCount
   commandPath
```

Arguments

Argument	Description
commandPath	Specifies the name of the field for which you want a value returned. By specifying more components at the command line, you further filter your resulting output. Enter a command, and optionally, field *and* field value.

Example

The following example illustrates the command both with and without the field value specified.

```
sun# getRowCount port portType gigEthernet
4
sun# getRowCount port
44
sun#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

history

Purpose

Displays a numbered list of the command history for this CLI session. The CLI keeps a history buffer of the last 50 commands you have entered, regardless of whether they were valid commands.

The history list starts with each CLI session. You cannot clear or save the history list.

Access mode

user

Syntax

```
history
```

Example

The following example displays the history list.

```
sun# history
    1  enable
    2  switchServices
    3  cli
    4  cli rows 48 autologouttimeout 60 defaultprompt sun
    5  exit
    6  history
sun#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

import runningConfig

Purpose

Imports the specified running configuration file. During the import operation, the system runs the commands in the running configuration file. The system creates any new entries, and modifies existing entries. It does not delete any configuration. If the system encounters configuration problems, such as too many vSwitches, it reports an error and stops the import process. Changes made up to that point are written to the new running configuration. Until you issue the `saveCfg` command, the changes are in the running configuration only.

You must log in as a systemAdmin user to import running configuration files. See the *Sun N2000 Series Release 2.0 – System Administration Guide* for details about user access.

Access mode

user

Syntax

```
import runningConfig
   fromFile fileName
   [password passwordText]
```

Arguments

Argument	Description
`fromFile` *`fileName`*	Specifies the name of the file to import. This is the file name you specified with the `show runningConfig saveToFile` command. The system looks for the file in the `/ftl0/dist` directory, unless you specify a path.
`password` *`passwordText`*	**Optional.** Specifies the password assigned to the file on export (with the `show runningConfig password` command). If you did not specify a password on export, any data that requires a password (for example, other password data and CKM information) is not included in the file.

Example

The following example imports a running configuration.

```
sun> pwd
/ftl0/usr/home/
sun> cd ../../
sun> import runningConfig fromFile runningconfig.txt password ****
sun>
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

install

Purpose

Installs the specified version of N2000 Series software from the specified directory path. Optionally, you can specify up to five prior software versions to remain on the system (if previously installed).

Access mode

config

Syntax

```
install
   filename path
   [{2|3|4|5} integer]
```

Arguments

Argument	Description
`filename` *`path`*	Specifies the fully qualified path name to the compound image installation file. **Example:** `/ftl0/dist/V2_0.nci`
`{2|3|4|5}` *`integer`*	**Optional.** Specifies the number of prior software releases to remain on the system when the installation process is completed. If no number is specified, no prior releases are kept on the system.

Example

The following example installs N2000 Series software V2_0R1.

```
sun> enable
sun# config
sun(config)# install /ftl0/dist/V2_0R1.nci
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

interactive

Purpose

Enables or disables system prompting. Prompting is enabled by default, but can be disabled for scripting. If you disable system prompts, and you execute a command that requires input in response to a prompt, the command fails. For example, if you try to import a CKM certificate that was exported with a password, the system cannot prompt for the password and the import fails. In situations that would prompt for a yes or no answer, the system behaves as if you had answered yes. For example, if you try to delete a vSwitch, instead of the usual confirmation prompt, the vSwitch is just deleted.

Access mode

user

Syntax

```
interactive
   {on | off}
```

Arguments

Argument	Description
on	Enables system prompting.
off	Disables system prompting. Actions that would otherwise result in a confirmation prompt are treated as if a “yes” were entered in response. Actions that require input as the result of a prompt fail.

Example

The following example disables system prompting.

```
sun> interactive off
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

monitor

Purpose

Monitors the specified N2000 Series system statistics using alarms that generate events when thresholds are crossed. The `monitor` command operates with the remote network monitor (NMON) that performs alarm polling and reports poll results.

Refer to Chapter 4, "Remote monitoring commands" for information about NMON alarm settings. Refer to the *Sun N2000 Series Release 2.0 – System Administration Guide*, for information about using the `monitor` command and NMON.

Access mode

enable

Syntax

```
monitor
   monitorCommandPath
```

Arguments

Argument	Description
monitorCommandPath	Specifies the path to the object to be monitored.

Example

```
sun# monitor port IpStatistics inBcastPkts64 alarm risingThreshold 400
```

Associated MIB

None

Web path

- Nmon → nmon → modify

mode

Purpose

Displays the current CLI command entry mode: user, enable, or config.

- Enable mode allows access to all commands that do not affect traffic flow to and from the system.
- Config mode allows access to all of the command modes and commands that configure and manage all aspects of the system. You are required to be in Config mode to execute all commands that impact the flow of traffic to and from the system. You must be in Enable mode before you can enter Config mode.
- User mode is the top-level command execution mode where users can view current configuration and status information using the `show` commands.

Access mode

enable, config, user

Syntax

```
mode
```

Example

```
sun> enable
sun# config
sun(config)# mode
config
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

no

Purpose

Deletes a configuration entry. To delete a configuration entry, use the no command followed by an executable command and its appropriate arguments. You must enter any required arguments with the command to identify a specific entry. To view the filter fields available for specifying an entry, enter a question mark after the no *commandName* entry. You can also use a wildcard symbol to delete multiple entries of a specific type. See Appendix A, “Advanced CLI use” for details.

Caution: There is no confirmation or undo feature. The no command immediately deletes the entry or entries.

Access mode

enable

Syntax

no *commandName*

Example

In the following example, the no command deletes the configuration entry for the NTP 3 server.

```
sun(switchServices)# show ntp 3
Server ID:          3
Server IP address:  1.1.1.1
Preferred server:   false
Burst mode:         false
Min. poll interval: 256
Max. poll interval: 1024
NTP version:        4

sun(switchServices)# no ntp 3
sun(switchServices)#

sun(switchServices)# show ntp 3
There are no entries matching your query.
sun(switchServices)#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

quit

Purpose

The `quit` command does the following, depending on where you are in the CLI hierarchy:

- If entered at a command mode, moves up to the previous command mode level.
- If entered at the top of the command hierarchy, moves up to the previous access mode level.
- If entered at the top-level access mode, exits the CLI.

Depending on the user profile definition, the top-level access mode is either User or Enable. See "CLI user profiles" on page 1-6 for details.

This command has the same functionality as the `exit` command.

Access mode

user

Syntax

```
quit
```

Example

The following example illustrates all three uses of the `quit` command.

```
sun(config-vSwitch-system vRouter-management)# quit
sun(config-vSwitch-system)# quit
sun(config)# quit
sun# quit
sun> quit
Goodbye
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

redo

Purpose

Executes the previously executed command.

You can use the `redo` command or [!] to reexecute commands in the history list. The following table lists the valid `redo` commands.

Table 2-2. Redo commands

Command syntax	Description
`redo or !`	Reexecute the last command you entered.
`redo` *nn* or [!] *nn*	Reexecute command *nn* in the history list.
`redo` *-nn* or [!]*-nn*	Reexecute the *nnth* previous command (where *nn* includes the last command you entered).
`redo` *aaa* or [!]*aaa*	Reexecute the most recently entered command beginning with the characters *aaa* in the history list.

Access mode

user

Syntax

```
redo
    command
```

Arguments

Argument	Description
command	Specifies the command to reexecute. See Table 2-2 for entry methods.

Example

The following example reexecutes a command identified by its number in the history list.

```
sun> enable
sun# config
sun(config)# lag 10
sun(config-lag-10)# exit
sun(config)# show lag
Lag ID:       10
Admin State:  enabled
Oper Status:  down
Admin MAC:    00:00:00:00:00:00
Oper MAC:     00:00:00:00:00:00
Jumbo Frames: enabled
Default Vlan: 4095
Flood Port:   N/A

sun(config)# lag 10
sun(config-lag-10)# jumboFrames disabled
sun(config-lag-10)# exit
sun(config)# history
    1  ena
    2  config
    3  lag 10
    4  show
    5  jumboFrames enabled
    6  exit
    7  history
    8  show
    9  q
   10  enable
   11  config
   12  lag 10
   13  exit
   14  show lag
   15  lag 10
   16  jumboFrames disabled
   17  exit
   18  history
```

```
sun(config)# redo 14
Lag ID:       10
Admin State:  enabled
Oper Status:  down
Admin MAC:    00:00:00:00:00:00
Oper MAC:     00:00:00:00:00:00
Jumbo Frames: disabled
Default Vlan: 4095
Flood Port:   N/A

sun(config)#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

rows

Purpose

Displays or sets the number of rows that the CLI displays in command output (also called terminal length). If the output includes more lines than the rows configuration specifies, the CLI prompts you to press [Return] or the spacebar to see the rest of the display. The default rows configuration displays 24 lines.

To display the current rows configuration, enter **rows** and press [Enter]. To change the current rows configuration enter **rows**, followed by a number and press [Enter].

If you change the row value, you must change the window size for the terminal emulator or Telnet/SSH application to match it.

Note: If you enter 0 as the rows value, you disable the display prompting function and the CLI output scrolls to the end.

This command operates the same as the `switchServices cli rows` command.

Access mode

user

Syntax

```
rows
   [integer]
```

Arguments

Argument	Description
integer	**Optional.** Specifies the number of rows the CLI displays in output, before prompting. The `rows` setting applies to the current session only.

Example

The following example illustrates the use of the `rows` command.

```
sun> rows
  24
sun> rows 36
sun> rows
  36
sun>
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

saveCfg

Purpose

Saves the current configuration entries to non-volatile flash memory. When you use the `saveCfg` command to save configuration changes, the system creates a backup file named `cdb.bak`. The `cdb.bak` file contains the settings from the last time you saved the configuration.

Until you save configurations, any changes you make apply to the running configuration only. If you exit the CLI, the running configuration is still in effect. However, if you reboot the system, the running configuration is lost.

The N2000 Series reads the configuration file in flash memory the next time it boots.

Note: Using the `saveCfg` command saves all configuration changes from a CLI session, a Web interface session, or from SNMP operations that are stored in the current running configuration.

Access mode

user

Syntax

```
saveCfg
```

Associated MIB

None

Web path

This command is available through the Web interface dashboard.

show

Purpose

Displays configuration entries. To view a listing of the `show` commands that are available in a command mode, navigate to the command mode and enter:

```
sun# show ?
```

To view the filter fields that are available for specifying an entry to display, enter a question mark (?) after the `show` *`commandName`* entry. You can also use the asterisk (*) wildcard when entering filter values. See Appendix A, “Advanced CLI use” for details.

Access mode

user

Syntax

`show` *`commandName`*

Example

The following example shows different points from which you can enter the same command.

```
sun(config-vSwitch-caplan2 vRouter-default)# show
ID:                 4
Name:               default
Description:        Default vRouter
Admin State:        enabled
Operational Status: up

sun(config-vSwitch-caplan2 vRouter-default)# quit
sun(config-vSwitch-caplan2)# show vrouter
ID:                 4
Name:               default
Description:        Default vRouter
Admin State:        enabled
Operational Status: up
```

```
sun(config-vSwitch-caplan2)# quit
sun(config)# show vswitch caplan2 vrouter
ID:                 4
Name:               default
Description:        Default vRouter
Admin State:        enabled
Operational Status: up
```

Associated MIB

None

Web path

Click on any command in the command menu to display configuration data.

show redundantConfig

Purpose

Displays the parts of the current running configuration that are required to synchronize the virtual service configuration of two Sun switches being used as a redundant pair. The N2000 Series displays only those commands that are accessible via the user profile associated with the active user. That is, it displays the commands that can be used to synchronize the configuration between two redundant switches, but the display is limited to those commands that may be accessed by the user. See Chapter 31, "Virtual Service Redundancy Protocol commands" and Chapter 32, "Virtual Router Redundancy Protocol commands" for information about the commands necessary to configure redundancy. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for a detailed discussion of the redundant pair configuration procedure.

To configure switches for redundancy, you only copy portions of the configuration. There are some parts, such as the CKM configuration, that are not portable for security reasons. Additionally, there are portions that should not be copied because they are specific to the switch and the switch's relative location in the network, for example, vRouter configuration. The following commands are copied between systems for redundancy:

```
vSwitch
vSwitch loadBalance cookiePersistence
vSwitch loadBalance healthCheckProfile
vSwitch loadBalance host
vSwitch loadBalance objectRule
vSwitch loadBalance realService
vSwitch loadBalance realService advanced
vSwitch loadBalance realService ssl
vSwitch loadBalance requestPolicy
vSwitch loadBalance requestTransform
vSwitch loadBalance responsePolicy
vSwitch loadBalance responseTransform
vSwitch loadBalance serviceGroup
vSwitch loadBalance serviceGroup ssl
vSwitch loadBalance staticNat
vSwitch loadBalance virtualService
vSwitch loadBalance virtualService advanced
vSwitch loadBalance virtualService ssl
vSwitch resource serviceBandwidth
vSwitch ckm import paste
vSwitch vRouter
```

If you want to view the running configuration for a specific command mode or system component only, enter the `show redundantConfig` command from within the component. You can save the running configuration settings in a text file, move the file to another system, edit the file using a text editor, and then copy the file to another N2000 Series. See the *Sun N2000 Series Release 2.0 – System Administration Guide* for a detailed procedure.

Access mode

user

Syntax

```
show redundantConfig
   [saveToFile fileName]
   [password passwordText]
   [defaultValues {true | false}]
   [showHeaders {true | false}]
```

Arguments

Argument	Description
`saveToFile` *`fileName`*	**Optional.** Specifies the name of the file in which you want to save the running configuration. The system saves the file in the `/ftl0/usrhome` directory, unless you specify a path.
`password` *`passwordText`*	**Optional.** Assigns a password to the running configuration file. The password must be greater than 4 and fewer than 255 characters. If you do not specify a password, the system does not display or write to a file any data that requires a password (for example, other password data and CKM information). **Note:** You need to specify this password when you import the running configuration.
`defaultValues {true | false}`	**Optional.** Specifies whether the system includes configuration entries that are using default values. If `true`, the system includes configuration entries with default values in the display or specified file. If `false`, the system omits these details. The default value is `false`.

Argument	Description
showHeaders {true | false}	**Optional.** Specifies whether the system includes headings for each configuration entry. If `true`, the system includes headings in the display or specified file. If `false`, the system omits the headings. The default value is `false`.

Example

```
sun> show redundantConfig
If a password is not provided, private data that needs to be encrypted
will not be displayed.
Do you wish to continue? (y or n): y
_vSwitch vSwitchName e-commerce
_vSwitch e-commerce vRouter name default description {Default vRouter}

_vSwitch vSwitchName elmo
_vSwitch elmo loadBalance host name fastball ipAddress 10.10.10.3

_vSwitch elmo loadBalance realService name fbrs hostName fastball port
8080 description { } sslCiphers {RSA_WITH_RC4_128_MD5;
RSA_WITH_RC4_128_SHA; RSA_WITH_3DES_EDE_CBC_SHA}
_vSwitch elmo loadBalance realService fbrs advanced xmtRetryLimit 8
estRetryLimit 3 shortRxTimer 32_seconds longRxTimer 128_seconds rcvWnd
16384


_vSwitch elmo vRouter name default description {Default vRouter}

_vSwitch vSwitchName system description {System vSwitch}
_vSwitch system vRouter name management description {System Management
vRouter}
_vSwitch system vRouter name shared description {Shared vRouter}


sun>
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

show runningConfig

Purpose

Displays all or parts of the current running configuration. If you want to view the complete running configuration, enter the command at the system level. If you want to view the running configuration for a specific command mode only, enter the `show runningConfig` command from within the command mode.

You can save the running configuration settings in a text file, move the file to another system, edit the file using a text editor, and then copy the file to another N2000 Series. See the *Sun N2000 Series Release 2.0 – System Administration Guide* for a detailed procedure.

Access mode

user

Syntax

```
show runningConfig
   [saveToFile fileName]
   [password passwordText]
   [defaultValues {true | false}]
   [showHeaders {true | false}]
```

Arguments

Argument	Description
`saveToFile` *`fileName`*	**Optional.** Specifies the name of the file in which you want to save the running configuration. The system saves the file in the `/ftl0/usrhome` directory, unless you specify a path.

Argument	Description
`password` *`passwordText`*	**Optional.** Assigns a password to the running configuration file. The password must be greater than 4 and fewer than 255 characters. If you do not specify a password, the system does not display or write to a file any data that requires a password (for example, other password data and CKM information). **Note:** You need to specify this password when you import the running configuration.
`defaultValues {true | false}`	**Optional.** Specifies whether the system includes configuration entries that are using default values. If `true`, the system includes configuration entries with default values in the display or specified file. If `false`, the system omits these details. The default value is `false`.
`showHeaders {true | false}`	**Optional.** Specifies whether the system includes headings for each configuration entry. If `true`, the system includes headings in the display or specified file. If `false`, the system omits the headings. The default value is `false`.

Example

This example shows how to use the CLI to view the complete running configuration. In this example, the system displays configuration values that are not default values and the output does not include headings. This example shows only part of the output that you typically see.

```
sun> show runningConfig
_event syslog host 192.168.1.172 port 50322 logLevel debug

_lag lagId 10
_lag 10 interface port eth.1.20 floodPref 8 weight 100


_port ifIndex eth.1.1 phyDuplex fullDuplex
_port ifIndex eth.1.2 phyDuplex fullDuplex
_port ifIndex eth.1.3 phyDuplex fullDuplex
_port ifIndex eth.1.4 phyDuplex fullDuplex
```

(continued)

```
_switchServices sshd confEncryption {des3Cbc
blowfishCbc
des} confHmac {md5
sha1
md5b96
sha1b96} userAuthentication {publicKey
password}

Enter password to encrypt/decrypt private data: test
_switchServices userAdministration keyInfo
EXPORT:3221fe1c4de6cdfb7b6fab5e3538ad
eeb03299391078803fe8a650ea71e20a37a903b711c1c5ad31e975cf78371549bc
_switchServices userAdministration user userName .default priority 1
_switchServices userAdministration user userName admin priority 1
authentication
Method alwaysAccept profileName systemAdmin userSshdPrivs
sftpReadWrite vSwitchName system

_vSwitch vSwitchName system description {System vSwitch}
_vSwitch system resource portBandwidth ifIndex eth.1.22
bandwidthAllocation 100
bandwidthMaximum 100 burstSize 65534 burstSizeMaximum 65535
_vSwitch system resource portBandwidth ifIndex eth.1.24
bandwidthAllocation 100
bandwidthMaximum 100 burstSize 65534 burstSizeMaximum 65535

_vSwitch system vRouter name management description {System Management
vRouter}
_vSwitch system vRouter management interfaces connectionIndex
sock.system:management linkUpDownTrap disabled eventFilter
informational alias { }
_vSwitch system vRouter management interfaces connectionIndex
sock.system:management/ip.system:management linkUpDownTrap disabled
eventFilter informational alias { } mtu 1500
.
.
.
```

This example shows how to use the CLI to view the running configuration for a specific vSwitch. In this example, the system displays configuration values that are not default values and the output does not include headings.

```
sun(vSwitch-e-commerce)# show runningConfig
_vSwitch e-commerce
_vSwitch e-commerce vRouter name default description {Default vRouter}
_vSwitch e-commerce vRouter default interfaces connectionIndex
sock.e-commerce:default linkUpDownTrap disabled eventFilter
informational alias { }
_vSwitch e-commerce vRouter default interfaces connectionIndex
sock.e-commerce:default/ip.e-commerce:default linkUpDownTrap disabled
eventFilter informational alias { } mtu 1500
_vSwitch e-commerce vRouter default interfaces connectionIndex
ip.e-commerce:default linkUpDownTrap disabled eventFilter
informational alias { }

_vSwitch e-commerce vRouter default ip forwarding enabled
_vSwitch e-commerce vRouter default ip icmp replyToEchos true
sendDestUnreachs false sendTimeExceeds true sendParamProbs false
replyToMasks true

sun(vSwitch-e-commerce)#
```

Associated MIB

None

Web path

This command is not applicable to the Web interface.

Part II. System management

The chapters in Part II describe the commands for managing the N2000 Series system:

Chapter 3. Chassis commands

Chassis command description

The chassis commands allow you to manage some of the Chassis Manager attributes. The primary responsibility of the Chassis Manager is to detect module availability and check available power and cooling capability before allowing any boards to power up.

You can use the chassis commands to specify system boot devices and manage the operational state of system modules. You can also monitor power and cooling operational statistics.

Chassis command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`switchServices chassis` *`commandName`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Chassis command summary

Table 3-1 lists and briefly describes the chassis commands.

Table 3-1. Chassis command summary

Command name	Description
`bootParameters`	Configure settings for the boot device that the system uses when it restarts.
`module`	Manage the operational state of the system board and function cards.
`privateKeySalt`	Set the privateKeySalt, which is used when encrypting private keys.
`reset`	Reset the system board.
`show bootParameters`	Display the settings for the boot device that the system uses when it restarts.
`show cpuload`	Display CPU utilization across N2000 Series hardware modules.
`show fan`	Display the operational status of the chassis fans.
`show module`	Display the operational state of the system board and function cards.
`show power`	Display the operational state of the power supplies in the chassis and the option modules.

bootParameters

Purpose

Specifies the device the system uses to load the operating system software during the boot process. During the boot process, the system loads a basic configuration from the system motherboard and then loads the network operating system from either the flash disk or from a Trivial File Transport Protocol (TFTP) server. If you plan to use TFTP, ensure that you enable the TFTP daemon. See Chapter 7, "TFTP commands" for details about enabling TFTP.

Access mode

enable

Syntax

```
switchServices chassis bootParameters
   [ipAddress ipAddress]
   [defaultGateway ipAddress]
   [tftpIP ipAddress]
   [tftpDir directoryName]
   [tftpFilename fileName]
   [mask ipAddress]
   [bootMethod1 {filesystem | tftp}]
   [bootMethod2 {filesystem | tftp}]
```

Arguments

Argument name	Description
`ipAddress` *`ipAddress`*	**Optional**. Specifies the IP address for the system.
`defaultGateway` *`ipAddress`*	**Optional**. Specifies the default gateway for the system.
`tftpIP` *`ipAddress`*	**Optional**. Specifies the IP address of a TFTP server, if you plan to use TFTP as a boot method.
`tftpDir` *`directoryName`*	**Optional**. Specifies the directory on the TFTP server from which you download files.

Argument name	Description
`tftpFilename` *`fileName`*	**Optional**. Specifies the file you want to download from the TFTP server. See the Release Notes for software version that you are using for the correct file name.
`mask` *`subnetMask`*	**Optional**. Assigns a subnet mask to the N2000 Series.
`bootMethod1 {filesystem | tftp}`	**Optional**. Selects the boot device the system tries first when it reboots. You download files from the flash disk or download a file from a TFTP server when rebooting the system. Valid values are `filesystem` or `tftp`; the default setting is `filesystem`.
`bootMethod2 {filesystem | tftp}`	**Optional**. Selects the boot device that the system tries if the first specified boot device is not available. You can download files from the flash disk or download a file from a TFTP server when rebooting the system. Valid values are `filesystem` or `tftp`; the default setting is `filesystem.`

Example

The following example shows how to configure the system to reboot using the local file system as the preferred boot method and TFTP as the backup boot method.

```
sun> enable
sun# switchServices
sun(switchServices)# chassis
sun(switchServices chassis)# bootParameters ipAddress 10.10.10.2
defaultGateway 10.10.20.18 tftpIP 10.10.10.4 tftpDir bootfile/
tftpFilename an20.elf mask 255.255.255.0 bootMethod1 filesystem
bootMethod2 filesystem
```

Associated MIB

```
hardware.mib
```

Web path

- switchServices → chassis → bootParameters → modify

module

Purpose

Allows you to manage the operational state of the hardware modules. The hardware modules are:

- SystemBoard — The main system board for the system.
- FunctionCard1 — The function card installed in the front space of the system.
- FunctionCard2 — The function card installed in the back space of the system.

Access mode

enable

Syntax

```
switchServices chassis module
   moduleId {systemBoard | functionCard1 | functionCard2}
   [adminAction {none | shutdown | restart}]
```

Arguments

Argument name	Description
`moduleID {systemBoard | functionCard1 | functionCard2}`	Specifies a module in the system. The valid values are: `systemBoard`: The system motherboard. `functionCard1`: The function card installed in the front space of the system. `functionCard2`: The function card installed in the back space of the system. This is not currently used.

Argument name	Description
`adminAction {none | restart | shutdown}`	**Optional**. Specifies the action you want to perform on the module. Valid values are: `none`: No action taken. `restart`: Restarts the specified module. Restarting the system motherboard causes the entire system to reboot. `shutdown`: Shuts down the module without restarting it.

Example

The following example shows how to restart the system motherboard. Restarting the system motherboard restarts the entire system.

```
sun> enable
sun# switchServices
sun(switchServices)# chassis
sun(switchServices chassis)# module moduleID systemBoard adminAction
restart
```

Associated MIB

`hardware.mib`

Web path

- switchServices → chassis → module → modify

privateKeySalt

Purpose

Sets a secret passphrase that is used as part of the encryption scheme for private keys. This passphrase is hashed and stored in an inaccessible location, preventing users from removing the flash and gaining access to private key material.

The system ships with an uninitialized salt. Until the `privateKeySalt` is set, you will not be able to perform any key management activities on the switch. (All key management commands will fail, and you will receive an error message stating that the salt must be set.) Once the salt is set, the N2000 Series incorporates it into the encryption of all keys stored on the switch.

If you execute the `privateKeySalt` command a second time with another passphrase, the system overwrites the old one. It is important to back up your configuration before doing this and to run `saveCfg` immediately after doing this. If the power is lost while the salt is being modified (or before you run the `saveCfg` command afterwards), all private keys could become inaccessible.

You can delete the `privateKeySalt` by entering two double quotes for a passphrase. This causes the system to revert to an uninitialized salt and disables the CKM. If the N2000 Series is turned off while the salt is unset, all private keys will become inaccessible until the old `privateKeySalt` value is reentered. If the salt is set again before the N2000 Series is turned off, then the behavior is the same as if the passphrase were changed without deleting it first (described above).

Access mode

config

Syntax

```
switchServices chassis privateKeySalt
   passphrase text
```

Arguments

Argument name	Description
`passphrase` *`text`*	Specifies a text string that the system uses as part of the encryption scheme. The phrase must be between 4 and 255 alphanumeric characters. Entering a set of double quotes (""") for the passphrase erases the existing, stored salt and resets the value to the uninitialized manufacturing default. You must then reset the salt for CKM to work again.

Example

The following example shows the warning that is displayed when you try to set the `privateKeySalt` on a system with a `privateKeySalt` already set.

```
sun> enable
sun# config
sun(config)# switchServices
sun(config-switchServices)# chassis
sun(config-switchServices chassis)# privateKeySalt passphrase
keepitprivate123
Please confirm the passphrase:

Changing the privateKeySalt is a dangerous operation. If the box loses
power during the operation (or before the next saveCfg), all private
keys will become inaccessible. Backing up your config is recommended.
Change the privateKeySalt now? (y or n): n
```

Associated MIB

`hardware.mib`

Web path

- switchServices → chassis → privateKeySalt → modify

reset

Purpose

Restarts the system motherboard. This command is the equivalent of the `module` command (`switchServices chassis module moduleId systemBoard adminAction restart`). Restarting the system causes the system to reboot, interrupting all services.

Note: This command is *not* executed from within the chassis command mode.

Access mode

enable

Syntax

```
switchServices reset
```

Example

The following example shows how to restart the system motherboard. Restarting the system motherboard restarts the entire system.

```
sun> enable
sun# switchServices
sun(switchServices)# reset
```

Associated MIB

```
hardware.mib
```

Web path

- switchServices → reset

show bootParameters

Purpose

Displays the current methods the system uses when it reboots. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices chassis bootParameters
```

Sample output

```
sun> switchServices
sun(switchServices)> chassis
sun(switchServices chassis)> show bootParameters
IP Address Of Switch: 10.10.10.1
Default Gateway:      10.10.10.18
TFTP IP Server:       10.10.10.66
TFTP Directory:       bootfile/
TFTP Filename:        ancc.elf
IP Mask:              255.255.255.0
1st Boot Method:      filesystem
2nd Boot Method:      filesystem
```

Output description

Field name	Description
IP Address Of Switch	The IP address of the system.
Default Gateway	The IP address of the default gateway for the system.
TFTP IP Server	The IP address of the TFTP server the system uses to retrieve software during the boot process.
TFTP Directory	The directory on the TFTP server from which the system retrieves software files.
TFTP Filename	The name of the file on the TFTP server that the system retrieves during the boot process.
IP Mask	The subnet mask for the TFTP server.

Field name	Description
`1st Boot Method`	The first boot method that the system tries. If set to `filesystem`, the system tries to boot using files stored on the flash disk; if set to `tftp`, the system tries to boot using TFTP to retrieve files from a remote host.
`2nd Boot Method`	The boot method that the system tries if the first boot method fails. If set to filesystem, the system tries to boot using files stored on the flash disk; if set to tftp, the system tries to boot using TFTP to retrieve files from a remote host.

Associated MIB

`hardware.mib`

Web path

- switchServices → chassis → bootParameters

show cpuload

Purpose

Displays the maximum central processing unit (CPU) load (in percentage) being placed on all N2000 Series hardware modules.

Access mode

user

Syntax

```
show switchServices chassis cpuload
```

Sample output

```
sun> switchServices
sun(switchServices)> chassis
sun(switchServices chassis)> show cpuload
Max CPU Load (%): 24
```

Output description

Field name	Description
Max CPU Load (%)	The current percentage of CPU utilization across all Sun Application Switch hardware modules.

Associated MIB

```
hardware.mib
```

Web path

- switchServices → chassis → cpuload

show fan

Purpose

Displays the current state of the fans in the chassis. The system automatically sets the fan speed, based on current temperature. In a cool environment, the fans run at a slower speed. In a warm environment, the fan speed increases to ensure adequate system cooling. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices chassis fan
```

Sample output

```
sun> switchServices
sun(switchServices)> chassis
sun(switchServices chassis)> show fan
Fan Speed:    slow
Fan 1 Status: working
Fan 2 Status: working
Fan 3 Status: working
Fan 4 Status: working
Fan 5 Status: working
Fan 6 Status: working
Fan 7 Status: working
```

Output description

Field name	Description
`Fan Speed`	The current speed of the fans in the chassis. Possible values are `slow` and `fast`.
`Fan 1..7 Status`	The current status of each fan. Possible values are `working` or `failed`.

Associated MIB

`hardware.mib`

Web path

- switchServices → chassis → fan

show module

Purpose

Displays details for all modules installed in the chassis.

Access mode

user

Syntax

```
show switchServices chassis module
```

Sample output

```
sun> switchServices
sun(switchServices)> chassis
sun(switchServices chassis)> show module
Module:            systemBoard
Description:       4 GbE, 40 10/100-baseT System Board
Type:              N2040
Hardware Revision: A
OperationalStatus: running
GppMemory:         268435456
UpTime:            309
Temp Sensor1 (C):  26
Temp Sensor2 (C):  32
Temp Sensor3 (C):  34
Temp Sensor4 (C):  26
Part Number:       530002700
Serial Number:     PLX08020314
Eeprom Version:    3
```

Output description

Field name	Description	Filter name
`Module {systemBoard | functionCard1 | functionCard2}`	The module installed in the system. Possible values are: `systemBoard:` The system motherboard. `functionCard1:` The function card installed in the front space of the system. `functionCard2:` The function card installed in the back space of the system. This is not used currently.	`moduleId {systemBoard | functionCard1 | functionCard2}`
`Description`	A textual description of the module. Possible values are: `4 GbE, 40 10/100-baseT system board`: The system motherboard with 4 Gigabit Ethernet ports and 40 10/100-baseT ports. `12 Gbe system board`: The system motherboard with 12 Gigabit Ethernet ports. `SLB function card`: The System Load Balancing Function Card. `SLB and SSL function card`: The System Load Balancing Function Card with SSL.	`description {4 GbE, 40 10/100-baseT system board | 12 Gbe system board | SLB function card | SLB and SSL function card}`

Field name	Description	Filter name
Type	The type of module. Possible values are: N2040: The system motherboard for the N2040. N2120: The system motherboard for the N2120. FxSSL: The System Load Balancing Function Card with SSL.	type {N2040 | N2120 | Fx-SSL}
Hardware Revision	The revision identification for the module (A through Z).	hardwareRevision *revNumber*
Operational Status	The current status of the module. Possible values are: booting: The module is in the process of booting. loading: The module is loading software. running: The module is operating normally. broken: The module is not operating normally. off: The module is shut down.	operationalStatus {booting | loading | running | broken | off}
GppMemory	The amount of memory, in bytes, that is installed for the general purpose processor on the module.	gppMemory *bytes*
CPU Load %	The percentage of the CPU currently in use.	CPU Load %
UpTime	The amount of time, in seconds, since the last time the module rebooted.	upTime *integer*
Temp Sensor1 (C)	The current temperature, in Celsius, for the first temperature sensor.	tempSensor1 *temp*
Temp Sensor2 (C)	The current temperature, in Celsius, for the second temperature sensor.	tempSensor2 *temp*

Field name	Description	Filter name
`Temp Sensor3 (C)`	The current temperature, in Celsius, for the third temperature sensor.	`tempSensor3` *`temp`*
`Temp Sensor4 (C)`	The current temperature, in Celsius, for the fourth temperature sensor.	`tempSensor4` *`temp`*
`Part Number`	The part number for the module.	`partNumber` *`partNumber`*
`Serial Number`	The serial number for the module.	`serialNumber` *`serialNumber`*
`Eeprom Version`	The version of the EEPROM (electrically erasable programmable read-only memory) on the module.	`eepromVersion` *`versionNumber`*
`Software Watchdog Fatal Errors`	The number of times a runaway process was killed by the Software Watchdog.	`softwareWatchdogFatalErrors` *`counter`*
`Software Watchdog Warnings`	The number of times the Software Watchdog noticed a process was running for too long but did not kill it.	`softwareWatchdogWarnings` *`counter`*

Associated MIB

`hardware.mib`

Web path

- switchServices → chassis → module

show power

Purpose

Displays the operational status of the power supplies installed in the chassis. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices chassis power
```

Sample output

```
sun> switchServices
sun(switchServices)> chassis
sun(switchServices chassis)> show power
Power Supply 1(DC):       operating
Power Supply 1(AC):       operating
Power Supply 2(DC):       notPresent
Power Supply 2(AC):       notPresent
Power Supply Option1(DC): operating
Power Supply Option2(DC): notPresent
```

Output description

Field name	Description
`Power Supply 1(DC)`	The operational state of the DC output of the main power supply. Possible values are: `operating`, `failed`, `notPresent`.
`Power Supply 1(AC)`	The operational status of the AC input of the main power supply. Possible values are: `operating`, `failed`, `notPresent`.
`Power Supply 2(DC)`	The operational state of the DC output of the redundant power supply. Possible values are: `operating`, `failed`, `notPresent`.
`Power Supply 2(AC)`	The operational status of the AC input of the redundant power supply. Possible values are: `operating`, `failed`, `notPresent`.

Field name	Description
`Power Supply Option1(DC)`	The operational status of power for functionCard1. Possible values are: `operating`, `failed`, `notPresent`.
`Power Supply Option2(DC)`	The operational status of power for functionCard2. Possible values are: `operating`, `failed`, `notPresent`.

Associated MIB

`hardware.mib`

Web path

- switchServices → chassis → power

Chapter 4. Remote monitoring commands

N2000 Series remote monitoring description

The remote network monitor (NMON) allows network administrators to monitor N2000 Series network and system statistics using configured alarms that generate events when thresholds are crossed.

NMON uses alarms that are configured with the global `monitor` command from the CLI, or with the Web interface monitor button. Refer to Chapter 2, “Global commands” for information about using the `monitor` command.

Note: Previously configured alarms can be modified using the `alarm` command.

NMON command path

The command names in this chapter show you how to execute the commands from within the following command modes:

`nmon`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

NMON command summary

Table 4-1 lists and briefly describes the NMON commands.

Table 4-1. NMON command summary

Command name	Description
`alarm`	Configure NMON alarm thresholds, polling intervals, and event levels.
`nmon (root)`	Enter NMON monitoring mode.
`show nmon`	Display NMON administrative and operating status.
`show nmon alarm`	Display the current NMON alarm table.
`show nmon alarm result`	Display NMON alarm polling results.

Basic NMON configuration

The following procedure explains the basic steps for configuring NMON on the system.

Steps for configuring NMON

Step	Action
1.	Enable the remote network monitor with the `nmon (root)` command.
2.	Set alarm thresholds with the `monitor` command.
3.	Verify the current alarm settings with the `show nmon alarm` command.

Example

```
sun> enable
sun# configure
sun(config)# nmon
sun(config-nmon)# adminState enabled
sun(config-nmon)# exit
sun(config)# monitor switchServices httpd currentSessions alarm 5
sun(config)# show nmon alarm
Index:                          1
vSwitch:                        N/A
vRouter:                        N/A
```

```
Base Command:                   switchServices httpd
Filter Field And Values:        N/A
Monitor Field:                  currentSessions
Poll Interval:                  60
Alarm Interval:                 60
Sample Type:                    deltaValue
Rising Threshold:               5
Falling Threshold:              5
Rising Event Level:             warning
Falling Event Level:            none
Poll Check Count:               1
Alarm Check Count:              1
Result Count:                   1
Current Result Count:           1
Total Rising Triggered Count:   0
Total Falling Triggered Count:  0
Rising Alarm Generated Count:   0
Falling Alarm Generated Count:  0
Last Poll Time:                 8/2/2004-13:33:49
Last Alarm Time:                8/2/2004-13:33:36
```

alarm

Purpose

Modifies the poll parameters of an alarm that was configured with the monitor command. The no form of the command removes the specified alarm.

Access mode

config

Syntax

```
nmon alarm
   index integer
   [risingThreshold text]
   [pollInterval integer]
   [fallingThreshold text]
   [alarmInterval integer]
   [sampleType {absoluteValue | deltaValue | factorDeltaValue}]
   [risingEventLevel {emergency | alert | critical | error | warning |
      notice | informational | debug | none}]
   [fallingEventLevel {emergency | alert | critical | error | warning
      | notice | informational | debug | none}]
```

Arguments

Argument name	Description
`index` *integer*	Specifies the number that uniquely identifies the NMON alarm entry. Each index entry defines a configured object to be polled at a particular interval on the N2000 Series. Specify the index number to modify the NMON alarm table configuration settings for a specific monitored object.
`risingThreshold` *text*	**Optional.** Sets the upper threshold value to be checked at each NMON poll. When the polled value is greater than or equal to this threshold, and the value at the last poll interval was less than this threshold, the system generates an event of level `risingEventLevel`. After a rising event is generated, the system does not generate another such event until the sampled value falls below this threshold and reaches the `alarmFallingThreshold`.
`pollInterval` *integer*	**Optional**. Sets the amount of time (in seconds) between polls. NMON polls values at the poll interval setting and compares the results to the threshold values. At this poll interval, no alarm is generated, but if a threshold is crossed, it is noted for the next alarm interval. The default setting is 60 (seconds). The poll interval range is 5 to 65535.
`fallingThreshold` *text*	**Optional.** Sets the lower threshold value to be checked at each NMON poll. When the current sampled value is less than or equal to this threshold, and the value at the last sampling interval was greater than this threshold, the system generates an event. After a falling event is generated, the system does not generate another such event until the sampled value rises above this threshold and reaches the `alarmRisingThreshold`. If no value is specified, `fallingThreshold` defaults to the value of `risingThreshold`.
`alarmInterval` *integer*	**Optional.** Sets the amount of time (in seconds) between alarm checks used to determine if an alarm should be generated if a poll detected a rising for falling threshold event. The default setting is 60 seconds. The alarm interval range is 5 to 65535.

Argument name	Description
`sampleType {absoluteValue | deltaValue | factoredDeltaValue}`	**Optional.** Sets the method used by the poll for sampling the selected variable and calculating the value to be compared against the thresholds. • `absoluteValue`—The value of the parameter will be compared directly with the thresholds at the end of the polling interval. • `deltaValue`—The value of the parameter at the last poll is subtracted from the current value, with the difference compared with the configured thresholds. • `factoredDeltaValue`—The value of the parameter at the last poll, subtracted from the current value, with the difference compared with the thresholds. The value will also be factored for the true duration of the poll if it is different than the poll interval. The default value is `deltaValue`.

Argument name	Description
`risingEventLevel {emergency | alert | critical | error | warning | notice | informational | debug | none}`	**Optional.** Sets the level of syslog event message that the system sends when a rising threshold alarm is generated. Possible values are: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; Immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: An event occurred that can cause a loss of some system functionality. Administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: Debug-level event; for Sun technical support use only. The default setting is `warning`.

Argument name	Description
`fallingEventLevel {emergency | alert | critical | error | warning | notice | informational | debug | none}`	**Optional.** Sets the level of syslog event message that the system sends when a falling threshold alarm is generated. Possible values are: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; Immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: An event occurred that can cause a loss of some system functionality. Administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: Debug-level event; for Sun technical support use only. The default setting is `none`.

Delete filters

```
no nmon alarm
   index integer
   [risingThreshold text]
   [pollInterval integer]
   [vSwitch text]
   [vRouter text]
   [baseCommand text]
   [filterFieldAndValues text]
   [tableName text]
   [fieldName text]
   [fallingThreshold text]
   [alarmInterval integer]
   [sampleType {absoluteValue | deltaValue | factorDeltaValue}]
   [risingEventLevel {emergency | alert | critical | error | warning |
      notice | informational | debug | none}]
```

```
[fallingEventLevel {emergency | alert | critical | error | warning
   | notice | informational | debug | none}]
[alarmCheckCount integer]
[pollCheckCount integer]
[resultCount integer]
[currentResultCount integer]
[totalRisingTriggeredCount integer]
[totalFallingTriggeredCount integer]
[risingAlarmGeneratedCount integer]
[fallingAlarmGeneratedCount integer]
[lastPollTime time]
[lastAlarmTime time]
```

Example

```
sun(config)# nmon alarm index 1 risingThreshold 15 fallingThreshold 8
alarmInterval 120 sampleType absoluteValue pollInterval 60
risingEventLevel critical fallingEventLevel critical
```

Associated MIB

```
rmon.mib
```

Web path

- nmon → alarm

nmon (root)

Purpose

Enters the NMON monitoring mode and allows you to enable or disable NMON.

Access mode

config

Syntax

```
nmon
   [adminState {enabled | disabled}]
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	**Optional**. Sets the administrative state of NMON on the N2000 Series, either `enabled` or `disabled`. Set a state of `disabled` to stop the polling of all the monitor alarms. NMON is `disabled` by default.

Example

```
sun> enable
sun# configure
sun(config)# nmon
sun(config-nmon)# adminState enabled
sun(config-nmon)#
```

Associated MIB

```
rmon.mib
```

Web path

- nmon → nmon → modify

show nmon

Purpose

Displays the current NMON administrative and operational states.

Access mode

user

Syntax

```
show nmon
```

Sample Output

```
sun> enable
sun# configure
sun(config)# show nmon
Administrative State: enabled
Operational Status:   up
```

Output description

Field name	Description
`Administrative State`	The administrative state of NMON on the N2000 Series, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring NMON offline or preconfigure it before bringing it online. NMON is `disabled` by default.
`Operational Status`	The current NMON operational state, either `up`, `down`, or `error`. An `error` indicates an internal software condition that is preventing operation.

Associated MIB

`rmon.mib`

Web path

- nmon → nmon

show nmon alarm

Purpose

Displays the currently configured NMON alarms.

Access mode

user

Syntax

```
show nmon alarm
```

Sample output

```
sun(config)# show nmon alarm
Index:                          1
vSwitch:                        e-commerce
vRouter:                        N/A
Base Command:                   vSwitch loadBalance proxyIPPool
statistics
Filter Field And Values:        name pool_2
Monitor Field:                  portsAllocated
Poll Interval:                  60
Alarm Interval:                 60
Sample Type:                    deltaValue
Rising Threshold:               6
Falling Threshold:              6
Rising Event Level:             warning
Falling Event Level:            none
Poll Check Count:               246
Alarm Check Count:              246
Result Count:                   245
Current Result Count:           1
Total Rising Triggered Count:   0
Total Falling Triggered Count:  1
Rising Alarm Generated Count:   0
Falling Alarm Generated Count:  1
Last Poll Time:                 7/31/2003-11:13:20
Last Alarm Time:                7/31/2003-11:13:20
```

Output description

Field name	Description	Filter name
`Index`	A number that uniquely identifies an entry in the alarm table. Each entry shows the polled results at a particular interval for a monitored object on the N2000 Series.	`index` *`integer`*
`vSwitch`	The name of the vSwitch on which the monitored object is running.	`vSwitch` *`text`*
`vRouter`	The name of the vRouter on which the monitored object is running.	`vRouter` *`text`*
`Base Command`	The name of the object being monitored, as configured with the `monitor` command.	`baseCommand` *`text`*
`Filter Field and Values`	The name of the field and corresponding value(s) for filtering at the next NMON poll. This filter can limit the values in the result table to only those that match the criteria. For example, you can limit fields based on regexp matches for specific parameters.	`filterFieldAndValues` *`text`*
`Monitor Field`	The name of the field in the alarm table that is being monitored.	`fieldName` *`text`*
`Poll Interval`	The amount of time (in seconds) between polls. NMON polls values at the poll interval setting and compares the results to the threshold values. At this poll interval, no alarm is generated, but if a threshold is crossed, it is noted for the next alarm interval.	`pollInterval` *`integer`*
`Alarm Interval`	The amount of time (in seconds) between alarm queries. This interval is used to determine if an alarm should be generated when a poll detects a rising or falling threshold event.	`alarmInterval` *`integer`*

Field name	Description	Filter name
`Sample Type`	The method used by the poll for sampling the selected variable and calculating the value to be compared against the thresholds. • `absoluteValue` -- The value of the parameter will be compared directly with the thresholds at the end of the polling interval. • `deltaValue` -- The value of the parameter at the last poll is subtracted from the current value, with the difference compared with the configured thresholds. • `factoredDeltaValue` -- The value of the parameter at the last poll, subtracted from the current value, with the difference compared with the thresholds. The value will also be factored for the true duration of the poll if it is different than the poll interval.	`sampleType {absoluteValue | deltaValue | factoredDeltaValue}`
`Rising Threshold`	The upper threshold value to be checked at each NMON poll. When the polled value is greater than or equal to this threshold, a rising event is recorded. A recorded event generates a syslog event when the alarm interval expires.	`risingThreshold` *`integer`*
`Falling Threshold`	The lower threshold value to be checked at each NMON poll. When the polled value is less than or equal to this threshold, a falling event is recorded. A recorded event generates a syslog event when the alarm interval expires.	`fallingThreshold` *`integer`*

Field name	Description	Filter name
`Rising Event Level`	The type of syslog event message generated for a rising alarm: • `emergency` • `alert` • `critical` • `error` • `warning` • `notice` • `informational` • `debug` • `none` The default setting is `warning`.	`risingEventLevel {emergency | alert | critical | error | warning | notice | informational | debug | none}`
`Falling Event Level`	The type of syslog event message generated for a falling alarm: • `emergency` • `alert` • `critical` • `error` • `warning` • `notice` • `informational` • `debug` • `none` The default setting is `none`.	`fallingEventLevel {emergency | alert | critical | error | warning | notice | informational | debug | none}`
`Poll Check Count`	The number of times the poll check routine was called. This in an internal counter that stores the count used to determine if a poll detected a rising or falling threshold violation.	`pollCheckCount` *`integer`*
`Alarm Check Count`	The number of times the alarm check was called. This is an internal counter that stores the count used to determine if an alarm should be sent based on a poll detecting a rising or falling threshold violation.	`alarmCheckCount` *`integer`*

Field name	Description	Filter name
`Result Count`	The number of NMON polling results, by row, since polling started. Since each alarm can check multiple rows in a table (monitor more than one object), this count reports the total number of results posted since the alarm started polling. Disabling the application or restarting the system resets the count to zero.	`resultCount` *`integer`*
`Current Result Count`	The number of rows returned when the system last executed an alarm poll. Since each alarm can check multiple rows in a table, this count reports how many results are represented by a single poll. As rows in a table are added and removed, this value changes.	`currentResultCount` *`integer`*
`Total Rising Triggered Count`	The total number of times the rising threshold was crossed. Each time a poll detects that the polled value is greater than or equal to the rising threshold, the count increments. This value is different than the count of alarms generated if the pollInterval and alarmInterval differ.	`totalRisingTriggeredCount` *`integer`*
`Total Falling Triggered Count`	The total number of times the falling threshold was crossed. Each time a poll detects that the polled value is less than or equal to the falling threshold, the count increments. This value is different than the count of alarms generated if the pollInterval and alarmInterval differ.	`totalFallingTriggeredCount` *`integer`*
`Rising Alarm Generated Count`	The number of alarms sent due to one or more rising threshold crossings. Each time the system generates a rising alarm, this count increments.	`risingAlarmGeneratedCount` *`integer`*
`Falling Alarm Generated Count`	The number of alarms sent due to one or more falling threshold crossings. Each time the system generates a falling alarm, this count increments.	`fallingAlarmGeneratedCount` *`integer`*

Field name	Description	Filter name
`Last Poll Time`	The date and time of the last NMON poll. This value is N/A until the first successful poll. Afterwards, it represents the last date and time that a poll of the current value and value check occurred. The interval between poll checks should be close to the poll interval.	`lastPollTime` *`mm/dd/yyyy-hh:mn:ss`*
`Last Alarm Time`	The date and time when the system last checked to see if an alarm should be sent due to crossings of the falling and rising thresholds. The time between alarm checks should be close to the alarm interval.	`lastAlarmTime` *`mm/dd/yyyy-hh:mn:ss`*

Associated MIB

`rmon.mib`

Web path

- nmon → alarm

show nmon alarm result

Purpose

Displays the current NMON alarm polling results for each monitored object.

Access mode

user

Syntax

```
show nmon alarm result
```

Sample output

```
sun(config)# show nmon alarm result
Index:                                      1
Key Info:                                   name pool_2
vSwitch Name:                               e-commerce
vRouter Name:
Time:                                       3936374080
Value:                                      0
Rising Triggered Count Since Last Alarm:    0
Falling Triggered Count Since Last Alarm:   0
Total Rising Triggered Count:               0
Total Falling Triggered Count:              1
Rising Alarm Generated Count:               0
Falling Alarm Generated Count:              1
Can Exceed Rising:                          true
Can Exceed Falling:                         false
```

Output description

Field name	Description	Filter name
`Index`	The number that uniquely identifies the NMON alarm entry for which this polling result is saved. The index matches a specific entry in the alarm table.	`index` *`integer`*
`Key Info`	The specific key data that represents a single row poll result. Since a single alarm entry can result in multiple results, the key info provides specific data to uniquely represent a single result.	`keyInfo` *`text`*
`vSwitch Name`	The name of the vSwitch associated with this alarm entry.	`vSwitch` *`text`*
`vRouter Name`	The name of the vRouter associated with this alarm entry. If the poll is directed at a specific vRouter, both the vSwitch name and the vRouter name fields display their configured names.	`vRouter` *`text`*
`Time`	The internal microsecond system timestamp indicating when the poll occurred. The timestamp is used for comparisons associated with the `factoredDeltaValue` and `deltaValue` sample types.	`time` *`integer`*
`Value`	The actual value of the statistic determined by the polling result during the last sampling period. For example, if the sample type is `deltaValue`, this value is the difference between the samples at the beginning and end of the period. If the sample type is `absoluteValue`, this value is the sampled value at the end of the period. This is the value that the system compares with the rising and falling thresholds. The value during the current sampling period is not made available until the period is completed, and remains available until the next period completes.	`value` *`text`*

Field name	Description	Filter name
`Rising Triggered Count Since Last Alarm`	The number of times a poll has determined a rising threshold since the last alarm. Since polls can occur more often than alarms, this count is the number of times a poll has detected an alarm threshold crossing since the last time the alarm has been sent. This value is reset whenever an alarm is sent.	`risingTriggeredCountSince LastAlarm` *`integer`*
`Falling Triggered Count Since Last Alarm`	The number of times a poll has determined a falling threshold since the last alarm. Since polls can occur more often than alarms, this count is the number of times a poll has detected an alarm threshold crossing since the last time the alarm has been sent. This value is reset whenever an alarm is sent.	`fallingTriggeredCountSince LastAlarm` *`integer`*
`Total Rising Triggered Count`	Total number of times a poll has determined a rising threshold crossing since monitoring of this row began.	`totalRisingTriggeredCount` *`integer`*
`Total Falling Triggered Count`	Total number of times a poll has determined a falling threshold crossing since monitoring of this row began.	`totalFallingTriggeredCount` *`integer`*
`Rising Alarm Generated Count`	The number of alarms sent because of one or more rising threshold crossings. Each time a row generates a rising threshold alarm, the count increments.	`risingAlarmGeneratedCount` *`integer`*
`Falling Alarm Generated Count`	The number of alarms sent because of one or more falling threshold crossings. Each time a row generates a falling threshold alarm, the count increments.	`fallingAlarmGeneratedCount` *`integer`*

Field name	Description	Filter name
`Can Exceed Rising`	The alarm is currently in a state where a rising alarm can be generated. After the first trigger of an rising alarm, alarms are not generated until the falling threshold is triggered. This value defaults to true so an alarm will be generated on the first poll that exceeds (greater than or equal to) the rising threshold.	`canExceedRising {true | false}`
`Can Exceed Falling`	The alarm is currently in a state where a rising alarm can be generated. After the first trigger of an rising alarm, alarms are not generated until the falling threshold is triggered. This value defaults to true so an alarm will be generated on the first poll that exceeds (greater than or equal to) the rising threshold.	`canExceedFalling {true | false}`

Associated MIB

`rmon.mib`

Web path

- nmon → alarm → result

Chapter 5. Event commands

Event generation description

When the N2000 Series generates system events, it stores the events in an internal event log. The event log can contain up to 512 entries or use up to 256K of memory space, whichever is less. When the log reaches the maximum size, the system begins overwriting existing messages, starting with the oldest messages. To view the contents of the event log on the system, use the `show event log` command.

The system also supports the syslog process (RFC 3164) to send event messages to remote syslog servers. The syslog protocol uses the User Datagram Protocol (UDP) to transmit messages to syslog servers. The typical UDP port number for event message transmission is 514.

You can configure the system to send events to up to 15 remote syslog servers.

Event severity levels

Events have the following severity levels (from highest severity to lowest).

Severity level and numeric value	Description
`Emergency(0)`	The system detected a serious problem. Further system operation can cause damage to the system or its surroundings. Power down the failed system and contact Sun Technical Support immediately.
`Alert(1)`	The system detected a problem that requires an operator's attention. Typically, the problem is a hardware failure, but it can also include software image loss or corruption (requiring a download of an image). An application's repetitive generation of critical events can also generate an alert event. In this situation, restarting the failed application was not sufficient to restore useful service; contact Sun Technical Support. Systems running with unresolved alert events are non-functional, but you should keep the power on to facilitate problem diagnosis and resolution.
`Critical(2)`	The system software detected a critical problem requiring an application restart. This can include events that the application or the operating system detected (for example, illegal memory accesses). The operating system attempts to restart the failed application and return to full service. Restarting the software may not be sufficient to restart the application. In this situation, the system attempts to reset hardware components and subsystems, or reboots the entire system. Systems running with unresolved critical events are unlikely to provide full service.
`Error(3)`	The system detected a significant problem, but continues operating. This problem can result from dropped packets or configuration and management actions. Contact Sun Technical Support.

Severity level and numeric value	Description
`Warning(4)`	The system detected a problem, but was able to recover. This problem can result from configuration or management actions.
`Notice(5)`	The system detected a normal operational activity, but the event may be of interest to the operator. These events include ports coming up or going down, configuration of services for vSwitches, or processes being started.
`Informational(6)`	The system detected a normal operational activity. Usually, these events are of interest only if they are needed at a later time to investigate a problem.
`Debug(7)`	The system detected normal operational activities and unexpected events. The event messages can contain detailed information that corresponds to the specific implementation of a given component or process in the system. These messages are intended for use by Sun Technical Support when diagnosing system problems.

Event command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`event` *`commandname`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Event command summary

Table 5-1 lists and briefly describes the event commands.

Table 5-1. Event command summary

Command name	Description
`event (root)`	Configure the system to store events of specific severity levels in the event log.
`filterProfile`	Create and name a filter profile.
`filterProfile rule`	Add rules to a filter profile.
`show event`	Display the number of events the system generated and the lowest severity level for events stored in the event log.
`show event log`	Display the contents of the event log.
`show filterProfile`	Display filter profile names and descriptions.
`show filterProfile rule`	Display filter profile rules.
`show syslog`	Display the syslog process configuration.
`show vSwitch event summary`	Display the number of vSwitch events the system generated and the lowest severity level for events stored in the event log.
`show vSwitch event log`	Display the contents of the vSwitch event log.
`syslog`	Configure the system to use the syslog process to send event messages to one or more syslog servers.

event (root)

Purpose

Specifies whether events are saved on disk as persistent, and if so, the characteristics of the file in which they are saved. Through this command you can enable and disable persistent event logging, as well as set filter profiles for event logs on disk and in memory.

Access mode

enable

Syntax

```
event
   [fileLogSize logSize]
   [fileLogFilter filterProfile]
   [fileLogName text]
   [logFilter text]
```

Arguments

Argument name	Description
`fileLogSize` *`logSize`*	**Optional.** Sets the size of the persistent event log, in bytes. The events that are saved to the log, determined by the `systemLogLevel` argument, are maintained in the log even if the system reboots. Valid range is 16384 through 100,000,000; the default value is 65536 bytes.
`fileLogFilter` *`filterName`*	**Optional.** Specifies the name of the filter profile to use for those events that are saved to disk. If you do not specify a filter name, the system saves all events of the correct log level.
`fileLogName` *`text`*	**Optional.** Specifies a filename for those events that are saved to disk. Enter up to 65 alphanumeric characters; do not use any backslash (/) characters. To disable the persistent event log so that no events are saved to disk (events are only saved in memory), set the log name to two single quotation marks ('').
`logFilter` *`text`*	**Optional.** Specifies the name of the filter profile to use for those events that are saved in memory. (These events are lost when the system reboots.) If you do not specify a filter name, the system saves all events of the correct log level.

Delete filters

```
no event summary
   [fileLogSize logSize]
   [fileLogFilter filterProfile]
   [fileLogName text]
   [logFilter text]
```

Example

The following example logs events of log level warning and above to a file on a disk named `Warnings`. It applies the filter named `matchAll`.

```
sun> enable
sun# event
sun(event)# summary fileLogFilter matchAll fileLogName Warnings
```

Associated MIB

```
event.mib
```

Web path

- switchServices → event → summary → modify

filterProfile

Purpose

Configures a new filter profile with the specified name. This filter can be used by the N2000 Series to control which events are written to the event log, and whether the log is saved to disk or stored in memory. Once a profile is created, use the `filterProfile rule` command to add rules that control which events are stored.

Access mode

enable

Syntax

```
event filterProfile
   name filterName
   [description text]
```

Arguments

Argument name	Description
`name` *`filterName`*	Specifies a name for a filter. Once named, you can use the `event filterProfile rule` command to assign rules to the filter. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`description` *`text`*	**Optional.** Assigns a text description to the filter profile. The description can be up to 64 characters, and is displayed with output from the `show filterProfile` command. If the description includes spaces, enclose it in quotation marks.

Delete filters

```
no event filterProfile
    name filterName
    [description text]
```

Example

The following example deletes a filter profile named Warnings:

```
sun> enable
sun# event
sun(event)# no filterProfile Warnings
```

Associated MIB

```
event.mib
```

Web path

- switchServices → event → filter profile → add
- switchServices → event → filter profile → modify
- switchServices → event → filter profile → delete

filterProfile rule

Purpose

Adds rules to a named filter profile. Create the filter profile with the `filterProfile` command. A filter profile and its rules control event filtering on the N2000 Series system, which determines the events you will see.

Defining event filters is a two-step process. First you define the filter profile and its rules. Then, you apply the filter profile to one of the system loggers. These applications receive events, and then store, transmit, or process them. The N2000 Series has four different loggers. Each logger has a default profile and default rules defined. The table below describes the loggers and their defaults.

Table 5-2. System loggers and profiles

Logger name	Description	Default profile name	Default rule
`in memory event log`	Saves a limited number of events in the system's memory. The in memory event log is often referred to as the event log. You view the in memory log with the `show event log` command.	`defaultLog`	Drop any event with a log level debug.
`file log`	Saves a configurable number of events to a file on the persistent storage device (that is, flash memory).	`defaultFile`	Drop any event with a log level of warning and below.

Table 5-2. System loggers and profiles (continued)

Logger name	Description	Default profile name	Default rule
`syslog host`	Transmits events out the system's management interface as UDP messages. You must be running a port scanner program, for example UDPLISTEN, to receive the events.	`defaultSyslog`	Drop any event with a log level of debug.
`trap destination`	Acts as a logger when events are sent out as traps. You must be running a trap receiver to receive these traps.	`defaultTrapD`	Drop any event with a log level of warning and below.

When the system generates an event, it applies the filter profile rules, starting with the rule with the lowest position value and continuing, in ascending order, until a match is made. If a rule matches the event, the event is either sent or dropped depending on the value of the `action` parameter. There is no further rule processing once a match is found. If no matching rule is found for the event, the action is `drop`.

Using this command, specify the lowest severity level for events that you want in the event log. The system stores events with the specified severity level *and* all events with a higher severity level in the log. For example, if you specify `error` as the value for the `systemLogLevel` argument, the system stores events with `error`, `critical`, `alert`, and `emergency` severity levels in the event log. The system does not store events with a severity level of `notice`, `informational`, or `debug`.

Access mode

enable

Syntax

To create filter rules:

```
event filterProfile rule
   name filterName
   position integer
   action {send | drop}
   [all {false | true}]
   [vSwitchName text]
   [vRouterName text]
   [eventSubsystem subsystemName]
   [eventId id]
   [logLevel {emergency | alert | critical | error | warning | notice
      | informational | debug | all}]
```

To modify filter rules:

```
event filterProfile rule
   name filterName
   [position integer]
   action {send | drop}
   [all {false | true}]
   [vSwitchName text]
   [vRouterName text]
   [eventSubsystem subsystemName]
   [eventId id]
   [logLevel {emergency | alert | critical | error | warning | notice
      | informational | debug | all}]
```

Arguments

Argument name	Description
`name` *`filterName`*	Specifies the name of the filter profile to which this rule is being added. If you specify the name of a filter that does not exist, the system creates it.
`position` *`integer`*	Sets the position of this rule in the profile list. The system stops applying rules as soon as it makes it's first match, so position is significant. Enter a value between 1 and 1000; the default rule position is 100. The system evaluates rules from lowest to highest.
`action {send | drop}`	Specifies the action to take when an event matches a rule in the named profile. Events are sent on to memory and, if configured, the file log. If no rules match the event, the default action is `drop`.
`all {true | false}`	**Optional.** Sets whether to match all events, instead of setting specific rules. If you set this field to `true`, the system ignores additional fields within the named profile (`vSwitchName`, `vRouterName`, `eventSubsystem`, `eventId` and `logLevel`).The default setting is `false`.
`vSwitchName` *`name`*	**Optional.** Filters events based on the named vSwitch. By default, the filter accepts all vSwitches.
`vRouterName` *`name`*	**Optional.** Filters events based on the named vRouter. By default, the filter accepts all vRouters.
`eventSubsystem` *`subsystemName`*	**Optional.** Filters events based on the named subsystem. The list of can be found in the `show log` command description, or by using tab completion after entering the `eventSubsystem` keyword. Enter subsystems in a space separated list, enclosed in quotation marks. By default, the filter accepts all subsystems.
`eventId` *`id`*	**Optional.** Filters events based on the specified event ID. By default, the filter accepts all IDs.

Argument name	Description
`logLevel {emergency | alert | critical | error | warning | notice | informational | debug | all}`	**Optional.** Specifies the severity level that the `action` parameter acts on. When the `action` is `drop`, the system drops all events of the specified level and below. When the `action` is `send`, the system sends all events of the specified level and above. Valid values are: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: An event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. `all`: Drops or sends all log levels; effectively disables filtering based on `logLevel`. The default setting is `all`.

Delete filters

```
no event filterProfile rule
   name filterName
   position integer
   [action {send | drop}]
   [all {false | true}]
   [vSwitchName text]
   [vRouterName text]
   [eventSubsystem subsystemName]
   [eventId id]
   [logLevel {emergency | alert | critical | error | warning | notice
      | informational | debug | all}]
```

Example

The following example creates a rule in position 10 that drops events originating from NTP and VSRP:

```
sun> enable
sun# event
sun(event)# rule position 10 action drop all false eventSubsystem "NTP
VSRP"
```

Associated MIB

```
event.mib
```

Web path

- switchServices → event → filter profile → rule → add
- switchServices → event → filter profile → rule → modify
- switchServices → event → filter profile → rule → delete

show event

Purpose

Displays the settings for the event file and the total number of each event that the system generated. The counts reflect the actual number of events generated even if the system does not store the events in the log or send them to remote syslog servers. This command does not support field filtering.

Access mode

user

Syntax

```
show event
```

Sample output

```
sun> event
sun (event)> show
File Log Size:                  65536
File Log Filter:                defaultFile
File Log Name:                  eventlog.txt
Log Filter:                     defaultLog
Total Event Count:              349
Emergency Event Count:          0
Alert Event Count:              0
Critical Event Count:           0
Error Event Count:              0
Warning Event Count:            1
Notice Event Count:             21
Informational Event Count:      82
Debug Event Count:              245
```

Output description

Field name	Description
File Log Size	The size, in bytes, of the persistent event log.
File Log Filter	The name of the filter profile applied to the persistent log file.

Field name	Description
File Log Name	The name of the log file to which events are written. If the name displays as two single quotation marks (''), persistent logging is disabled.
Log Filter	The name of the filter profile applied to the log file stored in memory.
Total Event Count	The total number of events that occurred since the last time the system rebooted.
Emergency Event Count	The total number of events with a severity level of Emergency.
Alert Event Count	The total number of events with a severity level of Alert.
Critical Event Count	The total number of events with a severity level of Critical.
Error Event Count	The total number of events with a severity level of Error.
Warning Event Count	The total number of events with a severity level of Warning.
Notice Event Count	The total number of events with a severity level of Notice.
Informational Event Count	The total number of events with a severity level of Informational.
Debug Event Count	The total number of events with a severity level of Debug.

Associated MIB

event.mib

Web path

- switchServices → event → summary

show event log

Purpose

Displays the contents of the event log. The system sends event messages from this log to all configured syslog servers on remote systems.

Access mode

user

Syntax

```
show event log
```

Sample output

```
sun> event
sun (event)> show log
ID     Date       Level         Subsystem  Message
22637  1/1/1970-0 informational MgmtAudit  101e7: telnet ::
       7:53:24                             /telnet_192.168.209.76:1064:0x2301dd
                                           :: admin :: event  :: Pending
22636  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:50                             N2000 :: Logging out ::
22635  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:49                             N2000 :: vSwitch COLUMBUS :: Success
22634  1/1/1970-0 debug         Provisioni 10140: SMF_ROUTINE_CALLED:
       7:52:49                  ng         subscriberStatisticsTable :: get_imp
22633  1/1/1970-0 debug         Provisioni 10140: SMF_ROUTINE_CALLED:
       7:52:49                  ng         subscriberStatisticsTable :: get_imp
22632  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:49                             N2000 :: vSwitch COLUMBUS :: Pending
22631  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:49                             N2000 :: vSwitch COLUMBUS :: Success

Press <return> or <space bar> for more, or 'q' to quit...
```

Output description

Field name	Description	Filter name
`ID`	A numeric identifier the system assigns to the event message based on the order in which the event occurred.	`id` *`integer`*
`Date`	The date and time when the event occurred. The format is: `MM/DD/YYYY-HH:MM:SS`	`date` *`"MM/DD/YYYY-HH:MM:SS"`*
`Level`	The severity level associated with the event. Possible values, from the highest severity level to the lowest, are: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; Immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: An event occurred that can cause a loss of some system functionality. Administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: Debug-level event; for Sun technical support use only.	`level {emergency | alert | critical | err | warning | notice | informational | debug}`
`Subsystem`	The system component that generated the event. See Table 5-3 for a list of subsystems that generate events.	`subSystem` *`subSystemName`*
`Message`	The event message text.	`message` *`text`*

Field name	Description	Filter name
No field name is displayed	The module that generated the event, either systemBoard (0), functionCard1 (1), or functionCard2 (2). Although the system does not display the module ID in the internal event log, you can use the module number to filter the output. The module filter is for advanced troubleshooting purposes.	`module {0 | 1 | 2}`
No field name is displayed	The process ID (PID) for the process that called this event. Although the system does not display the PID in the internal event log, you can use the PID to filter the event log output. You can find the PID by looking at the messages that the N2000 sends to a syslog server. The PID filter is for advanced troubleshooting purposes.	`pid` *`ID`*

Subsystems that generate events

The following table lists the N2000 Series subsystems that can generate events.

Table 5-3. Subsystem description

This subsystem:	Is responsible for:
AAA	User authentication, authorization, and accounting services
AppControl	Operating system resource management
AppMgr	Operating system process management
CertKeyMgr	Certificate and Key Manager utility
CertVerifier	Verification of re-encrypted certificates
ChassisMgr	Managing the system motherboard and the function cards
CommandLine	Management of TCL commands
ControlPlane	IP Forwarding control
EmbMgmt	Internal interface with the management configuration broker
Eth	Ethernet services
EventMgr	Event generation

Table 5-3. Subsystem description (continued)

This subsystem:	Is responsible for:
FPGALM	Hardware image management
FTP	File Transfer Protocol services
HandshakeCrypto	Handshake cryptography hardware
HealthChecks	Health check services
HwDrivers	Managing hardware drivers
HWMon	Monitoring hardware states
Install	Installation mechanism
IP	IP services
IPForwarding	Static route operations
IRDP	ICMP Router Discovery Protocol services
LAG	Link aggregation group services
LicenseMgr	Software license management
LoadBalance	Load balancing services
Logger	Displaying and formatting the event log
MgmtAudit	Managing the internal audit log
MgmtBroker	Maintaining system configuration information
MgmtJournal	Providing backup and redundant configuration information
MgmtLog	Management Broker debug log — for Sun technical support use
MgmtServer	Maintaining system configuration information
NetProcessor	Network processor operations
NMI	Network Maskable Interrupt internal verification
NMON	Remote network monitoring services
NOSINET	Internal IP stack
NTFTP	TFTP server
NTP	Network Time Protocol services
NTPK	Network Time Protocol kernel
NTPM	Network Time Protocol manager

Table 5-3. Subsystem description (continued)

This subsystem:	Is responsible for:
ObjSwitching	Object switching operations
OSPF	Open Shortest Path First Protocol services
PCI	PCI driver
Provisioning	Creating vSwitch applications and invoking the Resource Manager
PRSDRV	Switch fabric driver
PWR_INT	Power interrupt
ResourceMgr	vSwitch partitioning of system resources
RFCTrap	Standard RFC traps
RIP	Routing Information Protocol services
Routing	Routing services
RTC	Real-time clock operation
Scheduler	CRON configuration change scheduler
SchemaParser	Internal database parsing
SNMP	Simple Network Protocol Management Protocol services
Socket	Socket interfaces
SSHd	Secure Shell Server operations
SSL Handshake	Secure Sockets Layer negotiation
SSLCrypto	Encryption and decryption engine
SysMonitor	Monitoring process states
System	Debug events — for Sun technical support use
TELNETD	Telnet server
TFTP	Trivial File Transport Protocol client operations
TFTPd	Trivial File Transport Protocol server operations
TideRunner HW	Resource management for the function cards
TideRunnerDb	Debug messages for the function cards — for Sun technical support use

Table 5-3. Subsystem description (continued)

This subsystem:	Is responsible for:
Trap	SNMP traps
VersionCtrl	Software version control
VLAN	Virtual LAN services (for example, VLAN creation, deletion, and spanning tree configurations)
VPN	Virtual Private Network services (for example, negotiating VPN tunnels, supplying local IP addresses for VPN clients, and VPN encryption/decryption)
VRRP	VIrtual Router Redundancy Protocol services
VSRP	Virtual Service Redundancy Protocol services
WebServer	Web server operations

Associated Mib

`event.mib`

Web path

- switchServices → event → log

show filterProfile

Purpose

Displays the name and description of each configured filter profile.

Access mode

user

Syntax

```
show event filterProfile
```

Sample output

```
sun> event
sun(config-event)# show filterProfile
Name                        Description
defaultFile                 default saved filter
defaultLog                  default log filter
defaultSyslog               default syslog filter
defaultTrapd                default trapd filter
sun(config-event)#
```

Associated MIB

```
event.mib
```

Web path

- switchServices → event → filter profile

show filterProfile rule

Purpose

Displays the settings of either all rules or a specific rule if you are within the context of a specific filter profile.

Access mode

user

Syntax

```
show event filterProfile rule
```

Sample output

```
sun> event
sun (event)> show filterProfile rule
Name          Position     Action    Summary
defaultFile   130          drop      level<=warning
defaultFile   200          send      all
defaultLog    90           drop      level<=debug
defaultLog    100          send      all
defaultSyslog 90           drop      level<=debug
defaultSyslog 100          send      all
defaultTrapd 90            drop      level<=warning
defaultTrapd 100           send      all
```

Output description

Field name	Description
`Name`	The name of the filter profile for which rules are being displayed.
`Position`	The position of this rule in the profile list. A value between 1 and 1000; the default rule position is 100. The system evaluates rules from lowest to highest.
`Action`	The action the rule applies when an event matches a rule in the named profile. If the `action` is `send`, events are sent on to memory and, if configured, the file log. If no rules match the event, the default action is `drop`.
`Summary`	A system-generated summary that lists, for the named filter profile, any rule settings that are not defaults.

Associated MIB

`event.mib`

Web path

- switchServices → event → filter profile → rule

show syslog

Purpose

Displays the current configurations for syslog servers to which the system sends event messages. If you do not specify a port, the system uses the default port value of 514.

Access mode

user

Syntax

```
show event syslog
```

Sample output

```
sun> event
sun (event)> show syslog
SysLog           Syslog
Host             Port         Filter       Facility
192.168.215.38   50481        defaultSyslog local0
```

Output description

Field name	Description	Filter name
SysLog Host	The IP Address of a syslog server.	host *ipAddress]*
SysLog Port	The UDP port that the syslog server uses to listen for event messages. If you do not specify a port, the system uses port 514 as the default value.	port *portNumber*
Filter	The name of the event filter that is being applied to this syslog server	filter *filterName*
Facility	The user facility code assigned to syslog messages that the system sends to remote syslog hosts.	facility {local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7}

Associated MIB

`event.mib`

Web path

- switchServices → event → syslog

show vSwitch event summary

Purpose

Displays the settings for the event file and the total number of each event that the system generated per named vSwitch. The counts reflect the actual number of events generated even if the system does not store the events in the log or send them to remote syslog servers. This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch` *`vSwitchname`* `event summary`

Sample output

```
sun> show vSwitch e-commerce event summary
Total Count:                  349
Emergency Count:              0
Alert Count:                  0
Critical Count:               0
Error Count:                  0
Warning Count:                1
Notice Count:                 21
Informational Count:          82
Debug Event Count:            245
```

Output description

Field name	Description
`Total Count`	The total number of events that occurred.
`Emergency Count`	The total number of events with a severity level of `Emergency`.
`Alert Count`	The total number of events with a severity level of `Alert`.
`Critical Count`	The total number of events with a severity level of `Critical`.

Field name	Description
`Error Count`	The total number of events with a severity level of `Error`.
`Warning Count`	The total number of events with a severity level of `Warning`.
`Notice Count`	The total number of events with a severity level of `Notice`.
`Informational Count`	The total number of events with a severity level of `Informational`.
`Debug Count`	The total number of events with a severity level of `Debug`.

Associated MIB

`event.mib`

Web path

- vSwitch → *vSwitchname* → event → summary

show vSwitch event log

Purpose

Displays the contents of the event log per named vSwitch. The system sends event messages from this log to all configured syslog servers on remote systems.

Access mode

user

Syntax

`show vSwitch` *`vSwitchname`* `event log`

Sample output

```
sun > show vSwitch columbus event log
ID     Date       Level         Subsystem  Message
22637  1/1/1970-0 informational MgmtAudit  101e7: telnet ::
       7:53:24                             /telnet_192.168.209.76:1064:0x2301dd
                                           :: admin :: event  :: Pending
22636  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:50                             N2000 :: Logging out ::
22635  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:49                             N2000 :: vSwitch COLUMBUS :: Success
22634  1/1/1970-0 debug         Provisioni 10140: SMF_ROUTINE_CALLED:
       7:52:49                  ng         subscriberStatisticsTable :: get_imp
22633  1/1/1970-0 debug         Provisioni 10140: SMF_ROUTINE_CALLED:
       7:52:49                  ng         subscriberStatisticsTable :: get_imp
22632  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:49                             N2000 :: vSwitch COLUMBUS :: Pending
22631  1/1/1970-0 informational MgmtAudit  a01e2: console :: localhost ::
       7:52:49                             N2000 :: vSwitch COLUMBUS :: Success

Press <return> or <space bar> for more, or 'q' to quit...
```

Output description

Field name	Description	Filter name
`ID`	A numeric identifier the system assigns to the event message based on the order in which the event occurred.	`id` *`integer`*
`Date`	The date and time when the event occurred. The format is: `MM/DD/YYYY-HH:MM:SS`	`date` *`"MM/DD/YYYY-HH:MM:SS"`*
`Level`	The severity level associated with the event. Possible values, from the highest severity level to the lowest, are: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; Immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: An event occurred that can cause a loss of some system functionality. Administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: Debug-level event; for Sun technical support use only.	`level {emergency | alert | critical | err | warning | notice | informational | debug}`
`Subsystem`	The system component that generated the event. See Table 5-4 for a list of subsystems that generate events.	`subSystem` *`subSystemName`*
`Message`	The event message text.	`message` *`text`*

Field name	Description	Filter name
No field name is displayed	The module that generated the event, either systemBoard (0), functionCard1 (1), or functionCard2 (2). Although the system does not display the module ID in the internal event log, you can use the module number to filter the output. The module filter is for advanced troubleshooting purposes.	`module {0 | 1 | 2}`
No field name is displayed	The process ID (PID) for the process that called this event. Although the system does not display the PID in the internal event log, you can use the PID to filter the event log output. You can find the PID by looking at the messages that the N2000 sends to a syslog server. The PID filter is for advanced troubleshooting purposes.	`pid` *`ID`*

Subsystems that generate events

The following table lists the N2000 Series subsystems that can generate events:

Table 5-4. Subsystem description

This subsystem:	Is responsible for:
AAA	User authentication, authorization, and accounting services
AppControl	Operating system resource management
AppMgr	Operating system process management
CertKeyMgr	The Certificate and Key Manager utility
CertVerifier	Verification of re-encrypted certificates
ChassisMgr	Managing the system motherboard and the function cards
CommandLine	Management of Tcl commands
ControlPlane	IP forwarding control
EmbMgmt	Internal interface with the management configuration broker
Eth	Ethernet services
EventMgr	Event generation

Table 5-4. Subsystem description (continued)

This subsystem:	Is responsible for:
FPGALM	
FTP	File Transfer Protocol services
HandshakeCrypto	
HealthChecks	Health check services
HwDrivers	Managing hardware drivers
HWMon	Monitoring hardware states
Install	Installation mechanism
IP	IP services
IPForwarding	Static route operations
IRDP	ICMP Router Discovery Protocol services
LAG	Link aggregation group services
LicenseMgr	License management service
LoadBalance	Load balancing services
Logger	Displaying and formatting the event log
MgmtAudit	Managing the internal audit log
MgmtBroker	Maintaining system configuration information
MgmtJournal	Providing backup and redundant configuration information
MgmtLog	Management Broker debug log — for Sun technical support use
MgmtServer	Maintaining system configuration information
NetProcessor	Network processor operations
NMI	Network Maskable Interrupt internal verification
NOSINET	
NTFTP	
NTP	Network Time Protocol services
NTPK	
NTPM	
ObjSwitching	Object switching operations

Table 5-4. Subsystem description (continued)

This subsystem:	Is responsible for:
OSPF	Open Shortest Path First Protocol services
PCI	
Provisioning	Creating vSwitch applications and invoking the Resource Manager
PRSDRV	
PWR_INT	
ResourceMgr	vSwitch partitioning of system resources
RFCTrap	Standard RFC traps
RIP	Routing Information Protocol services
RMON	
Routing	Routing services
RTC	Real-time clock operation
Scheduler	
SchemaParser	
SNMP	Simple Network Management Protocol services
Socket	Socket interfaces
SSHd	Secure Shell Server operations
SSL Handshake	Secure Sockets Layer negotiation
SSLCrypto	Encryption and decryption engine
SysMonitor	Monitoring process states
System	Debug events — for Sun technical support use
TELNETD	
TFTP	Trivial File Transport Protocol client operations
TFTPd	Trivial File Transport Protocol server operations
TideRunner HW	Resource management for the function cards
TideRunnerDb	Debug messages for the function cards — for Sun technical support use

Table 5-4. Subsystem description (continued)

This subsystem:	Is responsible for:
Trap	SNMP traps
VersionCtrl	
VLAN	Virtual LAN services (for example, VLAN creation, deletion, and spanning tree configurations)
VPN	Virtual Private Network services (for example, negotiating VPN tunnels, supplying local IP addresses for VPN clients, and VPN encryption/decryption)
VRRP	
VSRP	Virtual Service Redundancy Protocol services
WebServer	Web server operations

Associated MIB

`event.mib`

Web path

- vSwitch → *vSwitchname* → event → log

syslog

Purpose

Configures the system to send events to one or more remote syslog servers. You can configure the system to use up to 15 syslog servers.

The no form of this command deletes one or more defined remote syslog servers. If you enter optional arguments, the CLI deletes the syslog configuration only if it matches all arguments. With the no form of the command, the required arguments are host and port.

Access mode

enable

Syntax

```
event syslog
   host ipAddress
   [port portNumber]
   [filter filterName]
   [facility {local0 | local1 | local2 | local3 | local4 | local5 |
      local6 | local7}]
```

Arguments

Argument Name	Description
host *ipAddress*	Configures the IP address of a syslog server to which the system sends events.
port *portNumber*	Specifies the port that the syslog server uses to listen for events. Valid values are from 1 through 65535. The default setting is 514.
filter *filterName*	**Optional**. Specifies the name of the event filter to apply to this syslog server.

Argument Name	Description
`facility {local0 | local2 | local3 | local4 | local5 | local6 | local7}`	**Optional**. Specifies the user facility for syslog messages. The facility helps to identify the system that generated the event. The facility is part of the priority value displayed at the beginning of a syslog message. The formula for calculating the priority is: `facility * 8 + severity level` The valid values are from `local0 through local7`. The default value is `local0`.

Delete filters

```
no event syslog
   host ipAddress
   port portNumber
   [filter filterName]
   [facility {local0 | local1 | local2 | local3 | local4 | local5 |
      local6 | local7}]
```

Example

```
sun> enable
sun# event
sun(config)# event syslog host 10.10.30.4 port 514 filter
defaultSyslog local3
sun(event)#
```

Associated MIB

```
event.mib
```

Web path

- switchServices → event → summary → modify

Chapter 6. SNMP and trap commands

SNMP description

The N2000 Series supports access using the Simple Network Management Protocol (SNMP). SNMP is a set of Internet protocols used to manage network devices. Using an SNMP application on a remote management station (referred to as an SNMP manager), you can communicate with the SNMP entity on the N2000 Series (the SNMP agent) to retrieve information about manageable objects on the switch as well as change configuration settings.

The N2000 Series supports the following:

- SNMPv1, SNMPv2c, and SNMPv3
- Standard MIB-II objects
- Enterprise objects
- GET, GETNEXT, SET, and TRAP commands.

Note: The N2000 Series does not support INFORM notifications or the SNMPv3 VACM MIB.

For a more detailed description of SNMP configuration and use, see the *Sun N2000 Series Release 2.0 – System Administration Guide*.

SNMPv3 on the N2000 Series

For SNMPv3 communication, the N2000 Series supports the User Access Model (USM). The USM provides secure communication between SNMP entities (SNMP agents and managers) through the use of authentication and encryption of SNMP packets.

In addition to the security that the USM provides, you can associate each SNMP user with an N2000 Series user profile, a vSwitch, and a vRouter to provide access control. You can assign read-write or read-only access to a specific vSwitch and vRouter.

Traps on the N2000 Series

The N2000 Series generates standard SNMP RFC traps as well as switch-specific traps. You can use the trap commands to configure the system to forward the traps to up to 10 remote trap hosts. You can also configure the types of switch-specific traps that you forward. You can send the following:

- Traps that occur when authentication fails for an SNMP message that the system receives
- Traps that the system generates for events that the system can log in the system event log

The N2000 Series automatically sends standard SNMP traps to trap hosts. You do not need to configure anything to send these traps.

SNMP and trap command paths

The command names in this chapter show you how to execute the commands from within the following command mode:

```
switchServices snmp commandName
switchServices trap commandName
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

SNMP and trap command summary

Table 6-1 lists and briefly describes the SNMP and trap commands.

Command name	Description
`show snmp`	Display the current SNMP administrative status and port configuration.
`show snmp stats`	Display statistics about SNMP packet processing on the N2000 Series.
`show snmp systemInfo`	Display information about the SNMP entity on the N2000 Series.
`show snmp user`	Display information about the configured SNMP users.
`show trap`	Display the current configuration for generating authentication failure traps.
`show trap destination`	Display a list of configured trap destination hosts.
`snmp (root)`	Configure the N2000 Series to receive SNMP packets and the port it uses for SNMP communication.
`snmp systemInfo`	Configure SNMP contact information for the SNMP agent on the system.
`snmp user`	Configure an SNMP user that can perform SNMP operations on the system.
`trap (root)`	Configure the N2000 Series to generate traps related to authentication failures.
`trap authenticationFailureTrap`	Configure the N2000 Series to generate traps related to authentication failures.
`trap destination`	Configure a remote host to which the switch sends traps.
`trap systemEvtTrap`	Configure the N2000 Series to generate traps for system events.

SNMP and trap basic configurations

SNMP basic configuration

Follow the steps below to configure SNMPv1 or SNMPv2c access on the N2000 Series.

Table 6-1. Steps for configuring SNMPv1 or SNMPv2c access

Step	Action
1.	Enable SNMP communication using the `switchServices snmp (root)` command.
2.	Enter a community string as the user name for the SNMP user, using the `snmp user` command.
3.	Specify `community` as the authentication method, using the `snmp user` command.
4.	Associate a user profile with the SNMP user, using the `snmp user` command.
5.	Associate a vSwitch and a vRouter with the SNMP user, using the `snmp user` command. You need to create a separate user entry for each vSwitch and vRouter that you want an SNMP user to access. **Reference:** See the *Sun N2000 Series Release 2.0 – System Administration Guide* for additional details about creating SNMP user entries.

Follow the steps below to configure SNMPv3 access on the N2000 Series.

Table 6-2. Steps for configuring SNMPv3 access

Step	Action
1.	Enable SNMP communication using the `snmp (root)` command.
2.	Enter the SNMPv3 user name as the user name for the SNMP user, using the `snmp user` command.
3.	Specify `usm` as the authentication method, using the `snmp user` command.
4.	Associate a user profile with the SNMP user, using the `snmp user` command.

Table 6-2. Steps for configuring SNMPv3 access (continued)

Step	Action
5.	Associate a vSwitch and a vRouter with the SNMP user, using the `snmp user` command. In addition, you can use the SNMPv3 context name in your SNMP application to specify a vSwitch and a vRouter. See the *Sun N2000 Series Release 2.0 – System Administration Guide* for details.
6.	The system prompts you to do the following: a. Specify the authoritative engine ID. Press [Return] to use the local engine ID. b. Select an authentication protocol: `none, md5, or sha`. If you select `none`, the configuration is complete. c. Enter the authentication password. d. Specify the privacy protocol: `des` or `none`. If you select `none`, the configuration is complete. e. Enter the privacy password.

Trap basic configuration

Use the `trap` command to configure the types of traps that the system generates, and if required, remote hosts to which the switch forwards the traps. Follow the steps below to configure traps on the N2000 Series.

Table 6-3. Steps for configuring traps

Step	Action
1.	Specify whether you want the system to generate traps for authentication failure, using the `trap (root)` command.
2.	Specify whether you want the system to generate traps for system events, using the `trap systemEvtTrap` command.
3.	Configure a remote trap destination host that receives traps that the switch generates. Use the `trap trap destination` command.

show snmp

Purpose

Displays the administrative state of the SNMP configuration and the port number that the system uses for SNMP communication. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices snmp
```

Sample output

```
sun> switchServices
sun(switchServices)> show snmp
Administrative State: enabled
SNMP Port:            161
Audit Logging:        on
```

Output description

Field Name	Description
`Administrative State`	The administrative state of SNMP communications on the system, either `enabled` or `disabled`.
`SNMP Port`	The number of the port the system uses to listen for SNMP packets.
`Audit Logging`	Indicates whether the system generates an event when the SNMP configuration changes.

Associated MIB

`snmpconfig.mib`

Web path

- switchServices → snmp

show snmp stats

Purpose

Displays statistics for SNMP packet processing that the SNMP entity on the N2000 Series performs. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices snmp stats
```

Sample output

```
sun> switchServices
sun(switchServices)> show snmp stats
Inpkts:                      0
InBadVersions:               0
InBadCommunityNames:         0
InBadCommunityUses:          0
InASNParseErrs:              0
SilentDrops:                 0
ProxyDrops:                  0
Unknown security models:     0
Invalid msgs:                0
Unknown PDU handlers:        0
Unsupported security levels: 0
Not in time window:          0
Unknown user names:          0
Unknown engine ids:          0
Wrong digests:               0
Decryption errors:           0
```

Output description

Field name	Description
Inpkts	The total number of messages that the SNMP entity on the N2000 Series received from the transport service.
InBadVersions	The total number of packets that the SNMP entity on the N2000 Series received for an SNMP version that it does not support.

Field name	Description
InBadCommunityNames	The total number of packets that the SNMP entity on the N2000 Series received that contains an unknown community name. The system expects to receive SNMP packets that contain the same community string configured for the associated SNMP user.
InBadCommunityUses	The total number of packets that the SNMP entity on the N2000 Series received for an SNMP operation that it does not allow for the SNMP community specified in the packet.
InASNParseErrs	The total number of ASN.1 or Basic Encoding Rules (BER) errors that the SNMP entity on the N2000 Series encountered while decoding SNMP packets that it received.
SilentDrops	The total number of Get and Set PDUs that the SNMP entity on the N2000 Series received and dropped silently. The SNMP entity drops the PDUs because the size of a reply containing an alternate Response-PDU with an empty variable-bindings field was greater than either a local constraint or the maximum message size associated with the originator of the request.
ProxyDrops	The total number of Get and Set PDUs that the SNMP entity on the N2000 Series received and dropped silently. The SNMP entity drops the PDUs because the transmission of the (possibly translated) packet to a proxy target failed in a manner (other than a time-out) such that the SNMP entity on the N2000 Series could not return a Response-PDU.
Unknown security models	The total number of packets that the SNMP entity on the N2000 Series received and dropped because the packets referenced a security model that the SNMP entity does not support or recognize.
Invalid msgs	The total number of packets that the SNMP entity on the N2000 Series received and dropped because the packets contained invalid or inconsistent components.
Unknown PDU handlers	The total number of packets that the SNMP entity on the N2000 Series received and dropped because it could not pass the PDU in the packet to an application responsible for handling the PDU type. (that is, no SNMP application had registered for the proper combination of the contextEngineID and the pduType).
Unsupported security levels	The total number of packets that the SNMP entity on the N2000 Series received and dropped because they requested a security level that the SNMP entity did not recognize or the security level was unavailable.
Not in time window	The total number of packets that the SNMP entity on the N2000 Series received and dropped because they were not delivered within the SNMP engine's acceptable delivery delay timeframe.

Field name	Description
`Unknown User names`	The total number of received packets that the SNMP entity on the N2000 Series received and dropped because they referenced a user that the SNMP entity did not recognize.
`Unknown engine ids`	The total number of packets that the SNMP entity on the N2000 Series received and dropped because they referenced an EngineID that was not known to the SNMP engine.
`Wrong digests`	The total number of received packets that the SNMP entity on the N2000 Series dropped because they did not contain the expected digest value.
`Decryption errors`	The total number of received packets that the SNMP entity on the N2000 Series dropped because it could not decrypt the packets.

Associated MIB

```
snmpv2.mib
snmp-mpd.mib
snmp-user-base-sm.mib
```

Web path

- switchServices → snmp → stats

show snmp systemInfo

Purpose

Displays contact and operational information for the SNMP entity on the N2000 Series. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices snmp systemInfo
```

Sample output

```
sun> switchServices
sun(switchServices)> show snmp systemInfo
Description:            N2000 Series
Objectid:               1.3.6.1.4.1.8857.0.1
Uptime (100ths of sec): 14048811
Contact:                Steph Bradley
Name:                   CrackerJack
Location:               Floor 3, Outlet 4
Services:               70
Engine ID:              0x80002299030007820E0C0A
Engine Boots:           25
Engine Time (sec):      140487
SNMP Max Message Size:  2048
```

Associated MIB

```
snmpv2.mib
snmp-framework.mib
```

Web path

- switchServices → snmp → systemInfo

show snmp user

Purpose

Displays the configuration entries for existing SNMP users. SNMP users can use SNMP to retrieve or configure settings on the system depending on the profile, vSwitch, and vRouter associated with the SNMP user.

Access mode

user

Syntax

```
show switchServices snmp user
```

Sample output

```
sun> switchServices
sun(switchServices)> show snmp user
User Name:      public
Auth Method:    community
Profile:        systemAdmin
Virtualization: system:management
Address:        192.168.209.44
Mask:           255.255.255.255
```

Output description

Field name	Description	Filter name
`User Name`	The name associated with the SNMP user. For an SNMPv1 or v2c user, the user name is the community string that the SNMP manager sends in SNMP packets. For an SNMPv3 user, the user name is the SNMPv3 user name that the SNMP manager sends in SNMP packets.	`userName` *`text`*
`Auth Method`	The type of authentication used for the SNMP user: `community` for SNMPv1 or v2c users, `usm` for SNMPv3 users.	`authMethod {community | usm}`

Field name	Description	Filter name
`Profile`	The user profile associated with the SNMP user. The profile determines the type of authentication to use when the SNMP user tries to access the switch and the type of access the SNMP user has on the switch. The profiles are: • `systemAdmin` — read and write access for the system vSwitch. • `systemOperator` — read only access for the systemVswitch. • `vSwitchAdmin` — read and write access for a specific vSwitch. • `vSwitchOperator` — read only access for a specific vSwitch.	`profileName {systemAdmin | systemOperator | vSwitchAdmin | vSwitchOperator}`
`Virtualization`	The vSwitch and vRouter that the SNMP user can access, in the format `vSwitchName:vRouterName`.	`virtualization` *`vSwitchName:vRouterName`*
`Address`	The IP or subnet address associated with the SNMP user. The system compares this address with the source address of an SNMP packet to determine whether it came from an authorized user.	`address` *`ipAddress`*
`Mask`	The subnet mask that the system applies to an SNMP packet's address. The system then compares the packet's address with the address associated with an authorized SNMP user. If the values do not match, the system drops the SNMP packet.	`mask` *`ipAddress`*
`Authentication Protocol`	**Optional**: The protocol that the SNMP agent uses to authenticate the SNMP packet. You can select `none`, `md5`, or `sna`. This is only available if the user authentication method is set to `usm`.	`authenticationProtocol {none | md5 | sna}`
`Authentication Password`	**Optional**: The password that the authentication protocol uses for the hash mesage authentication code (HMAC). This is only available if the user authentication method is set to `usm`.	`authenticationPassword` *`password`*

Field name	Description	Filter name
`Privacy Protocol`	**Optional**: The protocol that the SNMP agent uses to encrypt te SNMP packet. You can select none or des. This is only available if the user authentication method is set to usm.	`privacyProtocol {none | des}`
`Privacy Password`	**Optional**: The privacy key that the privacy protocol uses for encryption. The key neds to be a minimum of 8 characters. This is only available if the user authentication method is set to usm.	`privacyPassword` *`password`*

Associated MIB

`snmpv2.mib`

Web path

- switchServices → snmp → user

show trap

Purpose

Displays the current N2000 Series trap configuration. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices trap
```

Sample output

```
sun> switchServices
sun(switchServices)> show trap
Admin State:                  disabled
Oper State:                   down
System Event Traps:           disabled
Authentication Failure Traps: disabled
Traps Sent:                   0
```

Output description

Field Name	Description
`Admin State`	The current administrative state for the N2000 Series trap generation process, either `enabled` or `disabled`.
`Oper State`	The current trap generation operational state, either `up` or `down`.
`System Event Traps`	The current configuration for generating system event traps, either `enabled` or `disabled`.
`Authentication Failure Traps`	The current configuration for generating SNMP authentication failure traps, either `enabled` or `disabled`.
`Traps Sent`	The number of traps the system sent to a trap destination.

Associated MIB

```
SNMPV2-MIB (RFC 1907)
```

Web path

- switchServices → trap

show trap destination

Purpose

Displays the configured remote trap hosts. The system sends the traps it generates to these trap hosts.

Access mode

user

Syntax

```
show switchServices trap destination
```

Sample output

```
sun> switchServices
sun(switchservices)> show trap destination
Index:        1
Dest IP Addr: 10.16.20.42
Dest Port:    162
User Name:    public
Snmp Version: SNMPv2c
Filter:       defaultTrapd
```

Output description

Field name	Description	Filter name
`Index`	The index number associated with the trap destination configuration.	`index` *`integer`*
`Dest IP Addr`	The IP address for the remote trap host.	`ipAddress` *`ipAddress`*
`Dest Port`	The port the remote trap host uses to receive traps.	`port` *`integer`*
`User Name`	The SNMP community string or SNMPv3 user name that the remote trap host requires for communication.	`userName` *`text`*
`Snmp Version`	The format the N2000 Series uses for traps that it sends to a remote trap host.	`SNMPVersion {SNMPv1 | SNMPv2c | SNMPv3}`

Field name	Description	Filter name
`Filter`	The filter profile that is used to filter system events that are sent to this trap destination. System events are checked against the filter to determine if the system event should be converted to a trap and sent to the trap destination.	`Filter {defaultTrapd | defaultFile | defaultLog | defaultSyslog | text}`

Associated MIB

`Sun-App-Switch-Trap-MIB.mib`

Web path

- switchServices → trap → destination

snmp (root)

Purpose

Configures the N2000 Series to enable SNMP communication and optionally sets the port the system uses to listen for SNMP packets. You can also turn on audit logging for the SNMP application.

This command also enters the `snmp` command mode.

Access mode

enable

Syntax

```
switchServices snmp
   [adminState {enabled | disabled}]
   [snmpPort text]
   [auditLogging {on | off}]
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	**Optional.** Specifies whether the system allows SNMP communications. If SNMP is administratively disabled, the N2000 Series does not send traps. The default setting is `disabled`.
`snmpPort` *`text`*	**Optional.** Specifies the port the system uses to listen for SNMP packets. Enter a port number between 1 and 65536; the default port number is 161. If you change this setting, you must set the `adminState` to `disabled` first, change the port number, and reset the `adminState` to `enabled`.
`auditLogging {on | off}`	**Optional.** Specifies whether you want to create an entry in the internal audit log whenever an SNMP configuration changes. The default setting is `on`. This setting affects all configuration entries for the SNMP application.

Example

The following example shows how to configure the N2000 Series to allow SNMP communication and change the port number to 176.

```
sun> enable
sun# switchServices
sun(switchServices)# show snmp
adminState: enabled
snmpPort: 161
sun (switchServices)# snmp adminState disabled snmpPort 176
sun (switchServices)# snmp adminState enabled
sun (switchServices)#
```

Associated MIB

```
Sun-App-Switch-Trap-MIB.mib
```

Web path

- switchServices → snmp

snmp systemInfo

Purpose

Configures SNMP contact information for the SNMP entity on the N2000 Series. Typically this information identifies who to contact in case of an emergency or problem.

Access mode

enable

Syntax

```
switchServices snmp systemInfo
   [contact text]
   [name text]
   [location text]
```

Arguments

Argument name	Description
`contact` *text*	**Optional**. Specifies the SNMP contact information. This is usually a person's name and the emergency telephone number, pager number, or email address. Use an alphanumeric string of up to 64 characters. If the contact string includes spaces, enclose the string in quotation marks (" ").
`name` *text*	**Optional**. Specifies a name for the switch. Typically, the name is the fully qualified name for the N2000 Series. Use an alphanumeric string of up to 64 characters. If the name string includes spaces, enclose the string in quotation marks (" ").
`location` *text*	**Optional.** Specifies the physical location of the switch. Typically, this is the street address where the switch is located. Use an alphanumeric string of up to 64 characters. If the location string includes spaces, enclose the string in quotation marks (" ").

Example

The following example shows how to configure the SNMP system contact information.

```
sun> enable
sun# switchServices
sun(switchServices)# snmp
sun(switchServices snmp)# systemInfo contact “Call Tom, 555-111-1111”
name mysystem.mydomain.com location “123 Any St, Anytown, USA”
```

Associated MIB

```
SNMPv2.mib
```

Web path

- switchServices → snmp → systemInfo

snmp user

Purpose

Configures an SNMP user who can perform SNMP operations on the N2000 Series. The user entry defines how the system authenticates SNMP packets and the type of access the user has on the system.

The no form of the command deletes a configured SNMP user. If you enter optional arguments, the CLI deletes the user entry only if it matches all arguments.

Access mode

enable

Syntax

To create an SNMP user configuration:

```
switchServices snmp user
   userName text
   authMethod {community | usm}
   profileName {systemAdmin | systemOperator | vSwitchAdmin |
      vSwitchOperator}
   virtualization vSwitch:vRouter
   [address ipAddress]
   [mask ipAddress]
```

To modify an SNMP user configuration:

```
switchServices snmp user
   userName text
   authMethod {community | usm}
   [profileName {systemAdmin | systemOperator | vSwitchAdmin |
      vSwitchOperator}]
   [virtualization vSwitch:vRouter]
   [address ipAddress]
   [mask ipAddress]
```

Arguments

Argument Name	Description
`userName` *`text`*	Configures the user name associated with the SNMP user. • If you are configuring an SNMPv1 or v2c user, enter the community string that the SNMP manager uses. • If you are configuring an SNMPv3 user, enter the SNMP user name configured on the SNMP manager.
`authMethod {community | usm}`	Configures the authentication method to use: • If you are configuring an SNMPv1 or V2c user, specify `community`. • If you are configuring an SNMPv3 user, specify `usm`. If you specify `usm`, the system prompts you to enter the following: a. Authentication protocol — The protocol that the SNMP agent uses to authenticate the SNMP packet. You can select `none`, `md5` or `sha`. If you select `none`, the configuration is complete. b. Authentication password — A password with a minimum of 8 characters that the authentication protocol uses for the hash message authentication code (HMAC). c. Privacy protocol — The protocol the SNMP agent uses to encrypt the SNMP packet. You can select `none` or `des`. If you select `none`, the configuration is complete. d. Privacy password — The privacy key with a minimum of 8 characters that the privacy protocol uses for encryption.
`profileName {systemAdmin | systemOperator | vSwitchAdmin | vSwitchOperator }`	Specifies the profile you want to assign to the user. The profile determines the type of access the user has: `systemAdmin` provides read and write access the system vSwitch. `systemOperator` provides read only access to the system vSwitch. `vSwitchAdmin` provides read and write access to a specified vSwitch. `vSwitchOperator` provides read only access to a specified vSwitch.
`virtualization` *`vSwitch:vRouter`*	Specifies the vSwitch and vRouter pair that the SNMP user can access.

Argument Name	Description
`address` *`ipAddress`*	**Optional**. Specifies the IP or subnet address for the remote management station from which the user is trying to access the N2000 Series. The default value is 0.0.0.0. Configuring this value provides an additional level of security for SNMPv1 and SNMPv2 operations. The system drops all SNMP packets that do not match the IP or subnet address associated with the user. If you do not provide this value, any user that knows the community string can access the system.
`mask` *`ipAddress`*	**Optional**. Specifies a subnet mask that the N2000 Series applies to the source IP or subnet address. The default value is 0.0.0.0. After applying the mask, the system compares the address associated with an SNMP packet with the address associated with the SNMP user. If the vales do not match, the system drops the SNMP packet.
`Authentication Protocol`	**Optional**: The protocol that the SNMP agent uses to authenticate the SNMP packet. You can select none, md5, or sna. This is only available if the user authentication method is set to usm.
`Authentication Password`	**Optional**: The password that the authentication protocol uses for the hash mesage authentication code (HMAC). This is only available if the user authentication method is set to usm.
`Privacy Protocol`	**Optional**: The protocol that the SNMP agent uses to encrypt te SNMP packet. You can select none or des. This is only available if the user authentication method is set to usm.
`Privacy Password`	**Optional**: The privacy key that the privacy protocol uses for encryption. The key neds to be a minimum of 8 characters. This is only available if the user authentication method is set to usm.
Authentication Protocol	**Optional**: The protocol that the SNMP agent uses to authenticate the SNMP packet. You can select none, md5, or sna. This is only available if the user authentication method is set to usm.
Authentication Password	**Optional**: The password that the authentication protocol uses for the hash mesage authentication code (HMAC). This is only available if the user authentication method is set to usm.

Privacy Protocol	**Optional**: The protocol that the SNMP agent uses to encrypt te SNMP packet. You can select none or des. This is only available if the user authentication method is set to usm.
Privacy Password	**Optional**: The privacy key that the privacy protocol uses for encryption. The key neds to be a minimum of 8 characters. This is only available if the user authentication method is set to usm.

Delete FIlters

```
no switchServices snmp user
   userName text
   authMethod {community | usm}
   [profileName {systemAdmin | systemOperator | vSwitchAdmin |
      SwitchOperator}]
   [virtualization vSwitch:vRouter]
   [address ipAddress]
   [mask ipAddress]
```

Example

The following example shows how to configure an SNMPv2c user. In this example, the `userName` argument specifies the community string that the SNMP user sends to the switch, the `authenticationMethod` is community, indicating an SNMPv2c user, and the `profileName`, vswitchAdmin, provides read and write access to the management vRouter in the system vSwitch.

```
sun> enable
sun# switchServices
sun(switchServices)# snmp
sun(switchServices snmp)# user UserName private AuthenticationMethod
community ProfileName vSwitchAdmin virtualization system:management
```

The following example shows how to configure an SNMPv3 user. In this example, the `userName`, user1, is the SNMPv3 username that the SNMP user sends to the switch, the `authenticationMethod` is usm, and the `profileName`, vSwitchAdmin, provides read and write access to a specific vSwitch.

```
sun(switchServices snmp)# user UserName user1 AuthenticationMethod usm
ProfileName vSwitchAdmin virtualization vs1:default

Please choose an authentication protocol (none, md5, sha) [none]
:md5
```

```
Please enter authentication protocol password (minimum 8 characters long)
:********

Please enter authentication protocol password again:
********
Please choose a privacy protocol (none, des) [none]: des

Please enter privacy protocol password (minimum 8 characters long):
********
Please enter privacy protocol password again:
********
sun(switchServices snmp)#
```

Associated MIB

`snmpv2.mib`

Web path

- switchServices → snmp → user → add
- switchServices → snmp → user → copy
- switchServices → snmp → user → modify
- switchServices → snmp → user → delete

trap (root)

Purpose

Configures N2000 Series trap generation.

Access mode

enable

Syntax

```
switchServices trap
   [adminState {enabled | disabled}]
   [systemEvtTraps {enabled | disabled}]
   [authenticationFailureTraps {enabled | disabled}]
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	**Optional.** Enables or disables the trap generation process on the N2000 Series. The default is `disabled`.
`authenticationFailureTraps {enabled | disabled}`	**Optional.** Specifies whether the SNMP entity on the N2000 Series should generate traps when authentication for an SNMP message fails. If `enabled`, the system generates the authentication failure traps; if `disabled`, the system does not generate the authentication failure traps. The default setting is `disabled`.
`systemEvtTraps {enabled | disabled}`	**Optional.** Specifies whether the system sends system events as traps to configured trap hosts. If `enabled`, the system generates a trap for each system event; if `disabled`, the system does not generate any traps for system events.

Example

The following example shows how to configure the system to generate authentication failure traps.

```
sun> enable
sun# switchServices
sun(switchServices)# trap
sun(switchServices trap)# authenticationFailureTraps enabled
sun(switchServices)# _
```

Associated MIB

```
SNMPV2-MIB (RFC 1907)
```

Web path

- switchServices → trap

trap authenticationFailureTrap

Purpose

Configures the system to generate traps when authentication for SNMP messages fails.

Access mode

enable

Syntax

```
switchServices trap authenticationFailureTrap
   [authenticationFailureTraps {enabled | disabled}]
```

Arguments

Argument name	Description
`authenticationFailureTraps {enabled | disabled}`	**Optional.** Specifies whether the SNMP entity on the N2000 Series should generate traps when authentication for an SNMP message fails. If `enabled`, the system generates the authentication failure traps; if `disabled`, the system does not generate the authentication failure traps. The default setting is `disabled`.

Associated MIB

`Sun-App-Switch-Trap-MIB.mib`

Web path

- switchServices → trap

trap destination

Purpose

Configures one or more remote hosts to which the system sends SNMP traps. You can configure up to 10 remote trap destinations.

The no form of this command deletes one or more defined remote trap hosts. If you enter optional arguments, the CLI deletes the user entry only if it matches all arguments. With the no form of the command, the only required argument is `index`.

Access mode

enable

Syntax

To create a remote host configuration:

```
switchServices trap destination
   index indexNumber
   ipAddress ipAddress
   userName text
   snmpVersion (SNMPv1 | SNMPv2c | SNMPv3)
   [port portNumber]
   [filter {defaultTrapd | defaultFile | defaultLog | defaultSyslog |
      text}]
```

To modify a remote host configuration:

```
switchServices trap destination
   index indexNumber
   [ipAddress ipAddress]
   [userName text]
   [snmpVersion {SNMPv1 | SNMPv2c | SNMPv3}]
   [port portNumber]
   [filter {defaultTrapd | defaultFile | defaultLog | defaultSyslog |
      text}]
```

Arguments

Argument name	Description
`index` *`integer`*	Specifies an integer that uniquely identifies the trap host configuration. You can configure up to 10 trap hosts.
`ipAddress` *`ipAddress`*	Specifies the IP address of the remote trap host. Use 4-byte dotted decimal format. This argument is optional when modifying a configuration.
`userName` *`text`*	Specifies the SNMP community string or SNMPv3 user name configured on the remote trap host. This argument is optional when modifying a configuration.
`snmpVersion {SNMPv1 | SNMPv2c | SNMPv3}`	Specifies the SNMP format that the system uses for traps that it sends in an SNMP packet to a remote SNMP entity. The default setting is `SNMPv1`. This argument is optional when modifying a configuration.
`port` *`portNumber`*	**Optional.** Specifies the number of the UDP port that the remote trap host uses to receive traps. The valid range is 1 to 65535. The default port number is 162.
`filter {defaultTrapd | defaultFile | defaultLog | defaultSyslog |` *`text`*`}`	**Optional.** Specifies the filter profile that is used to filter system events that are sent to this trap destination. System events are checked against the filter to determine if the system event should be converted to a trap and sent to the trap destination. The default is `defaultTrapd`.

Delete filters

```
no switchServices trap destination
   index indexNumber
   [ipAddress ipAddress]
   [userName text]
   [snmpVersion {SNMPv1 | SNMPv2c | SNMPv3}]
   [port portNumber]
   [filter {defaultTrapd | defaultFile | defaultLog | defaultSyslog |
      text}]
```

Example

The following example shows how to configure a remote trap host that has an address of 10.10.10.10, uses the default UDP port of 162, uses a community string of private, and sends traps using the SNMPv2c format.

```
sun> enable
sun# switchServices
sun(switchServices)# trap
sun (switchServices trap)# destination index 1 ipAddress 10.10.10.10
userName private SNMPVersion SNMPv2c
```

Associated MIB

```
Sun-App-Switch-Trap-MIB.mib
```

Web path

- switchServices → trap → destination → add
- switchServices → trap → destination → copy
- switchServices → trap → destination → modify
- switchServices → trap → destination → delete

trap systemEvtTrap

Purpose

Specifies whether the system generates traps for system events. If enabled, the system generates a trap for each system event that it stores in the internal system log.

Access mode

enable

Syntax

```
switchServices trap systemEvtTrap
   [systemEvtTraps {enabled | disabled}]
```

Arguments

Argument name	Description
`systemEvtTraps {enabled | disabled}`	**Optional.** Specifies whether the system sends system events as traps to configured trap hosts. If `enabled`, the system generates a trap for each system event; if `disabled`, the system does not generate any traps for system events.

Example

The following example shows how to configure traps so that the system generates traps for system events.

```
sun> enable
sun# switchServices
sun(switchServices)# trap
sun(switchServices trap)# systemEvtTrap enable
```

Associated MIB

`Sun-App-Switch-Trap-MIB.mib`

Web path

- switchServices → trap → systemEvtTrap

Chapter 7. TFTP commands

TFTP description

The Trivial File Transfer Protocol (TFTP) is a simple User Datagram Protocol (UDP)-based file transfer protocol, similar to the File Transfer Protocol (FTP). However, unlike FTP, TFTP does not provide password protection or the ability to view a directory structure.

To view a list of files on the system, enter the command-line interface (CLI) command `ls` or `ls -l` at the CLI prompt.

TFTP command path

The command names in this chapter show you how to execute the commands from within the following command modes:

```
switchServices commandName
switchServices tftpd commandName
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

TFTP command summary

Table 7-1 lists and briefly describes the TFTP commands.

Table 7-1. TFTP command summary

Command name	Description
`tftp`	Transfer a file to or from a remote system using TFTP.
`tftpd`	Configure the TFTP daemon on the system.
`show tftpd`	Display the current TFTP daemon configuration.
`show tftpd sessions`	Display information about TFTP activity for one or more sessions.

tftp

Purpose

Transfers a file to, or retrieves a file from, a remote system.

Access mode

user

Syntax

```
switchServices tftp
   host ipAddress
   direction {get | put}
   source path/fileName
   [destination path/fileName]
   [binary {enabled | disabled}]
   [output {enabled | disabled}]
```

Arguments

Argument name	Description
`host` *`ipAddress`*	Specifies the IP address of a remote host that has a TFTP server enabled.
`direction {get | put}`	Specifies whether to send a file to a remote host (`put`) or retrieve a file from a remote host (`get`).
`source` *`path/fileName`*	Specifies the name of the file you want to transfer.
`destination` *`path/fileName`*	**Optional**. Specifies where to transfer the file (path and file name). If you do not specify this value, the system uses the source file name and transfers the file to the default TFTP directory.
`binary {enabled | disabled}`	**Optional**. Indicates whether to transfer the file in binary mode (byte by byte). Set the value to `enabled` when you transfer binary files; set the value to `disabled` when you transfer ASCII files.
`ouput {enabled | disabled }`	**Optional**. Indicates whether the system displays details about the TFTP file transfer. If `enabled`, the system displays detailed information about the TFTP activity; if `disabled`, the system does not display these details.

Example

This example shows how to use the CLI to transfer a backup copy of the configuration file to a remote host using TFTP. In this example, the `output` argument is enabled and the system shows details about the TFTP activity.

```
sun# switchServices
sun(switchServices)# tftp host 10.10.40.1 direction put source
config.bak destination config.bak binary enabled output enabled.

done
Transmitted 3299 bytes in 0.6 seconds.
Transmitted 42844 bytes/second.
Successfully transferred '/ftl0/cdb.bak' to 10.10.40.1 into 'cdb.bak'.
```

Associated MIB

`tftp.mib`

Web path

- switchServices → tftp

tftpd

Purpose

Configures the TFTP daemon on the N2000 Series. TFTP allows you to transfer files between the local system and a remote system.

Access mode

enable

Syntax

```
switchServices tftpd
   [adminState {enabled | disabled}]
   [maxSessions integer]
   [tftpdPort portNumber]
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	**Optional.** Specifies the administrative state of the TFTP daemon. If enabled, you can use TFTP to transfer files from a remote system to the N2000 Series; if disabled, you cannot use TFTP to transfer a file to the system. The default setting is `disabled`.
`maxSessions` *`integer`*	**Optional.** Configures the number of TFTP sessions allowed at one time. Valid values are from 0 to 10 sessions; the default setting is 10 sessions.
`tftpdPort` *`portNumber`*	**Optional.** Configures the port to use for TFTP file transfers. The valid range is from 0 to 65536; the default setting is 69.

Example

The following example shows how to configure the TFTP server and allow five sessions.

```
sun> enable
sun# switchServices
sun(switchServices)# tftpd
sun(switchServices tftpd)# adminState enabled maxSessions 5
sun(switchServices tftpd)#
```

Associated MIB

`tftpd.mib`

Web path

- switchServices → tftpd → modify

show tftpd

Purpose

Displays the current TFTP daemon configuration and session statistics. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices tftpd
```

Sample output

```
sun> switchServices
sun(switchServices)> show tftpd
Administrative State:    enabled
Maximum Sessions:        10
Tftpd Port:              69
Operational State:       up
Current Opened Sessions: 0
Total Opened Sessions:   0
```

Output description

Field name	Description
`Administrative State`	The current administrative state of the TFTP daemon. If set to `enabled`, the TFTP server is active on the system. If set to `disabled`, you cannot use TFTP to transfer files from a remote host to the system.
`Maximum Sessions`	The maximum TFTP sessions the system allows at one time.
`Tftpd Port`	The port the system uses for TFTP file transfers.
`Operational state`	The current operational state of the TFTP daemon. If `up`, the daemon is functioning properly; if `down`, the daemon is not functioning properly.
`Current Sessions`	The number of sessions currently open on the TFTP server.
`Total Sessions`	The total number of TFTP sessions that were opened on the system.

Associated MIB

`tftpd.mib`

Web path

- switchServices → tftpd

show tftpd sessions

Purpose

Displays details about active TFTP sessions.

Access mode

user

Syntax

```
show switchServices tftpd sessions
```

Sample output

```
sun> switchServices
sun(switchServices)> tftpd
ERROR: command not found!

sun(switchServices)> enable
sun(switchServices)# tftpd
sun(switchServices tftpd)# show sessions
Client IP         Client Port    Oper State       Bytes Sent     Bytes Received
192.168.209.18    3905           up               21736          2803428
```

or

```
sun> switchServices
sun(switchServices)> show tftpd sessions
Client IP         Client Port    Oper State       Bytes Sent     Bytes Received
192.168.209.18    3905           up               21736          2803428
```

Output description

Field name	Description	Filter name
`Client IP`	The IP address of the TFTP client.	`clientIp` *`ipAddress`*
`Client Port`	The port the client system is using for TFTP transfers.	`clientPort` *`portNumber`*
`Oper State`	The current state of the TFTP session.	`operState {up |down}`
`Bytes Sent`	The number of bytes the system sent to a client during the TFTP session.	`bytesSent` *`bytes`*
`Bytes Received`	The number of bytes the system received during the TFTP session.	`bytesReceived` *`bytes`*

Associated MIB

`tftpd.mib`

Web path

- switchServices → tftpd → sessions

Chapter 8. FTP client commands

FTP description

The File Transfer Protocol (FTP) client provides the ability to transfer files between the N2000 Series and a remote FTP server. In addition to file transfer operations, FTP provides password protection and the ability to view a directory structure. You cannot use FTP on a remote host to access the N2000 Series.

FTP command path

The examples in this chapter show you how to execute the commands from the FTP command mode:

`(switchServices ftp) commandName`

Note: To access the FTP command mode, you must first establish an FTP session using the `open` and `user` commands.

FTP command summary

Table 8-1 lists and briefly describes the FTP commands.

Table 8-1. FTP command summary

Command name	Description
ascii	Set the transfer type to ASCII.
binary	Set the transfer type to binary.
bye	Terminate the FTP session and exits the connection.
cd	Change the remote working directory.
cdup	Change the remote working directory to the parent directory.
close	Terminate an FTP session.
dir	Display the current directory.
disconnect	Terminate an FTP session.
get	Retrieve files from a remote host.
hash	Toggle printing of the hash symbol (#) for each transferred buffer.
lcd	Change the local working directory.
ls	List the contents of a remote directory.
mkdir	Create a directory on the remote host.
nlist	List the contents of a remote directory
open	Connect to a remote FTP server.
put	Send one file to a remote host.
pwd	Print the contents of the remote host's working directory.
quote	Send a literal string to a remote host.
rename	Rename a file.
reset	Clear queued command replies.
rhelp	Display the Help topics on a remote FTP server.
rmdir	Remove a directory from the remote host.
rstatus	Show status of a remote host.

Table 8-1. FTP command summary (continued)

Command name	Description
`status`	Show current status.
`system`	Show the remote system type.
`user`	Log a user into a remote FTP server.
`verbose`	Toggle verbose mode.

ascii

Purpose

Sets the transfer type to ASCII. Use this transfer mode when transferring files that contain ASCII text only.

Access mode

user

Syntax

```
ascii
```

Arguments

There are no arguments for this command.

Example

This example shows how to set the transfer mode to ASCII for transferring ASCII only files.

```
sun(switchServices ftp)# ascii
200 Type set to A.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

binary

Purpose

Sets the transfer type to binary. Use this transfer mode when transferring files that contain text that is not in an ASCII format.

Access mode

user

Syntax

```
binary
```

Arguments

There are no arguments for this command.

Example

This example shows how to set the transfer mode to binary for transferring files that are not in an ASCII format.

```
sun(switchServices ftp)# binary
200 Type set to I.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

bye

Purpose

Ends the FTP session and exits the FTP command mode.

Access mode

user

Syntax

`bye`

Arguments

There are no arguments for this command.

Example

This example shows how to use the `bye` command to end an FTP connection.

```
sun(switchServices ftp)# bye
221 Goodbye. Control connection closed.

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

cd

Purpose

Changes the working directory on the remote host.

Access mode

user

Syntax

```
cd
   [directory text]
```

Arguments

Argument name	Description
`directory` *`text`*	**Optional**. Specifies the directory on the remote host that you want to set as the current working directory.

Example

This command shows how to change the working directory on the remote host to `user/bin`. The `pwd` command confirms that the working directory changed.

```
sun(switchServices ftp)# cd usr/bin
250 CWD command successful.

sun(switchServices ftp)# pwd
257 "/usr/bin" is current directory.

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

cdup

Purpose

Changes the working directory on the remote host to the parent of the current directory.

Access mode

user

Syntax

```
cdup
   [directory text]
```

Arguments

Argument name	Description
`directory` *`text`*	**Optional**. Specifies the directory on the remote host that you want to set as the current working directory.

Example

The following example shows how to change the working directory on a remote host to the directory that is the parent of the current directory. The pwd command confirms that the working directory changed.

```
sun(switchServices ftp)# pwd
257 "/usr/bin" is current directory.

sun(switchServices ftp)# cdup
250 CWD command successful.

sun(switchServices ftp)# pwd
257 "/usr" is current directory.

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Associated MIB

- Not applicable. This command is available in the CLI only.

close

Purpose

Ends the FTP session but does not exit the FTP command mode. This command is the same as the `disconnect` command.

Access mode

user

Syntax

```
close
```

Arguments

There are no arguments for this command.

Example

The following example shows how to terminate an FTP connection.

```
sun(switchServices ftp)# close
221 Goodbye. Control connection closed.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

dir

Purpose

Displays the contents of a specified directory on the remote host.

Access mode

user

Syntax

```
dir
   [directory text]
```

Arguments

Argument name	Description
`directory` *text*	**Optional**. Specifies the name of the directory whose contents you want to display.

Example

The following example shows how to display the contents of a directory. If no directory name is specifed, the command displays the contents of the current working directory.

```
sun(switchServices ftp)# dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls (325 bytes).
total 7
drwxrw-rw- 1 root  root  0 Feb 24 13:05 .
drwxrw-rw- 1 root  root  0 Feb 24 13:05 ..
drwxrw-rw- 1 root  root  0 Feb 24 10:07 bin
drwxrw-rw- 1 root  root  0 Feb 24 13:01 pub
drwxrw-rw- 1 root  root  0 Feb 24 13:11 test
drwxrw-rw- 1 root  root  0 Feb 24 10:07 upload
drwxrw-rw- 1 root  root  0 Feb 24 10:20 usr
226 Transfer complete. 325 bytes in 0.02 sec. (15.869 Kb/s)

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

disconnect

Purpose

Ends the FTP session but does not exit the FTP command mode. This command is the same as the `close` command.

Access mode

user

Syntax

```
disconnect
```

Arguments

There are no arguments for this command.

Example

The following example shows how to terminate an FTP session.

```
sun(switchServices ftp)# disconnect
221 Goodbye.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

get

Purpose

Transfers the specified file from the remote host to the N2000 Series.

Access mode

user

Syntax

```
get
   remote fileName
   [local fileName]
```

Arguments

Argument name	Description
`remote` *`fileName`*	Specifies a file on the remote host to copy. If you do not specify the `local` argument, the system uses the remote file name as the local file name.
`local` *`fileName`*	**Optional**. Specifies the name that the system should use locally for the remote file.

Example

The following example shows how to transfer a file from a remote host to the local host.

```
sun(switchServices ftp)# ftp get pasta.rtf
200 PORT command successful.
150 Opening BINARY mode data connection for pasta.rtf (9934 bytes).
226 Transfer complete. 9934 bytes in 0.05 sec. (242.529 Kb/s)

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

hash

Purpose

Toggles hash (#) printing. When you issue this command, FTP displays one hash sign for each data block that it transfers. A data block is 1024 bytes.

Access mode

user

Syntax

```
hash
```

Arguments

There are no arguments for this command.

Example

```
sun(switchServices ftp)# hash
Hash mark printing on (1024 bytes/hash mark).

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

lcd

Purpose

Changes the local directory on the N2000 Series.

Access mode

user

Syntax

```
lcd
   directory text
```

Arguments

Argument name	Description
`dirctory` *`text`*	Specifies the name of the directory that you want to use as the working directory on the local system.

Example

```
sun(switchServices ftp)# lcd /ftl0/lib/
Local directory now /ftl0/lib/

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

ls

Purpose

Lists the contents of a specified directory on the remote host.

Access mode

user

Syntax

```
ls
   [directory text]
```

Arguments

Argument name	Description
`directory` *`text`*	**Optional**. Specifies the name of the directory whose contents you want to list.

Example

This example shows how to view a directory listing on a remote host. If no directory name is specifed, the command displays the contents of the current directory.

```
sun(switchServices ftp)# ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls (279 bytes).
total 6
drwxrw-rw- 1 root  root  0 Feb 24 10:07 .
drwxrw-rw- 1 root  root  0 Feb 24 10:07 ..
drwxrw-rw- 1 root  root  0 Feb 24 10:07 bin
drwxrw-rw- 1 root  root  0 Feb 24 10:07 pub
drwxrw-rw- 1 root  root  0 Feb 24 10:07 upload
drwxrw-rw- 1 root  root  0 Feb 24 10:20 usr
226 Transfer complete. 279 bytes in 0.03 sec. (9.082 Kb/s)
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

mkdir

Purpose

Creates the specified directory on the remote host.

Access mode

user

Syntax

```
mkdir
   directory text
```

Arguments

Argument name	Description
`directory` *`text`*	Specifies the directory that you want to create on the remote host.

Example

This example shows how to create a directory on a remote host.

```
sun(switchServices ftp)# mkdir test
257 Directory "test" created.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

nlist

Purpose

Displays a listing of the files in the current remote directory.

Access mode

user

Syntax

```
nlist
```

Arguments

There are no arguments for this command.

Example

This example shows how to view a list of files in a directory on a remote host.

```
sun(switchServices ftp)# nlist
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls (30 bytes).
sauce.doc
pasta.rtf
226 Transfer complete. 30 bytes in 0.03 sec. (0.977 Kb/s)
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

open

Purpose

Opens a connection to the FTP server on the specified remote host.

Access mode

user

Syntax

```
open
   host IPAddress
   user text
   password text
   [port portNumber]
```

Arguments

Argument name	Description
`host` *`IPAddress`*	Specifies the IP address of the remote FTP server host.
`user` *`text`*	Specifies the user name that the remote host requires for access.
`password` *`text`*	Specifies the password that the remote host requires for access.
`port` *`portNumber`*	**Optional**. Specifies the port number the remote host uses for FTP operations. The default port number is 21.

Example

This example shows how to open a connection to a remote FTP server as an anonymous user.

```
sun> enable
sun# switchServices
sun(switchServices)# ftp open 190.16.1.20 anonymous
jgoldstein@mynet.com
Connected to 190.16.1.20.
220 saba FTP server (SunOS 5.8) ready.
331 Guest login ok, send ident as password.
230 Guest login ok, access restrictions apply.
215 UNIX Type: L8 Version: SUNOS

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

put

Purpose

Transfers the specified file from the N2000 Series to a remote host.

Access mode

user

Syntax

```
put
    local fileName
    [remote fileName]
```

Arguments

Argument name	Description
`local` *`fileName`*	Specifies the name of the file on the N2000 Series that you want to transfer to the remote host.
`remote` *`fileName`*	**Optional**. Specifies the name to assign to the transferred file on the remote host.

Example

This example shows how to transfer a file from the local system to the remote host.

```
sun (switchServices ftp)# put tiderunner.txt
200 Port command successful.
150 Opening ASCII mode data connection for tiderunner.txt.
done
Transmitted 5380 bytes in 0.1 seconds.
Transmitted 51730 bytes/second.
226 Transfer complete.
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

pwd

Purpose

Displays the name of the remote host's current directory.

Access mode

user

Syntax

`pwd`

Arguments

There are no arguments for this command.

Example

This example shows how to display the current working directory on a remote host.

```
sun(switchServices ftp)# pwd
257 "/" is current directory.

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

quote

Purpose

Sends a literal string to the remote host as an FTP command.

Access mode

user

Syntax

```
quote
   quote text
```

Arguments

Argument name	Description
quote *text*	Specifies a literal string that you want to send to a remote host.

Example

The following example shows how to execute the pwd and cd commands on the remote host by sending the following quote commands.

```
sun(switchServices ftp)# quote “PWD”
257 “/” is current directory.

sun (switchservices ftp)# quote “CWD home”
250 CWD command successful.

sun(switchServices ftp)# quote “PWD”
257 “/home” is current directory.
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

rename

Purpose

Changes the name of a file on the remote host.

Access mode

user

Syntax

```
rename
   oldfile text
   newfile text
```

Arguments

Argument name	Description
`oldfile` *text*	Specifies the current name of the file.
`newfile` *text*	Specifies the new name that you want to assign to the file.

Example

This example shows how to change the name of a file on a remote host.

```
sun(switchServices ftp)# rename oldfile macaroni.rtf newfile pasta.rtf
350 RNFR Ok. Ready for destination name.
250 Requested file action okay, completed.

sun(switchServices ftp)#
```

Associated MIB

`ftp.mib`

Web path

- Not applicable. This command is available in the CLI only.

reset

Purpose

Clears the reply queue. This command resynchronizes command and reply sequencing with the remote FTP server. Resynchronization may be necessary if the remote server violates the FTP protocol.

Access mode

user

Syntax

```
reset
```

Arguments

There are no arguments for this command.

Example

This example shows how to clear the reply queue.

```
sun(switchServices ftp)# reset
Queue empty.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

rhelp

Purpose

Displays command Help stored on the remote FTP server.

Access mode

user

Syntax

```
rhelp
   [command text]
```

Arguments

Argument name	Description
`command` *`text`*	**Optional**. Indicates a specific command for which you want to display the Help stored on the remote FTP server. If you do not specify a command, the command displays a list of all Help topics.

Example

This example shows how to display the available Help topics on a remote host. It also shows how to display the Help for a specific command.

```
sun(switchServices ftp)# rhelp
214-The following commands are recognized:
   USER    EPRT    STRU    MAIL*   ALLO    CWD     STAT*   XRMD
   PASS    LPRT    MODE    MSND*   REST*   XCWD    HELP    PWD
   ACCT*   EPSV    RETR    MSOM*   RNFR    LIST    NOOP    XPWD
   REIN*   LPSV    STOR    MSAM*   RNTO    NLST    MKD     CDUP
   QUIT    PASV    APPE    MRSQ*   ABOR    SITE*   XMKD    XCUP
   PORT    TYPE    MLFL*   MRCP*   DELE    SYST    RMD     STOU
214 (*'s => unimplemented)

sun(switchServices ftp)# rhelp mkd
214 Syntax: MKD <sp> path-name

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

rmdir

Purpose

Removes a directory on the remote host. The directory must be empty before you can delete it.

Access mode

user

Syntax

```
rmdir
   directory text
```

Arguments

Argument name	Description
`directory` *`text`*	Specifies the name of the directory on the remote host that you want to remove.

Example

This example shows how to delete a directory on a remote host.

```
sun(switchServices ftp)# rmdir test
250 "/test" successfully deleted.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

rstatus

Purpose

Displays the current status of the FTP server.

Access mode

user

Syntax

```
rstatus
```

Arguments

There are no arguments for this command.

Example

The following example shows how to display the current status of an FTP server.

```
sun(switchServices ftp)# rstatus
211-Joan's FTP Server FTP server status:
    Version WarFTPd 1.71.02
    Connected to  (192.168.124.46:1173->192.168.209.76:21)
    Logged in anonymously
    No data connection
211 End of status

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

status

Purpose

Displays the current status of the FTP session on the N2000 Series.

Access mode

user

Syntax

```
status
```

Arguments

There are no arguments for this command.

Example

The following example shows how to display the status of the local system's connection to an FTP server.

```
sun(switchServices ftp)# status
Connected to 192.168.1.20

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

system

Purpose

Displays the type of operating system running on the remote host.

Access mode

user

Syntax

```
system
```

Arguments

There are no arguments for this command.

Example

The following example shows how to display the operating system that the remote host uses.

```
sun(switchServices ftp)# system
215 UNIX Type: L8 Version: SUNOS

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

user

Purpose

Logs the user on the N2000 Series into the remote FTP server.

Access mode

user

Syntax

```
user
    user userName
    password password
```

Arguments

Argument name	Description
`user` *`userName`*	Indicates the user name for the user logged into the N2000 Series.
`password` *`password`*	Indicates the password associated with the user name.

Example

The following example shows how to log the N2000 Series user into the FTP server after an FTP connection has been opened with the `open` command.

```
sun(switchServices ftp)# user jk1234 password password
230 User jk1234 logged in.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

verbose

Purpose

Toggles the verbose mode on and off. When verbose mode is on, the system displays all responses from the remote FTP server. The system also displays file transfer statistics when the transfers are complete.

Access mode

user

Syntax

```
verbose
```

Arguments

There are no arguments for this command.

Example

The following example shows how to toggle the verbose mode. By default the verbose mode is set to on.

```
sun(switchServices ftp)# verbose
Verbose mode off.

sun(switchServices ftp)#
```

Associated MIB

```
ftp.mib
```

Web path

- Not applicable. This command is available in the CLI only.

Chapter 9. Telnet commands

Telnet protocol description

Telnet is a simple, standard, TCP/IP-based terminal emulation protocol defined in RFC 854, *Telnet Protocol Specification.* Telnet provides a standard method for terminal devices and processes to "talk." Telnet allows a remote user to establish a terminal connection over an IP network.

The N2000 Series uses Telnet to establish a connection to the command-line interface (CLI). The commands described in this chapter allow you to set characteristics of the Telnet session and display operational information.

Basic Telnet configuration

Table 9-1 describes the steps for connecting to the CLI via Telnet. For more detailed information about opening a Telnet session and starting the CLI, see Chapter 1, "Using the management interfaces."

Table 9-1. Steps for connecting to the CLI via Telnet

Step	Action
1.	Ensure that the N2000 Series is connected to a network that the remote system can reach.
2.	Configure an IP address for the management port (ethMgmt.1).
3.	Start the Telnet client.
4.	Log in and start the CLI.

Telnet command path

The command names in this chapter show you how to execute the commands from within the following command modes:

`switchServices` *`commandName`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Telnet command summary

Table 9-2 lists and briefly describes the Telnet commands.

Table 9-2. Telnet command summary

Command name	Description
`telnetd`	Modify the Telnet configuration.
`show telnetd`	Display Telnet operational information.
`show telnetd sessions`	Display a list of active Telnet sessions, operational information, and counters.

telnetd

Purpose

Configures the main parameters of the Telnet protocol.

Access mode

user

Syntax

```
switchServices telnetd
   [adminState {enabled | disabled}]
   [maxSessions integer]
   [telnetPort integer]
   [rcvBufSize bytes]
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the protocol, either `enabled` (running) or `disabled`. When disabled, the parameters of Telnet can still be configured, but do not become active until `adminState` is set to `enabled`. The default is `enabled`.
`maxSessions` *`integer`*	**Optional.** Sets the maximum number of Telnet sessions allowed. Valid range is 0 through 10. The default number of sessions is 10.
`telnetPort` *`integer`*	**Optional.** Identifies the port the N2000 Series monitors to listen for Telnet sessions. Valid port numbers are 1 through 65535. The default port is 23.
`rcvBufSize` *`bytes`*	**Optional.** Configures the number of bytes for each Telnet session's receive buffer. The default should be sufficient, but in an instance when you must, for example, paste a large amount of text, increase this value to avoid overflowing the buffer. Valid values are 2000 through 65535. The default is 2000 bytes.

Example

The following example disables Telnet on the system, changes the default Telnet configuration to a 10-session maximum, and sets the port to 77.

```
sun> switchServices
sun(switchServices)# telnetd disabled maxSessions 10 telnetPort 77
```

Associated MIB

```
telnet.mib
```

Web path

- switchServices → telnetd → modify

show telnetd

Purpose

Displays the operational characteristics of the Telnet daemon. The output includes parameters set with the `telnetd` command and counters maintained by the system. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices telnetd
```

Sample output

```
sun> switchServices
sun(switchServices)# show telnetd
Administrative State:    enabled
Maximum Sessions:        10
Receive Buffer Size:     2000
Telnetd Port:            23
Operational State:       up
Current Opened Sessions: 2
Total Opened Sessions:   2
```

Output description

Field name	Description
`Administrative State`	The configured status for the Telnet session, either `enabled` or `disabled`.
`Maximum Sessions`	The maximum number of Telnet sessions this server can support at one time.
`Receive Buffer Size`	The number of bytes for each Telnet session's receive buffer. The receive buffer is a device from which an application can read or to which it can write.
`Telnetd Port`	The TCP port over which Telnet is currently operating.
`Operational State`	The operational status of the Telnet session, either `up` or `down`.

Field name	Description
`Current Opened Sessions`	The number of Telnet sessions currently open.
`Total Opened Sessions`	The number of Telnet sessions opened on this server since last system startup.

Associated MIB

`telnet.mib`

Web path

- switchServices → telnetd

show telnetd sessions

Purpose

Displays session-specific information about this Telnet connection, both client data and counters maintained by the system. You can filter this command output by any of the parameters listed in the Output Description table.

Access mode

user

Syntax

```
show switchServices telnetd sessions
```

Sample output

```
sun> switchServices
sun(switchServices)# show telnetd sessions
Client IP       Client Port Session State Bytes Sent        Bytes Received
192.168.209.66  1825        up            82                75
192.168.209.68  1077        up            98                80
192.168.209.68  1078        up            82                89
192.168.209.68  1079        up            82                75

sun(switchServices)# show telnetd sessions clientport 1078
Client IP       Client Port Session State Bytes Sent        Bytes Received
192.168.209.68  1078        up            82                89

sun(switchServices)# show telnetd sessions operState down
There are no entries matching your query.
```

Output description

Field name	Description	Filter name
`Client IP`	The IP address of the remotely connecting Telnet client.	`clientIp` *`ipAddress`*
`Client Port`	The TCP port on the N2000 Series over which the client is connecting.	`clientPort` *`integer`*

Field name	Description	Filter name
`Session State`	The state of the Telnet session, either `up` or `down`.	`operState {up | down}`
`Bytes Sent`	The total number of bytes sent during this Telnet session.	`bytesSent` *`integer`*
`Bytes Received`	The total number of bytes received during this Telnet session.	`bytesReceived` *`integer`*

Associated MIB

`telnet.mib`

Web path

- SwitchServices → telnetd → sessions

Chapter 10. NTP and clock commands

Timing description

The N2000 Series system allows you to set both an internal system clock as well as to synchronize with network clocks. If Network Time Protocol (NTP) is enabled, the value that the NTP probe returns causes the system to adjust the internal system clock so that it is synchronized with the network clock.

Network Time Protocol description

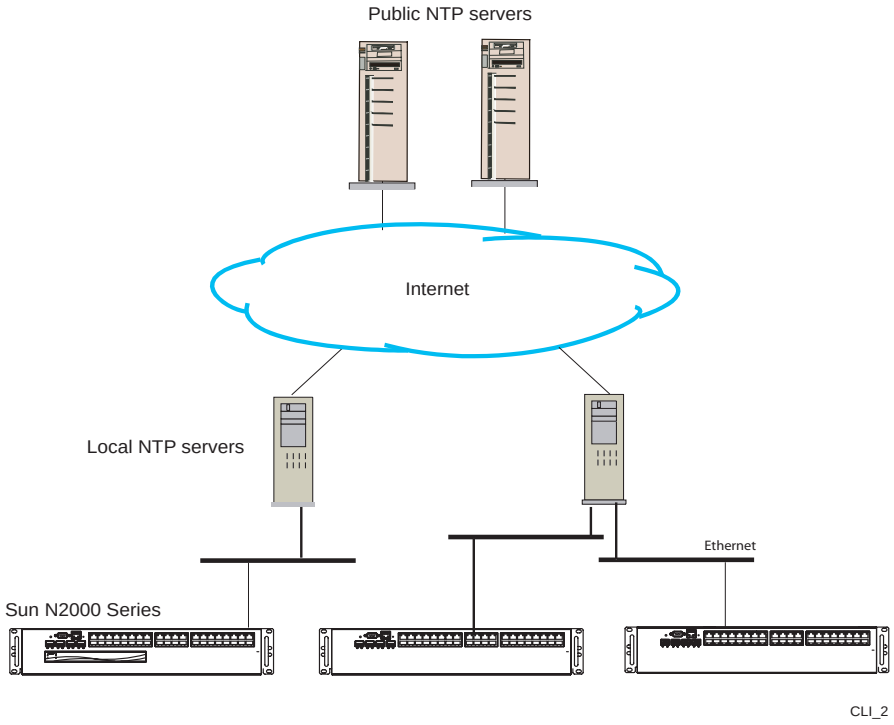
The N2000 Series system uses Network Time Protocol, Version 4 (see RFC 1305) to synchronize its clock with network clocks. Systems using NTP all try to set and maintain their internal clock to the same time, which is coordinated universal time (UTC).

Exact, synchronized time across a network is important for many functions; for example, packet and event time stamps and security certificate validation. Most ordinary computer clocks drift a few seconds a day, so networked systems would soon be seriously out of step without NTP. NTP uses time signals from standard reference time sources on the Internet that are typically very accurate.

There is a real-time clock running on each N2000 Series system with which the local clock must be synchronized. The system need not be synchronized with external clocks, however. The system uses the NTP manager to handle both internal and external synchronization. When configured, NTP server packets update the local clocks with time from the external NTP server. If NTP is not configured, local clocks derive time from the real-time clock on the system.

Figure 10-1 depicts a typical NTP configuration. The N2000 Series can retrieve time from either the local servers or stratum 2 or lower servers.

Figure 10-1. Typical NTP configuration



NTP on the N2000 Series

You can configure up to five NTP servers. When multiple NTP servers are configured, the N2000 Series computes a mean time from the servers.

Before using a public NTP server, be sure that you have permission and be certain to follow its access rules. The N2000 Series supplies efficient defaults for all optional configuration arguments. You need to change them only if the NTP server requires it. The N2000 Series system functions only as an NTP client.

NTP basic configuration

Table 10-1 briefly describes the steps for configuring NTP.

Table 10-1. Steps for configuring NTP

Step	Action
1.	Enable NTP.
2.	Configure at least one NTP server with the `ntp server` command.
3.	Verify permission and access rules for a public NTP server.
4.	Point to a public NTP server.

Command paths

The command names in this chapter show you how to execute the commands from within the following command modes:

```
clock
switchServices clock
switchServices ntp commandName
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Command summary

Table 10-2 lists and briefly describes the NTP and clock commands.

Table 10-2. NTP and clock command summary

Command name	Description
`clock`	Set the date and time on the internal clock.
`ntp (root)`	Set the administrative state of NTP.
`ntp advanced`	Modify the advanced NTP configuration parameters.
`ntp server`	Add or modify an NTP server configuration that the system uses to synchronize the system clock.
`show clock`	Display the internal clock's date and time.
`show clock advanced`	Display the internal clock's GMT date and time, with time zone offset.
`show ntp`	Display state of the NTP configuration on the system.
`show ntp advanced`	Display NTP operational statistics.
`show ntp server`	Display the configured NTP servers and their parameters.
`show ntp server advanced`	Display detailed operational statistics for configured NTP servers.

clock

Purpose

Sets the current date, local time, and time zone on the internal system clock. The system uses a 24-hour format. You can enter a time zone using an offset to adjust the time from Greenwich Mean Time (GMT). The changes take effect immediately.

If you do not enter a time zone, the system displays the time you enter and does not make any adjustments for your time zone. If you set the time zone, enter the offset for the `timeZoneOffset` argument first, then enter you the date and time. This ensures that the system adjusts the time appropriately for your time zone.

If you set a time using this command and enable NTP at a later time, the system will adjust the time, if there is a discrepancy between the set local time and the NTP time.

Note: After setting the clock, verify the current time. If you set the clock before setting the time zone, the time is interpreted using the currently configured time zone, or GMT. Repeat the `clock` command to make any corrections.

Access mode

enable

Syntax

```
clock
   [dateAndTime mm/dd/yyyy-hh:mn:ss]
   [timeZoneOffset {+ | -}h.mm]
```

or

```
switchServices clock
   [dateAndTime mm/dd/yyyy-hh:mn:ss]
   [timeZoneOffset {+ | -}h.mm]
```

Arguments

Argument name	Description
`dateAndTime` *`date-time`*	**Optional.** Sets the internal system clock's date and time in 24-hour format. Enter the value in the format: *`mm/dd/yyyy-hh:mn:ss`* • two-digit month and a forward slash (for example, `02/`) • two-digit day and a forward slash (for example, `20/`) • four-digit year and a dash (for example, `2003-`) • two-digit hour and a colon (for example, `22:`) • two-digit minute and a colon (for example, `00:`) • two-digit second (for example, `00`)
`timeZoneOffset {+ | -}`*`h.mm`*	**Optional.** Sets the time zone offset from GMT. A plus sign offsets the clock in advance of GMT; a minus sign offsets the clock behind GMT. Enter the value in the format: *`{+ | -}h.mm`* • A plus or minus sign with no space • one- or two-digit hour and a period (for example, `5.`) • two-digit minute (for example, `00`)

Examples

The following example sets the system clock to 10:00 p.m., Eastern Standard Time (EST):

```
sun> enable
sun# switchServices
sun(switchServices)# clock timeZoneOffset -5.00
dateAndTime 02/20/2003-22:00:00
```

Associated MIB

`hardware.mib`

Web path

- switchServices → clock → modify

ntp (root)

Purpose

Enables or disables the Network Time Protocol on the system. Also enters the `ntp` command mode.

Enabling NTP allows the system to synchronize its clock with other network clocks.

Access mode

enable

Syntax

```
switchServices ntp
   [adminState {enabled | disabled}]
```

Arguments

Argument Name	Description
`adminState {enabled | disabled}`	**Optional.** Enables or disables NTP on the system. If set to `enable`, the system can send requests to and receive time samples from other NTP servers. If set to disabled, you need to set the clock time using the `clock` command. The default setting is `disabled`.

Examples

The following example enables NTP on the system.

```
sun> enable
sun# switchServices
sun(switchServices) ntp adminState enabled
```

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp → modify

ntp advanced

Purpose

Modifies the value that controls the time that must elapse before the system makes a significant adjustment to the local clock. The system adjusts the clock based on time samples it receives from remote NTP servers. The time interval you set prevents the system from changing time too quickly, which can result in multiple clock adjustments.

Access mode

enable

Syntax

```
switchServices ntp advanced
   [clockMinStep seconds]
   [rtcUpdateInterval minutes]
```

Arguments

Argument name	Description
`clockMinStep` *`seconds`*	**Optional**. Sets the minimum interval (in seconds) that must elapse before the system makes significant changes to the local clock. The valid range is from 120 to 8000 seconds. The default value is 200.
`rtcUpdateInterval` *`minutes`*	**Optional**. Sets the real time clock (RTC) update interval so that the time setting is maintained should the system be restarted. The valid range is 0 to 1440 minutes. The default setting is 10 minutes. Setting the value to 0 disables the RTC update interval.

Example

The following example shows how to change the `clockMinStep` interval. With this change, the system waits approximately 3.33 minutes before making a significant adjustment to the system clock. To view the change, use the `show ntp advanced` command.

```
sun> enable
sun# switchServices
sun(switchServices)# ntp
sun(switchServices ntp)# advanced clockMinStep 300
sun(switchServices ntp)# show advanced
ClockMinStep:          300
RTC Update Interval:  10
Clock Offset:          -0.0524608
Combined Offset:       -0.054987
Watchdog Counter:     63
Clock Drift:           0.000201957
Clock Weighted Drift: 0
Socket ID:             0
```

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp → advanced → modify

ntp server

Purpose

Adds or modifies the NTP server that the N2000 Series system queries for network time updates. You can configure up to five NTP servers.

When modifying an existing NTP server configuration, you can change all values except the IP address. The system allows you to enter an IP address; however, you *must* enter the current IP address for the configuration. To configure an NTP server with a different IP address, you must delete the configuration and create a new one.

The `no` form of the command deletes a configured NTP server. If you enter optional arguments, the CLI deletes the server only if it matches all arguments. With the `no` form of the command, only the `id` argument is required. Use this command with caution as there is no confirmation or undo.

Access mode

enable

Syntax

To create an NTP server configuration:

```
switchServices ntp server
   id {1 | 2 | 3 | 4 | 5}
   ipAddress ipAddress
   [prefer {true | false}]
   [burst {true | false}]
   [minPoll seconds]
   [maxPoll seconds]
   [version versionNumber]
```

To modify an NTP server configuration:

```
switchServices ntp server
   ipAddress ipAddress
   [prefer {true | false}]
   [burst {true | false}]
   [minPoll seconds]
   [maxPoll seconds]
   [version versionNumber]
```

Arguments

Argument name	Description
`id` *`number`*	Assigns a unique identifier to this NTP server. Valid range is 1 through 5.
`ipAddress` *`ipAddress`*	Configures the IP address for this NTP server. Use 4-byte dotted decimal format. **Note:** Once a server is bound to an IP address, the address cannot be changed. The address must be deleted and then recreated.
`prefer {true | false]`	**Optional.** Designates this NTP server as the preferred server when the time samples from all NTP servers are of equal quality (for example same dispersion and noise). The default setting is `false`, which disables this function.
`burst {true | false}`	**Optional.** Sends NTP requests to this server in burst mode; that is, transmit eight send/receive cycles over a system-determined interval. The default setting is `false`, which disables this function. Enabling burst mode is a way to obtain time samples faster so that time synchronization takes less time. Burst mode is useful when the system needs additional clean time samples to determine the time to use for synchronization. It uses burst mode to get additional samples, when required to do so.

Argument name	Description
`minPoll` *integer*	**Optional.** Sets the minimum time, in seconds, between time queries sent to this NTP server.The default setting is 16. Possible values are: • 16 • 32 • 64 (~1 min.) • 128 (~2 min.) • 256 (~4 min.) • 512 (~8.5 min.) • 1024 (~17 min.) • 2048 (~34 min.) • 4096 (~68 min.) • 8192 (~2.25 hours) • 16384 (~4.55 hours) • 32768 (~9.1 hours) • 65536 (~18.2 hours) • 131072 (~36.4 hours) When the clock time is not yet stable, the system uses this value to obtain time samples. As the clock becomes more stable, the interval increases. The time interval that the system uses does not exceed the value set with the `maxPoll` argument.
`maxPoll` *integer*	**Optional.** Sets the maximum amount of time between time queries sent to this NTP server. When the clock time becomes stable, the system uses this value to obtain time samples. This argument supports the same values as the `minPoll` argument. The default value is 1024 seconds.
`version` *versionNumber*	**Optional.** Sets the NTP version number, which is transmitted in the NTP packets. That is, the setting restricts NTP communication to servers that implement NTP at or below the configured version. Valid settings are 1 through 4; the default setting is 4. The current NTP standard is version 3, but version 4 has been in development since 1994. Some public NTP servers are running version 4, so we suggest you accept the default, which allows all current and previous versions. You should set a lower version only if your NTP server requires it.

Delete filters

See the `show ntp server` command for filter descriptions.

```
no switchServices ntp server
    id {1 | 2 | 3 | 4 | 5}
    ipAddress ipAddress
    [prefer {true | false}]
    [burst {true | false}]
    [minPoll seconds]
    [maxPoll seconds]
    [version versionNumber]
    [stratum integer]
    [rxPkts integer]
    [txPkts integer]
    [droppedPkts integer]
    [timeouts integer]
    [operStatus {unknown | responding | notResponding}]
```

Example

The following example configures two NTP servers with different minimum and maximum poll times.

```
sun> enable
sun# switchServices
sun(switchServices)# ntp server 1 128.59.16.20 maxpoll 512
sun(switchServices)# ntp server 2 128.118.25.3 minpoll 64 maxpoll 2048
```

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp → server → add
- switchServices → ntp → server → copy
- switchServices → ntp → server → modify

show clock

Purpose

Displays the local time registered on the internal system clock. If you did not enter a time zone, the system does not do a local time calculation. The system displays what you enter. You can enter a time zone using the `clock` command. This command does not support field filtering.

Note: When setting local time with a time zone, enter the time zone first and then the time. This ensures that the clock displays the correct time for your time zone.

Access mode

user

Syntax

```
show switchServices clock
```

Sample output

```
sun> switchServices
sun(switchServices)# show clock
Date/Time: 12/19/2002-15:56:50
```

Output description

Fleld name	Description
`Date/Time`	The date and time set for the internal system clock, in the format `mm/dd/yyyy-hh:mn:ss`.

Associated MIB

`hardware.mib`

Web path

- switchServices → clock

show clock advanced

Purpose

Displays the time registered on the internal system clock as well as the GMT offset.

Access mode

user

Syntax

```
show switchServices clock advanced
```

Sample output

```
sun> switchServices
sun(switchServices)> show clock advanced
GMT Date/Time:    12/20/2002-08:27:55
Time Zone Offset: -5.00
```

Output description

Fleld name	Description
`GMT Date/Time`	The GMT date and time set for the internal system clock, in the format `mm/dd/yyyy-hh:mm:ss`. By adding or subtracting the `Time Zone Offset`, you can determine the local system time.
`Time Zone Offset`	The hours and minutes to add or subtract from GMT to establish the local time.

Associated MIB

`hardware.mib`

Web path

- switchServices → clock → advanced

show ntp

Purpose

Displays the state and operational statistics for NTP. Use this command to verify that NTP is operational. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices ntp
```

Sample output

```
sun> switchServices
sun(switchServices)# show ntp
Admin State:       enabled
Operational State: up
Operational Ticks: 75605
Current Ticks:     7194
Socket State:      connected
Dropped Packets:   0
```

Output description

Field name	Description
`Admin State`	The administrative state of NTP on the system. Possible values are `enabled` and `disabled`.
`Operational State`	The operating state of NTP on the system. Possible values are `up`, `down` and `initializing`.
`Operational Ticks`	An internal representation of the clock (approximately, in seconds). The system continuously increments this value; the `Admin State` setting does not have any affect on this value.
`Current Ticks`	A representation of the NTP internal clock (approximately, in seconds). This value increments only when the `Admin State` is enabled.

Field name	Description
`Socket State`	The current state of the NTP connection. Possible values are `connected` and `notConnected`. If an internal problem occurs that affects NTP communication, or if the administrative state of NTP is disabled, the system displays a `notConnected` state.
`Dropped Packets`	The number of NTP packets that the system dropped because it did not expect packets to come from a specific server.

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp

show ntp advanced

Purpose

Displays detailed operational statistics for NTP on the system. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices ntp advanced
```

Sample output

```
sun> switchServices
sun(switchServices)> show ntp advanced
ClockMinStep:          200
RTC Update Interval  10
Clock Offset:          20.4867
Combined Offset:       27.2602
Watch Dog Counter:     189
Clock Drift:           0.01
Clock Weighted Drift: 0.225
Socket ID:             0
```

Output description

Field name	Description
`ClockMinStep`	The minimum interval (in seconds) that must elapse before the system makes significant changes to the local clock.
`RTC Update Interval`	The real time clock (RTC) update interval so that the time setting is maintained should the system be restarted.
`Clock Offset`	The difference between the server time and the internal clock time for which the system is making adjustments.

Field name	Description
`Combined Offset`	A summary of the last time sample values the system received from remote NTP servers. This value is typically close in value to the clock offset. In the case where there is a large difference between the local clock time and time received from NTP servers (and the system has not yet adjusted the local clock), this value can be significantly different from the clock offset. The system may not currently be using this value to adjust the clock.
`Watch Dog Counter`	The number of seconds that have elapsed since the system adjusted the clock significantly. By viewing this value and the `ClockMinStep` value, you can determine the next time when the system can perform a significant adjustment to the clock. The `clockMinStep` value must elapse before the adjustment occurs).
`Clock Drift`	The number of seconds by which the local clock is slow or fast.
`Clock Weighted Drift`	A value that the system uses along with the `Clock Drift` value to more accurately determine the number of seconds to adjust the clock time.
`Socket ID`	A decimal value for the socket connection.

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp → advanced

show ntp server

Purpose

Displays the configured NTP servers and their parameters. If you enter the command with no arguments, the CLI displays all configured NTP servers. If you enter arguments, the CLI displays only those server(s) that match all arguments.

Access mode

user

Syntax

```
show switchServices ntp server
```

Sample output

```
sun> switchServices
sun(switchServices)> show ntp server
Server ID:          1
Server IP Address:  140.239.10.5
Preferred Server:   false
Burst Mode:         false
Min. Poll Interval: 256
Max. Poll Interval: 1024
NTP Version:        4
Stratum:            16
TX Packets:         4
RX Packets:         4
Timeouts:           4
Dropped Packets:    0
Oper Status:        responding

Server ID:          2
Server IP Address:  128.59.16.20
Preferred Server:   false
Burst Mode:         false
Min. Poll Interval: 256
Max. Poll Interval: 1024
NTP Version:        4
Stratum:            2
TX Packets:         4
RX Packets:         3
Timeouts:           1
Dropped Packets:    0
Oper Status:        responding
```

```
Server ID:          3
Server IP Address:  128.118.25.3
Preferred Server:   false
Burst Mode:         false
Min. Poll Interval: 256
Max. Poll Interval: 1024
NTP Version:        3
Stratum:            2
TX Packets:         4
RX Packets:         4
Timeouts:           0
Dropped Packets:    0
Oper Status:        responding
```

Output description

Field name	Description	Filter name
`Server ID`	The unique identifier of this NTP server.	`id {1 | 2 | 3 | 4 | 5}`
`Server IP Address`	The IP address of this NTP server.	`operIpAddress` *`IPAddress`*
`Preferred Server`	The operational state of the preferred status function. Normally, this value should be the same as the value set with the `ntp server` command.	`preferredServer {true | false}`
`Burst Mode`	The operational state of the burst mode function, used for calibrating the system clock. Normally, this value should be the same as the value set with the `ntp server` command.	`burstMode {true | false }`
`Min. Poll Interval`	The minimum time, in seconds, between time queries sent to this NTP server. Normally, this value should be the same as the value set with the `ntp server` command.	`minPoll` *`seconds`*
`Max. Poll Interval`	The maximum time, in seconds, between time queries sent to the NTP server. Normally, this value should be the same as the value set with the `ntp server` command.	`maxPoll` *`seconds`*
`NTP Version`	The NTP version number, which restricts NTP communication to servers that implement NTP at or below the configured version.	`version` *`versionNumber`*

Field name	Description	Filter name
`Stratum`	The stratum level of the NTP server.	`stratum` *`integer`*
`TX Packets`	The number of packets the system sent to the NTP server.	`txPkts` *`integer`*
`RX Packets`	The number of packets the system received from the NTP server within the expected time.	`rxPkts` *`integer`*
`Timeouts`	The number of timeouts that occurred when the NTP server did not respond in time.	`timeouts` *`integer`*
`Dropped Packets`	The number of NTP packets that the system dropped because it did not expect packets to come from a specific server.	`droppedPkts` *`integer`*
`Oper Status`	The operational status of the NTP server. Possible values are: `unknown`: The system cannot determine the NTP server status. `responding`: The system is receiving responses from the NTP server. `notResponding`: The system is not receiving responses from the NTP server.	`operStatus {unknown | responding | notResponding}`

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp → server

show ntp server advanced

Purpose

Displays detailed operational statistics for configured NTP servers. This information is useful if you need to verify that NTP is operating normally. It also helps you to determine when updates occur.

If you enter the command with no arguments, the CLI displays statistics for all NTP servers. If you enter arguments, the CLI displays details for only those server(s) that match all arguments.

Access mode

user

Syntax

```
show switchServices ntp server advanced
```

Sample output

```
sun> switchServices
sun(switchServices)> show ntp server advanced
Server ID:                  1
IP Address:                 140.239.10.5
Operational Preferred Mode: false
Operational Burst Mode:     false
Operational Minimum Poll:  4
Operational Maximum Poll:   10
Reference ID:               0x0
Update Ticks:               0
Next Update Ticks:          7395
Reference Time:

Server ID:                  2
IP Address:                 128.59.16.20
Operational Preferred Mode: true
Operational Burst Mode:     false
Operational Minimum Poll:   4
Reference ID:               0xCC7B0205
Update Ticks:               6828
Next Update Ticks:          6891
Reference Time:             2003-01-21 16:02:31.545
```

```
Server ID:                  3
IP Address:                 128.118.25.3
Operational Preferred Mode: false
Operational Burst Mode:     false
Operational Minimum Poll:  4
Operational Maximum Poll:   10
Reference ID:               0x80040101
Update Ticks:               6849
Next Update Ticks:          6912
Reference Time:             2003-01-21 16:16:05.935
```

Output description

Field name	Description	Filter name
`Server ID`	The unique identifier of this NTP server.	`id {1 | 2 | 3 | 4 | 5}`
`IP Address`	The IP address of this NTP server.	`operIpAddress` *`IPAddress`*
`Operational Preferred Mode`	The operational state of the preferred status function. Normally, this value should be the same as the value set with the `ntp server` command.	`operBurstMode {true | false}`
`Operational Burst Mode`	The operational state of the burst mode function, used for calibrating the system clock. Normally, this value should be the same as the value set with the `ntp server` command.	`operPerferedMode {true | false }`
`Operational Minimum Poll`	The minimum time, in seconds, between time queries sent to this NTP server. Normally, this value should be the same as the value set with the `ntp server` command.	`operMinPoll` *`seconds`*
`Operational Maximum Poll`	The maximum time, in seconds, between time queries sent to the NTP server. Normally, this value should be the same as the value set with the `ntp server` command.	`operMaxPoll` *`seconds`*
`Reference ID`	The reference identifier of the NTP server to which the system is synchronizing its time. The ID can be either an ASCII string or an IP address in hexadecimal format.	`refId` *`hexNumber`*

Field name	Description	Filter name
`Update Ticks`	The number of ticks when the system requested a time sample from the NTP server. A tick is approximately one second.	`updateTicks` *`seconds`*
`Next Update Ticks`	The number of ticks when the system will send the next time sample request to the NTP server. A tick is approximately one second.	`nextTicks` *`seconds`*
`Reference Time`	The time (in 24-hour format) when the remote NTP server last updated its local clock.	`refTime` *`dateAndTime`*

Associated MIB

`ntp.mib`

Web path

- switchServices → ntp → server → advanced

Chapter 11. CLI and HTTP commands

Embedded management command description

You can use the command-line interface (CLI) or the Web interface to manage all aspects of the N2000 Series. The `cli` commands allow you to customize the behavior of a CLI session, either for the current session only or for all CLI sessions that any user initiates. The `httpd` commands allow you to enable and manage Hypertext Transfer Protocol/Hypertext Transfer Protocol Secure (HTTP/HTTPS) access on the system.

You can also use the Simple Network Management Protocol (SNMP) to manage the system. See Chapter 6, “SNMP and trap commands” for detailed command descriptions. See Chapter 1, “Using the management interfaces” for additional details about using the CLI and the Web interface. See the *Sun N2000 Series Release 2.0 – Introduction Guide* for information about the Web interface.

CLI command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`switchServices` *`commandname`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Embedded management command summary

Table 11-1 lists and briefly describes the embedded management commands.

Table 11-1. CLI command summary

Command name	Description
`cli`	Configure values for the current session and the default values for all sessions.
`httpd`	Configure HTTP and HTTPS access on the system.
`show cli`	Display the current CLI session configuration.
`show httpd`	Display the current HTTP configuration.
`show httpd sessions`	Display the current HTTP session information.

cli

Purpose

Modifies CLI session values for the current session or for all sessions. You can use the Web interface to modify values that affect all sessions. You must use the CLI to modify values that affect a current CLI session.

Access mode

enable

Syntax

```
switchServices cli
   [rows integer]
   [autoLogoutTimeout minutes]
   [echo {enabled | disabled}]
   [prompt text]
   [lineWrap integer]
   [messageOnAutoLogout {enabled | disabled}]
   [historyLength integer]
   [defaultRows integer]
   [defaultEcho {enabled | disabled}]
   [defaultAutoLogoutTimeout minutes]
   [defaultMessageOnAutoLogout {enabled | disabled}]
   [defaultPrompt text]
   [auditLogging {on | off}]
   [defaultHistoryLength integer]
```

Arguments

Argument name	Description
`rows` *`integer`*	**Optional**. Configures the number of rows the system displays at one time for the current session only. The valid values are 0 through 250; the default is 24. If you set this argument to 0, the system displays all rows in the display without a prompt. If the number of rows in the display exceeds this value, the system prompts you to press [Return] or [Space Bar] to view the rest of the display, or **q** to terminate the display.
`prompt` *`text`*	**Optional**. Sets the command prompt for the current session.
`autoLogoutTimeout` *`minutes`*	**Optional**. Sets the amount of time, in minutes, that the current session can remain idle. The valid values are 0 through 60 minutes; the default is 10 minutes. If you select 0, the session does not time out.
`echo {enabled | disabled}`	**Optional**. Specifies whether the system redisplays the command key word after you enter it. This argument applies to the current session only. **Note:** This command is useful when using scripts for management operations.
`prompt` *`text`*	**Optional**. Specifies the system prompt for the current session. The system prompt must be 16 or fewer characters.
`lineWrap` *`integer`*	**Optional**. Specifies the number of characters that you can enter before a command wraps to the next line. The valid values are 79 through 135; the default is 79. This value applies to the current session only.
`messageOnAutoLogout {enabled | disabled}`	**Optional**. Specifies whether the system displays a message when the current session times out and the session is going to terminate the session. The valid values are `enabled` and `disabled`; the default value is `enabled`.
`historyLength` *`integer`*	**Optional**. Specifies the number of history lines to maintain in the CLI. The valid range is 50 to 200 lines. The default setting is 50 history lines.

Argument name	Description
`defaultRows` *integer*	**Optional**. Specifies the number of rows the system displays for all sessions. The valid values are 0 through 250; the default is 24. If you set this argument to 0, the system displays all rows in the display. If the number of rows in the display exceeds this value, the system prompts you to press [Return] or [Space Bar] to view the rest of the display, or **q** to terminate the display. **Note:** This value does not take effect until you start a new session.
`defaultEcho {enabled | disabled}`	**Optional**. Specifies whether the system redisplays the command key word after you enter it. This argument applies to all sessions. This command is useful when using scripts for management operations. **Note:** This value does not take effect until you start a new session.
`defaultAutoLogoutTimeout` *minutes*	**Optional**. Specifies the amount of time, in minutes, that any session can remain idle. The valid values are 0 through 60 minutes; the default is 10 minutes. If you select 0, the session does not time out. **Note:** This value does not take effect until you start a new session.
`defaultMessageOnAutoLogout {enabled | disabled}`	**Optional**. Specifies whether the system displays a message when a session times out and the session is going to terminate the session. The valid values are `enabled` and `disabled`; the default value is `enabled`. **Note:** This value does not take effect until you start a new session.
`defaultPrompt` *text*	**Optional**. Specifies the system prompt for all sessions. The system prompt must be 16 or fewer characters. **Note:** This value does not take effect until you start a new session.
`auditLogging {on | off}`	**Optional**. Specifies whether the system generates an event when the system configuration changes during a CLI session. If set to `on`, the system generates an event when the configuration changes; if set to `off`, the system does not generate an event. The default setting is `on`.
`defaultHistoryLength` *integer*	**Optional**. Specifies the default number of history lines to save in the CLI. The valid range is 50 to 200 lines. The default setting is 50 history lines.

Example

The following example shows how to change the CLI display for all sessions so that the system displays 50 rows at one time and times out after 45 minutes.

```
sun# switchServices
sun(switchServices)# cli defaultRows 50 defaultAutoLogoutTimeout 45
sun(switchServices)#
```

Associated MIB

```
amicfg.mib
```

Web path

- switchServices → cli → modify

httpd

Purpose

Configures the system to allow HTTP/HTTPS access to the Web interface.

Access mode

enable

Syntax

```
switchServices httpd
   [adminState {enabled | disabled}]
   [accessMode {http |https| both}]
   [httpPort portNumber]
   [httpsPort portNumber]
   [{serverKeyId text}]
   [{sessionTimeout minutes}]
   [auditLogging {on | off}]
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	**Optional**. Configures Web access on the system. If `enabled`, you can use the Web interface to manage the system; if `disabled`, you must use the CLI or SNMP to manage the system. The default setting is `disabled`. If you change this value while using the Web interface, the system terminates the current session immediately.
`accessMode {http | https | both}`	**Optional**. Configures the HTTP access mode on the system; HTTP, HTTPS, or both. The default setting is `both`.
`httpPort` *`portNumber`*	**Optional**. Configures the port that the system uses for HTTP communications. Valid values are from 1 to 65535; the default port is 80. If you change this value while using the Web interface, the system terminates the current session immediately. Close the Web browser, restart it, and log in again to access the Web interface.

Argument name	Description
`httpsPort` *`portNumber`*	**Optional**. Configures the port that the system uses for HTTPS communications. Valid values are from 1 to 65535; the default port is 443. If you change this value while using the Web interface, the system terminates the current session immediately. Close the Web browser, restart it, and log in again to access the Web interface.
`serverKeyId` *`text`*	**Optional.** Configures the HTTPS server's RSA key. The default setting is `webKey`.
`sessionTimeout` *`minutes`*	**Optional**. Configures the amount of time, in minutes, that a Web session can remain idle. When this time elapses, the system terminates the session. Valid values are from one to 60 minutes; the default setting is 10 minutes.
`auditLogging {on | off}`	**Optional**. Specifies whether the system generates an event when the system configuration changes during a Web Manager session. If set to `on`, the system generates an event when the configuration changes; if set to `off`, the system does not generate an event. The default setting is `on`.

Example

The following example shows how to enable HTTP access and set the session timeout to 60 minutes.

```
sun> enable
sun# switchServices
sun(switchServices)# httpd adminState enabled accessMode HTTP
sessionTimeout 60
```

Associated MIB

`wwwConfig.mib`

Web path

- switchServices → httpd → modify

show cli

Purpose

Displays the current configuration for the current CLI session and the default values for all sessions. This command does not support field filtering.

The Web interface displays only the fields that contain values that affect all CLI sessions.

Syntax

```
show switchServices cli
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# show cli
Rows (session):                      24
Auto logout timeout (session):       10
Echo Cmd:                            disabled
Prompt:                              sun
Line Wrap:                           79
Message On Timeout (session):        enabled
History Length                      50
Default Rows:                        24
Default Echo:                        disabled
Default Auto Logout Timeout:         10
Default Display Message On Timeout: enabled
Default Prompt:                      sun
Audit Logging:                       on
Default History Length              50
```

Output description

Field name	Description
`Rows (session)`	The number of rows the system displays at one time for the current session only. The valid range is from 0 to 250; the default setting is 24. If the row number in the display exceeds this number, the CLI displays the following message: `Press <return> or <space bar> for more, or 'q' to quit...` To view additional rows in the display, press [Return] or [Space Bar). To exit from the display, enter **q**.
`Auto logout timeout (session)`	The amount of time, in minutes, the current session can be idle. After this time elapse, the system logs you out automatically.
`Echo Cmd`	Indicates whether the system displays a command after you enter it. This value applies to the current session only.
`Prompt`	The CLI prompt that the system displays for the current session.
`Line Wrap`	The number of characters you can enter for a command line before it wraps to the next line. This value applies to the current session only.
`Message on Timeout`	Indicates whether the system displays a message when the current session times out and the system is going to terminate the session.
`History Length`	Indicates the number of history lines to maintain in the CLI. The default setting is 50 history lines.
`Default Rows`	The number of rows the system displays for all sessions.
`Default Echo`	Indicates whether the system displays a command after you enter the it. This value applies to all sessions.
`Default Auto Logout Timeout`	The amount of time, in minutes, that all sessions can be idle. After this time elapses, the system logs the user out automatically.

Field name	Description
`Default Display Message on Timeout`	Indicates whether the system displays a message when any session times out and the system is going to terminate the session.
`Default Prompt`	The CLI prompt that the system displays for all sessions.
`Audit Logging`	Indicates whether the system generates an event when the configuration changes during a CLI session.
`Default History Length`	Indicates the default number of history lines to save in the CLI. The default setting is 50 history lines.

Associated MIB

`amicfg.mib`

Web path

- switchServices → cli

show httpd

Purpose

Displays the current HTTP configuration. This command does not support field filtering.

Syntax

```
show switchServices httpd
```

Sample output

```
sun> enable
sun# switchServices
sun(config-switchServices)# show httpd
Administrative State: enabled
Access Mode:          both
HTTP Port:            80
HTTPS Port:           443
Server Key ID:        webKey
Server Key State:     validKey
Session Timeout:      10
Audit Logging:        on
Operational State:    up
Bytes Received:       28988
Bytes Sent:           220581
Total Sessions:       1
Current Sessions:     1
```

Output description

Field name	Description
`Administrative State`	Indicates whether the system allows Web access.
`Access Mode`	The type of access enabled: HTTP, HTTPS, or both.
`HTTP Port`	The port the system uses for HTTP communications.
`HTTPS Port`	The port the system uses for HTTPS communications.
`Server Key ID`	The HTTPS server's RSA key.

Field name	Description
`Server Key State`	The current state of the HTTPS server's operational key used to establish HTTPS sessions. • `validKey`: The key can be used to establish HTTPS sessions. • `invalidKey`: The key can not be used to establish HTTPS sessions.
`Session Timeout`	The amount of time, in minutes, a session can remain idle before the system terminates the session.
`Audit Logging`	Indicates whether the system generates an event when the configuration changes during a Web session.
`Operational State`	The current operational state of the web server on the system. Possible values are: `N/A`: The state of the web server is unknown. `down`: The web server on the system is not functioning. `up`: The web server on the system is functioning properly.
`Bytes Received`	The number of bytes the system received during HTTP/HTTPS communications.
`Bytes Sent`	The number of bytes the system sent during HTTP/HTTPS communications.
`Total Sessions`	The number of HTTP/HTTPS sessions opened since startup.
`Current Sessions`	The number of HTTP/HTTPS sessions currently open.

Associated MIB

`wwwConfig.mib`

Web path

- switchServices → httpd

show httpd sessions

Purpose

Displays the current HTTP session information.

Access mode

enable

Syntax

```
show switchServices httpd sessions
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# show httpd sessions
User Name                    admin
Session Id                   32602486770036634932
IP Address                   10.10.10.X
Start Time                   4/25/2003-08:38:04
Timeout                      120
Session Type                 HTTP
```

Output description

Field name	Description
User Name	Indicates the current user name logged on to the N2000 Series system.
Session ID	The system-assigned ID associated with the HTTP session.
IP Address	The source IP address associated with the current user name.
Start Time	The start time of the current HTTP session indicated by date, hours and minutes.
Time Remaining (sec)	Indicates the time remaining before the HTTP session times out due to inactivity.
Type	Indicates whether the session is using HTTP or HTTPS.

Associated MIB

`wwwConfig.mib`

Web path

- switchServices → httpd →sessions

Chapter 12. Software commands

Software command description

The software commands affect the system features that you can access and the software version that the system uses. The `switchServices key` command allows you to install a key that provides access to additional features that are not available in the basic software. The `switchServices version` command allows you to select the installed version of the software that the system uses at the next reboot.

Software key

Currently, the software key affects only the virtualization feature. When you install the system, the basic software installation allows you to create one operator-defined vSwitch. If you purchase a software key and configure the system to use the key you can create multiple vSwitches, with the actual number depending on the type of key.

To obtain a software key, you must supply the serial number of the system to Sun Technical Support. Use the `module systemBoard` argument with the `switchServices chassis module` command to obtain the serial number.

Software version

You can store two different versions of the software on the flash disk. When you change the software version, the change does not take effect until you restart the system.

Software command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`switchServices software` *`commandname`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Software command summary

Table 12-1 lists and briefly describes the software commands.

Table 12-1. Software command summary

Command name	Description
`key`	Configure a software key to enable additional software features.
`removecfg`	Remove configuration files from the flash file system.
`show key`	Display the current software key and the enabled software features.
`show version`	Display the software version currently in use and the version the system uses the next time it reboots.
`version`	Change the version of the software that the system uses when it reboots.

key

Purpose

Specifies the software key for the system. The software key allows access to additional features on the N2000 Series. The `show key` command displays the currently installed software key and the enabled features.

Access mode

config

Syntax

```
switchServices software key
   [softwareKey text]
```

Arguments

Argument name	Description
`softwareKey` *`text`*	**Optional.** Enables the software key when you enter the key that you received from Sun Technical Support. If you enter double quotes (“ “) instead of a text string, the system turns off the software key.

Example

The following example shows how to configure the system to use the software key, 01-0000-62399fa71d16833fc49a.

```
sun> enable
sun# config
sun(config)# switchServices
sun(config-switchServices)# software
sun(config-switchServices software)# key softwareKey
01-0000-62399fa71d16833fc49a
sun(config-switchServices software)#
```

Associated MIB

`license.mib`

Web path

- switchServices → software → key → modify

removecfg

Purpose

Removes the configuration files from the flash file system. Reboot the system after you remove the files. When you reboot the system, it creates a new configuration file with factory default settings called `cdb.dat`.

Access mode

config

Syntax

```
switchServices software removecfg
```

Arguments

There are no arguments for this command.

Example

The following example shows how to remove configuration files from the flash disk.

```
sun> enable
sun# config
sun(config)# switchServices
sun(config-switchServices)# software
sun(config-switchServices software)# removecfg
sun(config-switchServices software)#
```

Associated MIB

None

Web path

- switchServices → software → removecfg

show key

Purpose

Displays the key currently in use on the system, as configured with the `key` command. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices software key
```

Sample output

```
sun> switchServices
sun(switchServices)> software
sun(switchServices software)> show key
Software Key:                 01-0000-302c0
Software Feature:             virtualization
License Version:              V4
Virtualization Level:         N/A
```

Output description

Field name	Description
`Software Key`	The key text string that is configured on the system.
`Software Feature`	The software features associated with the key.
`License Version`	The unique license version associated with the key.
`Virtualization Level`	The administrative state of each software feature. Possible values are `on` or `off`.

Associated MIB

`license.mib`

Web path

- switchServices → software → key

show version

Purpose

Displays the version of the software currently in use and the version of the software the system uses the next time it reboots. The `version` command configures the software version that the system uses when it reboots.

Access mode

config

Syntax

```
show switchServices version
```

Sample output

```
sun> switchServices
sun(switchServices)> software
sun(switchServices software)> show version
Version currently running: V2_0R1
Version for next boot:     Not available
Available Vers:           V02_0A31608; V0_20A32304; V0_20A32683;
                          V0_20A32824; V0_20A32970
```

Output description

Field name	Description
`Version currently running`	The version of the software that the system is currently using.
`Version for next boot`	The version of the software that the system uses the next time it reboots.
`Available versions`	The versions of the software currently installed on the system.

Associated MIB

None

Web path

- switchServices → software → version

version

Specifies the software version to use the next time the system reboots. You can store two different versions of the software on the flash disk. The `show version` displays the version currently in use and the version the system uses the next time it reboots.

To determine which versions of the software are stored on the system, locate the software directory and files. The directory name has a syntax of V*x*_*yZ* and the file name has a syntax of V*x*_*yZp*:

- V*x* is the major release number.
- *y* is the minor release number.
- *Z* is the release type. The release types are:
  - A - Alpha (internal) release
  - B - Beta release
  - R - Standard release
- *p* is the patch number. The value of 1 indicates the first (non-patched) version of a release.

For example, the directory that contains the file for version 2.0 of the software would be V2_0R, and the file name would be V2_0R1.

Access mode

config

Syntax

```
switchServices software version
   [nextBootVersion softwareVersion]
```

Arguments

Argument name	Description
nextBootVersion *softwareVersion*	**Optional.** Specifies the software version to use on the next boot.

Example

The following example shows how to specify that the system should use V2.0R1 of the software instead of V1.1R1.

```
sun> enable
sun# config
sun(config)# switchServices
sun(config-switchServices)# software
sun(config-switchServices software)# version nextBootVersion V2_0R1
Version V2_0R1 will start on next boot
```

Associated MIB

None

Web path

- switchServices → software → version → modify

Part III. Security

The chapters in Part III describe the protocols and utilities that are available for securing the N2000 Series system.

Chapter 13. User administration commands

User administration description

The key tasks that the user administration commands allow you to complete are the following:

- Configuring user entries that determine who can access the system and their privileges on the system (for example, read-write or read-only access to the system or to a specific vSwitch). In addition, the user entry defines the authentication and authorization methods the system uses. You can configure multiple entries for a single user that define alternate authentication and authorization methods for a specific service or application.
- Configuring passwords for users, if using the internal database for authentication.
- Specifying the type of accounting information that you want to save on the N2000 Series, send to a Terminal Access Controller Access Control System (TACACS+) server, or send to a Remote Authentication Dial In User Service (RADIUS) server.
- Specifying remote TACACS+ servers or RADIUS servers that the N2000 Series uses for authentication, authorization, and accounting (AAA).
- Viewing summaries of accounting records, authentication attributes, and user activity. In addition, you can view the settings for all user administration configurations.

See Appendix D, “About authentication and authorization services,” and the *Sun N2000 Series Release 2.0 – System Administration Guide* for more information about configuring AAA.

User authentication on the N2000 Series

Using the user administration commands, you can create user entries that define authentication rules for users attempting to access the system. You can configure the N2000 Series to perform authentication or you can configure the system to send authentication requests to a TACACS+ server or to a RADIUS server.

The N2000 Series allows you to create and prioritize multiple entries associated with the same user name. This feature lets you configure multiple methods for authentication in the event that the first method attempted is not available. If the authentication method in the highest-priority entry that matches a user's credentials is not available (for example, a remote security server does not respond to a request), the system continues to evaluate each matching user entry, based on its user name, priority, and application rule until it finds an authentication method that responds to the authentication request.

Use the `user` command to configure user authentication rules. For a detailed description about how the N2000 Series performs AAA and how to configure user entries, see the *Sun N2000 Series Release 2.0 – System Administration Guide*.

User authorization on the N2000 Series

In addition to authentication, you configure basic authorization rules for each user entry. You can do the following:

- Assign profiles that define whether a user has read-write or read-only access to the commands in the command-line interface (CLI) or Web interface.
- Associate a user with a specific vSwitch and vRouter to further control which system characteristics the user can access.
- Configure authorization for services such as Telnet, HTTP, SSH, and console access.

Similar to configuring authentication in a user entry, you can configure multiple user entries with different authorization settings for a specific user. The system evaluates these settings based on the user name, priority, and application rule in the user entry. If authentication succeeds, but the service providing authorization is not available, the system searches for the highest-priority user entry that specifies an available service for authorization.

The system performs authorization at the service level only. If using a remote security server (for example, a TACACS+ server), the security server can overwrite configured settings in a user entry. For example, it can overwrite the privileges that the profiles set or the vSwitch the user can access.

Use the `user` command to configure user authorization rules. For a detailed description about how the N2000 Series performs AAA and how to configure user entries, see the *Sun N2000 Series Release 2.0 – System Administration Guide*.

Preconfigured user entries

The system includes the following preconfigured user entries:

- `admin` — The default administrator's user entry. This entry provides read and write access to all aspects of the system. By default, this entry does not require an assigned password; you can type any text for the password. For security purposes you should modify this entry so that users must enter a specific password.
- `.default` — A preconfigured user entry that allows multiple users to share a single entry. The system uses this user entry for authentication and authorization when a user tries to access the system in one of the following instances:
  - The system has no user entries that match the supplied user name.
  - The system has a matching name but does not have an appropriate service login argument set to `match`.

You cannot permanently delete the `.default` user entry. If you do not want to use the `.default` user entry, you can use the `user` command to set the `adminState` argument to `disabled`.

For additional details about user entries, see the *Sun N2000 Series Release 2.0 – System Administration Guide*.

User accounting on the N2000 Series

The N2000 Series can create the following types of accounting records for CLI sessions.

Accounting types

- Start records, indicating when a user session began
- Stop records, indicating when a user session ended
- Interim or update records, indicating that a user session is active at a specific point in time

Accounting methods

You can specify where the system saves accounting records. Possible locations are:

- In an internal log,
- At a remote TACACS+ server
- At a remote RADIUS server
- At a remote TACACS+ server and at a remote RADIUS server
- In an internal log and at a remote TACACS+ server
- In an internal log and at a remote RADIUS server
- In an internal log, at a remote TACACS+ server, and at a remote RADIUS server

Use the `userAdministration (root)` command to configure accounting.

For additional details about configuring accounting on the N2000 Series, see the *Sun N2000 Series Release 2.0 – System Administration Guide***.**

User administration command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`switchServices userAdministration` *`commandName`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

User administration command summary

Table 13-1 lists and briefly describes the user administration commands.

Table 13-1. User administration command summary

Command name	Description
`advanced`	Configure timers that: • Control the time allowed for internal processing activities • Specify the amount of time the system waits for an authentication task to complete
`server radius`	Specify remote RADIUS servers that the system uses for authentication, authorization, and accounting operations.
`server tacacs`	Specify remote TACACS+ servers that the system uses for authentication, authorization, and accounting operations.
`show`	Display top-level configuration, operational status, and statistics.
`show active users`	Display the configuration settings associated with one or more active users.
`show active users advanced`	Display a summary of the configuration settings for active users.
`show advanced`	Display advanced configuration, operational status, and statistics.
`show advanced attributes`	Display a summary of authentication attributes that the system exchanges with AAA servers.
`show log`	Display a summary of user activity in chronological order.
`show server radius`	Display configuration, operational status, and statistics for RADIUS servers that the N2000 Series uses for authentication, authorization, and accounting operations.

Table 13-1. User administration command summary (continued)

Command name	Description
`show server tacacs`	Display configuration, operational status, and statistics for TACACS+ servers that the N2000 Series uses for authentication, authorization, and accounting operations.
`show user`	Display the configurations for all user entries.
`show user detail`	Display a detailed view of the user configurations.
`show user summary`	Display a summarized list of all user entries.
`user`	Configure entries that the system uses to authenticate or authorize users trying to access the system.
`userAdministration (root)`	Configure the system-wide settings for: • Accounting information (log size, type, method) • Key information for sensitive data encryption • Server recovery time

System authentication and accounting basic configuration

Use the `userAdministration (root)` command and follow the steps below to configure the basic system-wide settings for authentication and accounting.

Table 13-2. Steps for configuring system-wide authentication and accounting

Step	Action
1.	Specify text that the system uses to generate an internal key that is used to encrypt sensitive data (for example, passwords and secrets). This action is required if you want to configure passwords and server secret keys.
2.	Optionally, specify the types of accounting records that you want the system to generate.
3.	Optionally, specify where you want to store accounting records.

Table 13-2. Steps for configuring system-wide authentication and accounting (continued)

Step	Action
4.	Optionally, specify the number of entries that you want to store in the internal accounting log. When the log reaches the maximum configured size, the system starts to overwrite existing records, starting with the oldest ones.
5.	Optionally, specify the update interval that the system waits before checking whether a session is still active and generating an accounting update record.
6.	Optionally, specify the amount of time that the system waits before changing the operational status of non-responsive TACACS+ servers or RADIUS servers to a status of `unknown`. This detects servers that are non-responsive and allows the system to try to use the servers at a later time.

Basic user entry configuration

Use the `user` command and follow the steps below to configure a basic user entry.

Table 13-3. Steps for configuring a basic user entry

Step	Action
1.	Specify a user name.
2.	Assign a priority. The system uses this value to determine the order in which it uses the user entries for authentication and authorization purposes.
3.	Optionally, specify the applications or services that can use this entry for authentication and authorization purposes. Setting an application argument to `ignore` means that the system does not use the entry for authentication or authorization if the user attempts to log in using that application. Instead, the system looks for the highest-priority user entry that has a setting of `match` for the application in use.
4.	Optionally, set the overall availability for the user entry (`adminState` argument). The default setting is `enabled`. When set to `enabled`, the system can use the entry for authentication and authorization. If you set `adminState` to `disabled`, the entry remains in the system's user database, but the system does not use it for authentication or authorization.

advanced

Purpose

Configures or modifies timers that control the time allowed for internal authentication processing tasks and the amount of time that the system waits for an authentication task to complete.

Access mode

enable

Syntax

```
switchServices userAdministration advanced
   [backgroundTimer integer]
   [maxRequestWaitTime integer]
```

Arguments

Argument name	Description
`backgroundTimer` *`integer`*	**Optional.** Configures the timer that determines the time allowed for various internal processing activities. Valid range is 30 through 300 seconds; the default is 30 seconds. Changing the value of this argument can affect the system's performance.
`maxRequestWaitTime` *`integer`*	**Optional.** Configures the time that the system waits for an authentication task to complete. Valid range is 30 through 300 seconds; the default is 60 seconds. The system uses this value for flow control purposes. If the time expires and the authentication task does not respond to a request, the system rejects the user authentication. Typically, you do not need to change this value. If you have a large number of security servers that the system can use, or if you have problems with network congestion, increasing this value can help you avoid unnecessary user authentication rejections.

Example

The following example shows how to change the background timer setting to 40 seconds.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(... userAdministration)# advanced backgroundTimer 40
sun(... userAdministration)#
```

The following example shows how to change the maximum request time to 80 seconds.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(... userAdministration)# advanced maxRequestWaitTime 80
sun(... userAdministration)#
```

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → advanced → modify

server radius

Purpose

Specifies a RADIUS server that the N2000 Series uses for authentication, authorization, and accounting (AAA).

The no form of this command deletes the RADIUS server configuration entry. The index argument is the only required argument for the no form of the command.

Access mode

user

Syntax

```
switchServices userAdministration server radius
   index integer
   ipAddress ipAddress
   secret passwordText
   [serverDisplayName text]
   [authenticationPort integer]
   [authorizationPort integer]
   [accountingPort integer]
   [timeout integer]
   [adminMode {disabled | authentication | authorization |
      authenticationAndAuthorization | accounting |
      authenticationAndAuthorizationAndAccounting}]
   [priority integer]
   [vendorIDOffset integer]
   [NASIdentifier text]
```

Arguments

Argument	Description
`index` *`integer`*	Specifies the numeric index for the server entries.
`ipAddress` *`ipAddress`*	Configures the IP address of the RADIUS server that you want the system to use for authentication, authorization, and accounting.
`secret` *`passwordText`*	Configures the shared secret or encryption key used to encrypt data when communicating with the RADIUS server. The shared secret must be 32 or fewer characters. This secret *must* match the secret configured on the RADIUS server. When trying to solve communication problems, it may be useful to set a null secret. In this situation, the system does not send encrypted data to the RADIUS server. To set a null secret, you can use two sets of double quotes (““).
`serverDisplayName` *`text`*	**Optional.** Identifies the RADIUS server for event reporting purposes. The server name must be 32 characters or fewer. The system does not use this name for DNS lookup purposes, but you can use the same name.
`authenticationPort` *`integer`*	**Optional.** Sets the number of the UDP port that the RADIUS server uses for authentication and authorization requests. The default is 1812.
`authorizationPort` *`integer`*	**Optional.** Sets the number of the UDP port that the RADIUS server uses for authorization requests. The default is 1812.
`accountingPort` *`integer`*	**Optional.** Sets the number of the UDP port that the RADIUS server uses for accounting requests. The default is 1812.
`timeout` *`integer`*	**Optional.** Configures the maximum amount of time (in seconds) that the Sun Application Switch waits for a RADIUS server to respond to a request. The range is 1 through 10 seconds. The default is 2.
`retries` *`integer`*	**Optional.** Specifies the maximum number of retries before the RADIUS server is deemed unavailable. The default is 2.

Argument	Description
`adminMode {disabled | authentication | authorization | authenticationAndAuthorization | accounting | authenticationAndAuthorizationAndAccounting}`	**Optional**. Specifies the type of requests that the N2000 Series sends to the RADIUS server. Valid values are: `disabled`: The system does not send requests to the RADIUS server. `authentication`: The system sends authentication requests only to the RADIUS server. `authorization`: The system sends authorization requests only to the RADIUS server. `authenticationAndAuthorization`: The system sends authentication and authorization requests to the RADIUS server. `accounting`: The system sends accounting requests only to the RADIUS server. `authenticationAndAuthorization AndAccounting`: The system sends all authentication, authorization, and accounting records to the RADIUS server. The default is `authenticationAndAuthorization AndAccounting`.

Argument	Description
`priority` *`integer`*	**Optional.** Groups and prioritizes RADIUS servers. The valid range is 1 (highest) to 10 (lowest). The default is 1. **Authentication** For authentication requests, if you assign the same priority to more than one authentication server, the system uses each server in a group in a round-robin fashion. **Accounting** For accounting records, the system always tries to send records to the accounting server that received the original start record. If the accounting server is not available and you assign the same priority to more than one accounting server, the system uses the next server in the group. Once the system accesses an available server, it tries to send subsequent accounting records to that server. **General** The system attempts to use the RADIUS server with a priority of 1 first. If this server (or server group) is unavailable, the system tries the next highest-priority server (or server group) until it finds a server that it can access. If the system cannot find an available RADIUS server, it tries to find another user entry that it can use for authentication or authorization. If it cannot find another valid user entry, the system rejects the authentication or authorization request.
`vendorIDOffset` *`text`*	**Optional.** Specifies the offset for Sun-specific attributes. O is used for vendor-specific attribute encoding.
`NASIdentifier` *`text`*	**Optional.** Specifies a text string to identify the Sun Application Switch to this RADIUS server.

Delete filters

```
no switchServices userAdministration server radius
   index integer
   [ipAddress ipAddress]
   [secret passwordText]
   [serverDisplayName text]
   [authenticationPort integer]
   [authorizationPort integer]
   [accountingPort integer]
   [timeout integer]
```

```
[adminMode {disabled | authentication | authorization |
   authenticationAndAuthorization | accounting |
   authenticationAndAuthorizationAndAccounting}]
[priority integer]
[vendorIDOffset integer]
[NASIdentifier text]
```

Example

The following example shows how to specify a RADIUS server that the N2000 Series uses for authentication requests. The priority assigned to the server is 1. The system attempts to send authentication requests to this server first. If this server is not available, the system attempts to use a lower-priority RADIUS server.

You should consult with the RADIUS server administrator to obtain the correct value for `secret`. See the *Sun N2000 Series Release 2.0 – System Administration Guide* for more information about specifying a RADIUS server.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# server
sun(switchServices userAdministration server)# radius index 1
ipAddress 20.20.20.20 secret radiussecret adminMode authentication
priority 1

sun(switchServices userAdministration)# show server radius
Index:                1
IP Address:           20.20.20.20
Server Name:          Authentication1
Authentication Port: 49
Authorization Port:  49
Accounting Port:     49
Timeout:              2
Retries:              2
Admin Mode:           authentication
Priority:             1
Vendor ID:            N/A
NAS ID:               Sun-Nauticus 8857
Operational State:    unknown
RX Packets:           0
RX Bytes:             0
TX Packets:           0
TX Bytes:             0
```

Associated MIB

`radiusAuthentication.mib`

Web path

- switchServices → userAdministration → server → radius → add
- switchServices → userAdministration → server → radius → copy
- switchServices → userAdministration → server → radius → modify
- switchServices → userAdministration → server → radius → delete

server tacacs

Purpose

Specifies a TACACS+ server that the N2000 Series uses for authentication, authorization, and accounting (AAA). You can configure the system to use up to 10 TACACS+ servers.

The no form of this command deletes the TACACS+ server configuration entry. The index argument is the only required argument for the no form of the command.

Access mode

user

Syntax

To create a TACACS+ server configuration:

```
switchServices userAdministration server tacacs
   index integer
   ipAddress ipAddress
   secret text
   [serverDisplayName text]
   [tcpPortAuthentication integer]
   [tcpPortAuthorization integer]
   [tcpPortAccounting integer]
   [timeout integer]
   [adminState {disabled | authentication | authorization |
      authenticationAndAuthorization | accounting |
      authenticationAndAuthorizationAndAccounting}]
   [priority integer]
```

To modify a TACACS+ server configuration:

```
switchServices userAdministration server tacacs
   index integer
   [ipAddress ipAddress]
   [secret text]
   [serverDisplayName text]
   [tcpPortAuthentication integer]
   [tcpPortAuthorization integer]
   [tcpPortAccounting integer]
   [timeout integer]
```

```
[adminState {disabled | authentication | authorization |
   authenticationAndAuthorization | accounting |
   authenticationAndAuthorizationAndAccounting}]
[priority integer]
```

Arguments

Argument name	Description
`index integer`	Specifies the index entry that identifies the TACACS+ server configuration entry. Valid range is 1 through 10.
`ipAddress ipAddress`	Configures the IP address of the TACACS+ server that you want the system to use for authentication, authorization, and accounting.
`secret text`	Configures the shared secret or encryption key used to encrypt data when communicating with the TACACS+ server. The shared secret must be 32 or fewer characters. This secret *must* match the secret configured on the TACACS+ server. When trying to solve communication problems, it may be useful to set a null secret. In this situation, the system does not send encrypted data to the TACACS+ server. To set a null secret, you can use two sets of double quotes (““).
`serverDisplayName text`	**Optional**. Specifies a textual name that identifies the TACACS+ server for event reporting purposes. The server name must be 32 or fewer characters. The system does not use this name for DNS lookup purposes, but you can use the same name.
`tcpPortAuthentication integer`	**Optional**. Enters the number of the TCP port that the TACACS+ server uses for authentication. The default is 49.
`tcpPortAccounting integer`	**Optional**. Enters the number of the TCP port that the TACACS+ server uses for accounting requests. The default is 49.
`tcpPortAuthorization integer`	**Optional**. Enters the number of the TCP port that the TACACS+ server uses for authorization requests. The default is 49.
`timeout integer`	**Optional**. Configures the maximum amount of time, in seconds, that the N2000 Series waits for a TACACS+ server to respond to a request. The range is from 1 through 10. The default is 2.

Argument name	Description
`adminState {disabled | authentication | authorization | authenticationAndAuthorization | accounting | authenticationAndAuthorization AndAccounting}]`	**Optional**. Specifies the type of requests that the N2000 Series sends to the TACACS+ server. Valid values are: `disabled`: The system does not send requests to the TACACS+ server. `authentication`: The system sends authentication requests only to the TACACS+ server. `authorization`: The system sends authorization requests only to the TACACS+ server. `authenticationAndAuthorization`: The system sends authentication and authorization requests to the TACACS+ server. `accounting`: The system sends accounting requests only to the TACACS+ server. `authenticationAndAuthorization AndAccounting`: The system sends all authentication, authorization, and accounting records to the TACACS+ server. The default is `authenticationAndAuthorization AndAccounting`.

Argument name	Description
`priority` *`integer`*	**Optional**. Groups and prioritizes TACACS+ servers. Valid range is 1 (highest priority) through 10 (lowest priority). The default is 1. **Authentication** For authentication requests, if you assign the same priority to more than one authentication server, the system uses each server in a group in a round-robin fashion. **Accounting** For accounting records, the system always tries to send records to the accounting server that received the original start record. If that accounting server is not available and you assign the same priority to more than one accounting server, the system uses the next server in the group. Once the system accesses an available server, it tries to send subsequent accounting records to that server. **General** The system attempts to use TACACS+ servers with a priority of 1 first. If this server (or server group) is unavailable, the system tries the next highest-priority server (or server group), until it finds a server it can access. If the system cannot find an available TACACS+ server, it tries to find another user entry that it can use for authentication or authorization. If it cannot find another valid user entry, the system rejects the authentication or authorization request.

Delete filters

See the `show server tacacs` command for additional argument descriptions.

```
no switchServices userAdministration server tacacs
   index integer
   [ipAddress ipAddress]
   [secret text]
   [serverDisplayName text]
   [tcpPortAuthentication integer]
   [tcpPortAuthorization integer]
   [tcpPortAccounting integer]
   [timeout integer]
```

```
[adminState {disabled | authentication | authorization |
   authenticationAndAuthorization | accounting |
   authenticationAndAuthorizationAndAccounting}]
[priority integer]
[operConnectionState {unknown | connectOK | connectError |
   rxTimeout | rxError | txError}
[packetsSent integer]
[packetsReceived integer]
[bytesSent integer]
[bytesReceived integer]
[encryptionStatus {unknown | encryptionOn | encryptionOff)
```

Example

The following example shows how to specify a TACACS+ server that the N2000 Series uses for authentication requests. The priority assigned to the server is 1. The system attempts to send authentication requests to this server first. If this server is not available, the system attempts to use a lower-priority TACACS+ server.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# server
sun(switchServices userAdministration server)# tacacs index 1
ipAddress 20.20.20.20 secret serversecret adminMode authentication
priority 1

sun(switchServices userAdministration)# show server tacacs
Index:                1
IP Address:           20.20.20.20
Server Name:          Authentication1
Authentication Port: 49
Authorization Port:  49
Accounting Port:     49
Secret:               --------
Timeout:              2
Admin Mode:           authentication
Priority:             1
Encryption Status:    encryptionOn
Operational State:    unknown
RX Packets:           0
RX Bytes:             0
TX Packets:           0
TX Bytes:             0

sun(switchServices userAdministration)#
```

Associated MIB

`tacacsAuthentication.mib`

Web path

- switchServices → userAdministration → server → tacacs → add
- switchServices → userAdministration → server → tacacs → copy
- switchServices → userAdministration → server → tacacs → modify
- switchServices → userAdministration → server → tacacs → delete

show

Purpose

Displays a summary of configuration settings and operational status for authentication, authorization, and accounting operations. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices userAdministration
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show
Key Info:                   N/A
Key State:                  invalid
Accounting Types:           startsAndStopsAndUpdates
Accounting Method:          internal
Accounting Log Size:        100
Accounting Update Interval: 10
Server Recovery Timer:      90
Current Request Handles:    2
Current Queued Deletes:     0
Total Requests:             308
New Requests:               35
Authentication Requests:    34
Authentication Successes:   34
Authentication Failures:    0
Authorization Requests:     34
Authorization Successes:    34
Authorization Failures:     0
Accounting Requests:        172
SshdPriv Requests:          0
Delete Requests:            33
Errored Requests:           0
General Errors:             0
Application Denied Access:  0
Memory Errors:              0
```

```
Rng Errors:                0
User Entries:              2
Log Entries:               100
```

Output description

Field name	Description
`Key Info`	The random text that the system uses to encrypt passwords for user entries. The system displays the characters in the key as hyphens (---).
`Key State`	The state of the internal authentication key that the system uses to encrypt sensitive data, either `valid` (the system generated the key) or `invalid` (the system has not yet generated the key).
`Accounting Types`	The types of accounting records that the system logs. Possible values are: • `start` • `startsAndStops` • `startsAndStopsAndUpdates`
`Accounting Method`	The location where the system stores accounting records. Possible values are: • `none` • `internal` • `tacacs` • `radius` • `tacacsAndRadius` • `internalAndTacacs` • `internalAndRadius` • `internalAndTacacsAndRadius`
`Accounting Log Size`	The number of entries the system stores in the accounting log. When the set threshold is reached, the system starts overwriting the records, starting with the oldest record.
`Accounting Update Interval`	The time, in minutes, between accounting interval updates, when the system determines whether a session is still active.
`Server Recovery Timer`	The interval, in seconds, after which the system sets the status of non-responding TACACS+ servers or RADIUS servers to `unknown`. This allows the system to try to access them at a later time.

Field name	Description
Current Request Handles	The number of currently tracked requests that the system received.
Current Queued Deletes	The number of currently tracked requests that are marked for deletion.
Total Requests	The total number of requests that the authentication system received (using a request handle) for either authentication, authorization, or accounting.
New Requests	The total number of new requests that the system received.
Authentication Requests	The number of authentication requests that the system received.
Authentication Successes	The number of authentication requests that succeeded.
Authentication Failures	The number of authentication requests that failed.
Authorization Requests	The number of authorization requests that the system received.
Authorization Successes	The number of authorization requests that succeeded.
Authorization Failures	The number of authorization requests that failed.
Accounting Requests	The number of accounting requests that the system received.
SshdPriv Requests	The total number of requests for SSH privilege values that the system received.
Delete Requests	The total number of delete requests that the system received.
Errored Requests	The number of requests that the system could not process because they contained errors.
General Errors	The total number of internal errors that occurred on the system. The typical cause of these errors is lack of resources, or unexpected conditions.
Application Denied Access	The number of times an application, such as HTTP, denied access to a user. Typically, this is the result of an invalid vSwitch, invalid profile, or an invalid vSwitch profile combination.
Memory Errors	The number of unsuccessful memory allocation requests that occurred on the system.
Rng Errors	The number of random number errors that occurred on the system.
User Entries	The number of user entries configured on the system.
Log Entries	The number of accounting entries in the internal log.

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration

show active users

Purpose

Displays the configuration settings associated with one or more active users.

Access mode

user

Syntax

```
show switchServices userAdministration active users
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show active users
Req ID:              0042
User Name:           mcnealy
Remote Info:         N/A
App Type:            SSH
Login Time:          Thu Sep 9 09:11:21 EDT 2004
```

Output description

Field name	Description	Filter name
`Req ID`	Internal numeric ID for user records.	`reqID` *`integer`*
`User Name`	The user name associated wit the entry.	`userName` *`text`*
`Remote Info`	Information about the remote connection.	`remoteInfo`
`App Type`	The application used to access the Sun Application Switch, either a console connection, Telnet, SSH, or the Web interface.	`appType` *`text`*
`Login Time`	The date and time that the user logged in and the session began.	`logintime` *`textstring`*

Associated MIB

`activeUsers.mib`

Web path

- switchServices → userAdministration → activeUsers

show active users advanced

Purpose

Displays a summary of the configuration settings for active users.

Access mode

user

Syntax

```
show switchServices userAdministration active users advanced
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show active users advanced
Req ID:              0042
User Name:           mcnealy
Session ID:          004
Elapsed Time:        45
```

Output description

Field name	Description	Filter name
`Req ID`	Internal numeric ID for user records.	`reqID` *`integer`*
`User Name`	The user name associated with the entry.	`userName` *`text`*
`Session ID`	The unique identifier associated with the system.	`sessionID` *`integer`*
`Elapsed Time`	The amount of time, in seconds, that the session has been active (the elapsed time).	`elapsedTime` *`integer`*

Associated MIB

`activeUsersAdvanced.mib`

Web path

- switchServices → userAdministration → active users advanced

show advanced

Purpose

Displays the operational values for the authentication task. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices userAdministration advanced
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show advanced
Background Timer:              30
Request Max Wait Time:         60
Dropped Requests:              0
Currently Spawned Tasks:       0
Top Level Flow Control:        0
Low Level Flow Control:        0
Accounting Level Flow Control: 0
```

Output description

Field name	Description
`Background Timer`	The time, in seconds, allowed for various internal authentication processing activities.
`Request Max Wait Time`	The maximum time, in seconds, the system waits for authentication, authorization, and accounting tasks to complete.
`Dropped Requests`	The number of requests the system dropped because of flow control issues.

Field name	Description
`Currently Spawned Tasks`	The number of internal tasks spawned to handle multiple authentication, authorization, and accounting tasks.
`Top Level Flow Control`	The number of times internal, top-level flow control was initiated.
`Low Level Flow Control`	The number of times internal, low-level flow control was initiated.
`Accounting Level Flow Control`	The number of times internal, accounting-level flow control was initiated.

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → advanced

show advanced attributes

Purpose

Displays a summary of the types of authentication and authorization attributes that the N2000 Series exchanges with an AAA server. The system supports the use of TACACS+ servers and RADIUS servers.

Access mode

user

Syntax

```
show switchServices userAdministration advanced attributes
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show advanced attributes
authenticationID: 44
attrName:         task_id
type:             string
description:      The unique session-id, which remains the same for the life
                  of the session
range:            A text string used to identify the session
vendorName:       genericRadiusAttribute
vendorId:         0
attrId:           44

authenticationID: 46
attrName:         elapsed_time
type:             number
description:      The number of the seconds that the session has been active
range:            raw number of seconds, representing 'session duration
vendorName:       genericRadiusAttribute
vendorId:         0
attrId:           46
authenticationID: 257
attrName:         start_time
type:             number
description:      The timestamp of the session start, in epoch seconds
range:            the value is in epoch seconds (seconds since Jan 1, 1970)
vendorName:       genericTacacsAttribute
vendorId:         0
```

```
attrId:           1

authenticationID: 258
attrName:         stop_time
type:             number
description:      The timestamp of the session stop, in epoch seconds
range:            the value is in epoch seconds (seconds since Jan 1, 1970)
vendorName:       genericTacacsAttribute
vendorId:         0
attrId:           2

authenticationID: 259
attrName:         update_time
type:             number
description:      The timestamp of the session update, in epoch seconds
range:            the value is in epoch seconds (seconds since Jan 1, 1970)
vendorName:       genericTacacsAttribute
vendorId:         0
attrId:           3

authenticationID: 301
attrName:         service
type:             string
description:      the type of application, or type of service being authorized
range:            ie consoleLogin, telnetLogin, sshdSession, sshdLogin,
                  httpLogin, xmlAccess, HTTPS
vendorName:       Sun/NauticusNetworks
vendorId:         8857
attrId:           1

authenticationID: 302
attrName:         vswitchName
type:             string
description:      the 'vswitchName' or ID associated with the user for CLI/UI
                  access
range:            usually 1-64 characters matching specificly configured vswitch
                  users
vendorName:       Sun/NauticusNetworks
vendorId:         8857
attrId:           2

authenticationID: 303
attrName:         profileName
type:             string
description:      the profile name associated with the vswitch user for CLI/UI
                  access
range:            usually 1-32 chars identifying an existing profile. Typically,
                  systemAdmin, systemOperator, vSwitchAdmin, vSwitchOperator
vendorName:       Sun/NauticusNetworks
vendorId:         8857
attrId:           3
```

```
authenticationID: 304
attrName:         sshdPrivs
type:             number
description:      the sshd privileges associated with a user for the
                  'sshdSession' service
range:            none(1), session(2), sftpRead(3), sftpReadWrite(4)
vendorName:       Sun/NauticusNetworks
vendorId:         8857
attrId:           4
```

Output description

Field name	Description	Filter name
`authentication ID`	A numeric value identifying an AAA attribute. The system uses this value for internal tracking purposes.	`authenticationId` *`integer`*
`attrName`	The name of the attribute defined in the protocol or by a vendor.	`attributeName` *`text`*
`type`	The type (`string`, `number` or `IP address`) that defines the attribute's encoding and decoding. The system uses this setting to interpret the attribute value.	`attributeType {string | number | ipAddress}`
`description`	A textual description of the attribute.	`description` *`text`*
`range`	A textual description of the range of an attribute's value.	No filter available
`vendorName`	The name of the protocol or vendor in which the attribute is defined. Possible values are: `genericTacacsAttribute:` The attribute is a TACACS+-defined attribute. `genericRadiusAttribute:` The attribute is a RADIUS-defined attribute. `Sun/NauticusNetworks:` The attribute is a vendor-specific attribute for Sun Microsystems, Inc.	`vendorName {genericTacacsAttribute | genericRadiusAttribute | Sun/NauticusNetworks}`
`vendorId`	The identification number associated with the attribute. A generic attribute has a value of zero (0).	No filter available

Field name	Description	Filter name
`attrId`	The attribute number identified under this vendor.	`attributeNumber` *`integer`*

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → advanced → attributes

show log

Purpose

Displays a summary of the accounting entries in the internal log.

Access mode

user

Syntax

```
show switchServices userAdministration log
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show log
Index:           78
User:            admin
Type:            stop
Application:     http
Start Time:      1061304991
Update Time:     0
End Time:        1061305652
Duration:        661
DateTime:        Tue Aug 19 15:07:32 2003
Session ID:      f411048700000013
TerminationCode: none

Index:       2
User:        admin
Type:        update
Application: telnet
Start Time:  1029955735
Update Time: 1029956340
End Time:    0
Duration:    605
DateTime:    Wed Aug 21 18:59:00 2002
Session ID:  e5b3fba900000003
TerminationCode: none
```

Output description

Field name	Description	Filter name
`Index`	The number of the entry in the log.	`index` *`integer`*
`User`	The user name associated with the log entry.	`user` *`text`*
`Type`	The type of accounting record associated with the log entry: `start`: The record created when a session begins. `stop`: The record created when a session ends. `update`: The record created when the session update timer expires and the session is still active.	`actgRecordType {start | stop | update}`
`Application`	The application used to access the N2000 Series, either a console connection, Telnet, SSH, or the Web interface (HTTP/HTTPS).	`applicationType {console | telnet | sshd | http | https}`
`Start Time`	The time when the session started. The log displays the time in epoch time (the time, in seconds, elapsed since 1 January 1970 00:00:00).	`startTime` *`integer`*
`Update Time`	The time, in epoch time, when the system generated an update record. This occurs when the `systemUpdateTimer` expires.	`updateTime` *`integer`*
`End Time`	The time, in epoch time, when the session ended.	`endTime` *`integer`*
`Duration`	The amount of time the session was active (the elapsed time between the start and update or end of the session).	`currDuration` *`integer`*
`DateTime`	The date and time the system generated the record.	`currDateTime` *`DDDMMM hh:mm:ss YYYY`*

Field name	Description	Filter name
`Session ID`	The unique session identifier associated with the system.	`sessionID` *`integer`*
`Termination Code`	The termination code for the session end.	`termCode {none | appUserExit | appHostExit | appIdleTimeout | appSessionTimeout | appError | abbAbort | noAccessInvalidVSwitch | noAccessInvalidProfile | noAccessInvalidVSwitchProfile | adminReset}`

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → log

show server radius

Purpose

Displays information about the configuration of the N2000 Series for RADIUS authentication.

Access mode

user

Syntax

```
show switchServices userAdministration server radius
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show server radius
Index:                1
IP Address:           20.20.20.20
Server Name:          Authentication1
Authentication Port: 49
Authorization Port:  49
Accounting Port:     49
Timeout:              2
Retries:              2
Admin Mode:           authentication
Priority:             1
Vendor ID:            N/A
NAS ID:               Sun-Nauticus 8857
Operational State:    unknown
RX Packets:           0
RX Bytes:             0
TX Packets:           0
TX Bytes:             0
```

Output description

Field name	Description	Filter name
Index	A number that uniquely identifies the server configuration. The range is 1 through 10. There is no default.	index *integer*
IP Address	The IP address for the RADIUS server.	IP address *ipAdress*
Secret	The shared-secret data for the RADIUS server. This value *must* match the secret configured on the security server.	secret *text*
Server Display Name	The textual name that identifies the RADIUS server for event reporting purposes.	serverDisplayName *text*
Authenti-cation Port	The number of the UDP port that the RADIUS server uses for authentication requests.	authenticationPort *integer*
Authoriza-tion Port	The number of the UDP port that the RADIUS server uses for authorization requests.	authorizationPort *integer*
Accounting Port	The number of the UDP port that the RADIUS server uses for accounting requests.	accountingPort *integer*
Timeout	The maximum amount of time (in seconds) that the Sun Application Switch waits for a RADIUS server to respond to a request.	timeout *integer*
Retries	The maximum number of retries before the RADIUS server is deemed unavailable.	retries *integer*
Admin Mode	The AAA function that the RADIUS server performs for the system.	adminMode {disabled | authentication | accounting | authorization | authenticationAndAuthorization | authenticationAndAuthorizationAndAccounting
Priority	The priority of the RADIUS server. The range is 1 (highest) through 10 (lowest). The default is 1.	priority *integer*
Vendor ID Offset	The offset for Sun-specific attributes. O is used for vendor-specific attribute encoding.	vendorIDOffset *text*
NAS Identifier	A string that identifies the Sun Application Switch to this RADIUS server.	NASIdentifier *text*

Field name	Description	Filter name
Operational State	The last known state of connection for the specified server. unknown: The RADIUS server was never used or was not available the last time the system sent a request to it but may be available now. The system will try to connect to the server, when necessary. connectOK: The RADIUS server was available the last time the system sent a request to it. connectError: The system encountered an error when it tried to connect to the RADIUS server. rxTimeout: The system did not receive a response from the RADIUS server within the allowed time. rxError: The system received an error when it received a response from the RADIUS server. txError: The system encountered an error while transmitting a request to the RADIUS server.	operState {unknown | connectOK | connectError | rxTimeout| rxError| txError}
RX Packets	The number of packets received from the server.	packetsSent
RX Bytes	The number of bytes received from the server.	bytesSent
TX Packets	The number of packets sent to the server.	packetsReceived
TX Bytes	The number of bytes sent to the server.	bytesReceived

Associated MIB

radiusauthentication.mib

Web path

- switchServices → userAdministration → server → radius

show server tacacs

Purpose

Displays configuration information and operational status for the TACACS+ servers that the N2000 Series uses for authentication, authorization, and accounting.

Access mode

user

Syntax

```
show switchServices userAdministration server tacacs
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show server tacacs
Index:               1
IP Address:          20.20.20.20
Server Display Name: Authentication1
Authentication Port: 49
Authorization Port:  49
Accounting Port:     49
Secret:              --------
Timeout:             2
Admin Mode:          authentication
Priority:            1
Encryption Status:   encryptionOn
Operational State:   unknown
RX Packets:          0
RX Bytes:            0
TX Packets:          0
TX Bytes:            0
```

Output description

Field name	Description	Filter name
Index	A number that uniquely identifies the server configuration. The range is 1 through 10 There is no default setting.	index *integer*
IP Address	The IP address for the TACACS+ server.	ipAddress *ipAddress*
Secret	The shared-secret data for the TACACS+ server. This value *must* match the secret configured on the security server.	secret *text*
Server Display Name	The textual name used to identify the TACACS+ server for event reporting purposes.	serverDisplayName *text*
Authentication Port	The number of the TCP port that the TACACS+ server uses for authentication requests.	tcpPortAuthentication *integer*
Authorization Port	The number of the TCP port that the TACACS+ server uses for authorization requests.	tcpPortAuthorization *integer*
Accounting Port	The number of the TCP port that the TACACS+ server uses for accounting requests.	tcpPortAccounting *integer*
Timeout	The amount of time, in seconds, that the system waits for a response from a TACACS+ server.	timeout *integer*
Admin State	The AAA function that the TACACS+ server performs for the system.	adminState {disabled | authentication | accounting | authorization| authenticationAndAuthorization | authenticationAndAuthorizationAndAccounting}
Priority	The priority of the TACACS+ server. The range is from 1 (highest priority) through 10 (lowest priority); the default is 1.	priority *integer*

Field name	Description	Filter name
`Operational State`	The last known connection state of the TACACS+ server: `unknown`: The TACACS+ server was never used or was not available the last time the system sent a request to it but may be available now. The system will try to connect to the server, when necessary. `connectOK`: The TACACS+ server was available the last time the system sent a request to it. `connectError`: The system encountered an error when it tried to connect to the TACACS+ server. `rxTimeout`: The system did not receive a response from the TACACS+ server within the allowed time. `rxError`: The system received an error when it received a response from the TACACS+ server. `txError`: The system encountered an error while transmitting a request to the TACACS+ server.	`operConnectionState {unknown | connectOK | connectError | rxTimeout | rxError | txError}`
`TX Packets`	The number of packets that the system sent to the TACACS+ server.	`packetsSent` *`counter32`*
`RX Packets`	The number of packets that the system received from the TACACS+ server.	`packetsReceived` *`counter32`*
`TX Bytes`	The number of bytes that the system sent to the TACACS+ server.	`bytesSent` *`counter32`*

Field name	Description	Filter name
`RX Bytes`	The number of bytes that the system received from the TACACS+ server.	`bytesReceived` *`counter32`*
`Encryption Status`	The flag indicating whether the packets that the TACACS+ server sent are encrypted. The valid values are: `unknown`: The system could not determine whether a packet it received from the TACACS+ server was encrypted. `encryptionOn`: The TACACS+ server sent encrypted packets to the system. `encryptionOff`: The TACACS+ server sent unencrypted packets to the system.	`encryptionStatus {unknown | encryptionOn | encryptionOff}`

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → server → tacacs

show user

Purpose

Displays a detailed list of the user entries configured on the system.

Access mode

user

Syntax

```
show switchServices userAdministration user
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show user
User Name:             .default
Priority:              1
User Password:
Matched Services:      consoleLogin,telnetLogin,sshd,sshdLogin,httpLogin,
                        httpsLogin,xml
Ignored Services:
Authentication Method: tacacs
Authorization Method:  alwaysAccept
Profile Name:          vSwitchAdmin
Sshd Privileges:       session
vSwitch:
Admin State:           enabled

User Name:             admin
Priority:              1
User Password:
Matched Services:      consoleLogin,telnetLogin,sshd,sshdLogin,httpLogin,
                        httpsLogin, xml
Ignored Services:
Authentication Method: alwaysAccept
Authorization Method:  alwaysAccept
Profile Name:          systemAdmin
Sshd Privileges:       sftpReadWrite
vSwitch:               system
Admin State:           enabled
```

Output description

Field name	Description	Filter name
`User Name`	The textual name that identifies the user entry. The user name must be 32 or fewer characters. You use this value when the system prompts you for a user name during the log in process.	`userName` *`text`*
`Priority`	The priority to the user entry. If a user has multiple entries, the system uses the highest-priority entry that: • Has a user name that matches the user name in the authentication request. • Allows authentication when you use a specific application to connect to the system. The N2000 Series then applies the authentication method defined in the user entry. The following conditions apply: • If the authentication and authorization succeeds, you can log in to the system and monitor or configure the system, based on the profile and vSwitch associated with the user entry. • If the authentication fails (the system cannot find an entry that matches a user's credentials), the login process ends. • If the defined authentication service for the user entry is not available, the system looks for a lower-priority user entry that matches the user name and application setting.	`priority` *`integer`*
`User Password`	The password associated with the user entry. The system uses this value only if you use the internal database for authentication.	`password` *`text`*

Field name	Description	Filter name
`Matched Services`	Lists all of the services for which the system uses this entry for authentication.	`matchedServices` *`text`*
`Ignored Services`	Lists all of the services for which the system does not use this entry for authentication.	`ignoredServices` *`text`*
`Authentication Method`	The type of authentication configured for this user entry. The possible values are: `alwaysAccept`: The system authenticates the user no matter what type of password the user enters. `alwaysReject`: The system never allows the user access to the system. `internalUserTable`: The system uses the user names and passwords stored in its internal database to authenticate the user, when it uses this entry. `tacacs`: The system uses a remote TACACS+ server to authenticate the user. `radius`: The system uses a remote RADIUS server to authenticate the user.	`authenticationMethod {alwaysAccept | alwaysReject | internalUserTable |tacacs | radius}`

Field name	Description	Filter name
`Authorization Method`	The type of authorization configured for this user entry. The possible values are: `alwaysAccept`: The system allows the user to use all supported services (for example, Telnet, HTTP, SSH, or console access). `alwaysReject`: The system does not allow the user to use any services. `tacacs`: The system uses a remote TACACS+ server to obtain the authorization settings for user. The TACACS+ server can overwrite user entry settings, for example access to a specific vSwitch or privileges that profiles allow. `radius`: The system uses a remote RADIUS server to obtain the authorization settings for user. The RADIUS server can overwrite user entry settings similar to the server above.	`authorizationMethod {alwaysAccept | alwaysReject | tacacs | radius}`

Field name	Description	Filter name
`Profile Name`	The profile associated with this user entry. The possible values are: `systemAdmin`: Provides read-write access to the system level commands and the system vSwitch. `systemOperator`: Provides read-only access to the system level commands and the system vSwitch. `vSwitchAdmin`: Provides read-write access to the commands for a specific vSwitch only. `vSwitchOperator`: Provides read-only access to the commands for a specific vSwitch only. `unspecified`: Specifies an invalid profile. Use this profile if you want to ensure that only the TACACS+ server or the RADIUS server provides this value. If the TACACS+ server or the RADIUS server does not return the appropriate attribute, the authentication fails.	`profileName {systemAdmin | systemOperator | vSwitchAdmin | vSwitchOperator | unspecified}`

Field name	Description	Filter name
`Sshd Privileges`	The SSH privileges associated with this user entry. The possible values are: `none`: Allows no SSH access to the system or ability to use SFTP. `session`: Allows a user to establish an SSH session. `sftpRead`: Allows a user to establish an SSH session (session privileges) and use SFTP to transfer files from the switch. `sftpReadWrite`: Allows a user to establish an SSH session (session privileges) and use SFTP to transfer files from and to the switch. If using a TACACS+ server or a RADIUS server for authorization, the TACACS+ or RADIUS configuration for the user can overwrite this setting. If you want to ensure that only the TACACS+ server or the RADIUS server provides this value, set this argument to `none`. If the TACACS+ server or the RADIUS server does not return the appropriate attribute, the authorization fails.	`userSshdPrivs {none | session| sftpRead | sftpReadWrite}`
`vSwitch`	The vSwitch associated with this user entry. You can specify a configured vSwitch or a vSwitch that you have not yet configured. If using an AAA server for authorization, the AAA server configuration for the user can overwrite this setting. If you want to ensure that only the AAA server provides this value, do not set a value for this argument. If the AAA server does not return the appropriate attribute, the authorization fails.	`vSwitchName` *`text`*

Field name	Description	Filter name
`Admin State`	Indicates whether the user entry is available for authentication or authorization purposes. If set to `enabled`, the system considers this entry when attempting to authenticate or authorize a user. If set to `disabled`, the system does not use this entry during authentication or authorization.	`adminState {enabled | disabled}`

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → user

show user detail

Purpose

Displays a detailed list of the user entries configured on the system. The system displays the output vertically.

Access mode

user

Syntax

```
show switchServices userAdministration user detail
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show user detail
User Name:             .default
Priority:              1
User Password:
Console Login:         match
Sshd Session:          match
Sshd Login:            match
Telnet Login:          match
Http Login:            match
Https Login:           match
Xml Access:            match
Authentication Method: tacacs
Authorization Method:  alwaysAccept
Profile Name:          vSwitchAdmin
Sshd Privileges:       session
vSwitch:               surprise!
Admin State:           enabled
```

Output description

Field name	Description	Filter name
`User Name`	The textual name that identifies the user entry. You use this value when the system prompts you for a user name during the log in process.	`userName` *`text`*
`Priority`	The priority of the user entry. If a user has multiple entries, the system uses the highest-priority entry that: • Has a user name that matches the user name in the authentication request. • Allows authentication when you use a specific application to connect to the system. The N2000 Series then applies the authentication method defined in the user entry. The following conditions apply: • If the authentication and authorization succeeds, you can log in to the system and monitor or configure the system, based on the profile and vSwitch associated with the user entry. • If the authentication fails (the system cannot find an entry that matches a user's credentials), the login process ends. • If the defined authentication service for the user entry is not available, the system looks for a lower-priority user entry that matches the user name and application setting.	`priority` *`integer`*
`User Password`	The password associated with the user entry. The system uses this value only if you use the internal database for authentication.	`password` *`text`*

Field name	Description	Filter name
`Console Login`	Indicates whether the system can use this entry for authentication when a user attempts to connect to the system using a console connection. If set to `match`, the system can use this entry for console login requests.If set to `ignore`, the system does not use this entry.	`consoleLogin {match | ignore}`
`Sshd Session`	Indicates whether the system can use this entry when an SSH session attempts to establish a connection with the N2000 Series for all SSH access. If set to `match`, the system can use this entry for SSHd session requests. If set to `ignore`, the system does not use this entry.	`sshdSession {match | ignore}`
`Sshd Login`	Indicates whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using SSH. If set to `match`, the system can use this entry for SSH login requests. If set to `ignore`, the system does not use this entry.	`sshdLogin {match | ignore}`
`Telnet Login`	Indicates whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using Telnet. If set to `match`, the system can use this entry for Telnet login requests. If set to `ignore`, the system does not use this entry.	`telnetLogin {match | ignore}`

Field name	Description	Filter name
`Http Login`	Indicates whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using HTTP. If set to `match`, the system can use this entry for HTTP login requests. If set to `ignore`, the system does not use this entry.	`httpLogin {match | ignore}`
`Https Login`	Indicates whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using HTTPS. If set to `match`, the system can use this entry for HTTPS login requests. If set to `ignore`, the system does not use this entry.	`httpsLogin {match | ignore}`
`Xml Access`	Indicates whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using an XML application. If set to `match`, the system can use this entry for XML login requests. If set to `ignore`, the system does not use this entry.	`xmlAccess {match | ignore}`

Field name	Description	Filter name
`Authentication Method`	The type of authentication configured for this user entry. The possible values are: `alwaysAccept`: The system authenticates the user no matter what password the user enters. `alwaysReject`: The system never allows the user access to the system. `internalUserTable`: The system uses the user names and passwords stored in its internal database to authenticate the user, when it uses this entry. `tacacs`: The system uses a remote TACACS+ server to authenticate the user. `radius`: The system uses a remote RADIUS server to authenticate the user.	`authenticationMethod {alwaysAccept | alwaysReject | internalUserTable |tacacs | radius}`

Field name	Description	Filter name
`Authorization Method`	The type of authorization configured for this user entry. The possible values are: `alwaysAccept`: The system allows the user to use all supported services. `alwaysReject`: The system does not allow the user to use any services. `tacacs`: The system uses a remote TACACS+ server to obtain the authorization settings for the user. The TACACS+ server can overwrite user entry settings, for example, access to a specific vSwitch or privileges set by profiles. `radius`: The system uses a remote RADIUS server to obtain the authorization settings for user. The RADIUS server can overwrite user entry settings similar to the server above.	`authorizationMethod {alwaysAccept | alwaysReject | tacacs | radius}`

Field name	Description	Filter name
`Profile Name`	The profile associated with this user entry. The possible values are: `systemAdmin`: Provides read-write access to the system level commands and the system vSwitch. `systemOperator`: Provides read-only access to the system level commands and the system vSwitch. `vSwitchAdmin`: Provides read-write access to the commands for a specific vSwitch only. `vSwitchOperator`: Provides read-only access to the commands for a specific vSwitch only. `unspecified`: Specifies an invalid profile. A security server must return the appropriate attribute for the user.	`profileName {systemAdmin | systemOperator | vSwitchAdmin | vSwitchOperator | unspecified}`

Field name	Description	Filter name
`Sshd Privileges`	The SSH privileges associated with this user entry. The possible values are: `none`: Allows no SSH access to the system or ability to use SFTP. `session`: Allows a user to establish an SSH session for logging in to management applications. `sftpRead`: Allows a user to establish an SSH session (session privileges) and use SFTP to transfer files from the switch. `sftpReadWrite`: Allows a user to establish an SSH session (session privileges) and use SFTP to transfer files from and to the switch.	`userSshdPrivs {none | session| sftpRead | sftpReadWrite}`
`vSwitch`	The vSwitch associated with this user entry.	`vSwitchName` *`text`*
`AdminState`	Indicates whether the user entry is available for authentication purposes. If `enabled`, the system considers this entry when attempting to authenticate a user. If `disabled`, the system does not use this entry during authentication.	`adminState {enabled | disabled}`

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → user → detail

show user summary

Purpose

Displays a summarized list of the user entries configured on the system. The system displays the list horizontally across the screen.

Access mode

user

Syntax

```
show switchServices userAdministration user summary
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show user summary
User Name       Priority Authentication    Authorization     Admin State
.default          1        tacacs             alwaysAccept       enabled
admin             1        alwaysAccept       alwaysAccept       enabled
davidc            1        tacacs             alwaysAccept       enabled
```

Output description

Field name	Description	Filter name
`User Name`	The textual name that identifies the user entry. The user name must be 32 or fewer characters. You use this value when the system prompts you for a user name during the log in process.	`userName` *`text`*
`Priority`	The priority of the user entry. If a user has multiple entries, the system uses the highest-priority entry that: • Has a user name that matches the user name in the authentication request. • Allows authentication when you use a specific application to connect to the system. The N2000 Series then applies the authentication method defined in the user entry. The following conditions apply: • If the authentication and authorization succeeds, you can log in to the system and monitor or configure the system, based on the profile and vSwitch associated with the user entry. • If the authentication fails (the system cannot find an entry that matches a user's credentials), the login process ends. • If the defined authentication service for the user entry is not available, the system looks for a lower-priority user entry that matches the user name and application setting.	`priority` *`integer`*

Field name	Description	Filter name
`Authentication`	The type of authentication configured for this user entry. The possible values are: `alwaysAccept`: The system authenticates the user no matter what password the user enters. `alwaysReject`: The system never allows the user access to the system. `internalUserTable`: The system uses the user names and passwords stored in its internal database to authenticate the user, when it uses this entry. `tacacs`: The system uses a remote TACACS+ server to authenticate the user. `radius`: The system uses a remote RADIUS server to authenticate the user.	`authenticationMethod {alwaysAccept | alwaysReject | internalUserTable |tacacs | radius}`

Field name	Description	Filter name
`Authorization`	The type of authorization configured for this user entry. The possible values are: `alwaysAccept`: The system allows the user to use all supported services. `alwaysReject`: The system does not allow the user to use any services. `tacacs`: The system uses a remote TACACS+ server to obtain the authorization settings for the user. The TACACS+ server can overwrite user entry settings, for example, access to a specific vSwitch or privileges set by profiles. `radius`: The system uses a remote RADIUS server to obtain the authorization settings for user. The RADIUS server can overwrite user entry settings similar to the server above.	`authorizationMethod {alwaysAccept | alwaysReject | tacacs | radius}`
`Admin State`	Indicates whether the user entry is available for authentication purposes. If `enabled`, the system considers this entry when attempting to authenticate a user. If `disabled`, the system does not use this entry during authentication.	`entryMode {enabled | disabled}`

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → user → summary

user

Purpose

Configures a user entry that defines the authentication and authorization rules that the system employs for a user trying to access the N2000 Series. You can create a maximum of 100 user entries on the system. You can associate multiple entries with a specific user name.

The system uses the priority assigned to each user entry, along with the user name and application settings, to evaluate which entry to use for authentication and authorization.

- If the system cannot find a user entry that matches a user's credentials, it will try to use the `.default` user entry, if it is available for authentication or authorization purposes.
- If the system finds a user entry that matches the user's credentials, but the service providing the authentication or authorization service is not available (for example, a TACACS+ server is unreachable), the system tries the next matching user entry, according to its priority setting, that can perform the required authentication or authorization task.

The `no` form of the command deletes a configured user entry. The `userName` and `priority` arguments are the only required arguments when using the `no` form of the command. The `.default` entry cannot be permanently deleted.

See the *Sun N2000 Series Release 2.0 – System Administration Guide* for additional details about creating user entries.

Access mode

enable

Syntax

```
switchServices userAdministration user
   userName text
   priority integer
   [password text]
   [consoleLogin {match | ignore}]
   [sshdSession {match | ignore}]
   [sshdLogin {match | ignore}]
   [telnetLogin {match | ignore}]
   [httpLogin {match | ignore}]
   [httpsLogin {match | ignore}]
   [xmlAccess {match | ignore}]
   [authenticationMethod {alwaysAccept | alwaysReject |
      internalUserTable | tacacs | radius}]
   [authorizationMethod {alwaysAccept | alwaysReject | tacacs |
      radius}]
   [profileName {systemAdmin | systemOperator | vSwitchAdmin |
      vSwitchOperator}]
   [userSshdPrivs {none | session | sftpRead | sftpReadWrite}]
   [vSwitchName text]
   [adminState {enabled | disabled}]
```

Arguments

Argument name	Description
`userName` *`text`*	Specifies the textual name that identifies the user entry. The user name must be 32 or fewer characters. You use this value when the system prompts you for a user name during the log in process. **Note:** Do not use the following special characters in the user name: • braces { } • parentheses ( ) • double quotes “ • apostrophes ‘ Although the system allows you to enter these characters, it does not allow the user to log in.

Argument name	Description
`priority` *integer*	Assigns a priority to the user entry. If a user has multiple entries, the system uses the highest-priority entry that: • Has a user name that matches the user name in the authentication request. • Allows authentication when you use a specific application to connect to the system. The N2000 Series then applies the authentication method defined in the user entry. The following conditions apply: • If the authentication and authorization succeeds, you can log in to the system and monitor or configure the system, based on the profile and vSwitch associated with the user entry. • If the authentication fails (the system cannot find an entry that matches a user's credentials), the login process ends. • If the defined authentication service for the user entry is not available (for example, the system is unable to access a TACACS+ server), the system looks for a lower-priority user entry that matches the user name and application setting.
`password` *text*	**Optional**. Specifies a password associated with the user entry. The password must be 32 or fewer characters. Configure this entry only if using the internal user database for authentication. In this case, this is the password you enter when the system prompts you to do so. **Note:** Do not use the following special characters in the user name: • braces { } • parentheses ( ) • double quotes “ • apostrophes ‘ Although the system allows you to enter these characters, it does not allow the user to log in.
`consoleLogin {match | ignore}`	**Optional**. Specifies whether the system can use this entry for authentication when a user attempts to connect to the system using a console connection. If the value is `match`, the system can use the entry; if the value is `ignore`, the system does not use the entry. The default setting is `match`.

Argument name	Description
`sshdSession {match | ignore}`	**Optional**. Specifies whether the system can use this entry when an SSH session attempts to establish a connection with the N2000 Series. If the value is `match`, the system can use the entry; if the value is `ignore`, the system does not use the entry. The default setting is `match`.
`sshdLogin {match | ignore}`	**Optional**. Specifies whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using SSH. If the value is `match`, the system can use the entry; if the value is `ignore`, the system does not use the entry. The default setting is `match`.
`telnetLogin {match | ignore}`	**Optional**. Specifies whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using Telnet. If the value is `match`, the system can use the entry; if the value is `ignore`, the system does not use the entry. The default setting is `match`.
`httpLogin {match | ignore}`	**Optional**. Specifies whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using HTTP. If the value is `match`, the system can use the entry; if the value is `ignore`, the system does not use the entry. The default setting is `match`.
`httpsLogin {match | ignore}`	**Optional**. Specifies whether the system can use this entry for authentication when a user attempts to log in to the N2000 Series using HTTPS. If the value is `match`, the system can use the entry; if the value is `ignore`, the system does not use the entry. The default setting is `match`.
`xmlAccess {match | ignore}`	**Optional**. Specifies whether XML applications can retrieve information from the system in an XML format. If the value is `match`, XML applications can access the system and request information. The system returns information as an XML document. If the value is `ignore`, the system does not use the entry. The default setting is `match`.

Argument name	Description
`authenticationMethod {alwaysAccept | alwaysReject | internalUserTable |tacacs | radius}`	**Optional**. Configures the type of authentication to use for this user entry. The valid values are: `alwaysAccept`: The system authenticates the user no matter what type of password the user enters. `alwaysReject`: The system never allows the user access to the system. `internalUserTable`: The system uses the user names and passwords stored in its internal database to authenticate the user, when it uses this entry. `tacacs`: The system uses a remote TACACS+ server to authenticate the user. `radius`: The system uses a remote RADIUS server to authenticate the user. This default setting is `tacacs`.
`authorizationMethod {alwaysAccept | alwaysReject | tacacs | radius}`	**Optional**. Configures the type of authorization to use for this user entry. The valid values are: `alwaysAccept`: The system allows the user to use all supported services (for example, Telnet, HTTP, SSH, or console access). `alwaysReject`: The system does not allow the user to use any services. `tacacs`: The system uses a remote TACACS+ server to obtain the authorization settings for the user. The TACACS+ server can overwrite user entry settings, for example access to a specific vSwitch or privileges that profiles allow. `radius`: The system uses a remote RADIUS server to obtain the authorization settings for user. The RADIUS server can overwrite user entry settings, for example access to a specific vSwitch or privileges that profiles allow. This default setting is `alwaysAccept`.

Argument name	Description
`profileName{systemAdmin | systemOperator | vSwitchAdmin | vSwitchOperator | unspecified}`	**Optional**. Specifies the profile associated with this user entry. The profile name must be 32 characters or fewer. The valid values are: `systemAdmin`: Provides read-write access to the system level commands and the system vSwitch. `systemOperator`: Provides read-only access to the system level commands and the system vSwitch. `vSwitchAdmin`: Provides read-write access to the commands for a specific vSwitch only. `vSwitchOperator`: Provides read-only access to the commands for a specific vSwitch only. `unspecified`: Specifies an invalid profile. Use this profile if you want to ensure that only the TACACS+ server or the RADIUS server provides this value. If the TACACS+ server or the RADIUS server does not return the appropriate attribute, the authentication fails. The default is `vSwitchAdmin`.

Argument name	Description
`userSshdPrivs {none | session| sftpRead | sftpReadWrite}`	**Optional**. Configures the SSH privileges associated with the user entry. The valid values are: `none`: Allows no SSH access to the system or ability to use SFTP. `session`: Allows a user to establish an SSH session. `sftpRead`: Allows a user to establish an SSH session (session privileges) and use SFTP to transfer files from the switch. `sftpReadWrite`: Allows a user to establish an SSH session (session privileges) and use SFTP to transfer files from and to the switch. The default setting for new user entries is `session`. **Note:** If using a TACACS+ server or RADIUS server for authorization, the TACACS+ configuration or RADIUS configuration for the user can overwrite this setting. If you want to ensure that only the TACACS+ server or the RADIUS server provides this value, set this argument to `none`. If the TACACS+ server or RADIUS server does not return the appropriate attribute, the authorization fails. The system uses the settings for this argument if it uses this entry for authentication *or* if you configure SSH on the system to use public keys for authentication. If you configure the system to use public key authentication for SSH, the system uses only the highest-priority entry that has the `sshdSessionMode` argument set to `considerEntry`. If you configure SSH to use no authentication, the system assigns session privileges to the SSH session and does not use the settings from this argument.

Argument name	Description
vSwitchName *text*	**Optional.** Specifies the vSwitch associated with this user entry. You can specify a configured vSwitch or a vSwitch that you have not yet configured. **Note:** If using a TACACS+ server or RADIUS server for authorization, the TACACS+ configuration or RADIUS configuration for the user can overwrite this setting. If you want to ensure that only the TACACS+ server or the RADIUS server provides this value, do not set a value for this argument. If the TACACS+ server or RADIUS server does not return the appropriate attribute, the authorization fails.
adminState {enabled | disabled}	**Optional**. Specifies whether the user entry is available for authentication purposes. If set to `enabled`, the system considers this entry when attempting to authenticate a user. If set to `disabled`, the system does not use this entry during authentication. The default setting is `disabled`.

Delete filters

See the `show user` command for additional argument descriptions.

```
no switchServices userAdministration user
   userName text
   priority integer
   [password text]
   [consoleLogin {match | ignore}]
   [sshdSession {match | ignore}]
   [sshdLogin {match | ignore}]
   [telnetLogin {match | ignore}]
   [httpLogin {match | ignore}]
   [httpsLogin {match | ignore}]
   [xmlAccess {match | ignore|]
   [authenticationMethod {alwaysAccept | alwaysReject |
      internalUserTable | tacacs | radius}]
   [authorizationMethod {alwaysAccept | alwaysReject | tacacs |
      radius}]
   [profileName {systemAdmin | systemOperator | vSwitchAdmin |
      vSwitchOperator}]
   [userSshdPrivs {none | session | sftpRead | sftpReadWrite}]
   [vSwitchName text]
   [adminState {enabled | disabled}]
   [matchedServices text]
   [ignoredServices text]
```

Example

The following example shows how to create a user called `user1` who can access the system using the console, Telnet, or HTTP. The authentication and authorization method is `tacacs` and the user has read and write access to the system vSwitch. The `entryMode` is set to `match` indicating that the system can use this entry for authentication and authorization purposes.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# user userName user1 password abc123
consoleLogin match telnetLogin match httpLogin match xmlAccess ignore
authenticationMethod tacacs authorizationMethod tacacs profileType systemAdmin
vSwitchName system adminMode enabled
```

Note: When you specify a password with the `user` command, you see the value in clear text; however, the system encrypts the password using the `keyInfo` specified with the `userAdministration` command. The system *does not* store the password in clear text.

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → user → add
- switchServices → userAdministration → user → copy
- switchServices → userAdministration → user → modify
- switchServices → userAdministration → user → delete

userAdministration (root)

Purpose

Configures or modifies system-wide sensitive data encryption and accounting attributes for authentication, authorization, and accounting (AAA) operations. Also enters the `userAdministration` command mode.

Access mode

enable

Syntax

```
switchServices userAdministration
   [keyinfo text]
   [accountingTypes {none | starts | startsAndStops |
      startsAndStopsandUpdates}]
   [accountingMethod {none | internal | tacacs | internalAndTacacs
      | radius | internalAndRadius | tacacsAndRadius |
      internalAndTacacsAnd Radius}]
   [accountingLogSize integer]
   [accountingUpdateInterval integer]
   [serverRecoveryTimer integer]
```

Arguments

Argument name	Description
`keyInfo text`	**Optional.** Enters random text that the system can use to generate a key for encryption of sensitive data (for example, passwords). You cannot create a password or secret key without setting this value first. The key text must be 255 characters or fewer. When you enter this value, the system displays it in clear text. However, when using the `show userAdministration` command, the system displays the key text as hyphens (--).

Argument name	Description
`accountingTypes {none | starts | startsAndStops | startsAndStopsandUpdates}`	**Optional.** Specifies the type of accounting records the system creates. Valid values are: `none`: The system does not create any accounting records. `starts`: The system creates a record indicating when a session began. `startsAndStops`: The system creates a record indicating when a session began and when it ended. `startsAndStopsandUpdates`: The system creates records indicating when a session began, whether a session is still active after a set interval elapses, and when the session ended. The default setting is `startsAndStopsandUpdates`.

Argument name	Description
`accountingMethod {none | internal | tacacs | internalAndTacacs radius | internalAndRadius | tacacsAndRadius | internalAndTacacsAndRadius}`	**Optional.** Specifies where the system saves accounting records. Valid values are: `none`: The system does not save any accounting records. `internal`: The system saves accounting records in its internal log. `tacacs`: The system sends accounting records to a TACACS+ server. The `userAdministration server tacacs` command configures the TACACS+ servers that the N2000 Series uses. `internalAndTacacs`: The system sends accounting records to a TACACS+ server *and* saves the records in the internal log. `radius`: The system sends accounting records to a RADIUS server. The `userAdministration server radius` command configures the RADIUS servers that the N2000 Series uses. `internalAndRadius`: The system sends accounting records to a RADIUS server *and* saves the records in the internal log. `tacacsAndRadius`: The system sends accounting records to a TACACS+ server and to a RADIUS server. `internalandTacacsAndRadius`: The system sends accounting records to a TACACS+ server and to a RADIUS server and saves the records in the internal log. The default setting is `internal`.
`accountingLogSize` *`integer`*	**Optional.** Specifies how many entries to save in the internal accounting log. Valid range is 10 through 200; the default value is 100. If you change this argument's value, the system clears the log of all entries and starts with an empty log. When the log reaches the specified threshold, the system overwrites existing records, starting with the oldest record.

Argument name	Description
`accountingUpdateInterval` *`integer`*	**Optional.** Sets the number of minutes that must elapse before the system checks for previously started sessions that are still active. Valid range is 0 through 1440 minutes; the default value is 10 minutes. If the specified time elapses and a session is still active, the system creates an update record for the session. You must configure the system to save update records if you want the system to generate these records. Setting this argument to 0 turns this feature off and the system does not generate update records.
`serverRecoveryTimer` *`integer`*	**Optional.** Sets a background timer that the system uses when checking the availability status of a TACACS+ server or a RADIUS server. Valid range is 60 through 600 seconds; the default value is 90 seconds. When the time set for this argument elapses, the system sets the status of previously non-responding servers to `unknown`. This allows the system to try to access the server for a future AAA request.

Example

The following example shows how to configure the system to do the following:

- Encrypt passwords using random text that you supply.
- Generate records for session start, updates, and stops.
- Save all accounting records on a TACACS+ server.
- Create update records every 15 minutes (if necessary).
- Set the status of previously unresponsive TACACS+ servers to `unknown` every 60 seconds.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration keyinfo afadfdkfma;ljf
accountingType startsAndStopsAndUpdates accountingMethod tacacs
accountingUpdateInterval 15 serverRecoverTimer 60
```

Associated MIB

`authentication.mib`

Web path

- switchServices → userAdministration → modify

Chapter 14. SSH Commands

Secure Shell Protocol description

The N2000 Series systems implement Secure Shell (SSH) Server Version 2 for secure client/server communication. SSH provides secure remote logins and file transfers using encryption and public key authentication. As part of establishing a secure connection, SSH uses a key pair retrieved from the Certificate and Key Manager (CKM) utility.

Once you establish an SSH session, you can transfer files with Secure Shell File Transfer Protocol (SFTP). SFTP provides more secure transfers than FTP, and uses a simple interface similar to FTP or Microsoft Windows environments. SSH includes counters that detail the SFTP activity taking place over a connection. You can view these counters, and the session type, with the `show sessions` command.

Understanding N2000 Series SSH operations

When using SSH on the N2000 Series, the SSH session is transparent. That is, while you are within an active session, the command-line interface (CLI) appears just as it would if you were connecting directly through the console or through Telnet. However, SSH does not support the full range of optional, user-configurable attributes that an SSH workstation would support. For example, the N2000 Series implementation does not support `setstat` and `setfstat`. If you try to change an attribute that the N2000 Series does not support, you will receive a warning noting that the setting failed.

SSH commands for the N2000 Series Switch

The commands described in this chapter set or modify the SSH server (`sshd` daemon) parameters. Commands include the following:

- *configuration* commands — Set the minimum requirements that the client must meet for the session to be established.
- *client key* commands — Enter the data that the server expects to see for each identified client. Use these commands if you are implementing public keys for user authentication.
- *show* commands — Display the configuration tables and the operational tables for the SSH server, client keys, and sessions.

Many of the commands display internal system timers. The timers report the time at which an event occurred, which is displayed in seconds. The seconds clock restarts at each boot, and the *time stamp* shows how many seconds have passed between the boot and the occurrence. For example, the time stamp of the last client key configuration change records the time at which the last `clientKey` command was executed.

In addition, the N2000 Series uses the following tables:

- *Configuration tables* — Store the configuration settings, not necessarily the active settings.
- *Operational tables* — Show the running parameters.

You can set up accounts before you need them. Configure the client but set the client's administrative state to `disabled`. The client's settings are displayed in the configuration table, but not in the operational table.

SSH command path

The commands in this chapter are executed from within the SSH command mode, which is a submode of the switchServices command mode.

SSH command summary

Table 14-1 lists and briefly describes the SSH commands.

Table 14-1. SSH command summary

Command name	Description
sshd (root)	Modify the basic SSH configuration parameters.
advanced	Modify advanced SSH options.
clientKey	Create or modifies the clientKey table.
sessions	Modify SSH session objects.
show	Display the SSH configuration parameters.
show advanced	Display advanced parameters of the SSH configuration.
show advanced testPatchInfo	Display information specific to patches applied to a connection.
show algorithms	Display supported and operational connection algorithms.
show clientKey	Display the clientKey table (configured elements).
show clientKeyStatus	Display the clientKeyStatus table (operational elements).
show sessions	Display normal SSH session objects.
show sessions advanced	Display advanced SSH session objects.
show sessions advanced negotiations	Display SSH negotiation parameters.

SSH basic configuration

Follow the steps below to configure a basic SSH setup on the N2000 Series:

Table 14-2. Steps for configuring SSH on the N2000 Series

Step	Action
1.	Generate a SSH server key using the N2000 Series CKM utility. Be certain to change the algorithm type to DSA when creating the key for SSH. Use the CKM `generate` command.
2.	Use the `serverKeyId` argument of the `sshd (root)` command to configure the key ID (used for CKM key generation) into SSH.
3.	Set the `adminState` to `enabled` with the `sshd (root)` command.
4.	**Optional.** Configure additional user profiles under userAdministration for SSH use. See Chapter 13, “User administration commands.”
5.	**Optional.** Configure SSHD connection criteria (authentication methods and algorithm exchanges), as well as session characteristics. Use the `sshd (root)` command. See also Chapter 13, “User administration commands” to use the key generated in Step 1.
6.	**Optional.** Verify the user authentication settings. See the `userAdministration show user` command in Chapter 13, “User administration commands.”.

advanced

Purpose

Configures additional parameters of SSH on the N2000 Series, allowing you to fine-tune the internal parameters. These settings are global and apply to all sessions. All of the parameter settings are optional. The connection will operate with default settings if you do not modify them. Typically, these parameters do not need to be modified.

Use this command to set the global patch information for SSH vendor patches. There are two methods for applying patches required by SSH client applications: you can configure patches manually, or you can set the system to apply known patches automatically. The system first checks for manually configured patches. If none are found, it checks the internal patch database for known fixes. If you have manually repaired client connection problems for your SSH vendor, you may want to turn off the auto-patch capability for that vendor to prevent the system from applying patches. Consult your SSH client documentation to determine whether you require patches.

The following table lists the patch information used to repair client connection applications. Entries that are shaded do not apply to the N2000 Series.

Table 14-3. Patch names and bit values

Patch identifier	Display name	Bit value
SSH_BUG_SIGBLOB	SigBlob-fix	1
SSH_BUG_PKSERVICE	PksSrvc	2
SSH_BUG_HMAC	HMAC-fix	4
SSH_BUG_X11FWD	x11fwd	8
SSH_OLD_SESSIONID	OldSessionId	16
SSH_BUG_PKAUTH	pkAuth	32
SSH_BUG_DEBUG	DebugMsg-fix	64
SSH_BUG_BANNER	Banner	128
SSH_BUG_IGNOREMSG	IgnMsg	256
SSH_BUG_PKOK	Pkok	512
SSH_BUG_PASSWORDPAD	PswdPad	1024
SSH_BUG_SCANNER	Scnr	2048

Table 14-3. Patch names and bit values (continued)

Patch identifier	Display name	Bit value
SSH_BUG_BIGENDIANAES	BigEnd	4096
SSH_BUG_RSASIGMD5	RSASigMd5	8192
SSH_OLD_DHGEX	OldDHGex	16384
SSH_BUG_NOREKEY	NoReky	32768
SSH_BUG_HBSERVICE	HBSrvc	65535
SSH_BUG_OPENFAILURE	OpnFail	131072
SSH_BUG_DERIVEKEY	DeriveKey-fix	262144
SSH_BUG_DUMMYCHAN	DmyCh	1048576
SSH_VDFX_NOUSERAUTH	SecureFX-NoUserAuth	2097152

Access mode

enable

Syntax

```
switchServices sshd advanced
   [sshdPort integer]
   [maxAuthenticationTime integer]
   [globalEventInterval integer]
   [handleErrors {enable | disable}]
   [executionMonitorMode {active | passive}]
   [clientKeepAliveInterval integer]
   [patchVendorIds {none | sshcommunications | fsecure |
      sshCommFsecure | vandyke | sshCommVandyke | secureVandyke |
      sshcomFsecureVandyke]
   [testPatchVersion text]
   [testPatchValue integer]
   [cliInheritsSshdLoginCredentials {enable | disable}]
```

Arguments

Argument name	Description
`sshdPort` *`integer`*	**Optional**. Assigns the port the system uses for SSH sessions. Valid port numbers are 0 through 65536; the default port is 22.
`maximumAuthenticationTime` *`integer`*	**Optional**. Sets the maximum number of seconds allowed for user authentication to complete. If the timer expires, the system cancels the request. Valid range is 0 through 300 seconds; the default is 30 seconds. Entering 0 disables the timeout timer.
`globalEventInterval` *`integer`*	**Optional**. Sets the number of seconds allowed between similar SSH global events. This allows you to limit the frequency with which the system reports similar events to avoid flooding the network. Valid range is 0 through 600; the default setting is 10 seconds.
`handleErrors {enable | disable}`	**Optional**. Sets the application's ability to handle system errors when possible. This capability is `enabled` by default.
`executionMonitorMode {active | passive}`	**Optional**. Sets the system to monitor its internal processes. When `active`, if loops aren't executed within a specific period of time, the systems shuts down and then restarts the application. If set to `passive`, you lose the ability for the system to automatically terminate sessions that have problem-causing characteristics. The default setting is `active`.
`clientKeepAliveInterval` *`integer`*	**Optional**. Sets the number of seconds between "keep alive" messages sent to the client. If the client does not respond after three keep alives, the system destroys the session. Valid range is 0 through 3600; the default setting is 0. Keep alive messages are not universally supported across client applications. When used, the system sends a channel request to an invalid service. This function is just like a keep alive, as an error response indicates an active client. If the client application doesn't support error response properly, disable this feature by setting the `clientKeepAliveInterval` to `0`.

Argument name	Description
`patchVendorIds` *`text`*	**Optional**. Specifies the name(s) of the SSH vendor. If that vendor has known problem clients, the system automatically applies patches for known issues. The most common SSH vendors are supported—SSH Communication, FSecure, and VanDyke. Possible settings either disable auto patching or select a "package" of vendors: `none`: disables the auto-patch feature. `sshcommunications`: applies SSH Communication patches. `fsecure`: applies FSecure patches. `sshCommFsecure`: applies SSH Communication and FSecure patches. `vandyke`: applies VanDyke patches. `sshcommVandyke`: applies SSH Communication and VanDyke patches. `fsecureVandyke`: applies FSecure and VanDyke patches. `sshcomFsecureVandyke`: applies SSH Communication, FSecure, and VanDyke patches. The default setting is `sshcomFsecureVandyke`. The setting is displayed with the `show sshd advanced` command.

Argument name	Description
`testPatchVersion` *text*	**Optional**. Specifies the version string of the client requiring the patch. Enter the exact string of the client's version. To determine the version information, either: 1. See the event log. The string is found in the `SSHD_EVENT_PatchInfo` event. Be sure that you have the event level set to DEBUG to view this output. 2. Use a sniffer for line monitoring. This setting works in conjunction with the `testPatchValue` argument, below, to identify vendor patches. The version string in use is displayed with the `show switchServices sshd advanced testPatchInfo` command. Use the `show switchServices sshd sessions advanced negotiations` command, in the `Client Version` field (which cannot be seen until after a successful session connection) to view the active session info.
`testPatchValue` *text*	**Optional**. Specifies the number representing the bit values of the patch. See the table, “Patch Names and Bit Values” for bit values. To add multiple patches, enter the sum of the patch IDs you want to apply, as shown in the example below. To determine which patch bit values to set, read the N2000 Series debug events and the client events. The information may point to which bit values will resolve well known problems. The setting is displayed with the `show switchServices sshd advanced testPatchInfo` command.
`cliInheritsSshdLoginCredentials {enable | disable}`	**Optional**. Enables or disables automatic CLI login if password authentication was used. If enabled, the system automatically passes the CLI the username and password that were used to validate the user at the SSH level. The default is `disabled`.

Example

The following example manually configures the system to apply patches for well-known problems `SigBlob` and `DebugMsg`. `SigBlob-fix` has bit value of 1 and `DebugMsg-fix` a bit value of 64. To apply both patches, enter the sum of their values. Use the `show advanced testPatchInfo` command to verify values.

```
sun> enable
sun# switchServices
sun(switchservices)# sshd
sun(switchServices sshd)# advanced testpatchversion companyAbc
testPatchValue 65

sun(switchServices sshd)# show advanced testPatchInfo
confTestPatchVersion: companyAbc
confTestPatchValue:   65
operTestPatchVersion: companyAbc
operTestPatchValue:   65
operTestPatchSummary: SigBlob-fix,DebugMsg-fix
operTestPatchBitVal:  65
```

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → advanced → modify

clientKey

Purpose

Enters the key data for the specified client into the server's database. After identifying the client by IP address and user name, the system prompts you for the client's public digital signature algorithm (DSA) key. If the IP address and user name are unknown to the server, this command creates a new client key. If these parameters already exist in the server's database, this command modifies the client key.

To identify the user in the database, you enter an IP address and user name. The IP address can be either the specific client address, or it can be a wildcard address of 0.0.0.0. The system first checks for a match with the specific address, but failing that, will match the user name to the less-specific address if it is configured.

Use the optional `keyStatus` argument to configure a client and its data key but prevent the account from becoming active. The data is then stored in the configuration table, but not the operational table. To make the client operational, execute the command supplying the IP address and host name, and set `keyStatus` to `active`. Because the key data is already configured, the system does not prompt for it.

Optionally, if the `keyStatus` is set to `notInService`, you can supply any text in the `keyData` field (to be used, for example, as a placeholder). However, the system requires specific kinds of key data for the client to become operational. If you later change the status from `notInService` to `active`, but do not change the key data, the system does not make the client key operational. To change the key to valid data, reexecute the command supplying the IP address and host name.

You can view the configured keys using the `show clientKey` command; view the operational keys using the `show clientKeyStatus` command.

Use the `no` form of the command to remove an entry from the ClientKey operational table, which stores the user name, IP address, and status. Any entry in the table that matches the IP address, user name, and status specified by this command is deleted. A wildcard, which matches all entries for the specific argument, is accepted for any or all arguments.

Access mode

enable

Syntax

To create a client key:

```
switchServices sshd clientKey
   hostAddress ipAddress
   userName text
   [keyStatus {active | notInService}]
   keyData text
```

To modify an existing client key:

```
switchServices sshd clientKey
   hostAddress ipAddress
   userName text
   [keyStatus {active | notInService}]
   [keyData text]
```

Arguments

Argument name	Description
`hostAddress` *ipAddress*	Identifies the IP address of the host. The IP address can be either the specific client address, or it can be a wildcard address of 0.0.0.0.
`userName` *text*	The user name that will be attempting public key authentication. The user name can be a maximum of 64 alphanumeric characters.
`keyStatus {active | notInService}`	**Optional.** Sets the administrative status of the client key, either active or not in service. The default is `active`.
`keyData` *text*	Enters the text that comprises the public key for the host (client) identified by IP address and user name. The key data must be no more than 2048 bytes.

Delete filters

```
no switchServices sshd clientKey
   hostAddress ipAddress
   userName text
   [keyStatus {active | notInService}]
   [keyData text]
```

Example

The following example sets up a client key for JohnDoe at IP address 1.1.1.1. Until the command is reexecuted to set the key to active, JohnDoe does not have access, using this key, to the server. In addition, JohnDoe's key data is a reminder to make his account active the following week.

```
sun> enable
sun# switchServices
sun(switchservices)# sshd
sun(switchservices sshd)# clientKey 1.1.1.1 JohnDoe notInService
Please enter your data ctrl-z to accept ctrl-c to cancel:

Activate June 1

sun(switchServices sshd)# show clientKey
Host Address: 1.1.1.1
User Name: JohnDoe
Key Status: notInService
Key Data: Activate June 1

sun# switchServices sshd clientKey 1.1.1.1 JohnDoe active

Please enter your data ctrl-z to accept ctrl-c to cancel:

-----BEGIN SSH2 PUBLIC KEY-----
Comment: 1024-bit DSA, converted from OpenSSH by Administrator@DORTIZ
MIIBuwIBAAKBgQDdJKhnTqMg0QjI/RhQJIwtsVxO70JENbG
djKm3ZLFAXJxR3uvVZ+Zd+EoZU6fbAPGtsNUIemNo1IHT0q4ibFJElWFmL8TVEvT9bkefA
LUuB9e2KEXB /
dRbKY4vokvXsKz1Aa7g3AhpZMpPcu9rKPzBHJciptfR3Th45nHNuzBaPwIVAMbg
N3htF2qWy8 tEJq/ZnVlkPRynAoGABzFcJkjo82QVPMDmo+nBXKrlBJSxSEJI8Kfc
8+n3qxNTTaNOzKX4kiXcMnFz cyCpizNHCwNlZwLVjhcc1SkJ99CtwxauEUdF5cIH E/
0vnvS3nrDfeJ/2TeLr2UW89HDuxFd25xUdLT BA4UtegbXAZFAtAwsv9ObyCAc3
OOF0ZWwCgYEAgaCLXaDSG+//0shdgPOmPLLoByUce3+kr34CIbbc UKyZoYFifhBc
RuoMdstn9cg1JQUB3DQbsqHNxg4XcJeqGHrAJeb7TSg91IgujRcYYFP7cYVQLsIR
```

The following example lists the client keys registered to host 1.1.1.1 and then deletes the key that matches user name JohnDoe, which is not in service.

```
sun(switchServices sshd)# show clientKey 1.1.1.1
Host Address: 1.1.1.1
User Name:    JohnDoe
Key Status:   notInService
Key Data:     Activate Monday
sun# no switchServices sshd clientKey hostAddress 1.1.1.1 userName
JohnDoe keyStatus notInService
sun# show switchServices sshd clientKey 1.1.1.1
sun# This table is currently empty.
```

The following example deletes information based on key data.

```
sun(switchServices sshd)# show clientKey

Host Address: 1.1.1.1
User Name:    JohnDoe
Key Status:   active
Key Data:     /maybe

Host Address: 2.2.2.2
User Name:    JaneDoe
Key Status:   active
Key Data:     ok-ok

Host Address: 3.3.3.3
User Name:    JimDoe
Key Status:   active
Key Data:     /noway

Host Address: 4.4.4.4
User Name:    JenDoe
Key Status:   active
Key Data:     /okfine


sun(switchServices sshd)# no clientKey * * keydata "/*"
3 entries were deleted.
```

Associated MIB

sshd.mib

Web path

- switchServices → sshd → clientKey → modify

sessions

Purpose

Provides a mechanism to terminate a client session by specifying its unique IP address and TCP port combination.

The system creates a row in the sessions table each time an SSH client connects (or attempts to connect) to the SSHD server in the N2000 Series. Ultimately, all active SSH connections are recorded in the sessions table, which tracks and describes several of the session characteristics. In addition to IP address and TCP port, these include session type (CLI or SFTP), client user name, packet and byte counts, negotiation summary, type of service requested, and authentication method. You can view these parameters with the `show sessions` command.

Most of the sessions table is read-only information. The `show sessions` and `show sessions advanced` commands allow you to display the output described above. In addition, you can view which sessions are currently active and, if desired, end specified sessions. This might be necessary if you want to remove an unwanted client or free up a session for another client. (The number of allowable sessions can also be changed with the `sshd (root)` command.)

Note: When using the Web interface, this command is not visible unless you have at least one active SSH session.

Access mode

enable

Syntax

```
switchServices sshd sessions
   clientIP ipAddress
   clientPort integer
   [sesStatus {active | destroy}]
```

Output description

Argument name	Description
`clientIP` *`ipAddress`*	Specifies IP address of the client that is connected in an SSH session to the N2000 Series.
`clientPort` *`text`*	Specifies the TCP port for the client's connection.
`sesStatus {active | destroy}`	**Optional.** Effects the session identified by the IP address and TCP port specified. Using the `destroy` keyword ends the session. Sessions are active by default; the `active` keyword is not accepted.

Example

The following example first displays one active client and then destroys the client uniquely identified by IP address 192.168.209.31 and TCP port 1528. See the `show sessions` command for descriptions of the various session parameters displayed.

```
sun> enable
sun# switchServices
sun(switchservices)# sshd
sun(switchServices sshd)# show sessions
Client IP:     192.168.209.31
Client Port:   1528
Session State: up
Session Type:  cli
UserName:      davidc
User Privs:    sftpWrite
UserProfile:   pro-davidc(4-SRW)
TX bytes:      1110
RX bytes:      2956
TX Cli bytes:  22
RX Cli bytes:  1
TX Sftp bytes: 0
RX Sftp bytes: 0
Neg Info:      clt->svr: 3des-cbc hmac-md5 none; svr->clt: 3des-cbc
hmac-md5
none; Kex: diffie-hellman-group1-sha1; KeyAlgorithms/Types: ssh-dss;
auth-method: password;
Session Time:  799
Idle Time:     29
SessionStatus: active
PatchIds:      0
PatchSummary:  noPatchesApplied
sun(switchServices sshd)# sessions clientIp 192.168.209.31 clientPort
1528 sessionStatus destroy
```

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → sessions → modify

show

Purpose

Displays the operational characteristics of the SSH connection—the current, running system information. The output includes parameters set with the `sshd (root)` command and counters maintained by the system. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices sshd
```

Sample output

```
sun> enable
sun# switchservices
sun(switchServices)# sshd
sun(switchServices sshd)# show
adminState:                    enabled
operState:                     up
serverKeyId:                  sshDsaKey
serverKeyState:                validKey
Idle Timeout:                  600
Authentication Banner:
Encryption Algorithms:         3des-cbc,blowfish-cbc,des
Hmac Algorithms:
hmac-md5,hmac-sha1,hmac-md5-96,hmac-sha1-96
User Authentication Methods: publickey,password
Host Authentication List:      none
Max Sessions:                  4
Current Sessions:              1
Total  Sessions:               4
TCP Port:                      22
Rejected Requests Total:       0
Rejected Requests TooMany:     0
Rejected Requests BadHost:     0
Memory Errors:                 0
Server Key Errors:             0
RNG Errors:                    0
```

Output description

Field name	Description
`Admin State`	The configured status for the SSH session, either `enabled` or `disabled`.
`Oper State`	The operational state of the SSH session, either `up`, `listeningButNoSvrKey`, or `down`. The `listeningButNoSvrKey` value indicates that the application is waiting on the communication stack, but there is no valid server key.
`Server Key ID`	The name you selected when creating key data and then associated with `sshd`.
`Server Key State`	The status of the operational server key used to establish SSH sessions, either `validKey` or `invalidKey`. Possible reasons for an invalid key state include, but are not limited to: • a key created in the CKM utility with the algorithm type set to RSA (SSH requires DSA) • a mismatched serverKeyID • a non-existent ckm keypair • misconfiguration.
`Key Diag Info`	Additional information for help with key diagnostics. This field is only displayed if the key is not valid.
`Idle Timeout`	The maximum number of seconds a session can remain idle before it is terminated. If the field displays `0`, the session can remain live regardless of the length of inactivity.
`Authentication Banner`	The message to be sent by this server to each of its SSH clients during authentication.
`Encryption Algorithms`	A list of one or more configured encryption algorithms. Possible values: `des`, `3des`, `3des-cbc`, `blowfish`, `blowfish-cbc`, and `none`. A value of `none` is not recommended.
`Hmac Algorithms`	A list of one or more configured HMAC algorithms. Possible values: `hmac-md5`, `hmac-sha1`, `hmac-md5-96`, and `hmac-sha1-96`.
`User Authentication Methods`	A list of one or more accepted user authentication methods, either `publicKey`, `password`, or `none`.
`Host Authentication List`	A list of client addresses that you want to explicitly permit or deny. If the output is preceded with an exclamation point (`!`), the system denies the identified hosts.
`Max Sessions`	The maximum number of SSH sessions this server can support at one time.

Field name	Description
`Current Sessions`	The number of sessions currently open.
`Total Sessions`	The number of sessions opened on this server since it was started.
`TCP Port`	The number of the port over which SSH is currently operating.
`Rejected Requests Total`	The number of sessions rejected because of either a bad host ID or because the system had reached its session limit.
`Rejected Requests TooMany`	The number of sessions rejected because the system had reached its session limit.
`Rejected Requests BadHost`	The number of sessions rejected because of a bad host ID.
`Memory Errors`	The number of memory errors detected (unable to get memory resources).
`Server Key Errors`	The number of server key errors detected (session attempts when the key is not valid).
`RNG Errors`	The number of random number generator errors.

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd

show advanced

Purpose

Displays the configured global values (set with the `advanced` command) and the internal system timers. The timers display a time stamp indicating, in seconds since last boot, when an event occurred. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices sshd advanced
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show advanced
SSH Server Port:            22
Max Authentication Time:    30
Error Handler:              enabled
Global Event Interval:      10
Execution Monitor Mode:     active
Patch Vendors:              sshcomFsecureVandyke
Keep Alive Interval:       0
SSH Server Public Key:
AAAAB3NzaC1kc3MAAACBALoklwCzguq6E9bGchKvXceeEXCJUOUFLwbfgx09ZBxuJ0P5Cy
6ysNl0hqCRlI4RjfINP+rg48i0qiONl9zL6aZ5i5Qyrz3Zf+B2lelj7anCGStEHLJ9LA8n
Irl4Lq7lodCQfr8DJHPsLKvZ4Cpjq1XeV4lfiimbeMe3gsOKIdY9AAAAFQDYpx++mn9uYi
+VUg+iQKYnkTavsQAAAIEAuFBZ6VVcgJjPRvnEGYXxMJYERYscxs34JZpJEZD9HVLdjkwY
I5EyNJNHk20bER59JolRxdVf6im7GnZr6Q5TYQZSeDwrxAqfDLWGntM7Q210qZ7fccCl+4
9C7s/EBWETpSedaLEx382Y8XpXNI1AVny5Kuab3lDj/g6fXgZbX94AAACAW2t8H/
UVlFkP45xwjlHfnN15bDlY/3viIrPGUkz+QQ5LQEc5WLqx58NUYMT5r0I/
PZqXmJorbnhOaK5siNypYF5DJaB8GMDLjl0uazWjLjn8phEWZpKnt3OpahRaEploJGoors
VqRmxMVwLypTNnJd5NFmvDAH5lfe9o78f4aSA=
```

Output description

Field name	Description
`SSH Server Port`	The port the system uses for SSH sessions.
`Max Authentication Time`	The configured maximum number of seconds allowed for user authentication to complete.
`CLI Inherits SSH Login Credentials`	This transparently passes the username and user password to the CLI from the sshd authentication mechanism so the user does not need to enter the information twice. **Note**: This is available only if the user is logged into the sshd session with a valid username and user password.
`Error Handler`	The state of the error handler, either `enabled` or `disabled`.
`Global Event Interval`	The configured frequency with which the system reports similar SSH events.
`Patch Vendors`	The SSH vendor(s) for whom patches may be applied. This vendor name is set with the `sshd advanced` command.
`Client Keep Alive Interval`	The number of seconds between “keep alive” messages sent to the client. If the client does not respond after three keep alives, the system destroys the session. Avalue of 0 may have been set because the client application doesn't support error response properly.

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → advanced

show advanced testPatchInfo

Purpose

Displays global configuration and operational test patch information. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices sshd advanced testPatchInfo
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show advanced testPatchInfo
confTestPatchVersion:
confTestPatchValue:
operTestPatchVersion:
operTestPatchValue:
operTestPatchSummary:
operTestPatchBitVal:  0
```

Output description

Field name	Description
confTestPatchVersion	The test patch version text, exactly as you set it with the sshd advanced command.
confTestPatchValue	The sum value of the test patch value bits set with the sshd advanced command.
operTestPatchVersion	The operational test patch version text, which should match confTestPatchVersion.
operTestPatchValue	The operational test patch value text, which should match confTestPatchValue.

Field name	Description
`operTestPatchSummary`	The names of all operational patches configured and applied.
`operTestPatchBitVal`	The decimal value of the test patch value. A 0 indicates that no patches are applied.

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → advanced → testPatchInfo

show algorithms

Purpose

Displays the configured and active global settings for the algorithms that control the SSH connection. The operational settings are the active global settings, while the supported settings are those that the N2000 Series supports. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices sshd algorithms
```

Sample output

```
sun> enable
sun# switchservices
sun(switchservices)# sshd
sun(switchServices sshd)# show algorithms
Operational Encryption:          3des-cbc,blowfish-cbc,des
Operational Hmac:
hmac-md5,hmac-sha1,hmac-md5-96,hmac-sha1-96
Operational Compression:         none
Operational User Authentication:  publickey,password
Operational Host Authentication: none
Supported Encryption:
none,des,3des,blowfish,3des-cbc,blowfish-cbc
Supported Hmac:
hmac-sha1,hmac-sha1-96,hmac-md5,hmac-md5-96
Supported Compression:           none
Supported User Authentication:   publickey,password,none
Supported Host Authentication:   'none' OR <comma separated list of IP
Addresses> (where: !-Not, '*'-multi-char wildcard, '?'-single-char
wildcard)
```

Output description

Field name	Description
`Encryption`	A list of one or more configured encryption algorithms. Possible values: `des`, `3des`, `3des-cbc`, `blowfish`, `blowfish-cbc`, and `none`.
`Hmac`	A list of one or more configured HMAC algorithms. Possible values: `hmac-md5`, `hmac-sha1`, `hmac-md5-96`, and `hmac-sha1-96`.
`User Authentication`	A list of one or more accepted user authentication methods. Possible values: `publicKey`, `password`, and `none`.
`Host Authentication`	A list of client addresses that you want to explicitly permit or deny. If the output is preceded with an exclamation point (`!`), the system denies the identified hosts.
`Supported Encryption`	The encryption algorithms that the system supports: none,des,3des,blowfish,3des-cbc, and blowfish-cbc.
`Supported Hmac`	The HMAC algorithms that the system supports: hmac-sha1,hmac-sha1-96,hmac-md5, and hmac-md5-96.
`Supported Compression`	The compression algorithms that the system supports: none.
`Supported User Authentication`	The user authentication methods that the system supports: publickey, password, and none.
`Supported Host Authentication`	The methods you can use to specify a list of client address to explicitly permit or deny.

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → algorithms

show clientKey

Purpose

Displays summary information about configured SSH client keys. To display operational client keys, use the `show clientKeyStatus` command. All fields displayed by the command are configurations assigned with the `clientKey` command. This command does not support field filtering.

Access mode

user

Syntax

```
show switchServices sshd clientKey
```

Sample output

The following output illustrates several ways of displaying configured clients. The first output is an operational client, the subsequent output shows examples of manually entered text and filtering mechanisms.

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show clientKey
Host Address: 1.1.1.1
User Name:    JohnDoe
Key Status:   notInService
Key Data:     Activate Monday

Host Address: 2.2.2.2
User Name:    JaneDoe
Key Status:   notInService
Key Data:     Activate Tuesday

sun(switchServices sshd)# show clientKey 1*
Host Address: 1.1.1.1
User Name:    JohnDoe
Key Status:   notInService
Key Data:     Activate Monday

sun(switchServices sshd)# show clientKey userName JaneDoe
Host Address: 2.2.2.2
```

```
User Name:    JaneDoe
Key Status:   notInService
Key Data:     Activate Tuesday

sun(switchServices sshd)# show clientKey keyData *Monday
Host Address: 1.1.1.1
User Name:    JohnDoe
Key Status:   notInService
Key Data:     Activate Monday
```

Output description

Field name	Description
`Host Address`	The IP address associated with the displayed key.
`User Name`	The user name assigned to the specified host address.
`Key Status`	The configured status of the client key, either active or not in service. Status can be modified with the `sshd clientKey` command.
`Key Data`	The key data entered, with the `sshd clientKey` command, when the specified client key was created or modified.

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → clientKey

show clientKeyStatus

Purpose

Displays summary information about operational SSH client keys. To display all configured client keys, use the `show clientKey` command. All fields displayed by the command are configurations assigned with the `clientKey` command. This command does not support field filtering.

Note: If an entry appears in the `show clientKey` command display, but does not appear in the display of this command after a reasonable amount of time, then there is probably something wrong with the entry. It may have been improperly formatted or entered.

Access mode

user

Syntax

```
show switchServices sshd clientKeyStatus
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show clientKeyStatus
Host Address: 0.0.0.0
User Name:    JohnDoe
Key Status:   active
Key Data:
AAAAB3NzaC1kc3MAAACBAN0kqGdOoyDRCMj9GFAkjC2xXE7vQkQ1sZ2MqbdksUBcnFHe69
Vn5l34ShlTp9sA8a2w1Qh6Y2jUgdPSriJsUkSVYWYvxNUS9P1uR58A
tS4H17YoRcH91Fspji+iS9ewrPUBruDcCGlkyk9y72so/MEclyKm19HdOHjmcc27MFo/
AAAAFQDG4Dd4bRdqlsvLRCav2Z1ZZD0cpwAAAIAHMVwmSOjzZBU8wOaj6cFcquUElLFIQk
jwp9zz6ferE1NNo07MpfiSJdwycXNzIKmLM0cLA2VnAtWOFxzVKQn30K3DFq4RR0XlwgcT
/S+e9LeesN94n/ZN4uvZRbz0cO7EV3bnFR0tMEDhS16BtcBkUC0DCy/05vIIBzc44XRl
bAAAAIEAgaCLXaDSG+//
0shdgPOmPLLoByUce3+kr34CIbbcUKyZoYFifhBcRuoMdstn9cg1JQUB3DQbsqHNxg4XcJ
eqGHrAJeb7TSg91IgujRcYYFP7cYVQLsIR/0BSVg+qqt6ssK
RlJLbxqciTHQhjSWHKcBuOwYYGIiRqH8PhBayyH+w=
```

Output description

Field name	Description
`Host Address` *`ipAddress`*	The IP address associated with the displayed key.
`User Name` *`text`*	The text description assigned to the specified host address.
`Key Status`	The operational status of the client key, either active or not ready.
`Key Data`	The key data entered, with the `sshd clientKey` command, when the specified client key was created or modified.

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → clientKeyStatus

show sessions

Purpose

Lists the current, active SSH sessions and various characteristics of the session.

Access mode

user

Syntax

```
show switchServices sshd sessions
```

Sample output

The examples below show both the full output of the `show sessions` command and the output filtered by a specified set of parameters.

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show sessions
Client IP:     192.168.209.31
Client Port:   1528
Session State: up
Session Type:  cli
UserName:      davidc
User Privs:    sftpWrite
UserProfile:   pro-davidc(4-SRW)
TX bytes:      1110
RX bytes:      2956
TX Cli bytes:  22
RX Cli bytes:  1
TX Sftp bytes: 0
RX Sftp bytes: 0
Neg Info:      clt->svr: 3des-cbc hmac-md5 none; svr->clt: 3des-cbc
hmac-md5
none; Kex: diffie-hellman-group1-sha1; KeyAlgorithms/Types: ssh-dss;
auth-method: password;
Session Time:  799
Idle Time:     29
SessionStatus: active
PatchIds:      0
PatchSummary:  noPatchesApplied
```

Output description

Field name	Description	Filter name
`Client IP`	The IP address of the SSH client (the user's application that is talking to the server).	`clientIP`
`Client Port`	The number of the TCP port the client is using to connect to the server.	`clientPort`
`Session State`	The state of the SSH session, either `up`, `negotiating`, or `down`.	`operState`
`Session Type`	The session type, either `cli`, `sftp`, or `unknown`. A value of unknown indicates that the client's service request has not been completed. The session type influences the types of counters that are incremented.	`sessionType`
`UserName`	The login name of the user connected over SSH to the server.	`userName`
`User Privs`	The privilege level of the connected user for this session. Possible values: `none`: The session is still negotiating `session`: The user has CLI session privileges `sftpRead`: The user has SFTP read-only session privileges `sftpWrite`: The user has SFTP read-write session privileges Privilege level is set by the authentication subsystem.	`userPrivilegeLevel`
`UserProfile`	The profile associated with the connected user.	`userProfileName`
`TX bytes`	The total number of bytes sent during this SSH session.	`bytesSentTotal`
`RX bytes`	The total number of bytes received during this SSH session.	`bytesReceivedTotal`
`TX Cli bytes`	The bytes sent from the CLI during this SSH session.	`bytesSentCli`

Field name	Description	Filter name
RX Cli bytes	The bytes received, destined for the CLI, during this SSH session.	bytesReceivedCli
TX Sftp bytes	The bytes sent from SFTP during this SSH session.	bytesSentSftp
RX Sftp bytes	The bytes received, destined for SFTP, during this SSH session.	bytesReceivedSftp
Neg Info	A summary of session negotiation information. This summary lists client-to-server and server-to-client encryption methods, HMAC algorithm types, client key algorithm setting, and user authentication method(s). For more detailed negotiation information, use the `show switchServices sshd show sessions advanced negotiations` command.	kexNegotiationSummary
Session Time	The number of seconds that the session has been active.	sesDuration
Idle Time	The number of seconds that the session has been idle. Use the `sshd` command to set the idle timeout timer.	idleDuration
SessionStatus	The operational status for the session.Use the `sshd sessions` command to destroy the session.	sesStatus
PatchIds	The sum of the patch bit IDs applied for this client version. This is session-specific patch information. These bits are applied with the `sshd advanced` command.	sessionPatchMask
PatchSummary	A summary of applied patches for this client version (see the table, “Patch Names and Bit Values” for patch names). This is session-specific patch information. These patches are applied with the `sshd advanced` command.	sessionPatchSummary

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → sessions

show sessions advanced

Purpose

Displays advanced session information. Optionally, you can filter on any of the fields displayed in the output. See Chapter 1, “Using the management interfaces” for information about filtering system output.

Access mode

user

Syntax

```
show switchServices sshd sessions advanced
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show sessions advanced
Client IP:                    10.10.10.1
CLient Port:                  54014
Session Index:                 0
Session Name:                  sshdS:255.0.192.168.209.68:1697
TTY Name:                      /sshdTty_192.168.209.68:1697
DEV Name:                      sshdTty_192.168.209.68:1697
SFTP Name:                     sshdF:192.168.209.68:1697
SFTP Block Name:               sshdFB:192.168.209.68:1697
Socket ID:                     1
Start Time:                    248022
Authentication Start Time:     248022
Authentication Complete Time: 248029
Idle Start Time:               248113
SessionLoop Execution Time:    248113
SftpLoop Execution Time:       248103
Kill Time:                     0
Kill Code:                     none
Termination Cause:
```

Output description

Field name	Description	Filter name
`Client IP`	The client IP address.	`clientIp` *`ipAddress`*
`Client Port`	The port number for the client side of the TCP connection.	`clientPort` *`integer`*
`Session Index`	The internally assigned index number for the session.	`sessionIndex` *`integer`*
`Session Name`	The internally assigned name of this SSH session, based on the client IP address and port.	`sessionName` *`text`*
`TTY Name`	The internally assigned device name of this SSH server, preceded with a slash for use in system directory scripts.	`ttyName` *`text`*
`DEV Name`	The internally assigned device name of this SSH server.	`devName` *`text`*
`SFTP Name`	The internally assigned name for the SFTP process.	`sftpName` *`text`*
`SFTP Block Name`	The internally assigned name for the SFTP memory block.	`sftpBlockName` *`text`*
`Socket ID`	The internal socket ID used by this session.	`sockId` *`integer`*
`Start Time`	The time stamp indicating, in seconds since last boot, when the session began.	`sesStartTime` *`integer`*
`Authentication Start Time`	The time stamp indicating, in seconds since last boot, when user authentication began.	`authStartTime` *`integer`*
`Authentication Complete Time`	The time stamp indicating, in seconds since last boot, when user authentication completed.	`authCompleteTime` *`integer`*
`Idle Start Time`	The time stamp indicating, in seconds since last boot, the beginning of the idle timer. This timer is started from the time of the last received character.	`idleStartTime` *`integer`*
`SessionLoop Execution Time`	The time stamp indicating, in seconds since last boot, the execution of the last session loop.	`lastLoopTimeSession` *`integer`*
`SftpLoop Execution Time`	The time stamp indicating, in seconds since last boot, the execution of the last SFTP loop.	`lastLoopTimeSftp` *`integer`*
`Kill Time`	The time stamp indicating, in seconds since last boot, when the system sent a session kill signal. This information is only available in the time between the event being sent and the action taken.	`sesKillTime` *`integer`*

Field name	Description	Filter name
`Kill Code`	The signal kill code received by session. Possible values: `none`: The reason for session termination is unknown `idleTimeout`: The session terminated because it remained idle beyond the configured time limit `authenticationTimeout`: The session terminated because it did not authenticate within the configured time limit `administrativeTermination`: The session was terminated by an administrator `applicationFailure`: The session terminated because the application terminated `privilegeViolation`: The session terminated because the user attempted an action for which they didn't have proper privileges `sessionExecutionFailure`: The session terminated because an internal execution cycle in the CLI took too long `sftpExecutionFailure`: The session terminated because an internal execution cycle in SFTP took too long `clientKeepAliveError`: The session terminated because the client did not respond to three consecutive keep alives This information is only available in the time between the event being sent and the action taken.	`sesKillCode` *`SshdSessionKillCode`*
`Termination Cause`	The reason for a session termination. This information is only available in the time between the event being sent and the action taken.	`sesTermCause` *`text`*

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → sessions → advanced

show sessions advanced negotiations

Purpose

Displays detailed session negotiation information. Optionally, you can filter on any of the fields displayed in the output. See the “Output description” table for a list of parameter names to use as filtering criteria. See Chapter 1, “Using the management interfaces” for information about filtering system output.

Access mode

user

Syntax

```
show switchServices sshd sessions advanced negotiations
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# sshd
sun(switchServices sshd)# show sessions advanced negotiation
Client IP:          10.10.10.1
CLient Port:        54014
userService:         ssh-connection
authMethod:          password
attemptedAuthMethod: password
Negotiation Details: client->server:   clt-enc
aes128-cbc,aes192-cbc,aes256-cbc,twofish-cbc,blowfish-cbc,3des-cbc,arc
four ; svr-enc  3des-cbc,blowfish-cbc,des ; com-enc  blowfish-cbc ;
clt-mac  hmac-md5,hmac-sha1,hmac-sha1-96,hmac-md5-96 ; svr-mac
hmac-md5,hmac-sha1,hmac-md5-96,hmac-sha1-96 ; com-mac  hmac-md5 ;
clt-cmp  none ; svr-cmp  none ; com-cmp  none ; server->client:
clt-enc
aes128-cbc,aes192-cbc,aes256-cbc,twofish-cbc,blowfish-cbc,3des-cbc,arc
four ; svr-enc  3des-cbc,blowfish-cbc,des ; com-enc  blowfish-cbc ;
clt-mac  hmac-md5,hmac-sha1,hmac-sha1-96,hmac-md5-96 ; svr-mac
hmac-md5,hmac-sha1,hmac-md5-96,hmac-sha1-96 ; com-mac  hmac-md5 ;
clt-cmp  none ; svr-cmp  none ; com-cmp  none ; clt-kex
diffie-hellman-group-exchange-sha1,diffie-hellman-group1-sha1 ;
svr-kex  diffie-hellman-group1-sha1 ; com-kex
diffie-hellman-group1-sha1 ; clt-keyAlg  ssh-dss,ssh-rsa ; svr-keyAlg
ssh-dss ; com-keyAlg  ssh-dss ; auth-method: password ;
Client Version:      SSH-2.0-3.4.4 SecureCRT
```

Output description

Field name	Description	Filter name
`Client IP`	The client IP address.	`clientIp` *`ipAddress`*
`Client Port`	The client port number.	`clientPort` *`integer`*
`User Service`	The type of service the user requested, for example SSH connection or SSH user authentication.	`userService` *`text`*
`Auth Method`	The authentication method SSH used to verify the user.	`userAuthenticationMethod` *`text`*
`Attempted Auth Method`	The last authentication method attempted by the client.	`attemptedUserAuthenticationMethod` *`text`*

Field name	Description	Filter name
`Negotiation Details`	The client-to-server and server-to-client negotiation and authentication settings. Note that the key algorithms lists DSS and RSA for the client, while the server only supports DSS. This is reflected in the "common" listing. The output uses the following abbreviations: `clt-enc`, `svr-enc`, `com-enc`: client, server, or common encryption `clt-mac`, `svr-mac`, `com-mac`: client, server, or common HMAC algorithms `clt-cmp`, `svr-cmp`, `com-cmp`: client, server, or common compression `clt-kex`, `svr-kex`, `com-kex`: client, server, or common key exchange `clt-keyAlg`, `svr-keyAlg`, `com-keyAlg`: client, server, or common key algorithm The authentication method listed is the method that was successful in verifying the user/host.	`kexNegotiationDetails` *`text`*
`Client Version`	The SSH client's version string, if patches are used. This is the version string required by the `testPatchVersion` argument of the `advanced` command.	`clientVersionString` *`text`*

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → sessions → advanced → negotiations

sshd (root)

Purpose

Configures the main parameters of the SSH protocol. Also enters the `sshd` command mode.

All of the parameter settings are optional. Although the connection will operate with default settings if you do not modify them, you must enter a server key ID (with the `serverKeyId` argument) and set `adminState` to `enabled` to activate the SSH protocol.

Access mode

enable

Syntax

```
switchServices sshd
   [adminState {enabled | disabled}]
   [maxSessions integer]
   [serverKeyId indexValue]
   [idleTimeout seconds]
   [authenticationBanner text]
   [confEncryption {none | des | des3 | blowfish | des3Cbc |
      blowfishCbc}]
   [confHmac {sha1 | sha1b96 | md5 | md5b96}]
   [confCompression text]
   [userAuthentication {none | password | publicKey}]
   [hostAuthentication text]
```

Arguments

Argument name	Description
`adminState`	**Optional.** Sets the administrative state of the SSH server, either enabled (running) or disabled (not running). When disabled, the parameters of SSH can still be configured, but do not become active until adminState is set to enabled. Default is `disabled`. You may have to restart the application for a change in the administrative state to take effect.
`maxSessions` *`integer`*	**Optional.** Sets the maximum number of SSH sessions allowed. Valid range is 0 through 10; default number of sessions is 4.

Argument name	Description
`serverKeyId` *`indexValue`*	**Optional.** Identifies the SSH server key. Indicates a named index within the Certificate and Key Manager entity. This key ID must be present for the server to operate. The key ID is generated using the N2000 Series CKM utility.
`idleTimeout` *`seconds`*	**Optional.** Sets the maximum number of seconds a session can remain idle before it is terminated. Use 0 to disable timeout (allow the session to remain live regardless of the duration of inactivity). Valid range is 0 through 86400 seconds; default idle time is 600 seconds (10 minutes).
`authenticationBanner`	**Optional.** Defines a text message sent to the SSH client during authentication. Enter between 0 and 255 alphanumeric characters.
`confEncryption`	**Optional.** Configures the encryption method(s) used. Enter the list of allowable encryption algorithms, in a comma-separated list. Valid values are: `des`: The original Data Encryption Standard (DES) algorithm supported by SSHv1. It uses a private key (from a selection of over 72 quadrillion keys), randomly chosen. DES applies the chosen 56-bit key to each 64 data bits. `des3`: Triple DES algorithm. Applies 3 keys to each 64-bit data block. `des3Cbc`: Triple DES algorithm, required by RFC 1201, with cipher-block chaining. With CBC, a given block of encrypted text is dependent on the encryption results of the previous blocks. `blowfish`: Blowfish algorithm. It is a private key block cipher that uses a variable-length key, from 32 bits to 448 bits. Blowfish is significantly faster than DES. `blowfishCbc`: Blowfish algorithm, with cipher-block chaining. `none`: No data encryption (not recommended). The default setting allows `3des-cbc`, `blowfish-cbc`, and `des`. The order in which you enter the encryption methods is important. When negotiating with the client, the server checks encryption methods based on your order of preference in configuring them, and the first method to match both systems is the method used.

Argument name	Description
confHmac	**Optional.** Configures the Hash Message Authentication Code (HMAC) method. Enter the list of allowable HMAC algorithms, in a comma-separated list. HMAC is a type of checksum to help verify data integrity. Valid values are: `md5:` a 16-byte hash `sha1:` a 20-byte hash `md5b96:` a 16-byte hash, truncated to 96 bits (12 bytes) `sha1b96:` a 20-byte hash, truncated to 96 bits (12 bytes) The default value allows all HMAC algorithms listed. The order in which you enter the algorithms is important. When negotiating with the client, the server checks algorithms based on your order of preference in configuring them, and the first method to match both systems is the method used.

Argument name	Description
`userAuthentication`	**Optional.** Configures the user authentication method. Enter the list of accepted user authentication methods, in a comma-separated list. Valid values for authentication are: `publicKey`: requires cryptographic keys `password`: requires login password `none`: requires no user authentication. If both systems agree on no authentication, you may have seriously compromised security. In addition, the system does not allow file transfers with no authentication. The default authentication accepts both `password` and `publickey`. The client offers methods according to its configured order of preference. All configured authentication methods are attempted until the systems establish a connection or too many attempts have been made.
`hostAuthentication`	**Optional.** Configures allowable clients. Enter a comma-separated list of client addresses. These are the addresses that you want to explicitly permit or deny. The following special characters are accepted: • !: the NOT character, to deny an address • *: a wildcard that replaces multiple characters • ?: a wildcard that replaces a single character **Note**: If you use wildcards, the information must be in brackets. If a wildcard is the last character, the list must be terminated with a comma and a space. For example, {1.2.3.*, }.

Example

The following example disables SSH on the server and changes the default SSH configuration so that the maximum number of sessions is increased and the idle time-out is decreased.

```
sun> enable
sun# switchServices
sun(switchServices)# sshd disabled maxSessions 10 idleTimeout 300
```

Associated MIB

`sshd.mib`

Web path

- switchServices → sshd → modify

Chapter 15. Certificate and Key Manager commands

Certificate and Key Manager description

The Certificate and Key Manager (CKM) utility is a subsystem on the N2000 Series system that allows you to create, manage, and store cryptographic keys and certificates. CKM provides a centralized key management system that serves multiple system applications, such as Secure Sockets Layer (SSL) Protocol and the Secure Shell (SSH) Protocol, to protect data and transactions.

CKM generates a new cryptographic key pair, mathematically related private and public data keys indexed by a unique name. A private key is kept secure—never displayed and never transmitted over the network. A public key, when bound to a fully qualified domain name (FQDN) by an authorized Certificate Authority (CA), becomes an X509 certificate. A certificate is a digitally signed document that identifies the subject, and contains the subject's public key and the digital signature of the CA. It is by this form of authentication that SSL transactions are validated.

Note: You must set the salt before CKM can operate. Until it is set, all CKM commands will fail, returning an error message. Once the salt is set, it is incorporated into the encryption of all keys stored on the switch. For information about setting the salt, see Chapter 3, "Chassis commands".

Identifying keys

Several of the CKM commands use a `keyId` argument to reference the key pairs. When you are working with existing keys, you must supply a *`namedIndex`* (allowing you to tab-complete the existing key pair name). This applies to the following commands:

- `csr`
- `export`
- `no keypair`
- `show keypair`
- `show keypair verbose`
- `import paste` (on an existing key pair)
- `import url` (on an existing key pair)

When you execute commands that create a new key, you must supply *`keyText`* that creates a new `keyId` name. This applies to the following commands:

- `import paste` (on a new key pair)
- `import url` (on a new key pair)
- `generate`

Note: To ensure the highest security, the N2000 Series does not store unencrypted private keys in flash or any other memory that can be dumped in a core image. They can not be displayed in the clear for any reason, regardless of the requestor's access privileges.

CKM for the secure sockets layer

CKM can generate two types of key pairs—Digital Signature Algorithm (DSA) for SSH operations or RSA (named for its authors) for SSL operations. Both types are public key algorithms used for digital signatures; RSA provides encryption as well.

Using CKM to manage keys for SSL

When terminating SSL, a virtual service requires an RSA private key and an RSA X.509 certificate. Through the CKM, you can either generate these keys or move them between systems. For example, you can use the `import` commands to transfer previously generated keys and certificates onto the N2000 Series. Or, you can use the `export` commands to transfer them to other systems.

If generating a new key pair, you can build a Certificate Signing Request (CSR), which you can then send to a CA via its Web site. (Verify the method of data transmission at your chosen CA's Web site.) When the CA returns the data in the form of a certificate, you import it onto the system and it overwrites the public key data with the new certificate data.

Note: Although you can create the keys and certificates necessary to implement SSL communication through the N2000 Series systems with these commands, unless your system contains the hardware to support SSL, the system does not support it. Verify that the FX-SSL module is installed before assuming that SSL is active on your system. Check with your sales representative if you are unsure.

SSL keys on the N2000 Series

The N2000 Series can either generate new keys, or if you have previously generated keys, import those for SSL use.

Configuring for SSL with new keys

Table 15-1 describes the steps for configuring your system as SSL-ready if you have no existing keys.

Table 15-1. Steps for configuring for SSL with no existing keys

Step	Action
1.	Generate new cryptographic key pairs for the system.
2.	Generate a Certificate Signing Request and send it to a Certificate Authority. Optionally, generate a temporary certificate to allow configuration testing.
3.	Import the certificate returned from the CA onto the N2000 Series system.

Configuring for SSL with existing keys

Table 15-2 describes the steps for configuring the N2000 Series system to support an existing SSL Web server.

Table 15-2. Steps for configuring for an existing SSL Web server

Step	Action
1.	Import existing keys onto the system.
2.	Import existing certificates onto the system.

CKM command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `ckm` *`commandname`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

CKM command summary

The following table lists and briefly describes the CKM commands.

Table 15-3. CKM command summary

Command name	Description
`csr`	Generate a Certificate Signing Request (CSR).
`export`	Export a certificate or private key from the N2000 Series.
`generate`	Generate a new cryptographic key pair.
`import paste`	Import ASCII key data to the system.
`import url`	Import ASCII and binary key data to the system.
`no keypair`	Delete a specified key pair.

Table 15-3. CKM command summary (continued)

Command name	Description
`show keypair`	Display all or a specified key pair information.
`show keypair verbose`	Display all or a specified key pair information with full certificate details, if applicable.

csr

Purpose

Generates a Certificate Signing Request (CSR), which you then send to a Certificate Authority (CA). The data is displayed to your screen and you then cut and paste it into a request form available from your chosen CA. After verification and processing, the CA sends a valid certificate that you can import, using either the `import paste` or `import url` command, onto your system. The data displayed between the `BEGIN CERTIFICATE REQUEST` and `END CERTIFICATE REQUEST` output on your screen is the Certificate Signing Request.

The country, state, city, organization and unit names, and fully qualified domain name (FQDN) make up the distinguished name. Each certificate has two such names, one for the requester (subject) and one for the issuing body. This information is maintained in the certificate to identify both parties in all SSL transactions.

This command includes a password option to help protect security. By specifying a password, you provide a mechanism to allow a CA to revoke the certificate should the private key portion become compromised. Check with your CA to verify whether they support this function.

Note: While many of the distinguished name fields are optional, we recommend that you fill out all that are appropriate. Check with your CA to verify which fields are expected.

Access mode

config

Syntax

```
vSwitch name ckm csr
   keyId namedIndex
   fqdn text
   [country text]
   [state text]
   [locality text]
   [orgName text]
   [orgUnitName text]
   [email text]
   [password passwordText]
   [makeTestCert {true | false}]
   [testCertDays days]
   [CA {true | false}]
```

Arguments

Argument name	Description
`keyId` *namedIndex*	Specifies the name that identifies the key pair. This value is the name you assigned when you generated the key pair with the `ckm generate` command.
`fqdn` *text*	Identifies the fully qualified domain name of the organization that holds this registration. This must be a registered name for the certificate to be issued. The FQDN must be between 3 and 64 characters, inclusive.
`country` *text*	**Optional.** Identifies the country in which the registering company is operating. Use the two-digit code listed in ISO 3166.
`state` *text*	**Optional.** Identifies the state or province in which the registering company is operating. Do not enter an abbreviated state name. If there is no state name, leave the field blank. The state name must be 128 or fewer characters.
`locality` *text*	**Optional.** Identifies the city or locality in which the registering organization is operating. The locality name must be 128 or fewer characters.
`orgName` *text*	**Optional.** Identifies the name of the organization that holds this registration. Use quotation marks around the name if you are entering multiple words with spaces. The organization name must be 60 or fewer characters.
`orgUnitName` *text*	**Optional.** Identifies the name of the department within the organization that is responsible for this registration. The organization unit name must be 32 or fewer characters.

Argument name	Description
email *text*	**Optional.** Enters a valid email address for a contact within the organization. The email address must be 128 or fewer characters.
password *passwordText*	**Optional.** Associates a password that the certificate holder can later use to revoke the certificate (if this mechanism is supported by the CA). The password must be 255 or fewer characters.
makeTestCert {true | false}	**Optional.** Configures the system to issue a temporary certificate signed by your own organization if set to `true`. This certificate can be used for testing the configuration, such as verifying SSL operations. The default setting is `false` (no temporary certificate issued).
testCertDays *days*	**Optional.** Sets the number of days that a temporary test certificate remains valid. Valid range is 1 through 730; the default is 30 days.
CA {true | false}	**Optional.** Requests from the CA the ability for this organization to have certificate issuing capabilities if set to `true`. The default is `false` (the requesting organization does not request issuing privileges).

Example

The following example generates a certificate signing request based on the key indexed by entry 22. The data that is displayed can be copied from the screen and pasted into the requested field at your CA's web site.

```
sun> enable
sun# vswitch system
sun(vSwitch-system)# ckm
sun(config-vSwitch-system ckm)# csr keyId user1 fqdn www.sun.com
country US state Massachusetts locality Framingham orgName "Sun
Networks" orgUnitName "IT Admin" email jdoe@sun.com password certTest
makeTestCert true
-----BEGIN CERTIFICATE REQUEST-----
MIICWjCCAcMCAQAwgYQxCzAJBgNVBAYTAlVTMRYwFAYDVQQIEw1NYXNzYWNodXNl
dHRzMRMwEQYDVQQHEwpGcmFtaW5naGFtMRowGAYDVQQKExFOYXV0aWN1cyBOZXR3
b3JrczERMA8GA1UECxMISVQgQWRtaW4xGTAXBgNVBAMTEHd3dy5uYXV0aWN1cy5j
b20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAM3qQrfpe/KQGOKtBAcDgovu
bAWStX0PE9Mar9yWR8qpF5ZG28bh1XDyHPUfr19hxOgw0BmoMtbF221b5GvJBX+2
6P+tEUuuOSUvQiQUatqC/gcOoFpsjoMfUf7XdY8aODPrwfOg+0STMflKez5yIv+a
nTO67YcS6rE+tTfuZjirAgMBAAGggZQwFwYJKoZIhvcNAQkHMQoTCGNlcnRUZXN0
MCAGCSqGSIb3DQEJATETFhFqZG9lQG5hdXRpY3VzLmNvbTBXBgkqhkiG9w0BCQ4x
SjBIMBsGA1UdEQQUMBKCEHd3dy5uYXV0aWN1cy5jb20wHAYDVR0RBBUwE4ERamRv
ZUBuYXV0aWN1cy5jb20wCwYDVR0PBAQDAgSwMA0GCSqGSIb3DQEBBQUAA4GBAJIg
WtZjnBaOwoVKV2I5WTE6QP/eIuajmi50gl8m0TgIVnSWyYw8GvXlFJavOHWna78d
eGMSKAolbX2ntXMdTfPT6Y0WOmeAFKnjC3t3m7AtzG6OsfHa+h+e7zciZdN+2CHf
vCu/GMybGHmItliISauBTKbZ6PWfDxTvpXm9ayyq
-----END CERTIFICATE REQUEST-----
```

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → csr

export

Purpose

Displays to the screen the data contained in a private key or certificate associated with the specified index entry. You can then copy the data and import it (by pasting or by URL) into a different N2000 Series system using the `import paste` or `import url` command. The format generated by the N2000 Series system for private keys and certificates is Sun proprietary. Using the CKM `import` and `export` commands allows you to install the required information onto multiple systems within an enterprise.

The `password` argument creates an association between the supplied password and the key being exported. Upon later import of that key, with either import command, you will be required to supply this password. The system encrypts the key on export, and requires the associated password to decrypt the key on import. Once imported, there is no password associated with the keys. If you are importing an existing private key that was not generated on an N2000 Series system, you must use the password that you associated with the key when you exported it from your Web server.

Access mode

config

Syntax

```
vSwitch name ckm export
   keyId namedIndex
   pairHalf {privateKey | certificate}
   [password passwordText]
```

Arguments

Argument name	Description
`keyId` *`namedIndex`*	Specifies the name that identifies the key pair. This value is the text string you assigned when you generated the key pair with the `ckm generate` command or imported it with one of the `import` commands.
`pairHalf {privateKey | certificate}`	Specifies which half of the key pair you are exporting, either the private key or the certificate.
`password` *`passwordText`*	**Optional.** Specifies a text string, between 4 and 255 characters, that the system associates with a private key specified by the `keyId` argument. The password argument is required when exporting private keys; the system does not use passwords when exporting certificates.

Example

The following example exports a certificate with a `keyID` of `keyHari`. When the system displays the certificate data, you can copy the certificate and import it onto another system.

```
sun> enable
sun# vswitch e-commerce
sun(vSwitch-e-commerce)# ckm
sun(config-vSwitch-e-commerce ckm)# export keyID keyHari pairHalf
certificate
-----BEGIN CERTIFICATE-----
MIICLDCCAdagAwIBAgIBATANBgkqhkiG9w0BAQUFADBiMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCTUExCzAJBgNVBAcTAkZSMQ0wCwYDVQQKEwROQVVUMQwwCgYDVQQL
EwNTUUExHDAaBgNVBAMTE3d3dy5uYXV0aWN1c25ldC5jb20wHhcNMDMwMTA4MTMw
ODE0WhcNMDQwMTA4MTMwODE0WjBiMQswCQYDVQQGEwJVUzELMAkGA1UECBMCTUEx
CzAJBgNVBAcTAkZSMQ0wCwYDVQQKEwROQVVUMQwwCgYDVQQLEwNTUUExHDAaBgNV
BAMTE3d3dy5uYXV0aWN1c25ldC5jb20wXDANBgkqhkiG9w0BAQEFAANLADBIAkEA
rvoNbeLZHXkvR2bOPXgbtFrjx2mTFrP6f9mLbeN16118vFnqmKQeWfN0Tju6/tyr
3wIZOML65EF/PTQPDqUXUwIDAQABo3cwdTAeBgNVHREEFzAVghN3d3cubmF1dGlj
dXNuZXQuY29tMBYGA1UdEQQPMA2BC2hrQG5hdXQuY29tMA8GA1UdEwEB/wQFMAMB
Af8wCwYDVR0PBAQDAgG2MB0GA1UdDgQWBBRNoGj4lxUsZkbaaXgUWocREXlajjAN
BgkqhkiG9w0BAQUFAANBAJg8bgtx4MrNFWK62uf8VdIk8QkYHmsCXtETr/C5jvvd
JBL/RehG1dRxfkQ82Z+NmqjAqBU8CMe4sD/MoUkp4RI=
-----END CERTIFICATE-----
sun(config-vSwitch-e-commerce)#
```

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → export

generate

Purpose

Creates cryptographic key pairs on the N2000 Series system. Given a key pair name, the command randomly generates a new key pair. The keys can be of two types—Digital Signature Algorithm (DSA) for SSH operations or RSA (named for its authors) for SSL operations. Both types are public key algorithms used for digital signatures; RSA provides encryption as well. Because of the complexity of the mathematical operations, the key generation may take several moments. The N2000 Series can support 512 1024-byte certificates per virtual switch.

Access mode

config

Syntax

```
vSwitch name ckm generate
   keyId keyText
   [bitlength {512 | 1024 | 2048}]
   [algorithm {dsa | rsa}]
```

Arguments

Argument name	Description
`keyId` *`keyText`*	Specifies the name that will identify the newly generated key pair. Enter a text string of up to 64 characters. You can include letters, digits, and the special characters hyphen (-), underscore (_), dot(.), colon (:), at sign (@), and forward slash (/).
`bitlength {512 | 1024 | 2048}`	**Optional.** Sets the number of bits of data the key pair contains. The more bits, the stronger the key, and the longer the key generation process takes. Valid values are 512, 1024, and 2048; the default is 1024.
`algorithm {dsa | rsa}`	**Optional.** Specifies the public key algorithm, either DSA or RSA. SSL requires RSA keys, and SSH requires DSA keys. The default algorithm type is `rsa`.

Example

The following example generates one RSA key pair and one DSA key pair.

```
sun> enable
sun# vswitch system
sun(vSwitch-system)# ckm
sun(vSwitch-system ckm)# generate keyId vs1 bitLength 1024 algorithm rsa
.....................................++++++
..........++++++
sun(config-vSwitch-system ckm)# generate keyId testKey bitLength 512 algorithm
dsa
sun(config-vSwitch-system ckm)# show keypair
Key ID:     testKey
Algorithm:  dsa
Bit Length: 512
Type:       uncertified key pair
Details:    Fingerprint:c3:fc:7b:63:74:85:24:50:4f:66:ed:e7:3b:c1:99:e1

Key ID:     user1
Algorithm:  rsa
Bit Length: 1024
Type:       uncertified key pair
Details:
            Fingerprint: 5b:e4:16:37:65:ab:f4:f5:31:fe:16:b5:98:1e:d4:99
```

Associated MIB

ckm.mib

Web path

- vSwitch → *name* → ckm → generate

import paste

Purpose

Imports private keys or certificates onto the N2000 Series system. Use this command to import data in ASCII format (Privacy Enhanced Mail - PEM or Sun formats). To import a key and certificate together, use the `import url` command.

When executing the command, after you press [Return] you are prompted to paste in your data, either the private key or certificate. This data was either exported from another system, generated with the `generate` command, or received from the Certificate Authority (CA).

Importing certificates

The certificate you import could either be a newly assigned certificate returned from a CA or an existing certificate moving from one system to another. If you are importing a certificate chain, the data of the subject of the certificate is entered first, and then the data of the issuers. The issuer data is ordered from most directly responsible for certificate issuance on back. That is, the intermediary directly responding to your request would be next, the CA backing the intermediary after that, a CA backing that CA after that, and so on. Do not press Ctrl+Z until all data has been pasted into the buffer.

This command has an optional password argument. If you are importing a certificate that you exported from another N2000 Series system, do not use the password argument. Because certificates are public keys and not sensitive information, exporting does not require a password.

Importing private keys

You may import private key data onto the N2000 Series system after exporting a key with the `export` command. Or, you can move data that was exported from another system on to your local system.

To import either private keys or certificates, you must have copied the data to your clipboard, as the command requires you paste it in at the command line.

Note: Any keys imported must meet the supported key size requirements of 512, 1024, or 2048 bytes.

The optional password argument is for private keys that are encrypted under a password. If the private key is not encrypted, you do not need to specify a password. See Table 15-4 for rules for password use on import.

Access mode

config

Syntax

```
vSwitch name ckm import paste
   keyId keyText
   pairHalf {privateKey | certificate}
   format {pem | nauticus}
   [password passwordText]
```

Arguments

Argument name	Description
`keyId` *`keyText`*	Specifies the index entry that identifies the certified public key (which became your certificate) or private key that you are importing. Enter a text string up to 64 alphanumeric characters. You can include the special characters hyphen (-), underscore (_), dot (.), colon (:), at sign (@), and forward slash (/). Although this data may have existed in another location, the name assigned on its creation is not imported. However, it is recommended that for consistency you use the same name between systems. In that way, if you import a configuration that uses a named `keyId`, the system will have that `keyId` entry stored.
`pairHalf {privateKey | certificate}`	Specifies which type of information (which half of the key pair) you are importing, either the private key or the certificate.

Argument name	Description
`format`	Specifies the certificate format, generated by the exporting web server: `pem`: Privacy Enhanced Mail format, from any OpenSSL-based web server (ASCII) `internalCkm`: Sun proprietary, generated by the `ckm export` command (ASCII)
`password` *`passwordText`*	**Optional.** Specifies an alphanumeric text string, 255 or fewer characters, that was associated with the certificate when it was exported. See the `export` command description for more information.

Example

The following example shows how to use the import paste feature to import a certificate that is in the Sun proprietary format.

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# ckm
sun(config-vSwitch-e-commerce ckm)# import paste keyID 5 pairHalf certificate
format nauticus
Please enter your data ctrl-z to accept ctrl-c to cancel:
-----BEGIN CERTIFICATE-----
MIICLDCCAdagAwIBAgIBATANBgkqhkiG9w0BAQUFADBiMQswCQYDVQQGEwJVUzEL
MAkGA1UECBMCTUExCzAJBgNVBAcTAkZSMQ0wCwYDVQQKEwROQVVUMQwwCgYDVQQL
EwNTUUExHDAaBgNVBAMTE3d3dy5uYXV0aWN1c25ldC5jb20wHhcNMDMwMTA4MTMw
ODE0WhcNMDQwMTA4MTMwODE0WjBiMQswCQYDVQQGEwJVUzELMAkGA1UECBMCTUEx
CzAJBgNVBAcTAkZSMQ0wCwYDVQQKEwROQVVUMQwwCgYDVQQLEwNTUUExHDAaBgNV
BAMTE3d3dy5uYXV0aWN1c25ldC5jb20wXDANBgkqhkiG9w0BAQEFAANLADBIAkEA
rvoNbeLZHXkvR2bOPXgbtFrjx2mTFrP6f9mLbeN16118vFnqmKQeWfN0Tju6/tyr
3wIZOML65EF/PTQPDqUXUwIDAQABo3cwdTAeBgNVHREEFzAVghN3d3cubmF1dGlj
dXNuZXQuY29tMBYGA1UdEQQPMA2BC2hrQG5hdXQuY29tMA8GA1UdEwEB/wQFMAMB
Af8wCwYDVR0PBAQDAgG2MB0GA1UdDgQWBBRNoGj4lxUsZkbaaXgUWocREXlajjAN
BgkqhkiG9w0BAQUFAANBAJg8bgtx4MrNFWK62uf8VdIk8QkYHmsCXtETr/C5jvvd
JBL/RehG1dRxfkQ82Z+NmqjAqBU8CMe4sD/MoUkp4RI=
-----END CERTIFICATE-----
[Ctrl-Z]
```

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → import → paste

import url

Purpose

Imports either private key data, public key data (or if already processed by a CA, a certificate), or the key pair data onto the N2000 Series system. See the `import paste` command for a full description of the key and certificate import process. Use this command to import data in ASCII or binary format (see the table below).

When using this command, you can import a private key and certificate together by specifying `both` with the `pairHalf` argument. If importing both, you must specify one of IIS4 or PKCS12 in the `format` argument.

This command has an optional password argument. The following table summarizes the rules for password use on import.

Table 15-4. Rules for password use on import

Format	Rule
PEM	Optional password on import. To determine, examine the data header. If a password is required, header displays “ENCRYPTED”.
DER	Never requires a password on import.
IIS4	Probably requires a password on import. If the password was specified as " " on export, leave the argument blank.
PKCS #12	Probably requires a password on import. If the password was specified as " " on export, leave the argument blank.
Nauticus	Always requires a password on private key import, never on certificate import.

Access mode

config

Syntax

```
vSwitch name ckm import url
   keyId keyText
   pairHalf {privateKey | certificate | both}
   format {pem | der | iis4 | pkcs12 | nauticus}
   data urlInput
   [password passwordText]
```

Arguments

Argument name	Description
`keyId` *`keyText`*	Specifies the name that will identify the imported private key or certificate. Enter a text string of up to 64 characters. You can include letters, digits, and the special characters hyphen (-), underscore (_), dot(.), colon (:), at sign (@), and forward slash (/).
`pairHalf {privateKey | certificate | both}`	Specifies which half of the key pair you are importing, the private key, the certificate, or both. If importing both, you must specify one of the two binary formats—IIS4 or PKCS12— in the `format` argument.
`format`	Specifies the key/certificate format generated by the exporting web server or N2000 Series system: `pem`: Privacy Enhanced Mail format, from any OpenSSL-based web server (ASCII) `der`: Distinguished Encoding Rules format, ASN.1, from any OpenSSL-based web server (binary) `iis4`: Microsoft Internet Information Server (IIS), Version 4 (binary) `pkcs12`: Public Key Cryptography Standard #12 format, often from Microsoft IIS Version 5 (binary) `internalCkm`: Sun proprietary, generated by the `ckm export` command (ASCII)
`password` *`passwordText`*	**Optional.** Specifies a text string, 255 or fewer characters, that was associated with the certificate when it was exported. See the `export` command description for more information.
`data` *`urlInput`*	Specifies the path to the data. Enter a name in the format *`file:/filePath`*. This data was either exported from another system or received back from a CA.

Example

The following example shows how to use the `import url` feature to import a public key and certificate file that originally came from an IIS4 web server.

```
sun> enable
sun# vSwitch e-commerce
sun(vSwitch-e-commerce)# ckm
sun(vSwitch-e-commerce ckm)# import url 5 pairHalf both format iis4
file:/keyfile.key
sun(vSwitch-e-commerce ckm)# savecfg
```

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → import → url

no keypair

Purpose

Deletes either all configured key pairs on the system, or if you specify optional arguments, only those that match all arguments. Use the `show keypair` command to view the values for the arguments used to identify the key pair with this command.

Note: When using the Web interface, this command is not visible unless you have at least one configured key pair.

Access mode

config

Syntax

```
vSwitch name ckm no keypair
   keyId namedIndex
   [algorithm {dsa | rsa}
   [keyLen {512 | 1024 | 2048}]
   [keyType {certifiedPair | uncertifiedPair | privateKey | publicKey
      | certificate | empty | malformed}]
   [keydesc text]
```

Arguments

Argument name	Description
`keyId` *namedIndex*	Deletes the key pair matching the specified name. This value is the text string you assigned when you generated the key with the `ckm generate` command or imported the data with one of the import commands.
`algorithm {dsa | rsa}`	**Optional.** Deletes key pairs matching the public key algorithm type DSA or RSA. SSL requires RSA keys, and SSH requires DSA keys.
`keyLen {512 | 1024 | 2048}`	**Optional.** Deletes key pairs of the specified length, in number of bits.

Argument name	Description
`keyType {certifiedPair | uncertifiedPair | privateKey | publicKey | certificate | empty | malformed}`	**Optional.** Deletes the specified key pair type. The valid values are: `certifiedPair`—both a private key and the corresponding X.509 certificate (used for SSL) `uncertifiedPair`—both a private key and the corresponding raw public key (used for SSH or for SSL while waiting for a CA response) `privateKey`—just a private key, no public key (probably due to forgetting to import the corresponding X.509 certificate) `public key`— just a raw public key, no private key (uncommon) `certificate)`— just an X.509 certificate, no private key (used for backend SSL, or due to failing to import the corresponding private key) `empty`—an empty key pair `malformed`—a malformed key pair
`keyDesc` *`text`*	You cannot filter on this field. See the `show keypair` command for details.

Example

The following example shows the deletion of a key pair named `test`. In this example, the `show keypair` command shows information about two key pairs configured on the system. After the delete operation, the `show keypair` command confirms that the `no keypair` command deleted the key pair named `test`.

The system does not prompt you to confirm the deletion or display a message after the deletion.

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# ckm
sun(vSwitch-e-commerce ckm)# show keypair
Key ID:     sshKey
Algorithm:  dsa
Bit Length: 1024
Type:       uncertifiedPair
Details:    Fingerprint: e6:57:09:2b:6d:af:0f:67:a6:c7:fe:7a:63:5c:41:b5

Key ID:     vs1Key
Algorithm:  rsa
Bit Length: 1024
Type:       certifiedPair
Details:    Fingerprint: aa:5f:ef:9c:d8:f9:23:d6:7c:ce:f0:fd:f3:b1:75:4d
            X509 Issuer: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com
            X509 Subject: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com

sun(vSwitch-e-commerce ckm))# no keypair sshKey

sun(vSwitch-e-commerce ckm)# show keypair
Key ID:     vs1Key
Algorithm:  rsa
Bit Length: 1024
Type:       certifiedPair
Details:    Fingerprint: aa:5f:ef:9c:d8:f9:23:d6:7c:ce:f0:fd:f3:b1:75:4d
            X509 Issuer: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com
            X509 Subject: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com

sun(vSwitch-e-commerce ckm)
```

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → keypair → delete

show keypair

Purpose

Displays the key ID, associated algorithm type, key length, and summary certificate information for each key stored in the system. You can filter the display by supplying either the key ID, algorithm type, or bit length.

Note: When using the Web interface, this command is not visible unless you have at least one configured key pair.

Access mode

user

Syntax

```
show vSwitch name ckm keypair
```

Sample output

```
sun> enable
sun# vswitch system
sun(vSwitch-system)# ckm
sun(vSwitch-system ckm)# show keypair
Key ID:     sshKey
Algorithm:  dsa
Bit Length: 1024
Type:       uncertifiedPair
Details:    Fingerprint: e6:57:09:2b:6d:af:0f:67:a6:c7:fe:7a:63:5c:41:b5

Key ID:     vs1Key
Algorithm:  rsa
Bit Length: 1024
Type:       certifiedPair
Details:    Fingerprint: aa:5f:ef:9c:d8:f9:23:d6:7c:ce:f0:fd:f3:b1:75:4d
            X509 Issuer: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com
            X509 Subject: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com
```

Output description

Field name	Description	Filter name
`keyId`	The name that identifies the key pair. This value is the text string you assigned when you generated the key with the ckm generate command or imported the data with one of the `import` commands.	`keyId` *`namedIndex`*
`algorithm`	The key's algorithm type, either DSA or RSA.	`algorithm {dsa | rsa}`
`bitlength`	The number of bits of data the key contains.	`keyLen` *`integer`*
`type`	The key pair type. The values are as follows, and indicate that a key pair is: `certifiedPair`—both a private key and the corresponding X.509 certificate (used for SSL) `uncertifiedPair`—both a private key and the corresponding raw public key (used for SSH or for SSL while waiting for a CA response) `privateKey`—just a private key, no public key (probably due to forgetting to import the corresponding X.509 certificate) `public key`— just a raw public key, no private key (uncommon) `certificate)`— just an X.509 certificate, no private key (used for backend SSL, or due to failing to import the corresponding private key) `empty`—an empty key pair `malformed`—a malformed key pair	`keyType {certifiedPair | uncertifiedPair | privateKey | publicKey | certificate | empty | malformed}`
`details`	Summary information about the public key data, including fingerprint data and certificate details if applicable. You cannot filter on this field.	N/A

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → keypair

show keypair verbose

Purpose

Displays detailed certificate information, specifically all certificate fields, as well as the key ID, associated algorithm type, and key length for each key stored in the system. You can filter the display by supplying either the key ID, algorithm type, or bit length.

Access mode

user

Syntax

```
show vSwitch name ckm keypair verbose
```

Sample output

```
sun> enable
sun# vswitch system
sun(vSwitch-system)# ckm
sun(vSwitch-system ckm)# show keypair verbose
Key ID:     sshKey
Algorithm:  dsa
Bit Length: 1024
Type:       uncertifiedPair
Details:    Fingerprint: e6:57:09:2b:6d:af:0f:67:a6:c7:fe:7a:63:5c:41:b5

Key ID:     vs1Key
Algorithm:  rsa
Bit Length: 1024
Type:       certifiedPair
Details:    Fingerprint: aa:5f:ef:9c:d8:f9:23:d6:7c:ce:f0:fd:f3:b1:75:4d
            X509 Certificate:
                Data:
                    Version: 3 (0x2)
                    Serial Number: 1 (0x1)
                    Signature Algorithm: sha1WithRSAEncryption
                    Issuer: C=US, ST=MA, L=Framingham, O=org_name,
            OU=vs-lnx0-L4S, CN=www.vs1.com
                    Validity
                        Not Before: Dec 11 16:53:01 2002 GMT
                        Not After : Dec 10 16:53:01 2004 GMT
```

```
        Subject: C=US, ST=MA, L=Framingham, O=org_name,
OU=vs-lnx0-L4S, CN=www.vs1.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:e4:c1:87:63:11:7d:34:e8:b1:4c:b3:9c:47:fa:
                    28:d5:c5:4c:e7:9f:17:67:9e:8a:de:f5:b5:c4:9c:
                    2e:a7:43:25:21:7b:30:54:28:7d:ea:f0:76:ff:e2:
                    ab:b8:82:b2:01:84:3e:d5:c4:18:b7:b4:92:1f:d2:
                    96:96:87:bc:0e:4e:f5:81:da:29:2c:d3:f7:76:36:
                    fa:dc:19:d7:de:c1:d5:a4:69:79:46:fb:2f:28:02:
                    0d:dd:da:8c:c5:02:cd:80:28:d0:fb:58:06:df:f1:
                    82:14:0a:d8:d5:05:71:3a:52:5d:86:c6:24:0b:6c:
                    24:8a:ca:1b:37:63:a4:10:cf
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:www.vs1.com
            X509v3 Subject Alternative Name:
                email:org_unit_name
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 Key Usage:
                Digital Signature, Key Encipherment, Data
Encipherment, Certificate Sign, CRL Sign
            X509v3 Subject Key Identifier:
                44:67:54:3F:F9:AC:42:29:10:A0:32:AB:4A:90:3D:0D:14:
1A:30:21
    Signature Algorithm: sha1WithRSAEncryption
        4a:db:ec:bc:fc:1f:ab:57:d5:54:f8:55:4d:f9:3f:c1:16:96:
        39:4c:cd:14:8f:98:c3:ff:e2:47:97:84:7b:b2:40:87:61:d3:
        2e:9d:ae:40:5c:92:53:1c:3e:f7:73:39:ed:7d:83:ae:14:bd:
        79:9b:bb:f8:70:23:6c:58:a3:96:9a:5c:83:bc:d3:af:aa:fd:
        72:b8:1d:4c:7e:5f:63:b9:a8:dc:c2:df:c4:f9:fc:99:5b:5c:
        d3:38:e5:43:2f:5f:f6:78:ee:d9:aa:d8:93:90:18:3c:28:50:
        28:30:63:d4:1d:18:5e:f9:35:91:e0:1d:fc:25:e0:e2:8c:52:
        f1:fb
```

Output description

Field name	Description	Filter name
`keyId`	The index entry that identifies the key pair. This value is the text string you assigned when you generated the key with the `ckm generate` command or imported the data with one of the `import` commands.	`keyId` *`namedIndex`*
`algorithm`	The key's algorithm type, either DSA or RSA.	`algorithm {dsa | rsa}`
`bitlength`	The number of bits of data the key contains.	`keyLen` *`integer`*

Field name	Description	Filter name
`type`	The key pair type. To filter on this field, be sure to enclose the text in quotation marks. `certifiedPair`—both a private key and the corresponding X.509 certificate (used for SSL) `uncertifiedPair`—both a private key and the corresponding raw public key (used for SSH or for SSL while waiting for a CA response) `privateKey`—just a private key, no public key (probably due to forgetting to import the corresponding X.509 certificate) `public key`— just a raw public key, no private key (uncommon) `certificate)`— just an X.509 certificate, no private key (used for backend SSL, or due to failing to import the corresponding private key) `empty`—an empty key pair `malformed`—a malformed key pair	`keyType {certifiedPair | uncertifiedPair | privateKey | publicKey | certificate | empty | malformed}`
`details`	Complete details about the public key data. If the type is `uncertifiedPair`, the output is the same as that of the `ckm show summary` command. If the `type` is `certifiedPair`, the system displays full X509 certification fields and details. You cannot filter on this field.	N/A

Associated MIB

`ckm.mib`

Web path

- vSwitch → *name* → ckm → verbose

Part IV. Virtualization

The chapters in Part IV describe the commands for configuring and managing virtual resources.

Chapter 16. vSwitch and vRouter configuration commands

vSwitch and vRouter overview

A virtual switch (vSwitch) is a logical partition of the N2000 Series physical platform. A vSwitch is a software construction that allows the N2000 Series to operate as a number of independent switches. Different vSwitches can be created to support different user organizations, data partitions, and so forth. Each vSwitch contains one or more virtual router (vRouter), a set of virtual services (such as SSL termination), and independent policies. vRouters are independent IP routers. Each vRouter has its own set of IP interfaces (including Ethernet, LAG, and VLAN interfaces), route table, and access control lists (ACLs). vRouters isolate networks from each other since one vRouter will only exchange data with another if it is explicitly configured to do so.

There are two types of vSwitches—the system vSwitch and operator-defined vSwitches. The system vSwitch contains the resources used for overall management of the switch, and those resources that are shared by all vSwitches, including:

- The management vRouter for management access, and
- The shared vRouter for access to shared networks and the Internet

The management vRouter is resident on the system vSwitch, and isolates management traffic from data traffic in the system. The system software automatically creates the management vRouter. The software also automatically creates one shared vRouter (named *shared*). The system supports up to four additional vRouters, each one providing an Internet interface for the operator-defined vSwitches (and the Web servers connected to them).

Each operator-defined vSwitch functions as a separate entity, serving a backend Web server group or distinct customer site.

When you work with the configuration commands in this chapter, you can create, modify, delete, and display vSwitch and vRouter configurations. You cannot delete the system vSwitch or its resident management vRouter. As you create a vSwitch, the system automatically creates an associated vRouter, which it names *default*.

Command paths

The command names in this chapter show you how to execute the vSwitch commands from within the vSwitch and vRouter commands modes:

`vSwitch-`*`name`* `vRouter`

or

`switchServices vRouters`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

vSwitch and vRouter command summary

Table 16-1 lists and briefly describes the commands.

Table 16-1. vSwitch and vRouter command summary

Command name	Description
`show IP TCP`	Display the current TCP statistics for this vRouter instance.
`show IP TCP connections`	Display the current list of TCP connections for this vRouter instance.
`show IP UDP`	Display the current User Datagram Protocol (UDP) statistics for this vRouter instance.
`show IP UDP Listeners`	Display the current list of of User Datagram Protocol (UDP) listeners for this vRouter instance.
`show vRouter <name>`	Display vRouter configuration parameters.
`show vRouter (all)`	Display all vRouters within a vSwitch on the N2000 Series.
`show vSwitch <name>`	Display vSwitch configuration parameters.

Table 16-1. vSwitch and vRouter command summary (continued)

Command name	Description
`show vSwitch (all)`	Display all vSwitches on the N2000 Series system.
`vRouter <name>`	Enter vRouter configuration mode.
`vSwitch <name>`	Enter vSwitch configuration mode.

show IP TCP

Purpose

Displays the current TCP statistics for this vRouter instance.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name IP TCP
```

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch system vRouter management
sun(config-vSwitch-system vRouter-management)# show ip tcp
Rto Algorithm:               vanj
Rto Min:                     10
Rto Max:                     640
Max Connections:             -1
Active Opens:                0
Passive Opens:               18
Attempt Fails:               0
Established Resets:          0
Current Established:         3
In Segments:                 670
Out Segments:                581
Retransmitted Segments:      5
In Errors:                   0
Out Resets:                  0
No Ports:                    26
```

Output description

Field Name	Description
`Rto Algorithm`	The algorithm used to determine the time-out value for retransmitting unacknowledged octets.
`Rto Min`	The minimum value permitted by a TCP implementation for the re-transmission time-out, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the re-transmission time-out. In particular, when the time-out algorithm is rsre(3), an object of this type has the semantics of the LBOUND quantity described in RFC 793.
`Rto Max`	The maximum value permitted by a TCP implementation for the re-transmission time-out, measured in milliseconds. More refined semantics for objects of this type depend upon the algorithm used to determine the re-transmision time-out. In particular, when the time-out algorithm is rsre(3), an object of this type has the semantics of the UBOUND quantity described in RFC 793.
`Max Connections`	The total number of TCP connections the vRouter can support. In vRouters where the maximum number of connections is dynamic, this field should contain the value -1.
`Active Opens`	The number of times TCP connections have made a direct transition to the SYN-SENT state from the CLOSED state.
`Passive Opens`	The number of times TCP connections have made a direct transition to the SYN-RCVD state from the LISTEN state.
`Attempt Fails`	The number of times TCP connections have made a direct transition to the CLOSED state from either the SYN-SENT state or the SYN-RCVD state.
`Established Resets`	The number of times TCP connections have made a direct transition to the CLOSED state from either the ESTABLISHED state or the CLOSE-WAIT state.
`Current Established`	The number of TCP connections for which the current state is either ESTABLISHED or CLOSEWAIT.

Field Name	Description
In Segments	The total number of segments received, including those received in error. This count includes segments received on currently established connections.
Out Segments	The total number of segments sent, including those on current connections but excluding those containing only retransmitted octets.
Retransmitted Segments	The total number of segments retransmitted, that is, the number of TCP segments transmitted containing one or more previously transmitted octets.
In Errors	The total number of segments received in error, that is, bad TCP checksums.
Out Resets	The number of TCP segments sent containing the RST flag.
No Ports	The total number of inbound TCP connections for which there was no application at the destination port.

Associated MIB

`tcp.mib`

Web path

- vSwitch → name → vRouter → name → Ip → Tcp

show IP TCP connections

Purpose

Displays the current list of TCP connections for this vRouter instance.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `IP TCP connections`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch system vRouter management
sun(config-vSwitch-system vRouter-management)# show ip tcp connections
```

Local Address	Local Port	Remote Address	Remote Port	State
0.0.0.0	23	0.0.0.0	0	listen
0.0.0.0	80	0.0.0.0	0	listen
10.8.172.66	23	129.148.30.166	47691	timeWait
10.8.172.66	23	129.148.30.166	47692	established
10.8.172.66	80	10.8.172.67	1094	timeWait
10.8.172.66	80	10.8.172.67	1106	established
13.13.13.1	80	13.13.13.67	1095	timeWait
13.13.13.1	80	13.13.13.67	1107	established

Output description

Field name	Description	Filter name
`Local Address`	The local IP address for this TCP connection. In the case of a connection in the listen state that is willing to accept connections for any IP interface associated with the node, the value 0.0.0.0 is used.	`localAddress` *`ipaddress`*
`Local Port`	The local port number for this TCP connection.	`localPort` *`integer`*
`Remote Address`	The remote IP address for this TCP connection.	`remoteAddress` *`ipaddress`*
`Remote Port`	The remote port number for this TCP connection.	`remotePort` *`integer`*
`State`	The state of this TCP connection. The only value that may be set by a management station is deleteTCB(12). Accordingly, it is appropriate for an agent to return a “badValue” response if a management station attempts to set this object to any other value. If a management station sets this object to the value deleteTCB(12), then this has the effect of deleting the TCB (as defined in RFC 793) of the corresponding connection on the managed node, resulting in immediate termination of the connection. As an implementation-specific option, a RST segment may be sent from the managed node to the other TCP endpoint. (Note, however, that RST segments cannot be sent reliably.)	`State` *`text string`*

Associated MIB

`tcp.mib`

Web path

- vSwitch → name → vRouter → name → Ip → Tcp → Connections

show IP UDP

Purpose

Displays the current User Datagram Protocol (UDP) statistics for this vRouter instance.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `IP UDP`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch system vRouter management
sun(config-vSwitch-system vRouter-management)# show ip udp
In Datagrams:       69
No Ports:           628
In Errors:          0
Out Datagrams:      3842
```

Output description

Field Name	Description
`In Datagrams`	The total number of UDP datagrams delivered to UDP users.
`No Ports`	The total number of received UDP datagrams for which there was no application at the destination port.
`In Errors`	The number of received UDP datagrams that could not be delivered for reasons other than the lack of an application at the destination port.
`Out Datagrams`	The total number of UDP datagrams sent from this entity.

Associated MIB

`udp.mib`

Web path

- vSwitch → name → vRouter → name → Ip → Udp

show IP UDP Listeners

Purpose

Displays the current list of User Datagram Protocol (UDP) listeners for this vRouter instance.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `IP UDP listeners`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch system vRouter management
sun(config-vSwitch-system vRouter-management)# show ip udp listeners


Local Address          Local Port
---------------        -----------
0.0.0.0                69
0.0.0.0                161
0.0.0.0                520
0.0.0.0                1024
```

Output description

Field name	Description	Filter name
`Local Address`	The local IP address for this UDP listener. The value is 0.0.0.0 for the UDP listener that is willing to accept datagrams for any IP interface associated with the node.	`localAddress` *`ipaddress`*
`Local Port`	The local port for this UDP listener.	`localPort` *`integer`*

Associated MIB

`udp.mib`

Web path

- vSwitch → name → vRouter → name → Ip → Udp → Listeners

show vRouter <name>

Purpose

Displays the current configuration for the vRouter on the named vSwitch. The user-configurable parameters are set with the `vRouter <name>` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# show vRouter
ID:                 4
Name:               default
Description:        Default vRouter
Admin State:        enabled
Operational Status: up
```

Output description

Field name	Description	Filter name
ID	The system-assigned unique numeric identifier for the vRouter, a value between 0 and 255.	id *integer*
Name	The name of the vRouter.	name *name*
Description	The text description associated with the vRouter.	description *text*
Admin State	The administrative state of the named vRouter, either enabled or disabled. The default administrative state is enabled.	adminState {enabled | disabled}
Operational Status	The operational status of the vRouter. That is, whether it can receive and transmit frames. The vRouter is either: up: The administrative state is enabled, the vRouter is configured and available. paused: The vRouter has been temporarily disabled (not currently used). down: The administrative state is disabled and the vRouter is shut down. stopping: The vRouter is shutting down because it was deleted. syswait: The vRouter is in the process of initialization and is not completely up.	operationalStatus {up | paused | down | stopping | syswait}

Associated MIB

switchProvisioning.mib

Web path

- vSwitch → *name* → vRouter
- vSwitch → *name* → vRouter → *name*

show vRouter (all)

Purpose

Displays the current configuration for all vRouters on the N2000 Series system.

Access mode

user

Syntax

```
show switchServices vRouters
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# show vrouters
ID    vRouter      vSwitch      Description          State    Status
----- ----------  ------------ ------------------   -------- --------
0     management   system       System Management    enabled  up
                                vRouter
3     shared       system       Shared vRouter       enabled  up
4     default      e-commerce   Default vRouter      enabled  up
```

Output description

Field name	Description	Filter name
ID	The system-assigned unique numeric identifier for the vRouter, a value between 0 and 255.	vRouterId *integer*
vRouter	The name of the vRouter.	vRouterName *name*
vSwitch	The name of the vSwitch on which this vRouter is configured.	vSwitchName *name*
Description	The text description associated with the vRouter.	description *text*
State	The administrative state of the named vRouter, either enabled or disabled. The default administrative state is enabled.	adminState {enabled | disabled}

Field name	Description	Filter name
`Status`	The operational status of the vRouter. That is, whether it can receive and transmit frames. The vRouter is either: `up`: The administrative state is enabled, the vRouter is configured and available. `paused`: The vRouter has been temporarily disabled (not currently used). `down`: The administrative state is disabled and the vRouter is shut down. `stopping`: The vRouter is shutting down because it was deleted. `syswait`: The vRouter is in the process of initialization and is not completely up.	`operationalStatus {up | paused | down | stopping | syswait}`

Associated MIB

`switchProvisioning.mib`

Web path

- switchServices → vRouter

show vSwitch <name>

Purpose

Displays the current configuration for the vSwitch. The user-configurable parameters are set with the `vSwitch <name>` command.

Access mode

user

Syntax

```
show vSwitch
```

Sample output

```
sun> enable
sun# configure
sun(config)# show vswitch
Name:                e-commerce
ID:                  1
Description:         N/A
Admin State:         enabled
Operational Status:  up

Name:                e-commerce2
ID:                  3
Description:         N/A
Admin State:         enabled
Operational Status:  up

Name:                newSales
ID:                  2
Description:         N/A
Admin State:         enabled
Operational Status:  up

Name:                system
ID:                  0
Description:         System vSwitch
Admin State:         enabled
Operational Status:  up
```

Output description

Field name	Description	Filter name
`Name`	The name of the vSwitch.	`vSwitchName` *`name`*
`ID`	The system-assigned unique numeric identifier for the vSwitch, a value between 1 and 256.	`vSwitchId` *`integer`*
`Description`	The text description associated with the vSwitch.	`description` *`text`*
`Admin State`	The administrative state of the named vSwitch, either `enabled` or `disabled`. The default administrative state is `enabled`.	`adminState {enabled | disabled}`
`Operational Status`	The operational status of the vSwitch. That is, whether it can receive and transmit frames. The vSwitch is either: `starting`: The vSwitch and its associated applications are starting. `up`: The administrative state is enabled, the vSwitch is configured and available. `paused`: The vSwitch has been temporarily disabled (not currently used). `down`: The administrative state is disabled and the vSwitch is shut down. `stopping`: The vSwitch is shutting down because it was deleted.	`operationalStatus {up | paused | down | stopping}`

Associated MIB

`switchProvisioning.mib`

Web path

- vSwitch
- vSwitch → *name*

show vSwitch (all)

Purpose

Displays the current configuration for all vSwitches on the N2000 Series system.

Access mode

user

Syntax

```
show vSwitch
```

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# show vswitches
Name          ID    Description          Admin         Operational
                                         State         Status
e-commerce    1     E-commerce vSwitch    enabled       up
system        0     System vSwitch        enabled       up
```

Output description

Field name	Description	Filter name
`Name`	The name of the vSwitch.	`vSwitchName` *`name`*
`ID`	The system-assigned unique numeric identifier for the vSwitch, a value between 1 and 256.	`vSwitchId` *`integer`*
`Description`	The text description associated with the vSwitch.	`description` *`text`*
`Admin State`	The administrative state of the named vSwitch, either `enabled` or `disabled`. The default administrative state is `enabled`.	`adminState` {`enabled` | `disabled`}

Field name	Description	Filter name
`Operational Status`	The operational status of the vSwitch. That is, whether it can receive and transmit frames. The vSwitch is either: `up`: The administrative state is enabled, the vSwitch is configured and available. `paused`: The vSwitch has been temporarily disabled (not currently used). `down`: The administrative state is disabled and the vSwitch is shut down. `stopping`: The vSwitch is shutting down because it was deleted.	`operationalStatus {up | paused | down | stopping}`

Associated MIB

`switchProvisioning.mib`

Web path

- switchServices → vSwitches

vRouter <name>

Purpose

Enters virtual router configuration mode for the named vRouter. If the vRouter already exists, you can modify the configuration. If the vRouter does not already exist, the system creates it. A single vRouter named *default* is automatically added for each operator-defined vSwitch.

From vRouter configuration mode, you can execute commands and enter other command modes within the `vRouter` command mode. The commands you can enter are:

- `interfaces`
- `ip`
- `vlan`
- `vrrp`
- `ping`
- `traceroute`

The command modes you can enter are:

- `ip`
- `irdp`
- `rip`
- `vlan`

Access mode

config

Syntax

```
vSwitch-name vRouter
   name name
   [description text]
   [adminState {enabled | disabled}]
```

Arguments

Argument	Description
`name` *`name`*	Specifies the name of the vRouter. This can be either an existing vRouter or a new name, which causes the system to create a vRouter with that name. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`description` *`text`*	**Optional.** Enters a text description to associate with the vRouter. The description can be up to 64 characters, and is displayed with the `show vRouter` command. If the description includes spaces, enclose it within quotation marks.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named vRouter, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring it offline or preconfigure a vRouter before bringing it online. The default administrative state is `enabled`.

Delete filters

```
no vSwitch-name vRouter
   name name
   [id integer]
   [description text]
   [adminState {enabled | disabled}]
```

Example

The following example creates an additional shared vRouter in the system vSwitch. Before creating the vRouter, the system prompts you to confirm whether you want to continue the operation. Enter y and press [Return] to create the vRouter. Otherwise, enter n and press [Return] to cancel the operation.

```
sun> enable
sun# configure
sun(config)# vswitch system
sun(config-vSwitch-system)# vRouter sharedIspNetwork

create new vRouter "sharedIspNetwork" ? (y or n): y
.....................................

sun(vSwitch-system vRouter-sharedIspNetwork)#
```

Associated MIB

```
switchProvisioning.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → add
- vSwitch → *name* → vRouter → *name* → copy
- vSwitch → *name* → vRouter → *name* → modify
- vSwitch → *name* → vRouter → *name* → delete

vSwitch <name>

Purpose

Enters virtual switch configuration mode for the named vSwitch. Also enters the specified `vSwitch` command mode.

If the vSwitch already exists, you can modify the configuration. If the vSwitch does not already exist, the system creates it. You can create up to five virtual switches per N2000 Series. vSwitches that you create are called *operator-defined vSwitches*.

From `vSwitch` configuration mode, you can execute commands and enter other command modes within the `vSwitch` command mode. The commands you can enter are:

- `vRouter`

The command modes you can enter are:

- `ckm`
- `event`
- `loadbalance`
- `vRouter`
- `resource`

Access mode

config

Syntax

```
vSwitch
   vSwitchName name
   [description text]
   [adminState {enabled | disabled}]
```

Arguments

Argument name	Description
`vSwitchName` *name*	Specifies the name of the vSwitch. This can be either an existing vSwitch or a new name, which causes the system to create a vSwitch with that name. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`description` *text*	**Optional.** Enters a text description to associate with the vSwitch. The description can be up to 64 characters, and is displayed with the `show vswitch` command. If the description includes spaces, enclose it within quotation marks.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named vSwitch, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring it offline or preconfigure a vSwitch before bringing it online. The default administrative state is `enabled`.

Delete filters

See the `show vSwitch <name>` command for additional argument descriptions.

```
no vSwitch
   vSwitchName name
   [vSwitchId integer]
   [description text]
   [adminState {enabled | disabled}]
```

Example

The following example shows how to create an operator-defined vSwitch. The system prompts you to confirm that you want to create the vSwitch. Enter y and press [Return] to create the vSwitch. Otherwise, enter n and press [Return] to cancel the operation.

```
sun> enable
sun# configure
sun(config)# vSwitch vSwitchName engineering description Software
adminState enabled

create new vSwitch "engineering" ? (y or n): y
.....................................................................

sun(config-vSwitch-engineering)#
```

Associated MIB

```
switchProvisioning.mib
```

Web path

- vSwitch → *name* → add
- vSwitch → *name* → copy
- vSwitch → *name* → modify
- vSwitch → *name* → delete

Chapter 17. Resource commands

Resource command description

The `resource` commands allow you to configure traffic policing attributes for vSwitch ports and allocate percentages of a function card's resources to specific operator-defined vSwitches. Use the `vSwitch` *`name`* `resource` commands to configure and view resources assigned to a specific vSwitch. Use the `switchServices resource` commands to view resources for all vSwitches.

Traffic policing overview

Traffic policing allows you to manage the rate of traffic received on a vSwitch port associated with an IP, VLAN, or LAG interface. When the data rate exceeds the maximum rate configured on the system, it drops data frames. By default, all port interfaces are set to use the full port bandwidth. By observing the number of forwarded and discarded frames, you can determine where you need to restrict or increase bandwidth for a specific port.

Note: The N2000 Series does not change Quality of Service (QoS) or Differentiated Services Code Point (DSCP) markers in the data that it receives and transmits.

Use the `vSwitch` *`name`* `resource portBandwidth` command to configure traffic policing attributes.

Service bandwidth overview

The `serviceBandwith` command allows you to allocate a percentage of a function card's resources to one or more operator-defined vSwitches. Allocating different amounts of the function card resources allows you to balance the needs of heavily used vSwitches with those that have lower processing needs.

You can specify the minimum percentage of the function card resources that the system provides to the vSwitch. The vSwitch will always be able to access this amount of the function card resources. If the vSwitch needs additional resources and the function card has unallocated resources available, the vSwitch will be able to access these resources.

You can also specify the absolute maximum percentage of the function card resources that the system provides to the vSwitch.

Resource command path

The command names in this chapter show you how to execute the commands from within the following command modes:

```
vSwitch name resource
switchServices resource
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Resource command summary

Table 17-1 lists and briefly describes the resource commands.

Table 17-1. Resource command summary

Command name	Description
`portBandwidth`	Configure the average and maximum bandwidth and burst rates for a port.
`serviceBandwidth`	Configure the percentage of function card resources for a vSwitch.

Table 17-1. Resource command summary (continued)

Command name	Description
`show vSwitch resource portBandwidth (specific vSwitch)`	Display the bandwidth configuration for ports in a specific vSwitch.
`show switchServices resource portBandwidth (all vSwitches)`	Display the bandwidth configuration for ports in all of the vSwitches on the system.
`show serviceBandwidth (specific vSwitch)`	Display resource allocation of a function card resource allocation for a specific vSwitch.
`show serviceBandwidth (all vSwitches)`	Display resource allocation of a function card resource allocation for all vSwitches on the system.

portBandwidth

Purpose

Configures traffic policing for ingress traffic on specific ports in a vSwitch. Use the `show switchServices resource portBandwidth (all vSwitches)` command or the `show switchServices resource portBandwidth (all vSwitches)` command to view the current frame forwarding and discard rates. The system automatically generates port bandwidth records for configured interfaces and allows 100% bandwidth. You can then limit traffic on the ports by modifying the records.

The no form of the command deletes a port bandwidth configuration. If you enter optional arguments, the CLI deletes the configuration only if it matches all arguments. With the no form of the command, the `ifIndex` argument is the only required argument.

Access mode

enable

Syntax

To create a traffic policing configuration:

```
vSwitch-name resource portBandwidth
   ifName ifName
   bandwidthAllocation percentage
   bandwidthMaximum percentage
   burstSize integer
   burstSizeMaximum integer
```

To modify a traffic policing configuration:

```
vSwitch-name resource portBandwidth
   ifName ifName
   [bandwidthAllocation percentage]
   [bandwidthMaximum percentage]
   [burstSize integer]
   [burstSizeMaximum integer]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the Ethernet, VLAN or LAG interface for which you want to change traffic policing attributes.
`bandwidthAllocation` *`percentage`*	Specifies the percentage of the bandwidth that the system uses as the committed bandwidth rate for this port. Valid values are 1 through 100 percent. The default setting is 100%.
`bandwidthMaximum` *`percentage`*	Specifies the percentage of the bandwidth that the system uses as the maximum bandwidth rate for this port. Valid values are 1 through 100 percent. The default setting is 100%.
`burstSize` *`integer`*	Configures the committed burst data transfer size in bytes. This value represents the average size of a single burst of traffic. Valid values are 1 through 65535; the default setting is 65534. The system forwards all frames that conform to this value.
`burstSizeMaximum` *`integer`*	Configures the maximum burst size for traffic. This value represents the maximum size of a single burst of traffic allowed on the port. Valid values are 1 through 65535; the default setting is 65535. The system drops any frames that exceed this value.

Delete filters

See the `show vSwitch resource portBandwidth (specific vSwitch)` command for argument descriptions.

```
no vSwitch-name resource portBandwidth
   ifName ifName
   [bandwidthAllocation percentage]
   [bandwidthMaximum percentage]
   [burstSize integer]
   [burstSizeMaximum integer]
   [discard integer]
   [forward integer]
```

Example

```
sun> enable
sun# vSwitch e-commerce
sun(vSwitch-e-commerce)# resource
sun(...resource)# portBandwidth ifName eth.1.5 bandwidthAllocation 50
bandwidthMaximum 75 burstSize 65534 burstSizeMaximum 65535
```

Associated MIB

```
switchProvisioning.mib
```

Web path

- vSwitch → *name* → resource → portBandwidth → add
- vSwitch → *name* → resource → portBandwidth → copy
- vSwitch → *name* → resource → portBandwidth → modify

serviceBandwidth

Purpose

Configures the percentage of the resources that a function card provides to a specific vSwitch. This command is available for operator-defined vSwitches only.

The no form of the command deletes a service bandwidth configuration. If you enter optional arguments, the CLI deletes the configuration only if it matches all arguments. With the no form of the command, the `card` argument is the only required argument.

Access mode

enable

Syntax

```
[no] vSwitch-name resource serviceBandwidth
   card {functionCard1 | functionCard2}
   [guaranteedMinPercent percentage]
   [absoluteMaxPercent percentage]
```

Arguments

Argument name	Description
`card {functionCard1 | functionCard2}`	Specifies the function card whose resources you want to allocate to the vSwitch. Valid values are: • `functionCard1`: The function card installed in the front of the chassis. • `functionCard2`: The function card installed in the back of the chassis. This is not currently used.
`guaranteedMinPercent` *`percentage`*	**Optional**. Configures the percentage of a function card's resources that you want to assign to the vSwitch. Valid values are from 1 through 100 percent. The default value is 1 percent.
`absoluteMaxPercent` *`percentage`*	**Optional**. If specified, absoluteMaxPercent >= guaranteedMinPercent. The default value is the minimum of 2*guaranteedMinPercent and 100. If absoluteMaxPercent is greater then guaranteedMinPercent, this vswitch is permitted to allocate resources (if available) from a "shared" pool when demand is greater than what can be supported by guaranteedMinPercent resources.

Example

This example shows how to configure the system to allocate 50 percent of function card 1's resources to the e-commerce vSwitch.

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# resource
sun(...vSwitch-e-commerce resource)# serviceBandwidth card
functionCard 1 guaranteedMinPercent 50
sun(...vSwitch-e-commerce resource)#
```

Associated MIB

`switchProvisioning.mib`

Web path

- vSwitch → *name* → resource → serviceBandwidth → add
- vSwitch → *name* → resource → serviceBandwidth → copy
- vSwitch → *name* → resource → serviceBandwidth → modify

show vSwitch resource portBandwidth (specific vSwitch)

Purpose

Displays the current bandwidth for ports associated with a specific vSwitch. This command also displays operational statistics for the number of frames forwarded and discarded.

To view the bandwidth for all vSwitch ports, use the `show switchServices resource portBandwidth (all vSwitches)` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `resource portBandwidth`

Sample output

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# resource
sun(vSwitch-e-commerce resource)# show portbandwidth
                Max   Max         Discarded  Forwarded  Auto
IfName     Bw % Bw % Burst Burst Frames     Frames     Gen
eth.1.20   100  100  65534 65535 0          0          no
```

Output description

Field name	Description	Filter name
IfName	Specifies the Ethernet, VLAN or LAG interface for which you want to change traffic policing attributes.	ifName *ifName*
Bw %	The percentage of the bandwidth that the system uses as the committed bandwidth allocated to the port.	bandwidthAllocation *percentage*

Field name	Description	Filter name
`Max Bw %`	The percentage of the bandwidth that the system uses as the maximum bandwidth to the port.	`bandwidthMaximum` *`percentage`*
`Burst`	The committed size of a single burst of traffic.	`burstSize` *`integer`*
`Max Burst`	The maximum size of a single burst of traffic.	`burstSizeMaximum` *`integer`*
`Discarded Frames`	The number of frames that exceeded the configured maximum bandwidth and burst size; the system drops these frames.	`discard` *`integer`*
`Forwarded Frames`	The number of frames that were less than or equal to the configured bandwidth or maximum bandwidth and burst size. The system forwards these frames to the network.	`forward` *`integer`*
`Auto Gen`	Specifies whether the record was automatically created by the system or by the user.	`autoGenerated` *`no`*`|`*`yes`*

Associated MIB

`switchProvisioning.mib`

Web path

- vSwitch → *name* → resource → portBandwidth

show switchServices resource portBandwidth (all vSwitches)

Purpose

Displays the current bandwidth for ports associated with all of the vSwitches on the system. This command also displays operational statistics for the number of frames forwarded and discarded.

To view the bandwidth for the ports in a specific vSwitch only, use the `show vSwitch resource portBandwidth (specific vSwitch)` command.

Access mode

user

Syntax

```
show switchServices resource portBandwidth
```

Sample output

```
sun> show switchServices resource portbandwidth
                          Max       Max    Discarded Forwarded Auto
vSwitch         IfName    Bw %  Bw %  Burst  Burst  Frames    Frames     Gen
system          eth.1.12  100   100   65534  65535  0           0        yes
elmo            eth.1.10  100   100   65534  65535  0           0        yes
elmo            eth.1.12  100   100   65534  65535  0           0        yes
e-commerce      eth.1.12  100   100   65534  65535  0           0        yes
e-commerce      eth.1.30  100   100   65534  65535  0           0        yes
e-commerce      eth.1.31  100   100   65534  65535  0           0        yes
e-commerce      eth.1.32  100   100   65534  65535  0           0        yes
sun(config-switchServices resource)#
```

Output description

Field Name	Description	Filter Name
vSwitch	The name of a vSwitch associated with the configuration. The vSwitch name is associated with a numeric identifier (ID). To find the vSwitch ID, use the `show vSwitch` command.	`vSwitchId` *`integer`*
IfName	The interface associated with the port.	`ifName` *`ifName`*
Bw %	The percentage of the bandwidth that the system uses as the committed bandwidth allocated to the port.	`bandwidthAllocation` *`percentage`*
Max Bw %	The percentage of the bandwith that the system uses as the maximum bandwidth to the port.	`bandwidthMaximum` *`percentage`*
Burst	The committed size of a single burst of traffic.	`burstSize` *`integer`*
Max Burst	The maximum size of a single burst of traffic.	`burstSizeMaximum` *`integer`*
Discarded Frames	The number of frames that exceeded the configured maximum bandwidth and burst size; the system drops these frames.	`discard` *`integer`*
Forwarded Frames	The number of frames that were less than or equal to the configured bandwidth or maximum bandwidth and burst size. The system forwards these frames to the network.	`forward` *`integer`*
`Auto Gen`	Specifies whether the record was automatically created by the system or by the user.	`autoGenerated` *`no`*`|`*`yes`*

Associated MIB

`switchProvisioning.mib`

Web path

- vSwitch → *name* → resource → portBandwidth

show serviceBandwidth (specific vSwitch)

Purpose

Displays `serviceBandwidth` configurations for an operator-defined vSwitch. The service bandwidth is the amount of function card resources allocated to a vSwitch.

To view the service bandwidth for all vSwitches, use the `show switchServices resource serviceBandwidth` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `resource serviceBandwidth`

Sample output

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# resource
sun(vSwitch-e-commerce resource)# show serviceBandwidth
Card           Min Svc Percent      Max Svc Percent

functionCard1  50                   100
functionCard2  25                  50
```

Output description

Field name	Description	Filter name
`Card`	The function card installed in the system. `functionCard1` is the function card installed in the front of the chassis. `functionCard2` is the function card installed in the back of the chassis. This is not currently used.	`Card {functionCard1 | functionCard2}`

Field name	Description	Filter name
`Min Svc Percent`	The percentage of the resources for the function card guaranteed to be allocated to the vSwitch.	`guaranteedMinPercent` *`percentage`*
`Max Svc Percent`	If absoluteMaxPercent is greater than guaranteedMinPercent, this vswitch is permitted to allocate resources (if available) from a “shared” pool when demand is greater than what can be supported by guaranteedMinPercent resources.	`absoluteMaxPercent` *`percentage`*

Associated MIB

`switchProvisioning.mib`

Web path

- vSwitch → *name* → Resource → serviceBandwidth

show serviceBandwidth (all vSwitches)

Purpose

Displays `serviceBandwidth` configurations for all operator-defined vSwitches. The service bandwidth is the amount of function card resources allocated to a vSwitch.

To view the service bandwidth for a specific vSwitch, enter the `show vSwitch-name resource serviceBandwidth` command.

Access mode

user

Syntax

```
show switchServices resource serviceBandwidth
```

Sample output

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# resource
sun(config-vSwitch-e-commerce resource)# show serviceBandwidth
vSwitch      Card           Min Svc Percent      Max Svc Percent
e-commerce   functionCard1  50                   100
marketing    functionCard2 25                   50
```

Output description

Field name	Description	Filter name
`vSwitch`	The name of the vSwitch associated with the configuration. The vSwitch name is associated with a numeric identifier (ID). To display the vSwitch ID, use the `show vSwitch` command.	`vSwitchID` *integer*

Field name	Description	Filter name
`Card`	The function card installed in the system. `functionCard1` is the function card installed in the front of the chassis. `functionCard2` is the function card installed in the back of the chassis.	`Card {functionCard1 | functionCard2}`
`Min Svc Percent`	The percentage of the resources for the function card guaranteed to be allocated to the vSwitch.	`guaranteedMinPercent` *`percentage`*
`Max Svc Percent`	If absoluteMaxPercent is greater than guaranteedMinPercent, this vswitch is permitted to allocate resources (if available) from a “shared” pool when demand is greater than what can be supported by guaranteedMinPercent resources.	`absoluteMaxPercent` *`percentage`*

Associated MIB

`switchProvisioning.mib`

Web path

- vSwitch → *name* → resource → serviceBandwidth

Part V. Interface configuration

The chapters in Part V describe the commands for configuring interfaces on the system.

Chapter 18. Ethernet management port commands

Ethernet management port description

The Ethernet management port on the N2000 Series system allows you to gain remote access by using Telnet to access the command-line interface (CLI). You can also use the Ethernet management port for HTTP, HTTPS, SSH, or SNMP access. Typically, the Ethernet management port operates best with the default settings, but there is a configuration command available to change settings as necessary.

EthMgmt command path

The commands in this chapter are executed from the top level of the CLI hierarchy.

Ethernet management port command summary

Table 18-1 lists and briefly describes the ethMgmt commands.

Table 18-1. ethMgmt command summary

Command name	Description
`ethMgmt`	Configure characteristics of the Ethernet management port.
`show ethMgmt`	Display configured characteristics of the Ethernet management port.

Ethernet management port basic configuration

Because the Ethernet management port on the N2000 Series system comes preconfigured, there is no configuration required. All configuration changes are optional. For configurable parameters, see the `ethMgmt` command.

ethMgmt

Purpose

Sets the Ethernet management port speed and duplex characteristics. Through the management port you can use Telnet or SSH to access the CLI, HTTP/HTTPS to access the Web interface, or SNMP.

In most cases, the management port operates best with the default settings. These defaults configure the `physpeed` setting as autonegotiate, with advertised speeds of 10M or 100M at half or full duplex. You can specify speed and duplex settings by configuring them with the `physpeed` and `phyduplex` arguments.

You cannot delete the Ethernet management interface.

Access mode

config

Syntax

```
ethMgmt
   [physpeed {10M | 100M | auto}]
   [phyduplex {halfDuplex | fullDuplex}]
   [adminMac macAddress]
```

Arguments

Argument name	Description
`physpeed {10M | 100M | auto}`	**Optional.** Sets the acceptable port speed(s), in Mbps. If set to `auto`, the system negotiates with the remote client to achieve the best common speed and duplex settings (based on the advertised speed and duplex settings).Valid values are `10M`, `100M`, and `auto`; the default is `auto`.
`phyduplex {halfDuplex | fullDuplex}`	**Optional.** Sets the acceptable duplex method(s) for the port, either half (asynchronous) or full (simultaneous) transmission. If `physpeed` is set to `auto`, you do not need to set this. The default setting is `halfDuplex`.

Argument name	Description
adminMac *macAddress*	**Optional.** Specifies the physical MAC address of the port, overriding the hardware-assigned MAC address. Enter a valid address in hex format. Because the port uses this MAC assignment even if it conflicts with other MAC addresses, be very careful before reassigning. If you want to change an administrative MAC address back to the factory preset, enter a value of 00:00:00:00:00:00.

Example

The following example first displays the management port settings, then reconfigures the port from half-duplex to full-duplex operations.

```
sun(config)# show ethMgmt
IfName:                ethMgmt.1
IfIndex:               0x71000000
Port Type:             fastEthernet
Admin MAC:             00:00:00:00:00:00
Oper MAC:              00:07:82:00:03:c0
Port Speed:            auto
Port Duplex:           halfDuplex
Operational Status:    up
Autonegotiated Status: complete
Autonegotiated Duplex: fullDuplex
Autonegotiated Speed:  100M
Link Status:           linkPass
Total RX Frames:       1603895
Total TX Frames:       2020985

sun(config)# ethmgmt physpeed 100 phyduplex full
sun(config)# show ethmgmt
IfName:                ethMgmt.1
IfIndex:               0x71000000
Port Type:             fastEthernet
Admin MAC:             00:00:00:00:00:00
Oper MAC:              00:07:82:00:03:c0
Port Speed:            100M
Port Duplex:           fullDuplex
Operational Status:    up
Autonegotiated Status: N/A
Autonegotiated Duplex: N/A
Autonegotiated Speed:  N/A
```

```
Link Status:            linkPass
Total RX Frames:        1605578
Total TX Frames:        2023224
```

Associated MIB

`ether.mib`

Web path

- EthMgmt → ethMgmt list → modify
- Port → port list → modify

show ethMgmt

Purpose

Displays the Ethernet management port's speed and duplex configured and negotiated settings, as well as link status information. You cannot filter on the fields of this command.

Access mode

user

Syntax

```
show ethMgmt
```

Sample output

```
sun> show ethMgmt
IfName:                 ethMgmt.1
IfIndex:                0x71000000
Port Type:              fastEthernet
Admin MAC:              00:00:00:00:00:00
Oper MAC:               00:07:82:00:03:c0
Port Speed:             auto
Port Duplex:            halfDuplex
Operational Status:     up
Autonegotiated Status: complete
Autonegotiated Duplex: fullDuplex
Autonegotiated Speed:  100M
Link Status:            linkPass
Total RX Frames:        1603895
Total TX Frames:        2020985
```

Output description

Field name	Description
IfName	The name of the interface, always ethMgmt.1.
IfIndex	The system-assigned port identifier, in hexadecimal format.
Port Type	The port's media type, 10M/100M Ethernet (fastEthernet).

Field name	Description
`Admin MAC`	The user-assigned MAC address, which overrides the factory preset address. If you have not assigned an address, the field displays 00:00:00:00:00:00.
`Oper MAC`	The MAC address the port is using, either the factory preset address or the administratively assigned address.
`Port Speed`	The port speed setting. The `auto` setting configures the system to negotiate with the remote client to achieve the best common speed and duplex settings. A value of `10M` or `100M` indicates a specific speed.
`Port Duplex`	The port's duplex setting. A value of `halfDuplex` indicates asynchronous transmission; a value of `fullDuplex` indicates simultaneous transmission. If port speed is `auto`, this setting is irrelevant.
`Operational Status`	The operational status of the port. The port is either configured and available (`up`) or not (`down`).
`Autonegotiated Status`	The status of the autonegotiation, either `complete`, `inProgress`, or `disabled` if autonegotiation was not enabled with the `physpeed` argument of the `ethMgmt` command.
`Autonegotiated Duplex`	The negotiated duplex setting. If port speed was not set to `auto` with the `ethMgmt` command, the field is not applicable. Verify the port speed setting in the `Port Speed` field, above. If the autonegotiated status is `inProgress` or `disabled`, the field displays `unknown`.
`Autonegotiated Speed`	The negotiated speed setting. If port speed was not set to `auto` with the `ethMgmt` command, the field is not applicable. Verify the port speed setting in the `Port Speed` field, above. If the autonegotiated status is `inProgress` or `disabled`, the field displays `N/A`.
`Link Status`	The status of the connection to the remote device, either `linkFail` or `linkPass`.
`Total RX Frames`	The number of frames received over the management port.
`Total TX Frames`	The number of frames transmitted over the management port.

Associated MIB

`ether.mib`

Web path

- EthMgmt → ethMgmt list
- Port → port list

Chapter 19. Ethernet data port commands

N2000 Series physical port description

The N2000 Series systems come with preconfigured media ports through which you connect to other equipment and networks. The system boots with a default configuration for each port. The commands in this chapter configure, manage, and display physical parameters of the Ethernet ports on the N2000 Series system.

The number and type of ports available depend on the N2000 Series model you are using. The N2040 system provides 4 Gigabit Ethernet ports and 40 Fast Ethernet ports. The N2120 system provides 12 Gigabit Ethernet ports. Ethernet port numbers are entered in the format *eth.slot.portNumber*. Since the N2000 Series modules are single slot, the slot is always 1.

The table below summarizes the port information for the two systems.

Table 19-1. Available system ports

Product name	Slots	Total gigabit Ethernet ports	System-named Gigabit Ethernet ports	Total Fast Ethernet ports	System-named Fast Ethernet ports
N2040	1	4	`eth.1.1` through `eth.1.4`	40	`eth.1.5` through `eth.1.44`
N2120	1	12	`eth.1.1` through `eth.1.12`	0	N/A

Port command path

The commands in this chapter are executed from the top level of the CLI hierarchy.

Port command summary

Table 19-2 lists and briefly describes the port commands.

Table 19-2. Port command summary

Command name	Description
`port`	Configure settings for the specified port.
`port mirror`	Configure a port mirroring source/observation port pairing.
`show port`	Display port settings.
`show port ipStatistics`	Display IP-specific 64-bit counters for IP interfaces.
`show port mirror`	Display the current port mirroring configuration.
`show port mirror availability`	Display the ports that are available as port mirror source ports.
`show port statsMIB`	Display all statistics for the specified port(s), based on RFC 2665, *Definitions of Managed Objects for the Ethernet-like Interface Types*.
`show port statsRX`	Display extended receive statistics for the specified port(s).
`show port statsTX`	Display extended transmit statistics for the specified port(s).
`show port verbose`	Display extended port settings.

Basic port configuration

Because all ports on the N2000 Series systems come preconfigured, there is typically no configuration required. All configuration changes are optional. For configurable parameters, see the `port` command.

port

Purpose

Configures characteristics of the specified Gigabit Ethernet or Fast Ethernet port on the N2000 Series system. The system comes with all ports preconfigured and prenamed. By executing this command, you are reconfiguring the specified port. You cannot delete a port from the system.

If you add a port to a link aggregation group (LAG), it inherits the administrative Media Access Control (MAC) address, default virtual LAN (VLAN), and jumbo frames settings of that LAG. The port settings you have configured with this command are saved, but are not used until you remove the port from the LAG. Similarly, if you change those settings for a port that is already in a LAG, they do not become active unless you remove the port from the LAG.

Access mode

enable

Syntax

```
port
    ifName ifName
    [phyMode {normal | loopback}]
    [phySpeed {10M | 100M | 1000M | auto}]
    [phyDuplex {halfDuplex | fullDuplex}]
    [jumboFrames {enabled | disabled}]
    [advSpeed {10M | 100M | both}]
    [advDuplex {halfDuplex | fullDuplex | both}]
    [defVlan integer]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the name of the port for which you want to modify characteristics. Interface names are entered in the format *`eth.slot.portNumber`*.

Argument name	Description
`phyMode {normal | loopback}`	**Optional.** Configures the specified port in either normal or loopback mode. The `normal` setting configures the port to pass traffic as a normal physical port. If set to `loopback`, the port operates as if a loopback cable were attached. That is, anything you transmit is received over the same port. This setting is used, typically, only for testing. Gigabit Ethernet ports cannot be set to `loopback`. The default is `normal`.
`phySpeed {10M | 100M | 1000M | auto}`	**Optional.** Sets the acceptable port speed(s), in Mbps. If set to `auto`, the system negotiates with the remote client to achieve the best common speed and duplex settings (based on the advertised speed and duplex settings). Valid values are `10M`, `100M`, `1000M`,and `auto`; the default is `auto`.
`phyDuplex {halfDuplex | fullDuplex}`	**Optional.** Sets the acceptable duplex method(s) for the port, either half (asynchronous) or full (simultaneous) transmission. If `physpeed` is set to `auto`, you do not need to set this. The default setting is `halfDuplex`.
`jumboFrames {enabled | disabled}`	**Optional.** Configures the port to accept or reject jumbo frames. (Jumbo frames extend the traditional 1500 byte Ethernet frame size to 9018 bytes for 100M and Gigabit Ethernet.) The default setting is `disabled`.
`advSpeed {10M | 100M | both}`	**Optional.** Configures the speed(s) that the port advertises to any remote devices if `physpeed` is set to `auto`. The default setting is `both`, which advertises both 10M and 100M.
`advDuplex {halfDuplex | fullDuplex | both}`	**Optional.** Configures the duplex setting(s) that the port advertises to the remote device if `physpeed` is set to `auto`. The default setting is `both`, which advertises both `halfDuplex` and `fullDuplex`.
`defVlan` *`vlanId`*	**Optional.** Sets the default VLAN ID if the port is part of a VLAN. Any packet arriving on the port that does not have a VLAN ID in its extended MAC header is assigned this ID. Valid values are 1 through 4095. The default is ID 4095, which is the discard VLAN. This means that if the port is part of a VLAN, by default, untagged frames arriving on this port will be discarded. If you do not want the traffic to be dropped, assign a different VLAN ID.

Example

The following example shows how to enable jumbo frames and set a default VLAN for a Gigabit Ethernet port.

```
sun> enable
sun# config
sun(config)# port eth.1.4 jumboFrames enabled defVlan 10
```

Associated MIB

`ether.mib`

Web path

- Port → port list → modify

port mirror

Purpose

Configures a mirror port that reflects N2000 Series network traffic from a source port to an observation port connected to network analyzer equipment. You can selectively monitor inbound traffic, outbound traffic, or bidirectional source port traffic without physically interfering with the actual network traffic data stream.

The system supports one mirror port per network processor (NP), where the NP controls a certain number of system ports. Depending on your system hardware, you can configure either two or three mirror ports per switch, as follows:

On the N2040 model

- NP1 supports ports 1, 2, 5 to 22 (that is, only one of these ports can be mirrored at a time)
- NP2 supports ports 3, 4 23 to 44

On the N2120 model

- NP1 supports ports 1 to 4
- NP2 supports ports 5 to 8
- NP3 supports ports 9 to 12

Note that observing ports cannot be used as IP or VLAN interfaces, nor can they be members of a LAG.

Access mode

config

Syntax

```
port ifName ifName mirror
   observerPort ifName
    [adminState {enabled | disabled}]
    [portMirrorDir {in | out | both}]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the name of the source port for which traffic is to be mirrored to an observation port. Interface names are entered in the format *`eth.slot.portNumber`*.
`observerPort` *`ifName`*	Specifies the name of the port to be configured as a mirror port where you are observing network traffic on an active N2000 Series port. Interface names are entered in the format *`eth.slot.portNumber`*.
`portMirrorDir {in | out| both}`	**Optional.** Sets the type of source port traffic to be mirrored on the observation port; inbound, outbound, or both. If set to `both`, the system mirrors bidirectional traffic to the observation port. Valid values are `in`, `out`, and `both`; the default is `both`.
`adminState {enabled | disabled}`	**Optional.** Enables or disables mirroring on the specified source port. The default is `enabled`.

Example

The following example mirrors bidirectional source traffic from Ethernet port eth.1.2 to the observation port eth.1.33

```
sun> enable
sun# config
sun(config)# port eth.1.2 mirror eth.1.3 enabled both
```

Associated MIB

`ether.mib`

Web path

- Port → mirror

show port

Purpose

Displays characteristics of all configured ports or a single port. The system comes with all ports preconfigured and prenamed. This command displays the current operational settings that are either default settings or configured with the `port` command. The number of ports displayed depends on the model. See Table 19-1, "Available system ports," on page 19-1 for port numbering information.

If a port is a member of a LAG, it inherits the administrative MAC address, default VLAN, and jumbo frames setting of that LAG. The settings you have configured with the `port` command are saved, but are not used unless you remove the port from the LAG. This command displays both the configured settings and, if the port is a LAG member, the operational settings.

Access mode

user

Syntax

```
show port
```

Sample output

```
sun(config)# show port
IfName     Port Type     Port Mode   Default Vlan  Oper Status   Link
Status

eth.1.1    gigEthernet   normal      4095          down          linkFail
eth.1.2    gigEthernet   normal      4095          down          linkFail
eth.1.3    gigEthernet   normal      4095          down          linkFail
eth.1.4    gigEthernet   normal      4095          down          linkFail
eth.1.5    fastEthernet  normal      4095          down          linkFail
eth.1.6    fastEthernet  normal      4095          down          linkFail
eth.1.7    fastEthernet  normal      4095          down          linkFail
eth.1.8    fastEthernet  normal      4095          down          linkFail
eth.1.9    fastEthernet  normal      4095          down          linkFail
eth.1.10   fastEthernet  normal      4095          up            linkPass
eth.1.11   fastEthernet  normal      4095          down          linkFail
eth.1.12   fastEthernet  normal      4095          up            linkPass
eth.1.13   fastEthernet  normal      4095          down          linkFail
```

```
sun(config)# show port eth.1.1
IfName     Port Type     Port Mode Default Vlan Oper Status Link Status
eth.1.1    gigEthernet normal      4095         down          linkFail
```

Output description

Field name	Description	Filter name
`IfName`	The system-assigned port name, in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`Port Type`	The port's media type, either gigabit Ethernet (`gigEthernet` or `copperGig`), 10M/100M Ethernet (`fastEthernet`).	`portType {gigEthernet | fastEthernet | copperGig}`
`Port Mode`	The port's physical mode, either `normal` or `loopback`. If `normal`, the port passes traffic as a normal physical port. If `loopback`, the port receives back the packets it transmits (for testing).	`phymode {normal | loopback}`
`Default Vlan`	The ID number of the VLAN to which untagged packets are forwarded. The default is ID 4095, which is the discard VLAN. That is, traffic assigned to this VLAN is dropped. If the port is a member of a LAG, this setting is overridden by the LAG's default VLAN setting.	`defVlan` *`vlanId`*
`Operational Status`	The operational status of the port, based on the port configuration and link status. This status reflects whether the port is configured under a VLAN or a virtual router *and* the link is operational. When the port is configured *and* the link is operational, the operational status is `up`; the port can begin to receive and transmit frames. In all other cases, the operational status is `down`.	`operStatus {down | up}`
`Link Status`	The status of the connection to the remote device, either `linkFail` or `linkPass`.	`linkStatus {linkFail | linkPass}`

Associated MIB

`ether.mib`

Web path

- Port → port list

show port ipStatistics

Purpose

Displays the IP-specific 64-bit counters (statistics) for ports that are configured as IP interfaces. You can display all or specified ports. All counters are cleared when you reboot the system. Counters begin incrementing for a port when you activate the port as an IP interface.

Access mode

user

Syntax

```
show port ipStatistics
```

Sample output

```
sun> port eth.1.12
sun(port-eth.1.12)# show ipStatistics
IfName:              eth.1.12
In Octets:           92452
In Unicast Pkts:     693
In Multicast Pkts:   0
Out Octets:          0
Out Unicast Pkts:    0
Out Multicast Pkts: 0
```

Output Description

Field name	Description	Filter name
IfName	The name of the port for which you are viewing statistics. Interface names are entered in the format *type.slot.portNumber*.	ifName *ifName*
In Octets	The total number of octets received on the interface (inbound octets).	inOctets64 *integer*

Field name	Description	Filter name
`In Unicast Pkts`	The total number of unicast packets received on the interface. That is, the number of packets not addressed to a broadcast or multicast address.	`inUcastPkts64` *`integer`*
`In Multicast Pkts`	The total number of packets received that were addressed to a broadcast address on the interface.	`inMcastPkts64` *`integer`*
`Out Octets`	The total number of octets transmitted over the interface (outbound octets).	`outOctets64` *`integer`*
`Out Unicast Pkts`	The total number of unicast packets requesting transmission over the interface. That is, the number of packets not addressed to a broadcast or multicast address, that were sent or discarded.	`outUcastPkts64` *`integer`*
`Out Multicast Pkts`	The total number of packets, both discarded and sent, requesting transmission that were addressed to a broadcast address on the interface.	`outMcastPkts64` *`integer`*

Associated MIB

`ether.mib`

Web path

- Port → ipStatistics

show port mirror

Purpose

Displays the current port mirroring configuration, including source port, observation port, current state, and the network traffic direction being mirrored.

Access mode

user

Syntax

```
show port mirror
```

Sample output

```
sun(config)# show port mirror
Source Port     Observer Port   State      Oper Status   Direction
eth.1.1         eth.1.2         enabled    down          both
eth.1.3         eth.1.2         enabled    down          both
```

Output description

Field name	Description	Filter name
`Source Port`	The system-assigned port name, in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`Observer Port`	The name of the port to be configured as a mirror port where you are observing network traffic from an active N2000 Series port. Interface names are in the format *`type.slot.portNumber`*.	`observerPort` *`ifName`*
`State`	The current enabled or disabled state of the port mirror. The default setting is `enabled`.	`state {enabled | disabled}`
`Oper Status`	The operational status of the port mirror, based on whether the mirrored port is operational and whether the observer port has a link.	`operStatus {down | up}`

Field name	Description	Filter name
`Direction`	The type of source port traffic to be mirrored on the observation port; inbound, outbound, or both. If `both`, the system mirrors bidirectional traffic to the observation port. Valid directions are `in`, `out`, and `both`; the default is `both`.	`portMirrorDir {in |out | both}`

Associated MIB

`ether.mib`

Web path

- Port → mirror

show port mirror availability

Purpose

Displays the available ports that you can include in a port mirror configuration. The port listing includes individual Ethernet ports and contiguous port ranges. The system supports one mirror port per network processor (NP), where the NP controls a certain number of system ports.

Depending on your system hardware, you can configure either two or three port mirrors per switch, as follows:

On the N2040 model

- NP1 supports ports 1, 2, 5 to 22
- NP2 supports ports 3, 4 23 to 44

On the N2120 model

- NP1 supports ports 1 to 4
- NP2 supports ports 5 to 8
- NP3 supports ports 9 to 12

Access mode

user

Syntax

```
show port mirror availability
```

Sample output

```
sun(config)# show port mirror availability
Ports Available
eth.1.3-eth.1.4, eth.1.25-eth.1.44
```

Output description

Field name	Description	Filter name
`Ports Available`	The system-assigned port names, in the format *`type.slot.portNumber`*. Port ranges are separated by the hyphen character (-), and ranges and individual ports are separated by a comma (,).	`N/A`

Associated MIB

`ether.mib`

Web path

- Port → mirror →availability

show port statsMIB

Purpose

Displays combined transmit and receive statistics for all ports or those ports matching the filtering criteria. These statistics are defined in RFC 2665, *Definitions of Managed Objects for the Ethernet-like Interface Types*.

Access mode

user

Syntax

```
show port statsMIB
```

Sample output

```
sun> show port statsMIB
IfName:                        eth.1.1
Alignment Errors:              0
FCS Errors:                    0
Single Collision Frames:       0
Multiple Collision Frames:     0
Deferred Transmissions:        0
Late Collisions:               0
Excessive Collisions:          0
Internal MAC Transmit Errors:  0
Frame Too Long:                0
Internal MAC Receive Errors:   0
Short Frames:                  0
Fragments And Runts:           0
Total RX Frames:               13272
Total TX Frames:               33445
Jabber:                        0
Jumbo Frames:                  0
```

Output description

Field name	Description	Filter name
`If Name`	The system-assigned port name, in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`Alignment Errors`	The number of frames received that are not an integral number of bytes long and that fail the FCS check.	`alignmentErrors` *`integer`*
`FCS Errors`	The number of frames received that are an integral number of bytes long but that fail the FCS check. This number does not include the frame-too-long number.	`fcsErrors` *`integer`*
`Single Collision Frames`	The number of frames transmitted that met with just one collision before the 64-bytes traversed the connection.	`singleCollisionFrames` *`integer`*
`Multiple Collision Frames`	The number of frames transmitted that met with more than one collision before the 64-bytes traversed the connection.	`multipleCollisionFrames` *`integer`*
`Deferred Transmissions`	The number of frames that were ultimately sent after the system initially deferred them (the first transmission attempt was delayed because the medium was busy). This number does not include frames involved in any collisions.	`deferredTransmissions` *`integer`*
`Late Collisions`	The number of times a collision is detected later than one slot-time from the start of packet transmission. On 10/100-Mbps systems, one slot-time = 512 bit-times = 51.2 and 5.12 microseconds, respectively. On 1000-Mbps systems, one slot-time = 4096 bit-times = 4.096 microseconds.	`lateCollisions` *`integer`*
`Excessive Collisions`	The number of frames transmitted that exceeded the 16-collision limit on attempting to cross the connection, causing the system to drop the packet.	`excessiveCollisions` *`integer`*

Field name	Description	Filter name
`Internal MAC Transmit Errors`	The number of frames that fail transmission due to an internal MAC sublayer transmit error. This number does not include late collisions or excessive collisions.	`internalMacTransmitErrors` *`integer`*
`Frame Too Long`	The number of frames received on the port that exceeded the standard Ethernet size limitation but had a valid CRC: Without jumbo frames enabled with the `port` command, frames over 1518 bytes or 1522 bytes for VLAN frames. With jumbo frames enabled with the `port` command, frames over 9018 bytes or 9022 bytes for VLAN frames.	`frameTooLongs` *`integer`*
`Internal MAC Receive Errors`	The number of frames that fail reception due to an internal MAC sublayer receive error. This number does not include alignment, FCS, or frame-too-long errors.	`internalMacReceiveErrors` *`integer`*
`Short Frames`	The number of frames transmitted and received that were less than 64 bytes and had a good CRC.	`shortframes` *`integer`*
`Fragments and Runts`	The number of frames that were less than 64 bytes and had a bad CRC. (Fragments are received frames and runts are transmitted frames.)	`fragRunt` *`integer`*
`Total RX frames`	The total number of received frames.	`rxFrames` *`integer`*
`Total TX frames`	The total number of transmitted frames.	`txFrames` *`integer`*
`Jabber`	The number of transmitted and received long frames (see `Frame Too Long` field, above) on the port that exceeded the standard Ethernet size limitation but had a bad CRC.	`jabber` *`integer`*

Field name	Description	Filter name
`Jumbo Frames`	The number of frames transmitted and received that qualified as jumbo frames (between 1518 and 9018 bytes to support 100M and Gigabit Ethernet). This count is only applicable if jumbo frames were enabled with the `port` command.	`jumbo` *`integer`*
`Duplex Status`	The port's duplex setting. A value of `halfDuplex` indicates asynchronous transmission, `fullDuplex` indicates simultaneous transmission, and `unknown` indicates the system can't tell because the other side is not sending advertisements.	`duplexstatus`

Associated MIB

`ether.mib`

Web path

- Port → statsMIB

show port statsRX

Purpose

Displays receive statistics for all ports or those ports matching the filtering criteria. The statistics displayed by this command exceed those defined in RFC 2665, *Definitions of Managed Objects for the Ethernet-like Interface Types*.

Access mode

user

Syntax

```
show port statsRX
```

Sample output

```
sun> show port statsRX eth.1.9
IfName:                    eth.1.9
Short Frames:              0
Fragments:                 0
64 Byte Frames:            14
65 To 127 Byte Frames:     0
128 To 255 Byte Frames:    0
256 To 511 Byte Frames:    0
512 To 1023 Byte Frames:   0
1024 To 1518 Byte Frames:  0
Long Frames:               0
Jabber:                    0
Bad CRC:                   0
Unicast Frames:            0
Broadcast Frames:          0
Multicast Frames:          14
Total RX Frames:           14
Errors:                    0
Receive Octets:            896
Receive Overruns:          0
Jumbo Frames:              0
```

Output description

Field name	Description	Filter name
`IfName`	The system-assigned port name, in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`Short Frames`	The number of frames received that were less than 64 bytes and had a good CRC.	`rcvShort` *`integer`*
`Fragments`	The number of frames received that were less than 64 bytes and had a bad CRC.	`rcvFragment` *`integer`*
`64 Byte Frames`	The number of frames received, valid and invalid, that were 64 bytes.	`rcvS64` *`integer`*
`65 to 127 Byte Frames`	The number of frames received, valid and invalid, that were between 65 and 127 bytes.	`rcvS127` *`integer`*
`128 to 255 Byte Frames`	The number of frames received, valid and invalid, that were between 128 and 255 bytes.	`rcvS255` *`integer`*
`256 to 511 Byte Frames`	The number of frames received, valid and invalid, that were between 256 and 511 bytes.	`rcvS511` *`integer`*
`512 to 1023 Byte Frames`	The number of frames received, valid and invalid, that were between 512 and 1023 bytes.	`rcvS1023` *`integer`*
`1024 to 1518 Byte Frames`	The number of frames received, valid and invalid, that were between 1024 and 1518 bytes.	`rcvS1518` *`integer`*

Field name	Description	Filter name
`Long Frames`	The number of frames received on the port that exceeded the standard Ethernet size limitation but had a valid CRC: Without jumbo frames enabled with the `port` command, frames over 1518 bytes or 1522 bytes for VLAN frames. With jumbo frames enabled with the `port` command, frames over 9018 bytes or 9022 bytes for VLAN frames.	`rcvLong` *`integer`*
`Jabber`	The number of long frames (see the `Long Frame` field above) received on the port that exceeded the standard Ethernet size limitation but had a bad CRC.	`rcvJabber` *`integer`*
`Bad CRC`	The number of frames received with a bad CRC.	`rcvBadcrc` *`integer`*
`Unicast Frames`	The number of frames received that were destined for a unicast address. This count excludes multicast, broadcast, and long frames.	`rcvUnicast` *`integer`*
`Broadcast Frames`	The number of frames received that were destined for a broadcast address. This count excludes multicast and long frames.	`rcvBroadcast` *`integer`*
`Multicast Frames`	The number of frames received that were destined for a multicast address. This count excludes broadcast and long frames.	`rcvMulticast` *`integer`*
`Total RX frames`	The total number of frames received, unicast, broadcast, and multicast, with good and bad CRC.	`rcvTotal` *`integer`*
`Errors`	The total number of frames received that generated an error signal.	`rcvError` *`integer`*
`Receive octets`	The number of bytes of data received on the port.	`rcvOctet` *`integer`*

Field name	Description	Filter name
`Receive overruns`	The number of frames that were received when the internal buffer was full, and were therefore discarded.	`rcvOverrun` *`integer`*
`Jumbo frames`	The number of frames received that qualified as jumbo frames (between 1518 and 9018 bytes to support 100M and Gigabit Ethernet). This count is only applicable if jumbo frames were enabled with the `port` command.	`rcvJumbo` *`integer`*

Associated MIB

`ether.mib`

Web path

- Port → statsRX

show port statsTX

Purpose

Displays transmit statistics for all ports or those ports matching the filtering criteria. The statistics displayed by this command exceed those defined in RFC 2665, *Definitions of Managed Objects for the Ethernet-like Interface Types*.

Access mode

user

Syntax

```
show port statsTX
```

Sample output

```
sun# show port statsTX eth.1.9
IfName:                    eth.1.9
Short Frames:              0
Runts:                     0
64 Byte Frames:            115
65 To 127 Byte Frames:     0
128 To 255 Byte Frames:    0
256 To 511 Byte Frames:    0
512 To 1023 Byte Frames:   0
1024 To 1518 Byte Frames:  0
Long Frames:               0
Jabber:                    0
Late Collisions:           0
Total Collisions:          0
Single Collisions:         0
Multiple Collisions:       0
Deferrals:                 0
Underruns:                 0
Bad CRCs:                  0
Excessive Collisions:      0
Unicast Frames:            0
Broadcast Frames:          0
Multicast Frames:          115
Octets:                    7360
Jumbo Frames:              0
Aborts:                    0
```

Output description

Field name	Description	Filter name
`IfName`	The system-assigned port name, in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`Short Frames`	The number of transmitted frames that were less than 64 bytes and had a good CRC.	`txShort` *`integer`*
`Runts`	The number of transmitted frames that do not meet Ethernet's 64-byte size requirement. Runts are usually a result of collisions.	`txRunt` *`integer`*
`64 byte frames`	The number of frames transmitted, valid and invalid, that were 64 bytes.	`txS64` *`integer`*
`65 to 127 Byte Frames`	The number of frames transmitted, valid and invalid, that were between 65 and 127 bytes.	`txS127` *`integer`*
`128 to 255 Byte Frames`	The number of frames transmitted, valid and invalid, that were between 128 and 255 bytes.	`txS255` *`integer`*
`256 to 511 Byte Frames`	The number of frames transmitted, valid and invalid, that were between 256 and 511 bytes.	`txS511` *`integer`*
`512 to 1023 Byte Frames`	The number of frames transmitted, valid and invalid, that were between 512 and 1023 bytes.	`txS1023` *`integer`*
`1024 to 1518 Byte Frames`	The number of frames transmitted, valid and invalid, that were between 1024 and 1518 bytes.	`txS1518` *`integer`*

Field name	Description	Filter name
`Long Frames`	The number of frames transmitted on the port that exceeded the standard Ethernet size limitation but had a valid CRC: Without jumbo frames enabled with the `port` command, frames over 1518 bytes or 1522 bytes for VLAN frames. With jumbo frames enabled with the `port` command, frames over 9018 bytes or 9022 bytes for VLAN frames.	`txLong` *`integer`*
`Jabber`	The number of long frames (see the `Long Frame` field above) transmitted on the port that exceeded the standard Ethernet size limitation but had a bad CRC.	`txJabber` *`integer`*
`Late Collisions`	The number of frames transmitted that met with a collision after at least 64-bytes traversed the connection.	`txCollLate` *`integer`*
`Total Collisions`	The total number of collisions resulting from transmitted frames.	`txCollTotal` *`integer`*
`Single Collisions`	The number of frames transmitted that met with just one collision before the 64-bytes traversed the connection.	`txCollSingle` *`integer`*
`Multiple Collisions`	The number of frames transmitted that met with more than one collision before the 64-bytes traversed the connection.	`txCollMulti` *`integer`*
`Deferrals`	The number of frames that were not transmitted because the deferral timer expired.	`txDeferral` *`integer`*
`Underruns`	The number of frames that were not transmitted because the data could not be pushed out the port fast enough for the configured media type's rate.	`txUnderrun` *`integer`*
`Bad CRCs`	The number of frames transmitted that had bad CRCs.	`txBadCrc` *`integer`*

Field name	Description	Filter name
`Excessive Collisions`	The number of frames transmitted that exceeded the 16-collision limit on attempting to cross the connection, causing the system to drop the packet.	`txCollExcess` *`integer`*
`Unicast Frames`	The number of frames transmitted that were destined for a unicast address. This count excludes multicast, broadcast, and long frames.	`txUnicast` *`integer`*
`Broadcast Frames`	The number of frames transmitted that were destined for a broadcast address. This count excludes multicast and long frames.	`txBroadcast` *`integer`*
`Multicast Frames`	The number of frames transmitted that were destined for a multicast address. This count excludes broadcast and long frames.	`txMulticast` *`integer`*
`Octets`	The number of bytes of data transmitted over the port.	`txOctet` *`integer`*
`Jumbo Frames`	The number of packets transmitted that qualified as jumbo frames (between 1518 and 9018 bytes to support 100M and Gigabit Ethernet). This count is only applicable if jumbo frames were enabled with the `port` command.	`txJumbo` *`integer`*
`Aborts`	The number of frames that were not transmitted due to parity errors, header errors, or invalid BCI specification errors.	`txAbort` *`integer`*

Associated MIB

`ether.mib`

Web path

Port → statsTX

show port verbose

Purpose

Displays extended characteristics of each configured port. The system comes with all ports preconfigured and prenamed. This command displays the current operational settings that are either default settings or configured with the `port` command. The number of ports displayed depends on the model. See Table 19-1, “Available system ports,” on page 19-1 for port numbering information.

If a port is a member of a LAG, it inherits the administrative MAC address, default VLAN, and jumbo frames setting of that LAG. The settings you have configured with the `port` command are saved, but are not used unless you remove the port from the LAG. This command displays both the configured settings and, if the port is a LAG member, the operational settings.

Access mode

user

Syntax

```
show port verbose
```

Sample output

The following output displays the different settings for a port when it is in a LAG and after it has been removed:

```
sun> show port verbose
IfName:                 eth.1.1
IfIndex:                0x81010000
Port Type:              gigEthernet
Port MAC:               00:07:82:00:07:cb
Port Mode:              normal
Port Speed:             auto
Port Duplex:            fullDuplex
Jumbo Frames:           disabled
Operational Status:     up
Advertised Speed:       both
Advertised Duplex:      both
Default Vlan:           55
Autonegotiated Status: inProgress
Autonegotiated Duplex: N/A
```

```
Autonegotiated Speed:  N/A
Link Status:           linkFail
Lag Membership:        30
Lag Defvlan Setting:   4095
Lag Jumbo Setting:     disabled

sun(config)# port eth.1.1 defVlan 1000

WARNING: values for Admin MAC, default vlan, and jumbo frame will not
take effect until port is removed from LAG

sun(config)# lag 30
sun(config-lag-30)# no interface eth.1.1
sun(config-lag-30)# show port verbose eth.1.1
IfName:                eth.1.1
IfIndex:               0x81010000
Port Type:             gigEthernet
Port MAC:              00:01:01:01:01:01
Port Mode:             normal
Port Speed:            auto
Port Duplex:           fullDuplex
Jumbo Frames:          disabled
Operational Status:    up
Advertised Speed:      both
Advertised Duplex:     both
Default Vlan:          1000
Autonegotiated Status: inProgress
Autonegotiated Duplex: N/A
Autonegotiated Speed:  N/A
Link Status:           linkFail
Lag Membership:        N/A
Lag Defvlan Setting:   N/A
Lag Jumbo Setting:     N/A
```

Output description

Field name	Description	Filter name
`IfName`	The system-assigned port name, in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`IfIndex`	The system-assigned port name, in the hexadecimal.	`ifindex` *`hexValue`*
`Port Type`	The port's media type, either gigabit Ethernet (`gigEthernet` or `copperGig`) or 10M/100M Ethernet (`fastEthernet`).	`portType {gigEthernet | fastEthernet | copperGig}`

Field name	Description	Filter name
`Port MAC`	The MAC address the port is using, either the factory preset address or the administratively assigned address. Or, if the port is a member of a LAG, the MAC address of the flood port for the LAG.	`operMac` *`macAddress`*
`Port Mode`	The port's physical mode, either `normal` or `loopback`. If `normal`, the port passes traffic as a normal physical port. If `loopback`, the port receives back the packets it transmits (for testing).	`phyMode {normal | loopback}`
`Port Speed`	The port speed setting. The `auto` setting configures the system to negotiate with the remote client to achieve the best common speed and duplex settings. A value of `10M`, `100M`, or `1000M` indicates a specific speed. If the port type is copper gigabit Ethernet, the configured speed setting is displayed, but that is not necessarily the operational setting.	`phySpeed {10M | 100M | 1000M | auto}`
`Port Duplex`	The port's duplex setting. A value of `halfDuplex` indicates asynchronous transmission; a value of `fullDuplex` indicates simultaneous transmission.	`phyDuplex {halfDuplex | fullDuplex}`
`Jumbo Frames`	The administrative setting for the port's acceptance of jumbo frames (9018-byte frames for 100M and Gigabit Ethernet.) If `enabled` jumbo frames are accepted; if `disabled` they are discarded. If the port is a member of a LAG, this setting is overridden by the LAG's jumbo frames setting.	`jumboFrames {enabled | disabled}`

Field name	Description	Filter name
`Operational Status`	The operational status of the port, based on the port configuration and link status. This status reflects whether the port is configured under a VLAN or a virtual router *and* the link is operational. When the port is configured *and* the link is operational, the operational status is `up`; the port can begin to receive and transmit frames. In all other cases, the operational status is `down`.	`operStatus {down | up}`
`Advertised Speed`	The speed the port advertises to the remote device if port speed is set to `auto` with the `port` command.	`advSpeed {10M | 100M | both}`
`Advertised Duplex`	The duplex setting that the port advertises to the remote device if port speed is set to `auto` with the `port` command.	`advDuplex {halfDuplex | fullDuplex | both}`
`Default Vlan`	The ID number of the VLAN to which untagged packets are forwarded. The default is ID 4095, which is the discard VLAN. That is, traffic assigned to this VLAN is dropped. If the port is a member of a LAG, this setting is overridden by the LAG's default VLAN setting.	`defVlan` *`vlanId`*
`Autonegotiated Status`	The status of the auto-negotiation, either `complete` or `inProgress`. If port speed was not set to `auto` with the `port` command, the field displays `inProgress`.	`aNegStatus {complete | inProgress}`
`Autonegotiated Duplex`	The negotiated duplex setting. If port speed was not set to `auto` with the `port` command, the field displays `N/A`. Verify the port speed setting in the `Port Speed` field, above.	`aNegDuplex {halfDuplex | fullDuplex | N/A}`
`Autonegotiated Speed`	The negotiated speed setting. If port speed was not set to `auto` with the `port` command, the field displays `N/A`. Verify the port speed setting in the `Port Speed` field, above.	`aNegSpeed {10M | 100M |auto | N/A}`

Field name	Description	Filter name
`Link Status`	The status of the connection to the remote device, either `linkFail` or `linkPass`.	`linkStatus {linkFail | linkPass}`
`Lag Membership`	The ID of the LAG to which this port belongs. If the port is not a member of a LAG, the field displays `N/A`.	`lagMembership` *`integer`*
`Lag Defvlan Setting`	The default VLAN setting for the port, which is inherited from the LAG in which this port is a member. (See the `Default Vlan` field, above, for more information.) If the port is not a member of a LAG, the field displays `N/A`.	`lagDefvlanSetting` *`integer`*
`Lag Jumbo Setting`	The setting for the port's acceptance of jumbo frames, which is inherited from the LAG in which this port is a member. (See the `Jumbo Frames` field, above, for more information.) If the port is not a member of a LAG, the field displays `N/A`.	`lagJumboSetting {disabled | enabled}`

Associated MIB

`ether.mib`

Web path

- Port → port list

Chapter 20. LAG commands

LAG description

Link aggregation is a method of grouping multiple Ethernet ports into a virtual link with aggregated bandwidth. The system treats the set of ports in a link aggregation group (LAG) as a single port. All the ports within the LAG use the same Layer 2 MAC address and same default virtual LAN (VLAN). Traffic is distributed across the LAG in a way that ensures that traffic for a particular user stays in order. You can specify which port within the LAG to use as the flood port for broadcast or multicast traffic.

Within the system's interface and port hierarchy, a LAG can connect to the following upper layers:

- A single virtual router interface (IP running directly over the LAG)
- One or more VLANs

A LAG can connect to the following lower layers:

- One or more Ethernet ports

LAGs support the following:

- Mixed media (that is, both Gigabit Ethernet and 10/100-Mbps Ethernet ports within the same LAG)
- A maximum of 22 LAGs per system
- A maximum of 16 Ethernet ports per LAG

Using weights for traffic distribution across a LAG

Traffic is distributed over the LAG based on the weight set for each port in the configuration. The weight is set with the `weight` argument of the `interface` command. The value that you assign as a weight is relative to the weights of the other ports in the LAG. Any given port in the LAG carries a fraction of the entire LAG traffic that is equal to its weight divided by the sum of all weights of all ports in the LAG. For example, if you have a LAG with one 100M port and one 1000M port, you can configure the weighted distribution to be 10% for the 100M port and 90% for the 1000M port by specifying weights of 1 and 10, respectively.

As the system receives packets, it hashes information in each received packet to an 8-bit value. (For Layer 2 traffic, the hash is based on MAC destination address and source address; for Layer 3 traffic, it is based on the IP destination address and source address.) The system uses this value, in conjunction with the configured weight, to select the port that will carry the traffic. In this way, all traffic that belongs to a given flow will always be forwarded across the same port in the LAG (and therefore is kept in order).

If a port fails, the weight for each active port is regenerated based on the remaining active ports in the LAG. Traffic is then redistributed over all active links based on the new weights.

For example, consider that you have three ports, weighted as follows:

Port	Weight
port1	5
port2	10
port3	15

Assume that port2 became inactive. The traffic would then be redistributed based on the weight recalculation that excludes port2. The result would be that port1 receives one fourth of the traffic and port3 receives three fourths of the traffic. If the port is later reactivated, the weights are again regenerated and the traffic is redistributed over all active links based on the new weights.

Flood ports on a LAG

LAG interfaces use a flood port to broadcast address requests for the switch and to flood packets belonging to unknown addresses, and for other broadcast and multicast traffic. Traffic whose destination address has not yet been learned is “flooded” out the flood port in an attempt to find the destination.

You can specify the flood port preference with the `interface` command. When you configure an Ethernet port as part of the LAG, you assign a preference for selection as the flood port. The active port with the lowest value is selected as the flood port. The system displays the port preference value with the `show interface` command and displays the active flood port with the `show` command.

If the port becomes unavailable, the software selects another flood port based on the flood port assignments (or defaults). When the original flood port again becomes active, it resumes as the flood port assuming it still has the highest ranking (lowest configured flood port preference). In the event of a tie in ranking, the system selects the port that first became active on the LAG.

Port configuration priority

You can configure characteristics of Ethernet ports through several mechanisms:

- At the port level, using the `port` command (see Chapter 19, “Ethernet data port commands”).
- At the LAG level, using the `lag` command. These settings override those set with the `port` command.
- At the LAG interface level, using the `interface` command. These settings override those set with the `port` or `lag` commands. Any parameters not set with the `interface` command are inherited from the `lag` command, or if none, from the `port` command. If not set with the `port` command, default values are used.

LAG command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`lag`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

LAG command summary

Table 20-1 lists and briefly describes the LAG commands.

Table 20-1. LAG command summary

Command name	Description
`interface`	Add an Ethernet port to the specified LAG.
`lag`	Create a new LAG and enter the LAG configuration context.
`show`	Display basic LAG configuration parameters.
`show verbose`	Display more detailed LAG configuration parameters.
`show interface`	Display basic configuration parameters for Ethernet ports in the LAG.
`show interface verbose`	Display more detailed configuration parameters for Ethernet ports in the LAG.

Basic LAG configuration

The following procedure explains the basic steps for configuring a LAG.

Table 20-2. Steps for configuring a LAG

Step	Action
1.	Assign a numeric identifier as LAG ID with the `lag` command.
2.	Assign Ethernet ports to the LAG with the `interface` command.
3.	Verify the configuration with the `show interface` command.
4.	Enable the LAG by stacking the interface in a vrouter or a VLAN.

interface

Purpose

Adds an Ethernet port to the specified LAG. When you add an Ethernet port with this command, the port inherits the characteristics set with the lag command. If you set characteristics with this command, those settings take precedence over the settings of the lag command.

The no form of the command deletes the specified port from the LAG. When a port is removed from a LAG, it reverts to its previous port configuration.

Access mode

config

Syntax

For creating LAG interfaces:

```
lag lagId interface
   ifName ifName
   [floodPref ranking]
   [weight value]
```

For modifying LAG interfaces:

```
lag lagId interface
   ifName ifName
   [floodPref ranking]
   [weight value]
   [adminState {disabled | enabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {disabled | enabled}]
   [mtu integer]
   [packetTrace {enabled | disabled}]
   [description text]
```

Arguments

Argument name	Description
`lagName` *`name`*	Specifies the name of the LAG.
`ifName` *`ifName`*	Specifies the name of the port you want to add to the LAG. Interface names are entered in the format *`type.slot.portNumber`*.
`floodPref` *`ranking`*	**Optional**. Specifies a preference for the port in its selection as flood port. Valid values are 1 through 16; the lower the value, the higher the preference. The default value is 4 for Gigabit Ethernet ports and 8 for 10/100 MB Ethernet ports. See "Flood ports on a LAG" for more information.
`weight` *`integer`*	**Optional**. Specifies the relative weight of the port. Valid values are 1 through 65535; the higher the value, the higher the utilization. The default value is 1000 for Gigabit Ethernet ports and 100 for 10/100 MB Ethernet ports. See "Using weights for traffic distribution across a LAG" for more information.
The following arguments are only available when modifying a LAG configuration:	
`adminState {enabled | disabled}`	**Optional**. Sets the administrative mode of the LAG. The LAG is enabled by default; use this command to disable it on the switch.

Argument name	Description
`eventFilter` *`filterLevel`*	**Optional**. Sets the event filter level for the interface. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`.
`linkUpDownTrap {enabled | disabled}`	**Optional**. Specifies whether the system should generate a trap when this interface changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 6, "SNMP and trap commands."
`mtu` *`integer`*	**Optional**. Sets the maximum transmission unit (MTU) for all Ethernet ports configured within this LAG. This is the maximum length, in number of bytes, of a packet transmitted over the ports. Packets are fragmented to this size. Valid range is 0 to 9000 bytes; the default MTU is 1500 bytes. The MTU cannot be modified on LAG interfaces.

Argument name	Description
`packetTrace {enabled | disabled}`	**Optional**. Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.
`description` *`text`*	**Optional**. Assigns a text string to an interface. Use this command to identify an interface name (for example, eth.1.27) with a meaningful description. This string is displayed with the `show interface verbose` command.

Delete filters

See the `show interface verbose` command for argument descriptions.

```
no lag lagId interface
   ifName ifName
   [lagIfName ifName]
   [floodPref ranking]
   [weight value]
   [operWeight percentage]
   [adminState {disabled | enabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {disabled | enabled}]
   [status {up | down | testing | unknown | dormant}]
   [type etherLAG]
   [mtu integer]
   [speed integer]
   [lastChange time]
   [packetTrace {enabled | disabled}]
   [description text]
   [ifIndex hexInteger]
```

Example

The following example shows how to assign the ports eth.1.40, eth.1.41, and eth.14.2 to LAG 10. In this example, the flood port is eth.1.40 because it has the lowest flood preference. The eth.1.41 port carries the largest fraction of traffic because it has the highest weight value.

```
sun(config-lag-10)# interface eth.1.40 floodPref 2 weight 25
```

```
sun(config-lag-10)# interface eth.1.41 floodPref 10 weight 50
sun(config-lag-10)# interface eth.1.42 floodPref 15 weight 25
```

Associated MIB

```
ethLag.mib
```

Web path

- Lag → interface → add
- Lag → interface → copy
- Lag → interface → modify
- Lag → interface → delete
- Lag → interface → verbose → add
- Lag → interface → verbose → copy
- Lag → interface → verbose → modify
- Lag → interface → verbose → delete

lag

Purpose

Creates a new LAG with the specified number as an ID and enters the LAG configuration mode. From this command you can configure several characteristics of the LAG. All ports assigned to the LAG inherit these characteristics. If you have previously configured these characteristics with the `port` command (described in Chapter 19, "Ethernet data port commands"), the settings configured with this command override those settings. Any of these characteristics set with the `interface` command take precedence over this command's settings.

The no form deletes the specified LAG. Values that are available for filtering when deleting a LAG can be viewed with the various `show` commands.

Access mode

config

Syntax

For creating LAGs:

```
lag
   lagId integer
   [jumboFrames {disabled | enabled}]
   [defVlan vlanId]
```

For modifying LAGs:

```
lag lagId
   [jumboFrames {disabled | enabled}]
   [defVlan vlanId]
   [adminState {disabled | enabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {enabled | disabled}]
   [packetTrace {enabled | disabled}]
   [description text]
```

Arguments

Argument name	Description
`lagName` *name*	Specifies the name of the LAG. Only 22 LAGs are supported by the N2000.
`jumboFrames {enabled | disabled}`	**Optional**. Configures the port to accept or reject jumbo frames. (Jumbo frames extend the traditional 1500 byte Ethernet frame size to 9018 bytes for 100M and Gigabit Ethernet.) The default setting is `disabled`.
`defVlan` *name*	**Optional**. Specifies the name of the default VLAN. Any packet arriving on the port that does not have a VLAN name in its extended MAC header is assigned this VLAN name. If the LAG is in a VLAN, untagged frames arriving on this port will be discarded. If you do not want the traffic to be dropped, assign a different VLAN name.
The following arguments are only available when modifying a LAG configuration:	
`adminState {enabled | disabled}`	**Optional**. Sets the administrative mode of the LAG. The LAG is enabled by default; use this command to disable it on the switch.
`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`	**Optional**. Sets the event filter level for the LAG. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`.

Argument name	Description
`linkUpDownTrap {enabled | disabled}`	**Optional**. Specifies whether the system should generate a trap when this LAG changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 6, "SNMP and trap commands."
`packetTrace {enabled | disabled}`	**Optional**. Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by this setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.
`description` *`text`*	**Optional**. Assigns a text string to the LAG. Use this command to identify an interface name (for example, lag.10) with a meaningful description. This string is displayed with the `show lag verbose` command.

Delete filters

See the `show verbose` command for argument descriptions.

For deleting LAGs:

```
no lag lagId
    [lagName IfName]
    [adminMac macAddress]
    [jumboFrames {disabled | enabled}]
    [defVlan vlanId]
    [operMac macAddress]
    [floodPort IfName]
    [adminState {disabled | enabled}]
    [eventFilter {emergency | alert | critical | error | warning |
       notice | informational | debug}]
    [linkUpDownTrap {disabled | enabled}]
    [status {up | down | testing | unknown | dormant}]
    [lastChange time]
    [packetTrace {enabled | disabled}]
    [description text]
    [ifIndex hexInteger]
```

Example

The following example shows how to create a LAG that supports jumbo frames and sends untagged frames to VLAN 25.

```
sun> enable
sun# config
sun(config)# lag 10 jumboFrames enabled defVlan 25
sun(config)#
```

After creating the LAG, you need to assign ports to the LAG using the lag `interface` command, and add the LAG to VLAN 25, using the `vRouter-`*`name`* `vlan` command and `vRouter-`*`name`* `vlan interface` command.

Associated MIB

`ethLag.mib`

Web path

- Lag → lag list → add
- Lag → lag list → copy
- Lag → lag list → modify
- Lag → lag list → delete
- Lag → verbose → add
- Lag → verbose → copy
- Lag → verbose → modify
- Lag → verbose → delete

show

Purpose

Displays the current configuration for the LAG. The user-configurable parameters are set with the `lag` command. For a more detailed display of the LAG's configuration, use the `show verbose` command.

Access mode

user

Syntax

```
show lag
```

Sample output

```
sun> show lag
Lag ID:       1
Admin State:  enabled
Oper Status:  up
Port MAC:     00:07:82:00:03:86
Jumbo Frames: disabled
Default Vlan: 4095
Flood Port:   eth.1.5

Lag ID:       10
Admin State:  enabled
Oper Status:  up
Port MAC:     00:07:82:00:03:88
Jumbo Frames: disabled
Default Vlan: 4095
Flood Port:   eth.1.7
```

Output description

Field name	Description	Filter name
`Lag Name`	The name of the LAG.	`lagName` *`text`*
`Admin State`	The administrative mode of the LAG. The LAG is `enabled` by default.	`adminState {disabled | enabled}`
`Oper Status`	The operational status of the LAG. That is, whether it has been configured under a VLAN or a virtual router and can begin to receive and transmit frames. The LAG is either configured and available (up) or not (down).	`status {up | down | testing | unknown | dormant}`
`Port MAC`	The MAC address the port is using, either the MAC address of the first port added to the LAG or the administratively assigned address.	`operMac` *`macAddress`*
`Jumbo Frames`	The setting for the LAG's acceptance of jumbo frames (9018-byte frames for 100M and Gigabit Ethernet.) If `enabled` jumbo frames are accepted; if `disabled` they are discarded.	`jumboFrames {disabled | enabled}`
`Default Vlan`	The name of the VLAN to which untagged packets are forwarded if the LAG is in a VLAN. Traffic assigned to this VLAN is dropped.	`defVlan` *`text`*
`Flood Port`	The interface the LAG is using to broadcast address inquiries (the operational flood port). This port is assigned by the system based on the values you configure with the `interface` command.	`floodPort` *`ifName`*

Associated MIB

`ethLag.mib`

Web path

- Lag → lag list

show verbose

Purpose

Displays a detailed picture of the current configuration for the LAG. The user-configurable parameters are set with the `lag` command. For a brief display of the LAG's configuration, which allows a horizontal view with multiple interface listings on a single screen, use the `show` command.

Access mode

user

Syntax

```
show lag verbose
```

Sample output

```
sun(config)# show lag verbose
Lag ID:       10
Admin State:  enabled
Oper Status:  up
Port MAC:     00:07:82:00:03:88
Jumbo Frames: disabled
Default Vlan: 4095
Flood Port:   eth.1.7
LagName:      lag.10
IfIndex:      0x600A0000
Link Traps:   disabled
Event Filter: informational
Packet Trace: disabled
Last Change:  41139809
Description:
```

Output description

Field name	Description	Filter name
`Lag Name`	The name of the LAG.	`lagName` *name*
`Admin State`	The administrative mode of the LAG. The LAG is `enabled` by default.	`adminState {disabled | enabled}`

Field name	Description	Filter name
`Oper Status`	The operational state of the LAG. That is, the state the LAG is currently in, either: `up`: The LAG is operational `down`: The LAG is not operational `testing`: The LAG is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The LAG is not in any of the listed states `dormant`: The LAG is uninitialized, the configuration database and hardware have not yet become synchronized	`status {up | down | testing | unknown | dormant}`
`Admin MAC`	The user-assigned MAC address, which overrides the factory preset address. If you have not assigned an address, the field displays 00:00:00:00:00:00.	`adminMac` *`macAddress`*
`Port MAC`	The MAC address the port is using, either the factory preset address or the administratively assigned address.	`portMac` *`macAddress`*
`Jumbo Frames`	The setting for the port's acceptance of jumbo frames (9018-byte frames for 100M and Gigabit Ethernet.) If `enabled` jumbo frames are accepted; if `disabled` they are discarded.	`jumboFrames {disabled | enabled}`
`Default Name`	The name of the VLAN to which untagged packets are forwarded if the LAG is in a VLAN. Traffic assigned to this VLAN is dropped.	`defVlan` *`name`*
`Flood Port`	The port the LAG is using to broadcast address inquiries (the operational flood port). This port is assigned by the system based on the values you configure with the `interface` command.	`floodPort` *`ifName`*
`LagIfName`	The system-generated identifier for the LAG. The format is `lag.`*`lagID`*, where: • *`lagID`*: the numeric identifier you assigned to the LAG with the lag command. For example, a value of `lag.`3 equates to the LAG you specified as LAG 3.	`lagifName` *`ifName`*

Field name	Description	Filter name
`IfIndex`	The hexadecimal equivalent of the LAG name. Use this value within SNMP to specify the LAG name.	`ifIndex` *`hexInteger`*
`Link Traps`	The setting for whether the system should generate a trap when this LAG changes state (from up to down, or vice versa). The default setting is `disabled`.	`linkUpDownTrap {enabled | disabled}`
`Event Filter`	The event filter level set for the LAG, one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default level is `informational`.	`eventFilter` *`level`*

Field name	Description	Filter name
`Pkt Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.	`packetTrace {enabled | disabled}`
`Last Change`	The value, in ticks, of the `sysUpTime` variable at the time of the last change to the LAG. This could be a change to the configuration, or in the administrative state or operational status. If the field displays 0, there have been no changes to the LAG.	`lastChange` *`integer`*
`Description`	The text string you assigned to the LAG. You can use this value to identify a LAG name (for example, lag.10) with a meaningful description.	`description` *`text`*

Associated MIB

`ethLag.mib`

Web path

- Lag → verbose

show interface

Purpose

Displays interface configuration data for either all or specified Ethernet interfaces in this LAG. For a more detailed display, use the `show interface verbose` command. Use the `interface` command to set the user-configurable parameters in this display.

Access mode

user

Syntax

`show lag` *`lagId`* `interface`

Sample output

```
sun(config)# show lag interface
Lag ID   IfName     Admin State  Oper Status  Flood Pref Weight   Oper Weight
10       eth.1.35   enabled      down         8          100      0
```

Output description

Field name	Description	Filter name
`Lag Name`	The name of the LAG..	`lagName` *`name`*
`ifName`	The name of the port you want to add to the LAG. Interface names are entered in the format *`type.slot.portNumber`*.	`ifName` *`ifName`*
`Admin State`	The administrative mode of the interface. The interface is `enabled` by default.	`adminState {enabled | disabled}`

Field name	Description	Filter name
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`status {up | down | testing | unknown | dormant}`
`Flood Pref`	The preference for the port in its selection as flood port. Valid values are 1 through 16; the lower the value, the higher the preference. The default value is 4 for Gigabit Ethernet ports and 8 for 10/100 MB Ethernet ports. See Flood ports on a LAG for more information.	`floodPref` *`ranking`*
`Weight`	The relative weight, as configured with the interface command. See Using weights for traffic distribution across a LAG for more information.	`weight` *`integer`*
`Oper Weight`	The operational weight, described as a percentage. See Using weights for traffic distribution across a LAG for more information.	`operWeight` *`integer`*

Associated MIB

`ethLag.mib`

Web path

- Lag → interface

show interface verbose

Purpose

Displays interface configuration data for either all or specified Ethernet ports in the LAG. For a brief display of interfaces, which allows a horizontal view with multiple interface listings on a single screen, use the `show interface` command. Use the `interface` command to set the user-configurable parameters in this display.

To view all interface configurations, do not enter a LAG ID. To view a specific LAG configuration, enter the ID.

Access mode

user

Syntax

`show lag` *`lagId`* `interface verbose`

Sample output

```
sun> show lag interface verbose
Lag ID:       10
IfName:       eth.1.35
Admin State:  enabled
Oper Status:  down
Flood Pref:   8
Weight:       100
Oper Weight:  100
LagIfName:    lag.10.1
IfIndex:      0x600A1001
Link Traps:   disabled
Event Filter: informational
Packet Trace: disabled
MTU:          1500
Speed:        100000000
Last Change:  176540
Description:
```

Output description

Field name	Description	Filter name
`Lag Name`	The name of the LAG.	`lagName` *name*
`IfName`	Specifies the name of the port you want to add to the LAG. Interface names are entered in the format *`type.slot.portNumber`*.	`ifName` *ifName*
`Admin State`	The administrative mode of the interface. The interface is `enabled` by default.	`adminState {enabled | disabled}`
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`status {up | down | testing | unknown | dormant}`
`Flood Pref`	The preference for the port in its selection as flood port. Valid values are 1 through 16; the lower the value, the higher the preference. The default value is 4 for Gigabit Ethernet ports and 8 for 10/100 MB Ethernet ports. See "Flood ports on a LAG" for more information.	`floodPref` *ranking*
`Weight`	The relative weight, as configured with the `interface` command. See "Using weights for traffic distribution across a LAG" for more information.	`weight` *integer*
`Oper Weight`	The operational weight, described as a percentage. See "Using weights for traffic distribution across a LAG" for more information.	`operWeight` *percentage*

Field name	Description	Filter name
`LagIfName`	The system-generated identifier for the interface. Note that there are no configuration commands that operate on these ifName assignments, they are strictly internal reference points. The format is `lag.`*`lagID.downlayer`*, where: *`lagID`*: the numeric identifier you assigned to the LAG with the `lag` command. *`downlayer`*: an interface index that the system assigns to the port when it is added to the LAG. For example, a value of `lag.3.2` equates to the second port added to the LAG you specified as LAG 3.	`lagIfName` *`ifName`*
`IfIndex`	The hexadecimal equivalent of the LAG interface name. Use this value within SNMP to specify an interface name.	`ifIndex` *`hexInteger`*
`Link Traps`	The setting for whether the system should generate a trap when this interface changes state (from up to down, or vice versa). The default setting is `disabled`.	`linkUpDownTrap {enabled | disabled}`

Field name	Description	Filter name
`Event Filter`	The event filter level set for the interface, one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default level is `informational`.	`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`
`Packet Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.	`packetTrace {enabled | disabled}`

Field name	Description	Filter name
MTU	The maximum transmission unit (MTU) for all Ethernet ports configured within this LAG. This is the maximum length, in number of bytes, of a packet transmitted over the ports. Packets are fragmented to this size. Valid range is 576 to 9000 bytes; the default MTU is 1500 bytes.	`mtu` *`integer`*
Speed	The speed setting for the specified port. The `auto` setting configures the system to negotiate with the remote client to achieve the best common speed and duplex settings. A value of `10M` or `100M` indicates a specific speed.	`speed` *`integer`*
Last Change	The value, in ticks, of the `sysUpTime` variable at the time of the last interface modification recorded in the interface table. If there have been no changes to the table, the field displays 0.	`lastChange` *`time`*
Description	The text string you assigned to the interface. You can use this value to identify an interface name (for example, eth.1.1) with a meaningful description.	`description` *`text`*

Associated MIB

`ethLag.mib`

Web path

- Lag → interface → verbose

Chapter 21. VLAN commands

VLAN description

A virtual LAN (VLAN) is a logical grouping of systems that is not constrained by geographic boundaries. These groupings create a broadcast domain, and function just like a traditional LAN. Systems within the VLAN are not necessarily physically co-located, but do not require a router to connect them. (Routers are used to connect two separate VLANs.) VLANs are interconnected using system bridging software. The N2000 Series supports up to 4094 VLANs system wide. Each VLAN is contained within a single virtual router context. One virtual router can support up to 512 VLANs with up to 5632 interfaces.

VLAN tagging

The N2000 Series supports VLAN tagging. Tagging allows a 802.1Q-compliant VLAN identifier to be added to the packet before it is sent.VLAN tagging (that is, whether a VLAN tag is added to the packet on the transmit port) is enabled on a per VLAN, per interface basis. When a packet is received with a VLAN tag, the packet is accepted if the ingress port is an active member of the indicated VLAN.

Spanning Tree Protocol description

Spanning Tree Protocol (STP) is a bridge-based protocol that allows you to configure an L2 network on the N2000 Series with redundant traffic paths. To prevent loops (multiple active paths between switches), the Spanning Tree algorithm allows only one path to exist between any two network devices. If STP finds a redundant path, it forces one of the paths into a blocked or standby state and uses the path with the lowest cost to forward data. If the original path fails, STP activates the redundant path to ensure the network continues to operate properly.

All switches in the VLAN exchange messages called Bridge Protocol Data Units (BPDUs) to determine the root bridge (the main reference point in the Spanning Tree network), learn about other switches, and to determine the network topology.

You can define one spanning tree instance for each VLAN on the N2000 Series. The bridging topology that is established by STP is VLAN-specific. Different VLANs run STP independently and may establish different forwarding topologies.

Spanning tree network phases

Table 21-1 describes the phases that the Spanning Tree protocol uses to create the network topology.

Table 21-1. Spanning Tree phases

Phase	Description
1.	**Root bridge selection** When the Spanning Tree is configured, all switches in the VLAN advertise themselves as the root bridge. Typically, the switch with the lowest bridge ID becomes the root bridge. The bridge ID consists of the configured priority and the switch MAC address. The root bridge is the only switch that generates configuration information to the rest of the switches in the VLAN.
2.	**Determine root path costs** STP assigns a cost to all of the paths from the root bridge to each port on a switch in the VLAN. STP bases this value on the number of network segments that a frame must traverse and the network segment's speed. Path cost can also be configured.
3.	**Root port selection** After determining the path costs, each switch determines which of their ports has the lowest cost path to the root bridge. This port becomes the root port for the switch. The root port receives the configuration BPDUs from the root bridge. All non-root switches participating in the Spanning Tree network have one root port; the root bridge does not have a root port.

Table 21-1. Spanning Tree phases (continued)

Phase	Description
4.	**Designated switch selection** A *designated switch* is the switch that forwards data for a network segment to the root bridge. STP determines which switch in a network segment has the lowest cost path to reach the root bridge; this switch becomes the designated switch. The designated switch is the only switch that forwards configuration BPDUs. Each network segment has only one designated switch. The port that connects a network segment to the designated switch is called the *designated port*. Designated ports are always in a forwarding state. The lowest path cost from a network segment to the root bridge determines which port is the designated port. All of the ports on a root bridge are designated ports.
5.	**Ports transition to a forwarding or blocked states** Root and designated ports are set to a forwarding state. These ports can send and receive BPDUs and data. STP sets all other ports on the switches in the VLAN to a blocked state. These ports can send and receive BPDUs; however, they cannot send or receive data.

Spanning tree network for VLANs

The following illustration shows an example of Spanning Tree network for two VLANs (VLAN 10 and VLAN 20) configured in the default vRouter in the e-commerce vSwitch. In VLAN 10, the root bridge is a switch in the VLAN. In VLAN 20, the e-commerce vSwitch is the root bridge. The forwarding paths are the paths that STP uses to transmit data; the blocked paths are in standby mode and do not transmit data.

Note that since STP runs per-VLAN, a link that is present in more than one VLAN may be forwarding for some VLANs, and blocked for others.

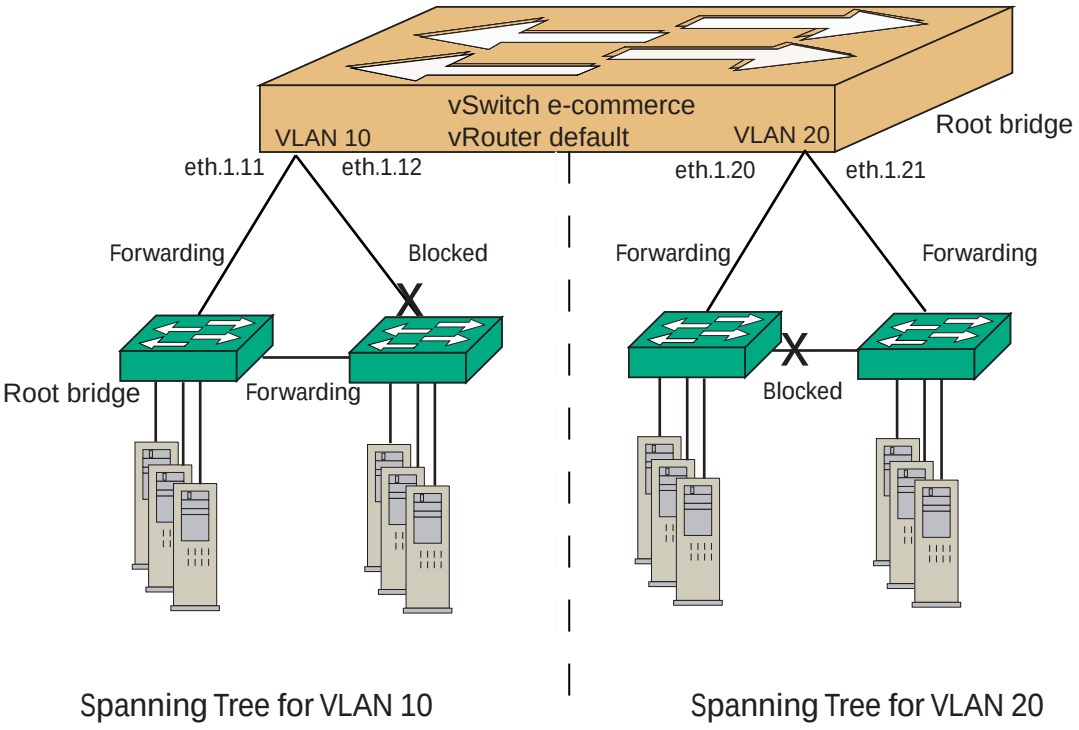


Figure 21-1. Example of Spanning Tree Network for two VLANs

VLAN command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `vRouter` *`name`* `vlan`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

VLAN command summary

Table 21-2 lists and briefly describes the VLAN commands.

Table 21-2. VLAN command summary

Command name	Description
`address flush`	Remove VLAN entries from the address table.
`address static`	Create or modify a static address entry for a VLAN.
`interface`	Create or modify a VLAN interface.
`interface spanningTree`	Configure ports to participate in a Spanning Tree network.
`show address`	Display VLAN addresses.
`show address static`	Display only static addresses.
`show interface`	Display VLAN interfaces.
`show interface spanningTree`	Display theSpanning Tree port configuration.
`show interface statistics`	Display VLAN interface statistics.
`show interface verbose`	Display details of the configuration of the VLAN.
`show spanningTree`	Display Spanning Tree bridge configuration.
`show statistics`	Display VLAN statistics.
`show`	Display VLAN configuration.
`show verbose`	Display detailed VLAN configuration information.
`spanningTree`	Configure the Spanning Tree Protocol.
`vlan`	Create or modify a VLAN; enter VLAN command mode.

VLAN basic configuration

The following procedure explains the basic steps for configuring a VLAN.

Table 21-3. Steps for configuring a VLAN

Step	Action
1.	Create a VLAN with the `vlan` command.
2.	Assign ports to the VLAN with the `interface` command.

Spanning Tree basic configuration

The following procedure explains the basic steps for configuring the STP.

Table 21-4. Steps for configuring STP

Step	Action
1.	Configure the VLAN for which you want to enable the Spanning Tree Protocol.
2.	Configure the bridge that the VLAN uses for the Spanning Tree. Use the `spanningTree` command.
3.	Configure the ports in the VLAN that participate in the Spanning Tree network topology. Use the vlan `interface spanningTree` command.

address flush

Purpose

Removes all dynamic MAC address entries from the VLAN's address table. This may be useful if your network is having problems, such as a loop, and you want to clean out the address cache to allow the interface to relearn addresses.

Access mode

config

Syntax

```
vSwitch name vRouter name vlan address flush
    [vlanId ID]
```

Arguments

Argument name	Description
`vlanId` *ID*	**Optional.** Removes addresses associated with the VLAN specified. Valid range is 1 through 4094.

Example

The following example displays the address table for VLAN 50, flushes the dynamic addresses, and displays the relearned addresses.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# show vlan 50 address
Vlan ID         MAC Address          IfName          Type
50              00:01:30:b5:26:80    eth.1.6         dynamic
50              00:07:82:00:00:80    eth.1.6         dynamic
50              00:07:82:00:03:87    N/A             permanent
50              00:07:82:00:04:40    eth.1.6         dynamic
50              00:07:82:00:04:ff    eth.1.6         dynamic
50              00:07:82:0e:0c:49    eth.1.6         dynamic
sun(config-vSwitch-system vRouter-management)# vlan 50 address flush
sun(config-vSwitch-system vRouter-management)# show vlan 50 address
```

```
Vlan ID         MAC Address          IfName          Type
50              00:07:82:00:03:87    N/A             permanent
sun(config-vSwitch-system vRouter-management)# show vlan 50 address
Vlan ID         MAC Address          IfName          Type
50              00:01:30:b5:26:80    eth.1.6         dynamic
50              00:07:82:00:03:87    N/A             permanent
50              00:07:82:00:04:ff    eth.1.6         dynamic
50              00:07:82:0e:0c:0b    eth.1.6         dynamic
50              00:07:82:0e:0c:49    eth.1.6         dynamic
sun(config-vSwitch-system vRouter-management)# show vlan 50 address
Vlan ID         MAC Address          IfName          Type
50              00:01:30:b5:26:80    eth.1.6         dynamic
50              00:07:82:00:03:87    N/A             permanent
50              00:07:82:00:04:40    eth.1.6         dynamic
50              00:07:82:00:04:ff    eth.1.6         dynamic
50              00:07:82:0e:0c:0b    eth.1.6         dynamic
50              00:07:82:0e:0c:49    eth.1.6         dynamic
50              00:80:ba:a0:08:80    eth.1.6         dynamic
```

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Vlan → address → flush

address static

Purpose

Creates or modifies a static MAC address entry for the VLAN.

Access mode

config

Syntax

To create a static address:

```
vSwitch name vRouter name vlan vlanId address static
   macAddress macAddress
   ifName ifName
```

To modify a static address:

```
vSwitch name vRouter name vlan vlanId address static
   macAddress macAddress
   [ifName ifName]
```

Arguments

Argument name	Description
`macAddress` *`macAddress`*	Specifies the physical MAC address of the entry. Enter a valid address in hex format.
`ifName` *`ifName`*	Specifies the name of the Ethernet port or LAG on which this MAC address is reachable. Interface names are entered in the format *`type.slot.portNumber`*. The interface name is the only value you can change when modifying a static address.

Delete filters

```
no vSwitch name vRouter name vlan vlanId address static
   macAddress macAddress
   [ifName ifName]
```

Example

The following example configures static MAC addresses on interfaces eth.1.22 and eth.1.30.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 12
sun(config-vSwitch-e-commerce vRouter-default vlan-12)# address static
macAddress 00:07:82:0e:ec:1f port eth.1.22
sun(config-vSwitch-e-commerce vRouter-default vlan-12)# address static
macAddress 00:07:82:0e:01:2f port eth.1.30
sun(config-vSwitch-e-commerce vRouter-default vlan-12)# show address
Vlan ID         MAC Address          IfName          Type
12              00:07:82:0e:01:2f    eth.1.30        static
12              00:07:82:0e:ec:1f    eth.1.22        static
```

Associated MIB

```
vlan.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → Vlan → address → static

interface

Purpose

Adds an Ethernet or LAG interface to a VLAN, creating a VLAN interface.

The no form of the command deletes the specified interface.

Access mode

config

Syntax

To create a VLAN interface:

```
vSwitch name vRouter name vlan vlanId interface
   ifName ifName
   [tagging {enabled | disabled}]
```

To modify a VLAN interface:

```
vSwitch name vRouter name vlan vlanId interface ifName
   [adminState {enabled | disabled}]
   [tagging {enabled | disabled}]
   [mtu integer]
   [packetTrace {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {enabled | disabled}]
   [description text]
```

Arguments

Argument name	Description
`Vlan Name`	The name of the VLAN.
`ifName` *`ifName`*	Specifies the name of the Ethernet port or LAG. Interface names are entered in the format *`eth.slot.portNumber`* or *`lag.lagId`*.
`tagging {enabled | disabled}`	**Optional**. Specifies whether the system should insert an 802.1Q-compliant tag to packets on egress. When `enabled`, all packets exiting this interface will include the identifying tag. The default is `disabled`.
The following arguments are available when modifying a VLAN interface:	
`adminState {disabled | enabled}`	**Optional**. Sets the administrative state of the named VLAN, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring it offline or preconfigure a VLAN before bringing it online. The default administrative state is `enabled`.
`mtu` *`integer`*	**Optional**. Sets the maximum transmission unit (MTU) of the interface. This is the maximum length, in number of bytes, of a packet transmitted over this interface. Packets are fragmented to this size. Valid range is 576 to 9000 bytes; the default MTU is 1500 bytes. This variable cannot be modified for VLAN interfaces.
`packetTrace {enabled | disabled}`	**Optional**. Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.

Argument name	Description
`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`	**Optional**. Sets the event filter level for the interface. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`. This argument is not available when creating an interface, only when modifying or deleting it.
`linkUpDownTrap {enabled | disabled}`	**Optional**. Specifies whether the system should generate a trap when this interface changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 5, “SNMP and Trap commands”.

Argument name	Description
`description` *`text`*	**Optional**. Assigns a text string to an interface. Use this command to identify an interface name (for example, lag.10) with a meaningful description. This string is displayed with the `show interface verbose` command.

Delete filters

See the `show interface verbose` command for argument descriptions.

```
no vlan interface vlanId
   ifName ifName
   [vlanIfName ifName]
   [adminState {enabled | disabled}]
   [status {up | down | testing | unknown | dormant}]
   [tagging {enabled | disabled}]
   [mtu integer]
   [speed integer]
   [lastChange integer]
   [packetTrace {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {enabled | disabled}]
   [description text]
   [ifIndex hexinteger]
```

Example

The following example shows how to add a VLAN interface to port eth.1.15. The VLAN uses tagging for packets.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 25 port eth.1.15
tagging enabled
sun(config-vSwitch-e-commerce vRouter-default)#
```

(continued)

The following example shows how to modify the VLAN to set the event filter level to warning and enable link up or down traps.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 25 eventFilter
warning linkUpDownTrap enabled
```

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → interface → add
- vSwitch → *name* → vRouter → *name* → vlan → interface → copy
- vSwitch → *name* → vRouter → *name* → vlan → interface → modify

interface spanningTree

Purpose

Configures the VLAN ports that are participating in a Spanning Tree network.

Access mode

config

Syntax

```
vSwitch-name vRouter-name vlan interface spanningTree
   ifName ifName
   [priority integer]
   [adminState {disabled | enabled}]
   [portfast {disabled | enabled}]
   [pathCost integer]
   [rootGuard {disabled | enabled}]
   [bpduGuard {disabled | enabled}]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the VLAN interface name the system uses for spanning tree
`priority` *`integer`*	**Optional.** Configures the priority for a port in the VLAN for STP. The valid range is from 0 through 255; the default setting is 128. The higher the value, the lower the priority; the highest priority is 0. If a loop occurs, STP uses this priority value to determine which port becomes a designated port.
`adminState {disabled | enabled}`	**Optional.** Indicates whether STP is enabled for this port. If `enabled`, the port participates in the spanning tree topology; if `disabled`, the port does not participate in the spanning tree topology. The default setting is `enabled`.

Argument name	Description
`portfast {disabled | enabled}`	**Optional.** Specifies whether this port enters the fowarding state immediately. If `enabled,` the port enters the forwarding state once it is connected to a device; if `disabled`, the port transitions through the normal STP port states. The default setting is `disabled`. When the port is in the forwarding state, it continues to process BPDUs to learn the network topology. If loops exists, the port detects the loop in 2 x Forward Delay (by default, 30 seconds). **Important:** Enable this feature *only* if the port is connected to a single host or workstation. If you enable this feature on a port connected to a network device (such as another switch), you can create loops in the network topology.
`pathCost` *`integer`*	**Optional.** Sets the path cost for spanning-tree calculations. The lower the cost, the higher the priority. The valid range is from 1 through 65535; the default setting depends on the port speed. Typically, you can calculate the path cost by dividing the port speed by 1000. Common path costs are: • For a 10Mbps port, the cost is 100. • For a 100Mbps port, the cost is 19. • For a 1000Mbps port, the cost is 4 If a loop occurs, STP considers the path cost when selecting a port to place in a forwarding state. Set a lower value for ports that you want spanning tree to use to forward data.

Argument name	Description
`rootGuard {disabled | enabled}`	**Optional.** Specifies whether this port uses the Root Guard feature. If `enabled` and port receives a BDPU from a device trying to become the root bridge, the port state changes to `rootInconsistent` and the port does not forward traffic until the port stops receiving BPDUs that would cause the root bridge to change. If `disabled`, the port accepts the BDPU and the device sending the BDPU can, potentially, become the root bridge. The default setting is `disabled`. Enabling Root guard allows a device to participate in a spanning tree network as long as it does not try to become the root bridge. If using this feature, enable this feature on *all* ports on the switches
`bpduGuard {disabled | enabled}`	**Optional.** Specifies whether the port uses the Port Fast BDPU Guard feature. If `enabled`, the device connected to the port cannot influence the spanning tree network topology. If the port receives a BDPU from the connected device, it blocks the port until the device stops sending configuration BDPUs. The default setting is `disabled`. You must set the `portfast` argument to `enable`, if you want to use `PortFastBDPUGuard`. If `disabled for a port` with `portFast` enabled, the device can, potentially, become the root bridge, which may not be optimal for the network.

Example

The following example enables STP on the port, eth.1.12 (which is in VLAN 12). It sets the port to use Port Fast and Port Fast BPDU Guard because it is connected to a workstation in the VLAN. For all other arguments, this configuration uses the default values.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 12
sun(...vRouter-default vlan.12)# interface
sun(...vlan.12 interface)# spanningTree ifName eth.1.12 adminState
enabled portfast enabled bdpuGuard enabled
sun(...vlan.12 interface)# show spanningTree
IfName:                 eth.1.12
Priority:               128
Spanning Tree State:    enabled
Admin State:            enabled
Port Fast:              enabled
Root BPDU Guard:        disabled
Port Fast BPDU Guard:   enabled
Path Cost:              19
Designated Root:        80:00:00:07:82:0e:0c:17
Designated Cost:        0
Designated Bridge:      80:00:00:07:82:0e:0c:17
Designated Port:        80:01
Forwarding Transitions: 5
```

Associated MIB

stp.mib

Web path

- vSwitch → *name* → vRouter → *name* → vlan → interface → spanningTree → modify

show address

Purpose

Displays a summary of all dynamic and static MAC addresses in the address table.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan address`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> show vlan address
Vlan ID         MAC Address          IfName          Type
130             00:03:47:42:4e:53    eth.1.21        dynamic
130             00:03:47:42:4f:f1    eth.1.23        dynamic
130             00:07:82:00:09:d6    N/A             permanent
130             00:07:82:00:09:d7    N/A             permanent
130             00:07:82:00:09:d8    N/A             permanent
130             00:07:82:00:09:d9    N/A             permanent
130             00:07:82:00:09:da    N/A             permanent
130             00:07:82:00:09:db    N/A             permanent
130             00:07:82:00:09:dc    N/A             permanent
130             00:07:82:00:09:dd    N/A             permanent
131             00:02:b3:2f:53:b0    eth.1.42        dynamic
131             00:03:47:42:4e:54    eth.1.44        dynamic
131             00:03:47:42:4f:f2    eth.1.43        dynamic
131             00:03:47:42:50:52    eth.1.41        dynamic
131             00:07:82:00:09:ea    N/A             permanent
131             00:07:82:00:09:eb    N/A             permanent
131             00:07:82:00:09:ec    N/A             permanent
131             00:07:82:00:09:ed    N/A             permanent
```

Output description

Field name	Description	Filter name
`Vlan ID`	The unique identifier for the VLAN. Valid range is 1 through 4094.	`vlanId` *`ID`*
`MAC Address`	The physical MAC address of the entry. The physical address of a device reachable through this interface.	`macAddress` *`macAddress`*
`IfName`	The name of the Ethernet port or LAG.	`ifName` *`ifName`*
`Type`	The type of address associated with the VLAN interface, one of the following: `dynamic`: a dynamically learned address `static`: a manually configured address `permanent`: the port's physical address or other internal address	`type {dynamic | static | permanent}`

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → address

show address static

Purpose

Displays all static addresses configured in the VLAN.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan address static`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> vlan 12
sun(config-vSwitch-e-commerce vRouter-default vlan-12)> show address
static
Vlan ID:     12
MAC Address: 00:07:82:0e:01:2f
IfName:      eth.1.30

Vlan ID:     12
MAC Address: 00:07:82:0e:0f:1f
IfName:      eth.1.22

Vlan ID:     12
MAC Address: 00:07:82:0e:ec:1f
IfName:      eth.1.30
```

Output Description

Field name	Description	Filter name
`Vlan Name`	The name of the VLAN.	`vlanName Name`
`MAC Address`	The physical MAC address of the entry.	`macAddress` *`macAddress`*
`IfName`	The interface name of the Ethernet port or LAG out which the MAC address is reachable.	`ifName` *`ifName`*

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → address → static

show interface

Purpose

Displays configured and operational characteristics of VLAN interfaces.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan interface`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> vlan 12
sun(config-vSwitch-e-commerce vRouter-default vlan-12)> show interface
Vlan ID    IfName          Admin State  Oper Status  Tagging
12         eth.1.22        enabled      down         enabled
12         eth.1.23        enabled      down         enabled
12         eth.1.24        enabled      down         enabled
12         eth.1.30        enabled      down         enabled
```

Output description

Field name	Description	Filter name
`Vlan Name`	The name of the VLAN.	`vlanName` *`Name`*
`IfName`	The name of the Ethernet port or LAG.	`ifName` *`ifName`*
`Admin State`	The administrative state of the named VLAN, either `enabled` or `disabled`.	`adminState {disabled | enabled}`

Field name	Description	Filter name
`Oper Status`	The operational status of the VLAN interface. This is the state the VLAN interface is in, not necessarily the state you had configured for the interface. Possible values: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`status {up | down | testing | unknown | dormant}`
`Tagging`	Whether the system is inserting an 802.1Q-compliant tag to packets on egress. When `enabled`, all packets exiting this interface include the identifying tag. The default is `disabled`.	`tagging {enabled | disabled}`

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → interface

show interface spanningTree

Purpose

Displays the Spanning Tree port configuration for the ports in a VLAN.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan interface spanningTree`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> vlan 12
sun(vSwitch-e-commerce vRouter-default vlan-12)> show interface
spanningtree
IfName:                 eth.1.12
Priority:               128
Spanning Tree State:    enabled
Admin State:            enabled
Port Fast:              disabled
Root BPDU Guard:        disabled
Port Fast BPDU Guard:   disabled
Path Cost:              19
Designated Root:        80:00:00:07:82:0e:0c:29
Designated Cost:        0
Designated Bridge:      80:00:00:07:82:0e:0c:29
Designated Port:        80:01
Forwarding Transitions: 0
```

Output Description

Field name	**Description**	**Filter name**
`IfName`	The port associated with a VLAN.	`ifName` *`IfName`*
`Priority`	The preference that STP gives this port relative to other ports that it uses to forward traffic on the network. A higher numerical value specifies a lower priority; the highest priority is 0.	`priority` *`integer`*

Field name	Description	Filter name
`Spanning Tree State`	The STP stat of the port. Possible values are: `disabled`: The port is not participating in STP. This can occur when the port is disconnected or you disable STP on the port. `blocking`: The port is in standby mode and does not send or receive data, but continues to receive BPDUs. STP places a port in this state to prevent a loop. `listening`: The port is listening for a BPDU from neighboring switches so it can determine the network topology. The port does not write MAC addresses to its internal table during this state. I also does not receive or forward data during this state. `learning`: The port is listening for BDPUs from neighboring switches and can write the MAC addresses it learns in its internal table. The port does not transmit or receive data during this state. `forwarding`: STP is allowing the port to send and receive data. `rootInconsistent`: The port has Root BDPU Guard enabled and the port received a BDPU from a connected device. This state is equivalent to the listening state. `errDisable`: This port Port Fast BDPU Guard enabled and the device connected to it sent a BDPU to the port. This is a disabled state	`state {disabled | blocking | listening | learning | forwarding | broken | rootInconsistent | errDisable}`
`Admin State`	The administrative state of the STP configuration for the port. If `enabled`, the configuration is operational; if `disabled`, the system does not use the configuration.	`adminState {disabled | enabled}`
`Port Fast`	Indicates whether the port enters the forwarding state immediately, bypassing the listening and learning states.	`portfast {disabled | enabled}`

Field name	Description	Filter name
`Root BPDU Guard`	Indicates whether the port uses the Root BPDU feature..	`rootGuard {disabled | enabled}`
`BDPU Guard`	Indicates whether the port uses the BDPU Guard feature.	`bpduGuard {disabled | enabled}`
`Path Cost`	Indicates the path cost to the spanning tree's root switch for the VLAN ports. A higher numerical value means a lower priority; the highest priority is 0. Each port on a switch adds a cost to the path that a frame must travel to reach the root bridge.	`pathCost integer`
`Designated Root`	The root bridge identifier (ID) for the spanning tree network. The ID consists of a 2-byte priority value followed by a 6-byte MAC address.	`designatedRoot` *`physicalAddress`*
`Designated Cost`	The path cost from the designated port connected to a network segment to the root bridge. If the designated port is on the root bridge, the cost is 0.	`designatedCost` *`integer`*
`Designated Bridge`	The bridge on a network segment that forwards traffic to the root bridge.	`designatedBridge` *`physicalAddress`*
`Designated Port`	The identifier for the port on the designated bridge for this network segment of the spanning tree that communicates with the root bridge.	`designatedPort` *`physicalAddress`*
`Forwarding Transitions`	The number of times this port has transitioned from the learning state to the forwarding state.	`forwardTransitions` *`integer`*

Associated MIB

`stp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → interface → spanningTree

show interface statistics

Purpose

Displays statistics sorted by interface. For statistics representing the entire VLAN, use the `show statistics` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan interface statistics`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> show vlan interface
statistics
Vlan ID:            131
IfName:             eth.1.42
Received Frames:    4915
Transmitted Frames: 4804
Discarded Receives: 0
Over MTU:           0

Vlan ID:            131
IfName:             eth.1.43
Received Frames:    1519
Transmitted Frames: 1488
Discarded Receives: 0
Over MTU:           0

Vlan ID:            131
IfName:             eth.1.44
Received Frames:    1649
Transmitted Frames: 1645
Discarded Receives: 0
Over MTU:           0
```

Output description

Field name	Description	Filter name
`Vlan Name`	The name of the VLAN.	`vlanName` *`Name`*
`IfName`	The name of the Ethernet port or LAG.	`ifName` *`ifName`*
`Received Frames`	The total number of frames received over this VLAN interface.	`inFrames` *`integer`*
`Transmitted Frames`	The total number of frames transmitted over this VLAN interface.	`outFrames` *`integer`*
`Discarded Receives`	The total number of packets discarded on the VLAN interface. Reasons for discard include: • the packet was sent to a reserved multicast address • the packet arrived untagged and the port's default VLAN is the discard VLAN • the packet arrived tagged with the discard VLAN • the packet arrived tagged for a VLAN to which the port is not a member • the port is not in a forwarding state	`inDiscards` *`integer`*
`Over MTU`	The total number of packets discarded because the MTU size was exceeded and the `Dont Frag` bit was set.	`mtuExceeded` *`integer`*

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → interface → statistics

show interface verbose

Purpose

Displays interface configuration data for either all or specified IP interfaces in the routing table. For a brief display of interfaces, which allows a horizontal view with multiple interface listings on a single screen, use the `show interface` command. Use the `interface` command to set the user-configurable parameters in this display.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan interface verbose`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> vlan 12
sun(vSwitch-e-commerce vRouter-default vlan-12)> show interface
verbose
Vlan ID:      12
IfName:       eth.1.22
Admin State:  enabled
Oper Status:  down
Tagging:      enabled
VlanIfName:   vlan.12.2
IfIndex:      0x500C1002
Link Traps:   enabled
Evt Filter:   critical
Pkt Trace:    disabled
MTU:          1500
Speed:        100000000
Last Change:  81954352
Description:  ipRouter

Vlan ID:      12
IfName:       eth.1.24
Admin State:  enabled
Oper Status:  down
Tagging:      enabled
VlanIfName:   vlan.12.4
IfIndex:      0x500C1004
Link Traps:   enabled
```

```
Evt Filter:   critical
Pkt Trace:    disabled
MTU:          1500
Speed:        100000000
Last Change:  84680939
Description:

Vlan ID:      12
IfName:       eth.1.30
Admin State:   enabled
Oper Status:  down
Tagging:      enabled
VlanIfName:   vlan.1.1
IfIndex:      0x500C1001
Link Traps:   disabled
Evt Filter:   informational
Pkt Trace:    disabled
MTU:          1500
Speed:        100000000
Last Change:  81003876
Description:  jpeg
```

Output description

Field name	Description	Filter name
`Vlan Name`	The name of the VLAN.	`vlanName` *`Name`*
`IfName`	Specifies the name of the Ethernet port or LAG.	`ifName` *`ifName`*
`Admin State`	The administrative (manually configured) status of the interface, either `enabled` or `disabled`. Use the `interface` command to change the administrative status of an interface.	`adminState {enabled | disabled}`

Field name	Description	Filter name
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`operStatus {up | down | testing | unknown | dormant}`
`Tagging`	Whether the system is inserting an 802.1Q-compliant tag to packets on egress. When `enabled`, all packets exiting this interface on this VLAN include the identifying tag. The default is `disabled`.	`tagging {enabled | disabled}`

Field name	Description	Filter name
`VlanIfName`	The port matching the system-generated identifier for the VLAN. Note that there are no configuration commands that operate on these ifIndex assignments, they are strictly internal reference points. The format is `vlan.`*`vlanID.downlayer`*, where: *`vlanID`*: the numeric identifier you assigned to the VLAN with the `vlan` command. *`downlayer`*: an interface index that the system assigns to port when it is added to the VLAN. For example, a value of `vlan.3.2` equates to the second port added to the vlan you specified as VLAN 3. The value of `IfIndex` can be displayed with the `show vlan` command.	`vlanIfName` *`ifName`*
`IfIndex`	The hexadecimal equivalent of the VLAN interface name. Use this value within SNMP to specify an interface name.	`ifIndex` *`hexInteger`*
`Link Traps`	The setting for whether the system should generate a trap when this interface changes state (from up to down, or vice versa). The default setting is `disabled`.	`linkUpDownTrap {enabled | disabled}`

Field name	Description	Filter name
`Evt Filter`	The event filter level set for the interface, one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default level is `informational`.	`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`
`Pkt Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.	`packetTrace {enabled | disabled}`

Field name	Description	Filter name
MTU	The maximum transmission unit (MTU) of the interface. This is the maximum length, in number of bytes, of a packet transmitted over this interface. Packets are fragmented to this size. Valid range is 576to 9000 bytes; the default MTU is 1500 bytes.	mtu *integer*
Speed	The speed of the interface. This value is derived from the speed of the Ethernet port (at the bottom of the stack) that is connected to this interface.	speed *integer*
Last Change	The value, in ticks, of the sysUpTime variable at the time of the last change to the interface. This could be a change to the configuration, or in the administrative state or operational status. If the field displays 0, there have been no changes to the interface since the last system reboot.	lastChange *integer*
Description	The text string you assigned to the interface. You can use this value to identify a interface name (for example, lag.10) with a meaningful description.	description *text*

Associated MIB

vlan.mib

Web path

- vSwitch → *name* → vRouter → *name* → vlan → interface → verbose

show spanningTree

Purpose

Displays the current Spanning Tree bridge configuration for a VLAN.

Access mode

user

Syntax

show vSwitch-*name* vRouter-*name* vlan spanningTree

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 10
sun(config-vSwitch-e-commerce vRouter-default vlan-10)# show
spanningTree

Vlan Id:                    10
Admin State:                disabled
Priority:                   32768
Time Since Topology Change: 51552334
Topology Changes:           1
Designated Root:            80:00:00:07:82:0e:0c:29
Root Cost:                  0
Root Port:                  0
Max Age:                    20
Hello Time:                 2
Hold Time:                  1
Forward Delay:              15
Bridge Max Age:             20
Bridge Hello Time:          2
Bridge Forward Delay:       15
```

Output description

Field name	Description	Filter name
`Vlan ID`	The index identifying the VLAN.	No filter available for this field
`Admin State`	Indicates whether the VLAN participates in a spanning tree network.	`adminState {disabled | enabled}`
`Priority`	The priority assigned to the bridge for this VLAN. The lower the value, the higher the priority. STP uses this value to determine the root bridge in the spanning tree topology.	`priority` *`integer`*
`Time Since Topology Change`	The time, in hundredths of a second, since the spanning tree network topology changed.	`timeSinceTopologyChange` *`integer`*
`Topology Changes`	The number of topology changes that have occurred.	`topologyChanges` *`integer`*
`Designated Root`	The root bridge identifier (ID), consisting of a two-byte priority and a 6-byte MAC address.	`designatedRoot` *`Physical Address`*
`Root Cost`	The cost of the path to the root bridge for this switch. If this bridge is the root bridge, the cost is 0.	`rootCost` *`integer`*
`Root Port`	The port that this switch uses to reach the root bridge. This port has the lowest cost path between the switch and the root bridge.	`rootPort` *`IfName`*
`Max Age`	The maximum amount of time, in seconds, that the switch waits without receiving a configuration BPDU before it reconfigures the spanning tree topology. If this switch is the root bridge, all switches in the network use this value.	`maxAge` *`seconds`*

Field name	Description	Filter name
`Hello Time`	The Hello Time is the time interval, in seconds, between a root bridge's transmissions of a configuration bridge protocol data unit (BPDU) to other devices. If this switch is the root bridge, all switches in the network use this value.	`helloTime` *`seconds`*
`Hold Time`	The amount of time between this bridge's transmissions of configuration BPDUs to other devices on the network.	`holdTime` *`seconds`*
`Forward Delay`	The Forward Delay is the time, in seconds, that a port spends in the listening and learning states before it changes to a forwarding state. If this switch is the root bridge, all switches in the network use this value.	`forwardDelay` *`seconds`*
`Bridge Max Age`	The current value of the Bridge Max Age. The root bridge sets this value for the other switches in the network. The Bridge Max Age is the maximum amount of time, in seconds, that the switch waits without receiving a configuration BPDU before it reconfigures the STP network.	`bridgeMaxAge` *`seconds`*

Field name	Description	Filter name
`Bridge Hello Time`	The current value of the Bridge Hello Time. The root bridge sets this value for the other switches in the network. The Bridge Hello Time is the time interval, in seconds, between a root bridge's transmissions of configuration bridge protocol data units (BPDUs) to other devices.	`bridgeHelloTime` *`seconds`*
`Bridge Forward Delay`	The current value of the Bridge Forward Delay. The root bridge sets this value for the other switches in the network. The Bridge Forward Delay is the time, in seconds, that a port wait waits before it changes from a learning state to a forwarding state.	`bridgeForwardDelay` *`seconds`*

Associated MIB

`stp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → spanningTree

show statistics

Purpose

Displays statistics for one or more VLANs. For interface-specific statistics, use the `show interface statistics` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)> show vlan statistics
Vlan ID:            130
Received Frames:    13048
Transmitted Frames: 0
Discarded Receives: 0

Vlan ID:            131
Received Frames:    10409
Transmitted Frames: 10314
Discarded Receives: 0
```

Output description

Field name	Description	Filter name
`Vlan Name`	The name of the VLAN.	`vlanName` *`Name`*
`Received Frames`	The total number of frames received over this VLAN.	`inFrames` *`integer`*
`Transmitted Frames`	The total number of frames transmitted over this VLAN.	`outFrames` *`integer`*

Field name	Description	Filter name
`Discarded Receives`	The total number of packets discarded on the VLAN. Reasons for discard include: • the packet was sent to a reserved multicast address • the packet arrived untagged and the port's default VLAN is the discard VLAN • the packet arrived tagged with the discard VLAN • the packet arrived tagged for a VLAN to which the port is not a member • the port is not in a forwarding state	`inDiscards` *`integer`*

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → statistics

show

Purpose

Displays general configuration for the specified VLAN(s).

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# vlan 12
sun(vSwitch-e-commerce vRouter-default vlan-12)# show
Vlan ID    Admin State  Oper Status  Learning
12         enabled      down         enabled
```

Output description

Field name	Description	Filter name
`Vlan Name`	The unique name for the VLAN. VLAN names must be unique across the entire switch (you cannot use the same name within different vRouters on the system).	`vlanName` *`Name`*
`MAC Address`	The configured MAC address that appears to be reachable from that port.	
`IfName`	The name of the Ethernet port or LAG. Interface names are entered in the format type.slot.portNumber.	
`Type`	The type of VLAN address: static, dynamic, or permanent.	

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → vlan list

show verbose

Purpose

Displays detailed configuration for the specified VLAN(s).

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vlan verbose`

Sample output

```
sun> vswitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# show vlan verbose
Vlan ID:      12
Admin State:  enabled
Oper Status:  down
Learning:     enabled
VlanName:     vlan.12
IfIndex:      0x500C0000
Link Traps:   disabled
Event Filter: informational
Packet Trace: disabled
Last Change:  80979614
Description:  ipRouter
```

Output Description

Field name	Description	Filter name
`Vlan Name`	The name of the VLAN.	`vlanName` *`text`*
`VLAN ID`	The numeric ID of the VLAN.	`vlanID` *`integer`*
`Admin State`	The administrative state of the named VLAN, either `enabled` or `disabled`.	`adminState {enabled | disabled}`

Field name	Description	Filter name
`Oper Status`	The operational status of the VLAN. This is the state the VLAN is in, not necessarily the state you had configured for the VLAN. Possible values: `up`: The VLAN is operational `down`: The VLAN is not operational `testing`: The VLAN is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The VLAN is not in any of the listed states `dormant`: The VLAN is uninitialized, the configuration database and hardware have not yet become synchronized	`status {up | down | testing | unknown | dormant}`
`Learning`	The setting for the VLAN's ability to learn and store addresses as they are received (dynamically). You can display the address learning method with the `Type` field of the `show address` command. If `enabled`, the VLAN stores addresses in its address table. If `disabled`, addresses are not stored.	`learning {enabled | disabled}`
`Bridge Mode Load Balancing`	The setting for bridge mode load balancing.	`bridgeModeLoadBalancing {enabled | disabled}`

Field name	Description	Filter name
`VlanIfName`	The name matching the system-generated identifier for the VLAN. The format is `vlan.`*`vlanID`*, where: *`vlanID`*: the numeric identifier you assigned to the VLAN with the `vlan` command. For example, a value of `vlan.3` equates to the vlan you specified as VLAN 3.	`vlanIfName` *`ifName`*
`IfIndex`	The hexadecimal equivalent of the VLAN name. Use this value within SNMP to specify a VLAN name.	`ifIndex` *`hexInteger`*
`Link Traps`	The setting for whether the system should generate a trap when this VLAN changes state (from up to down, or vice versa). The default setting is `disabled`.	`linkUpDownTrap {enabled | disabled}`

Field name	Description	Filter name
`Event Filter`	The event filter level set for the VLAN, one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default level is `informational`.	`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`
`Packet Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.	`packetTrace {enabled | disabled}`

Field name	Description	Filter name
`Last Change`	The value, in ticks, of the `sysUpTime` variable at the time of the last change to the VLAN. This could be a change to the configuration, or in the administrative state or operational status. If the field displays 0, there have been no changes to the IP instance since the last system reboot.	`lastChange` *`integer`*
`Description`	The text string you assigned to the VLAN. You can use this value to identify a VLAN name (for example, vlan.10) with a meaningful description.	`description` *`text`*

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → verbose

spanningTree

Purpose

Configures the Spanning Tree Protocol (STP) for the bridge associated with a VLAN. You can associate one bridge instance with each VLAN.

Access mode

config

Syntax

```
vSwitch name vRouter name vlan vlanID spanningTree
   [adminState {disabled | enabled}]
   [priority integer]
   [bridgeMaxAge seconds]
   [bridgeHelloTime seconds]
   [bridgeForwardDelay seconds]
```

Output Description

Argument name	Description
`adminState {disabled | enabled}`	**Optional.** Specifies whether the VLAN uses the Spanning Tree Protocol. If `enabled`, the VLAN uses Spanning Tree; if `disabled`, the VLAN does not use Spanning Tree. The default setting is `disabled`.
`priority` *`integer`*	**Optional.** Sets the priority for this bridge. The valid range is from 0 to 65535; the default setting is 37268. The spanning tree protocol uses this value to select the root bridge in the VLAN. The lower the priority value, the higher the priority assigned to the bridge.

Argument name	Description
`bridgeMaxAge` *`seconds`*	**Optional.** Sets the maximum amount of time, in seconds, that a bridge waits without receiving a configuration bridge protocol data unit before it reconfigures the STP network. The valid range is from 6 through 40 seconds; the default setting is 20 seconds. The value for this argument should be greater than or equal to 2 x (Hello Time + 1) and less than or equal to 2 x (Forward Delay -1). If this switch is the root bridge, all other switches in the network use this value.
`bridgeHelloTime` *`seconds`*	**Optional.** Sets the time, in seconds, that the root bridge waits to transmit a configuration bridge protocol data unit (BPDU) to other devices. The valid range is from 1 through 10 seconds; the default setting is 2 seconds. If this switch is the root bridge, all other switches in the network use this value.
`bridgeForwardDelay` *`seconds`*	**Optional.** Sets the time, in seconds, that a port spends in the listening and learning states before it changes to a forwarding state. The valid range is from 4 through 30 seconds; the default setting is 15 seconds. If this switch is the root bridge, all other switches in the network use this value.

Example

The following example shows how to enable Spanning Tree for VLAN 12 in the default vRouter in the marketing vSwitch. This example sets the bridge priority lower than the default so that this vSwitch is likely to become the root bridge in the Spanning Tree topology. This configuration uses the defaults for the bridge timers.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 12
sun(...vRouter-default vlan-12)# spanningTree adminState enabled
priority 1000
sun(...vRouter-default vlan-12)# show spanningTree
Vlan ID:                    12
Admin State:                enabled
Priority:                   1000
Time Since Topology Change: 410812
Topology Changes:           5
Designated Root:            03:e8:00:07:82:0e:0c:17
Root Cost:                  0
Root Port:                  0
Max Age:                    20
Hello Time:                 2
Hold Time:                  1
Forward Delay:              15
Bridge Max Age:             20
Bridge Hello Time:          2
Bridge Forward Delay:       15
```

Associated MIB

```
stp.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → vlan → spanningTree → add
- vSwitch → *name* → vRouter → *name* → vlan → spanningTree → modify
- vSwitch → *name* → vRouter → *name* → vlan → spanningTree → copy

vlan

Purpose

Creates a new VLAN with the specified number as an ID and enters VLAN configuration mode. From this command you can configure several characteristics of the VLAN. All ports assigned to the VLAN inherit these characteristics. If you have previously configured these characteristics with the `port` command (described in Chapter 19, "Ethernet data port commands"), the settings configured with this command override those settings. Any of these characteristics set with the `interface` command take precedence over this command's settings.

The `no` form deletes the specified VLAN. Values that are available for deleting a VLAN can be viewed with the various `show` commands.

Access mode

config

Syntax

To create a VLAN:

```
vSwitch name vRouter name vlan
   vlanName text
   vlanId integer
   [learning {enabled | disabled}]
   [bridgeModeLoadBalancing {enabled | disabled}]
```

To modify a VLAN:

```
vSwitch name vRouter name vlanID integer
   [adminState {enabled | disabled}]
   [learning {enabled | disabled}]
   [linkTraps {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [packetTrace {enabled | disabled}]
   [description text]
   [bridgeModeLoadBalancing {enabled | disabled}]
```

Arguments

Argument name	Description
`vlanName` *`text`*	Specifies the name of the VLAN.
`vlanID` *`integer`*	Specifies the numeric ID of the VLAN. Valid values are 1 through 4095.
`learning {enabled | disabled}`	**Optional.** Sets whether the VLAN learns and stores addresses as they are received (dynamically). You can display the address learning method with the `Type` field of the `show address` command. If `enabled`, the VLAN stores addresses in its address table. If `disabled`, addresses are not stored. The default is `enabled`.
`bridgeModeLoadBalancing`	**Optional.** Enables bridge mode load balancing. The default is `disabled`.
The following arguments are only available when modifying a VLAN configuration:	
`adminState {disabled | enabled}`	**Optional.** Sets the administrative state of the named VLAN, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring it offline or preconfigure a VLAN before bringing it online. The default administrative state is `enabled`.
`linkTraps {enabled | disabled}`	**Optional.** Specifies whether the system should generate a trap when this VLAN changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 5, “SNMP and Trap commands”.

Argument name	Description
`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`	**Optional.** Sets the event filter level for the VLAN. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`. This argument is not available when creating a VLAN, only when modifying or deleting it.
`packetTrace {enabled | disabled}`	**Optional.** Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.

Argument name	Description
`description` *text*	**Optional.** Associates a text description with the specified VLAN. The description can be up to 64 characters, and is displayed with the `show vlan verbose` command. If the description contains multiple words separated by spaces, enclose it in quotation marks.

Delete filters

See the `show verbose` command for argument descriptions.

```
no vlan vlanId
   [vlanName]
   [adminState {enabled | disabled}]
   [operStatus {up | down | testing | unknown | dormant}]
   [learning {enabled | disabled}]
   [linkTraps {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [lastChange integer]
   [packetTrace {enabled | disabled}]
   [description text]
   [ifIndex hexinteger]
```

Example

The following example creates VLAN 15 and then modifies it by adding a text description.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan wxyz 15 learning
disabled

sun(config-vSwitch-e-commerce vRouter-default)# show vlan
Vlan Name   Vlan ID    Admin State  Oper Status  Learning
abcd        12         enabled      down         enabled
wxyz        15         enabled      down         disabled
sun(config-vSwitch-e-commerce vRouter-default)# vlan 15 description
webSurf
```

```
sun(config-vSwitch-e-commerce vRouter-default)# show vlan verbose
Vlan Name                       abcd
Vlan ID:                        12
Admin State:                    enabled
Oper Status:                    down
Learning:                       enabled
Bridge Mode Load Balancing:     disabled
VlanIfName:                     vlan.abcd
IfIndex:                        0x500C0000
Link Traps:                     disabled
Event Filter:                   informational
Packet Trace:                   disabled
Last Change:                    80979614
Description:

Vlan Name                       wxyz
Vlan ID:                        15
Admin State:                    enabled
Oper Status:                    down
Learning:                       enabled
Bridge Mode Load Balancing:     disabled
VlanIfName:                     vlan.wxyz
IfIndex:                        0x538C0000
Link Traps:                     disabled
Event Filter:                   informational
Packet Trace:                   disabled
Last Change:                    89295306
Description:                    webSurf
```

Associated MIB

`vlan.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vlan → vlan list→ add

Chapter 22. IP interface commands

IP description

This chapter describes the Internet Protocol (IP) address assignment and interface configuration commands available from the N2000 Series system. The system supports IP Version 4. The N2000 Series supports several other IP-related command functions, which are detailed in the following chapters.

IP Feature	Chapter
Routing Information Protocol (RIP) and static routing	Chapter 24.
• Address Resolution Protocol (ARP) • Internet Control Message Protocol (ICMP) • Internet Domain Routing Protocol (IRDP)	Chapter 26.
Ping and traceroute utilities	Chapter 27.
Access control lists (ACLs)	Chapter 28.

IP interfaces

Interfaces within the N2000 Series system are “stacked” in a layer model. That is, the lowest layer is the Ethernet port. On top of that, you can build LAGs or VLANs, or run IP directly. Those can, in turn, be associated with an IP interface. You must configure the lower layers before you try to assign IP interfaces. The actions and statistics supported by the IP commands are based on the objects defined in the IF-MIB for network interfaces, the extension to RFC 1229.

IP command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `vRouter` *`name`* `ip`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

IP command summary

Table 22-1 lists and briefly describes the IP commands.

Table 22-1. IP command summary

Command name	Description
`address`	Assign an IP address to an interface.
`interface`	Create an IP interface on the vRouter.
`ip`	Configure IP characteristics on the vRouter.
`show address`	Display a summary of the configured IP address-to-interface association.
`show interface`	Display IP interface configuration data horizontally.
`show interface verbose`	Display IP interface configuration data vertically.
`show`	Display the current configuration for the instance of IP on the vRouter.
`show redirectTraffic`	Display the IP addresses that the system uses for redirected VSRP traffic.
`show verbose`	Display more details of the configuration for the instance of IP on the vRouter.
`show statistics`	Display standard statistics for this instance of IP (statistics from the IP MIB).

IP interface configuration

The following procedure defines the basic steps for configuring an IP interface.

Table 22-2. Steps for configuring IP Interfaces

Step	Action
1.	Define the lower layer of the connection, either a LAG or VLAN. (If you are connected to the Ethernet port, this does not require any additional configuration.)
2.	Associate the lower layer with the IP interface with the `interface` command.
3.	Assign an IP address to the interface with the `address` command.

address

Purpose

Assigns an IP address, and optionally, subnet mask to an IP interface. You must first define the lower layer connection (with the `interface` command), and then you can assign these addresses. IP is enabled on every IP interface by default.

The no form of the command removes the IP address from the IP interface.

Access mode

config

Syntax

```
vSwitch-name vRouter-name ip address
   ifName ifName
   ipAddr ipAddress
   [netMask ipAddress]
   [vsrpRedirect {enabled | disabled}]
```

Arguments

Argument name	Description
`ifName` *ifName*	Specifies the name of the interface to which you want to assign an IP address. (the interface that connects to the IP instance). This could be either `vlan.`*x*, `lag.`*x*, `eth.`*x*`.`*x*., `loopback, or ip.`*vswitch*`:`*vrouter*`.` If the interface does not already exist, the system returns an error. Use the `show interface` command to verify configured IP-to-lower layer interface associations. Use the `interface` command to configure new associations.
`ipAddr` *ipAddress*	Specifies the IP address to assign to the interface.
`netMask` *ipAddress*	**Optional.** Specifies the subnet mask, which identifies the network portion of the address. The default is the natural subnet mask.

Argument name	Description
`vsrpRedirect {enabled | disabled}`	**Optional**. Specifies how the system can use this address for redirected VSRP traffic. If `enabled`, the system uses this address when: • The interface associated with the address is up. • The address is the lowest of all the addresses that have `vsrpRedirect` set to `enabled`. If `disabled`, the system uses this address when: • There are no other IP addresses with `vsrpRedirect` set to `enabled`. • This address is the lowest address of the addresses associated with interfaces that are up. The default value is `disabled`.
`managedVRouter`	Specifies the vRouter that is SNMP-manageable through this address. The field is applicable only to the ethMgmt interface.

Delete syntax

```
no vSwitch-name vRouter-name ip address
   ifName ifName
   ipAddr ipAddress
   [netMask ipAddress]
   [vsrpRedirect {enabled | disabled}]
```

Example

The following example defines an interface and then assigns an IP address to it.

```
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# interface eth.1.26
sun(config-vSwitch-e-commerce vRouter-default ip)# address eth.1.26
10.10.10.9
```

Associated MIB

```
ip.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → Ip → address → add
- vSwitch → *name* → vRouter → *name* → Ip → address → copy
- vSwitch → *name* → vRouter → *name* → Ip → address → modify
- vSwitch → *name* → vRouter → *name* → Ip → address → delete

interface

Purpose

Creates an IP interface on the vRouter. It is over this interface that inbound and outbound traffic can be filtered using access control lists (ACLs). The interface links a lower-layer port or port representation to the IP instance on the switch. This command creates the link and configures the characteristics of the interface. You can configure an IP interface directly on an Ethernet port, on a LAG, on a VLAN, on a loopback instance, or to a different vRouter (a different IP instance). You can configure up to 128 IP interfaces per vRouter.

Note that if you set a value for the `packetTrace` or `eventFilter` arguments on an IP instance, and then create an IP interface, the value is inherited. For example, if you set packet trace to enabled, and then add interface eth.1.5, packet trace is enabled for interface eth.1.5. This allows you to begin debugging issues before an interface is fully configured.

The no form of the command deletes the specified interface.

Access mode

config

Syntax

To create or modify an IP interface:

```
vSwitch-name vRouter-name ip interface
   ifName ifName
   [adminState {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {enabled | disabled}]
   [mtu integer]
   [packetTrace {enabled | disabled}]
   [description text]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the name of the interface that connects to the IP instance. This could be either `vlan.`*`x`*, `lag.`*`x`*, `eth.`*`x.x`*., `loopback, or ip.`*`vswitch`*`:`*`vrouter`*. If the interface does not already exist, the system returns an error. Use the `show interface` command to verify configured IP-to-lower layer interface associations. Use this command to configure new associations.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative mode of the interface. Until the interface is enabled, traffic can not be transmitted or received over it.

Argument name	Description
`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`	**Optional.** Sets the event filter level for the interface. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`. This argument is not available when creating an interface, only when modifying or deleting it.
`linkUpDownTrap {enabled | disabled}`	**Optional.** Specifies whether the system should generate a trap when this port changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 5, “SNMP and Trap commands”.

Argument name	Description
`mtu` *`integer`*	**Optional.** Sets the maximum transmission unit (MTU) of the interface. This is the maximum length, in number of bytes, of a packet transmitted over this interface. Packets are fragmented to this size. Valid range is 576 to 9000 bytes; the default MTU is 1500 bytes.
`packetTrace {enabled | disabled}`	**Optional.** Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.
`description` *`text`*	**Optional.** Assigns a text string to an interface. Use this command to identify a interface name (for example, vlan.10) with a meaningful description. This string is displayed with the `show interface verbose` command.

Delete syntax

See the `show interface verbose` command for argument descriptions.

```
no vSwitch-name vRouter-name ip interface
   ifName ifName
   [adminState {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {enabled | disabled}]
   [status {up | down| testing | unknown | dormant]
   [type type]
   [mtu integer]
   [speed integer]
   [physAddress hexString>]
   [lastChange integer]
   [packetTrace {enabled | disabled}]
   [description text]
   [ifIndex hexInteger]
```

Example

The following example defines an IP interface over an Ethernet interface and another over a VLAN.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# interface eth.1.26
description GIFs
sun(config-vSwitch-e-commerce vRouter-default ip)# exit
sun(config-vSwitch-e-commerce vRouter-default)# vlan 30
sun(config-vSwitch-e-commerce vRouter-default vlan-30)# exit
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# interface vlan.30
description engineering
```

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → interface → add
- vSwitch → *name* → vRouter → *name* → Ip → interface → copy
- vSwitch → *name* → vRouter → *name* → Ip → interface → modify
- vSwitch → *name* → vRouter → *name* → Ip → interface → delete
- vSwitch → *name* → vRouter → *name* → Ip → interface → verbose → add
- vSwitch → *name* → vRouter → *name* → Ip → interface → verbose → copy
- vSwitch → *name* → vRouter → *name* → Ip → interface → verbose → modify
- vSwitch → *name* → vRouter → *name* → Ip → interface → verbose → delete

ip

Purpose

Configures the characteristics of the instance of IP running on this virtual router. Each vRouter supports a single instance of IP. IP is enabled by default, but can be disabled with this command.

When you enter this command from the vRouter command mode, your CLI session enters IP command mode. You can configure characteristics from either level.

Access mode

config

Syntax

```
vSwitch-name vRouter-name ip
   [ttl integer]
   [forwarding {enabled | disabled}]
   [adminState {enabled | disabled}]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [linkUpDownTrap {enabled | disabled}]
   [packetTrace {enabled | disabled}]
   [description text]
```

Arguments

Argument name	Description
`ttl integer`	**Optional**. Specifies the time-to-live value inserted into packet headers for all traffic originated within this IP instance. Valid range is 1 through 255; the default value is 64.

Argument name	Description
`forwarding {enabled | disabled}`	**Optional**. Sets the ability to forward traffic over IP interfaces on this vRouter. When `enabled`, the interfaces act as gateways, forwarding received traffic that is not destined for them. If `disabled`, any traffic destined for external destinations is dropped. Specifically: • The system does not export routes via its routing protocols, but does learn them via routing protocols. • IRDP is disabled. • The system drops any IP frames that aren't locally destined. In the system management vRouter the default is disabled. In all other vRouters, the default value is `enabled`.
`adminState {enabled | disabled}`	**Optional**. Sets the administrative mode of the IP instance. IP is enabled by default; use this command to disable IP on the switch. If the protocol is disabled, IP traffic can not be transmitted or received.
`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`	**Optional**. Sets the event filter level for this IP instance. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`.

Argument name	Description
`linkUpDownTrap {enabled | disabled}`	**Optional**. Specifies whether the system should generate a trap when this IP instance changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 5, "SNMP and Trap commands".
`packetTrace {enabled | disabled}`	**Optional**. Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.
`description` *`text`*	**Optional**. Assigns a text string to the IP instance. This string is displayed with the `show ip verbose` command.

Example

The following example enables packet trace for troubleshooting and then disables IP as a result of the debug messages. When the issues are resolved, packet trace is disabled and the protocol is again enabled.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip packetTrace enabled
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# state disabled
sun(config-vSwitch-e-commerce vRouter-default ip)# show verbose
Name:          ip.e-commerce:default
TTL:           64
Forwarding:    enabled
Admin State:   disabled
Oper Status:   down
IfIndex:       0x40040000
Link Traps:    disabled
Event Filter: informational
Packet Trace: enabled
Last Change:   268412
Description:   ipRouter
```

```
sun(config-vSwitch-e-commerce vRouter-default ip)# state enabled
sun(config-vSwitch-e-commerce vRouter-default ip)# packetTrace
disabled
```

Associated MIB

```
ip.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → Ip → ip list → modify

show address

Purpose

Displays a summary of the configured IP address-to-interface association.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip address`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-caplan vRouter-default ip)> show address
IfName       IP Address      Subnet Mask     VSRP Redirect
vlan.100     10.10.10.10     255.0.0.0      disabled
vlan.140     10.20.10.1      255.255.0.0    disabled
eth.1.33     100.200.100.1   255.255.255.0  enabled
```

Output description

Field name	Description	Filter name
`IF Name`	The name of the interface associated with the IP address. This is the interface that connects to the IP instance, either `vlan.`*`x`*, `lag.`*`x`*, `eth.`*`x.x`*., `loopback, or ip.`*`vswitch`*`:`*`vrouter`*. Use the `show interface` command to view all configured IP-to-lower layer interface associations. Use the `interface` command to configure new associations.	`ifName` *`ifName`*
`IP Address`	The IP address assigned to the named interface.	`ipAddr` *`ipAddress`*

Field name	Description	Filter name
`Subnet Mask`	The subnet mask, which identifies the network portion of the address, assigned to the named interface.	`netMask` *`ipAddress`*
`VSRP Redirect`	The setting for whether an interface will most likely be used for redirected traffic. The setting is disabled by default. If all interfaces have redirect disabled, or if more than one interface has redirect enabled, the N2000 Series selects the interface with the lowest numerical IP address.	`vsrpRedirect {enabled | disabled}`
`managedVRouter`	Specifies the vRouter that is SNMP-manageable through this address. The field is applicable only to the ethMgmt interface.	`managedVRouter` *`name`*

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → address

show interface

Purpose

Displays interface configuration data for either all or specified IP interfaces in the routing table. For a display that contains more interface-specific fields, use the `show interface verbose` command. Use the `interface` command to set the user-configurable parameters in this display.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip interface`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-caplan vRouter-default ip)> show interface
IfName     Admin State    Oper Status    MTU      Phys Addr
vlan.30    enabled        down           1500     N/A
eth.1.20   enabled        down           1500     00:07:82:0e:0c:1f
eth.1.26   enabled        down           1500     00:07:82:0e:0c:25
```

Output description

Field name	Description	FIlter name
`IfName`	The name of the interface for which you are displaying configuration information. This is the interface that connects to the IP instance, either `vlan.`*`x`*, `lag.`*`x`*, `eth.`*`x.x.`*, `loopback, or ip.`*`vswitch`*`:`*`vrouter`*. Use the `show interface` command to view all configured IP-to-lower layer interface associations. Use the `interface` command to configure new associations.	`ifName` *`ifName`*
`Admin State`	The administrative (manually configured) status of the interface, either `enabled` or `disabled`. Use the `interface` command to change the administrative status of an interface.	`adminStatus {enabled | disabled}`
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`operStatus {up | down | testing | unknown | dormant}`

Field name	Description	FIlter name
`MTU`	The maximum transmission unit (MTU) of the interface. This is the maximum length, in number of bytes, of a packet transmitted over this interface. Packets are fragmented to this size.	`mtu` *`integer`*
`Phys Addr`	The Layer 2 physical address (the MAC address) of the IP interface. The system associates this address with the interface displayed. Any interface that does not have a physical address associated with it, for example a virtual LAN (VLAN), displays `N/A` in this field.	`macAddress` *`hexString`*

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → interface

show interface verbose

Purpose

Displays interface configuration data for either all or specified IP interfaces in the system's interface table. For a brief display of interfaces, which allows a horizontal view with multiple interface listings on a single screen, use the `show interface` command. Use the `interface` command to set the user-configurable parameters in this display.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip interface verbose`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-caplan vRouter-default ip)> show interface verbose

IfName:       eth.1.10
Admin State:  enabled
Oper Status:  up
MTU:          1500
Phys Addr:    00:07:82:00:03:c1
IfIndex:      0x40041001
Link Traps:   disabled
Event Filter: informational
Packet Trace: disabled
Speed:        100000000
Last Change:  59098
Description:  engineering
```

Output description

Field name	Description	Filter name
`Interface`	The name of the interface for which you are displaying configuration information (the downlink connection to the IP instance). This is the interface that connects to the IP instance, either `vlan.`*x*, `lag.`*x*, or `eth.`*x*`.`*x*. Use the `interface` command to configure new associations.	`loLayer` *ifIndex*
`Admin State`	The administrative (manually configured) status of the interface, either `enabled` or `disabled`. Use the `interface` command to change the administrative status of an interface.	`adminStatus {enabled | disabled}`
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`operStatus {up | down | testing | unknown | dormant}`

Field name	Description	Filter name
MTU	The maximum transmission unit (MTU) of the interface. This is the maximum length, in number of bytes, of a packet transmitted over this interface. Packets are fragmented to this size. Valid range is 0 to 9000 bytes; the default MTU is 1500 bytes.	mtu *integer*
Phys Addr	The Layer 2 physical address (the MAC address) of the IP interface. The system associates this address with the interface displayed. Any interface that does not have a physical address associated with it, for example a virtual LAN (VLAN), displays N/A in this field.	macAddress *hexString*
IfIndex	The hexadecimal equivalent of the IP interface name. Use this value within SNMP to specify and interface name.	ifIndexNum *hexInteger*
Link Traps	The setting for whether the system should generate a trap when this interface changes state (from up to down, or vice versa). The default setting is disabled.	linkUpDownTrap {enabled | disabled}

Field name	Description	Filter name
`Event Filter`	The event filter level set for the interface, one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default level is `informational`.	`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`
`Packet Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by this tool, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.	`packetTrace {enabled | disabled}`

Field name	Description	Filter name
`Speed`	The speed of the interface. This value is derived from the speed of the Ethernet port (at the bottom of the stack) that is connected to this interface.	`speed` *`integer`*
`Last Change`	The value, in ticks, of the `sysUpTime` variable at the time of the last change to the interface. This could be a change to the configuration, or in the administrative state or operational status. If the field displays 0, there have been no changes to the IP instance since the last system reboot.	`lastChange` *`integer`*
`Description`	The text string you assigned to the interface. You can use this value to identify a numeric interface name (for example, vlan.10) with a meaningful description.	`description` *`text`*

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → interface → verbose

show

Purpose

Displays the current configuration for the instance of IP on the vRouter. The user-configurable parameters are set with the `ip` command. This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-e-commerce vRouter-default ip)> show
Name:        ip.sales:default
TTL:         64
Forwarding:  enabled
Admin State: enabled
Oper Status: up
```

Output description

Field name	Description
`IfName`	The system-generated identifier for the instance of IP on the vRouter. The format is `ip.`*`vSwitch`*`:`*`vRouter`*.
`TTL`	The time-to-live value inserted into packet headers for all traffic originated within this IP instance.

Field name	Description
`Forwarding`	The system's setting for the ability to forward traffic over the IP interfaces on this vRouter. When `enabled`, the interfaces act as gateways, forwarding received traffic that is not destined for them. If `disabled`, any traffic destined for external destinations is dropped. Specifically: • The system does not export routes via its routing protocols, but does learn them via routing protocols. • IRDP is disabled. • The system drops any IP frames that aren't locally destined.
`Admin State`	The administrative (manually configured) status of IP, either `enabled` or `disabled`. Use the `ip` command to change the administrative status of the protocol.
`Oper Status`	The operational state of the IP instance. That is, the state the IP instance is currently in, either: `up`: The IP instance is operational `down`: The IP instance is not operational `testing`: The IP instance is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The IP instance is not in any of the listed states `dormant`: The IP instance is uninitialized, the configuration database and hardware have not yet become synchronized

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → ip list

show redirectTraffic

Purpose

Displays the IP address that the system uses to direct virtual service traffic to a VSRP peer node. The system selects this address as follows:

1. The system compares the IP addresses that have `redirectTraffic` set to `enabled` (using the address command) and selects the lowest address associated with an interface that is enabled and up.
2. If no IP addresses have `redirectTraffic` set to `enabled` or if the interfaces associated with these addresses are down, the system compares the IP addresses that have redirectTraffic set to disabled but are associated with interfaces that are enabled and up and selects the lowest address.
3. If no IP addresses are configured or if no interfaces are enabled and up, the system uses an IP address of 0.0.0.0. Typically, this indicates that there is a system problem or the system is not configured properly.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip redirectTraffic`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-e-commerce vRouter-default ip)> show redirectTraffic
IP Address
10.10.10.1
```

Output description

Field name	Description
`IP Address`	The IP address that the system uses for redirected VSRP traffic.

Associated MIB

`ip.mib`

Web path

vSwitch → *name* → vRouter → *name* → Ip → redirectTraffic

show verbose

Purpose

Displays a detailed configuration for the instance of IP on the vRouter. The user-configurable parameters are set with the ip command. This command does not support field filtering.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ip verbose
```

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-e-commerce vRouter-default ip)> show verbose
Name:         ip.e-commerce:default
TTL:          64
Forwarding:   enabled
Admin State:  disabled
Oper Status:  down
IfIndex:      0x40040000
Link Traps:   disabled
Event Filter: informational
Packet Trace: enabled
Last Change:  268412
Description:  ipRouter
```

Output description

Field name	Description
IfName	The system-generated identifier for the instance of IP on the vRouter. The format is ip.*vSwitch*:*vRouter*.
TTL	The time-to-live value inserted into packet headers for all traffic originated within this IP instance.

Field name	Description
`Forwarding`	The system's setting for the ability to forward traffic over the IP interfaces on this vRouter. When `enabled`, the interfaces act as gateways, forwarding received traffic that is not destined for them. If `disabled`, any traffic destined for external destinations is dropped. Specifically: • The system does not export routes via its routing protocols, but does learn them via routing protocols. • IRDP is disabled. • The system drops any IP frames that aren't locally destined.IP instance.
`Admin State`	The administrative (manually configured) status of IP, either `enabled` or `disabled`. Use the `ip` command to change the administrative status of the protocol. IP is enabled by default. If the protocol is disabled, IP traffic can not be transmitted or received.
`Oper Status`	The operational state of the IP instance. That is, the state the IP instance is currently in, either: `up`: The IP instance is operational `down`: The IP instance is not operational `testing`: The IP instance is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The IP instance is not in any of the listed states `dormant`: The IP instance is uninitialized, the configuration database and hardware have not yet become synchronized
`IfIndex`	The hexadecimal equivalent of the ifIndex value of the IP instance.
`Link Traps`	The setting for whether the system should generate a trap when this IP instance changes state (from up to down, or vice versa).

Field name	Description
`Event Filter`	The event filter level set for the IP instance, one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only.
`Packet Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system.
`Last Change`	The value, in ticks, of the `sysUpTime` variable at the time of the last change to the IP instance. This could be a change to the configuration, or in the administrative state or operational status. If the field displays 0, there have been no changes to the IP instance since the last system reboot.
`Description`	The text string you assigned to the IP instance.

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → verbose

show statistics

Purpose

Displays standard statistics for this instance of IP (statistics from the IP MIB).

This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip statistics`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> show ip statistics
In Receives:                                  0
In Header Errors:                             0
In Address Errors:                            0
Pkts Routed:                                  0
Unknown Protos:                               0
In Pkts Discarded:                            0
In Pkts Delivered:                            2944
Out Requests:                                 2960
Out Discards:                                 0
Out No Routes:                                0
Reassembly Timeouts:                          0
Reassemblies Required:                        0
Reassemblies OK:                              0
Reassemblies Failed:                          0
Fragments OK:                                 0
Fragments Failed:                             0
Fragments Created:                            0
Total ARP packet queue size:                  0
Total ARP packet queue overflow:              0
Bad IP checksums:                             0
Martian Addresses:                            0
Dropped Fragments:                            0
Completed Pkts Bad:                           0
Completed Pkts Dropped:                       0
Incomplete Pkts Timeout:                      0
Fragment Length Less Than IP Header Length:   0
```

```
Fragment Length Greater Than IP Header Length: 0
Tiny Fragments:                                0
Ping Of Death:                                 0
Fragment No Memory:                            0
```

Output description

Field name	Description
In Receives	The total number of packets received by this instance of IP. This count includes both forwarded and discarded packets.
In Header Errors	The number of packets received by this IP instance and then discarded because of IP headers errors. These error types include checksum and format errors, version number and option mismatch, and TTL count expiration.
In Address Errors	The number of packets received by this IP instance and then discarded because of an invalid address or unsupported class in the IP header.
Pkts Routed	The number of packets received by this IP instance and then forwarded on toward the destination. This count does not include packets in which this router was the final destination.
Unknown Protos	The number of packets received by this IP instance and then discarded because the protocol was unknown or unsupported.
In Pkts Discarded	The number of packets received by this IP instance and then discarded even though the packet was received without error. Reasons for discard include queue overflow and buffer problems.
In Pkts Delivered	The total number of packets received by this IP instance and delivered to IP user-protocols (including ICMP).
Out Requests	The total number of packets received on this IP instance from user protocols that were requesting transmission.
Out Discards	The number of outbound packets received by this IP instance for transmission and then discarded even though the packet was received without error. Reasons for discard include queue overflow and buffer problems.
Out No Routes	The number of outbound IP packets that were not forwarded on to their destination because no route could be found.
Reassembly Timeouts	The maximum number of seconds an IP fragment is held while awaiting reassembly on this IP instance.
Reassemblies Required	The total number of fragments received by this IP instance that required reassembly.

Field name	Description
`Reassemblies OK`	The number of successful reassemblies into IP packets at this IP instance.
`Reassemblies Failed`	The number of reassembly failures detected by the IP re-assembly algorithm on this IP instance.
`Fragments OK`	The number of IP packets that this IP instance has successfully fragmented. See the `Fragments Created` argument for a count of total fragments generated.
`Fragments Failed`	The number of IP packets that this IP instance was not able to fragment even though the packet required it. This could be the result of a `dontFrag` bit setting.
`Fragments Created`	The total number of fragments created by this IP instance in the fragmentation process. See the `Fragments OK` argument for a count of total packets fragmented.
`Total ARP packet queue size`	The total number of packets queued and waiting for ARP resolution.
`Total ARP packet queue overflow`	The total number of ARP packets that the system dropped because the ARP queue was full.
`Bad IP Checksums`	The total number of packets dropped because they were received with a bad IP checksum.
`Martian Addresses`	The total number of packets dropped because of errors in their addresses, including: the source address equals the destination; the source address is zero, a multicase address (class E), a loopback address (127.x.x.x) or a class F address (240.0.0.0 or higher); or the destination address is zero, a loopback address (127.x.x.x) or a class F address (240.0.0.0 or higher).
`Dropped Fragments`	The total number of fragmented packets dropped because there were no reassembly resources available.
`Completed Packets Bad`	The total number of fragmented packets dropped after reassembly because they exceeded the maximum packet size supported by the system.
`Completed Packets Dropped`	The total number of fragmented packets dropped after reassembly due to resource constraints (e.g., insufficient buffers).
`Incomplete Packets Timeout`	The total number of fragmented packets dropped because the reassembly timeout expired before all fragments were received.
`Fragment Length Less than IP Header Length`	The total number of fragmented packets dropped because the total length of all the fragments was less than the length indicated in the IP header length.

Field name	Description
`Fragment Length Greater than IP Header Length`	The total number of fragmented packets dropped because the total length of all the fragments was greater than the length indicated in the IP header length.
`Tiny Fragments`	The total number of fragmented packets dropped because they appeared to be fragmented into tiny pieces intended to hide TCP and UDP header information.
`Ping Of Death`	The total number of fragmented packets dropped because they attempted to create packets that exceed 64KB in length.
`Fragment No Memory`	The total number of fragmented packets dropped because there was not sufficient memory for reassembly.

Associated MIB

`ip.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → statistics

Chapter 23. Virtual router interfaces commands

vRouter interfaces description

The vRouter interfaces command mode allows you to show and set multiple interfaces on the active vRouter with a single command by using the wildcard symbol. For example, if you set the maximum transmission unit (MTU) for interfaces `ip.*` within this context, the MTU value is changed on all IP interfaces associated with the vRouter. If you then configure one of the individual interfaces within a separate command mode, the new values are applied to that interface. For example, if you then enter the IP command mode and set the MTU for a specific interface, the MTU is reconfigured, but all the other settings from the `vrouter interfaces` command remain.

You can also use this command to configure multiple interface types without having to move between command modes. For example, you could first change the IP MTU by specifying `vSwitch` *`name`* `vRouter` *`name`* `interfaces ip.* MTU 5000`, and then change the trap level for link aggregation groups (LAGs) by specifying `vSwitch` *`name`* `vRouter` *`name`* `interfaces lag.* eventFilter warning`. This method is faster than entering and exiting the IP and LAG command modes to make changes.

Interface types

The N2000 Series system supports several interface types, as described in the following table.

Table 23-1. Interface types supported on the N2000 system

Interface types	Description
`sock`	Socket. Represents the socket layer that resides above IP. Traffic originates from a socket passes through this layer. The socket layer is created and stacked as a side effect of issuing other commands.
`ip`	IP instance.The interface output in the command displays either just the instance number (assigned by the system and based on the virtual router) or the instance number and an interface index.
`eth`	Ethernet interface. The interface name indicates which Ethernet port is assigned to an interface.
`vlan`	Virtual LAN. Composed of Ethernet ports and/or LAGs, VLANs are groups of segments that appear to be on the same Layer 2 network.
`lag`	Link aggregation group. Multiple Ethernet interfaces configured to aggregate bandwidth and appear as a single logical interface to higher layer interfaces
`loopback`	Loopback layer. IP interfaces to this layer can be created when it is necessary to configure reachable IP addresses on interfaces that are not tied to physical interfaces, and therefore never go down.

Tracing the stack through the system

Protocols are implemented in the system as layer objects, and are stacked or interconnected by layer interfaces. These interfaces pass control and status messages between layers, while the network processors forward traffic.

By viewing the output display of the vRouter `show interfaces` or `show interfaces verbose` commands, you can trace a connection from the Ethernet port layer to the layers above the port layer. The information provided to you includes the vSwitch and vRouter, the Ethernet port assignments (when applicable), any intervening LAGs or VLANs, the IP instance, and the IP interface.

Note: The ports assigned to a LAG and individual Ethernet ports do not appear in the `show interfaces` display that you execute from an operated-defined vSwitch. Because Ethernet ports and LAGs are a global system resource, the CLI only displays these components from the `system:management` context.

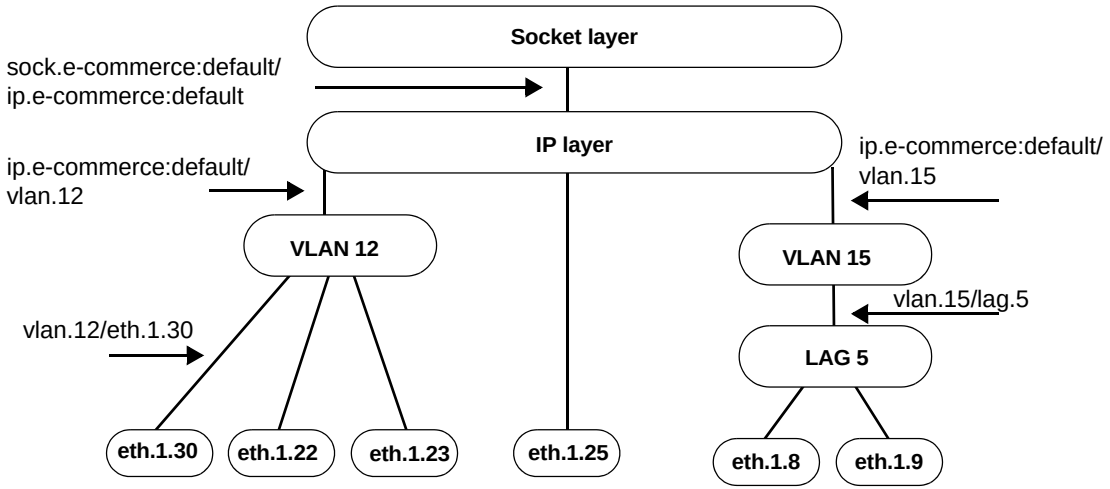
The following sample output is from the `show interfaces` command within the vRouter `default` context on vSwitch `e-commerce`.

```
sun(config-vSwitch-e-commerce vRouter-default)# show interfaces
Interface                      Admin State    Oper Status   Type
sock.e-commerce:default        enabled        up            N/A
sock.e-commerce:default/       enabled        up            N/A
ip.e-commerce:default
ip.e-commerce:default          enabled        up            ip
ip.e-commerce:default/vlan.15  enabled        up            ip
ip.e-commerce:default/         enabled        up            ip
eth.1.25
ip.e-commerce:default/vlan.12  enabled        up            ip
vlan.12                        enabled        up            N/A
vlan.12/eth.1.30               enabled        up            N/A
vlan.12/eth.1.22               enabled        up            N/A
vlan.12/eth.1.23               enabled        up            N/A
vlan.15/lag.5                  enabled        up            N/A
```

The following table describes the content types of the sample output.

Interface entry	Description
`sock.e-commerce:default`	The instance of the socket layer.
`sock.e-commerce:default/ ip.e-commerce:default`	The connection between the socket layer and the instance of IP on vSwitch `e-commerce`, vRouter `default`.
`ip.e-commerce:default`	The instance of IP on vSwitch `e-commerce`, vRouter `default`.
`ip.e-commerce:default/vlan.15`	The connection between VLAN 15 and the instance of IP on vSwitch `e-commerce`, vRouter `default`.
`ip.e-commerce:default/eth.1.25`	The connection between eth.1.25 and the instance of IP on vSwitch `e-commerce`, vRouter `default`.
`ip.e-commerce:default/vlan.12`	The connection between VLAN 12 and the instance of IP on vSwitch `e-commerce`, vRouter `default`.
`vlan.12`	The instance of VLAN 12.
`vlan.12/eth.1.30`	The connection between VLAN 12 and port eth.1.30.
`vlan.15/lag.5`	The connection between VLAN 15 and LAG 5. The Ethernet ports that are part of this LAG can only be viewed from the `system:management` context.

The following figure illustrates the layers of the connection hierarchy from the sample output.



Stack rules

The following sections list the connection rules in the system.

IP interfaces

A single IP interface can connect to the following lower layers:

- A single VLAN, or
- A single LAG, or
- A single Ethernet port, or
- A single loopback layer, or
- A single different IP instance

VLANs

A single VLAN can connect to the following upper layer:

- A single IP instance

A single VLAN can connect to the following lower layers:

- Multiple LAGs, and/or
- Multiple Ethernet ports

LAGs

A single LAG can connect to the following upper layers:

- A single IP instance, or
- Multiple VLANs

A single LAG can connect to the following lower layers:

- Multiple Ethernet ports

Ethernet ports

A single Ethernet port can connect to the following upper layers:

- Multiple VLANs, or
- A single LAG, or
- A single IP instance

vRouter interfaces command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `vRouter` *`name`* `interfaces`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

vRouter interfaces command summary

Table 23-2 lists and briefly describes the vRouter interfaces commands.

Table 23-2. vRouter interfaces command summary

Command name	Description
`interfaces`	Configure parameters for all interfaces on the vRouter.
`show interfaces`	Display settings for all interfaces on the vRouter.
`show interfaces verbose`	Display status information for all interfaces on the vRouter.

interfaces

Purpose

Modifies the characteristics of all interfaces specified on a given vRouter. Use the wildcard (*) character to change multiple interfaces at once. See the section, "Interface types" for a description of `ifIndex` names.

Syntax

```
vSwitch name vRouter name interfaces
   connectionName connIfName
   [adminState {enabled | disabled}]
   [linkUpDownTrap (enabled | disabled)]
   [eventFilter {emergency | alert | critical | error | warning |
      notice | informational | debug}]
   [packetTrace {enabled | disabled}]
   [description text]
   [mtu integer]
```

Arguments

Argument name	Description
`connectionName` *`connIfName`*	Specifies the name of the interface(s) that you want to configure. By using a wildcard, you can set all interfaces matching the entry with a single command. For example, you can set all IP interfaces by specifying `ip.*` for `connIfIndex`. To change all interface on the vRouter, enter `*.*`.
`adminState {enabled | disabled}`	**Optional**. Sets the administrative mode for all specified interfaces. Interfaces are enabled by default; use this command to disable them on the switch.
`linkUpDownTrap {enabled | disabled}`	**Optional**. Specifies whether the system should generate a trap when the interface changes state (from up to down, or vice versa). The default setting is `disabled`. Traps are sent to the location configured as your SNMP event log (trap host) with the SNMP commands. See Chapter 5, "SNMP and Trap commands".

Argument name	Description
`eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}`	**Optional**. Sets the event filter level for the interface. The system reports and stores all events of that severity level and lower in the event log. Enter one of the following values: `emergency`: A fatal error occurred; the system is unusable. `alert`: An error occurred; immediate action is required. `critical`: A serious condition exists; administrative action is required. `error`: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. `warning`: An event occurred that may cause a system problem. `notice`: An event occurred during normal operations that may require administrative action. `informational`: An informational event occurred; no administrative action is required. `debug`: A debug-level event occurred; for internal and technical support use only. The default filter level is `informational`.
`packetTrace {enabled | disabled}`	**Optional**. Sets the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.
`description` *`text`*	**Optional**. Assigns a text description to the interface.
`mtu` *`integer`*	**Optional**. Sets the maximum transmission unit (MTU) for the Ethernet port associated with the interface. This is the maximum length, in number of bytes, of a packet transmitted over the port. Packets are fragmented to this size. Valid range is 576 to 9000 bytes; the default MTU is 1500 bytes. You can only set the MTU on non-management IP interfaces.

Example

The following example displays the current interfaces configured on the vRouter, and then disables all VLAN interfaces.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# show interfaces
Interface                 Admin State  Oper Status  Type
sock.e-commerce:default   enabled      down         N/A
sock.e-commerce:default/  enabled      down         N/A
ip.e-commerce:default
ip.e-commerce:default     enabled      down         ip
ip.e-commerce:default/    enabled      down         ip eth.1.25
vlan.12                   enabled      down         N/A
vlan.12/eth.1.30          enabled      down         N/A
vlan.12/eth.1.22          enabled      down         N/A
vlan.12/eth.1.23          enabled      down         N/A
vlan.12/eth.1.24          enabled      down         N/A
vlan.15                   enabled      down         N/A
sun(config-vSwitch-e-commerce vRouter-default)# interfaces vlan.*
state disabled
6 entries were modified.
sun(config-vSwitch-e-commerce vRouter-default)# show interfaces
Interface                 Admin State  Oper Status  Type
sock.e-commerce:default   enabled      down         N/A
sock.e-commerce:default/  enabled      down         N/A
ip.e-commerce:default
ip.e-commerce:default     enabled      down         ip
ip.e-commerce:default/    enabled      down         ip
eth.1.25
vlan.12                   disabled     down         N/A
vlan.12/eth.1.30          disabled     down         N/A
vlan.12/eth.1.22          disabled     down         N/A
vlan.12/eth.1.23          disabled     down         N/A
vlan.12/eth.1.24          disabled     down         N/A
vlan.15                   disabled     down         N/A
```

Associated MIB

`ifTable.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Interfaces → interfaces list → add
- vSwitch → *name* → vRouter → *name* → Interfaces → interfaces list → copy
- vSwitch → *name* → vRouter → *name* → Interfaces → interfaces list → modify
- vSwitch → *name* → vRouter → *name* → Interfaces → interfaces list → delete

show interfaces

Purpose

Displays interface configuration data for either all or specified IP interfaces in the routing table.

Access mode

user

Syntax

`vSwitch-`*`name`* `vRouter-`*`name`* `show interfaces`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# show interfaces
Interface                      Admin State    Oper Status   Type
sock.e-commerce:default        enabled        up            N/A
sock.e-commerce:default/       enabled        up            N/A
ip.e-commerce:default
ip.e-commerce:default          enabled        up             ip
ip.e-commerce:default/lag.10   enabled        up             ip
ip.e-commerce:default/         enabled        up             ip
eth.1.25
ip.e-commerce:default/vlan.12  enabled        up             ip
vlan.12                        enabled        up             N/A
vlan.12/eth.1.30               enabled        up             N/A
vlan.12/eth.1.22               enabled        up             N/A
vlan.12/eth.1.23               enabled        down           N/A
vlan.12/eth.1.24               enabled        down           N/A
```

Output description

Field name	Description	Filter name
`Interface`	The name of the interface for which you are displaying configuration information. It is through these interface names that you can trace the layers through the system. See "Tracing the stack through the system" for more information.	`connectionName` *`ConnIfName`*
`Admin State`	The administrative (manually configured) status of the interface, either `enabled` or `disabled`. Use the interfaces command to change the administrative status of an interface.	`adminState {enabled | disabled}`
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational. `down`: The interface is not operational. `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present. `unknown`: The interface is not in any of the listed states. `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized.	`operStatus {up | down | testing | unknown | dormant}`

Field name	Description	Filter name
Type	The interface type, either `fastEther` (Fast Ethernet), `gigEther` (Gigabit Ethernet), `etherLag` (LAG interface), or `ip` (VLAN or IP instance). For internal interfaces, such as `sock` the field displays `N/A`. These fields, other than IP, are not displayed in operator-defined vSwitches. You must display the data from the system:management vSwitch to view the values.	`type` *`ifType`*

Associated MIB

`ifTable.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Interfaces → interfaces list

show interfaces verbose

Purpose

Displays detailed configuration information for all interfaces on the vRouter.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name interfaces verbose
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# show interfaces
verbose
Interface:   sock.e-commerce:default
Admin State: enabled
Oper Status: down
Type:        N/A
IfIndex:     0x30040000
Speed:       N/A
MTU:         N/A
Link Traps:  disabled
Evt Filter:  informational
Pkt Trace:   disabled
Last Change: 282429
Description:

Interface:   sock.e-commerce:default/ip.e-comme
Admin State: enabled
Oper Status: down
Type:        N/A
IfIndex:     0x30041001
Speed:       N/A
MTU:         1500
Link Traps:  disabled
Evt Filter:  informational
Pkt Trace:   disabled
Last Change: 282433
Description:
```

```
Interface:   ip.e-commerce:default
Admin State: enabled
Oper Status: down
Type:        ip
IfIndex:     0x40040000
Speed:       N/A
MTU:         N/A
Link Traps:  disabled
Evt Filter:  informational
Pkt Trace:   disabled
Last Change: 268412
Description: ipRouter

Interface:   ip.e-commerce:default/lag.10
Admin State: enabled
Oper Status: down
Type:        ip
IfIndex:     0x40041001
Speed:       N/A
MTU:         1500
Link Traps:  disabled
Evt Filter:  informational
Pkt Trace:   disabled
Last Change: 2122244
Description: engineering

Interface:   vlan.10
Admin State: enabled
Oper Status: down
Type:        N/A
IfIndex:     0x500A0000
Speed:       N/A
MTU:         N/A
Link Traps:  disabled
Evt Filter:  informational
Pkt Trace:   disabled
Last Change: 101779450
Description:

Interface:   vlan.10/eth.1.39
Admin State: enabled
Oper Status: down
Type:        N/A
IfIndex:     0x500A1001
Speed:       100000000
MTU:         1500
Link Traps:  disabled
Evt Filter:  informational
Pkt Trace:   disabled
Last Change: 101927300
Description:
```

Output description

Field name	Description	Filter name
`Interface`	The name of the interface for which you are displaying configuration information. It is through these interface names that you can trace the layers through the system. See "Tracing the stack through the system" for more information.	`connectionName` *`connIfName`*
`Admin State`	The administrative (manually configured) status of the interface, either `enabled` or `disabled`. Use the interfaces command to change the administrative status of an interface.	`adminState {enabled | disabled}`
`Oper Status`	The operational state of the interface. That is, the state the interface is currently in, either: `up`: The interface is operational `down`: The interface is not operational `testing`: The interface is attempting to come up, and is verifying that configuration necessary for operation is present `unknown`: The interface is not in any of the listed states `dormant`: The interface is uninitialized, the configuration database and hardware have not yet become synchronized	`operStatus {up | down | testing | unknown | dormant}`
`Type`	The interface type, either `fastEther` (Fast Ethernet), `gigEther` (Gigabit Ethernet), `etherLag` (LAG interface), or `ip` (VLAN or IP instance). For internal interfaces, such as `sock`, the field displays `N/A`. These fields, other than IP, are not displayed in operator-defined vSwitches. You must display the data from the system:management vSwitch to view the values.	`type` *`ifType`*
`IfIndex`	The hexadecimal equivalent of the interface name. Use this value within SNMP to specify an interface name.	`ifIndex` *`hexInteger`*

Field name	Description	Filter name
Speed	The speed of the interface. This value is derived from the speed of the Ethernet port (at the bottom of the stack) that is connected to this interface.	speed *integer*
MTU	The maximum transmission unit (MTU) of the interface. This is the maximum length, in number of bytes, of a packet transmitted over this interface. Packets are fragmented to this size.	mtu *integer*
Link Traps	The setting for whether the system should generate a trap when this interface changes state (from up to down, or vice versa). The default setting is disabled.	linkUpDownTrap {enabled | disabled}
Event Filter	The event filter level set for the interface, one of the following values: emergency: A fatal error occurred; the system is unusable. alert: An error occurred; immediate action is required. critical: A serious condition exists; administrative action is required. error: A event occurred that can cause a loss of some system functionality; administrative action may be necessary. warning: An event occurred that may cause a system problem. notice: An event occurred during normal operations that may require administrative action. informational: An informational event occurred; no administrative action is required. debug: A debug-level event occurred; for internal and technical support use only. The default level is informational.	eventFilter {emergency | alert | critical | error | warning | notice | informational | debug}

Field name	Description	Filter name
`Packet Trace`	The setting for the packet trace capability. When `enabled`, the system sends debug messages relating to packets moving through the system. Because of the large amount of data generated by setting this variable, you should only enable it when you want to debug your network by verifying where packets are going in the system. The default is `disabled`.	`packetTrace {enabled | disabled}`
`Last Change`	The value, in ticks, of the `sysUpTime` variable at the time of the last change to the interface. This could be a change to the configuration, or in the administrative state or operational status. If the field displays 0, there have been no changes to the IP instance since the last system reboot.	`lastChange` *`integer`*
`Description`	The text string you assigned to the interface.	`description` *`text`*

Associated MIB

`ifTable.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Interfaces → status

Part VI. IP protocols and utilities

The chapters in Part VI describe the commands for managing IP functions in the system:

Chapter 24. RIP and static routing commands

IP commands description

This chapter describes the Routing Information Protocol (RIP) and static routing commands available from the N2000 Series system. The system supports Internet PRotocol (IP) Version 4. The N2000 Series supports several other IP-related commands, which are detailed in the following chapters:

IP Feature	Chapter
IP interfaces and addressing	Chapter 22.
• Address Resolution Protocol (ARP) • Internet Control Message Protocol (ICMP) • Internet Domain Routing Protocol (IRDP)	Chapter 26.
Ping and traceroute utilities	Chapter 27.
Access control lists (ACLs)	Chapter 28.

IP routing description

The N2000 Series supports static routing and the Routing Information Protocol for routing functions. Each vSwitch in the N2000 Series represents a physical switch with its own routing domain, functional policies, and resources. The vRouters in a vSwitch are responsible for routing traffic to the network and between other vSwitches. You can configure the following routing functions for each vRouter:

- **Static routes** — Manually defines a route for a specific destination network or host or a default route. The system uses a default route to forward data if it does not have the destination in its routing table. See Chapter 1 in the *Sun N2000 Series Release 2.0 – System Administration Guide* for details about configuring a default route.
- **Routing Information Protocol** — Allows vRouters in vSwitches to exchange route information with other network routers.

RIP description

The Routing Information Protocol (RIP) is a distance vector protocol that routers use to exchange information about available routes in a local area network. Routers using RIP send periodic update messages to other routers in the network and when the network topology changes. These updates allow a router to learn the network topology and calculate the best path to a destination network.

When a router receives routing information from another router, it updates its local routing table and sends a RIP update containing the revised routing table to its neighboring routers. By default, the N2000 Series sends RIP updates to neighboring routers every 30 seconds. You can change the update interval using the `rip globalSettings` command.

The router stores only the best path to a destination in its routing table. RIP uses a hop count to calculate the best path. The best path is the path that has the fewest number of hops (routers) that packets must traverse to reach the destination network. The maximum number of hops allowed in a path in a RIP network is 15 hops.

The N2000 Series supports RIP v1 and RIP v2.

IP routing command path

The command names in this chapter show you how to execute the commands from within the following command modes:

```
vSwitch name vRouter name ip
vSwitch name vRouter name rip
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

RIP and IP route command summary

Table 24-1 lists and briefly describes the IP route and RIP commands.

Table 24-1. RIP and IP route command summary

Command name	Description
`rip advertise`	Configure RIP advertisement settings.
`rip globalSettings`	Configure global RIP attributes.
`rip interface`	Configure one or more RIP interfaces for a vRouter.
`rip sourceGateway`	Specify routers to which the N2000 Series sends RIP updates directly.
`rip trustedGateway`	Specify the systems from which the N2000 Series accepts RIP updates.
`route static`	Configure a static route for a vRouter.
`show rip advertise`	Display the current RIP advertisement settings.
`show rip globalSettings`	Display global RIP configuration.

Table 24-1. RIP and IP route command summary (continued)

Command name	Description
`show rip interface`	Display the configured RIP interfaces.
`show rip interface statistics`	Display statistics for specified RIP interfaces.
`show rip peers`	Display summary information about routers from which the N2000 Series received valid RIP updates.
`show rip sourceGateway`	Display a list of systems to which the N2000 Series sends RIP updates directly.
`show rip statistics`	Display global RIP statistics.
`show rip trustedGateway`	Display a list of systems from which the N2000 Series accepts RIP updates.
`show route`	Display the routes in the IP routing table.
`show route static`	Display the static route configurations.

rip advertise

Purpose

Configures RIP advertisement settings.

Access mode

config

Syntax

```
vswitch name vrouter name rip advertise
   [direct {enabled | disabled}]
   [staticRt {enabled | disabled}]
   [ospf {enabled | disabled}]
   [ospfAse {enabled | disabled}]
   [directMetric integer]
   [staticRtMetric integer]
   [ospfMetric integer]
   [ospfAseMetric integer]
```

Arguments

Argument name	Description
`direct {enabled | disabled}`	**Optional.** Allows or disallows direct routes to be redistributed into RIP as external routes and be advertised in the link state advertisement (LSA). The default is `enabled`.
`staticRt {enabled | disabled}`	**Optional.** Allows or disallows static routes to be redistributed into RIP as external routes and be advertised in the LSA. The default is `disabled`.
`ospf {enabled | disabled}`	**Optional.** Allows or disallows OSPF routes to be redistributed into RIP as external routes and be advertised in the LSA. The default is `disabled.`
`ospfAse {enabled | disabled}`	**Optional.** Allows or disallows OSPF ASE routes to be redistributed into RIP as external routes and be advertised in the LSA. The default is `disabled.`
`directMetric` *`integer`*	**Optional.** The cost applied to default direct routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.

Argument name	Description
`staticRtMetric` *`integer`*	**Optional.** The cost applied to default static routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.
`ospfMetric` *`integer`*	**Optional.** The cost applied to default OSPF routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.
`ospfAseMetric` *`integer`*	**Optional.** The cost applied to default OSPF ASE routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.

Example

This example shows how to configure the management vRouter in the system vSwitch so that routes in the routing table expire after 60 seconds and the system sends RIP updates every 20 seconds.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config)# vrouter default
sun(config-vswitch-e-commerce vrouter-default rip)#
```

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → advertise → modify

rip globalSettings

Purpose

Configures global RIP variables, for example, timers that affect how often the system sends RIP updates and how long routes can exist in the routing table without being refreshed.

Access mode

config

Syntax

```
vswitch name vrouter name rip globalSettings
   adminState {enabled | disabled}
   expireTime seconds
   updateTime seconds
```

Arguments

Argument name	Description
`adminState {enabled | disabled}`	Configures the administrative state of the RIP configuration. If `enabled`, RIP is enabled for the vRouter; if `disabled`, RIP is disabled for the vRouter. The default setting is `enabled`.
`expireTime` *seconds*	Configures the amount of time, in seconds, a route that the system receives from a RIP update can exist in the IP routing table without being refreshed. The valid range is 5 through 180 seconds; the default setting is 180 seconds. If a RIP update does not refresh the route information within the set time, the system removes the route from the IP routing table.
`updateTime` *seconds*	Configures the time, in seconds between RIP updates that the system sends to neighboring routers. The valid range is 1through 30 seconds, the default setting is 30 seconds.

Example

This example shows how to configure the management vRouter in the system vSwitch so that routes in the routing table expire after 60 seconds and the system sends RIP updates every 20 seconds.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default rip
sun(config-vswitch-e-commerce vrouter-default rip)# globalSettings
expireTime 60 updateTime 20
```

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → globalSettings → modify

rip interface

Purpose

Configures an interface for the sending and receiving of RIP packets.

The no form of the command deletes a configured RIP interface. If you enter optional arguments, the CLI deletes the configuration only if it matches all arguments. With the no form of the command, the `address` argument is the only required argument.

Access mode

config

Syntax

```
vswitch name vrouter name rip interface
   address IP address
   [authType {noAuthentication | simplePassword | md5}]
   [authKey text]
   [send {doNotSend | ripVersion 1 | rip1Compatible | ripVersion2}]
   [receive {rip1OrRip2 | doNotReceive}]
   [defaultMetric integer]
   [authId integer]
```

Arguments

Argument name	Description
address	IP address of the RIP interface. Needs to match a local IP address for RIP to run on that interface.

Argument name	Description
`authType`	**Optional**. Sets the type of authentication you want to use for RIP v2 updates. Valid values are: `noAuthentication`: Do not use authentication. `simplePassword`: A text password that the system includes in the RIP updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the RIP v2 interfaces in the same network. This method is the least secure authentication method. `md5`: The system uses the MD5 algorithm to create an encoded checksum RIP update, which it includes in the RIP updates it sends to other routers. The router that receives the RIP update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password. The default setting is `noAuthentication`.
`authKey`	**Optional**. Configures the password or MD5 key required for RIP v2 authentication. If using simplePassword authentication, enter a password of 8 characters or fewer. If using MD5, enter a key that the MD5 protocol uses to create the encoded checksum that the system sends in the RIP update. This value *must* match the keys configured on the other RIP routers in the network. The default setting is `none`.

Argument name	Description
`send`	**Optional**. Configures the type of RIP updates the N2000 Series sends to neighboring routers. The valid values are: `doNotSend`: The system does not sent RIP updates. `ripVersion1`: The system sends RIP v1 packets when sending RIP updates. `rip1Compatible`: The system broadcasts RIP v2 packets that are compliant with RFC 1058 route subsumption rules. `ripVersion2`: The system multicasts RIP v2 packets when sending RIP updates. The default setting is `rip1Compatible`.
`receive`	**Optional**. Configures the type of RIP updates the N2000 Series can receive from neighboring routers. The value values are: `rip1OrRip2`: the system can receive RIP updates in a RIP v1 or RIP v2 format. `doNotReceive`: the system does not accept RIP updates from neighboring routers. The default setting is `rip10orRip2`.
`defaultMetric`	**Optional**. Indicates the metric to use for the default route entry in RIP updates that this interface sends to neighboring routers. Valid values are 0 through 15; the default setting is 1. Use a value of zero if you configured a default route and do not want to advertise it in RIP updates.

Argument name	Description
`authID`	**Optional.** Assigns an authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID **Note:** You must set the `authType` argument to `md5` to use this argument.

Delete filters

See the `show rip interface` command for argument descriptions.

```
no vswitch name vrouter name rip interface
   address IP address
   [authType {noAuthentication | simplePassword | md5}]
   [authKey text]
   [send {doNotSend | ripVersion 1 | rip1Compatible | ripVersion2}]
   [receive {rip1OrRip2 | doNotReceive}]
   [defaultMetric integer]
   [authId integer]
```

Example

In this example, a RIP interface is established for IP address 10.10.10.40, using a simple password for the authentication of RIP Version 2 packets.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vSwitch-e-commerce vRouter-default rip)# interface address
10.10.10.40 authType simplePassword authKey passwd23 send ripVersion2
receive rip1OrRip2
```

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → interface → add
- vSwitch → *name* → vRouter → *name* → Rip → interface → copy
- vSwitch → *name* → vRouter → *name* → Rip → interface → modify
- vSwitch → *name* → vRouter → *name* → Rip → interface → delete

rip sourceGateway

Purpose

Configures the IP address of a system to which you want to send unicast RIP updates directly instead of broadcasting or multicasting RIP updates. If you configure source gateways, the system sends RIP updates to the specified systems only.

The no form of the command deletes a configured source gateway.

Access mode

config

Syntax

```
vswitch name vrouter name rip sourceGateway
   address ipAddress
```

Arguments

Argument name	Description
address *ipAddress*	Specifies the IP address, in dotted-decimal format, for the system to which you want to send RIP updates.

Delete filters

See the show rip sourceGateway command for argument descriptions.

```
no vswitch name vrouter name rip sourceGateway
   address ipAddress
```

Example

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# sourceGateway
10.10.40.51
```

Associated MIB

```
ripTbl.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → Rip → sourceGateway → add
- vSwitch → *name* → vRouter → *name* → Rip → sourceGateway → copy

rip trustedGateway

Purpose

Configures the system to accept RIP updates from the specified systems only. If you do not configure a trusted gateway, the system accepts RIP updates from all routers on the network.

The no form of the command removes the trusted gateway configuration.

Access mode

config

Syntax

```
vswitch name vrouter name rip trustedGateway
   address ipAddress
```

Arguments

Argument name	Description
address *IP address*	Specifies the IP address, in dotted-decimal format, for the system from which you want to receive RIP updates.

Delete filters

```
no vswitch name vrouter name rip trustedGateway
   address ipAddress
```

Example

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# trustedGateway
10.10.50.45
```

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → trustedGateway → add
- vSwitch → *name* → vRouter → *name* → Rip → trustedGateway → `copy`
- vSwitch → *name* → vRouter → *name* → Rip → trustedGateway → `delete`

route static

Purpose

Manually configures a route that the system uses to send data to a destination network. The system can use a static route before using routes it learns from dynamic routing protocols, depending on the static route's preference. A static route on the N2000 Series requires the following values:

- An IP address for the destination host or network for this route
- Subnet mask for the destination IP address
- The address of a locally attached router where packets are sent for forwarding to the destination network (the address of the next hop router)

You can configure up to 2000 static routes. For backup purposes, it is useful to configure multiple static routes that have the same destination and subnet mask but use different next-hop routers. In this situation, if a next-hop router becomes unreachable, the system can use a different static route for the destination, based on the route's preference and metric setting. Preference takes priority over metric, however.

You can use this command to define a default route for the system. To define a default route, configure a destination address of 0.0.0.0 and a subnet mask of 0.0.0.0.

The `no` form of this command deletes one or more defined static route configurations. If you enter optional arguments, the CLI deletes the user entry only if it matches all arguments. With the `no` form of the command, the required arguments are `destAddr`, `mask`, and `nextHop`.

Access mode

config

Syntax

To create a static route:

```
vswitch name vrouter name ip route static
   destAddr ipAddress
   mask ipAddress
   nextHop ipAddress
   [ifName {ifName | unspecified}]
   [preference {low | high}]
   [metric integer]
```

To modify a static route:

```
vswitch name vrouter name ip route static
   destAddr ipAddress
   mask ipAddress
   nextHop ipAddress
   ifName ifName
   [preference {low | high}]
   [metric integer]
```

Arguments

Argument name	Description
`destAddr ipAddress`	Specifies the destination IP address of this route. When creating a default route, use 0.0.0. 0. for this entry and use 0.0.0.0. for the mask argument. The default route is the route the system uses if it does not have a route for a destination in its IP routing table.
`mask ipAddress`	Assigns a subnet mask to the destination address. When creating a default route, use 0.0.0.0 for this argument and use 0.0.0.0 for the destAddr argument.
`nextHop ipAddress`	Specifies the IP address of a locally attached router where the system sends packets for forwarding to the destination. IP address should be in a local subnet. If just the ifName is needed, this value should be 0.0.0.0.

Argument name	Description
`ifName {`*`ifName`* `| unspecified`	**Optional.** Configures the ifName for the Ethernet port used to reach next hop. The interface can be either ip`.`*`vswitch:vrouter`*, `vlan.`*`x`*, `lag.`*`x`*, loopback or `eth.`*`x.x`*. By using a wildcard, you can set all interfaces matching the entry with a single command. For example, you can set all IP interfaces by specifying `ip.*`. The default value is `unspecified`. If you use the default value, the system uses the IP address of the next-hop router to determine which interface to use. This argument is required when modifying or deleting a route.
`preference {low | high}`	**Optional.** Configures the value the system uses to determine which static route to use for a specific destination network. The valid values are `low` and `high`; the default setting is `low`. When the destination addresses and subnet masks are the same for multiple static routes, the system will only select those static routes that have a nextHop on an interface that is up. If there is more than one of these, the one with the highest preference will be selected. If there is more than one of these, the one with the lowest metric will be selected. If there is more than one of these, the one with the highest nextHop address will be selected. Route Preference: 1. Local routes 2. OSPF Internal 3. Static routes with a preference of `high` 4. RIP routes 5. Static routes with a preference of `low` 6. OSPF External
`metric` *`integer`*	**Optional.** Sets the primary routing metric for this route. Valid values are 1 through 255; the default setting is 1. This value controls how the metric is advertised in routing protocol updates.

Delete filters

See the `show route static` command for argument descriptions.

```
no vswitch name vrouter name ip route static
   destAddr ipAddress
   mask ipAddress
   nextHop ipAddress
   ifName ifName]
   [preference StaticRoutePreference]
   [metric integer]
```

Associated MIB

`ipFwdTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip route → static → add
- vSwitch → *name* → vRouter → *name* → Ip route → static → copy
- vSwitch → *name* → vRouter → *name* → Ip route → static → modify
- vSwitch → *name* → vRouter → *name* → Ip route → static → delete

show rip advertise

Purpose

Displays a summary of the RIP advertisement configuration.

Access mode

user

Syntax

```
show vswitch name vrouter name rip advertise
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show advertise
Direct:          enabled
Static:          disabled
Ospf:            disabled
Ospfase:         disabled
Direct Metric:   1
Static Metric:   1
Ospf Metric:     1
Ospf ASE Metric: 1
```

Output description

Field name	Description
`Direct`	Allows or disallows direct routes to be redistributed into RIP as external routes and be advertised in the link state advertisement (LSA).
`Static`	Allows or disallows static routes to be redistributed into RIP as external routes and be advertised in the LSA. The default is `disabled`.
`Ospf`	Allows or disallows OSPF routes to be redistributed into RIP as external routes and be advertised in the LSA. The default is `disabled.`

Field name	Description
`Ospfase`	Allows or disallows OSPF ASE routes to be redistributed into RIP as external routes and be advertised in the LSA. The default is `disabled`.
`Direct Metric`	The cost applied to default direct routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.
`Static Metric`	The cost applied to default static routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1
`Ospf Metric`	The cost applied to default OSPF routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.
`Ospf ASE Metric`	The cost applied to default OSPF ASE routes entering the RIP routing domain. Valid range is 1 through 15; the default value is 1.

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → advertise

show rip globalSettings

Purpose

Displays the configured administrative state. This command does not support field filtering.

Access mode

user

Syntax

```
show vswitch name vrouter name rip globalSettings
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sunsun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show
globalSettings
    Admin State: enabled
    Expire Time: 5
    Update Time: 30
```

Output description

Field name	Description
Admin State	The administrative state of the RIP configuration. The default is enabled.

Field name	Description
`Expire Time`	The amount of time, in seconds, a route that the system receives from a RIP update can exist in the IP routing table without being refreshed. The valid range is 5 through 180 seconds; the default setting is 180 seconds. If a RIP update does not refresh the route information within the set time, the system removes the route from the IP routing table.
`Update Time`	The time, in seconds between RIP updates that the system sends to neighboring routers. The valid range is 1through 30 seconds, the default setting is 30 seconds.

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → rip → globalSettings

show rip interface

Purpose

Displays a summary of the RIP interface configurations.

Access mode

user

Syntax

```
show vswitch name vrouter name rip interface
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show interface
Address:       10.10.10.10
AuthType:      noAuthentication
AuthKey:
AuthID:
Send:          doNotSend
Receive:       doNotReceive
DefaultMetric: 1
```

Output description

Field name	Description	Filter name
Address	The IP address associated with the RIP interface.	address *IP address*
AuthType	The type of RIP v2 authentication used.	authType {noAuthentication | simplePassword | md5}
AuthKey	The password or MD5 key required for RIP v2 simple password or MD5 authentication.	authKey *text*

Field name	Description	Filter name
`AuthID`	The authentication ID for MD5 authentication. This value is used when other routers in the network use an authentication ID.	`authID` *`integer`*
`Send`	The type of RIP updates that the system sends to neighboring routers.	`send {doNotSend | ripVersion1 | rip1Compatible | ripVersion2}`
`Receive`	The type of RIP packets the system receives from neighboring routers.	`receive {rip1OrRip2 | doNotReceive}`
`DefaultMetric`	The metric used for the default route entry in RIP updates that this interface sends to neighboring routers.	`defaultMetric` *`integer`*

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → interface

show rip interface statistics

Purpose

Displays operational statistics for configured RIP interfaces.

Access mode

user

Syntax

```
show vswitch name vrouter name rip interface statistics
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show interface statistics
Address         Bad Packets    Bad Routes   Sent Updates  Sent Pkts   Rec Pkts
127.0.0.0       25             5            28            0           0
192.168.124.46  0              0            0             0           0
```

Output description

Field name	Description	Filter name
Address	The IP address associated with the RIP interface.	address *IP address*
Bad Pkts	The number of bad packets the system received on this interface.	rcvBadPackets *integer*
Bad Routes	The number of bad routes the system received on this interface.	rcvBadRoutes *integer*
Sent Updates	The number of RIP updates the system sent on this interface.	sendUpdates *integer*

Field name	Description	Filter name
`Sent Pkts`	The number of RIP packets the system sent on this interface.	`sentPkts` *`integer`*
`Rec Pkts`	The number of RIP packets the system received on this interface.	`recPkts` *`integer`*

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → interface → statistics

show rip peers

Purpose

Displays summary information about routers from which the system received valid RIP updates within the last 180 seconds. This information can be useful when you need to solve communication problems that exist between the N2000 Series and other RIP routers.

Access mode

user

Syntax

```
show vswitch name vrouter name rip peers
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show peers
Address:         192.168.124.253
Domain:          0x0000
Last Update:     Sun Oct 11 00:10:01 2002
Version:         1
Rcv Bad Packets: 5
Rcv Bad Routes:  0
```

Output description

Field name	Description	Filter name
Address	The IP Address that the RIP peer is using as its source address.	address *IP address*
Domain	The value in the Routing Domain field in RIP packets that the system received from the peer system. Domain support is deprecated; this is zero.	domain *hexidecimal string*

Field name	Description	Filter name
`Last Update`	The number of timeTicks indicating when the system received the most recent RIP update from the peer system.	`lastUpdate` *`timeTicks`*
`Version`	The RIP version number in the header of the last RIP packet that the system received.	`version` *`integer`*
`Rcv Bad Packets`	The number of RIP response packets that the system received from this peer and discarded as invalid.	`rcvBadPackets` *`integer`*
`Rcv Bad Routes`	The number of routes from this peer that the system ignored because the entry format was invalid.	`rcvBadRoutes` *`integer`*

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → peers

show rip sourceGateway

Purpose

Displays a list of the routers to which the system sends RIP updates. If no sourceGateways exist, the system sends RIP updates to all neighboring routers.

Access mode

user

Syntax

```
show vswitch name vrouter name rip sourceGateway
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show sourceGateway
Address
10.10.40.51
```

Output description

Field name	Description	Filter name
Address	The IP address for router to which the system sends RIP updates.	address *IP Address*

Associated MIB

ripTbl.mib

Web path

- vSwitch → *name* → vRouter → *name* → Rip → sourceGateway

show rip statistics

Purpose

Displays a summary of global RIP statistics. This command does not support field filtering.

Access mode

user

Syntax

```
show vswitch name vrouter name rip statistics
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show statistics
RouteChanges: 5
Queries:      35
```

Output description

Field name	Description
RouteChanges	The number of route changes that RIP made to the IP routing table. This value does not include the number or refreshes of a route's age.
Queries	The number of RIP queries the system received from other routers.

Associated MIB

`ripTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Rip → statistics

show rip trustedGateway

Purpose

Displays a list of the routers from which the system accepts RIP updates.

Access mode

user

Syntax

```
show vswitch name vrouter name rip trustedGateway
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# rip
sun(config-vswitch-e-commerce vrouter-default rip)# show
trustedGateway
Address
10.10.50.65
```

Output description

Field name	Description	Filter name
Address	The IP address for router from which the system receives RIP updates.	address *IP Address*

Associated MIB

ripTbl.mib

Web path

- vSwitch → *name* → vRouter → *name* → Rip → trustedGateway

show route

Purpose

Displays the contents of the IP routing table.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip route`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# ip
sun(config-vswitch-e-commerce vrouter-default ip)# show route
Dest Addr       Mask            Next Hop        IfName     Protocol Age  Metric
0.0.0.0         0.0.0.0         192.168.124.18  ethMgmt.1  netmgt  0    1
192.168.124.0   255.255.255.0   0.0.0.0         ethMgmt.1  local   0    1
```

Output description

Field name	Description	Filter name
`Dest Addr`	The destination IP address of the route.	`destAddr` *`IP Address`*
`Mask`	The subnet mask for the destination IP address.	`mask` *`IP Address`*
`nextHop`	The address of the next system for this route. 0.0.0.0 indicates a local route.	`nextHop` *`IP Address`*
`IfName`	The value that identifies the Ethernet interface that the system uses to reach the next hop in the route. A value of 0 indicates the system uses the IP address of the next-hop router to determine which interface to use.	`ifName` *`ifName`*

Field name	Description	Filter name
`Protocol`	Protocol by which the system learned this route. Possible protocols are: `other:` not specified `local:` local interface `netmgmt:` static route `rip:` Berkeley RIP or RIP-II `ospf:` Open Shortest Path First	`routeProto {other | local | netmgmt | rip | ospf}`
`Age`	Seconds since the system updated the route or determined that the route was correct.	`routeAge` *`integer`*
`Metric`	The primary routing metric for this route. The system advertises this metric in routing protocol updates.	`Metric` *`integer`*

Associated MIB

`ipFwdTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → route

show route static

Purpose

Displays the static route configurations on the system.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ip route static
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# ip
sun(config-vswitch-e-commerce vrouter-default ip)# route
sun(config-vSwitch-system vRouter-management ip route)# show static
DestAddr       Mask          NextHop        IfName       Preference  Metric
0.0.0.0        0.0.0.0       192.168.124.18 eth.1.1      low         1
```

Output description

Field name	Description	Filter name
DestAddr	The destination IP address of the route. The system considers an entry with a value of 0.0.0.0 to be the default route.	destAddr *IP Address*
Mask	The subnet mask for the destination IP address.	mask *IP Address*
NextHop	The address of the next router in the destination route.	nextHop *IP Address*

Field name	Description	Filter name
`IfName`	The Ethernet interface that the system uses to reach the next hop in the route. A value of `unspecified` indicates the system uses the IP address of the next-hop router to determine which interface to use.	`ifName` *`IfName`*
`Preference`	The value the system uses to determine how this static route compares to other types of routes.	`preference {low | high}`
`Metric`	The primary routing metric for this route. The system advertises this metric in routing protocol updates. Possible values are 1 through 255.	`metric` *`integer`*

Associated MIB

`ipFwdTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → route → static

Chapter 25. OSPF commands

OSPF description

Open Shortest Path First (OSPF) is a link-state routing protocol that runs within an autonomous system (AS). Each router in the AS maintains a database of the shortest paths to all other valid routes within the AS, and calculates the best path to each destination. The router regularly recalculates the routes to ensure that its database reflects topological changes.

OSPF uses areas to minimize routing tables by grouping networks into a single entity. There are several commands that configure area characteristics. In these commands, the areas are described by their area ID, which is specified in IP address format. Area 0.0.0.0 is reserved for the OSPF backbone. If you specify an area that does not exist, the system creates it.

In addition, OSPF uses a router ID to uniquely identify the router in the autonomous system. The router ID uses the format of an IP address. The default router ID is the IP address of one of the router's IP interfaces, or you can configure the router ID explicitly.

RFC compatibility

The N2000 Series implementation supports the majority of RFC 1850, *OSPF Version 2 Management Information Base*, but some fields are unsupported. For RFC compatibility, the field is accepted at the CLI or through the Web interface, but the value does not change the software behavior.

OSPF command path

The command names in this chapter show you how to execute the commands from within the following command modes:

`vSwitch` *`name`* `vRouter` *`name`* `ospf`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

OSPF command summary

Table 25-1 lists and briefly describes the OSPF commands.

Table 25-1. OSPF command summary

Command name	Description
`advertise-ase`	Configure handling of Autonomous System External (ASE) routes.
`advertise-nssa`	Configure handling of routes into the not-so-stubby area (NSSA).
`area`	Configure OSPF area characteristics.
`areaAggregate`	Define an area aggregate range.
`globalSettings`	Set general OSPF characteristics.
`host`	Configure directly connected hosts.
`interface`	Configure an OSPF interface.
`show advertise-ase`	Display settings for handling ASE (type 5) routes.
`show advertise-nssa`	Display settings for handling NSSA (type 7) routes.
`show area`	Display OSPF area settings.
`show areaAggregate`	Display area aggregate settings.
`show extLsdb`	Display the external link state database.
`show globalSettings`	Display general OSPF settings.

Table 25-1. OSPF command summary (continued)

Command name	Description
`show host`	Display directly connected host settings.
`show interface`	Display configured interface settings.
`show lsdb`	Display the link state database.
`show neighbors`	Display neighbor information.
`show virtualInterface`	Display characteristics of configured OSPF virtual interfaces.
`show virtualNeighbors`	Display virtual neighbor information.
`virtualInterface`	Configure an OSPF virtual interface.

Basic OSPF configuration

You configure OSPF on each vRouter on which it will run.

Configuring an OSPF backbone router

Table 25-2. Steps for configuring an OSPF backbone router

Step	Action
1.	Configure an OSPF backbone area, 0.0.0.0, for the vRouter with the `area` command.
2.	Configure a nonbackbone area for the vRouter with the `area` command.
3.	Configure an OSPF interface for the backbone area with the `interface` command.
4.	Configure an OSPF interface for the nonbackbone area with the `interface` command.
5.	Verify the configuration with the `show area` and `show interface` commands.

advertise-ase

Purpose

Controls the flow of non-OSPF (Autonomous System External (ASE) Type 5) routes into the OSPF area. Based on type, you can allow or disallow a route originating from a non-OSPF source to be redistributed into OSPF as an OSPF external route (ASE). In addition you can specify what ASE type they are (Type 1 or Type 2) and the metric associated with them.

Access mode

config

Syntax

```
vSwitch name vRouter name ospf advertise-ase
   [direct {enabled | disabled}]
   [staticRt {enabled | disabled}]
   [rip {enabled | disabled}]
   [directMetric integer]
   [staticRtMetric integer]
   [ripMetric integer]
   [directType {type1 | type2}]
   [staticRtType {type1 | type2}]
   [ripType {type1 | type2}]
```

Arguments

Argument name	Description
`direct {enabled | disabled}`	**Optional.** Allows or disallows direct routes to be redistributed into OSPF as external routes and be advertised in the link state advertisement (LSA). The default is `enabled`.
`staticRt {enabled | disabled}`	**Optional.** Allows or disallows static routes to be redistributed into OSPF as external routes and be advertised in the LSA. The default is `disabled`.
`rip {enabled | disabled}`	**Optional.** Allows or disallows RIP routes to be redistributed into OSPF as external routes and be advertised in the LSA. The default is `disabled`.

Argument name	Description
`directMetric` *`integer`*	**Optional.** The cost applied to direct routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`staticRtMetric` *`integer`*	**Optional.** The cost applied to static routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`ripMetric` *`integer`*	**Optional.** The cost applied to RIP routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`directType {type1 | type2}`	**Optional.** Specifies the external link type that the vRouter advertises with direct routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Direct metric.
`staticRtType {type1 | type2}`	**Optional.** Specifies the external link type that the vRouter advertises with static routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Static metric.
`ripType {type1 | type2}`	**Optional.** Specifies the external link type that the vRouter advertises with RIP routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the RIP metric.

Example

In the following example, the router advertises direct routes as Type 1 with a metric of 5 and RIP routes as Type 2 with metric 8, but does not advertise static routes.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ospf
sun(config-vSwitch-e-commerce vRouter-default ospf)# advertise-ase ?
  [direct (enabled|disabled)]     Direct route (default: enabled)
  [staticRt (enabled|disabled)]   Static route (default: disabled)
  [rip (enabled|disabled)]        RIP route (default: disabled)
  [directMetric (0..16777215)]    Direct route metric (default: 1)
  [staticRtMetric (0..16777215)]  Static route metric (default: 1)
  [ripMetric (0..16777215)]       RIP route metric (default: 1)
  [directType (type1|type2)]      Direct route type (default: type1)
  [staticRtType (type1|type2)]    Static route (default: type1)
  [ripType (type1|type2)]         RIP route type (default: type1)

sun(config-vSwitch-e-commerce vRouter-default ospf)# advertise-ase
direct enabled rip enabled directMetric 5 ripMetric 8 directType type1
```

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → advertise-ase → add
- vSwitch → *name* → vRouter → *name* → ospf → advertise-ase → modify
- vSwitch → *name* → vRouter → *name* → ospf → advertise-ase → delete

advertise-nssa

Purpose

Controls the flow of non-OSPF routes in the not-so-stubby area (NSSA) and how they get advertised as NSSA type routes (Type 7). You can allow or disallow a route, based on type, to be advertised throughout the area. Once a route has been accepted in the NSSA, it can be advertised in the LSA as an external route. In addition, you can specify the type (Type 1 or Type 2) and the metric associated with them.

Access mode

config

Syntax

```
vSwitch name vRouter name ospf advertise-nssa
   direct {enabled | disabled}
   staticRt {enabled | disabled}
   rip {enabled | disabled}
   [directMetric integer]
   [staticRtMetric integer]
   [ripMetric integer]
   [directType {type1 | type2}]
   [staticRtType {type1 | type2}]
   [ripType {type1 | type2}]
```

Arguments

Argument name	Description
`direct {enabled | disabled}`	**Optional.** Allows or disallows direct routes to be injected into the NSSA and be advertised in the link state advertisement (LSA). The default is `enabled`.
`staticRt {enabled | disabled}`	**Optional.** Allows or disallows static routes to be injected into the NSSA and be advertised in the link state advertisement (LSA). The default is `disabled`.
`rip {enabled | disabled}`	**Optional.** Allows or disallows RIP routes to be injected into the NSSA and be advertised in the link state advertisement (LSA). The default is `disabled`.

Argument name	Description
`directMetric` *`integer`*	**Optional.** The cost applied to direct routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`staticRtMetric` *`integer`*	**Optional.** The cost applied to static routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`ripMetric` *`integer`*	**Optional.** The cost applied to RIP routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`directType {type1 | type2}`	**Optional.** Specifies the external link type that the vRouter advertises with direct routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Direct metric
`staticRtType {type1 | type2}`	**Optional.** Specifies the external link type that the vRouter advertises with static routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Static metric.
`ripType {type1 | type2}`	**Optional.** Specifies the external link type that the vRouter advertises with RIP routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the RIP metric

Example

In the following example, the router advertises direct routes as type 1 with a metric of 15 and static routes as type 1 with metric 5, but does not advertise RIP routes:

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ospf
sun(config-vSwitch-e-commerce vRouter-default ospf)# advertise-nssa ?
  [direct (enabled|disabled)]     Direct route (default: enabled)
  [staticRt (enabled|disabled)]   Static route (default: disabled)
  [rip (enabled|disabled)]        RIP route (default: disabled)
  [directMetric (0..16777215)]    Direct route metric (default: 1)
  [staticRtMetric (0..16777215)]  Static route metric (default: 1)
  [ripMetric (0..16777215)]       RIP route metric (default: 1)
  [directType (type1|type2)]      Direct route type (default: type1)
  [staticRtType (type1|type2)]    Static route (default: type1)
  [ripType (type1|type2)]         RIP route type (default: type1)
sun(config-vSwitch-e-commerce vRouter-default ospf)# advertise-nssa
direct enabled static enabled directMetric 15 StaticRtMetric 5
directType type1 staticType type1
```

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → advertise-nssa → add
- vSwitch → *name* → vRouter → *name* → ospf → advertise-nssa → modify
- vSwitch → *name* → vRouter → *name* → ospf → advertise-nssa → delete

area

Purpose

Configures basic characteristics of the OSPF area. These include the area ID, the area authentication type, the summary setting, and stub area type information. The area ID is an IP address that identifies the OSPF area. An OSPF system must be a part of at least one OSPF area.

The no form of the command deletes the specified area.

Access mode

user

Syntax

```
vSwitch name vRouter name ospf area
   id ipAddress
   [authType {none | simple Password | md5}]
   [areaType {nonStub | stub | nssa}]
   [summary {noAreaSummary | sendAreaSummary}]
   [stubMetric integer]
   [authKey text]
   [authId integer]
   [stubDefaultRoute {enabled | disabled}]
   [nssaDefaultRouteType {type1 | type2}]
```

Arguments

Argument name	Description
`id` *`ipAddress`*	Specifies an IP address that uniquely identifies the area. The address 0.0.0.0 is reserved for the OSPF backbone.
`authType {none | simplePassword | md5}`	**Optional.** Specifies the authentication type used throughout the OSPF area, and identified in the OSPF packet header. Select one of the following types: `none`—no authentication. `simplePassword`—A text password that the system includes in the OSPF updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the OSPF routers in the same area. This method is the least secure authentication method. `md5`—md5 cryptographic authentication, a 16-character authentication key. The system uses the MD5 algorithm to create an encoded checksum OSPF advertisement, which it includes in the OSPF advertisements it sends to other routers. The router that receives the OSPF update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password. The default authentication type is `none`.
`areaType {nonStub | stub | nssa}`	**Optional.** Specifies the type of area being created. The type defines how OSPF handles external routes that are redistributed into the area. `nonStub`—configures the area as a regular OSPF area, meaning that all external routes are flooded throughout the entire network. `stub`—configures a stub area, which prevents external routes form the backbone (area 0) from being flooded into the area. `nssa`—configures the area as a Not-So-Stubby-Area (NSSA), based on RFC 1587. The default area type is `nonStub`.

Argument name	Description
`summary {noAreaSummary | sendAreaSummary}`	**Optional.** For stub areas only. Specifies whether OSPF summary routes can be imported into the stub area. The `noAreaSummary` option prevents OSPF summary routes (Type-3 LSAs) from importing into the stub area. The `sendAreaSummary` option allows Type 3 summaries from entering the stub area. The default setting is `noAreaSummary`.
`stubMetric` *`integer`*	**Optional.** For stub areas only. Specifies the metric associated with the default route advertised into the stub area. The default cost is 1.
`authKey` *`text`*	**Optional**. Supplies the source text that OSPF uses to generate and verify the authentication field of the OSPF header. This is the password or MD5 key required for OSPF authentication. If using simplePassword authentication, enter a password of 8 characters or fewer. If using MD5, enter a key that the MD5 protocol uses to create the encoded checksum that the system sends in the OSPF advertisement. This value *must* match the keys configured on the other OSPF routers in the area.
`authId` *`integer`*	**Optional.** Assigns an authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID. **Note:** You must set the `authType` argument to `md5` to use this argument.
`stubDefaultRoute {enabled | disabled}`	**Optional.** Specifies whether the stub area receives an advertised default route. The default is `enabled`.
`nssaDefaultRouteType {type1 | type2}`	**Optional.** Specifies the route type of the default route when advertised into an NSSA area. Type 1 adds the internal cost to the external metric. Type 2 uses the internal cost to the border router servicing the external router. The default is `type2`.

Delete syntax

See the `show area` command for argument descriptions.

```
no vSwitch name vRouter name ospf area
   id ipAddress
   [authType {none |simple | md5}]
   [areaType {nonStub | stub | nssa}]
   [spfruns integer]
   [bdrrtrcount integer]
   [asbdrrtrcount integer]
   [lsacount integer]
   [lsacksumsum integer]
   [summary {noAreaSummary | sendAreaSummary}]
   [stubMetric integer]
   [authKey text]
   [authId integer]
   [stubDefaultRoute {enabled | disabled}]
   [nssaDefaultRouteType {type1 | type2}]
```

Example

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> area id 10.10.10.22
authtype simplePassword authkey admin
```

Associated MIB

```
ospfTbl.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → ospf → area → add
- vSwitch → *name* → vRouter → *name* → ospf → area → modify
- vSwitch → *name* → vRouter → *name* → ospf → area → delete

areaAggregate

Purpose

Defines the range that will be summarized by an aggregated route. The command configures an area border router (ABR) to summarize routes as specified by the IP address and subnet mask.

The no form of the command disables route aggregation.

Access mode

user

Syntax

```
vSwitch name vRouter name ospf areaAggregate
   areaid ipAddress
   lsdbtype {routerLink | networkLink | summaryLink | asSummaryLink |
      asExternalLink | multicastLink | nssaExternalLink}
   net ipAddress
   mask ipAddress
   [effect {advertiseMatching | doNotAdvertiseMatching}]
```

Arguments

Argument name	Description
`areaid` *`ipAddress`*	Specifies a IP address that uniquely identifies the area. The address 0.0.0.0 is reserved for the OSPF backbone.
`lsdbtype {summaryLink | nssaExternalLink}`	Specifies the type of link state advertisements from the database. Select one of the following: • `summaryLink` — An advertisement that describes interarea routes to a network (LS 3). • `nssaExternalLink` — An advertisement that describes routes to destinations that are external to the NSSA (Type 7).
`net` *`ipAddress`*	Specifies the network component of the IP address used to define the address range for summarization.
`mask` *`ipAddress`*	Specifies the subnet mask, used in conjunction with the `net` argument, to define the address range for summarization.
`effect {advertiseMatching | doNotAdvertiseMatching}`	Specifies whether summary link state advertisements (LSAs) are generated or suppressed. For `lsdbtype summaryLink`, if you select `advertiseMatching`, the system generates a Type 3 summary LSA. If you select `doNotAdvertiseMatching`, the system does not generate Type 3 summary LSAs. For `lsdbtype nssaExternalLink`, Type 7 LSAs are translated into Type 5 LSAs.

Delete syntax

See the “Arguments” section for this command for argument descriptions.

```
no vSwitch name vRouter name ospf areaAggregate
   areaid ipAddress
   lsdbtype {summaryLink | nssaExternalLink}
   net ipAddress
   mask ipAddress
   [effect {advertiseMatching | doNotAdvertiseMatching}]
```

Example

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> areaAggregate areaid
10.10.10.33 lsdbtype summaryLink net 10.10.10.1 mask 255.255.255.0
```

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → areaAggregate → add
- vSwitch → *name* → vRouter → *name* → ospf → areaAggregate → modify
- vSwitch → *name* → vRouter → *name* → ospf → areaAggregate → delete

globalSettings

Purpose

Modifies OSPF global settings, which control whether the protocol is enabled and which version of OSPF runs. In addition, you can assign an OSPF router ID to the vRouter.

Access mode

user

Syntax

```
vSwitch name vRouter name ospf globalSettings
   [adminstat {enabled | disabled}]
   [routerid ipAddress
   [rfc1583compatibility {enabled | disabled}]
```

Arguments

Argument name	Description
`adminstat {enabled | disabled}`	**Optional**. Sets the administrative state of OSPF on the vRouter, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring an OSPF offline or preconfigure it before bringing it online. OSPF is `enabled` by default.
`routerid ipAddress`	**Optional.** Assigns a router ID to OSPF.
`rfc1583compatibility {enabled | disabled}`	**Optional.** Sets the type of OSPF calculations to be performed, determined by the compatibility of routers within the area. If all OSPF routers in the area are using RFC 2328-based Version 2 or later implementation of OSPF, set this option to disabled. If any router is using an RFC 1583-based version, set this option to enabled. All routers in the area must use the same setting. The default setting is `enabled`.

Example

The following example assigns a router ID and sets the RFC compatibility to 2328.

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> globalSettings routerid
10.10.10.1 rfc1583compatibility disabled
```

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → globalSettings → modify

host

Purpose

Configures the hosts that are directly connected to this vRouter, and the metric and area ID to advertise for them.

The no form of the command deletes the specified host configuration.

Access mode

user

Syntax

```
vSwitch name vRouter name ospf host
   ipaddress ipAddress
   [metric integer]
   [areaid ipAddress]
```

Arguments

Argument name	Description
`ipaddress ipAddress`	Specifies the IP address of the directly connected host.
`metric integer`	**Optional.** Specifies the metric (value) to advertise with the host. The valid range is 1 to 65535. The default metric is 1.
`areaid ipAddress`	**Optional.** Specifies the area that the host resides in. The default area is 0.0.0.0.

Delete syntax

See the `show host` command for argument descriptions.

```
no vSwitch name vRouter name ospf host
   ipaddress ipAddress
   [metric integer]
   [areaid ipAddress]
```

Example

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> host ipaddress
10.10.10.1 metric 2 areaid 10.10.10.22
```

Associated MIB

```
ospfTbl.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → ospf → host → add
- vSwitch → *name* → vRouter → *name* → ospf → host → modify
- vSwitch → *name* → vRouter → *name* → ospf → host → delete

interface

Purpose

Creates or modifies an OSPF interface on the vRouter. An interface can only belong to one OSPF area. You must assign an interface to an area before OSPF can become operational.

The interface is defined by the IP address that you specify, but there are several optional parameters you can configure. For example, the default metric is 1, but you can change it to influence the cost of a route. You can also require authorization information to be included in packets that traverse the interface.

The no form of the command deletes the specified interface.

Access mode

user

Syntax

```
vSwitch name vRouter name ospf interface
   ipaddress ipAddress
   [areaid ipAddress]
   [adminstat {enabled | disabled}]
   [rtrpriority integer]
   [transitdelay integer]
   [retransinterval integer]
   [hellointerval seconds]
   [rtrdeadinterval seconds]
   [pollinterval integer]
   [metric integer]
   [authKey text]
   [authType {none | simplePassword | md5}]
   [authId integer]
```

Arguments

Argument name	Description
`ipaddress` *ipAddress*	Specifies the IP address associated with the OSPF interface.
`areaid` *ipAddress*	**Optional**. Specifies the area to which the interface is assigned, in IP address format. The default area is area 0.0.0.0 (area 0), which is the backbone.
`adminstat {enabled | disabled}`	**Optional**. Sets the administrative state of the interface, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring an OSPF interface offline or preconfigure it before bringing it online. The interface is `enabled` by default.
`rtrpriority` *integer*	**Optional**. Configures the priority of the vRouter. The value set is used when electing a designated router (DR). The router with the highest priority is elected the DR. In the case of a tie, the router with the highest router ID is elected. The valid range is 0 to 255. Enter 0 to disable the election. The default priority is 1.
`transitdelay` *integer*	**Optional**. Specifies the length of delay, in seconds, to advertise link state advertisement (LSA) transmissions out this interface. The default value is 1 second.
`retransinterval` *seconds*	**Optional**. Specifies the length of time, in seconds, between LSA retransmissions. The interface retransmits an LSA if it does not receive an acknowledgement for the original transmission. The vRouter waits for this interval to pass before retransmitting the LSA. The default value is 5 seconds.
`hellointerval` *seconds*	**Optional**. Specifies the length of time, in seconds, between hello packets sent out by the vRouter that this interface is associated with. The hello interval must be the same for all routers in the backbone area. The default number of seconds is 10.
`rtrdeadinterval` *seconds*	**Optional**. Specifies the number of seconds that a vRouter is considered “live” by its neighbors, even if they have not received any hello packets from it. The dead interval must be the same for routers in the backbone area. The default value is 40 seconds.
`pollinterval` *seconds*	**Optional**. Specifies the number of seconds between hello packets to a neighbor that is currently down. This value should usually be set higher than the hello interval. The default value is 120 seconds.

Argument name	Description
`metric` *integer*	**Optional**. Specifies the cost to assign to this interface. Routes advertised over this interface will have this value added to them. The default metric is 1.
`authKey` *text*	**Optional**. Supplies the source text that OSPF uses to generate and verify the authentication field of the OSPF header. This is the password or MD5 key required for OSPF authentication. If using simplePassword authentication, enter a password of 8 characters or fewer. If using MD5, enter a key that the MD5 protocol uses to create the encoded checksum that the system sends in the OSPF advertisement. This value *must* match the keys configured on the other OSPF routers in the network.

Argument name	Description
`authType {none | simplePassword | md5}`	**Optional**. Sets the type of authentication you want to use for OSPF advertisements sent from this interface. Valid values are: `none`: Do not use authentication. `simplePassword`: A text password that the system includes in the OSPF updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the OSPF interfaces in the same network. This method is the least secure authentication method. `md5`: The system uses the MD5 algorithm to create an encoded checksum OSPF advertisement, which it includes in the OSPF advertisements it sends to other routers. The router that receives the OSPF update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password. The default setting is `none`.
`authId` *`integer`*	**Optional.** Assigns an authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID. **Note:** You must set the `authType` argument to `md5` to use this argument.

Delete syntax

See the “Arguments” section of this command for argument descriptions.

```
no vSwitch name vRouter name ospf interface
   ipaddress ipAddress
   [areaid ipAddress]
   [adminstat {enabled | disabled}]
   [rtrpriority integer]
   [transitdelay integer]
   [retransinterval integer]
   [hellointerval seconds]
   [rtrdeadinterval seconds]
```

```
[pollinterval integer]
[metric integer]
[state {down | loopback | waiting | pointToPoint | designatedRouter
   | backupDesignatedRouter | otherDesignatedRouter}]
[designatedrouter ipAddress]
[backupdesignatedrouter ipAddress]
[events integer]
[authKey text]
[authType {none | simplePassword | md5}]
[authId integer]
```

Example

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> interface ipaddress
10.10.10.100 rtrpriority 2 authkey admin authtype simplePassword
```

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → interface → add
- vSwitch → *name* → vRouter → *name* → ospf → interface → modify
- vSwitch → *name* → vRouter → *name* → ospf → interface → delete

show advertise-ase

Purpose

Displays the settings for which route types can be redistributed into an OSPF area and advertised as external routes. This command does not support field filtering.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ospf advertise-ase
```

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show ospf advertise-ase
Direct:        enabled
Direct Metric: 1
Direct Type:   type1
Static:        disabled
Static Metric: 1
Static Type:   type1
Rip:           disabled
Rip Metric:    1
Rip Type:      type1
```

Output description

Field name	Description
Direct Routes	The setting allowing or disallowing direct routes to be redistributed into the OSPF area and advertised as external LSA.
Direct Metric	The metric applied to direct routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.

Field name	Description
`Direct Type`	The external link type that the vRouter advertises with direct routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Direct metric.
`Static Routes`	The setting allowing or disallowing static routes to be redistributed into the OSPF area and advertised in the LSA.
`Static Metric`	The metric applied to static routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`Static Type`	The external link type that the vRouter advertises with static routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Static metric.
`RIP Routes`	The setting allowing or disallowing RIP routes to be redistributed into the OSPF area and advertised in the LSA.
`RIP Metric`	The metric applied to RIP routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`RIP Type`	The external link type that the vRouter advertises with RIP routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the RIP metric.

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → advertise-ase

show advertise-nssa

Purpose

Displays the settings for which route types can be injected into an OSPF NSSA area and advertised as external routes. This command does not support field filtering.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ospf advertise-nssa
```

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show ospf advertise-nssa
Direct:        enabled
Direct Metric: 1
Direct Type:   type1
Static:        disabled
Static Metric: 1
Static Type:   type1
Rip:           disabled
Rip Metric:    1
Rip Type:      type1
```

Output description

Field name	Description
Direct Routes	The setting allowing or disallowing direct routes to be redistributed into the OSPF area and advertised as external LSA.
Direct Metric	The metric applied to direct routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.

Field name	Description
`Direct Type`	The external link type that the vRouter advertises with direct routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Direct metric.
`Static Routes`	The setting allowing or disallowing static routes to be redistributed into the OSPF area and advertised in the LSA.
`Static Metric`	The metric applied to static routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`Static Type`	The external link type that the vRouter advertises with static routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the Static metric.
`RIP Routes`	The setting allowing or disallowing RIP routes to be redistributed into the OSPF area and advertised in the LSA.
`RIP Metric`	The metric applied to RIP routes entering the OSPF routing domain. Valid range is 0 through 16777215; the default value is 1.
`RIP Type`	The external link type that the vRouter advertises with RIP routes redistributed into OSPF, either Type 1 or Type 2. The default is `type1`. Type 1 adds the internal cost to the external metric. Type 2 uses the RIP metric.

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → advertise-nssa

show area

Purpose

Displays the settings for a configured OSPF area, as well as various statistical counts.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ospf area
```

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show area
Id:                   10.10.10.22
AreaType:             nonStub
SpfRuns:              N/A
BdrRtrCount:          N/A
AsBdrRtrCount:        N/A
LsaCount:             N/A
LsaCksumSum:          N/A
Summary:              noAreaSummary
StubMetric:           1
NssaDefaultRouteType: type1
StubDefaultRoute:     enabled
AuthType:             simplePassword
AuthKey:              admin
AuthId:               0
```

Output description

Field name	Description	Filter name
Id	The IP address that uniquely identifies the area. The address 0.0.0.0 is reserved for the OSPF backbone.	id *ipAddress*

Field name	Description	Filter name
AreaType	The type of area, which defines how OSPF handles external routes that are redistributed into the area. nonStub—configures the area as a regular OSPF area, meaning that all external routes are flooded throughout the entire network. stub—configures a stub area, which prevents external routes form the backbone (area 0) from being flooded into the area. nssa—configures the area as a Not-So-Stubby-Area (NSSA), based on RFC 1587.	areaType {nonStub | stub | nssa}
SPF Runs	The number of times that the shortest path first (SPF) algorithm has been calculated. The calculation is done using this area's link state database, with the the Dijkstra's algorithm.	spfruns *integer*
Bdr Rtr Count	The number of area border routers that can be reached by this area.	bdrrtrcount *integer*
As Bdr Rtr Count	The number of autonomous system area border routers that can be reached by this area.	asbdrrtrcount *integer*
Lsa Count	The number of link state advertisements contained within this area's link state database. This count does not include external LSAs.	lsacount *integer*
Lsa Cksums	The sum of all checksums associated with the link-state advertisements in this area's link state database, excluding type 5 (external). This value is used to compare link state databases between routers.	lsacksumsum *integer*

Field name	Description	Filter name
`Summary`	The setting of whether OSPF summary routes can be imported into the stub area. The `noAreaSummary` option prevents OSPF summary routes (Type-3 LSAs) from importing into the stub area. The `sendAreaSummary` option prevents Type 3 summaries from entering the stub area.	`summary {noAreaSummary | sendAreaSummary}`
`StubMetric`	The metric associated with the default route advertised into the stub area.	`stubMetric` *`integer`*
`NssaDefaultRoute`	The type of default route advertised into an NSSA area.	`nssaDefaultRouteType {type1 | type2}`
`StubDefaultRoute`	The setting that controls whether the stub area receives an advertised default route.	`stubDefaultRoute {enabled | disabled}`

Field name	Description	Filter name
AuthType	The type of authentication you want to use for OSPF advertisements sent from this area. Valid values are: none: Do not use authentication. simplePassword: A text password that the system includes in the OSPF updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the OSPF routers in the same area. This method is the least secure authentication method. md5: The system uses the MD5 algorithm to create an encoded checksum OSPF advertisement, which it includes in the OSPF advertisements it sends to other routers. The router that receives the OSPF update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password. The default setting is none.	authType {none |simplePassword | md5}
AuthKey	The source text that OSPF uses to generate and verify the authentication field of the OSPF header. This is the password or MD5 key required for OSPF authentication. This value *must* match the keys configured on the other OSPF routers in the area.	authKey *text*

Field name	Description	Filter name
`AuthId`	The authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID.	`authId` *`integer`*

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → area

show areaAggregate

Purpose

Displays the settings that define an aggregated area.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf areaAggregate`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show areaAggregate
AreaID:   10.10.10.33
LsdbType: summaryLink
Net:      10.10.10.1
Mask:     255.255.255.0
Effect:   advertiseMatching
```

Output description

Field name	Description	Filter name
`Area Id`	The IP address that uniquely identifies the area. The address 0.0.0.0 is reserved for the OSPF backbone.	`areaid` *`ipAddress`*
`Lsdb Type`	The type of link state advertisements from the database: `summaryLink`—an advertisement that describes interarea routes to a network (LS 3). `nssaExternalLink`—an advertisement that describes routes to destinations that are external to the NSSA (Type 7).	`lsdbtype {summaryLink | nssaExternalLink}`

Field name	Description	Filter name
`Network`	The network component of the IP address used to define the address range for summarization.	`net` *`ipAddress`*
`Mask`	The subnet mask, used in conjunction with the `net` argument, to define the address range for summarization.	`mask` *`ipAddress`*
`Effect`	Specifies whether summary link state advertisements (LSA) are generated or suppressed. For `lsdbtype summaryLink`, if you select `advertiseMatching`, the system generates a Type 3 summary LSA. If you select `doNotAdvertiseMatching`, the system does not generate Type 3 summary LSAs. For `lsdbtype nssaExternalLink`, Type 7 LSAs are translated into Type 5 LSAs.	`effect {advertiseMatching | doNotAdvertiseMatching}`

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → areaAggregate

show extLsdb

Purpose

Displays characteristics of the external link state database, with one entry for each unique link state ID.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf extLsdb`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show extLsdb
Type:                       asExternalLink
Lsid:                       10.10.75.1
RouterId:                   10.10.10.1
Sequence:                   2147483649
Age:                        857
Checksum:                   12705
AS External Route Type:     type1
AS External Route Metric: 1

Type:                       asExternalLink
Lsid:                       10.10.80.1
RouterId:                   10.10.10.1
Sequence:                   2147483649
Age:                        857
Checksum:                   63955
AS External Route Type:     type1
AS External Route Metric: 1

Type:                       asExternalLink
Lsid:                       10.10.80.2
RouterId:                   10.10.10.1
Sequence:                   2147483649
Age:                        857
Checksum:                   61404
AS External Route Type:     type1
AS External Route Metric: 1
```

Field name	Description	Filter name
`Type`	The type of link state advertisements from the database. `asExternalLink`— An advertisement that describes routes to destinations that are external to the AS (LS5).	`type` ***type***
`Lsid`	The router ID or IP address that identifies the part of the routing domain that is being described by the advertisement.	`lsid` ***ipAddress***
`RouterId`	The router ID setting of the originating router.	`routerid` ***ipAddress***
`Sequence`	A number used to verify the integrity of a link state advertisement. Each LSA is assigned a number sequentially, the larger numbers are more recent, and they help to keep all items in the database current.	`sequence` ***integer***
`Age`	The age, in seconds, of the link state advertisement.	`age` ***integer***
`Checksum`	A value used to validate the integrity of a link state advertisement. This is a checksum of the complete contents of the advertisement.	`checksum` ***integer***
`AS External Route Type`	The external link type that the vRouter advertises with external routes redistributed into OSPF, either Type 1 or Type 2.	`rtType {type1|type2}`
`AS External Route Metric`	The cost applied to external routes entering the OSPF routing domain. V	`rtMetric` ***integer***

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → extLsdb

show globalSettings

Purpose

Displays general OSPF settings and link state advertisement counts. This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf globalSettings`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show globalSettings
Configured RouterId:    10.10.10.1
Active RouterId:        0.0.0.0
AdminStat:              enabled
VersionNumber:          version2
AreaBdrRtrStatus:       false
ExternLsaCount:         0
ExternLsaCksumSum:      0
OriginateNewLsas:       0
RxNewLsas:              0
RFC 1583 Compatibility: enabled
```

Output description

Field name	Description
`Router Id`	The IP address configured, with the `globalSettings` command, to be the router ID. A router ID of 0.0.0.0 indicates that the ID is not configured.

Field name	Description
`Active RouterId`	The router ID that is currently in use for this OSPF router. This will be the configured router ID if not 0.0.0.0, otherwise it will be a configured IP address that is up.
`Admin State`	The administrative state of OSPF on the vRouter, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring an OSPF offline or preconfigure it before bringing it online. OSPF is `enabled` by default.
`Version Number`	The version of OSPF being used by this vRouter.
`Area Bdr Rtr Status`	The status of the router, either true or false, which identifies whether the router is an area border router. If `true`, the router is an ABR; if `false`, it is not.
`Extern Lsa Count`	The number of external link state advertisements received by this vRouter.
`External Lsa CksumSums`	The sum of all external LSA checksums.
`Originate New Lsas`	The number of new link state advertisements this vRouter has originated.
`Rx New Lsas`	The number of new link state advertisements this vRouter has received.
`RFC 1583 Compatibility`	The type of OSPF calculations to be performed, determined by the compatibility of routers within the area. If all OSPF routers in the area are using RFC 2328-based Version 2 or later implementation of OSPF, set this option to disabled. If any router is using an RFC 1583-based version, set this option to enabled. All routers in the area must use the same setting.

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → globalSettings

show host

Purpose

Displays settings for directly connected hosts.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ospf host
```

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show host
IpAddress: 10.10.10.1
Metric:    2
AreaID:    10.10.10.22
```

Output description

Field name	Description	Filter field
IpAddress	The IP address of the directly connected host.	ipaddress *ipAddress*
Metric	The metric (value) to advertise with the host.	metric *integer*
AreaID	The area in which the host resides.	areaid *ipAddress*

Associated MIB

ospfTbl.mib

Web path

- vSwitch → *name* → vRouter → *name* → ospf → host

show interface

Purpose

Displays characteristics of configured OSPF interfaces.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf interface`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)# show interface
IpAddress:              10.10.10.100
AreaId:                 0.0.0.0
AdminStat:              enabled
RtrPriority:            2
TransitDelay:           1
RetransInterval:        5
HelloInterval:          10
RtrDeadInterval:        40
PollInterval:           120
State:                  N/A
DesignatedRouter:       N/A
BackupDesignatedRouter: N/A
Events:                 N/A
metric:                 1
AuthType:               simplePassword
AuthKey:                admin
AuthId:                 0
```

Output description

Field name	Description	Filter field
`IpAddress`	The IP address associated with the OSPF interface.	`ipaddress` *`ipAddress`*
`AreaId`	The area to which the interface is assigned, in IP address format. The default area is area 0.0.0.0 (area 0), which is the backbone.	`areaid` *`ipAddress`*
`AdminState`	The administrative state of OSPF on this interface, either `enabled` or `disabled`.	`adminstat {enabled | disabled}`
`RtrPriority`	The priority of the vRouter. The value set is used when electing a designated router (DR). The router with the highest priority is elected the DR. In the case of a tie, the router with the highest router ID is elected.	`rtrpriority` *`integer`*
`TransitDelay`	The length of delay, in seconds, to advertise link state advertisement (LSA) transmissions out this interface.	`transitdelay` *`seconds`*
`RetransInterval`	The length of time, in seconds, between LSA retransmissions. The interface retransmits an LSA if it does not receive an acknowledgement for the original transmission. The vRouter waits for this interval to pass before retransmitting the LSA.	`retransinterval` *`seconds`*
`HelloInterval`	The length of time, in seconds between hello packets sent out by the vRouter on this interface.	`hellointerval` *`seconds`*
`RtrDeadInterval`	The number of seconds that a vRouter is considered “live” by its neighbors, even if they have not received any hello packets from it. The dead interval must be the same for routers in the backbone area.	`rtrdeadinterval` *`seconds`*

Field name	Description	Filter field
`PollInterval`	The number of seconds between hello packets to a neighbor that is currently down. This value should usually be set higher than the hello interval.	`pollinterval` *`integer`*
`Metric`	The cost to assign to this interface. Routes advertised over this interface will have this value added to them.	`metric` *`integer`*

Field name	Description	Filter field
`State`	The state that the vRouter interface is currently in: `down`—the initial interface state, no traffic is sent or received. `loopback`—the interface is looped back and cannot process regular traffic, but can respond to pings and bit testing. `waiting`—the router is determining the network's designated router; no DR or backup DR elections can be made. `pointToPoint`—the interface is operational and the router is attempting to form adjacencies. THe interface could be connected either to a physical point-to-point network or to a virtual link. `designatedRouter`—This vRouter is the designated router on the attached network, and establishes adjacencies with all other network routers. `backupDesignatedRouter`—This vRouter is the backup designated router on the attached network, and establishes adjacencies with all other network routers. `otherDesignatedRouter`—the interface is connected to a broadcast or NBMA network. THe designated router resides on that network.	`state {down | loopback | waiting | pointToPoint | designatedRouter | backupDesignatedRouter | otherDesignatedRouter}`

Field name	Description	Filter field
`DesignatedRouter`	The IP address of the vRouter currently acting as the designated router. Use the `rtrpriority` argument to influence DR election.	`designatedRouter` *`ipAddress`*
`BackupDesignated Router`	The IP address of the vRouter currently acting as the backup designated router. Use the `rtrpriority` argument to influence DR election.	`backupDesignatedRouter` *`ipAddress`*
`Events`	The number of times this interface has changed state or experienced an error.	`events` *`integer`*
`AuthKey`	The source text that OSPF uses to generate and verify the authentication field of the OSPF header. This is the password or MD5 key required for OSPF authentication. This value *must* match the keys configured on the other OSPF routers in the network.	`authkey` *`text`*

Field name	Description	Filter field
`AuthType`	The type of authentication you want to use for OSPF advertisements sent from this interface. Valid values are: `none`: Do not use authentication. `simplePassword`: A text password that the system includes in the OSPF updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the OSPF interfaces in the same network. This method is the least secure authentication method. `md5`: The system uses the MD5 algorithm to create an encoded checksum OSPF advertisement, which it includes in the OSPF advertisements it sends to other routers. The router that receives the OSPF update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password.	`authtype {none | simplePassword | md5}`
`AuthId`	The authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID.	`authId` *`integer`*

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → interface

show lsdb

Purpose

Displays the contents of the OSPF link state database.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf lsdb`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show lsdb
AreaId:   10.10.10.22
Type:     routerLink
Lsid:     10.10.10.1
RouterId: 10.10.10.1
Sequence: 2147483649
Age:      193
Checksum: 34156
```

Output description

Field name	Description	Filter name
`AreaId`	The identifier for the area in which this vRouter is configured.	`areaid` *`ipAddress`*
`Type`	The type of link state advertisements from the database. Either: `routerLink`— a router's link advertisement that describes the state of the router's link to the area (LS 1). `networkLink`—an advertisement that describes a networks attached routers (LS 2). `summaryLink`—an advertisement that describes interarea routes to a network (LS 3). `asSummaryLink`—an advertisement that describes interarea routes to a AS border router (LS 4). `multicastLink`—an advertisement that describes routes that are part of a multicast domain. `nssaExternalLink`—an advertisement that describes routes to destinations that are external to the NSSA.	`type {routerLink | networkLink | summaryLink | asSummaryLink | multicastLink | nssaExternalLink)`
`Lsid`	The router ID or IP address that identifies the part of the routing domain that is being described by the advertisement.	`lsid` *`ipAddress`*
`RouterId`	The router ID setting of the originating router.	`routerid` *`ipAddress`*

Field name	Description	Filter name
Sequence	A number used to verify the integrity of a link state advertisement. Each LSA is assigned a number sequentially, the larger numbers are more recent, and they help to keep all items in the database current.	sequence *integer*
Age	The age, in seconds, of the link state advertisement.	age *integer*
Checksum	A value used to validate the integrity of a link state advertisement. This is the checksum of the complete contents of the advertisement.	checksum *integer*

Associated MIB

ospfTbl.mib

Web path

- vSwitch → *name* → vRouter → *name* → ospf → lsdb

show neighbors

Purpose

Displays settings for a neighboring router with which this router formed an adjacency.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf neighbors`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show neighbors
IpAddr: 15.15.15.4
RtrId: 14.14.14.4
Options: 2
Priority: 1
State: full
Events: 6
LsRetransQLen: 0
HelloSuppressed: false
```

Output description

Field name	Description	Filter name
IpAddr	The IP address of a neighbor. A neighbor is a router with which this router was allowed to form a neighbor adjacency.	ipaddr *ipAddress*
RtrId	The ID setting of the neighbor's router.	rtrid *ipAddress*

Field name	Description	Filter name
`Options`	The setting of the neighbor's options field: Bit 0 indicates that the system will operate on Type of Service metrics other than TOS 0. If zero, the neighbor will ignore all metrics except the TOS 0 metric. Bit 1 indicates that the associated area accepts and operates on external information; if zero, it is a stub area. Bit 2 indicates that the system is capable of routing IP Multicast datagrams; i.e., that it implements the Multicast Extensions to OSPF. Bit 3 indicates that the associated area is an NSSA. These areas are capable of carrying type 7 external advertisements, which are translated into type 5 external advertisements at NSSA borders.	`options` *`integer`*
`Priority`	An 8-bit number router priority of the non-broadcast neighbor associated with the specified IP address.	`priority` *`integer`*

Field name	Description	Filter name
`State`	The state of the neighbor relationship: `down`: the initial state of the neighbor relationship. `attempt`: the router is sending Hello packets in an effort to establish contact with the neighbor; only valid when attached to an NBMA network. `init`: a hello packet has been received but communication is not yet established. `twoWay`: bidirectional communication is established, no adjacencies formed yet. `exchangeStart`: the first step in adjacency formation, the routers are determining roles. `exchange`: the neighbors are sending database description packets to each other; adjacencies are capable of sending and receiving OSPF traffic. `loading`: the neighbors request link state request packets with the most recent LSAs. `full`: the neighbors have established full adjacency.	`state {down | attempt | init | twoWay | exchangeStart | exchange | loading | full)`
`Events`	The number of times this neighbor has changed state or experienced an error.	`events` *`integer`*
`LsRetransQLen`	The number of packets in the retransmission queue.	`lsretransqlen` *`integer`*
`HelloSuppressed`	The setting on the hello suppression option. If `true`, the vRouter is suppressing hellos to the neighbor; if `false`, it is not.	`hellosuppressed {true | false}`

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → neighbors

show virtualInterface

Purpose

Displays characteristics of configured OSPF virtual interfaces. This interface is a virtual link between two area border routers.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ospf virtualInterface`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show virtualinterface
AreaId:          10.10.10.100
Neighbor:        10.10.10.200
rtrPriority:     1
TransitDelay:    1
RetransInterval: 5
HelloInterval:   10
RtrDeadInterval: 40
PollInterval:    120
State:           N/A
Events:          N/A
AuthKey:         admin
AuthType:        simplePassword
AuthId:          0
```

Output description

Field name	Description	Filter name
`AreaId`	The area to which the virtual interface transits, in IP address format. The default area is area 0.0.0.0 (area 0), which is the backbone.	`areaid` *`ipAddress`*
`Neighbor`	The IP address of a neighbor with which this interface can form an adjacency.	`neighbor` *`ipAddress`*
`rtrPriority`	The priority of the vRouter. The value set is used when electing a designated router (DR). The router with the highest priority is elected the DR. In the case of a tie, the router with the highest router ID is elected.	`rtrpriority` *`integer`*
`TransitDelay`	The length of delay, in seconds, between link state advertisement (LSA) transmissions out this virtual interface.	`transitdelay` *`seconds`*
`RetransInterval`	The length of time, in seconds, to advertise link state advertisement (LSA) transmissions out this virtual interface.	`retransinterval` *`seconds`*
`HelloInterval`	The length of time, in seconds between hello packets sent out by the vRouter on this interface.	`hellointerval` *`seconds`*
`RtrDeadInterval`	The number of seconds that a vRouter is considered “live” by its neighbors, even if they have not received any hello packets from it. The dead interval must be the same for routers in the backbone area.	`rtrdeadinterval` *`seconds`*
`PollInterval`	The number of seconds between hello packets to a neighbor that is currently down. This value should usually be set higher than the hello interval.	`pollinterval` *`seconds`*

Field name	Description	Filter name
`State`	The state of the virtual interface, either down or point-to-point. If the state is `down`, the interface is not operational. When active, the interface has a state of `pointToPoint`, which means that is providing connectivity through the backbone to the specified neighbor.	`state {down | pointToPoint}`
`Events`	The number of times this virtual connection has changed state or experienced an error.	`events` *`integer`*
`AuthKey`	The source text that OSPF uses to generate and verify the authentication field of the OSPF header. This is the password or MD5 key required for OSPF authentication. This value *must* match the keys configured on the other OSPF routers in the network.	`authKey` *`text`*

Field name	Description	Filter name
`AuthType`	The type of authentication you want to use for OSPF advertisements sent from this virtual interface. Valid values are: `none`: Do not use authentication. `simplePassword`: A text password that the system includes in the OSPF updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the OSPF interfaces in the same network. This method is the least secure authentication method. `md5`: The system uses the MD5 algorithm to create an encoded checksum OSPF advertisement, which it includes in the OSPF advertisements it sends to other routers. The router that receives the OSPF update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password.	`authType {none | simplePassword | md5}`
`AuthId`	The authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID.	`authId` *`integer`*

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → virtualInterface

show virtualNeighbors

Purpose

Displays the virtual neighbor routers for which this vRouter has formed an adjacency.

Access mode

user

Syntax

```
show vSwitch-name vRouter-name ospf virtualNeighbors
```

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> show virtualNeighbors
Area: 1.1.1.1
RtrId: 14.14.14.4
IpAddr: 15.15.15.4
Options: 0
State: full
Events: 6
LsRetransQLen: 0
HelloSuppressed: false
```

Output description

Field name	Description	Filter name
Area	The area to which this virtual neighbor is assigned.	area *ipAddress*
RtrId	The ID setting of the virtual neighbor's router.	rtrid *ipAddress*
IpAddr	The IP address of a virtual neighbor. A neighbor is a router with which this router was allowed to form a neighbor adjacency.	ipaddr *ipAddress*

Field name	Description	Filter name
`Options`	The setting of the neighbor's options field: Bit 0 indicates that the system will operate on Type of Service metrics other than TOS 0. If zero, the neighbor will ignore all metrics except the TOS 0 metric. Bit 1 indicates that the associated area accepts and operates on external information; if zero, it is a stub area. Bit 2 indicates that the system is capable of routing IP Multicast datagrams; i.e., that it implements the Multicast Extensions to OSPF. Bit 3 indicates that the associated area is an NSSA. These areas are capable of carrying type 7 external advertisements, which are translated into type 5 external advertisements at NSSA borders.	`options` *`integer`*

Field name	Description	Filter name
`State`	The state of the virtual neighbor relationship: `down`: the initial state of the neighbor relationship. `attempt`: the router is sending Hello packets in an effort to establish contact with the neighbor; only valid when attached to an NBMA network. `init`: a hello packet has been received but communication is not yet established. `twoWay`: bidirectional communication is established, no adjacencies formed yet. `exchangeStart`: the first step in adjacency formation, the routers are determining roles. `exchange`: the neighbors are sending database description packets to each other; adjacencies are capable of sending and receiving OSPF traffic. `loading`: the neighbors request link state request packets with the most recent LSAs. `full`: the neighbors have established full adjacency.	`state {down | attempt | init | twoWay | exchangeStart | exchange | loading | full}`
`Events`	The number of times this virtual neighbor has changed state or experienced an error.	`events` *`integer`*
`LsRetransQLen`	The number of packets in the virtual neighbor's retransmission queue.	`lsretransqlen` *`integer`*

Field name	Description	Filter name
`HelloSuppressed`	The setting on the hello suppression option. If `true`, the vRouter is suppressing hellos to the virtual neighbor; if `false`, it is not.	`hellosuppressed {true | false}`

Associated MIB

`ospfTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ospf → virtualNeighbor

virtualInterface

Purpose

Configures a virtual link between this vRouter and the specified neighbor. This allows OSPF routers that are not physically connected to the backbone to be part of the backbone and to be an area border router (ABR). Virtual links use transit areas for connectivity to the backbone. A virtual link must be configured on each router specifying the other router.

Additionally, the router not physically connected to the backbone must still have a configured backbone area containing the same settings as the router physically on the backbone.

The no form of the command deletes the specified virtual interface.

Access mode

user

Syntax

```
vSwitch name vRouter name ospf virtualInterface
   areaid ipAddress
   neighbor ipAddress
   [transitdelay integer]
   [retransinterval integer]
   [hellointerval seconds]
   [rtrdeadinterval seconds]
   [rtrpriority integer]
   [pollinterval integer]
   [authKey text]
   [authType {none | simplePassword | md5}]
   [authId integer]
```

Arguments

Argument name	Description
`areaid` *ipAddress*	Specifies the area to which the virtual interface transits, in IP address format. The default area is area 0.0.0.0 (area 0), which is the backbone.
`neighbor` *ipAddress*	Specifies the IP address of a neighbor with which this interface can form an adjacency. This neighbor serves as the far end of the virtual link to this vRouter.
`transitdelay` *integer*	**Optional**. Specifies the length of delay, in seconds, between link state advertisement (LSA) transmissions out this interface. The default value is 1 second.
`retransinterval` *seconds*	**Optional**. Specifies the length of time, in seconds, to advertise link state advertisement (LSA) transmissions out this virtual interface. The default value is 5 seconds.
`hellointerval` *seconds*	**Optional**. Specifies the length of time, in seconds, between hello packets sent out by the vRouter that this virtual interface is associated with. The hello interval must be the same for all routers in the backbone area. The default number of seconds is 10.
`rtrdeadinterval` *seconds*	**Optional**. Specifies the number of seconds that a vRouter is considered “live” by its neighbors, even if they have not received any hello packets from it. The dead interval must be the same for routers in the backbone area. The default value is 40 seconds.
`rtrpriority` *integer*	**Optional**. Configures the priority of the vRouter. The value set is used when electing a designated router (DR). The router with the highest priority is elected the DR. In the case of a tie, the router with the highest router ID is elected. The valid range is 0 to 255. Enter 0 to disable the election. The default priority is 1.
`pollinterval` *seconds*	**Optional**. Specifies the number of seconds between hello packets to a neighbor that is currently down. This value should usually be set higher than the hello interval. The default value is 120 seconds.

Argument name	Description
`authKey` *`text`*	**Optional**. Supplies the source text that OSPF uses to generate and verify the authentication field of the OSPF header. This is the password or MD5 key required for OSPF authentication. If using simplePassword authentication, enter a password of 8 characters or fewer. If using MD5, enter a key that the MD5 protocol uses to create the encoded checksum that the system sends in the OSPF advertisement. This value *must* match the keys configured on the other OSPF routers in the network.
`authType {none | simplePassword | md5}`	**Optional**. Sets the type of authentication you want to use for OSPF advertisements sent from this virtual interface. Valid values are: `none`: Do not use authentication. `simplePassword`: A text password that the system includes in the OSPF updates it sends to other routers. The password must be 15 or fewer characters. You must configure the same password on all of the OSPF interfaces in the same network. This method is the least secure authentication method. `md5`: The system uses the MD5 algorithm to create an encoded checksum OSPF advertisement, which it includes in the OSPF advertisements it sends to other routers. The router that receives the OSPF update uses the authentication key to verify the packet, discarding it if the digest does not match. This method is more secure than using a simple password. The default setting is `none`.
`authId` *`integer`*	**Optional.** Assigns an authentication identification (ID) number used during MD5 authentication. This value *must* match the authentication ID configured on the other routers in the network. The default setting is 0. Not all routers use the authentication ID; you can use the default value if your routers do not use an authentication ID. **Note:** You must set the `authType` argument to `md5` to use this argument.

Delete syntax

See the "Arguments" section for this command for argument descriptions.

```
no vSwitch name vRouter name ospf virtualInterface
   areaid ipAddress
   neighbor ipAddress
   [transitdelay integer]
   [retransinterval integer]
   [hellointerval integer]
   [rtrdeadinterval seconds]
   [rtrpriority integer]
   [pollinterval integer]
   [state {down | pointToPoint}]
   [events integer]
   [authKey text]
   [authType {none | simplePassword | md5}]
   [authId integer]
```

Example

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ospf
sun(vSwitch-e-commerce vRouter-default ospf)> virtualInterface areaid
10.10.10.100 neighbor 10.10.10.200 authkey admin authType
simplePassword
```

Associated MIB

```
ospfTbl.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → ospf → virtualInterface → add
- vSwitch → *name* → vRouter → *name* → ospf → virtualInterface → modify
- vSwitch → *name* → vRouter → *name* → ospf → virtualInterface → delete

Chapter 26. ARP, ICMP, and IRDP commands

IP protocol command descriptions

This chapter describes the Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), and Internet Domain Routing Protocol (IRDP) commands available from the N2000 Series system. The system supports IP Version 4. The N2000 Series supports several other IP-related command functions, which are detailed in the following chapters.

IP feature	Chapter
IP interfaces and addressing	Chapter 22.
Routing Information Protocol (RIP) and static routing	Chapter 24.
Ping and traceroute utilities	Chapter 27.
Access control lists (ACLs)	Chapter 28.

ARP description

Address Resolution Protocol (ARP), defined in RFC 826, is the mechanism the system uses to map IP addresses to Layer 2 addresses (the physical network). The system establishes the entries by broadcasting unknown IP addressees to local hosts. Any host that recognizes the IP address responds with the corresponding Layer 2 MAC address.These mappings are stored in the ARP table. The system supports 3000 ARP entries per vRouter.

ICMP description

Internet Control Message Protocol (ICMP), defined in RFC 792, is a protocol used to determine whether a destination is unreachable. A TCP/IP-based protocol, ICMP verifies, through error and control messages between a host and an Internet gateway, the validity of an IP address. For example, ICMP functions are used by the `ping` utility to verify network connectivity.

IRDP description

Internet Router Discovery Protocol, defined in RFC 1256, is a mechanism to allow hosts to determine a preferred default gateway for client traffic. IRDP is a dynamic protocol that allows clients to automatically recover from network failures. That is, if an IRDP client encounters a failed gateway in its path or an administrative redirect, IRDP can update the next-hop information to based on configured preference information.

The N2000 Series functions as an IRDP server. An IRDP server broadcasts or multicasts router advertisements to all IRDP-configured interfaces to which it is attached. For each vRouter interface, the advertisement lists all addresses and the corresponding preferences. The address with the highest preference is selected as the default router address (and therefore, default gateway) for that interface.

IP protocols command path

The command names in this chapter show you how to execute the commands from within the following command modes:

```
vSwitch name vRouter name ip arp
vSwitch name vRouter name ip icmp
vSwitch name vRouter name irdp
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

IP protocol command summary

Table 26-1 lists and briefly describes the IP protocol commands.

Table 26-1. ARP, ICMP, and IRDP command summary

Command Name	Description
`arp flush`	Remove the specified entry from the ARP table.
`arp settings`	Configure basic ARP parameters.
`arp static`	Manually add entries to the ARP table.
`icmp`	Configure basic ICMP parameters.
`irdp addresses`	Set the system to advertise the specified address(es) as a default gateway.
`irdp interfaces`	Activate IRDP on the specified interface.
`show arp settings`	Display the ARP configuration parameters.
`show arp static`	Display the manually configured (static) entries in the system's ARP table.
`show arp statistics`	Display ARP-specific statistics for all or a specified interface.
`show icmp`	Display ICMP configuration settings.
`show icmp inStats`	Display statistics for ICMP messages received on this instance of IP.
`show icmp outStats`	Display statistics for ICMP messages transmitted on this instance of IP.
`show irdp addresses`	Display each address configured as a default gateway and its preference setting.
`Show irdp interfaces`	Display all interfaces configured to run IRDP.

arp flush

Purpose

Removes the specified entry from the ARP table. If you do not specify any arguments, the system removes all entries from its ARP table. Optionally, you can identify specific entries by interface name or IP address.

Access mode

user

Syntax

```
vSwitch-name vRouter-name ip arp flush
   [ifName ifIndex]
   [ipAddress ipAddress]
```

Argument

Argument name	Description
`ifName` *`ifName`*	**Optional.** Specifies the name of the interface whose ARP entries you want to remove from the ARP table (the interface that connects to the IP instance). This could be either `vlan.`*`x`*, `lag.`*`x`*, or `eth.`*`x.x`*. If the interface does not already exist, the system returns an error. Use the `show ip interface` command to verify configured IP-to-lower layer interface associations. Use the `ip interface` command to configure new associations.
`ipAddress` *`ipAddress`*	**Optional.** The IP address associated with the ARP entry you want to delete.

Example

The following example removes the ARP mapping for interface eth.1.20.

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)> vRouter default
sun(vSwitch-e-commerce vRouter-default)> ip
sun(vSwitch-e-commerce vRouter-default ip)> arp
sun(vSwitch-e-commerce vRouter-default ip arp)> flush ifName eth.1.20
```

Associated MIB

```
arp.mib
```

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp → flush

arp settings

Purpose

Configures basic ARP parameters that control the table-building mechanism. With this command you can set the valid lifetime for an ARP entry in the table, set the number of times the system will broadcast an IP address in search of Layer 2 information, and set the wait time on broadcast requests.

Access mode

config

Syntax

```
vSwitch-name vRouter-name ip arp settings
   [agingTimeout integer]
   [retryCount integer]
   [requestTimeout integer]
   [checkForDuplicateMacs {enabled | disabled}]
```

Arguments

Argument name	Description
`agingTimeout` *integer*	**Optional.** Specifies the amount of time, in seconds, that an ARP entry remains valid and in the ARP table. When the timer has expired, the system deletes the entry from the table. The valid range is 0 through 86400 seconds. The default timer is 300 seconds; a value of 0 disables the time out feature, and entries never age out.
`retryCount` *integer*	**Optional.** Specifies the number of times the system will rebroadcast an unknown IP address to local hosts looking for a Layer 2 mapping. The valid range is 0 through 100 retries; the default value is 2.
`requestTimeout` *integer*	**Optional.** Specifies the number of seconds the system waits between sending out ARP requests. The valid range is 1 through 20 seconds; the default value is 2 seconds.

Argument name	Description
`checkForDuplicateMacs {enabled | disabled}`	**Optional.** Specifies whether to check for duplicate MAC addresses when learning ARP entries. If you want two vRouters on the same switch looped back to one another, disable this function instead of setting the router MAC address on a port. The default is `enabled`.

Example

The following example configures the system to rebroadcast an unknown address five times.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# arp
sun(config-vSwitch-e-commerce vRouter-default ip arp)# settings 5
```

Associated MIB

`arp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp → settings → modify

arp static

Purpose

Allows you to manually add entries to the ARP table. The system typically establishes these mappings by broadcasting unknown IP addresses to local hosts and awaiting a reply. A host that recognizes the IP address responds with the corresponding Layer 2 MAC address. This command, however, allows you to manually configure IP address-to-Layer 2 physical address mappings.

Because an interface can be assigned more than one IP address, you must specify the interface name and address, as well as the MAC address.

The no form of the command deletes the manual mapping.

Access mode

config

Syntax

To create a static entry:

```
vSwitch-name vRouter-name ip arp static
   ifName ifName
   netAddr ipAddress
   physAddr macAddress
```

To modify a static entry:

```
vSwitch-name vRouter-name ip arp static
   ifName ifName
   netAddr ipAddress
   [physAddr macAddress]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the name of the interface for which you want to configure an IP address-to-physical address mapping. This is the interface that connects to the IP instance, either `vlan.`*`x`*, `lag.`*`x`*, or `eth.`*`x.x`*. If the interface does not already exist, the system returns an error. Use the `show ip interface` command to verify configured IP-to-lower layer interface associations. Use the `ip interface` command to configure new associations.
`netAddr` *`ipAddress`*	Specifies the IP address the system should associate with the physical address specified. This pairing is stored in the system's ARP table.
`physAddr` *`macAddress`*	Specifies the Layer 2 physical address (the MAC address) to associate with the specified IP address/ interface. This argument is required when creating a static ARP entry, but optional when modifying or deleting it.

Delete syntax

See the "Arguments" section for this command for argument descriptions.

```
no vSwitch-name vRouter-name ip arp static
   ifNameifName
   netAddr ipAddress
   [physAddr macAddress]
```

Example

The following example maps the IP-to-MAC address for interface eth.1.20.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# arp
sun(config-vSwitch-e-commerce vRouter-default ip arp)# static ifName
eth.1.20 netAddr 10.10.10.9 physAddr 00:e0:2b:5f:ca:00
```

Associated MIB

`arp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp → static → add
- vSwitch → *name* → vRouter → *name* → Ip → arp → static → copy
- vSwitch → *name* → vRouter → *name* → Ip → arp → static → modify
- vSwitch → *name* → vRouter → *name* → Ip → arp → static → delete

icmp

Purpose

Sets some basic options of the Internet Control Message Protocol (ICMP).

Access mode

config

Syntax

```
vSwitch-name vRouter-name ip icmp
   [replyToEchos {true | false}]
   [sendTimeExceeds {true | false}]
   [sendParamProbs {true | false}]
   [replyToMasks {true | false}]
```

Arguments

Argument name	Description
`replyToEchos`	**Optional.** Sets the system's ability to reply to ICMP echo requests. If set to `true`, the system responds to these requests (typically `ping` packets); if set to `false` the packet is dropped. The default setting is `true`.
`sendTimeExceeds`	**Optional.** Sets the system's ability to send an ICMP TTL exceeded error message. If set to `true`, the system sends this message to the error log. When the time-to-live timer has expired in the IP header. If set to `false`, the packet is dropped. The default setting is `true`.
`sendParamProbs`	**Optional.** Sets the system's ability to send an ICMP Parameter Problem error message. If set to `true`, the system forwards the packet and responds to the source with this message if the IP header contains options that are not supported on the N2000 Series system. If set to `false`, the system forwards the packet but does not alert the source. The default setting is `false`.

Argument name	Description
`replyToMasks`	**Optional.** Sets the system's ability to reply to subnet mask lookup requests. If set to `true`, when the system receives a lookup request, it compares the source address to the routing table, and if a subnet is returned, sends that information back in a reply. If set to `false`, the system ignores the request. The default setting is `true`.

Example

The following example configures the system to send to source systems a report of unsupported IP options.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# icmp
sun(config-vSwitch-e-commerce vRouter-default icmp)# sendParamProbs
true
```

Associated MIB

`icmp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → icmp → modify

irdp addresses

Purpose

Sets the system to advertise the specified address(es) as a default gateway. An IRDP client receives the address in IRDP router advertisement updates, and with the preference associated with the address, forwards accordingly.

Access mode

config

Syntax

To create or modify an address:

```
vSwitch-name vRouter-name irdp addresses
   address ipAddress
   [advertise {enabled | disabled}]
   [advertisementAddress {multicast | broadcast}]
   [preference integer]
```

Arguments

Argument name	Description
`address` *ipAddress*	Specifies an IP address to be used as a default gateway for IRDP clients. Enter a valid IP address. You can enter as many addresses as you'd like, but must enter each individually. Specify the preference for the address with the `preference` argument, below.
`advertise {enabled | disabled}`	**Optional.** Specifies whether to include this address in router advertisement updates. When enabled, the system includes the specified address in advertisements. Use `disable` to temporarily disable an address as a default gateway. The default: is `enabled`.

Argument name	Description
`advertisementAddress {multicast | broadcast}`	**Optional.** Specifies which type of router advertisement update you want this address included in.When set to `multicast`, this address is part of the router advertisement that is sent to the all-systems multicast address of 224.0.0.1. When set to `broadcast`, this address is part of the router advertisement that is sent to the all-systems multicast address of 224.0.0.1. The default setting is `multicast`.
`preference` *`integer`*	**Optional.** Specifies a preference for this address as the default gateway. The higher the value, the greater the preference. Valid range is 0 through 2147483647; the default preference is 0.

Delete syntax

See the “Arguments” section for this command for argument descriptions.

```
no vSwitch-name vRouter-name irdp addresses
   address ipAddress
   [advertise {enabled | disabled}]
   [advertisementAddress {multicast | broadcast}]
   [preference integer]
```

Example

The following example sets address 10.10.10.1 to be used as a default gateway for IRDP clients.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# irdp
sun(config-vSwitch-e-commerce vRouter-default irdp)# addresses
10.10.10.1 preference 15
```

Associated MIB

`irdpTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Irdp → addresses → add
- vSwitch → *name* → vRouter → *name* → Irdp → addresses → copy
- vSwitch → *name* → vRouter → *name* → Irdp → addresses → modify
- vSwitch → *name* → vRouter → *name* → Irdp → addresses → delete

irdp interfaces

Purpose

Configures and activates IRDP on the specified interface. You can configure up to 4000 IRDP interfaces per virtual router. By default, IRDP is enabled for all interfaces. However, as soon as you specify an interface, IRDP becomes active only for specified interfaces.

Access mode

config

Syntax

To create or modify an interface:

```
vSwitch-name vRouter-name irdp interfaces
   ifName ifName
   [maxAdvInterval seconds]
   [minAdvInterval seconds]
   [lifeTime seconds]
```

Arguments

Argument name	Description
`ifName` *`ifName`*	Specifies the interface on which you want to activate IRDP. The interface can be either ip.*x*, `vlan.`*x*, `lag.`*x*, or `eth.`*x*`.`*x*. By using a wildcard, you can set all interfaces matching the entry with a single command. For example, you can set all IP interfaces by specifying `ip.*`.
`maxAdvInterval` *`seconds`*	**Optional.** Specifies the maximum number of seconds between unsolicited router advertisement updates originating from this interface. Valid range is 4 through 1800; the default maximum interval is 600 seconds.
`minAdvInterval` *`seconds`*	**Optional.** Specifies the minimum number of seconds between unsolicited router advertisement updates originating from this interface. Valid range is 3 through 1800; the default minimum interval is three-quarters of the `maxAdvInterval` (that is, `maxAdvInterval * .75`).

Argument name	Description
`lifeTime` *`seconds`*	**Optional.** Specifies the number of seconds for which addresses in a router advertisement update are valid. Valid range is 4 through 9000; the default lifetime is three times the `maxAdvInterval` (that is, `maxAdvInterval * 3`).

Delete syntax

See the "Arguments" section for this command for argument descriptions.

```
no vSwitch-name vRouter-name irdp interfaces
   ifName ifName
   [maxAdvInterval seconds]
   [minAdvInterval seconds]
   [lifeTime seconds]
```

Example

The following example activates IRDP on interface eth.1.20.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# irdp
sun(config-vSwitch-e-commerce vRouter-default irdp)# interfaces
eth.1.20
```

Associated MIB

`irdpTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Irdp → interfaces → add
- vSwitch → *name* → vRouter → *name* → Irdp → interfaces → copy
- vSwitch → *name* → vRouter → *name* → Irdp → interfaces → modify
- vSwitch → *name* → vRouter → *name* → Irdp → interfaces → delete

show arp

Purpose

Displays all the system's ARP table, which displays both static and dynamic ARP entries. To display only the manually configured entries, use the `show arp static` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip arp`

Sample output

```
sun> vSwitch system
sun(vSwitch-system)# vRouter management
sun(vSwitch-system vRouter-management)# ip
sun(vSwitch-system vRouter-management ip)# show arp
IfName                IP Address           Phys Address         type
ethMgmt.1             192.168.125.17       00:e0:2b:5f:ca:00    dynamic
ethMgmt.1             192.168.125.18       00:01:30:b5:26:80    dynamic
ethMgmt.1             192.168.125.31       00:80:ba:60:28:d6    dynamic
ethMgmt.1             192.168.125.32       00:80:ba:a0:16:e9    dynamic
ethMgmt.1             192.168.125.64       00:07:82:00:0d:40    dynamic
ethMgmt.1             192.168.125.66       00:07:82:00:01:80    dynamic
ethMgmt.1             192.168.125.69       00:07:82:00:05:80    dynamic
ethMgmt.1             192.168.125.72       00:07:82:00:0c:c0    dynamic
ethMgmt.1             192.168.125.202      00:07:82:aa:00:08    dynamic
ethMgmt.1             192.168.125.203      00:07:82:00:0c:40    dynamic
ethMgmt.1             192.168.125.204      00:07:82:00:0d:40    dynamic
ethMgmt.1             192.168.125.205      00:07:82:00:0d:80    dynamic
ethMgmt.1             192.168.125.206      00:07:82:00:01:80    dynamic
ethMgmt.1             192.168.125.207      00:07:82:00:04:80    dynamic
ethMgmt.1             192.168.125.209      00:07:82:00:05:80    dynamic
ethMgmt.1             192.168.125.210      00:07:82:00:05:00    dynamic
ethMgmt.1             192.168.125.211      00:07:82:00:02:40    dynamic
```

Output description

Field Name	Description	Filter Name
`ifName`	The name of the interface associated with the ARP entry. This is the interface that connects to the IP instance, either `vlan.x`, `lag.x`, or `eth.x.x`. Use the `show ip interface` command to view all configured IP-to-lower layer interface associations. Use the `ip interface` command to configure new associations.	`ifName` *`ifName`*
`IP Address`	The IP address the system associates with the physical address specified. This pairing is stored in the system's ARP table.	`netAddr` *`ipAddress`*
`Phys Address`	The Layer 2 physical address (the MAC address) associated with the specified IP address/interface in the system's ARP table.	`physAddr` *`macAddress`*
`Type`	A designation of the ARP entry type, one of the following: `other`: `invalid`: `dynamic`: An entry learned by the system broadcasting unknown addresses to local hosts and storing responses. `static`: A manually configured entry, added to the table with the `arp static` command. `application`: An entry added by an internal application (still needing resolution, however). Examples include server hosts and nextHop routers	`type {other | invalid | dynamic | static | application}`

Associated MIB

`arp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp

show arp settings

Purpose

Displays the ARP configuration parameters set with the `arp settings` command. This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip arp settings`

Sample output

```
sun> vSwitch system
sun(vSwitch-system)# vRouter management
sun(vSwitch-system vRouter-management)# ip
sun(vSwitch-system vRouter-management ip)# show arp settings
ARP Entry Aging Time:    300
ARP Request Retry Count: 2
ARP Request Timeout:     2
Duplicate MAC check:     enabled
```

Output description

Field Name	Description
`ARP Entry Aging Time`	The amount of time, in seconds, that an ARP entry remains valid and in the ARP table. When the timer has expired, the system deletes the entry from the table. The valid range is 0 through 86400 seconds. The default timer is 300 seconds; a value of 0 disables the time out feature, and entries never age out.
`ARP Request Retry Count`	The number of times the system will rebroadcast an unknown IP address to local hosts looking for a Layer 2 mapping. The valid range is 0 through 100 retries; the default value is 2. A value of 0 means no retries are attempted
`ARP Request Timeout`	The number of seconds the system waits between sending out ARP requests. The valid range is 1 through 20 seconds; the default value is 2 seconds.
`Duplicate MAC check`	Whether the system checks for duplicate MAC addresses when learning ARP entries. The default is `enabled`.

Associated MIB

`arp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp → settings

show arp static

Purpose

Displays the manually configured (static) entries in the system's ARP table. To display both dynamic and static entries, use the `show arp` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip arp static`

Sample output

```
sun> vSwitch system
sun(vSwitch-system)# vRouter management
sun(vSwitch-system vRouter-management)# ip
sun(vSwitch-system vRouter-management ip)# show arp static
IfName              IP Address          Phys Address
eth.1.14            10.10.36.4          00:07:82:00:05:40
```

Output description

Field Name	Description	Filter Name
`IfName`	The name of the interface associated with the ARP entry. This is the interface that connects to the IP instance, either `vlan.`*`x`*, `lag.`*`x`*, or `eth.`*`x.x`*. Use the `show ip interface` command to view all configured IP-to-lower layer interface associations. Use the `ip interface` command to configure new associations.	`IfName` *`ifName`*
`IP Address`	The IP address portion of the ARP table mapping. The system associates this address with the physical address displayed in the following field.	`netAddr` *`ipAddress`*

Field Name	Description	Filter Name
`Phys Address`	The Layer 2 physical address (the MAC address) portion of the ARP table mapping. The system associates this address with the IP address and interface displayed.	`physAddr` *`macAddress`*

Associated MIB

`arp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp → static

show arp statistics

Purpose

Displays ARP-specific statistics for all or a specified interface. These statistics provide information about the ARP cache, as well as transmitted and received ARP traffic.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip arp statistics`

Sample output

```
sun> vSwitch system
sun(vSwitch-system)# vRouter management
sun(vSwitch-system vRouter-management)# ip
sun(vSwitch-system vRouter-management ip)# show arp statistics
IfName:           vlan.100
Entries:          0
Entries Aged Out: 0
Out Requests:     0
In Total:         0
In Requests:      0
In Replies:       0
In Discards:      0
In Learned:       0
Que Overflow:     0

IfName:           vlan.110
Entries:          0
Entries Aged Out: 0
Out Requests:     0
In Total:         0
In Requests:      0
In Replies:       0
In Discards:      0
In Learned:       0
Que Overflow:     0

IfName:           vlan.140
Entries:          0
Entries Aged Out: 0
```

```
Out Requests:     0
In Total:         0
In Requests:      0
In Replies:       0
In Discards:      0
In Learned:       0
Que Overflow:     0
```

Output description

Field Name	Description	Filter Name
IfName	The name of the interface to which the ARP statistics apply. This is the interface that connects to the IP instance, either `vlan.x`, `lag.x`, or `eth.x.x`. Use the `show ip interface` command to view all configured IP-to-lower layer interface associations. Use the `ip interface` command to configure new associations.	ifName *ifName*
Entries	The number of entries in the ARP cache on the interface.	entries *integer*
Entries Aged Out	The number of entries aged out of the ARP cache on the interface.	agedOut *integer*
Out Requests	The number of ARP requests transmitted over the interface. These are the broadcasts to local hosts requesting IP address-to-physical address mapping information.	outRequests *integer*
In Total	The total number of ARP packets received on the interface. This count includes requests for and replies to IP address-to-physical address mapping information.	inTotal *integer*
In Requests	The number of ARP requests received on the interface. These are the broadcasts from local hosts requesting IP address-to-physical address mapping information.	inRequests *integer*
In Replies	The number of ARP replies received over the interface. These are the responses to this interface's broadcast requesting IP address-to-physical address mapping information.	inReplies *integer*
In Discards	The number of ARP packets received that are not of a known type. That is, they are not ARP requests or ARP responses.	inDiscards *integer*

Field Name	Description	Filter Name
`In Learned`	The number of ARP entries that were learned from requests packets that were not destined for this interface. That is, the protocol extracts the IP and physical addresses of the packet's source.	`inLearned` *`integer`*
`Que Overflow`	The number of packets dropped due to a queue overflow. These are IP packets queued for transmission while awaiting an ARP response.	`queOverflow` *`integer`*

Associated MIB

`arp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → arp → statistics

show icmp

Purpose

Displays the configuration settings for the Internet Control Message Protocol (ICMP). A TCP/IP-based protocol, ICMP verifies, through error and control messages, the validity of an IP address.

This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip icmp`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# ip
sun(vSwitch-e-commerce vRouter-default ip)# show icmp
Reply To Echos:          true
Send Dest Unreachables:  false
Send Time Exceededs:     true
Send Parameter Problems: false
Reply To Mask Requests:  true
In Messages:             0
In Errors:               0
Out Messages:            0
Out Errors:              0
```

Output description

Field Name	Description
`Reply To Echos`	The setting of the system's ability to reply to ICMP echo requests. If set to `true`, the system responds to these requests (typically `ping` packets); if set to `false` the packet is dropped. The default setting is `true`.

Field Name	Description
`Send Dest Unreachables`	The setting of the system's ability to send ICMP destination unreachable errors when it does not have the necessary information to forward a packet. Requiring the system to reply with these error messages could expose you to malicious network flooding. If set to `true`, the system responds with a destination unreachable message; if set to `false` the packet is dropped. The default setting is `false`.
`Send Time Exceededs`	The setting of the system's ability to send an ICMP TTL exceeded error message. If set to `true`, the system sends this message to the error log when the time-to-live timer has expired in the IP header. If set to `false`, the packet is dropped. The default setting is `true`.
`Send Parameter Problems`	The setting of the system's ability to send an ICMP Parameter Problem error message. If set to `true`, the system forwards the packet and responds to the source with this message if the IP header contains options that are not supported on the N2000 Series system. If set to `false`, the system forwards the packet but does not alert the source. The default setting is `false`.
`Reply To Mask Requests`	The setting of the system's ability to reply to subnet mask lookup requests. If set to `true`, when the system receives a lookup request, it compares the source address to the routing table, and if a subnet is returned, sends that information back in a reply. If set to `false`, the system ignores the request. The default setting is `true`.
`In Messages`	The total number of ICMP messages received over this interface. This total includes messages that had errors (and are therefore included in the `In Errors` count).
`In Errors`	The number of ICMP messages received that had ICMP-specific errors. These include bad length, bad icmp checksum, or unsupported ICMP type.
`Out Messages`	The total number of ICMP messages transmitted. This total includes messages that had errors (and are therefore included in the `Out Errors` count).
`Out Errors`	The number of ICMP messages transmitted that had ICMP-specific errors. These include no route to host and ARP not resolved.

Associated MIB

`icmp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → icmp

show icmp inStats

Purpose

Displays statistics for ICMP messages received on this instance of IP. This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip icmp inStats`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# ip
sun(vSwitch-e-commerce vRouter-default ip)# show icmp inStats
In Messages:              0
In Errors:                0
In Dest Unreachables:     0
In Time Exceededs:        0
In Param Problems:        0
In Source Quenches:       0
In Redirects:             0
In Echo Requests:         0
In Echo Replies:          0
In Timestamp Requests:    0
In Timestamp Replies:     0
In Address Mask Requests: 0
In Address Mask Replies:  0
In Info Requests:         0
In Info Replies:          0
In Router Advertisements: 0
In Router Solicits:       0
```

Output description

Field Name	Description
`In Messages`	The total number of ICMP messages received. This total includes messages that had errors (and are therefore included in the `In Errors` count).
`In Errors`	The number of ICMP messages received that had ICMP-specific errors.
`In Dest Unreachables`	The number of ICMP destination unreachable messages received since last boot (type field 3). These are messages sent by a router when it cannot forward an ICMP packet.
`In Time Exceededs`	The number of ICMP time exceeded messages received since last boot (type field 11). These messages, sent to a packet's source, indicate that the router dropped the packet due to an exceeded hop count or timer.
`In Param Problems`	The number of ICMP parameter problem messages received since last boot (type field 12). These messages indicate some sort of problem not covered by a specific type field.
`In Source Quenches`	The number of ICMP source quench messages received since last boot (type field 4). These messages report congestion on an interface to the source.
`In Redirects`	The number of ICMP redirect messages received since last boot (type field 5). These messages report to a host that it should change its stored route to a destination.
`In Echo Requests`	The number of ICMP echo request messages received since last boot (type field 8). These are the queries sent out that, when responded to with an echo reply, verify connectivity
`In Echo Replies`	The number of ICMP echo reply messages received since last boot (type field 0). These replies are sent to echo requests, verifying connectivity.
`In Timestamp Requests`	The number of ICMP timestamp request messages received since last boot (type field 13). These messages request a current value for the time of day from the recipient.
`In Timestamp Replies`	The number of ICMP timestamp reply messages received since last boot (type field 14). These messages respond to `time stamp requests`, reporting the current time of day.
`In Address Mask Requests`	The number of ICMP address mask request messages received since last boot (type field 17). These messages are queries for the subnet mask of the local network.

Field Name	Description
`In Address Mask Replies`	The number of ICMP address mask reply messages received since last boot (type field 18). These messages are responses to requests for the subnet mask of the local network.
`In Info Requests`	Obsolete.
`In Info Replies`	Obsolete.
`In Router Advertisements`	The number of ICMP router advertisements received since last boot. These messages come periodically from the multicast interfaces of an attached router, and contain the IP address(es) on interface.
`In Router Solicits`	The number of ICMP router solicitations received since last boot. These messages come from hosts that want an immediate routing update instead of waiting for the periodic `router advertisement`.

Associated MIB

`icmp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → icmp → inStats

show icmp outStats

Purpose

Displays statistics for ICMP messages transmitted over this instance of IP. This command does not support field filtering.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `ip icmp outStats`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# ip
sun(vSwitch-e-commerce vRouter-default ip)# show icmp outStats
Out Messages:              0
Out Errors:                0
Out Time Exceededs:        0
Out Param Problems:        0
Out Source Quenches:       0
Out Redirects:             0
Out Echo Requests:         0
Out Echo Replies:          0
Out Timestamp Requests:    0
Out Timestamp Replies:     0
Out Address Mask Requests: 0
Out Address Mask Replies:  0
Out Router Advertisements: 0
Out Router Solicits:       0
```

Output description

Field Name	Description
`Out Messages`	The number of ICMP destination unreachable messages transmitted since last boot (type field 3). These are messages sent by a router when it cannot forward an ICMP packet.
`Out Errors`	The number of ICMP time exceeded messages transmitted since last boot (type field 11). These messages, sent to a packet's source, indicate that the router dropped the packet due to an exceeded hop count or timer.
`Out Time Exceededs`	The number of ICMP source quench messages transmitted since last boot (type field 4). These messages report congestion on an interface to the source.
`Out Param Problems`	The number of ICMP redirect messages transmitted since last boot (type field 5). These messages report to a host that it should change its stored route to a destination.
`Out Source Quenches`	The number of ICMP echo request messages transmitted since last boot (type field 8). These are the queries sent out that, when responded to with an echo reply, verify connectivity
`Out Redirects`	The number of ICMP echo reply messages transmitted since last boot (type field 0). These replies are sent to echo requests, verifying connectivity.
`Out Echo Requests`	The number of ICMP timestamp request messages transmitted since last boot (type field 13). These messages request a current value for the time of day from the recipient.
`Out Echo Replies`	The number of ICMP timestamp reply messages transmitted since last boot (type field 14). THese messages respond to `time stamp requests`, reporting the current time of day.
`Out Timestamp Requests`	The number of ICMP address mask request messages transmitted since last boot (type field 17). These messages are queries for the subnet mask of the local network.
`Out Timestamp Replies`	The number of ICMP address mask reply messages transmitted since last boot (type field 18). These messages are responses to requests for the subnet mask of the local network.
`Out Address Mask Requests`	Obsolete.
`Out Address Mask Replies`	Obsolete.

Field Name	Description
`Out Router Advertisements`	The number of ICMP router advertisement messages transmitted since last boot (type field 3). These are messages sent by a router when it cannot forward an ICMP packet.
`Out Router Solicits`	The number of ICMP router solicit messages transmitted since last boot. (type field 11). These messages, sent to a packet's source, indicate that the router dropped the packet due to an exceeded hop count or timer.

Associated MIB

`icmp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → icmp → outStats

show irdp addresses

Purpose

Displays each address configured as a default gateway for IRDP clients. Each display reports configuration specifics, including the address's preference. Clients attempt to use addresses in order of preference, from highest to lowest.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `irdp addresses`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# show irdp addresses
Address:               1.1.1.1
Advertise:             enabled
Advertisement Address: multicast
Preference:            0
```

Output description

Field Name	Description	Filter Name
`Address`	The IP address being advertised to the client as a default gateway. The value in the `Preference` field, below, indicates the order in which a client attempts to use the address.	`address` *`ipAddress`*
`Advertise`	The inclusion of this address in router advertisement updates. When enabled, the system includes the specified address in advertisements. Use `disable` to temporarily disable an address as a default gateway. The default: is `enabled`.	`advertise {enabled | disabled}`

Field Name	Description	Filter Name
`Advertisement Address`	The type of router advertisement update that the system includes this address in. When set to `multicast`, router advertisements are sent to the all-systems multicast address of 224.0.0.1. When set to `broadcast`, router advertisements are sent to the all-systems multicast address of 224.0.0.1The default setting is `multicast`.	`advertisementAddress {multicast | broadcast}`
`Preference`	The preference for an address as the default gateway. The higher the value, the greater the preference. Valid range is 0 through 2147483647; the default preference is 0.	`preference` *`integer`*

Associated MIB

`irdpTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Irdp → addresses

Show irdp interfaces

Purpose

Displays each interface that is configured to run IRDP, as well as the configuration specifics.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `irdp interfaces`

Sample output

```
sun> vSwitch e-commerce
sun(vSwitch-e-commerce)# vRouter default
sun(vSwitch-e-commerce vRouter-default)# show irdp interfaces
IfName:                  eth.1.20
Max Advertised Interval: 600
Min Advertised Interval: 450
Lifetime:                1800

IfName:                  eth.1.27
Max Advertised Interval: 600
Min Advertised Interval: 450
Lifetime:                1800
```

Output description

Field Name	Description	Filter Name
IfName	The interface on which IRDP is active. The interface can be either ip.*x*, vlan.*x*, lag.*x*, or eth.*x*.*x*. By using a wildcard, you can view all interfaces matching the entry with a single command. For example, you can view all IP interfaces by specifying ip.*.	ifName *ifName*

Field Name	Description	Filter Name
`Max Advertised Interval`	The maximum number of seconds between unsolicited router advertisement updates originating from this interface. Valid range is 4 through 1800; the default maximum interval is 600 seconds.	`maxAdvInterval` *`seconds`*
`Min Advertised Interval`	The minimum number of seconds between unsolicited router advertisement updates originating from this interface. Valid range is 3 through 1800; the default minimum interval is three-quarters of the `maxAdvInterval` (that is, `maxAdvInterval * .75`).	`minAdvInterval` *`seconds`*
`Lifetime`	The number of seconds for which addresses in a router advertisement update are valid. Valid range is 4 through 9000; the default lifetime is three times the `maxAdvInterval` (that is, `maxAdvInterval * 3`).	`lifeTime` *`seconds`*

Associated MIB

`irdpTbl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Irdp → interfaces

Chapter 27. Ping and traceroute utility commands

IP utility description

This chapter describes the ping and traceroute utilities available from within a virtual router on the N2000 Series system. These utilities allow you to verify the existence and connectivity of a specific IP address (`ping`) and troubleshoot the connection by tracing the path of a packet from source to destination (`traceroute`). Used in combination, these utilities are a standard troubleshooting tool for IP networks. The system supports IP Version 4.

Ping and traceroute command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `vRouter` *`name`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Utility command summary

Table 27-1 lists and briefly describes the `ping` and `traceroute` commands.

Table 27-1. Ping and traceroute command summary

Command Name	Description
`ping`	Send a packet and listen for an answer.
`traceroute`	Trace the route of a packet from the source host to a remote destination.

ping

Purpose

Sends an ICMP echo request packet to the specified host and listens for an answer. The response includes information regarding the time it took the packet to traverse the network and reach the remote host device.

The N2000 Series supports one virtual IP router per vSwitch, so the `ping` command is sourced from the virtual router. The host that receives the query responds as if to a physical router, verifying its connectivity to the virtual router. Each virtual router configured within the N2000 Series can query and respond to the `ping` command.

Access mode

user

Syntax

```
vSwitch-name vRouter-name ping
   host ipAddress
   [count integer]
   [size bytes]
   [incrSize integer]
   [timeout integer]
   [ttl integer]
   [tos integer]
   [dontFrag {true | false}]
   [validate {true | false}]
   [pattern hexString]
   [srcAddr ipAddress]
```

Arguments

Argument name	Description
`host` *ipAddress*	Specifies the IP address of the host whose connectivity you want to verify.
`count` *integer*	**Optional.** Specifies the number of ping requests (packets) to send to the remote device. The valid range in the CLI is 1 through 65535; default number of packets is 5. The valid range in the Web Manager is 1 through 5; default number of packets is 1.
`size` *bytes*	**Optional.** Sets the number of bytes in the ping request packet.
`incrSize` *integer*	**Optional.** Sets the number of bytes by which to increment the ping packet size with each request. Valid values are 0, 32, 64, 128, 256, 512, and 1024; default incrementation is 0 bytes.
`timeout` *integer*	**Optional.** Sets the maximum number of seconds the command can run. The command quits if this timer expires, regardless of the status of the packets in transit. The valid values for the CLI are 100 through 20000 milliseconds; default timeout value is 2000 milliseconds. The valid values for the Web Manager are 100 through 5000 milliseconds; default timeout value is 2000 milliseconds. Setting the timeout to 0 disables the command.
`ttl` *integer*	**Optional.** Sets the Time To Live (TTL) value, which is the maximum number of hops the packet can traverse before being discarded. This argument only applies if the destination is a multicast address. Valid values are 1 through 255; default TTL is 64 hops.

Argument name	Description
`tos` *`integer`*	**Optional.** Specifies the TOS (Differentiated Services Field) value in the IP header of the ICMP request. All IP datagrams have a TOS field, which allows the sender to get different types of service. For example, a router may have a separate queue for each TOS and may handle those queues according to the type of service desired. Valid values are 0 through 255; default TOS value is 0.
`dontFrag {true | false}`	**Optional.** Specifies whether to set the IP Don't Fragment bit. This bit in the IP header indicates to routers along the way whether it is acceptable to fragment this IP packet. If set to `true`, the system can fragment the datagram. If set to `false`, the system does not fragment the packet. The default setting is `false`.
`validate {true | false}`	**Optional.** Specifies that the command verify whether the data in the outgoing packet is the same as the data in the reply packet. If set to `true`, the system validates responses. If set to `false`, the system does not validate responses. The default setting is `false`.
`pattern` *`hexString`*	**Optional.** Sets the data in the ping request to the specified data pattern. Enter a hexadecimal value.
`srcAddr` *`ipAddress`*	**Optional.** Sets the ping request packet to include its source address. This modifies the source address in the IP header, allowing you to specify the IP address to use in the reply.

Example

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ping 10.10.36.3
Ping reply #1 from 10.10.36.3 ipBytes=84 time=4000us TTL=62
Ping reply #2 from 10.10.36.3 ipBytes=84 time=0us TTL=62
Ping reply #3 from 10.10.36.3 ipBytes=84 time=0us TTL=62
Ping reply #4 from 10.10.36.3 ipBytes=84 time=4000us TTL=62
Ping reply #5 from 10.10.36.3 ipBytes=84 time=8000us TTL=62
Ping statistics for 10.10.36.3
     Packet: Sent = 5, Received = 5, Lost = 0 (0% loss)
     Approximate round trip times in microseconds:

     Minimum = 4000us, Maximum = 8000us, Average = 5250us
```

Associated MIB

```
ip.mib
```

Web path

vSwitch → *name* → vRouter → *name* → ipUtils → ping

traceroute

Purpose

Traces the route of a packet from the source host to a remote destination. A standard IP debugging utility, `traceroute` reports each gateway it encounters—the actual route a packet takes—on its path from source to destination. This process continues until the probe reaches the host, or if other errors are encountered.

The `traceroute` command is executed from within the vRouter command mode. The N2000 Series supports one virtual IP router per vSwitch, so the `traceroute` command is sourced from the virtual router. It can trace a path to either another virtual router within the same chassis, a virtual router in another system, or a physical router. The host that receives the query responds as if to a physical router, and the resulting path is from there to the virtual router. Each virtual router configured within the N2000 Series can query and respond to the `traceroute` command. You can send multiple `traceroute` requests to the same destination.

Access mode

user

Syntax

```
vSwitch-name vRouter-name traceroute
   host ipAddress
   [srcAddr ipAddress]
   [timeout integer]
   [count integer]
   [minTTL integer]
   [maxTTL integer]
   [port integer]
   [size bytes]
```

Arguments

Argument name	Description
`host` *`ipAddress`*	Specifies the IP address of the host at the far end of the path.
`srcAddr` *`ipAddress`*	**Optional.** Specifies the source address to use in an outgoing probe. This option is used for devices with multiple IP addresses assigned. It allows you to specify that probes return to an interface other than the interface from which they were sent. The address must be a valid IP address for the device.
`timeout` *`integer`*	**Optional.** Sets the time, in milliseconds, that the command waits for a response to a probe. The valid range for the CLI is 0 through 4294967295 milliseconds (50 days); default timeout value is 3000 milliseconds. The valid range for the Web Manager is 100 through 5000; default timeout value is 2000 milliseconds.
`count` *`integer`*	**Optional.** Sets the number of probe packets to send. Valid values are 1 through 10; default is 1 probes.
`minTTL` *`integer`*	**Optional.** Sets the starting TTL value (time to live) for the probe. The default TTL is 1.
`maxTTL` *`integer`*	**Optional.** Sets the maximum TLL for the probe. The value must be greater than the minimum TTL. The default is 64.
`port` *`integer`*	**Optional.** Sets the UDP destination port number. This port number should be invalid, which causes ICMP to return a port unreachable message, which terminates the command. Valid values are 1 through 65535; default UDP port is a TTL equal to 33434.
`size` *`bytes`*	**Optional.** Specifies the size, in bytes, of the UDP probes (not icluding the IP or UDP headers). Because on some systems (such as many win32 machines) IP packets with a total length that is a mutliple of 256 return invalid ICMP error messages, the following sizes are not recommended: 228, 484, 740, 996, 1252. Valid values are 0 to 1472; the default setting is 40.

Example

```
sun(vSwitch-system vRouter-management)# traceroute 10.10.209.76
 hopping 10.10.10.253 on hop #1   sequence #1  in 11 ms
 hopping 10.10.10.18 on hop #2   sequence #1  in 11 ms
 hopping 10.10.10.4 on hop #3   sequence #1  in 14 ms
 reached 10.10.209.76 on hop #4   sequence #1  in 12 ms
 TraceRoute to 10.10.209.76 Finished
```

Associated MIB

`ip.mib`

Web path

vSwitch → *name* → vRouter → *name* → ipUtils → traceroute

Chapter 28. ACL commands

IP and ACLs description

This chapter describes the IP-based access control list (ACL) commands available from the N2000 Series system. The system supports IP Version 4. The N2000 Series supports several other IP-related command functions, which are detailed in the following chapters.

IP feature	Chapter
IP interfaces and addressing	Chapter 22.
Routing Information Protocol (RIP) and static routing	Chapter 24.
• Address Resolution Protocol (ARP) • Internet Control Message Protocol (ICMP) • Internet Domain Routing Protocol (IRDP)	Chapter 26.
Ping and traceroute utilities	Chapter 27.

ACL description

An access control list is a set of rules permitting or denying access based on source and destination IP addresses or subnets, protocol, and application ports. You create a list using the `accessList` command and add match statements using one of the `accessList rule` commands. When you specify a list name for either command, if the named list does not exist, the system creates it.

Note: Although the `accessList rule` command is technically a single command with parameters that vary depending on the arguments supplied (a type-based view), for clarity it is presented as multiple commands here, based on the protocol type.

Setting rule precedence

When you create a rule for an ACL, you associate a “precedence” with the rule using the `ruleIndex` argument. When more than one rule applies to an individual packet, the rule with the lowest `ruleIndex` is selected.

It is a good idea to position more specific rules with a lower `ruleIndex` value, otherwise they could be ineffective. For example, if rule #1 denies all ICMP traffic, and rule #2 permits ICMP echo requests, echo requests will be dropped because rule #1, with higher precedence, denied ICMP.

Every access list ends with an implicit “deny all” rule. In the case that traffic does not match any of the configured rules, it is handled by the implicit deny, and the traffic is dropped. This rule cannot be deleted or modified.

Working with ACLs

Once an ACL is created, it is applied to one or more IP interfaces with the `accessGroup` command. Every IP interface can support one inbound and one outbound ACL and a single ACL can be applied to many interfaces. Each vRouter can support up to four lists, and each list can support up to 256 rules. Rules within an access list can be added, deleted, modified, enabled, and disabled. Access control lists are enabled by default.

Entering source and destination addresses

Both source and destination addresses are optional filters, and each can be entered in any of the forms below. That is, you can use either the same or different formats. Use the following formats to enter the `ruleSrcAddrs` or `ruleDstAddrs` arguments to define match criteria for permitting or denying traffic.

Format	Description	Example
`any`	Permits or denies all addresses (default).	`any`
`hostAddress`	Specifies a single host address to permit or deny.	`10.10.10.3`
`loAddress-hiAddress`	Specifies an address range to permit or deny.	`10.10.10.1-10.10.10.255`
`address/mask`	Specifies a subnet to permit or deny.	`10.10.10.3/ 255.255.255.0`
`address/hostMask`	Subnet to specify which addresses to permit or deny (reverses the mask bits of the *`address/mask`* format).	`10.10.10.3/0.0.0.255`
`address/netBits`	Uses a CIDR style subnet to specify which addresses to permit or deny.	`192.168.1.0/24`

ACL protocol types

The following protocols can be specified as a match for filtering. Any of these values can be used with the `ruleProto` argument to define match criteria for permitting or denying traffic. For well-known protocols, you can supply a name or integer (names are listed in the table below). For less common protocols, you can use the protocol number, as defined by the Internet assigned numbers authority http://www.IANA.org/numbers.html

Table 28-1. Well-known protocols for filter matching

Protocol	Definition
integer (1-255)	Represents an IP protocol number
ah	Authentication Header (RFC 2402)
any	Matches any protocol
comp	IP Compression (RFC 3173)
egp	External Gateway Protocol (RFC 827)
esp	Encapsulating Security Payload (RFC 2406)
gre	Generic Routing Encapsulation (RFC 2784)
icmp	Internet Control Message Protocol (RFC 2463)
idrp	Inter-Domain Routing Protocol (RFC 1745)
igmp	Internet Group Management Protocol (RFC 3228)
igrp	Interior Gateway Routing Protocol (RFC 2072)
isis	Intermediate System-to-Intermediate System (RFC 1142)
ospf	Open Shortest Path First (RFC 2740)
rsvp	Resource Reservation Protocol (RFC 2205)
tcp	Transmission Control Protocol (RFC 1213)
udp	User Datagram Protocol (RFC 1213)
vrrp	Virtual Router Redundancy Protocol (RFC 2338)

ACL command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `vRouter` *`name`* `ip`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

ACL command summary

Table 28-2 lists and briefly describes the access control list commands.

Table 28-2. ACL command summary

Command name	Description
`accessGroup`	Apply an access list to an IP interface.
`accessList`	Create or modify an access list.
`accessList rule (for generic protocol)`	Assign rules to an access list based on a protocol type other than ICMP, TCP, or UDP.
`accessList rule (for ICMP)`	Assign rules to an access list when the protocol type is ICMP.
`accessList rule (for TCP)`	Assign rules to an access list when the protocol type is TCP.
`accessList rule (for UDP)`	Assign rules to an access list when the protocol type is UDP.
`show accessGroup`	Display each interface and its associated access list.
`show accessGroup status`	Display access lists and hits per rule.
`show accessList`	Display the operational and administrative details of access control lists (brief).
`show accessList verbose`	Display the operational and administrative details of access control list (detailed).
`show accessList rule`	Display a brief one line summary of each named access list rule on the vRouter.

Table 28-2. ACL command summary (continued)

Command name	Description
`show accessList rule status`	Provide a summary display of access control list rules, including hit counts.
`show accessList rule verbose`	Display a detailed summary of each named access list rule on the vRouter.

ACL configuration

The following table explains the basic steps for configuring access control lists and applying them to an interface.

Table 28-3. Steps for configuring IP ACLs

Step	Action
1.	Create an empty list with the `accessList` command.
2.	Assign rules to the access list with the `accessList rule` command.
3.	Apply the access list to an IP interface with the `accessGroup` command.

accessGroup

Purpose

Applies an access list to an IP interface for inbound or outbound traffic. Every IP interface supports up to one inbound and one outbound ACL, and a single ACL can be applied to any number of interfaces.

The no form of the command removes the interface-to-ACL association.

Access mode

config

Syntax

To create an interface-to-ACL association:

```
vSwitch-name vRouter-name ip accessGroup
   ifName interfaceName
   direction {in | out}
   aclListName text
```

To modify an interface-to-ACL association:

```
vSwitch-name vRouter-name ip accessGroup
   ifName interfaceName
   direction {in | out}
   [aclListName text]
```

Arguments

Argument name	Description
`interface interfaceName`	Specifies the name of the interface to which the ACL is being applied (the interface that connects to the IP instance). This could be either `vlan.x`, `lag.x`, or `eth.x.x`. If the interface does not already exist, the system returns an error. Use the `show interface` command to verify configured IP-to-lower layer interface associations. Use the `accessGroup` command to configure new associations.

Argument name	Description
`direction {in | out}`	Assigns the named ACL to either the inbound or outbound interface.
`aclListName` *`text`*	Specifies the name of the access control list that will filter this interface's traffic. The name you specify is a list you created with the `accessList` command. If you specify a name of an ACL that does not yet exist, the command creates it. This argument is optional when deleting the interface-to-ACL association.

Example

The following example defines an access group that contains two access lists, one for inbound traffic and one for outbound traffic.

```
sun> enable
sun# config
sun(config)# vswitch e-commerce vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(...default)# ip accessGroup eth.1.10 direction in aclListName
ACL_1
sun(...default)# ip accessGroup eth.1.10 direction out aclListName
ACL_2
```

Delete filters

See the `show accessGroup status` command for argument descriptions.

```
no vSwitch-name vRouter-name ip accessGroup
   ifName interfaceName
   direction {in | out}
   [aclListName text]
   [operStatus {active | inactive | disabled}]
   [hits integer]
```

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessGroup → add
- vSwitch → *name* → vRouter → *name* → Ip → accessGroup → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessGroup → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessGroup → delete

accessList

Purpose

Creates, enables, disables, or deletes an access list. Also, optionally, allows you to associate a description with the list. If you a specify a list name and the list does not yet exist, the system creates it.

The no form of the command deletes the named list.

Access mode

config

Syntax

```
vSwitch-name vRouter-name ip accessList
   aclListName text
   [adminState {enabled | disabled}]
   [description text]
```

Arguments

Argument name	Description
`aclListName` *text*	Specifies the name of the access control list. This is the name you assigned when you created the list, or if the list doesn't yet exist, the system creates it.
`adminState {enabled | disabled}`	**Optional**. Sets the ACL's administrative mode, either `enabled` or `disabled`. By default, ACLs are enabled.
`description` *text*	**Optional**. Associates a text description with the named list. The description can be up to 64 characters, and is displayed with the `show accessList verbose` command. If the description contains multiple words separated by spaces, enclose it in quotation marks.

Example

The following example defines an access list named ACL1.

```
sun> enable
sun# config
sun(config)# vswitch e-commerce vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# accessList ACL1
adminState enabled description “main access list”
```

Delete filters

See the `show accessList` command for argument descriptions.

```
no vSwitch-name vRouter-name ip accessList
   aclListName text
   [adminState {enabled | disabled}]
   [description text]
   [operStatus {active | inactive | disabled | pending}]
   [ipAclIfs interfaceName]
```

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → delete
- vSwitch → *name* → vRouter → *name* → Ip → accessList → verbose → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → verbose → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → verbose → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → verbose → delete

accessList rule (for generic protocol)

Purpose

Assigns rules to an access list based on a protocol type other than ICMP, TCP, or UDP. If you create access rules without first having created an access list, an access list with the name you specify is automatically created. This format also supports matches on the first two bytes of the packet payload.

Each argument of this command identifies criteria that a packet must meet for a match to occur. Optional arguments have a default setting that is also part of the match criteria. A packet must meet all criteria to be considered a match. When a match is established, the system implements the configured action of `permit` or `deny`.

The `no` form of the command deletes the specified rule.

Access mode

config

Syntax

To create a generic rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   ruleAction {deny | permit}
   ruleProto protocol
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [rulePayloadByte1Value hexUInt]
   [rulePayloadByte1Mask hexUInt]
   [rulePayloadByte2Value hexUInt]
   [rulePayloadByte2Mask hexUInt]
   [ruleDescr text]
```

To modify a generic rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
```

```
[ruleProto protocol]
[ruleSrcAddrs text]
[ruleDstAddrs text]
[rulePayloadByte1Value hexUInt]
[rulePayloadByte1Mask hexUInt]
[rulePayloadByte2Value hexUInt]
[rulePayloadByte2Mask hexUInt]
[ruleDescr text]
```

Arguments

Argument name	Description
`aclListName` *text*	Specifies the name of the access control list to which you are adding this rule. This is the name you assigned when you created the list, or if the list doesn't yet exist, the system creates it.
`ruleIndex` *index*	Assigns a precedence to the rule, which sets the order in which the rule will be evaluated. The lower the number, the earlier it is evaluated. Valid rule numbers are 1 through 4095.
`ruleAction {deny | permit}`	Specifies the action to take on packets meeting the rule criteria. If the rule has an action of `permit`, and the packet matches, it is forwarded. If the action is `deny`, the packet is rejected. There is no default, as this is a required argument. However, the default action for a packet not matching any rule is `deny`. This argument is required when creating an ACL rule, but optional when modifying or deleting it.
`ruleProto` *protocol*	Identifies the IP protocol that must be matched to meet filtering conditions. See "ACL protocol types" for a list of protocols you can select as matching criteria.
`ruleSrcAddrs` *text*	**Optional.** Specifies the host or network from which the packet originated. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all source addresses.
`ruleDstAddrs` *text*	**Optional.** Specifies the host or network for which the packet is destined. See ""Entering source and destination addresses"" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all destinations.
`rulePayloadByte1Value` *hexValue*	**Optional.** Specifies as a filter the first byte of the IP data field. Enter a hex value to match the first byte. The default value is 0.

Argument name	Description
`rulePayloadByte1Mask` *`hexUInt`*	**Optional.** Specifies a mask for the byte 1 value. The default is 0, which masks nothing (matches anything). The default is 0xff if a non-zero value is specified for value.
`rulePayloadByte2Value` *`hexUInt`*	**Optional.** Specifies as a filter the second byte only of the IP data field. Enter a hex value to match the first byte. The default value is 0.
`rulePayloadByte2Mask` *`hexUInt`*	**Optional.** Specifies a mask for the byte 2 value. The default is 0, which masks nothing (matches everything). The default is 0xff if a non-zero value is specified for value.
`ruleDescr` *`text`*	**Optional.** Associates a text description with the rule. Enter a description up to 64 characters. If the description contains multiple words separated by spaces, enclose it in quotation marks. This description is displayed with the `show accessList rule verbose` command.

Example

The following example defines an access rule for the access list named ACL1 and specifies the Generic Routing Encapsulation Protocol (gre). In this example, the specified access list already exists (configured with the `accessList` command).

```
sun> enable
sun# config
sun(config)# vswitch e-commerce vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# accesslist ACL1
sun(...ip accessList-ACL1)# rule ruleIndex 1 ruleAction permit
ruleProto gre ruleDescr “rule for generic TCP protocol”
```

Delete filters

See the `show accessList rule` command for argument descriptions.

```
no vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [rulePayloadByte1Value hexUInt]
   [rulePayloadByte1Mask hexUInt]
   [rulePayloadByte2Value hexUInt]
   [rulePayloadByte2Mask hexUInt]
   [ruleDescr text]
```

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → delete
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → delete

accessList rule (for ICMP)

Purpose

Assigns rules to an access list for ICMP packets. If you create access rules without first having created an access list, an access list with the name you specify is automatically created.

Each argument of this command identifies criteria that a packet must meet for a match to occur. Optional arguments have a default setting that is also part of the match criteria. A packet must meet all criteria to be considered a match. When a match is established, the system implements the configured action of `permit` or `deny`.

The `no` form of the command deletes the specified rule.

ICMP messages

The following is a complete list of ICMP messages. Any of these values can be used with the `ruleIcmpType` argument to define match criteria for permitting or denying traffic.

ICMP message	Message description or definition
`administratively-prohibited`	ICMP type 3 (destination unreachable), code 13 (prohibited)
`dod-host-prohibited`	ICMP type 3 (destination unreachable), code 10 (host prohibited)
`dod-net-prohibited`	ICMP type 3 (destination unreachable), code 9 (net prohibited)
`echo`	ICMP type 8, echo (ping) request
`echo-reply`	ICMP type 0, echo (ping) reply
`general-parameter-problem`	ICMP type 12
`host-isolated`	ICMP type 3 (destination unreachable), code 8 (host isolated)
`host-precedence-unreachable`	ICMP type 3 (destination unreachable), code 14 (host precedence violation)
`host-redirect`	ICMP type 5 (redirect), code 1 (host redirect)
`host-tos-redirect`	ICMP type 5 (redirect), code 3 (host TOS redirect)

ICMP message	Message description or definition
host-tos-unreachable	ICMP type 3 (destination unreachable), code 12 (invalid host TOS)
host-unknown	ICMP type 3 (destination unreachable), code 7 (host unknown)
host-unreachable	ICMP type 3 (destination unreachable), code 1 (host unreachable)
information-reply	ICMP type 16
information-request	ICMP type 15
mask-reply	ICMP type 18
mask-request	ICMP type 17
net-redirect	ICMP type 5 (redirect), code 0 (net redirect)
net-tos-redirect	ICMP type 5 (redirect), code 2 (net TOS redirect)
net-tos-unreachable	ICMP type 3 (destination unreachable), code 11 (invalid net TOS)
net-unreachable	ICMP type 3 (destination unreachable), code 0 (net unreachable)
network-unknown	ICMP type 3 (destination unreachable), code 6 (net unknown)
parameter-problem	ICMP type 12
port-unreachable	ICMP type 3 (destination unreachable), code 3 (port unreachable)
precedence-unreachable	ICMP type 3 (destination unreachable), code 15 (precedence cutoff)
reassembly-timeout	ICMP type 11, code 1 (TTL exceeded in reassembly)
redirect	ICMP type 5
router-advertisement	ICMP type 9
router-solicitation	ICMP type 10
source-quench	ICMP type 4
source-route-failed	ICMP type 3 (destination unreachable), code 5 (source route fail)
time-exceeded	ICMP type 11
timestamp-reply	ICMP type 14
timestamp-request	ICMP type 13

ICMP message	Message description or definition
`traceroute`	ICMP type 3 (destination unreachable), code 4 (need fragmentation)
`ttl-exceeded`	ICMP type 11, code 0 (TTL exceeded in transit)
`unreachable`	ICMP type 3, all ICMP unreachables

Access mode

config

Syntax

To create an ICMP rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   ruleAction {deny | permit}
   ruleProto protocol
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleIcmpType message]
   [ruleDescr text]
```

To modify an ICMP rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleIcmpType message]
   [ruleDescr text]
```

Arguments

Argument name	Description
`aclListName` *`text`*	Specifies the name of the access control list to which you are adding rules. This is the name you assigned when you created the list, or if the list doesn't yet exist, the system creates it.
`ruleIndex` *`index`*	Assigns a precedence to the rule, which sets the order in which the rule will be evaluated. The lower the number, the earlier it is evaluated. Valid rule numbers are 1 through 4095.
`ruleAction {deny | permit}`	Specifies the action to take on packets meeting the rule criteria. If the rule has an action of `permit`, and the packet matches, it is forwarded. If the action is `deny`, the packet is rejected. There is no default, as this is a required argument. However, the default action for a packet not matching any rule is `deny`. This argument is required when creating an ACL rule, but optional when modifying or deleting it.
`ruleProto` *`protocol`*	Identifies the IP protocol that must be matched to meet filtering conditions. In this case, ICMP is the protocol.
`ruleSrcAddrs` *`text`*	**Optional**. Specifies the host or network from which the packet originated. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all source addresses.
`ruleDstAddrs` *`text`*	**Optional**. Specifies the host or network for which the packet is destined. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all destinations.
`ruleIcmpType` *`message`*	**Optional**. Specifies the ICMP message type to match against ICMP packets. (That is, any inbound or outbound ICMP packet that additionally matches this specified message type.) See "ICMP messages" for a list of messages. The default message type is `any`, which matches all ICMP messages.
`ruleDescr` *`text`*	**Optional**. Associates a text description with the rule. Enter a description up to 64 characters. If the description contains multiple words separated by spaces, enclose it in quotation marks. This description is displayed with the `show accessList rule verbose` command.

Example

The following example defines an access rule for ACL2 that denies traffic that matches the administratively prohibited ICMP message. In this example, the specified access list already exists (configured with the `accessList` command).

```
sun> enable
sun# config
sun(config)# vswitch e-commerce vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# accesslist ACL2
sun(...ip accessList-ACL2)# rule ruleIndex 1 ruleAction deny ruleProto
icmp ruleIcmpType administratively-prohibited ruleDescr “rule for
ICMP”
```

Delete filters

See the `show accessList rule` command for argument descriptions.

```
no vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleIcmpType message]
   [ruleDescr text]
```

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → delete
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → delete

accessList rule (for TCP)

Purpose

Assigns rules to an access list when the protocol type is TCP. If you create access rules without first having created an access list, an access list with the name you specify is automatically created.

Each argument of this command identifies criteria that a packet must meet for a match to occur. Optional arguments have a default setting that is also part of the match criteria. A packet must meet all criteria to be considered a match. When a match is established, the system implements the configured action of `permit` or `deny`.

The `no` form of the command deletes the specified rule.

TCP port keywords

The following table lists all the acceptable TCP port keywords. Any of these values can be used with the `ruleTcpSrcPort` or `ruleTcpDstPort` argument to define match criteria for permitting or denying traffic.

TCP source or destination port keyword	Keyword description or definition
`bgp`	Border Gateway Protocol (179)
`chargen`	Character generator (19)
`daytime`	Daytime (13)
`discard`	Discard (9)
`domain`	Domain Name Service (53)
`echo`	Echo (7)
`exec`	Exec (rsh, 512)
`finger`	Finger (79)
`ftp`	File Transfer Protocol (21)
`ftp-data`	FTP data connections (used infrequently, 20)
`gopher`	Gopher (70)

TCP source or destination port keyword	Keyword description or definition
hostname	NIC hostname server (101)
http	Hypertext Transfer Protocol (80)
https	Hypertext Transfer Protocol Secure (443)
ident	Ident Protocol (113)
irc	Internet Relay Chat (194)
klogin	Kerberos login (543)
kshell	Kerberos shell (544)
login	Login (rlogin, 513)
lpd	Printer service (515)
nntp	Network News Transport Protocol (119)
pim-auto-rp	PIM Auto-RP (496)
pop2	Post Office Protocol v2 (109)
pop3	Post Office Protocol v3 (110)
smtp	Simple Mail Transport Protocol (25)
sunrpc	Sun Remote Procedure Call (111)
syslog	Syslog (514)
tacacs	TAC Access Control System (49)
talk	Talk (517)
telnet	Telnet (23)
time	Time (37)
uucp	Unix-to-Unix Copy Program (540)
whois	Nicname (43)

Access mode

config

Syntax

To create a TCP rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   ruleAction {deny | permit}
   ruleProto protocol
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleTcpSrcPort tcpSourcePort]
   [ruleTcpDstPort tcpDestPort
   [established {true | false}]
   [ruleDescr text]
```

To modify a TCP rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleTcpSrcPort tcpSourcePort]
   [ruleTcpDstPort tcpDestPort
   [established {true | false}]
   [ruleDescr text]
```

Arguments

Argument name	Description
`aclListName text`	Specifies the name of the access control list to which you are adding this rule. This is the name you assigned when you created the list, or if the list doesn't yet exist, the system creates it.
`ruleIndex index`	Assigns a precedence to the rule, which sets the order in which the rule will be evaluated. The lower the number, the earlier it is evaluated. Valid rule numbers are 1 through 4095.

Argument name	Description
`ruleAction {deny | permit}`	Specifies the action to take on packets meeting the rule criteria. If the rule has an action of `permit`, and the packet matches, it is forwarded. If the action is `deny`, the packet is rejected. There is no default, as this is a required argument. However, the default action for a packet not matching any rule is `deny`. This argument is required when creating an ACL rule, but optional when modifying or deleting it.
`ruleProto` *`protocol`*	Identifies the IP protocol that must be matched to meet filtering conditions. In this case, TCP is the protocol. See "ACL protocol types" for a list of protocols you can select as matching criteria. This argument is required when creating an ACL rule, but optional when modifying or deleting it.
`ruleSrcAddrs` *`text`*	**Optional.** Specifies the host or network from which the packet originated. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all source addresses.
`ruleDstAddrs` *`text`*	**Optional.** Specifies the host or network for which the packet is destined. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all destinations.
`ruleTcpSrcPort` *`tcpSourcePort`*`]`	**Optional.** Specifies the source TCP port to match against TCP packets. See "TCP port keywords" for a list of port types. The default port type is `any`, which matches all TCP ports.
`ruleTcpDstPort` *`tcpDestPort`*	**Optional.** Specifies the destination TCP port to match against TCP packets. See "TCP port keywords" for a list of port types. The default port type is `any`, which matches all TCP ports.
`established {true | false}`	**Optional.** Specifies whether to match only on established sessions. When set to `true`, the rule matches packets on connections that have already been opened from the other side of the TCP connection, indicated by the ACK bit being set. The default setting is `false` (no filtering on the ACK bit).
`ruleDescr` *`text`*	**Optional.** Associates a text description with the rule. Enter a description up to 64 characters. If the description contains multiple words separated by spaces, enclose it in quotation marks. This description is displayed with the `show accessList rule verbose` command.

Example

The following example defines an access rule for ACL3 that permits traffic that matches a TCP HTTP source. In this example, the specified access list already exists (configured with the `accessList` command).

```
sun> enable
sun# config
sun(config)# vswitch e-commerce vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# accesslist ACL3
sun(...ip accessList-ACL3)# rule ruleIndex 1 ruleAction permit
ruleProto tcp ruleTcpSrcPort http ruleDescr “rule for TCP”
```

Delete filters

See the `show accessList rule` command for argument descriptions.

```
no vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleTcpSrcPort tcpSourcePort]
   [ruleTcpDstPort tcpDestPort
   [established {true | false}]
   [ruleDescr text]
```

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → delete
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → delete

`accessList rule` (for UDP)

Purpose

Assigns rules to an access list when the protocol type is UDP. If you create access rules without first having created an access list, an access list with the name you specify is automatically created.

Each argument of this command identifies criteria that a packet must meet for a match to occur. Optional arguments include UDP port numbers, with a default setting that is also part of the match criteria. A packet must meet all criteria to be considered a match. When a match is established, the system implements the configured action of `permit` or `deny`.

The `no` form of the command deletes the specified rule.

UDP port keywords

The following table lists all the acceptable UDP port keywords. Any of these values can be used with the `ruleUdpSrcPort` or `ruleUdpDstPort` argument to define match criteria for permitting or denying traffic.

UDP source or destination port keyword	Keyword description or definition
`0-65535`	Port number
`biff`	Biff (mail notification, comsat, 512)
`bootpc`	Bootstrap Protocol (BOOTP) client (68)
`bootps`	Bootstrap Protocol (BOOTP) server (67)
`discard`	Discard (9)
`dnsix`	DNSIX security protocol auditing (195)
`domain`	Domain Name Service (DNS, 53)
`echo`	Echo (7)
`isakmp`	Internet Security Association and Key Management Protocol (500)
`mobile-ip`	Mobile IP registration (434)
`nameserver`	IEN116 name service (obsolete, 42)

UDP source or destination port keyword	Keyword description or definition
`netbios-dgm`	NetBios datagram service (138)
`netbios-ns`	NetBios name service (137)
`netbios-ss`	NetBios session service (139)
`ntp`	Network Time Protocol (123)
`pim-auto-rp`	PIM Auto-RP (496)
`rip`	Routing Information Protocol (router, in.routed, 520)
`snmp`	Simple Network Management Protocol (161)
`Snmptrap`	SNMP Traps (162)
`sunrpc`	Sun Remote Procedure Call (111)
`syslog`	System Logger (514)
`tacacs`	TAC Access Control System (49)
`talk`	Talk (517)
`tftp`	Trivial File Transfer Protocol (69)
`time`	Time (37)
`who`	Who service (rwho, 513)
`xdmcp`	X Display Manager Control Protocol (177)

Access mode

config

Syntax

To create a UDP rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   ruleAction {deny | permit}
   ruleProto protocol
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleUdpSrcPort udpSourcePort]
   [ruleUdpDstPort udpDestPort]
   [ruleDescr text]
```

To modify a UDP rule:

```
vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleUdpSrcPort udpSourcePort]
   [ruleUdpDstPort udpDestPort]
   [ruleDescr text]
```

Arguments

Argument name	Description
`aclListName` *text*	Specifies the name of the access control list to which you are adding rules. This is the name you assigned when you created the list, or if the list doesn't yet exist, the system creates it.
`ruleIndex` *index*	Assigns a precedence to the rule, which sets the order in which the rule will be evaluated. The lower the number, the earlier it is evaluated. Valid rule numbers are 1 through 4095.

Argument name	Description
`ruleAction {deny | permit}`	Specifies the action to take on packets meeting the rule criteria. If the rule has an action of `permit`, and the packet matches, it is forwarded. If the action is `deny`, the packet is rejected. There is no default, as this is a required argument. However, the default action for a packet not matching any rule is `deny`. This argument is required when creating an ACL rule, but optional when modifying or deleting it.
`ruleProto` *`protocol`*	Identifies the IP protocol that must be matched to meet filtering conditions. In this case, UDP is the protocol. See "ACL protocol types" for a list of protocols you can select as matching criteria.
`ruleSrcAddrs` *`text`*	**Optional.** Specifies the host or network from which the packet originated. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all source addresses.
`ruleDstAddrs` *`text`*	**Optional.** Specifies the host or network for which the packet is destined. See for a "Entering source and destination addresses" list of formats with which you can enter the address(es). The default source address is `any`, which matches all destinations.
`ruleUdpSrcPort` *`udpsourcePort`*	**Optional.** Specifies the source UDP port to match against UDP packets. See "UDP port keywords" for a list of ports. The default port type is `any`, which matches all UDP ports.
`ruleUdpDstPort` *`udpdestPort`*	**Optional.** Specifies the destination UDP port to match against UDP packets. See "UDP port keywords" for a list of ports. The default port type is `any`, which matches all UDP ports.
`ruleDescr` *`text`*	**Optional.** Associates a text description with the rule. Enter a description up to 64 characters. If the description contains multiple words separated by spaces, enclose it in quotation marks. This description is displayed with the `show accessList rule verbose` command.

Example

The following example defines an access rule for ACL4 that permits traffic that matches a UDP NTP source. In this example, the specified access list already exists (configured with the `accessList` command).

```
sun> enable
sun# config
sun(config)# vswitch e-commerce vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# accesslist ACL4
sun(...ip accessList-ACL4)# rule ruleIndex 1 ruleAction permit
ruleProto udp ruleUdpSrcPort ntp ruleDescr “rule for UDP”
```

Delete filters

See the `show accessList rule` command for argument descriptions.

```
no vSwitch-name vRouter-name ip accessList rule
   aclListName text
   ruleIndex integer
   [ruleAction {deny | permit}]
   [ruleProto protocol]
   [ruleSrcAddrs text]
   [ruleDstAddrs text]
   [ruleUdpSrcPort udpSourcePort]
   [ruleUdpDstPort udpDestPort]
   [ruleDescr text]
```

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → delete
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → add
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → copy
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → modify
- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose → delete

show accessGroup

Purpose

Displays each interface and its associated access list. Each interface can have up to one inbound and one outbound list associated with it. In addition, the output details the state of the ACL on the interface and the inbound or outbound application of the filter. There are three ways to view the “hits” (the number of times a packet matched the rule) to a particular list, which are described below.

Command	Description
show accessGroup	Displays an accumulation of all hits to a named list applied to an interface in a single direction (per access group). The command shows the total number of hits to a list, calculated by adding the number of hits to each rule.
show accessGroup status	Displays a breakdown of the number of hits to each rule within a list per access group.
show accessList rule status	Displays the total number of hits to a rule across all interfaces in the vSwitch.

Access mode

user

Syntax

vSwitch-*name* vRouter-*name* show ip accessGroup

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show accessgroup
IfName          Direction  Name                 Status     Hits
eth.1.2         in         testlist             active     0
eth.1.2         out        testlist             active     0
```

Output description

Field name	Description	Filter name
`IfName`	The name of the interface to which you applied a specific access list (the interface that connects to the IP instance). This could be either `vlan.x`, `lag.x`, or `eth.x.x`. Use the `show interface` command to verify configured IP-to-lower layer interface associations. Use the `interface` command to configure new associations.	`ifName` *`interfaceName`*
`Direction`	The direction of traffic, either inbound or outbound, that is affected by named ACL.	`direction {in | out}`
`Name`	The name of the access control list that will filter this interface's traffic. The name is a list you created with the `accessList` command.	`aclListName` *`text`*
`Status`	The status of the access control list on the interface: `active`: the ACL is configured and applied to the interface and is filtering traffic. `inactive`: the ACL has been configured, but has not been applied to an interface yet (with the `accessGroup` command). `disabled`: the ACL has been configured, but has been administratively disabled with the `accessList` command.	`operStatus {active | inactive | disabled}`
`Hits`	A summation of the hits listed in the `show accessGroup status` command.	`hits` *`integer`*

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessGroup

show accessGroup status

Purpose

Displays for each interface its associated access list and hits per rule. There are three ways to view the “hits” to a particular list, which are described below.

Command	Description
show accessGroup	Displays an accumulation of all hits to a named list applied to an interface in a single direction (per access group). That is, shows the total number of hits to a list, calculated by adding the number of hits to each rule.
show accessGroup status	Displays a breakdown of the number of hits to each rule within a list per access group.
show accessList rule status	Displays the total number of hits to a rule across all interfaces in the vSwitch.

Access mode

user

Syntax

vSwitch-*name* vRouter-*name* show ip accessGroup status

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show ip accessgroup
status
 IfName      Direction  Name     Rule Index  Hits
 eth.1.21    in         doc      1           0
 eth.1.21    in         doc      3           0
 eth.1.21    in         doc      4           0
```

```
IfName      Direction  Name     Rule Index  Hits
eth.1.21    out        sqa      1           5
eth.1.21    out        sqa      2           0
eth.1.21    out        sqa      4           0
```

Output description

Field name	Description	Filter name
`IfName`	The name of the interface to which you applied a specific access list (the interface that connects to the IP instance). This could be either `vlan.`*`x`*, `lag.`*`x`*, or `eth.`*`x.x`*. Use the `show interface` command to verify configured IP-to-lower layer interface associations. Use the `interface` command to configure new associations.	`ifName` *`interfaceName`*
`Direction`	The direction of traffic, either inbound or outbound, that is affected by named ACL.	`direction {in | out}`
`Rule Index`	The precedence of the rule, which indicates the order in which the system evaluates the rule. The lower the number, the earlier the system evaluates it. Valid rule numbers are 1 through 4095.	`ruleIndex` *`integer`*
`Name`	The name of the access control list that will filter this interface's traffic. The name is a list you created with the `accessList` command.	`aclListName` *`text`*
`Hits`	The number of times the rule has been applied.	`hits` *`integer`*

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* →Ip → accessGroup → status

show accessList

Purpose

Displays the operational and administrative details of all or a specified access control list on the vRouter. The conditions of the list are set with the `accessList` command; the rules for the list are set with the `accessList rule` command.

Access mode

user

Syntax

`vSwitch-`*`name`* `vRouter-`*`name`* `show ip accessList`

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show accesslist
Name                State     Status     Interfaces
CaplanList          enabled   active     vlan.100-in
```

Output description

Field name	Description	Filter name
`Name`	The name of the access control list. This is the name you assigned when you created the list with the `accessList` command.	`aclListName` *`text`*
`State`	The ACL's administrative mode, either `enabled` or `disabled`.	`adminState {enabled | disabled}`

Field name	Description	Filter name
`Status`	The status of the access control list: `active`: the ACL is configured and applied to the interface and is filtering traffic. `inactive`: the ACL has been configured, but has not been applied to an interface yet (with the `accessGroup` command). `disabled`: the ACL has been configured, but has been administratively disabled with the `accessList` command.	`operStatus {active | inactive | disabled}`
`Interfaces`	The interface(s) on which this access list is activated. In addition, the direction in which the access list applies is appended to the interface name, in the format `-direction`.	`ipAclIfs` *`text`*

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList

show accessList verbose

Purpose

Displays the operational and administrative details of all or a specified access control list on the vRouter. In addition, the verbose display includes the configured text description with the output.

The conditions of the list are set with the `accessList` command; the rules for the list are set with the `accessList rule` command.

Access mode

user

Syntax

`vSwitch-`*`name`* `vRouter-`*`name`* `show ip accessList verbose`

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show accesslist
verbose
```

Output description

Field name	Description	Filter name
`Name`	The name of the access control list. This is the name you assigned when you created the list with the `accessList` command.	`aclListName` *`text`*
`State`	The ACL's administrative mode, either `enabled` or `disabled`. You must set the mode to `disabled` when you want to modify a list.	`adminState {enabled | disabled}`

Field name	Description	Filter name
`Description`	The text description associated with the named list.	`description` *`text`*
`Status`	The status of the access control list: active: the ACL is configured and applied to the interface and is filtering traffic. inactive: the ACL has been configured, but has not been applied to an interface yet (with the `accessGroup` command). disabled: the ACL has been configured, but has been administratively disabled with the `accessList` command.	`operStatus {active | inactive | disabled}`
`Interfaces`	The interface(s) on which this access list is activated. In addition, the direction in which the access list applies is appended to the interface name, in the format *`-direction`*.	`ipAclIfs` *`text`*

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → ip → accessList → verbose

show accessList rule

Purpose

Displays a brief summary of each named access list rule on the vRouter.

Access mode

user

Syntax

`vSwitch-`*`name`* `vRouter-`*`name`* `show ip accessList rule`

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show accesslist
rule
Summary
CaplanList.1 deny udp any  any
CaplanList.2 permit tcp any  any
```

Output description

Field name	Description
`Summary`	A single line summary of the list name (concatenated with the rule index number), list action, protocol, and associated parameters. You can filter this output using the argument names in the `accessList rule` command.

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule

show accessList rule status

Purpose

Provides a summary display of all or specified access control list rules. The summary includes a brief view of the configured elements of the rule (only the non-default argument values) and the number of packets affected by the rule (hits). There are three ways to view the "hits" to a particular list, which are described below.

Command	Description
show accessGroup	Displays an accumulation of all hits to a named list applied to an interface in a single direction (per access group). That is, shows the total number of hits to a list, calculated by adding the number of hits to each rule.
show accessGroup status	Displays a breakdown of the number of hits to each rule within a list per access group.
show accessList rule status	Displays the total number of hits to a rule across all interfaces in the vSwitch.

Access mode

user

Syntax

vSwitch-*name* vRouter-*name* show ip accessList rule status

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# vrouter default
sun(config-vswitch-e-commerce vrouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show accesslist
rule status

Summary                                                    Hits
CaplanList.1 deny udp any  any                             0
CaplanList.2 permit tcp any  any                           0
sun(config-vSwitch-caplan vRouter-default ip)#
```

Output description

Field name	Description	Filter name
Name	The name of the access control list that filters this interface's traffic. The name you specify is a list you created with the accessList command.	aclListName text
Rule Index	The table index for the rule, which sets rule precedence (the order in which the rules are evaluated). The lower the number, the earlier it is evaluated. Valid rule numbers are 1 through 4095.	ruleIndex *integer*
Rule Summary	A summary of the ACL's configuration. The summary begins with a concatenated *name.index*, and continues with the elements of the rule that have been manually configured.	ruleSummary *text*
Hits	The number of packets that have been permitted or denied by this ACL rule.	ruleHits *integer*

Associated MIB

acl.mib

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → status

show accessList rule verbose

Purpose

Displays a detailed summary of each named access list on the vRouter.

Access mode

user

Syntax

vSwitch-*name* vRouter-*name* show ip accessList rule verbose

Sample output

```
sun> enable
sun# config
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# ip
sun(config-vSwitch-e-commerce vRouter-default ip)# show accesslist
rule verbose
Summary:      CaplanList.1 deny udp any  any
Name:         CaplanList
Index:        1
Action:       deny
Protocol:     udp
Src Addr(s):  any
Dst Addr(s):  any
Udp Src Port: any
Udp Dst Port: any
Src Port(s):  0-65535
Dst Port(s):  0-65535
Description:  N/A

Summary:      CaplanList.2 permit tcp any  any
Name:         CaplanList
Index:        2
Action:       permit
Protocol:     tcp
Src Addr(s):  any
Dst Addr(s):  any
Tcp Src Port: any
Tcp Dst Port: any
Src Port(s):  0-65535
Dst Port(s):  0-65535
Description:  N/A
```

Output description

Field name	Description	Filter name
`Summary`	A single line summary of the list name, list action, protocol, and associated parameters. To view just this summary, use the `show accessList rule` command.	`ruleSummary` *`text`*
`Name`	The name of the access control list to which the displayed rules apply. This is the name you assigned when you created the list with the `accessList` command.	`aclListName` *`text`*
`Index`	The table index for the rule, which sets rule precedence (the order in which the rules are evaluated). The lower the number, the earlier it is evaluated. Valid rule numbers are 1 through 4095.	`ruleIndex` *`integer`*
`Action`	The action to take on packets meeting the rule criteria. If the rule has an action of `permit`, and the packet matches, it is forwarded. If the action is `deny`, the packet is rejected. There is no default, as this is a required argument. However, the default action for a packet not matching any rule is `deny`.	`ruleAction {permit | deny}`
`Protocol`	The IP protocol that must be matched to meet filtering conditions. See "ACL protocol types" for a list of protocols you can select as matching criteria.	`ruleProto` *`protocol`*
`Src Addr(s)`	The host or network from which the packet originated. See "Entering source and destination addresses"for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all source addresses.	`ruleSrcAddrs` *`text`*

Field name	Description	Filter name
`Dst Addr(s)`	The host or network for which the packet is destined. See "Entering source and destination addresses" for a list of formats with which you can enter the address(es). The default source address is `any`, which matches all destinations.	`ruleDstAddrs` *`text`*
`Byte 1 Value`	The first byte of the IP data field, in hex. The default value is 0.	`rulePayloadByte1Value` *`hexUInt`*
`Byte 1 Mask`	The mask for the byte 1 value. The default is 0, which masks nothing (matches anything). The default is 0xff if a non-zero value is specified for value.	`rulePayloadByte1Mask` *`hexUInt`*
`Byte 2 Value`	The second byte of the IP data field, in hex. The default value is 0.	`rulePayloadByte2Value` *`hexUInt`*
`Byte 2 Mask`	The mask for the byte 2 value. The default is 0, which masks nothing (matches anything). The default is 0xff if a non-zero value is specified for value.	`rulePayloadByte2Mask` *`hexUInt`*
`Icmp Type`	The ICMP message type to match against ICMP packets. (That is, any inbound or outbound ICMP packet that additionally matches this specified message type.) See "ICMP messages" for a list of messages. The default message type is `any`, which matches all ICMP messages.	`ruleIcmpType` *`message`*
`Udp Src Port`	The source UDP port to match against UDP packets. (That is, any outbound UDP packet that additionally matches this specified port.) See "UDP port keywords" for a list of ports. The default port type is `any`, which matches all UDP ports.	`ruleUdpSrcPort` *`udpsourcePort`*
`Udp Dst Port`	The destination UDP port to match against UDP packets. (That is, any inbound UDP packet that additionally matches this specified port.) See "UDP port keywords" for a list of ports. The default port type is `any`, which matches all UDP ports.	`ruleUdpDstPort` *`udpdestPort`*

Field name	Description	Filter name
`TCP Src Port`	The source TCP port to match against TCP packets. (That is, any outbound TCP packet that additionally matches this specified port.) See "TCP port keywords" for a list of ports. The default port type is `any`, which matches all TCP ports.	`ruleTcpSrcPort` *`tcpsourcePort`*
`TCP Dst Port`	The destination TCP port to match against TCP packets. (That is, any inbound TCP packet that additionally matches this specified port.) See "TCP port keywords" for a list of ports The default port type is `any`, which matches all TCP ports.	`ruleTcpDstPort` *`tcpdestPort`*
`Src Port(s)`	The numeric value for the source port. (Compare to the port keyword value.)	`ruleSrcPorts` *`text`*
`Dst Port(s)`	The numeric value of the destination port. (Compare to the port keyword value.)	`ruleDstPorts` *`text`*
`Description`	A text description associated with the rule. This description is configured with the `accessList` command.	`ruleDescr` *`text`*

Associated MIB

`acl.mib`

Web path

- vSwitch → *name* → vRouter → *name* → Ip → accessList → rule → verbose

Part VII. Load balancing and Secure Sockets Layer

The chapters in Part VII describe the commands for configuring and monitoring load balancing and SSL functions in the system.

Chapter 29. Load-balancing commands

Load-balancing description

This chapter describes the commands for configuring L4 to L7 load balancing and object switching on the N2000 Series application switch. For complete information on how to configure load-balancing capabilities using the commands in this chapter, and for integrated configuration examples, see the *Sun N2000 Series Release 2.0 – System Configuration Guide*.

The N2000 Series supports the following load-balancing functionality:

- Basic L4 TCP traffic load balancing
- Advanced L4 traffic load balancing with additional algorithm selections available
- Advanced L4 traffic load balancing with Secure Sockets Layer (SSL) transactions between the client and server for secure applications
- L5 to L7 HTTP policy-based matching
- L5 to L7 HTTPS policy-based matching with SSL

loadBalance command path

The sections in this chapter show you how to execute the commands from within the following command mode:

`vSwitch` *`name`* `loadBalance`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

Note: The `loadBalance` menu is not available for the system vSwitch.

Load-balancing command summary

Table 29-1 lists and briefly describes the load-balancing commands.

Table 29-1. Load-balancing command summary

Command name	Description
`cookiePersistence`	Configure server persistence rules.
`healthCheckProfile`	Configure health check probe parameters.
`healthCheckProfile passive`	Configure passive health check probe settings.
`healthCheckTest`	Direct the named server health check profile to check a specific target/real service.
`host`	Configure an IP address for a server.
`objectRule`	Configure predicate statements that describe how traffic is evaluated.
`outboundNat dynamic`	Configure outbound dynamic network address translation to shield private IP addresses.
`outboundNat dynamic hostIpRange`	Configure the host IP network address range that translates to the public IP network address.
`outboundNat static`	Configure outbound static network address translation to shield private IP addresses.
`proxyIPPool`	Configure a pool of IP addresses to be used by client address translation and TCP multiplexing.
`realService`	Map a named host to a port and configure load-balancing decision criteria and SSL settings.
`realService advanced`	Fine-tune TCP for the system-to-server portion of the connection.
`realService ssl`	Modify SSL properties for a real service.

Table 29-1. Load-balancing command summary (continued)

Command name	Description
requestPolicy	Configure a virtual service policy that is evaluated to make a forwarding decision to a service group or perform the action defined by a sorry service.
requestTransform	Define any changes to the HTTP header text of an HTTP request if the traffic matches the object rule.
responsePolicy	Provide control over errors returned by real service hosts in back-end service groups.
responseTransform	Modify objects before they are passed through the Virtual Switch load balancer when the contents of these objects match criteria defined in the named object rule.
serviceGroup	Assign real services to groups for load balancing.
show cookiePersistence	Display cookie persistence rules.
show healthCheckProfile	Display health check probe results.
show healthCheckProfile passive	Display passive health check probe settings.
show host	Display server-to-IP address mappings.
show objectRule	Display predicate and action match statements configured for an object rule.
show outboundNat dynamic	Display the current dynamic network address translation configuration.
show outboundNat dynamic hostIpRange	Display the host vRouter and the private IP address range configured for dynamic network address translation.
show outboundNat dynamic statistics	Display byte and packet data that the system transmitted and received under the named dynamic network address translation configuration.
show outboundNat static	Display static network address translation mapping.
show outboundNat static statistics	Display static network address translation statistics for traffic through the network address translation configuration.

Table 29-1. Load-balancing command summary (continued)

Command name	Description
`show proxyIPPool`	Display the configured proxy IP pools on a vRouter.
`show proxyIPPool statistics`	Display the proxy IP pool statistics on a vRouter.
`show realService`	Display server load-balancing configuration.
`show realService advanced`	Display TCP settings for the system-to-server portion of the connection.
`show realService slbInfo`	Display health check probe results for specified real service(s).
`show realService ssl`	Display SSL configurations for a real service.
`show realService ssl statistics`	Display SSL connection statistics for a real service.
`show realService statistics`	Display total receive, transmit, and session statistics for a real service, and Layer 5 through Layer 7 connection data.
`show requestPolicy`	Display settings for the named request policy or all request policies configured.
`show requestPolicy statistics`	Display counters associated with the named request policy.
`show requestTransform`	Display settings for one or more request transforms.
`show responsePolicy`	Display the attributes of one or more response policies.
`show responsePolicy statistics`	Display the number of responses that were initiated as a result of an object rule and action match.
`show responseTransform`	Display the attributes for one or more response transforms.
`show serviceGroup`	Display service group configuration.
`show serviceGroup slbInfo`	Display health check probe results for specified service group(s).
`show serviceGroup slbinfo activation`	Display information about standby activation.
`show serviceGroup slbinfo script status`	Display the status of the scripted health check probe.

Table 29-1. Load-balancing command summary (continued)

Command name	Description
`show serviceGroup slbinfo standby`	Display the activity of standby devices.
`show serviceGroup slbInfo advanced counters`	Display counters indicating probe results.
`show serviceGroup slbInfo advanced history`	Display a summary of the probe history.
`show serviceGroup slbInfo inline`	Display inline health check probe results for specified service group(s).
`show serviceGroup slbInfo interval`	Display operational probe intervals and statistics.
`show serviceGroup statistics realServiceSummary`	Display statistics for one or more real services in the service group.
`show serviceGroup statistics summary`	Display server load-balancing empty counts, which is the count of the number of times an object could not be forwarded because no real services were available.
`show sorryData`	Display the attributes of a named sorry data definition.
`show summary`	Display active and inactive virtual services.
`show tideRunner congestion status`	Display congestion statistics for a N2000 Series function card.
`show tideRunner realService statistics`	Display statistics for real service communication on a function card.
`show tideRunner realService sslStatistics`	Display SSL statistics for real service communication on a function card.
`show tideRunner virtualService statistics`	Display statistics for virtual service communication on a function card.
`show tideRunner virtualService sslStatistics`	Display SSL statistics for virtual service communication on a function card.
`show virtualService`	Display the configuration for one or more virtual services.
`show virtualService advanced`	Display TCP settings for the client-to-switch portion of the connection.
`show virtualService ssl`	Display SSL configurations for a virtual service.

Table 29-1. Load-balancing command summary (continued)

Command name	Description
`show virtualService ssl statistics`	Display SSL connection statistics for a virtual service.
`show virtualService statistics`	Display total receive, transmit, and session statistics for a virtual service, and Layer 5 through Layer 7 connection data.
`show vsGroup`	Display the virtual service group configuration(s) on a vSwitch.
`sorryData`	Define the action the load balancer should take when a sorry action is specified for a request policy or response policy.
`virtualService`	Assign a virtual IP (VIP) address to the load balancer and configure client-side settings.
`virtualService advanced`	Fine-tune TCP for the client-to-switch portion of the connection.
`virtualService ssl`	Modify SSL properties for a virtual service.
`vsGroup`	Configure one or more virtual service groups, where the named virtual services included in a group may share request policies, service groups, or real services.

Load-balancing basic configuration

Table 29-2. Steps for configuring load balancing

Step	Action
1.	Configure servers as hosts (using the `host` command).
2.	Configure hosts as real services (using the `realService` command).
3.	Assign real services to service groups (using the `serviceGroup` command).
4.	Create object rules for matching HTTP request traffic (using the `objectRule` command).
5.	Create request policies to specify the action (forward or sorry service) of matching HTTP traffic (using the `requestPolicy` command).
6.	Configure the virtual service (using the `virtualService` command).

Note: The entity (for example, a real service or a host) will not exist until you create it. However, you can perform the steps out of order because you can reference an entity before you create it. You can then configure the elements of that named entity at a later time. The configuration will not be complete until you create the entity.

cookiePersistence

Purpose

Defines the string that is inserted into an HTTP response message header before it is returned to the client.

In switch-managed cookie mode, the N2000 Series load balancer inserts cookie strings into HTTP server response packets returned to the client. Cookies will be inserted only when the `requestPolicy persistType` is set to `switchCookie`. The client stores the cookies and includes them in subsequent HTTP client requests to the same server. When the load balancer receives a request containing a cookie, the load balancer deciphers the cookie, and then uses an object rule to forward the traffic to the same real Web server where the cookie originated. During the session, HTTP responses use the same cookie, keeping the connection persistent until the client closes the session.

This command defines the persistence rule for the session. Use this command to define the cookie; use the `requestPolicy` command to assign a named cookie to a request policy.

The N2000 Series uses switched-managed cookie mode (also known as cookie-insert) in load balancing. In this mode, the system makes a load-balancing decision, forwards the request to the service, and creates and inserts the cookie into the server's response packet. In subsequent client requests, the system deciphers the cookie and selects the same real service for forwarding.

When you create a real service, the system generates a unique, 32-bit hash key based on the real service name. This key is inserted into the `cookieName` field, and used to identify the client session. If you specify the `cookieDomain` and `cookiePath` fields, they are concatenated with `cookieName` to produce the actual string that is inserted into the packet header. For detailed instructions on entering these fields, refer to:

`http://wp.netscape.com/newsref/std/cookie_spec.html`

Session persistence, as provided by the N2000 Series, is only enabled if you set the `cookiePersist` field in the `requestPolicy` command and the `requestPolicy persistType` is set to `switchCookie`. (There may be other cookie fields in the HTTP header that were inserted by the client, however.)

Cookie persistence limitations

Cookie persistence operates with the following limitations:

- Each virtual service definition supports up to six unique cookie persistence definitions.
- Each cookie persistence rule that has a unique `cookieName` counts as one of the six cookies in the virtual service.
- If more than one request policy uses cookie persistence, then the `cookieName` needs to be unique for each cookie persistence rule *or* the `cookiePath` field in the cookie persistence rule entry must be present and unique, and requests to the request policy must only come from that path.

The no form of the command deletes the named cookie from the cookie persistence table.

Access mode

config

Syntax

```
vSwitch-name loadBalance cookiePersistence
   name text
   cookieName text
   [cookieDomain text]
   [cookiePath text]
   [cookieExpires text]
   [secure {true | false}]
```

Arguments

Argument name	Description
`name` *`text`*	Assigns a text string to the persistence rule being defined. If the name specified already exists, the system modifies the configuration for that persistence rule as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`cookieName` *`text`*	Specifies the actual string to be inserted into the packet header. The cookie value is equal to the hash key that the switch generated from the real service name, and takes the form: *`cookieNamecookieDomaincookiePath=hash_key`* The values, either user-specified or default, of `cookieDomain` and `cookiePath` are concatenated with this. The complete text becomes the persistence rule on which forwarding decisions are based. See `http://wp.netscape.com/newsref/std/cookie_spec.html` for details on the appropriate format. This argument is required when creating the named cookie, but optional when modifying or deleting it. The default value is `nnSessionID`.
`cookieDomain` *`text`*	**Optional.** Specifies the domain attribute to compare against the domain portion of the host's fully qualified domain name (FQDN). If you do not specify a `cookieDomain`, the system leaves that field blank in the Set-Cookie header sent to the client. The client will keep the domain of the request (that is, the host name in the URL of the request) with the cookie and will only send the cookie back to the originating server.
`cookiePath` *`text`*	**Optional.** Specifies a path name, which, if the request has passed name and domain matching, the switch compares against the URL path attribute. The default value is a slash (/).

Argument name	Description
`cookieExpires` *`text`*	**Optional.** Specifies either an absolute time and date, or a delta from the current time. The switch compares this setting to the time and date in the header to determine whether a cookie is still valid. If the timer has expired, the client stops sending the cookie. Enter the date-string in one of the following two formats: • Absolute time: If using absolute time, GMT is the only legal time zone and you must use dashes between the date elements. If you do not specify this argument, the cookie will expire at the end of the user's session. `Wdy, DD-Mon-YYYY HH:MM:SS GMT` Strings can be only three characters. Use the following abbreviations to enter the day and month: `Wdy` = Sun Mon Tue Wed Thu Fri Sat `Mon` = Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec • Delta time is entered as follows: *`integer Days | Hours | Minutes | Seconds`* Delta time can be qualified by entering Days, Hours, Minutes, or Seconds following the number and separated by a space. If no qualifier is entered, seconds will be used.
`secure {true | false}`	**Optional**. Specifies whether the secure keyword attribute is used to protect the confidentiality and authenticity of the cookie when the switch issues switch-managed Set-cookie response headers. The secure keyword indicates that the HTTP client should not include the cookie in subsequent non-secure requests, or should only include the cookie in secure requests. The default setting is `false`.

Delete filters

```
no vSwitch-name loadBalance cookiePersistence
name text
[cookieName text]
[cookieDomain text]
[cookiePath text]
[cookieExpires text]
[secure {true | false}]
```

Example

The following example defines a string to be inserted into the HTTP response header. This string has no expiration date.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# cookiePersistence
persistRule1 cookiename cookie123 cookieDomain www.host1.com
cookiePath /shop
```

Associated MIB

```
op.mib
```

Web path

- vSwitch → *name* → LoadBalance → cookiePersistence → delete

healthCheckProfile

Purpose

Configures parameters for server health checks. Health checks provide out-of-band availability status for the servers configured within a service group. (The health check profile is assigned to servers through the `serviceGroup` command.) You can configure up to 512 health check profiles.

Based on the parameters set with this command, the system polls configured hosts and creates an availability table used for forwarding decisions (a load-balancing forwarding table). If a server is unresponsive and taken out of the table, the system continues polling and returns the server to the table when it again responds to health checks.

When the weight of a real service is set to `dynamic`, the system uses the server health checks response results to determine the load balance weights of the real services. The lowest latency value is calculated using the Exponentially Weighted Moving Average (EWMA) to average and maintain health check keep-alive performance. This algorithm determines the delta between the keep-alive latency experienced with this check, divided by 2^5 (32), to the current average latency.

The `no` form of the command deletes the named profile from the health check profile table.

Access mode

config

Syntax

To create a health check profile:

```
vSwitch-name loadBalance healthCheckProfile
   name text
   type {ICMP | TCP | HTTP | FTP | LIST | DNS_UDP | DNS_TCP | RADIUS |
      RAW_UDP | RAW_TCP | RTSP | SCRIPT | POP3 | SMTP | IMAP4}
   [interval integer]
   [retries integer]
   [successRate percentage]
   [timeout integer]
   [count integer]
```

Note: See the Arguments table for a list of available arguments for each probe type.

To modify a health check profile:

```
vSwitch-name loadBalance healthCheckProfile name text
```

Arguments

Argument name	Description
The following arguments are required for all probe types.	
`name` *`text`*	Assigns a text string to the health check profile you are defining. If the name specified already exists, the system modifies the configuration for that policy, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-) You cannot specify the text string `none` as the name.

Argument name	Description
`type {ICMP | TCP | HTTP | FTP | LIST | DNS_UDP | DNS_TCP | RADIUS | RAW_UDP | RAW_TCP | RTSP | SCRIPT | POP3 | SMTP | IMAP4}`	Specifies which probe method the server health check software uses to verify the availability of each server within a service group. This argument is required when creating a health check profile, but optional when deleting one. You cannot modify the `type` set in a profile; instead, you must create a new profile with the desired `type`. Possible `type` values are: `ICMP`: Sends a ping from the load-balancing software to the network interface card (NIC) in the server to verify connectivity and basic functionality `TCP`: Verifies functionality through the NIC and within the operating system. Specifically, this option verifies that the TCP server application is running and associated with a TCP port. `HTTP`: Verifies NIC and OS functionality, as well as the server's storage system. Storage is confirmed by verifying the contents of an HTML page. (The specific page is set with the `requestString` argument.) See the `requestString` argument. `FTP`: Verifies server availability by using FTP transactions. `LIST`: Provides the same verification as the HTTP option, but uses multiple server health check profiles, which can result in HTML pages. See the `profileList` argument. (If `type` is set to `LIST`, the `interval`, `retries`, `successRate`, `timeout`, and `count` argument settings supersede the values defined within an individual health check profile.) `DNS_UDP`: Verifies Domain Name Server using a UDP probe. `DNS_TCP`: Verifies Domain Name Server using a TCP probe. `RADIUS`: Verifies Remote Authentication Dial-In User Service. `RAW_UDP`: Specifies the raw byte stream sent to the server and the expected response over UDP. `RAW_TCP`: Specifies the raw byte stream sent to the server and the expected response over TCP. `RTSP`: Verifies server health by using Real Time Stream Protocol transactions. `SCRIPT`: Verifies network connectivity by using a script. `POP3`: Verifies server availability by using POP3 protocol transactions. `SMTP`: Verifies server availability by using SMTP transactions. `IMAP4`: Verifies server availability by using IMAP4 protocol transactions.

Argument name	Description
The following optional arguments are applicable to all probe types. The required `name` argument and the following optional arguments are the only arguments applicable to the `ICMP` and `TCP` probe types.	
`interval` *seconds*	**Optional.** Specifies the number of seconds between the start of health check probes. Valid range is 2 through 3600 seconds; the default is 5 seconds. The value of `interval` must be greater than the value of `timeout`.
`retries` *probes*	**Optional.** Specifies the number of failed probes allowed before a server is declared out of service and removed from the load-balancing table. You must configure either a `retries` value or a `successRate` value, but not both. The range is 0 through 32; the default value is 3.
`successRate` *percentage*	**Optional.** Specifies the percentage of probes that must succeed before a server is declared out-of-service and removed from the load-balancing table. Setting a percentage helps address those servers that intermittently lose connectivity due to heavy traffic and inadvertently reset the `retries` count making the server appear reliable. You must configure either a `retries` value or a `successRate` value. The range is 0 through 100; the default value is technically 0, but only because the field is mutually exclusive with `retries` and must be set to a value if used.
`timeout` *seconds*	**Optional.** Specifies the number of seconds to wait for a probe response before declaring the probe unanswered. The range is 1 through 10; the default value is 2 seconds. The value of `timeout` must be less than the value of `interval`.
`count` *probes*	**Optional.** Specifies the number of consecutive successful probes the system must receive before it declares the server active and returns its entry to the load-balancing table. The range is 1 through 32; the default value is 3 probes.

Argument name	Description
The following optional arguments are applicable to the HTTP probe type.	
`HTTPMode {checkResponseCodes | hash | searchForResponseString}`	**Optional.** Specifies which method the HTTP probe should use to verify content of the requested object by searching server responses. Possible values are: `checkResponseCodes`: The system searches the response for an HTTP 200 OK response. `hash`: The system runs an arithmetic hash on the object to verify that the requested object has not changed. `searchForResponseString`: The system searches the response for a specified text string. Enter the string with the `responseString` argument. The default setting is `checkResponseCodes`.
`httpRequestString` *`text`*	**Optional.** Specifies the request string that the HTTP probe sends to verify content. Follow the rules of HTTP formats to enter the string. For example: `GET /default.html HTTP/1.1\r\nHOST: 1.1.1.1\r\n\r\n`
`httpResponseString` *`text`*	**Optional.** Specifies the string the probe searches for in server responses. Enter the string in double quotes; spaces are maintained as literal within the quotes. This parameter is only valid when the `type` argument is set to `HTTP` and `HTTPMode` is set to `searchForResponseString`. If `HTTPMode` is not `searchForResponseString`, this parameter is ignored.

Argument name	Description
The following optional arguments are applicable to the FTP probe type.	
`FTPMode {anyServerResponse | goodServerResponse | anyUserResponse | goodUserResponse | anyLoginResponse | goodLoginResponse}`	**Optional.** Specifies the method that the FTP probe uses to verify the server health check responses. Possible values are: `anyServerResponse`: Indicates any response from the server to a connection request. `goodServerResponse`: Indicates a valid response from the server to a connection request. `anyUserResponse`: Indicates any user response from the server after a valid connection response. `goodUserResponse`: Indicates a valid user response from the server after a valid connection response. `anyLoginResponse`: Indicates any login response from the server after a valid connection and valid user response. `goodLoginResponse`: Indicates a valid login response from the server after a valid connection and valid user response.
`FTPTerminationMode {fastDisconnet | quitThenDisconnect}`	**Optional.** Specifies the method of ending an FTP probe session. The default setting is `fastDisconnect`.
`userName` *`text`*	**Optional.** Specifies the user name string used for this FTP probe. The default setting is `anonymous`.
`userPassword` *`text`*	**Optional.** Specifies the password string used for this FTP probe. The default setting is `anonymous@sun.com`.

Argument name	Description
The following optional argument is applicable to the `LIST` probe type.	
`profileList` *`text`*	**Optional.** Specifies a list of named profiles, which this named profile aggregates to make an “umbrella” profile. You can specify up to five profiles in the list. This argument is only available if the `type` argument is set to `LIST`. **Note:** If `type` is set to `LIST`, the `interval`, `retries`, `successRate`, `timeout`, and `count` argument settings for the umbrella profile supersede the values defined within an individual health check profile.

Argument name	Description
The following optional arguments are applicable to the `DNS_TCP` and `DNS_UDP` probe types.	
`dnsMode {anyHostResponse | goodHostResponse | specificHostAddress | anyMailExResponse | goodMailExResponse | specificMailExResponse | verySpecificMailExResponse}`	**Optional.** Specifies the method that the DNS probe uses to verify the server health check responses. Possible values are: `anyHostResponse`: Indicates any response to a DNS host lookup query. `goodHostResponse`: Indicates any valid response to a DNS host lookup query. `specificHostAddress`: Indicates the DNS name resolves to a specific host address. `anyMailExResponse`: Indicates any response to a Mail Exchange query. `goodMailExResponse`: Indicates any valid response to a Mail Exchange query. `specificMailExResponse`: Indicates a valid response to a Mail Exchange query with the specified server entries. `verySpecificMailExResponse`: Indicates a valid response to a Mail Exchange query with the specified preferences/servers. The default setting is `anyHostResponse`.
`dnsReqString` *`text`*	**Optional.** Specifies the domain name for the DNS request. The default setting is `ISI.EDU`.
`dnsRspString` *`text`*	**Optional.** Specifies the information used for evaluating DNS responses, usually an IP address entered in dotted-decimal notation for host requests or a list of mail servers if one of the requests is a Mail Exchange query.

Argument name	Description
The following optional arguments are applicable to the `RADIUS` probe type.	
`radiusMode {anyAuthResponse | goodAuthResponse | goodActgResponse}`	**Optional.** Specifies the method that the RADIUS probe uses for response verification. Possible values are: `anyAuthResponse`: Indicates any authentication response (accept or reject). `goodAuthResponse`: Indicates an authentication accept. `goodActgResponse`: Indicates a good accounting response. The default setting is `anyAuthResponse`.
`userName` *`text`*	**Optional.** Specifies the user name string applied in this RADIUS probe. The default setting is `unspecified`.
`userPassword` *`text`*	**Optional.** Specifies the password string applied in this RADIUS probe. The default setting is `unspecified`.
`secret` *`text`*	**Optional.** Specifies the server's secret that is used to encrypt data and verify responses. The default setting is `unspecified`.
`optionalRequestAttrs` *`text`*	**Optional.** Specifies the custom list of RADIUS attributes in the form {`id-I` | A |S-*`value`*}. The format is {`id-type-`*`value`*}, where the id is the numeric attribute ID; type is`I` for integer, A for Address (ip), or S for string; and value is the value to be set. **Example:** {`5-I-19`} could be used to specify NAS-Port number 19. {`6-A-1.1.1.1`} could be used to specify the IP address. {`18-S-hello`} could be used to specify the string value.
`nasId` *`text`*	**Optional.** The name used for identifying this switch in RADIUS server requests. The default setting is `Sun Radius Probe`.

Argument name	Description
The following optional arguments are applicable to the RAW_TCP and RAW_UDP probe types.	
`rawMode {anyResponse | specificResponse}`	**Optional.** Specifies the method that the raw TCP or raw UDP probe uses to verify the server health check responses. Possible values are: `anyResponse`: Indicates any response to a raw TCP or UDP query. `specificResponse`: Indicates a matched expected response from the server to a raw TCP or UDP query. The default setting is `anyResponse`.
`rawReqString` *`text`*	**Optional.** Specifies the raw hex bytes used to build the TCP or UDP server health check request. **Note:** The request and response strings can be configured for multiple transactions by using brackets ({}) around the request and response streams. The default setting is 00.
`rawRspString` *`text`*	**Optional.** Specifies the raw hex bytes used for evaluating raw TCP or UDP responses to health check probes. **Note:** The request and response strings can be configured for multiple transactions by using brackets ({}) around the request and response streams. The default setting is 00.
`profileDescription` *`text`*	**Optional.** The textual description for the health check profile. The description can be up to 64 characters. Place the description within double quotes (“ “). The default is “ “.

Argument name	Description
The following optional arguments are applicable to the RTSP probe type.	
`RTSPMode {anyOptionResponse | goodOptionResponse | anyDescribeResponse | goodDescribeResponse}`	**Optional.** Specifies the method that the RTSP probe uses to verify the server health check responses. Possible values are: `anyOptionResponse`: Indicates any reponse from the server to an option request. `goodOptionResponse`: Indicates a valid response from the server to an option request. `anyDescribeResponse`: Indicates any response from a server to a describe request. `goodDescribeResponse`: Indicates a valid response from a server to a describe request.
`rtspUrlName` *`text`*	**Optional.** Specifies the `rtsp` Universal Resource Locator (URL).

Argument name	Description
The following optional arguments are applicable to the SCRIPT probe type. See Appendix B, "TCL usage" for examples of configuring scripted health checks using TCL.	
`scriptMode`	**Optional.** Specifies the method that the SCRIPT probe uses to verify the server health check responses.
`scriptFile` *`text`*	**Optional.** Indicates the file containing commands to be executed for a probe.
`scriptCommands` *`text`*	**Optional.** Indicates commands that will be executed for a probe.
`scriptArg1` *`text`* `scriptArg2` *`text`* `scriptArg3` *`text`* `scriptArg4` *`text`*	**Optional.** Specifies a value for a script variable.
`profileDescription` *`text`*	**Optional.** The textual description for the health check profile. The description can be up to 64 characters. Enter the description within double quotes (" "). The default is " ".

Argument name	Description
The following optional arguments are applicable to the POP3 probe type.	
`POP3Mode {anyServerResponse | goodServerResponse | anyUserResponse | goodUserResponse | anyLoginResponse | goodLoginResponse | anyStatResponse | goodStatResponse}`	**Optional.** Specifies the method that the POP3 probe uses to verify the server health check responses. Possible values are: `anyServerResponse`: Indicates any response from the server to a connection request. `goodServerResponse`: Indicates a valid response from the server to a connection request. `anyUserResponse`: Indicates any user response from the server after a valid connection response. `goodUserResponse`: Indicates a valid user response from the server after a valid connection response. `anyLoginResponse`: Indicates any login response from the server after a valid connection and valid user response. `goodLoginResponse`: Indicates a valid login response from the server after a valid connection and valid user response. `anyStatResponse`: Indicates any stat response from the server after a valid connection and valid login. `goodStatResponse`: Indicates a valid stat response from the server after a valid connection and valid login. The default setting is `anyServerResponse`.
`POP3TerminationMode {fastDisconnet | quitThenDisconnect}`	**Optional.** Specifies the method of ending an POP3 probe session. The default setting is `fastDisconnect`.
`userName` *`text`*	**Optional.** Specifies the user name string applied in this POP3 probe. The default setting is `unspecified`.
`userPassword` *`text`*	**Optional.** Specifies the password string applied in this POP3 probe. The default setting is `unspecified`.

Argument name	Description
The following optional arguments are applicable to the SMTP probe type.	
`SMTPMode {anyServerResponse | goodServerResponse | anyHelloResponse | goodHelloResponse | anyVerifyResponse | goodVerifyResponse | anySendResponse | goodSendResponse}`	**Optional.** Specifies the method that the SMTP probe uses to verify the server health check responses. Possible values are: `anyServerResponse`: Indicates any response from the server to a connection request. `goodServerResponse`: Indicates a valid response from the server to a connection request. `anyHelloResponse`: Indicates any hello response from the server after a valid connection response. `goodHelloResponse`: Indicates a valid hello response from the server after a valid connection response. `anyVerifyResponse`: Indicates any verify response from the server after a valid connection and valid hello response. `goodVerifyResponse`: Indicates a valid verify response from the server after a valid connection and valid hello response. `anySendResponse`: Indicates any send response from the server after a valid connection and valid hello response. `goodSendResponse`: Indicates a valid send response from the server after a valid connection and valid hello response. The default setting is `anyServerResponse`.
`SMTPTerminationMode {fastDisconnet | quitThenDisconnect}`	**Optional.** Specifies the method of ending an SMTP probe session. The default setting is `fastDisconnect`.
`srcName` *`text`*	**Optional.** Specifies the user address string used as the originator for a message.
`dstName` *`text`*	**Optional.** Specifies the user address/name string used to send a message or verify a user.
`msgSubject` *`text`*	**Optional.** Specifies the subject of the message.
`msgBody` *`text`*	**Optional.** Specifies the textual content of the message.

Argument name	Description
The following optional arguments are applicable to the `IMAP4` probe type.	
`IMAP4Mode {anyServerResponse | goodServerResponse | anyCapabilityResponse | goodCapabilityResponse | anyLoginResponse | goodLoginResponse | anyListResponse | goodListResponse | anyExamineResponse | goodExamineResponse}`	**Optional.** Specifies the method that the IMAP4 probe uses to verify the server health check responses. Possible values are: `anyServerResponse`: Indicates any response from the server to a connection request. `goodServerResponse`: Indicates a valid response from the server to a connection request. `anyCapabilityResponse`: Indicates any capability response from the server after a valid connection response. `goodCapabilityResponse`: Indicates a valid capability response from the server after a valid connection response. `anyLoginResponse`: Indicates any login response from the server after a valid connection and valid capability response. `goodLoginResponse`: Indicates a valid login response from the server after a valid connection and valid capability response. `anyListResponse`: Indicates any list response from the server after a valid connection and valid login. `goodListResponse`: Indicates a valid list response from the server after a valid connection and valid login. `anyExamineResponse`: Indicates any examine response from the server after a valid connection, valid login, and valid list response. `goodExamineResponse`: Indicates any valid examine response from the server after a valid connection, valid login, and valid list response. The default setting is `anyServerResponse`.
`IMAP4TerminationMode {fastDisconnet | logoutThenDisconnect}`	**Optional.** Specifies the method of ending an IMAP4 probe session. The default setting is `fastDisconnect`.

Argument name	Description
`userName` *`text`*	**Optional.** Specifies the user name string used for this IMAP4 probe. The default setting is `unspecified`.
`userPassword` *`text`*	**Optional.** Specifies the password string used for this IMAP4 probe. The default setting is `unspecified`.
`mboxName` *`text`*	**Optional.** Specifies the user address string used as the originator for a message. The default setting is `inbox`.

Delete filters

```
no vSwitch-name loadBalance healthCheckProfile
   name text
```

Example

The following examples configure an ICMP and HTTP health check profile. Note the different arguments that are available.

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile name HCP1 type
ICMP ?
   [interval (2..3600)]       Number of seconds between probes (default: 5)
   [retries (0..32)]          Number of failed probes allowed before out of
                              service (default: 0)
   [successRate (0..100)]     Necessary success rate of probes for server to
                              remain up (default: 0)
   [timeout (1..10)]          Number of seconds to wait for probe responses
                              (default: 2)
   [count (1..32)]            Number of successful consecutive probes
                              required for reactivation (default: 3)


sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile name HCP1 type
ICMP interval 25 count 5
```

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile name HCP2 type
HTTP ?
   [httpMode (checkResponseCodes|hash|searchForResponseString)]
                              The mode the HTTP probe runs in
   [requestString <crlfText>] The request string for the HTTP probe to send
   [responseString <text>]    String for probe to search for in server
                              responses
   [interval (2..3600)]       Number of seconds between probes (default: 5)
   [retries (0..32)]          Number of failed probes allowed before out of
                              service (default: 0)
   [successrate (0..100)]     Necessary success rate of probes for server to
                              remain up (default: 0)
   [timeout (1..10)]          Number of seconds to wait for probe responses
                              (default: 2)
   [count (1..32)]            Number of successful consecutive probes
                              required for reactivation (default: 3)
```

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile name HCP2 type
HTTP httpmode hash requestString “GET /default.html HTTP/
1.1\r\nHost:1.1.1.1\r\n\r\n” successRate 80
```

Associated MIB

shc.mib

Web path

- vSwitch → *name* → LoadBalance → healthCheckProfile

healthCheckProfile passive

Purpose

Modifies settings for passive server health checks. Passive probes are out-of-band health checks that skip regularly scheduled health check probes if in-line probes continue to report good server health. Using passive health checks will optimize system performance and reduce the load on the servers.

By default, passive server health checks are OFF. To enable this feature, set the `successiveCount` argument to a number greater than 0. All hosts in the service group must be UP and reporting good health for the passive health checking feature to operate.

Access mode

config

Syntax

To modify a passive health check profile:

```
vSwitch-name loadBalance healthCheckProfile-name passive
   [successiveCount integer]
   [activityThreshold integer]
   [updateTolerance integer]
   [minPollCycles integer]
   [interProcessTimeout integer]
```

Arguments

Argument	Description
`name` *`text`*	The text string that identifies the health check profile.
`successiveCount` *`integer`*	**Optional.** Specifies the maximum number of successive passive probes allowed before forcing at least one normal probe. Once the successive count is reached, at least one normal poll will be executed. The valid range is 0 through 100. If set to 0 (default), passive probes are disabled.

Argument	Description
`activityThreshold` *`integer`*	**Optional.** Specifies the maximum time in seconds since a server last reported good health status. If the activity threshold time expires, normal server health check probes will be used. The valid is range is 6 through 3600 (seconds). The default setting is 15 (seconds).
`updateTolerance` *`integer`*	**Optional.** Specifies the allowable age (length of time) of updates from the fastpath indicating good server health as a result of successful transactions. The valid range is 5 through 60 (seconds). The default setting is 10 seconds.
`minPollCycles` *`integer`*	**Optional.** Specifies the minimum number of normal probes to execute initially before passive probes are used to determine server health. The valid range is 2 through 1000 probes. The default setting is 10 probes.
`interProcessTimeout` *`integer`*	**Optional.** Specifies the length of time in milliseconds to wait for a health check response update from a service group. Set this parameter with the assistance of Sun Microsystems. The valid range is 10 through 2000 (milliseconds). The default setting is 50 (milliseconds).

Example

The following example modifies the passive health check settings on the configured profile named `health_9`.

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile
health_9 passive successiveCount 5 activityThreshold 50
updateTolerance 20 minPollCycles 30 interProcessTimeout 25
```

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → healthCheckProfile → *name* → passive

healthCheckTest

Purpose

Directs a named health check profile toward a named target (real service). Can be used to verify whether the real service is responding as expected to a specific health check before you assign the health check profile, which could remove a server from the load-balancing tables if servers are unresponsive.

Access mode

config

Syntax

```
vSwitch-name loadBalance healthCheckTest
   name text
   realServiceAddress integer
   [realServerPort integer]
   [vRouter text]
```

Arguments

Argument name	Description
`name text`	A text string identifying the health check test. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`realServerAddress ipAddress`	IP address assigned to the real server.
`realServerPort integer`	**Optional.** The port that the host listens over. The default value is 80.
`vRouter text`	**Optional.** The name of the vRouter (in the format vSwitch:vRouter) over which traffic reaches this host.

Example

The following example directs a health check test toward a real service.

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckTest hcl
realServerAddress 1.2.3.4 realServerPort 80
```

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → healthCheckTest

host

Purpose

Creates a host by mapping a user-specified name to a server's IP address. You can also use this command to modify an existing host configuration.

The system uses a specified name to identify the server. You must configure a server with this command (as well as complete additional configuration) before it can be part of the load-balancing process. Each host must have a unique IP address. You can create up to 1024 hosts per vSwitch.

The no form of the command deletes the named host from the host table.

Access mode

config

Syntax

To create a host:

```
vSwitch-name loadBalance host
   name text
   ipAddress ipAddress
   [description text]
   [adminState {enabled | disabled}]
   [vRouter vSwitch:vRouter]
```

To modify a host:

```
vSwitch-name loadBalance host
   name text
   [ipAddress ipAddress]
   [description text]
   [adminState {enabled | disabled}]
   [vRouter vSwitch:vRouter]
```

Arguments

Argument name	Description
`name` *`text`*	Assigns a text string to the host being defined, and maps the name to the server identified by `ipAddress`. If the name specified already exists, the system modifies the configuration for that policy, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`ipAddress` *`ipAddress`*	Specifies the IP address of the server. Each host must have a unique IP address. This argument is required when creating a host, but optional when modifying or deleting one.
`description` *`text`*	**Optional.** Assigns a text description to the host. The description can be up to 64 characters, and is displayed with output from the `show host` command. If the description includes spaces, enclose it within quotation marks.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named host, either `enabled` or `disabled`. Set a state of `disabled` if you want to bring it offline or preconfigure a host before bringing it online. The default administrative state is `enabled`.
`vRouter`	**Optional.** The name of the vRouter (in the format vSwitch:vRouter) over which traffic reaches this host. The default value is the default vRouter for the vSwitch.

Delete filters

```
no vSwitch-name loadBalance host
   name text
   [ipAddress ipAddress]
   [description text]
   [adminState {enabled | disabled}]
   [vRouter vSwitch:vRouter]
```

Example

The following example highlights associating IP addresses with internal named hosts in the load-balance configuration.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# host host1 1.1.1.1
sun(config-vSwitch-e-commerce loadBalance)# host host2 2.2.2.2
sun(config-vSwitch-e-commerce loadBalance)# realService RS1 host1
sun(config-vSwitch-e-commerce loadBalance)# realService RS2 host2
```

Associated MIB

`host.mib`

Web path

- vSwitch → *name* → LoadBalance → host

objectRule

Purpose

An object rule is a named statement that describes how the switch must evaluate an object such as an HTTP request, HTTP response, or Uniform Resource Identifier (URI). An object rule defines a predicate that is compared to incoming or outgoing traffic. If the traffic matches the object rule, a request or response policy will execute a specific action, such as forward or sorry, or apply specific header transformations.

Each object rule requires a name and a predicate.

Note: The system has a default behavior to drop all traffic that is not otherwise matched. If you want to configure a “default” object rule that can be used to match all traffic that is not otherwise matched, you can use the following rule:
`objectRule defaultMatchRule predicate {URI_PATH matches "*"}`

The no form of the command deletes the named object rule from the object rule table.

Access mode

config

Syntax

To create an object rule:

```
vSwitch-name loadBalance objectRule
   name text
   predicate {URI field_name: <operator> [integer | string | keyword]}
```

Arguments

Argument name	Description
`name` *`text`*	Assigns a text string to the object rule being defined. If the name specified already exists, the system modifies the configuration for that policy, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`predicate`	Specifies an HTTP or URI field name. A predicate can examine all or parts of the URI, HTTP header fields, cookies, and other request and response data. It can also perform integer and string comparisons with prefix, suffix, and substring type operations. See Appendix C, “Object rule predicate statements” for a complete description of predicate statements and the predefined set of field names.

Delete filters

```
no vSwitch-name loadBalance objectRule
   name text
   predicate {URI field_name: <operator> [integer | string | keyword]}
```

Example

The following example creates object rules `matchImages` and `matchAll`.

```
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# objectRule matchImages
predicate {URI_PATH matches "/images/*"}
sun(config-vSwitch-e-commerce loadBalance)# objectRule matchAll
predicate {URI_PATH matches "*"}
```

Associated MIB

`op.mib`

Web path

- vSwitch → *name* → LoadBalance → objectRule

outboundNat dynamic

Purpose

Configures outbound dynamic network address translation (NAT) on the N2000 Series. Dynamic NAT translates multiple private addresses to a single public address. This means that one global public address can be used for a range of real IP addresses in the backend network.

Since dynamic NAT maps many backend server IP addresses to a single global address, dynamic NAT must also translate the ephemeral port in each outbound packet and maintain state information for each connection. Dynamic NAT provides more security than static NAT since sessions can only be initiated from the backend network. Sessions initiated from the Internet are dropped.

Dynamic NAT supports TCP and UDP protocols. Once an `outboundNat dynamic` entry is created, at least one outbound NAT host IP range must also be created to complete the dynamic NAT configuration.

Access mode

config

Syntax

To create a dynamic NAT configuration:

```
vSwitch-name loadBalance outboundNat dynamic
   name text
   natIPAddress ipAddress
   [adminState {enabled | disabled}]
   [vRouter vSwitch:vRouter]
```

To modify a dynamic NAT configuration:

```
vSwitch-name loadBalance outboundNat dynamic
   name text
   [natIPAddress ipAddress]
   [adminState {enabled | disabled}]
   [vRouter vSwitch:vRouter]
```

Arguments

Argument name	Description
`name` *`text`*	Creates a dynamic NAT configuration with the specified name. If the name specified already exists, the system modifies the configuration for that service group, as specified by subsequent arguments you enter. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`natIpAddress` *`ipAddress`*	The single public IP address representing configured hosts for outbound packets.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the dynamic address translation capability, either `enabled` or `disabled`. The default administrative status is `enabled`.
`vRouter` *`vrouterName`*	**Optional.** Specifies the name of the virtual router, typically the Internet vRouter, to which the host's vRouter will forward an HTTP response. Enter the name in the format *`vSwitch`*`:`*`vrouterName`*. The default vRouter is `system:shared`.

Delete filters

See the "Arguments" section for the `dynamic` command for argument descriptions.

```
no vSwitch-name loadBalance outboundNat dynamic
   name text
   [natIPAddress ipAddress]
   [adminState {enabled | disabled}]
   [Vrouter vSwitch:vRouter]
```

Example

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadbalance)# outboundNat
sun(config-vSwitch-e-commerce loadBalance outboundNat)# dynamic name
nat1 natIPAddress 206.10.90.1 adminState enabled vRouter system:shared
```

Associated MIB

```
dynNAT.mib
```

Web path

- vSwitch → *name* → LoadBalance → outboundNat → dynamic

outboundNat dynamic hostIpRange

Purpose

Specifies the range of host IP addresses that translate to the configured network address translation (NAT) public IP address when creating a named dynamic NAT configuration.

Access mode

config

Syntax

To create an outbound NAT dynamic host IP range configuration:

```
vSwitch-name loadBalance outboundNat dynamic
   name text
   hostVrouter vSwitch:vRouter
   hostIPRangeList {ipAddress-ipAddress}
```

To modify an outbound NAT dynamic host IP range configuration:

```
vSwitch-name loadBalance outboundNat dynamic
   name text
   [hostVrouter vSwitch:vRouter]
   [hostIPRangeList {ipAddress-ipAddress}]
```

Arguments

Argument name	Description
`name` *`text`*	Specifies the text string that identifies an existing dynamic NAT configuration for which you are configuring a host IP network range.
`hostVrouter` *`vrouterName`*	Specifies the name of the virtual router to which the host will forward an HTTP response. Enter the name in the format *`vSwitch`*`:`*`vrouterName`*.
`hostIPRangeList` `{`*`ipAddress-ipAddress`*`}`	Specifies a list of IP ranges for configured hosts. Enter the range, in dotted-decimal format between braces, using one of the following two methods: 1. With a hyphen representing the span, e.g., {1.1.1.1 - 1.1.1.10, 2.2.2.2 - 2.2.2.10}. 2. As *IP*/*Range*, e.g., {1.1.1.1/255.255.255.0}, which equates to {1.1.1.1-1.1.1.255}.

Delete filters

See the "Arguments" section for the `outboundNat dynamic hostIpRange` command for argument descriptions.

```
no vSwitch-name loadBalance outboundNat dynamic hostIpRange
   name text
   hostVrouter vSwitch:vRouter
   [hostIPRangeList {ipAddress-ipAddress}]
```

Example

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce loadbalance)# outboundNat
...loadBalance outboundNat) dynamic name nat1
...loadBalance outboundNat dynamic-nat1)# hostIpRange hostvRouter
e-commerce:default hostIPRangeList {10.10.80.1-10.10.80.20}
```

Associated MIB

`dynNAT.mib`

Web path

- vSwitch → *name* → LoadBalance → outboundNat → dynamic → hostIpRange

outboundNat static

Purpose

Configures outbound static network address translation (NAT) on the N2000 Series. Static NAT translates private IP addresses to unique public addresses. Static NAT takes outbound traffic initiated from a backend Web server (such as email) and maps the traffic to a global public IP address that masks the server IP address. The N2000 Series performs the address translation using a global address from the configured IP address range before forwarding the traffic to the Internet client. Similarly, when the N2000 Series receives traffic from an Internet client destined to one of the global addresses, it converts the global address to the private address of the backend server.

Since there is a one-to-one mapping configuration between backend vRouter addresses and global addresses, static NAT translates backend outbound requests statically without having to save state information about each TCP connection. Static NAT supports all IP protocols, including TCP, UDP, ICMP, and DNS.

When configuring these addresses, you specify a range that encompasses all the hosts defined in your service group and a range of outbound virtual addresses. Be sure to specify the same number of addresses in each range. Each host address can only be configured in a single static NAT configuration.

The no form of the command deletes the named NAT configuration from the NAT table.

Access mode

config

Syntax

To create a static NAT configuration:

```
vSwitch-name loadBalance outboundNat static
   name text
   hostIPRange {ipAddress-ipAddress}
   natIPRange {ipAddress-ipAddress}
   [hostVRouter vSwitch:vRouter]
   [vRouter vSwitch:vRouter]
   [adminState {enabled | disabled}]
   [clientSrcIPRange {ipAddress-ipAddress}]
```

To modify a static NAT configuration:

```
vSwitch-name loadBalance outboundNat static
   name text
   [hostIPRange {ipAddress-ipAddress}]
   [natIPRange {ipAddress-ipAddress}]
   [hostVRouter vSwitch:vRouter]
   [vRouter vSwitch:vRouter]
   [adminState {enabled | disabled}]
   [clientSrcIPRange {ipAddress-ipAddress}]
```

Arguments

Argument name	Description
`name` *`text`*	Creates a static NAT configuration with the specified name. If the name specified already exists, the system modifies the configuration for that service group, as specified by subsequent arguments you enter. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`hostIPRange` *`ipAddress`*	Specifies the range of real addresses for configured hosts. Enter the range, in dotted-decimal format between braces, using one of the following two methods: 1. With a hyphen representing the span, e.g., {1.1.1.1 - 1.1.1.10, 2.2.2.1-2.2.2.10}. 2. As *IP*/*Range*, e.g., {1.1.1.1/255.255.255.0}, which equates to {1.1.1.1-1.1.1.255}. This argument is required when creating the static NAT configuration, but optional when modifying or deleting it.
`natIPRange` *`ipAddress`*	Specifies the range of virtual addresses mapped to (and representing) configured hosts. Enter Internet-accessible addresses in a range in the format described in the `hostIPRange` argument. If you specify a single address as the outbound address, you cannot specify the IP address of a virtual service address or static NAT will fail. The system performs static NAT only on non-VIP addresses. This argument is required when creating the static NAT configuration, but optional when modifying or deleting it.
`hostvRouter` *`vrouterName`*	**Optional.** Specifies the name of the virtual router to which the backend hosts are connected. Enter the name in the format *`vSwitch`*`:`*`vrouterName`*. The default value is the *default* vRouter on the operator-defined vSwitch.

Argument name	Description
`vRouter` *`vrouterName`*	**Optional.** Specifies the name of the NAT virtual router, typically the Internet vRouter, to which the host's vRouter will forward an HTTP response. Enter the name in the format *`vSwitch`*`:`*`vrouterName`*. The default vRouter is `system:shared`.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the address translation capability, either `enabled` or `disabled`. The default administrative status is `enabled`.
`clientSourceIPRange` *`ipAddress-ipAddress`*	**Optional.** Sets the range of client IP addresses that are allowed access to the backend servers via this static NAT configuration.

Delete filters

```
no vSwitch-name loadBalance outboundNat static
   name text
   [hostIPRange {ipAddress-ipAddress}]
   [natIPRange {ipAddress-ipAddress}]
   [hostVrouter vSwitch:vRouter]
   [vRouter vSwitch:vRouter]
   [adminState {enabled | disabled}]
   [clientSourceIPRange {ipAddress-ipAddress}]
```

Example

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce)# outboundNat
sun(config-vSwitch-e-commerce loadBalance outboundNat)# static name
nat1 hostIPRange {10.10.2.1-10.10.2.254} natIPRange
{206.2.1.1-206.2.1.254} hostVrouter e-commerce:default vrouter
system:shared adminState enabled
```

Associated MIB

```
staticNAT.mib
```

Web path

- vSwitch → *name* → LoadBalance → outboundNat → static

proxyIPPool

Purpose

Configures a pool of IP addresses to be used for client address translation (CAT) and TCP multiplexing. You can create up to 256 pools.

Each pool can have up to 64 IP addresses using a combination of IP ranges (separated with a hyphen character (-)) and individual IP addresses (separated with the semicolon character (;)).

Access mode

config

Syntax

To create a proxy IP address pool configuration:

```
vSwitch-name loadBalance proxyIPPool
   name text
   ipRangeList {ipAddress-ipAddress}
   vRouter vSwitch:vRouter
   [appServiceType {L4SLB | L4SLB_ADV_OR_HTTP_OR_SSL}]
```

To modify a proxy IP address pool configuration:

```
vSwitch-name loadBalance proxyIPPool name
   [ipRangeList {ipAddress-ipAddress}]
   [vRouter vSwitch:vRouter]
   [appServiceType {L4SLB | L4SLB_ADV_OR_HTTP_OR_SSL}]
```

Arguments

Argument name	Description
`name` *`text`*	Specifies the new or existing proxy IP pool name. If the name specified already exists, the system modifies the configuration for that pool. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)

`ipRangeList {`*`ipAddress-ipAddress`*`}`
 	
or
 	
`ipRangeList {`*`ipAddress;ipAddress`*`}`
	Specifies up to 64 proxy IP addresses. Enter a range in dotted-decimal format between braces, using one of the following two methods:
	
	1. With a hyphen representing the span. For example, {1.1.1.1 - 1.1.1.10, 2.2.2.2 - 2.2.2.10}.
	
	2. As *IP*/*Range*, e.g., {1.1.1.1/255.255.255.0}, which equates to {1.1.1.1-1.1.1.255}.
	
	Separate individual IP addresses using the semicolon (;). For example, {1.1.1.1 ;1.1.1.10}

`vRouter` *`vrouterName`*	Specifies the name of the virtual router to which the host is connected. Enter the name in the format *`vSwitch`*`:`*`vrouterName`*. The default setting is the default vRouter in the operator-defined vSwitch.
`appServiceType {L4SLB| L4SLB_ADV_OR_HTTP_OR_SSL}`	**Optional.** The virtual service application service type for which this proxy IP pool is used; either L4SLB (non-terminated TCP), or all other application service types (terminated TCP). The default value is L4SLB.

Delete filters

See the `show proxyIPPool` command for argument descriptions.

```
no vSwitch-name loadBalance proxyIPPool name
   [ipRangeList {ipAddress-ipAddress}]
   [vRrouter vSwitch:vRouter]
   [appServiceType {L4SLB | L4SLB_ADV_OR_HTTP_OR_SSL}]
```

Example

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce loadbalance)# proxyIpPool pool_1
{10.10.80.1-10.10.80.64} e-commerce:default L4SLB
```

Associated MIB

```
proxyIP.mib
```

Web path

- vSwitch → *name* → LoadBalance → proxyIpPool

realService

Purpose

Creates a real service that binds a named host (created with the `host` command) and port to a named service. You can also use this command to modify an existing service configuration. You can configure up to 1024 real services per service group, and up to 1024 per vSwitch.

There are several configurable parameters available with this command, the most important being the protocol this service will use and the port that it will monitor for connections. You can specify multiple ports on each host. The command defaults to protocol of TCP and port number 80.

You cannot configure more than one real service with the same host/port combination.

Determining dynamic weight

The actual weight for a real service is the inverse of the proportion of the real service's average keep-alive latency as compared to the sum of all average keep-alive latencies. This value is determined within the acceptable range for weights, 2^{16}, or 65536. After determining the total latency, the specific real services average latency is subtracted, and its value is determined as a percentage of the whole. This percentage is multiplied by the value the maximum total weight (65536) to determine the dynamic weight.

For example, if you have two real services, RS1 and RS2, and their respective average latencies are 40 ms and 60 ms, then the dynamic weight is determined as follows:

Average latency

RS1 = 40 ms

RS2 = 60 ms

Total average latency

RS1 + RS2 = 100 ms

Total latency (100) minus specific average

RS1: 100 - 40 = 60

RS2: 100 - 60 = 40

Calculation divided by total latency, multiplied by maximum total weight

60 / 100 = 60% x 65536 = 39321

40 / 100 = 40% x 65536 = 26214

In the case of rounding, after the system has calculated these values, it adds the truncated rounding error (in this case 1) to the service with the largest weight. In this example, RS1's weight becomes 39322.

The no form of the command deletes the named service from the realService table.

Access mode

config

Syntax

To create a real service:

```
vSwitch-name loadBalance realService
   name text
   hostName host
   [protocol {TCP |UDP}]
   [port portNumber]
   [weight {dynamic | weight}]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {seconds | unlimited}]
   [ilSHCFailureRateThreshold integer]
   [clientAddressTranslation {enabled | disabled}]
   [encryption {unencrypted | SSL}]
   [proxyIpPool text]
   [bridgemode {enabled | disabled}]
   [persistName text]
   [certName text]
   [certType {CA | literal}]
   [sslProto {SSLv3 | TLSv1}]
   [sslCiphers cipher]
   [reneg {true | false}]
   [resume {true | false}]
```

To modify a real service:

```
vSwitch-name loadBalance realService
   name text
   [hostName host]
   [protocol {TCP |UDP}]
   [port portNumber]
   [weight {dynamic | weight}]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {seconds | unlimited}]
   [ilSHCFailureRateThreshold integer]
   [clientAddressTranslation {enabled | disabled}]
   [proxyIpPool text]
   [bridgemode {enabled | disabled}]
   [persistName text]
   [encryption {unencrypted | SSL}]
   [certName text]
   [certType {CA | literal}]
   [sslProto {SSLv3 | TLSv1}]
   [sslCiphers cipher]
```

```
[reneg {true | false}]
[resume {true | false}]
```

Arguments

Argument name	Description
`name` *`text`*	Creates a real service with the specified name, and maps the name to the host identified by `hostName`. If the name specified already exists, the system modifies the configuration for that real service, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`hostName` *`host`*	Specifies the named host, configured with the `host` command, that is associated with this real service. This argument is required when creating a service, but optional when modifying or deleting one.
`protocol {TCP|UDP}`	**Optional**. Specifies the supported protocol for this real service, TCP or UDP. This setting must match the virtual service protocol configuration being used by the real services. The default setting is TCP.
`port` *`portNumber`*	**Optional.** Specifies the port that the host uses to listen for connections of the type specified with the `protocol` argument. You can assign multiple ports to a host, but each is defined as an individual real service. That is, each real service can only associate a single port with a host. Port 0 is not a valid configuration. The default port is port 80.

Argument name	Description
`weight {dynamic | `*`weight`*`}`	**Optional.** Assigns a weight to the associated host (server), which the system uses in combination with the algorithm (set with the `serviceGroup` command) to determine the number of assigned connections. The larger the weight, the greater the number of connections allocated by the load-balance algorithm. If a server is capable of handling more traffic than other servers in the service group, then set the weight to a higher value than those weights assigned to other servers in the group. Each server receives a percentage of traffic equal to the percentage its weight is of the cumulative weight (for all servers in the service group). You can either set a specific (static) weight or leave it as a dynamic variable. If you have not defined `healthName`, you must set a static weight. A setting of `dynamic` determines a weight calculation based on the server health check latency results. See the “Purpose” section for a detailed explanation of the weight calculation. Valid range is 1 through 65535; the default weight is 1.
`description `*`text`*	**Optional.** Assigns a text description to the real service. This description is displayed with output from the `show realService` command.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named service, either `enabled` or `disabled`. Set a status of `disabled` if you want to preconfigure a real service before bringing it online or disable a running real service. The default administrative status is `enabled`.
`disableDelay {seconds | unlimited}`	**Optional**. Sets the time, in seconds, that must elapse before the system disables a real service when the `adminState` is set to `disabled`. Possible values are 0 to 600 seconds or `unlimited`. The default setting is 0 seconds. If this value is set to non-zero or `unlimited` and the `adminState` is set to `disabled`, the real service does not accept any new connections but current connections are not terminated. At the end of the `disableDelay` time, the system terminates all current connections to the real service.

Argument name	Description
`ilSHCFailureRateThreshold` *integer*	**Optional.** Specifies the number of in-line server health check failures allowed before the real service is brought down. In-line server health checks determine server health without waiting for the next out-of-band polling interval. The default setting is 1. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information on server health checking.
`clientAddressTranslation {enabled | disabled}`	**Optional.** The administrative state of client address translation for this real service configuration, either `enabled` or `disabled`. If enabled, the Internet client's source address and source port information contained in the IP header are translated using a pool of proxy IP addresses configured on the N2000 Series. The default setting is `disabled`. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information about client address translation (CAT) and the proxy IP pool.
`proxyIpPool` *text*	**Optional.** The name of the proxy IP pool to be used when `clientAddressTranslation` is set to `enabled`. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information about client address translation (CAT) and the proxy IP pool.
`bridgeMode {enabled | disabled}`	**Optional.** When enabled, traffic that is sent to the real service will be bridged (forwarded with the client's source address and virtual service destination address). The source and destination addresses will not be modified. The default is `disabled`.
`persistName` *text*	**Optional.** Specifies the persistence name string to match against the HTTP request when the request policy `persistType` setting is the `fieldMatchName` (one possible algorithm used when the system makes session persistence decisions when forwarding HTTP requests).
`encryption {unencrypted | SSL}`	**Optional.** The encryption method for this real service traffic. Possible values are: SSL: SSL is enabled for this real service. `unencrypted`: No encryption is enabled for this real service. The default setting is `unencrypted`.
The following arguments are only applicable if the `encryption` **value is set to** `ssl`	

Argument name	Description
`certName` *text*	**Optional.** Identifies the name of an imported certificate used by the system to verify real service(s). If `certName` is not a valid CKM table entry, no other SSL configuration parameters set through this real service are used. For example, if SSL has been configured at the service group level, that configuration is used.
`certType {CA | literal}`	**Optional.** Sets the SSL authentication type for the real service to either `CA` or `literal`. If set to `literal`, the system only accepts the certificate `certName` during SSL certificate verification. If set to `CA`, the system accepts any valid certificate signed by the key holder of the certificate `certName`. The default is `CA`.
`sslProto {SSLv3 | TLSv1}`	**Optional.** Specifies the SSL protocols supported by the system when connecting to the real service (SSL at the back end). Possible values are: `SSLv3`: Secure Sockets Layer Protocol, Version 3.0 `TLSv1`: Transport Layer Security (TLS) Version 1.0, as defined in RFC 2246. (TLS is also known as SSL, Version 3.1.) The default is to support both `SSLv3` and `TLSv1`.

Argument name	Description

`sslCiphers` *`cipher`*	**Optional.** Specifies the encryption methods (cipher suites) used on traffic passing through this real service. (See RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*, for detailed descriptions of each cipher.) Enter the cipher suites in a list separated by semicolons and enclosed in parentheses. For example:
	
	`(RSA_WITH_RC4_128_MDF;RSA_WITH_RC4_128_SHA)`
	
	Possible supported cipher suites include:
	
	`RSA_EXPORT_WITH_RC4_40_MD5`
	
	`RSA_WITH_RC4_128_MD5`
	
	`RSA_WITH_RC4_128_SHA`
	
	`RSA_EXPORT_WITH_DES40_CBC_SHA`
	
	`RSA_WITH_DES_CBC_SHA`
	
	`RSA_WITH_3DES_EDE_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_DES_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_RC4_56_SHA`
	
	The list of default cipher suites, and those highly recommended for use, is `RSA_WITH_RC4_128_MD5`, `RSA_WITH_RC4_128_SHA`, and `RSA_WITH_3DES_EDE_CBC_SHA`.

`reneg {true | false}`	**Optional.** Sets the system's capability for renegotiating cryptographic parameters over an existing connection to this real service. When set to `true`, the system allows multiple SSL handshakes to occur over an existing SSL connection to the real service. When set to `false`, the system ignores renegotiation requests from the real service. The default setting is `true`.
`resume {true | false}`	**Optional.** Sets the system's capability for doing SSL session resumption when connecting to this real service. When set to `true`, the system attempts to resume a previous SSL session when connecting to the real service, resulting in better performance. When set to `false`, the system initiates a full SSL handshake on all new connections to the real service. The default setting is `true`.

Delete filters

```
no vSwitch-name loadBalance realService
   name text
   [hostName host]
   [protocol {TCP |UDP}]
   [port portNumber]
   [weight {dynamic | weight}]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {seconds | unlimited}]
   [ilSHCFailureRateThreshold integer]
   [clientAddressTranslation {enabled | disabled}]
   [proxyIpPool text]
   [bridgeMode {enabled | disabled}]
   [persistName text]
   [encryption {unencrypted | SSL}]
   [certName text]
   [certType {CA | literal}]
   [sslProto {SSLv3 | TLSv1}]
   [sslCiphers cipher]
   [reneg {true | false}]
   [resume {true | false}]
```

Example

The following example highlights associating a named host with a real service and port.

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vswitch-e-commerce)# loadBalance
sun(config-vSwitch-e-commerce loadBalance)# host host1 1.1.1.1
sun(config-vSwitch-e-commerce loadBalance)# host host2 2.2.2.2
sun(config-vSwitch-e-commerce loadBalance)# realService name RS1
hostName host1 port 80
sun(config-vSwitch-e-commerce loadBalance)# realService name RS2
hostName host2 port 80
```

Associated MIB

```
realSvc.mib
```

Web path

- vSwitch → *name* → LoadBalance → realService

realService advanced

Purpose

Configures Layer 4 properties of the specified real service's TCP/IP sessions. By defining the initial state between the system and the real service, you can fine-tune the TCP connection for performance improvement through resource management.

You must enter the `realService` command with an existing named real service to access the advanced command.

Caution: It is possible to stop all traffic flow to the system by misconfiguring these values. The system comes preconfigured with defaults that serve most TCP connections. You should not change these defaults unless you are confident that you understand TCP operations.

Access mode

config

Syntax

```
vSwitch-name loadBalance realService rsName advanced
   name text
   [tcbTemplateKey 0]
   [ipTOS {Normal |MinCost | MaxReliablity | MaxThroughput |MinDelay}]
   [xmtRetryLimit integer]
   [estRetryLimit integer]
   [shortRxTimer timeoutValue]
   [longRxTimer timeoutValue]
   [rcvWnd integer]
   [xmtRTT rttValue]
   [smmStreamLimit limitValue]
   [estShortTimeout {ExpRFC793 | ExpRetr}]
   [rcvWndDisabled {true | false}]
   [rcvMss (256..1460)]
   [xmtMss (256..1460)]
   [enableHttpMode {true | false}]
   [httpGetPiggyBack {true | false}]
   [rxUseLongTime {true | false}]
```

Arguments

Argument name	Description
`name` *`text`*	Specifies a name of a configured real service. It is the connection between the switch and this real service that is affected by modifications in this command.
`tcbTemplateKey 0`	**Optional.** Specifies the TCB table template key in use by this connection. The default, and only acceptable value, is 0. This argument works in conjunction with troubleshooting commands that are only available through field service support.
`ipTOS {Normal | MinCost | MaxReliablity | MaxThroughput | MinDelay}`	**Optional.** Sets the bit in the type of service (TOS) field of IP headers for traffic traversing this connection. This informs intervening routers how they should handle the packets. Possible values are: `Normal`: Used for ICMP, BOOTP, DNS, TCP queries traffic `MinCost`: Used for Usenet traffic `MaxReliablity`: Used for SNMP and routing traffic `MaxThroughput`: Used for FTP traffic `MinDelay`: Used for interactive login applications, such as Telnet and rlogin. The default setting is `Normal`.
`xmtRetryLimit` *`integer`*	**Optional.** Sets the base interval for the retransmit retry limit within an established connection. That is, the time between retransmits of data sent and not acknowledged. The values are as follows: • `3_seconds` • `9_seconds` • `21_seconds` • `43_seconds` • `91_seconds` • `155_seconds` • `219_seconds` • `283_seconds` • `347_seconds` • `411_seconds` The default setting is `283_seconds`.

Argument name	Description
`estRetryLimit` *integer*	**Optional.** Sets the base interval for the establishment retry limit, which is how long you are willing to attempt a connection. The values are as follows: • `6_seconds` • `30_seconds` • `78_seconds` • `142_seconds` • `206_seconds` • `270_seconds` • `334_seconds` • `398_seconds` • `462_seconds` • `526_seconds` The default setting is `78_seconds`.
`shortRxTimer` *timeoutValue*	**Optional.** Sets the receive timer short time-out value, which defines how long you are willing to wait to receive the first data packet from a real service after a session has been established. If the timer expires, the switch terminates the session. Values are defined to allow for argument completion at the command line. Possible values are: • `disable` • `125_msec` • `250_msec` • `500_msec` • `1_second` • `2_seconds` • `4_seconds` • `8_seconds` • `16_seconds` • `32_seconds` • `64_seconds` • `128_seconds` • `256_seconds` • `512_seconds` • `1024_seconds` • `2048_seconds` The default value is `4_seconds`.

Argument name	Description
`longRxTimer` *`timeoutValue`*	**Optional.** Sets the receive timer long time-out value, which defines how long you are willing to wait to receive subsequent data from the real service after the initial packet. If the timer expires, the switch terminates the session. Values are defined to allow for argument completion at the command line. Possible values are displayed in the `shortRxTimer` argument description, above. The default value is `32_seconds`.
`rcvWnd` *`integer`*	**Optional.** Sets the maximum amount of memory, in bytes, allocated to received data on a session. Valid range is 1024 through 8388608. The default value is 16384 bytes.
`xmtRTT` *`roundTripValue`*	**Optional.** Sets the maximum value for the packet round-trip transmit time in seconds or milliseconds. Possible values are: • `375_msec` • `500_msec` • `625_msec` • `750_msec` • `875_msec` • `1_second` • `1125_msec` • `1250_msec` • `1375_msec` • `1500_msec` The default setting is `750_msec`.
`smmStreamLimit` *`streamLimitValue`*	**Optional.** Sets the Stream Memory Manager multiplier from the Receive Window Size field. Possible values are: • `1xRecvWnd` • `2xRecvWnd` • `4xRecvWnd` • `8xRecvWnd` The default setting is `1xRecvWnd`.
`estShortTimeout {ExRFC793 | ExpRetr}`	**Optional.** Sets the type of Establishment Time time-out; either the value from `ExpRFC793` or `ExpRetr`. The default setting is `ExpRetr`.

Argument name	Description
`rcvWndDisable {true | false}`	**Optional**. Specifies whether the TCP receive flow control is performed on a per-connection basis and advertised to the sender. The default setting is `false`.
`rcvMss` *`integer`*	**Optional**. Specifies the receive maximum segment size; broadcasted with a TCP SYN segment. The default setting is 1460.
`xmtMss` *`integer`*	**Optional**. Specifies the maximum segment size; advertised during the SYN handshake. The default setting is 1460.
`enableHttpMode {true | false}`	**Optional**. Enables or disables HTTP mode. This is a performance tuning parameter that optimizes HTTP. The default setting is `false`.
`httpGetPiggyBack {true | false}`	**Optional**. Specifies whether TCP GET requests are piggybacked on the last portion of 3-way handshake. This parameter only applies to server side connections. The default setting is `true`.
`rxUseLongTime {true | false}`	**Optional**. Specifies whether the internal receive progress timer is used to recover dormant connections. The default setting is `true`.

Delete Filters

```
no vSwitch-name loadBalance realService rsName advanced
   name text
   [tcbTemplateKey 0]
   [ipTOS {Normal |MinCost | MaxReliablity | MaxThroughput |MinDelay}]
   [xmtRetryLimit integer]
   [estRetryLimit integer]
   [shortRxTimer timeoutValue]
   [longRxTimer timeoutValue]
   [rcvWnd integer]
   [xmtRTT rttValue]
   [smmStreamLimit limitValue]
   [estShortTimeout {ExpRFC793 | ExpRetr}]
   [rcvWndDisabled {true | false}]
   [rcvMss (256..1460)]
   [xmtMss (256..1460)]
   [enableHttpMode {true | false}]
   [httpGetPiggyBack {true | false}]
   [rxUseLongTime {true | false}]
```

Example

The following example defines two new real services and changes advanced characteristics of one.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# realService name RS3
hostName host3
sun(config-vSwitch-e-commerce loadBalance)# realService RS3 advanced
ipTos mincost rcvWnd 8000
```

Associated MIB

```
realSvc.mib
```

Web path

- vSwitch → *name* → LoadBalance → realService → advanced

realService ssl

Purpose

Modifies the Secure Sockets Layer (SSL) properties of a specified real service's TCP/IP session. You can also use this command to modify the real service properties set with the `realService` command. See the `realService` command for more information.

The no form of the command deletes the specified real service.

Access mode

config

Syntax

```
vSwitch-name loadBalance realService rsName ssl
   hostName host
   [protocol {TCP |UDP}]
   [port portNumber]
   [weight {dynamic | weight}]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {seconds | unlimited}]
   [ilSHCFailureRateThreshold integer]
   [clientAddressTranslation {enabled | disabled}]
   [proxyIpPool text]
   [bridgemode {enabled | disabled}]
   [persistName text]
   [encryption {unencrypted | SSL}]
   [certName text]
   [certType {CA | literal}]
   [sslProto {SSLv3 | TLSv1}]
   [sslCiphers cipher]
   [reneg {true | false}]
   [resume {true | false}]
```

Arguments

Argument name	Description
`hostName` *`host`*	Specifies the named host, configured with the `host` command, that is associated with this real service.
`protocol {TCP | UDP}`	**Optional**. Specifies the supported protocol for this real service, TCP or UDP. This setting must match the virtual service protocol configuration being used by the real services. The default setting is TCP.
`port` *`portNumber`*	**Optional**. Specifies the port that the host uses to listen for connections of the type specified with the `protocol` argument. You can assign multiple ports to a host, but each is defined as an individual real service. That is, each real service can only associate a single port with a host. Port 0 is not a valid configuration. The default port is port 80
`weight {dynamic |` *`weight`*`}`	**Optional.** Assigns a weight to the associated host (server), which the system uses in combination with the algorithm (set with the `serviceGroup` command) to determine the number of assigned connections. The larger the weight, the greater the number of connections allocated by the load-balance algorithm. If a server is capable of handling more traffic than other servers in the service group, then set the weight to a higher value than those weights assigned to other servers in the group. Each server receives a percentage of traffic equal to the percentage its weight is of the cumulative weight (for all servers in the service group). You can either set a specific (static) weight or leave it as a dynamic variable. If, with the `serviceGroup` command, you have set the `loadBalanceType` to `srcAddress` or you have not defined `healthName`, you must set a static weight. A setting of `dynamic` determines a weight calculation based on the server health check latency results. See the command description for a detailed explanation of the weight calculation. Valid range is 1 through 65535. The default weight is 1.
`description` *`text`*	**Optional**. Assigns a text description to the real service. This description is displayed with output from the `show realService` command.

Argument name	Description
`adminState {enabled | disabled}`	**Optional**. Sets the administrative state of the named service, either `enabled` or `disabled`. Set a status of `disabled` if you want to preconfigure a real service before bringing it online or disable a running real service. The default administrative status is `enabled`.
`disableDelay {`*`integer`*` | unlimited}`	**Optional**. The time, in seconds, that must elapse before the system disables a real service when the `adminState` is set to `disabled`. Possible values are 0 to 600 seconds or `unlimited`. The default setting is 0 seconds. If this value is set to non-zero or `unlimited` and the `adminState` is set to `disabled`, the real service does not accept any new connections but current connections are not terminated. At the end of the `disableDelay` time, the system terminates all current connections to the real service.
`ilSHCFailureRateThreshold` *`integer`*	**Optional.** Specifies the number of in-line server health check failures allowed before the real service is brought down. In-line server health checks determine server health without waiting for the next out-of-band polling interval. The default setting is 1. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information on server health checking.
`clientAddressTranslation {enabled | disabled}`	**Optional.** The administrative state of client address translation for this real service configuration, either `enabled` or `disabled`. If enabled, the Internet client's source address and source port information contained in the IP header are translated using a pool of proxy IP addresses configured on the N2000 Series. The default setting is `disabled`. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information on client address translation (CAT) and the proxy IP pool.
`proxyIpPool` *`text`*	**Optional.** The name of the proxy IP pool to be used when `clientAddressTranslation` is set to `enabled`. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information on client address translation (CAT) and the proxy IP pool.
`bridgeMode {enabled | disabled}`	**Optional.** When enabled, traffic that is sent to the real service will be bridged (forwarded with the client's source address and virtual service destination address). The source and destination addresses will not be modified. The default is `disabled`.

Argument name	Description
`persistName` *text*	**Optional.** Specifies the persistence name string to match against the HTTP request when the forwarding policy `persistType` setting is the `fieldMatchName` (one possible algorithm used when the system makes session persistence decisions when forwarding HTTP requests).
`encryption {unencrypted | SSL}`	**Optional.** The encryption method for this real service traffic. Possible values are: SSL: SSL is enabled for this real service. `unencrypted`: No encryption is enabled for this real service. The default setting is `unencrypted`.
`certName` *text*	**Optional.** Identifies the name of an imported certificate used by the system to verify real service(s). If `certName` is not a valid CKM table entry, no other SSL configuration parameters set through this real service are used. For example, if SSL has been configured at the service group level, that configuration is used.
`certType {CA | literal}`	**Optional.** Sets the SSL authentication type for the real service to either CA or `literal`. If set to `literal`, the system only accepts the certificate `certName` during SSL certificate verification. If set to CA, the system accepts any valid certificate signed by the key holder of the certificate named in `certName`. The default is CA.
`sslProto {SSLv3 | TLSv1}`	**Optional.** Specifies the SSL protocols supported by the system when connecting to the real service (SSL at the back end). Possible values are: SSLv3: Secure Sockets Layer Protocol, Version 3.0 TLSv1: Transport Layer Security (TLS) Version 1.0, as defined in RFC 2246. (TLS is also known as SSL, Version 3.1.) The default is to support both SSLv3 and TLSv1.

Argument name	Description

`sslCiphers` *cipher*	**Optional.** Specifies the encryption methods (cipher suites) used on traffic passing through this real service. (See RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*, for detailed descriptions of each cipher.) Enter the cipher suites in a list separated by semicolons and enclosed in parentheses. For example:
	
	`(RSA_WITH_RC4_128_MDF;RSA_WITH_RC4_128_SHA)`
	
	Possible supported cipher suites include:
	
	`RSA_EXPORT_WITH_RC4_40_MD5`
	
	`RSA_WITH_RC4_128_MD5`
	
	`RSA_WITH_RC4_128_SHA`
	
	`RSA_EXPORT_WITH_DES40_CBC_SHA`
	
	`RSA_WITH_DES_CBC_SHA`
	
	`RSA_WITH_3DES_EDE_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_DES_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_RC4_56_SHA`
	
	The default list of cipher suites, and those highly recommended for use, is `RSA_WITH_RC4_128_MD5`, `RSA_WITH_RC4_128_SHA`, and `RSA_WITH_3DES_EDE_CBC_SHA`.

`reneg {true | false}`	**Optional.** Sets the system's capability for renegotiating cryptographic parameters over an existing connection to this real service. When set to `true`, the system allows multiple SSL handshakes to occur over an existing SSL connection to the real service. When set to `false`, the system ignores renegotiation requests from the real service. The default setting is `true`.
`resume {true | false}`	**Optional.** Sets the system's capability for doing SSL session resumption when connecting to this real service. When set to `true`, the system attempts to resume a previous SSL session when connecting to the real service, resulting in better performance. When set to `false`, the system initiates a full SSL handshake on all new connections to the real service. The default setting is `true`.

Delete filters

```
no vSwitch-name loadBalance realService rsName ssl
   [hostName host]
   [protocol {TCP |UDP}]
   [port portNumber]
   [weight {dynamic | weight}]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {seconds | unlimited}]
   [ilSHCFailureRateThreshold integer]
   [clientAddressTranslation {enabled | disabled}]
   [proxyIpPool text]
   [bridgemode {enabled | disabled}]
   [persistName text]
   [encryption {unencrypted | SSL}]
   [certName text]
   [certType {CA | literal}]
   [sslProto {SSLv3 | TLSv1}]
   [sslCiphers cipher]
   [reneg {true | false}]
   [resume {true | false}]
```

Example

The following example highlights modifying SSL properties for a real service.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# realService name RS3 ssl
port 8080 weight dynamic certName RSA55 certType CA sslProto SSLv3
```

Associated MIB

```
realSvc.mib
```

Web path

- vSwitch → *name* → LoadBalance → realService → ssl → modify
- vSwitch → *name* → LoadBalance → realService → ssl → delete

requestPolicy

Purpose

Defines the action of all traffic that matches the named object rule.

- If the action is set to `forward`, all HTTP traffic from the client that matches the named object rule will be sent to the specified service group for load balancing.
- If the action is set to `sorry`, all matching traffic can either close the connection, reset the connection, redirect the request to another URI, or send back a predefined HTML page. (You define the `sorry` action with the `sorryData` command.)

A request policy also defines the session persistence type (“stickiness”) to maintain session persistence to real services.

Each request policy has an ordered precedence with respect to the other configured request policies within a virtual service. The request policy that matches the request and has the highest precedence is selected. When a request is received, the request policies are evaluated in decreasing-precedence order. The first policy with a matching predicate determines how the request will be processed. If no matching policy is found, the session on which the request was received is reset, and a “no policy match counter” is incremented.

The `no` form of the command deletes the named request policy from the request policy table.

Access mode

config

Syntax

```
vSwitch-name loadBalance requestPolicy
   name text
   action [forward | sorry]
   objectRule text
   [serviceGroupName text]
   [sorryData text]
   [precedence integer]
   [persistType {none | fieldHash | fieldMatchName | srcAddress |
      switchCookie}]
   [cookiePersist text]
   [fieldPrefix text]
   [srcAddressMask ipaddress]
   [optimizeLastResponse {enabled | disabled}]
   [usePooledConnections {enabled | disabled}]
   [firstObjectSwitching {enabled | disabled}]
```

Arguments

Argument name	Description
`name text`	Assigns a text string to the request policy being defined. If the name specified already exists, the system modifies the configuration for that policy, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)

Argument name	Description
`action [forward | sorry]`	Specifies the action the load balancer should take when it finds a match to the named object rule. Possible actions are: `forward`: Allows the HTTP request to pass through to the appropriate server. `sorry`: Executes a specific sorry action. (Do not specify a service group for a sorry action.) Possible sorry actions include: `Close`: Gracefully ends the TCP connection with 4-way handshake and FIN. `Page`: Returns an HTML page to the client. `Redirect`: Returns an HTTP 302 redirect response redirecting the request to a different URI. `Reset`: Returns a reset to the client.
`objectRule` *`text`*	Identifies the name of an object rule, which describes evaluation criteria of an incoming object such as an HTTP Request or URI. If the incoming object matches the criteria of the named object rule, the action specified with the `action` argument of the `requestPolicy` command will be executed. The object rule name is defined with the `name` argument of the `objectRule` command. You can only assign one rule per request policy.
`serviceGroup` *`text`*	**Optional.** Identifies the name of the service group affected by this policy if the policy `action` is `forward`. This is the name that was assigned with the `name` argument of the `serviceGroup` command.
`sorryData` *`text`*	**Optional**. Specifies the sorry data associated with this policy. This field is required if the policy `action` is `sorry`; empty string for none.
`precedence` *`integer`*	**Optional**. Sets the precedence of the request policy, that is, the order in which the system considers the actions configured within it. If multiple policies exist with the same precedence, they are evaluated in the order in which they were configured. Valid range is 1 through 2,147,483,647 (a signed 32-bit integer). The default value is 1 (highest precedence).

Argument name	Description
`persistType {none | fieldHash | fieldMatchName |srcAddress | switchCookie}`	**Optional.** Specifies the algorithm used when the system makes session persistence decisions when forwarding HTTP requests. The options are `none`, `fieldHash`, `fieldMatchName`, `srcAddress`, and `switchCookie`. `fieldHash`: Provides support to DoCoMo network I-mode operation where persistence based on a URI_QUERY field is desired. Persistence is provided by hashing across the URI_QUERY field specified in the field prefix. `fieldMatchName`: Provides support to DoCoMo network I-mode operation when deterministic persistence based on a URI_QUERY field is desired. Persistence is provided by parsing the URI_QUERY field specified in the field prefix and using the `persistName` specified for each real service. `srcAddress`: Runs a hash algorithm across the client browser's source IP address (or the significant bits if a mask was assigned with the `sourceAddressMask` argument). The "sticky by source" assignment is maintained through all future sessions unless you modify the configuration. The following modifications will disrupt persistence: 1. Changes to the `loadBalanceType`. 2. Changes to the `serviceGroup` (such as adding and deleting real servers). 3. Changes to the weighting of a real service. For equal weighting of each real service, assign an equal weight to each real service, or keep the default weight setting of 1. To apply double weight to a particular server to double the IP address space over which it is sticky, apply a double weight with respect to the other real servers in the service group. `switchCookie`: Uses switch-inserted cookies to persist users to services. You must set the cookie name with the `cookiePersist` argument.
`cookiePersist` *`text`*	**Optional.** Specifies the name of the cookie to be inserted into forwarded packets, from the cookie persistence table. You configure the parameters of the cookie with the `cookiePersistence` command.

Argument name	Description
`fieldPrefix` *`text`*	**Optional.** Specifies the string used to delineate the field in the HTTP request when the persistence type is `fieldHash` or `fieldMatchName`. If the URI_QUERY is expected to be of the format `ide=abcdefg01` then `fieldPrefix` would be `ide`.
`srcAddressMask` *`ipaddress`*	**Optional.** Specifies the mask to apply to the IP address retrieved with the `srcAddress` value of the `loadBalanceType` command. The default mask is 255.255.255.255.
`optimizeLastResponse {enabled | disabled}`	**Optional**. Optimizes the data path after the client sends a close request. The default setting is `disabled`.
`usePooledConnections {enabled | disabled}`	**Optional**. Maintains a pool of connections to real services when enabled. The default setting is `disabled`.
`firstObjectSwitching {enabled | disabled}`	**Optional**. Sets the method of load-balance processing of client requests in a single TCP session. When disabled, the system performs load balancing for each client request in a persistent HTTP session. If the request results in a different service group assignment, the system initiates a new TCP session. The default setting is `disabled`. When enabled, all requests in a single TCP session are sent to the same real service. This lessens the granularity of the load-balancing function, but can speed processing by simplifying load-balancing decisions. Enabling `firstObjectSwitching` stops the following operations: • Switch cookie insertion • Custom HTTP header rewriting • SSL cipher additions in the HTTP `headerAll` response policy processing • All response policy processing • All request transform processing • All response transform processing

Delete filters

```
no vSwitch-name loadBalance requestPolicy
   name text
   action [forward | sorry]
   objectRule text
   [serviceGroupName text]
   [sorryData text]
   [precedence integer]
   [persistType {none | fieldHash | fieldMatchName | srcAddress |
      switchCookie}
   [cookiePersist text]
   [fieldPrefix text]
   [srcAddressMask ipaddress]
   [optimizeLastResponse {enabled | disabled}]
   [usePooledConnections {enabled | disabled}]
   [firstObjectSwitching {enabled | disabled}]
```

Example

The following example shows how to assign a request policy in the load-balance configuration.

```
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# requestPolicy images
action forward objectRule matchImages serviceGroup imageServers4
```

Associated MIB

```
SUN-APP-SWITCH-REQUEST-POLICY-MIB.mib
```

Web path

- vSwitch → *name* → LoadBalance → requestPolicy

requestTransform

Purpose

Defines changes to the HTTP header text in an HTTP request. The N2000 Series will apply a request transform to an incoming request from a client if the object rule matches the HTTP traffic received from the client and the name of the request transform is included in the virtual service definition.

The request transform supports header insertions, such as headers that include the original source IP address, and SSL cipher headers.

Access mode

config

Syntax

```
vSwitch-name loadBalance requestTransform
   name text
   objectRule text
   [customHTTPHeader text]
   [insertSSLCipher {enabled | disabled}]
   [SSLCipherHeaderName text]
   [insertSourceIP {enabled | disabled}]
   [sourceIPHeaderName text]
   [deleteHTTPHeader text]
```

Arguments

Argument name	Description
`name` *`text`*	Assigns a text string to the request transform being defined. If the name specified already exists, the system modifies the configuration for that transform, as specified by subsequent arguments that you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`objectRule` *`text`*	Identifies the name of an object rule, which describes evaluation criteria of an incoming object, such as an HTTP request. If the incoming object matches the criteria of the named object rule, the HTTP header changes defined by the arguments of the `requestTransform` command will be applied to the incoming object. The object rule name is defined with the `name` argument of the `objectRule` command. You can assign only one rule per request transform.
`custom HTTPHeader` *`text`*	**Optional.** Specifies HTTP header text that will be inserted into all forwarded requests.
`insertSSLcipher {enabled | disabled}`	**Optional.** If enabled, inserts independent header line conveying the SSL Cipher suite of the client session into each forwarded request. This parameter is valid only if the `appServiceType` is `HTTPS`. The default setting is `disabled`.
`SSLCipherHeaderName` *`text`*	**Optional.** When `insertSSLCipher` is enabled, this is the name of the HTTP header used to send the SSL cipher information.
`insertSourceIP {enabled | disabled}`	**Optional.** Inserts an independent header line conveying the source IP address of the client into each forwarded request. The default setting is `disabled`.
`sourceIPHeaderName` *`text`*	**Optional.** When `insertSourceIP` is enabled, this is the name of the HTTP header used to send the client source IP address field.
`deleteHTTPHeader` *`text`*	**Optional.** Deletes an HTTP header from all forwarded requests.

Delete filters

```
no vSwitch-name loadBalance requestTransform
   name text
   objectRule text
   customHTTPHeader text
   [insertSSLCipher {enabled | disabled}]
   [SSLCipherHeaderName text]
   [insertSourceIP {enabled | disabled}]
   [sourceIPHeaderName text]
   [deleteHTTPHeader text]
```

Example

The following example shows how to assign a request transform in the load-balance configuration.

```
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# requestTransform rqt1
matchall insertSSLCipher enabled SSLCipherHeaderName XaSslClientCipher
insertSourceIP enabled sourceIPHeaderName X-IP
```

Associated MIB

```
SUN-APP-SWITCH-REQUEST-TRANSFORM-MIB.mib
```

Web path

- vSwitch → *name* → LoadBalance → requestTransform

responsePolicy

Purpose

Provides control over errors returned by real service hosts in backend service groups. The object rule specified for the response policy defines comparison criteria for objects returned by servers. If the returned object matches the criteria defined by the named object rule, the action specified with the `action` argument of the `responsePolicy` command will be executed. If no matching object rule criteria are found, the response defaults to return the traffic to the client.

For example, a backend Web server may return “Error 404 Page Not Found” in response to a client request for a Web page. You can define an object rule to match “Error 404” and configure a response policy to return the object to the host server, retry the original request to the host server, or perform the action assigned in `sorryData`.

Indirectly, response policies can also serve as another in-line server health check probe.

The `no` form of the command deletes the named response policy from the response policy table.

Access mode

config

Syntax

```
vSwitch-name loadBalance responsePolicy
   name text
   action [retry | return | sorry]
   objectRule text
   [sorryData text]
   [precedence integer]
   [overrideCookieInsert {enabled | disabled}]
   [inlineHealthCheckFailure {enabled | disabled}]
```

Arguments

Argument name	Description
`name` *`text`*	Assigns a text string to the response policy being defined. If the name specified already exists, the system modifies the configuration for that policy, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`action [retry | return | sorry]`	Identifies the action the load balancer should take when it finds a match to the named object rule. Possible actions are: `retry`: Attempts to retry the request to a different server in the service group. `return`: Allows the HTTP response to pass through to the appropriate client. `sorry`: Performs the action assigned in `sorryData`.
`objectRule` *`text`*	Identifies the name of an object rule, which describes evaluation criteria of an object returned by host servers. If the returned object matches the criteria of the named object rule, the action specified with the `action` argument will be executed. The object rule name is defined with the `name` argument of the `objectRule` command. You can only assign one rule per request policy.
`sorryData` *`text`*	**Optional**. Specifies the sorry data associated with this policy if the `action` is `sorry`; empty string for none.
`precedence` *`integer`*	**Optional**. Specifies the precedence of the response policy, that is, the order in which the system considers the actions configured within it. If multiple policies exist with the same precedence, they are evaluated in the order in which they were configured. The default value is 1 (highest precedence).

Argument name	Description
`overrideCookieInsert {enabled | disabled}`	**Optional**. If enabled, the `set-Cookie` field will not be put into the response. The default value is `disabled`.
`inlineHealthCheckFailure {enabled | disabled}`	**Optional**. When enabled, treats a response as an inline health check failure. The default value is `disabled`.

Delete filters

```
no vSwitch-name loadBalance responsePolicy
   name text
   action [retry | return | sorry]
   objectRule text
   [sorryData text]
   [precedence integer]
   [overrideCookieInsert {enabled | disabled}]
   [inlineHealthCheckFailure {enabled | disabled}]
```

Example

The following example shows how to assign a response policy in the load-balance configuration.

```
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# responsePolicy resp1
action retry objectRule matchImages precedence 1
```

Associated MIB

`SUN-APP-SWITCH-RESPONSE-POLICY-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → responsePolicy

responseTransform

Purpose

Defines changes to be made to an HTTP header in an HTTP response. These changes will be made to an outgoing response from the server if the object rule matches the response received from the server.

The response transform supports server cloaking, which is a removal of the server signature from HTTP headers.

Access mode

config

Syntax

```
vSwitch-name loadBalance responseTransform
   name text
   objectRule text
   [rewriteRedirects {enabled | disabled}]
   [rewritePort {integer | all}]
   [customHTTPHeader text]
   [deleteHTTPHeader text]
```

Arguments

Argument name	Description
`name` *`text`*	Assigns a text string to the response transform being defined. If the name specified already exists, the system modifies the configuration for that transform, as specified by subsequent arguments you enter. If the name does not exist, the system creates it. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`objectRule` *`text`*	Identifies the name of an object rule, which describes evaluation criteria of an object returned by host servers. If the returned object matches the criteria of the named object rule, the changes specified with the arguments of the `responseTransform` command will be applied to the returned object.
`rewriteRedirects {enabled | disabled}`	**Optional.** If traffic for an SSL session is redirected, indicates whether the header is rewritten to keep the client session private. The default value is `disabled`.
`rewritePort {`*`integer`* `| all}`	**Optional.** Indicates the ports on which the system performs redirect rewrites. The default value is `all`.
`custom HTTPHeader` *`text`*	**Optional.** Specifies HTTP header text that will be inserted into each returned response.
`deleteHTTPHeader` *`text`*	**Optional.** Deletes an HTTP header from each returned response.

Delete filters

```
no vSwitch-name loadBalance responseTransform
   name text
   objectRule text
   [rewriteRedirects {enabled | disabled}]
   [rewritePort {integer | all}]
   [customHTTPHeader text]
   [deleteHTTPHeader text]
```

Example

The following example shows how to assign a response transform in the load-balance configuration.

```
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# responseTransform rst1
objectRule matchImages rewriteRedirects enabled
```

Associated MIB

```
SUN-APP-SWITCH-RESPONSE-TRANSFORM-MIB.mib
```

Web path

- vSwitch → *name* → LoadBalance → responseTransform

serviceGroup

Purpose

Creates a service group and assigns real services (created with the `realService` command) to that group. This command groups servers, assigns them a particular load-balancing algorithm, and optionally, sets other configurable characteristics. You can create up to 512 service groups.

You can also use this command to modify an existing service group configuration.

The `no` form of the command deletes the named service group from the service group table.

Choosing a load-balance type

The N2000 Series supports a variety of load-balancing algorithms to disperse traffic across the available real services. They are:

- `roundRobin`: The system assigns requests to each active real service (server) in turn, one per service. When all servers have been assigned, the system begins again with the first.
- `weightedRandom`: The system provides a pseudo-random distribution across service groups, according to the weights set.
- `weightedHash`: The system runs a hash algorithm across the source IP address and source TCP port to distribute traffic evenly across a service group. During heavy traffic periods, the weighted hash algorithm uses the real service's weight setting to see where it can distribute more (or less) traffic. This algorithm can only be used if the `appServiceType` argument of the `virtualService` command is set to `L4SLB`.
- `leastConnections`: The system makes load-balancing decisions based on which server has the fewest connections.

Load-balance metric

The system uses a lowest latency algorithm in weight calculations if the weight is set to `dynamic` in the `realService` command. This setting is ignored if a static value is configured for the weight.

Lowest latency computes the response time to and from a server, and uses that value to determine the current fastest real service. The lowest latency value is calculated using the Exponentially Weighted Moving Average (EWMA). This algorithm determines the delta between the keep-alive latency experienced with this check, divided by 2^5 (32), to the current average latency.

Access mode

config

Syntax

To create a service group:

```
vSwitch-name loadBalance serviceGroup
   name text
   loadBalanceType {roundRobin | weightedRandom | weightedHash |
      leastConnections}
   cfgRealServices text
   [standbyRealServices textt]
   [adminState {enabled | disabled}]
   [healthName text]
   [inlineHealthCheck {enabled | disabled}]
   [retryCount {0 | 1 | 2}]
   [responsePolicyList text]
   [responseTransformList text]
```

To modify a service group:

```
vSwitch-name loadBalance serviceGroup name text
   [loadBalanceType {roundRobin | weightedRandom | weightedHash |
      leastConnections}]
   [cfgRealServices text]
   [standbyRealServices text]
   [adminState {enabled | disabled}]
   [healthName text]
   [inlineHealthCheck {enabled | disabled}]
   [retryCount {0 | 1 | 2}]
   [responsePolicyList text]
   [responseTransformList text]
```

Arguments

Argument name	Description
`name` *`text`*	Creates a service group with the specified name, and maps the named real services, identified by the `cfgRealServices` argument, to that service group. If the name specified already exists, the system modifies the configuration for that service group, as specified by subsequent arguments you enter. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`loadBalanceType {roundRobin | weightedRandom | weightedHash | leastConnections}`	Specifies the algorithm by which the system makes load-balancing decisions. See “Choosing a load-balance type” at the beginning of this command description for a full explanation of the types. This argument is required when creating a service group, but optional when modifying or deleting it.
`cfgRealServices` *`text`*	Specifies the list of real services (servers) that the system groups and makes load-balancing decisions across. Enter the real services in a space-separated list enclosed in braces { }. This argument is required when creating a service group, but optional when modifying or deleting it.
`standbyReal Services` *`text`*	The list of real services (servers) that will be used for load balancing if any of the `cfgRealServices` are unavailable.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named service group, either `enabled` or `disabled`. Set a status of `disabled` if you want to preconfigure a service group before bringing it online or disable a running service group. The default administrative status is `enabled`.

Argument name	Description
`healthName` *`text`*	**Optional**. Specifies the name of the health check profile to use with the real services (hosts) of this service group. Enter the profile you have configured, identified by the `name` argument of the `healthCheckProfile` command. If no profile is defined, be certain to set static weights for the real services (with the `weight` argument of the `realServices` command). There is no profile enabled until you specify one with this argument.
`inlineHealthCheck {enabled | disabled}`	**Optional**. Specifies whether the system uses in-band health checks to determine server health without waiting for the next out-of-band polling interval. The default is `disabled`.
`retryCount {0 | 1 | 2}`	**Optional.** Specifies the number of attempts the switch should make to connect to a different real service (host) within the same service group before failover. For L4SLB, the system tries only once and sends back a reset if the request fails. For all other types, a reset is sent back to the client after the number of retries is executed. The default number of retries is 1.
`responsePolicyList` *`text`*	**Optional.** Specifies the list of response policies that will be applied to responses received from real services. A response policy defines an action to take if the response matches the object rule. This list may be empty, in which case the server response will be returned to the client.
`responseTransformList` *`text`*	**Optional.** Specifies the list of response transforms that will be applied to responses received from real services. A response transform defines specific changes to make to the HTTP header in the response if the response matches the object rule. This list may be empty.

Delete filters

```
no vSwitch-name loadBalance serviceGroup name text
   [loadBalanceType {roundRobin | weightedRandom | weightedHash |
      leastConnections}]
   [cfgRealServices text]
   [standbyRealServices text]
   [adminState {enabled | disabled}]
   [healthName text]
   [inlineHealthCheck {enabled | disabled}]
   [retryCount {0 | 1 | 2}]
   [responsePolicyList text]
   [responseTransformList text]
```

Example

The following example highlights associating real services (hosts) with service groups in the load-balance configuration.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# host host1 1.1.1.1
sun(config-vSwitch-e-commerce loadBalance)# host host2 2.2.2.2
sun(config-vSwitch-e-commerce loadBalance)# realService name RS1
hostName host1 port 8080
sun(config-vSwitch-e-commerce loadBalance)# realService name RS2
hostName host2 port 8080
sun(config-vSwitch-e-commerce loadBalance)# serviceGroup SG1
loadBalanceType roundrobin cfgRealServices {RS1 RS2}
```

Associated MIB

```
svcGrp.mib
```

Web path

- vSwitch → *name* → LoadBalance → serviceGroup

show cookiePersistence

Purpose

Displays the persistence rule configuration for a specified cookie or all cookies stored on the system. These settings were configured with the `cookiePersistence` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance cookiePersistence`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show cookiePersistence
Name:                    PersistRule1cookiename
Cookie Name:             cookie123
Cookie Domain:           www.host1.com
Cookie Path:             /shop
Cookie Expires:
Secure:                  false

Name:                    rule1
Cookie Name:             nncookie123
Cookie Domain:           www.host1.com
Cookie Path:             /
Cookie Expires:
Secure:                  false

Name:                    test
Cookie Name:             cookie1
Cookie Domain:
Cookie Path:             /
Cookie Expires:
Secure:                  false
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the named persistence rule.	`name` *`text`*
`Cookie Name`	The actual string inserted into the packet header. The cookie value is equal to the hash key that the switch generated from the real service name, concatenated with the values, either user-specified or default, of `cookieDomain` and `cookiePath`.	`cookieName` *`text`*
`Cookie Domain`	The domain attribute to compare against the domain portion of the host's fully qualified domain name (FQDN), either user-specified or server-generated.	`cookieDomain` *`text`*
`Cookie Path`	The path name, which, if the request has passed name and domain matching, the switch compares against the URL path attribute, either user-specified or switch-generated.	`cookiePath` *`text`*
`Cookie Expires`	The time and date string that the client compares to the time and date in the header to see whether the cookie is still valid. Delta time can be qualified by entering days, hours, minutes, or seconds following the number and separated by a space using the `cookieExpires` argument of the `cookiePersistence` command. If no qualifier was entered, seconds will be used.	`cookieExpires` *`text`*

Field name	Description	Filter name
`Secure`	Specifies whether the secure keyword attribute is used to protect the confidentiality and authenticity of the cookie when the switch issues switch-managed Set-cookie response headers. The secure keyword indicates that the HTTP client should not include the cookie in subsequent non-secure requests, or should only include the cookie in secure requests.	`secure {true | false}`

Associated MIB

`op.mib`

Web path

- vSwitch → *name* → LoadBalance → cookiePersistence

show healthCheckProfile

Purpose

Displays the configuration for a named server health check profile or all health check profiles configured on the system.

Access mode

user

Syntax

```
show vSwitch-name loadBalance healthCheckProfile
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show healthCheckProfile
Name:                    health_1
Type:                    TCP
Interval:                5
Retries:                 3
Success Rate:            0
Timeout:                 2
Count:                   3

Name:                    health_100
Type:                    DNS_TCP
Interval:                5
Retries:                 3
Success Rate:            0
Timeout:                 2
Count:                   3
DNS Mode:                anyHostResponse
DNS Req String:          isi.edu
DNS Rsp String:          10.10.10.5
```

Output description

Field name	Description	Filter name
The `name` and `type` fields display for all probe types.		
`Name`	The text string that identifies the health check profile.	`name` *`text`*
`Type`	The method this health check profile uses to verify the availability of each server within a service group. Possible `type` values are: `ICMP`: The method that sends a ping from the load-balancing software to the network interface card (NIC) in the server to verify connectivity and basic functionality. `TCP`: The method that verifies functionality through the NIC and within the operating system. Specifically, this option verifies that the TCP server application is running and associated with a TCP port. `HTTP`: The method that verifies NIC and operating system functionality, as well as the server's storage system. Storage is confirmed by verifying the contents of an HTML page. (The specific page is set with the `requestString` argument.) `FTP`: The method that verifies server availability by using FTP transactions. `LIST`: The method that provides the same verification as the HTTP option, but uses multiple shc profiles. `DNS_UDP`: The method that verifies Domain Name Server using a UDP probe. `DNS_TCP`: The method that verifies Domain Name Server using a TCP probe.	`type {ICMP | TCP | HTTP FTP | LIST | DNS_UDP | DNS_TCP | RADIUS | RAW_UDP | RAW_TCP | RTSP | SCRIPT | POP3 | SMTP | IMAP4}`

Field name	Description	Filter name
Type (Cont)	RADIUS: The method that verifies Remote Authentication Dial-In User Service. RAW_UDP: The method that verifies Raw UDP. RAW_TCP: The method that verifies Raw TCP. RTSP: The method that verifies Real Time Stream Protocol server. SCRIPT: The command file to be executed as a probe. POP3: The method that verifies server availability by using POP3 protocol transactions. SMTP: The method that verifies server availability by using SMTP transactions. IMAP4: The method that verifies server availability by using IMAP4 protocol transactions.	
The following fields display for all probe types. The name and type fields and the following optional fields are the only fields that display for the ICMP and TCP probe types.		
Interval	The configured number of seconds between the start of health check probes.	interval *integer*
Retries	The configured number of failed probes allowed before a server is declared out of service and removed from the load-balancing table.	retries *integer*
Success Rate	The configured percentage of probes that must succeed before a server is declared out-of-service and removed from the load-balancing table. This is calculated from the past 32 probe iterations.	successRate *integer*
Timeout	The configured number of seconds to wait for a probe response before declaring the probe unanswered.	timeout *integer*
Count	The configured number of consecutive successful probes the system must receive before it declares the server active and returns its entry to the load-balancing table.	count *integer*

Field name	Description	Filter name
The following fields display if the probe type is HTTP.		
`HTTP Mode`	The method the HTTP probe should use to verify content of the requested object by searching server responses. Possible values are: `checkResponseCodes`: The system searches the response for an HTTP 200 OK response. `hash`: The system runs an arithmetic hash on the object to verify that the requested object has not changed. `searchForResponseString`: The system searches the response for a specified text string. Enter the string with the `responseString` argument.	`httpMode {checkResponseCodes | hash | searchForResponseString}`
`HTTP Req String`	The request string that the HTTP probe sends to verify content.	`httpRequestString` *`text`*
`HTTP Resp String`	The string the probe searches for in server responses.	`httpResponseString` *`text`*

Field name	Description	Filter name
The following fields display if the probe type is FTP.		
FTP Mode	The method that the FTP probe should use to verify the server health check responses. anyServerResponse: Indicates any response from the server to a connection request. goodServerResponse: Indicates a valid response from the server to a connection request. anyUserResponse: Indicates any user response from the server after a valid connection response. goodUserResponse: Indicates a valid user response from the server after a valid connection response. anyLoginResponse: Indicates any login response from the server after a valid connection and valid user response. goodLoginResponse: Indicates a valid login response from the server after a valid connection and valid user response.	FTPMode {anyServerResponse | goodServerResponse | anyUserResponse | goodUserResponse | anyLoginResponse | goodLoginResponse}
FTP Termination Mode	The method of ending an FTP probe session.	FTPTerminationMode {fastDisconnet | quitThenDisconnect}
User Name	The user name string applied in this FTP probe.	userName *text*
User Password	The password string applied in this FTP probe.	userPassword *text*
The following field displays if the probe type is LIST .		
Profile List	The list of health check profiles that are being aggregated into the larger “umbrella” profile..	profileList *text*

Field name	Description	Filter name
The following fields display if the probe types are DNS_TCP and DNS_UDP.		
DNS Mode	The method that the DNS probe uses to verify the server health check responses. anyHostResponse: Indicates any response to a DNS host lookup query. goodHostResponse: Indicates any valid response to a DNS host lookup query. specificHostAddress: Indicates the DNS name resolves to a specific host address. anyMailExResponse: Indicates any response to a Mail Exchange query. goodMailExResponse: Indicates any valid response to a Mail Exchange query. specificMailExResponse: Indicates a valid response to a Mail Exchange query with the specified server entries. verySpecificMailExResponse: Indicates a valid response to a Mail Exchange query with the specified preferences/servers. The default setting is anyHostResponse.	dnsMode {anyHostResponse | goodHostResponse | specificHostAddress | anyMailExResponse | goodMailExResponse | specificMailExResponse | verySpecificMailExResponse}
DNS Req String	The domain name for the DNS request. The default setting is isi.edu.	dnsReqString *text*
DNS Rsp String	The information used for evaluating DNS responses, usually an IP address entered in dotted-decimal notation or a list of Mail servers if one of the requests is a Mail Exchange query.	dnsRspString *text*

Field name	Description	Filter name
The following fields display if the probe type is `RADIUS`.		
`RADIUS Mode`	The method specified for the RADIUS probe and response verification. Possible values are: `anyAuthResponse`: Indicates any authentication response (accept or reject). `goodAuthResponse`: Indicates an authentication accept. `goodActgResponse`: Indicates a good accounting response. The default setting is `anyAuthResponse`.	`radiusMode {anyAuthResponse | goodAuthResponse | goodActgResponse}`
`User Name`	The user name string applied in this RADIUS probe. The default setting is `unspecified`.	`userName` *`text`*
`User Password`	The password string applied in this RADIUS probe. The default setting is `unspecified`.	`userPassword` *`text`*
`Secret`	The server's secret that is used to encrypt data and verify responses. The default setting is `unspecified`.	`secret` *`text`*
`Optional Request Attrs`	The custom list of RADIUS attributes in the form {`id-I` | A |S-*`value`*}, where the `id` is the numeric attribute ID; type is`I` for integer, A for address (ip), or S for string; and *`value`* is the value to be set.	`optionalRequestAttrs` *`text`*
`nasId`	The name used for identifying this switch in RADIUS server requests. The default setting is `Sun Radius Probe`.	`nasId` *`text`*

Field name	Description	Filter name
The following fields display if the probe types are `RAW_TCP` or `RAW_UDP`.		
`Raw Mode`	The method that the raw TCP or raw UDP probe uses to verify the server health check responses. `anyResponse`: Indicates any response to a raw TCP or UDP query. `specificResponse`: Indicates a matched expected response from the server to a raw TCP or UDP query. The default setting is `anyResponse`.	`rawMode {anyResponse|specificResponse}`
`Raw Req String`	The raw hex bytes used to build the TCP or UDP server health check request. The default setting is 00.	`rawReqString` *`text`*
`Raw Rsp String`	The raw hex information used for evaluating raw TCP or UDP responses to health check probes. The default setting is 00.	`rawRspString` *`text`*
`Profile Description`	The textual description for the health check profile.	`profileDescription` *`text`*

Field name	Description	Filter name
The following fields display if the probe type is RTSP .		
`RTSP Mode`	The method that the RTSP probe uses to verify the server health check responses. `anyOptionResponse`: Indicates any reponse from the server to an option request. `goodOptionResponse`: Indicates a valid response from the server to an option request. `anyDescribeResponse`: Indicates any response from a server to a describe request. `goodDescribeResponse`: Indicates a valid response from a server to a describe request.	`RTSPMode {anyOptionResponse | goodOptionResponse | anyDescribeResponse | goodDescribeResponse}`
`RTSP Url Name` *`text`*	The RTSP Universal Resource Locator (URL).	`rtspUrlName` *`text`*

Field name	Description	Filter name
The following fields display when the probe type is SCRIPT.		
`Script Mode`	The method that the Script probe uses to verify the server health check responses.	`scriptMode`
`Script File`	The file containing commands to be executed as a probe.	`scriptFile` *`text`*
`Script Commands`	The commands that will be executed per probe.	`scriptCommands` *`text`*
`Script Argument`	The values for script variables.	`scriptArg1` *`text`* `scriptArg2` *`text`* `scriptArg3` *`text`* `scriptArg4` *`text`*
`Profile Description`	The textual description for the health check profile.	`profileDescription` *`text`*

Field name	Description	Filter name
The following fields display when the probe type is POP3.		
POP3 Mode	The method that the POP3 probe uses to verify the server health check responses. anyServerResponse: Indicates any response from the server to a connection request. goodServerResponse: Indicates a valid response from the server to a connection request. anyUserResponse: Indicates any user response from the server after a valid connection response. goodUserResponse: Indicates a valid user response from the server after a valid connection response. anyLoginResponse: Indicates any login response from the server after a valid connection and valid user response. goodLoginResponse: Indicates a valid login response from the server after a valid connection and valid user response. anyStatResponse: Indicates any stat response from the server after a valid connection and valid login. goodStatResponse: Indicates a valid stat response from the server after a valid connection and valid login. The default setting is anyServerResponse.	POP3Mode {anyServerResponse | goodServerResponse | anyUserResponse | goodUserResponse | anyLoginResponse | goodLoginResponse | anyStatResponse | goodStatResponse}
POP3 Termination Mode	The method of ending a POP3 probe session. The default setting is fastDisconnect.	POP3TerminationMode {fastDisconnet | quitThenDisconnect}

Field name	Description	Filter name
`User Name`	The user name string applied in this POP3 probe. The default setting is `unspecified`.	`userName` *`text`*
`User Password`	The password string applied in this POP3 probe. The default setting is `unspecified`.	`userPassword` *`text`*

Field name	Description	Filter name
The following fields display if the probe type is SMTP.		
SMTP Mode	The method that the SMTP probe uses to verify the server health check responses. anyServerResponse: Indicates any response from the server to a connection request. goodServerResponse: Indicates a valid response from the server to a connection request. anyHelloResponse: Indicates any hello response from the server after a valid connection response. goodHelloResponse: Indicates a valid hello response from the server after a valid connection response. anyVerifyResponse: Indicates any verify response from the server after a valid connection and valid hello response. goodVerifyResponse: Indicates a valid verify response from the server after a valid connection and valid hello response. anySendResponse: Indicates any send response from the server after a valid connection and valid hello response. goodSendResponse: Indicates a valid send response from the server after a valid connection and valid hello response. The default setting is anyServerResponse.	SMTPMode {anyServerResponse | goodServerResponse | anyHelloResponse | goodHelloResponse | anyVerifyResponse | goodVerifyResponse | anySendResponse | goodSendResponse}

Field name	Description	Filter name
SMTP Termination Mode	The method of ending an SMTP probe session. The default setting is `fastDisconnect`.	SMTPTerminationMode {fastDisconnet | quitThenDisconnect}
SRC Name	The user address string used as the originator for a message.	srcName *text*
Dst Name	The user address/name string used to send a message or verify a user.	dstName *text*
Msg Subject	The subject of the message.	msgSubject *text*
Msg Body	The textual content of the message.	msgBody *text*

Field name	Description	Filter name
The following fields are displayed if the probe type is `IMAP4` .		
`IMAP4 Mode`	The method that the IMAP4 probe uses to verify the server health check responses. `anyServerResponse`: Indicates any response from the server to a connection request. `goodServerResponse`: Indicates a valid response from the server to a connection request. `anyCapabilityResponse`: Indicates any capability response from the server after a valid connection response. `goodCapabilityResponse`: Indicates a valid capability response from the server after a valid connection response. `anyLoginResponse`: Indicates any login response from the server after a valid connection and valid capability response. `goodLoginResponse`: Indicates a valid login response from the server after a valid connection and valid capability response. `anyListResponse`: Indicates any list response from the server after a valid connection and valid login. `goodListResponse`: Indicates a valid list response from the server after a valid connection and valid login.	`IMAP4Mode {anyServerResponse | goodServerResponse | anyCapabilityResponse | goodCapabilityResponse | anyLoginResponse | goodLoginResponse | anyListResponse | goodListResponse | anyExamineResponse | goodExamineResponse}`

Field name	Description	Filter name
IMAP4 Mode (cont)	anyExamineResponse: Indicates any examine response from the server after a valid connection, valid login, and valid list response. goodExamineResponse: Indicates any valid examine response from the server after a valid connection, valid login, and valid list response. The default setting is anyServerResponse.	
IMAP4 Termination Mode	The method of ending an IMAP4 probe session. The default setting is fastDisconnect.	IMAP4TerminationMode {fastDisconnet | logoutThenDisconnect}
User Name	The user name string applied in this IMAP4 probe. The default setting is unspecified.	userName *text*
User Password	The password string applied in this IMAP4 probe. The default setting is unspecified.	userPassword *text*
Mbox Name	The user address string used as the originator for a message.	mboxName *text*

Associated MIB

shc.mib

Web path

- vSwitch → *name* → LoadBalance → healthCheckProfile

show healthCheckProfile passive

Purpose

Displays the passive server health check probe settings for a named server health check profile or all health check profiles configured on the system. Passive probes are out-of-band health checks that slow down normal in-line health check probes if the in-line probes continue to report good server health.

By default, passive server health checks are OFF. To enable this feature, set the `successiveCount` argument to a number greater than 0. All hosts in the service group must be UP and reporting good health for the passive health check feature to operate.

Access mode

config

Syntax

```
show vSwitch-name loadBalance healthCheckProfile passive
show vSwitch-name loadBalance healthCheckProfile-name passive
```

Sample output

```
sun# configure
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show healthCheckProfile
health_9 passive
Name:                    health_9
Successive Count:        5
Activity Threshold:      50
Update Tolerance:        20
Min Poll Cycles:         30
Inter Process Timeout:   25
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the health check profile.	`name` *`text`*
`Successive Count`	The maximum number of successive passive probes allowed before forcing at least one normal probe. Once the successive count is reached, at least one normal poll will be executed. If set to 0 (default), passive probes are disabled.	`successiveCount` *`integer`*
`Activity Threshold`	The maximum time in seconds since a server last reported good health status. If the activity threshold time expires, normal server health check probes will be used.	`activityThreshold` *`integer`*
`Update Tolerance`	The length of time in seconds that a server health check is considered valid. Once this time has expired, a new normal health check update is required before passive probes can resume. Set this parameter with assistance from Sun Microsystems.	`updateTolerance` *`integer`*
`Min Poll Cycles`	The minimum number of normal probes to execute initially before passive probes are used to determine server health.	`minPollCycles` *`integer`*
`Inter Process Timeout`	The length of time in milliseconds to wait for a health check response update from a service group. Set this parameter with assistance from Sun Microsystems.	`interProcessTimeout` *`integer`*

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → healthCheckProfile → passive

show host

Purpose

Displays the server's name-to-IP address mapping, as well as the administrative state. These settings were configured with the `host` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance host`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show host
Name        IP Address   Admin State  Description   vRouter
nina       10.10.10.101  enabled      marketing     corp:default
pinta      10.10.10.102  enabled      finance       corp:default
santaMaria 10.10.10.103  enabled      sales         corp:default
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the server.	`name` *`text`*
`IP Address`	The unique IP address of the server.	`ipAddress` *`ipAddress`*
`Admin State`	The administrative state of the named host, either `enabled` or `disabled`.	`adminState {enabled | disabled}`
`Description`	A text description of the host.	`description` *`text`*
`vRouter`	The name of the vRouter (in the format *`vSwitch:vRouter`*) over which traffic reaches this host.	`vRouter` *`vSwitch:vRouter`*

Associated MIB

`host.mib`

Web path

- vSwitch → *name* → LoadBalance → host

show objectRule

Purpose

Displays the predicate statement configured for a named object rule. The predicate statement was configured with the `objectRule` command.

Access mode

user

Syntax

```
show vSwitch-name loadBalance objectRule
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show objectRule
Name:                    matchAll
Predicate Field:         URI_PATH matches "*"

Name:                    matchImages
Predicate Field:         URI_PATH matches "*/images/*"
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the object rule.	`name text`
`Predicate Field`	The configuration of the predicate statement. See Appendix C, "Object rule predicate statements" for a complete description of predicate statements.	`predicate text`

Associated MIB

`op.mib`

Web path

- vSwitch → *name* → LoadBalance → objectRule

show outboundNat dynamic

Purpose

Displays the current dynamic network address translation (NAT) configuration.

Network address translation, defined in RFC 1631, is an Internet routing standard that allows an IP-based network to manage its public (Internet) addresses separately from its private (intranet) addresses. Dynamic NAT translates multiple private addresses to a single public address. This means that one global public address can be used for a range of real IP addresses in the backend network.

Since dynamic NAT supports many backend server real IP addresses that map to a single global address, dynamic NAT must also translate the ephemeral port in each outbound request and maintain state information for each TCP connection.

Access mode

user

Syntax

```
show vSwitch-name loadBalance outboundNat dynamic
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce)# outboundNat
sun(config-vSwitch-e-commerce loadBalance outboundNat)# show dynamic
Name:           NAT2
NAT IP Address: 206.30.10.10
vRouter:        system:shared
Admin State:    enabled
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the dynamic NAT configuration.	`name` *`text`*
`NAT IP Address`	The single public IP address representing configured hosts for outbound packets.	`natIPRange` *`ipAddress`*
`vRouter`	The name of the virtual router, typically the Internet vRouter named *system:shared*, to which the host's vRouter will forward an HTTP response, in the format *`vSwitch`*`:`*`vrouterName`*.	`vrouter` *`vSwitch:vrouterName`*
`Admin State`	The current administrative state of dynamic NAT, either `enabled` or `disabled`.	`adminState {enabled | disabled}`

Associated MIB

`dynNAT.mib`

Web path

- vSwitch → *name* → LoadBalance → outboundNat → dynamic

show outboundNat dynamic hostIpRange

Purpose

Displays the host vRouter and the private IP address range configured for dynamic network address translation (NAT).

Access mode

user

Syntax

```
show vSwitch-name loadBalance outboundNat dynamic hostIpRange
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce)# outboundNat
sun(config-vSwitch-e-commerce loadBalance outboundNat)# show dynamic
hostIpRange
Name:                          nat1
Host Vrouter:                  system:shared
Origin Host IP Address Range: 10.10.80.1-10.10.80.20
```

Output description

Field name	Description	Filter name
Name	The text string that identifies the dynamic NAT configuration.	name *text*
Host vRouter	The name of the virtual router to which the host will forward an HTTP response, in the format *vSwitch*:*vrouterName*.	hostVrouter *vSwitch:vrouterName*
Origin Host IP Address Range	The range of private IP addresses that dynamic NAT maps to a single public IP address.	hostIPRangeList {*ipAddress-ipAddress*}

Associated MIB

`dynNAT.mib`

Web path

- vSwitch → *name* → LoadBalance → outboundNat → dynamic → hostIpRange

show outboundNat dynamic statistics

Purpose

Displays byte and packet data that the system transmitted and received under the named dynamic network address translation (NAT) configuration.

Access mode

user

Syntax

```
show vSwitch-name loadBalance outboundNat dynamic statistics
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce)# outboundNat
sun(config-vSwitch-e-commerce loadBalance outboundNat)# show dynamic
statistics
Name:                     nat1
Host Bytes Received:      0
Host Bytes Transmitted:   0
Host Packets Received:    0
Host Packets Transmitted: 0
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the dynamic NAT configuration.	`name` *`text`*
`Host Bytes Received`	The total number of inbound bytes received by this NAT configuration.	`hostBytesReceived` *`integer`*
`Host Bytes Transmitted`	The total number of outbound bytes transmitted using this NAT configuration.	`hostBytesTransmitted` *`integer`*
`Host Packets Received`	The total number of inbound packets received by this NAT configuration.	`hostPacketsReceived` *`integer`*
`Host Packets Transmitted`	The total number of outbound packets transmitted using this NAT configuration.	`hostPacketsTransmitted` *`integer`*

Associated MIB

`dynNAT.mib`

Web path

- vSwitch → *name* → LoadBalance → outboundNat → dynamic → statistics

show outboundNat static

Purpose

Displays the address ranges configured for real-to-virtual network address translations (NAT). These settings were configured with the `virtualService` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance outboundNat static`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce)# outboundNat
sun(config-vSwitch-e-commerce loadBalance outboundNat)# show static

Name:                          SNAT_internet
Origin Host IP Address Range: 10.10.4.0-10.10.4.255
NAT IP Address Range:          2.2.2.0-2.2.2.255
Admin State:                   enabled
vRouter:                       system:shared
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the static NAT configuration.	`name` *`text`*
`Origin Host IP Address Range`	The range of real addresses for configured hosts, in dotted-decimal format.	`hostIPRange` *`{ipAddress-ipAddress}`*
`NAT IP Address Range`	The range of virtual addresses mapped to (and representing) configured hosts.	`natIPRange` *`{ipAddress-ipAddress}`*

Field name	Description	Filter name
`Origin Host vRouter`	The name of the virtual router to which the backend hosts are connected, in the format *`vSwitch`*`:`*`vrouterName`*.	`hostVrouter` *`vrouterName`*
`Nat Vrouter`	The name of the virtual router, typically the Internet vRouter, to which the host's vRouter will forward an HTTP response, in the format *`vSwitch`*`:`*`vrouterName`*. The default is `system:shared`.	`vrouter` *`vrouterName`*
`Admin State`	The administrative state of the address translation capability, either `enabled` or `disabled`.	`adminState {enabled | disabled}`
`Client Source IP Range`	The range of client IP addresses that are allowed access to the backend servers via this static NAT configuration.	`clientSrcIPRange {`*`ipAddress-ipAddress`*`}`

Associated MIB

`staticNAT.mib`

Web path

- vSwitch → *name* → LoadBalance → outboundNat → static

show outboundNat static statistics

Purpose

Displays data of transmitted and received bytes and packets whose addresses fall within the named static network address translation (NAT) configuration. If you have a static NAT configuration, but no clients or hosts fall within the address ranges, the statistics counters display 0.

Access mode

user

Syntax

```
show vSwitch-name loadBalance outboundNat static statistics
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce)# outboundNat
sun(config-vSwitch-e-commerce loadBalance)# show static statistics
Name:                     nat1
Host Bytes Received:      0
Host Bytes Transmitted:   0
Host Packets Received:    0
Host Packets Transmitted: 0
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the static NAT configuration.	`name` *`text`*
`Host Bytes Received`	The total number of inbound bytes received by this NAT configuration.	`hostBytesReceived` *`integer`*
`Host Bytes Transmitted`	The total number of outbound bytes transmitted using this NAT configuration.	`hostBytesTransmitted` *`integer`*
`Host Packets Received`	The total number of inbound packets received by this NAT configuration.	`hostPacketsReceived` *`integer`*
`Host Packets Transmitted`	The total number of outbound packets transmitted using this NAT configuration.	`hostPacketsTransmitted` *`integer`*

Associated MIB

`staticNAT.mib`

Web path

- vSwitch → *name* → LoadBalance → outboundNat → static → statistics

show proxyIPPool

Purpose

Displays the configured pools of proxy IP addresses to be used for client address translation and TCP multiplexing. A configuration can have up to 256 pools.

Access mode

user

Syntax

```
show vSwitch-name loadBalance proxyIpPool
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vswitch-e-commerce loadbalance)# show proxyIpPool
Name:                    pip1
Proxy IP Address Pool:   1.1.1.1-1.1.1.10; 2.2.2.1-2.2.2.10
Host vRouter:            SVS1:default
Service Type:            L4SLB_ADV_OR_HTTP_OR_SSL
```

Output description

Field name	Description	Filter name
`Name`	The new or existing proxy IP pool name.	`name` *`text`*
`Proxy IP Address Pool`	The IP addresses contained in the pool using up to 64 IP addresses. Ranges (separated by a dash), and individual IP addresses (each separated by a comma) are allowed.	`ipRangeList {`*`ipAddress-ipAddress`*`}` or `ipRangeList` *`ipAddress`*
`vRouter`	The name of the virtual router to which the host is connected, in the format *`vSwitch`*`:`*`vrouterName`*. The default setting is the default vRouter in the operator-defined vSwitch.	`vRouter` *`vSwitch:vrouterName`*
`Service Type`	The virtual service application service type for which this proxy IP pool is used; either L4SLB (non-terminated TCP), or all other application service types (terminated TCP).	`appServiceType {L4SLB| L4SLB_ADV_OR_HTTP_OR_SSL}`

Associated MIB

`proxyIP.mib`

Web path

- vSwitch → *name* → LoadBalance → proxyIpPool

show proxyIPPool statistics

Purpose

Displays the proxy IP pool statistics on a vRouter.

Access mode

user

Syntax

```
show vSwitch-name loadBalance proxyIpPool statistics
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch SVS1
sun(config-vswitch-SVS1)# loadbalance
sun(config-vswitch-SVS1 loadbalance)# show proxyIpPool statistics

Name:                        pip1
Addresses In Pool:           10
Ports In Pool:               645100
Ports Allocated:             0
Ports Released:              0
Ports In Use:                0
Ports Available:             645100
Port Allocation Failures:    0
```

Output description

Field name	Description	Filter name
`Name`	The name of the proxy IP pool.	`name` *`text`*
`Addresses In Pool`	The total number of proxy IP addresses in the pool (up to 64 using a combination of ranges and individual IP addresses).	`addressesInPool` *`integer`*
`Ports In Pool`	The total number of IP ports in the pool.	`portsInPool` *`integer`*
`Ports Allocated`	The total number of port allocations that have occurred in the pool.	`portsAllocated` *`integer`*
`Ports Released`	The total number of ports released back to the pool.	`portsReleased` *`integer`*
`Ports In Use`	The number of ports currently in use from the proxy IP pool. `portsAllocated` minus `portsReleased` = `portsInUse`	`portsInUse` *`integer`*
`Ports Available`	The number of ports currently available from the pool. `portsInPool` minus `portsInUse` = `portsAvailable`	`portsAvailable` *`integer`*
`Port Allocation Failures`	The number of port allocation failures that have occurred for the pool.	`portAllocationFailures` *`integer`*

Associated MIB

`proxyIP.mib`

Web path

- vSwitch → *name* → LoadBalance → proxyIpPool

show realService

Purpose

Displays the real service configuration, including the IP address, protocol, and port. Configure these settings with the `realService` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance realService`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch SVS1
sun(config-vswitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# show realService
Name:                                rs1
Host Name:                           lnx1
Protocol:                            TCP
Port:                                8000
Weight In ServiceGroup:              1
Description:                         linux1
Admin State:                         enabled
Disable Delay:                       0
In Line SHC Failure Rate Threshold:1
Client Address Translation:          disabled
Bridge Mode:                         disabled
Encryption:                          unencrypted
Virtual Services:                    vs1
Oper Status:                         active

Name:                                rs2
Host Name:                           sol1
Protocol:                            TCP
Port:                                8001
Weight In ServiceGroup:              1
Description:                         solaris1
Admin State:                         enabled
Disable Delay:                       0
inLine SHC Failure Rate Threshold:  1
Client Address Translation:          disabled
```

```
Bridge Mode:                    disabled
Encryption:                     unencrypted
Virtual Services:               vs1
Oper Status:                    active
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies this real service.	`name` *`text`*
`Host Name`	The host configured for this real service.	`hostName` *`text`*
`Protocol`	The supported protocol for this real service, TCP or UDP. This setting must match the virtual service protocol configuration being used by the real services. The default setting is TCP.	`protocol {TCP | UDP}`
`Port`	The port that the host uses to listen for connections of the type specified with the `protocol` argument.	`port` *`portNumber`*
`Weight in Service Group`	The weight the associated host is configured with, which the system is using in combination with the algorithm (set with the `serviceGroup` command) to determine the number of assigned connections. The display is either a specific *`weight`* or `dynamic`. A setting of `dynamic` determines a weight calculation based on the server health check latency results.	`weight {dynamic |` *`weight`*`}`
`Description`	A text description of the real service.	`description` *`text`*
`Admin State`	The administrative state of the named service, either `enabled` or `disabled`.	`adminState {enabled | disabled}`
`Disable Delay`	The time, in seconds, that must elapse before the system disables a real service when the `adminState` is set to `disabled`. Possible values are 0 to 600 `seconds` or `unlimited`. The default setting is 0 seconds. If this value is set to non-zero or `unlimited` and the `adminState` is set to `disabled`, the real service does not accept any new connections but current connections are not terminated. At the end of the `disableDelay` time, the system terminates all current connections to the real service.	`disableDelay {`*`seconds`*` | unlimited}`

Field name	Description	Filter name
`In Line SHC Failure Rate Threshold`	The number of in-line server health check failures allowed before the real service is brought down. In-line server health checks determine server health without waiting for the next out-of-band polling interval. The default setting is 1. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information on server health checking.	`ilSHCFailureRateThreshold` *`integer`*
`Client Address Translation`	The administrative state of client address translation for this real service configuration, either `enabled` or `disabled`. If enabled, the Internet client's source address and source port information contained in the IP header are translated using a pool of proxy IP addresses configured on the N2000 Series. The default setting is `disabled`.	`clientAddressTranslation {enabled | disabled}`
`Proxy Ip Pool`	The name of the proxy IP pool to be used when `clientAddressTranslation` is set to `enabled`. See the *Sun N2000 Series Release 2.0 – System Configuration Guide* for information on client address translation (CAT) and the proxy IP pool.	`proxyIpPool` *`text`*
`Bridge Mode`	When enabled, traffic that is sent to the real service will be bridged (forwarded with the client's source address and virtual service destination address). The source and destination addresses will not be modified. The default is `disabled`.	`bridgeMode {disabled | enabled}`
`Encryption`	The encryption method for this real service traffic. Possible values are: `SSL`: SSL is enabled for this real service. `unencrypted`: No encryption is enabled for this real service. The default setting is `unencrypted`.	`encryption {unencrypted | SSL}`

Field name	Description	Filter name
`Persistence Name`	The persistence name string to match against the HTTP request when the policy `persistType` setting is the `fieldMatchName` (one possible algorithm used when the system makes session persistence decisions when forwarding HTTP requests).	`persistName` *`text`*
`Virtual Service`	The names of the virtual services to which this real service belongs. If the real service has not yet been assigned to a virtual service, the field will be blank.	`virtualService` *`text`*
`Oper Status`	The operational state of this real service, either: `active`: The real service is operational and traffic is being load balanced across it. `inactive`: The real service is not currently operational. Traffic destined for this real service is being dropped. `pending`: The real service is transitioning to the active state. `unavailable`: Health checks have determined that the real service is down. `disabled`: The real service `adminState` is set to `disabled`. `quiescing`: The real service is disabled but sessions have not been terminated. `standby`: The real service has been configured to be a standby server and is currently not in service.	`operStatus {active | inactive | pending |unavailable | disabled | quiescing | standby}`
`VSRP Redirect Address`	The IP address to which packets are forwarded when the real service is configured as a VSRP backup. All packets addressed to the backup real service's IP address are redirected to this IP address, and forwarded to the real service in the active switch.	`vsrpRedirectAddress` *`ipAddress`*

Associated MIB

`realSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → realService

show realService advanced

Purpose

Displays specific TCP settings for the system-to-server connection. These settings were configured with the `realService advanced` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance realService advanced`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-example loadBalance)# show realService advanced
Name:                          RS1
TCB Table Template Key ID:     0
IP Type Of Service:            Normal
Retransmit Retry Limit:        4
Establishment Retry Limit:     4
RX Timer Short Timeout Value: 16_seconds
RX Timer Long Timeout Value:  64_seconds
Receive Window Size:           20480
Round Trip Time:               750_msec
SMM Stream Limit:              1xRcvWnd
Est Short Timeout:             ExpRetr
Rx Window Disable:             false
Rcv Mss:                       1460
Xmt Mss:                       1460
Enable Http Mode:              false
Http Get Piggy Back:           true
Rx Use Long Time:              true
```

Output description

Field name	Description	Filter name
`Name`	The name of the configured real service whose advanced TCP connections you are viewing.	`name` *`text`*
`TCB Table Template Key ID`	The TCB table template key in use by this connection. The default, and only acceptable value, is 0.	`tcbTemplateKey` *`integer`*
`IP Type Of Service`	The bit set in the type of service (TOS) field of IP headers for traffic traversing this connection. This informs intervening routers how they should handle the packets. Possible values are: `Normal`: Used for ICMP, BOOTP, DNS, TCP queries traffic `MinCost`: Used for Usenet traffic `MaxReliablity`: Used for SNMP and routing traffic `MaxThroughput`: Used for FTP traffic `MinDelay`: Used for interactive login applications, such as Telnet and rlogin	`ipTOS {Normal | MinCost | MaxReliablity | MaxThroughput | MinDelay}`
`Retransmit Retry Limit`	The base interval for the retransmit retry limit within an established connection. That is, the time between retransmits of data sent and not acknowledged.	`xmtRetryLimit` *`integer`*
`Establishment Retry Limit`	The base interval for the establishment retry limit, which is how long you are willing to attempt a connection.	`estRetryLimit` *`integer`*
`RX Timer Short Timeout Value`	The receive timer short time-out value, which defines how long you are willing to wait to receive the first data packet from a real service after a session has been established.	`shortRxTimer` *`integer`*
`RX Timer Long Timeout Value`	The receive timer long time-out value, which defines how long you are willing to wait to receive subsequent data from the real service after the initial packet.	`longRxTimer` *`integer`*

Field name	Description	Filter name
`Receive Window Size`	The maximum amount of memory allocated to received data on a session.	`rcvWnd` *integer*
`Round Trip Time`	The maximum value for the packet round-trip transmit time in seconds or milliseconds.	`xmtRTT` *roundTripValue*
`SMM Stream Limit`	The Stream Memory Manager multiplier from the `Receive Window Size` field.	`smmStreamLimit` *streamLimitValue*
`Est Short Timeout`	The type of Establishment Time time-out; either the value from `ExpRFC793` or `ExpRetr`.	`estShortTimout {ExRFC793 | ExpRetr}`
`Receive Window Disable`	Specifies whether the TCP receive flow control is performed on a per-connection basis and advertised to the sender. The default setting is `false`.	`rcvWndDisable {true | false}`
`Rcv Mss`	Specifies the receive maximum segment size; broadcasted with a TCP SYN segment. The default setting is 1460.	`rcvMss` *integer*
`Xmt Mss`	Specifies the maximum segment size; advertised during the SYN handshake. The default setting is 1460.	`xmtMss` *integer*
`Enable HTTP Mode`	Indicates whether HTTP mode is enabled, either `true` or `false`. This is a performance-tuning parameter that optimizes HTTP. The default setting is `false`.	`enableHttpMode {true | false}`
`HTTP Get PiggyBack`	Specifies whether TCP `GET` requests are piggybacked on the last portion of 3-way handshake. This parameter only applies to server side connections. The default setting is `true`.	`httpGetPiggyBack {true | false}`
`Rx Use Long Time`	Specifies whether the internal receive progress timer is used to recover dormant connections. The default setting is `true`.	`rxUseLongTime {true | false}`

Associated MIB

`realSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → realService → advanced

show realService slbInfo

Purpose

Displays summary information returned by health check probes to the named real service.

Access mode

user

Syntax

```
show vSwitch-name loadBalance realService slbInfo
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show realService slbInfo
Real Service Name    Status Last Latency Average Latency     Weight
linux-110:80             UP            0                0      16383
linux-111:80             UP            0                0      16383
linux-112:80             UP            0                0      65535
linux-114:80             UP            0                0      32767
sun91:80                 UP            0                0      65535
sun92:80                 UP            0                0      65535
w2k-82:80                UP            0                0      32767
w2k-83:80                UP            0                0      16383
w2k-85:80                UP            0                0      32768
w2k-86:80                UP            0                0      16386
winnt-81:80              UP            0                0      32768
winxp-87:80              UP            0                0      63015
winxp-88:80              UP            0                0      63015
winxp-90:80              UP            0                0       2520
winxp89:80               UP            0                0       2520
```

Output description

Field name	Description	Filter name
`Real Service Name`	The text string that identifies the real service (server) from which these statistic were recorded.	`realServiceName` *`text`*
`Status`	The status of the named real service, either: `down`: There was no response to the probe. `dying`: The most recent probe(s) were unanswered, but the retry count has not yet been exceeded. `up`: Probe returned and confirmed availability.	`status {down | dying | up}`
`Last Latency`	The most recent probe result reported to the named real service.	`lastLatency` *`integer`*
`Average Latency`	The average latency to the named real service, which the switch recalculates with each returned probe.	`averageLatency` *`integer`*
`Weight`	The configured weight (static) or calculated weight (dynamic) for the named real service. The static weight attribute is set with the `realService` command.	`weight` *`integer`*

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → realService → slbInfo

show realService ssl

Purpose

Displays the real service Secure Sockets Layer (SSL) configuration, including the SSL certificate name, type, protocols, cipher suites, renegotiation support, and resumption support. Configure these settings with the `realService` or `realService ssl` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance realService ssl`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show realService ssl
Name:                       RS1
Host Name:                  host1
Port:                       8080
Encryption:                 N/A
SSL Certificate Name:       N/A
SSL Certificate Type:       CA
SSL Protocols:              SSLv3; TLSv1
SSL Ciphersuites:           RSA_WITH_RC4_128_MD5; RSA_WITH_RC4_128_SHA;
                            RSA_WITH_3DES_EDE_CBC_SHA
SSL Renegotiation Support: true
SSL Resumption Support:    true
Oper Status:                inactive
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the real service.	`name` *`text`*
`Host Name`	The host configured for this real service.	`hostName` *`text`*

Field name	Description	Filter name
`Port`	The port that the host uses to listen for connections.	`port` *`portNumber`*
`Encryption`	The encryption method for this real service traffic. Possible values are: SSL: SSL is enabled for this real service. `unencrypted`: No encryption is enabled for this real service.	`encryption {unencrypted | SSL}`
`SSL Certificate Name`	The name of an imported certificate used by the system to verify real service(s). If `certName` is not a valid CKM table entry, no other SSL configuration parameters set through this real service are used.	`certName` *`text`*
`SSL Certificate Type`	The SSL authentication type for the real service, either CA or `literal`. If set to `literal`, the system only accepts the certificate named in `certName` during SSL certificate verification. If set to CA, the system accepts any valid certificate signed by the key holder of the certificate named in `certName`.	`certType {CA | literal}`
`SSL Protocols`	The SSL protocols supported by this real service (SSL at the back end). Possible values are: SSLv3: Secure Sockets Layer Protocol, Version 3.0 TLSv1: Transport Layer Security (TLS) Version 1.0, as defined in RFC 2246. (TLS is also known as SSL, Version 3.1.)	`sslProto {SSLv3 | TLSv1}`

Field name	Description	Filter name

`SSL Ciphersuites`	The encryption methods (cipher suites) used on traffic passing through this real service. (See RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*, for detailed descriptions of each cipher.) Possible supported cipher suites include:
	
	`RSA_EXPORT_WITH_RC4_40_MD5`
	
	`RSA_WITH_RC4_128_MD5`
	
	`RSA_WITH_RC4_128_SHA`
	
	`RSA_EXPORT_WITH_DES40_CBC_SHA`
	
	`RSA_WITH_DES_CBC_SHA`
	
	`RSA_WITH_3DES_EDE_CBC_SHA`
	
	`RSA_WITH_AES_128_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_DES_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_RC4_56_SHA`
		`sslCiphers` *`ciphers`*

`SSL Renegotiation Support`	This real service's capability for renegotiating cryptographic parameters over an existing connection when necessary, either `true` or `false`. When set to `true`, the system allows multiple SSL handshakes to occur over an existing SSL connection to the real service. When set to `false`, the system ignores renegotiation requests from the real service. The default is `true`.	`reneg {true | false}`

Field name	Description	Filter name
`SSL Resumption Support`	The system's capability for doing SSL session resumption when connecting to this real service. When set to `true`, the system attempts to resume a previous SSL session when connecting to the real service, resulting in better performance. When set to `false`, the system initiates a full SSL handshake on all new connections to the real service. The default setting is `true`.	`resume {true | false}`
`Oper Status`	The operational state of this real service. Possible values are: `active`: The real service is operational and traffic is being load balanced across it. `inactive`: The real service is not currently operational. Traffic destined for this real service is being dropped. `pending`: The real service is transitioning to the active state. `unavailable`: Health checks have determined that the real service is down. `disabled`: The real service `adminState` is set to `disabled`. `quiescing`: The real service is disabled but sessions have not been terminated. `standby`: The real service has been configured to be a standby server and is currently not in service.	`operStatus {active | inactive | pending |unavailable | disabled | quiescing | standby}`

Additional filters

You can also use the following to filter the display output.

```
weight {integer | dynamic}
description text
adminState {enabled | disabled}
disableDelay {integer | unlimited}
virtualService text
vsrpRedirectAddress ipAddress
```

Associated MIB

```
realSvc.mib
```

Web path

- vSwitch → *name* → LoadBalance → realService → ssl

show realService ssl statistics

Purpose

Displays Secure Sockets Layer (SSL)-specific connection data.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance realService ssl statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show realService ssl
statistics
Name:                            linux-112:80
Current Open SSL Connections:    0
Peak Open SSL Connections:       0
Total Successful SSL Handshakes: 0
Total Failed SSL Handshakes:     0

Name:                            linux-114:80
Current Open SSL Connections:    0
Peak Open SSL Connections:       0
Total Successful SSL Handshakes: 0
Total Failed SSL Handshakes:     0

Name:                            sun91:80
Current Open SSL Connections:    0
Peak Open SSL Connections:       0
Total Successful SSL Handshakes: 0
Total Failed SSL Handshakes:     0
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the real service.	`name` *`text`*
`Current Open SSL Connections`	The number of SSL connections currently open on this real service.	`sslCurrentConnections` *`integer`*
`Peak Open SSL Connections`	The greatest number of SSL connections open at one time on this real service.	`sslPeakConnections` *`integer`*
`Total Successful SSL Handshakes`	The total number of completed SSL handshakes on this real service. (Each connection can require more than a single handshake.)	`sslHandshakesSuccess` *`integer`*
`Total Failed SSL Handshakes`	The total number of attempted but failed SSL handshakes on this real service.	`sslHandshakesFailure` *`integer`*

Associated MIB

`realSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → realService → ssl → statistics

show realService statistics

Purpose

Displays the total number of transmitted and received bytes and packets on a named real service, as well as session information.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance realService statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show realService
statistics
Name:                                linux-114:80
Bytes Transmitted to Real Service:   981796834
Bytes Received from Real Service:    232567388
Packets Transmitted to Real Service: 1811292
Packets Received from Real Service:  1087553
Cumulative Open Sessions:            362194
Cumulative Closed Sessions:          337872
Current Open Sessions:               1
Peak Open Sessions:                  7
Requests Forwarded:                  362105
Responses Received:                  362104
Connect Failures:                    0
Write Failures:                      0
Read Failures:                       0
Invalid HTTP Responses:              0
Current Pooled:                      11783
Number Pooled:                       334121
```

```
Name:                                sun91:80
Bytes Transmitted to Real Service:   0
Bytes Received from Real Service:    0
Packets Transmitted to Real Service: 0
Packets Received from Real Service:  0
Cumulative Open Sessions:            0
Cumulative Closed Sessions:          0
Current Open Sessions:               0
Peak Active Sessions:                0
Requests Forwarded:                  0
Responses Received:                  0
Connect Failures:                    0
Write Failures:                      0
Read Failures:                       0
Invalid HTTP Responses:              0
Current Pooled:                      0
Number Pooled:                       0
```

Output description

Field name	Description	Filter name
Name	The text string that identifies the real service.	name *text*
Bytes Transmitted to Real Service	The total number of bytes transmitted over this real service.	bytesSent *integer*
Bytes Received from Real Service	The total number of bytes received over this real service.	bytesReceived *integer*
Packets Transmitted to Real Service	The total number of packets transmitted over this real service.	packetsSent *integer*
Packets Received from Real Service	The total number of packets received over this real service.	packetsReceived *integer*
Cumulative Open Sessions	The total number of sessions open on this real service.	openSessions *integer*
Cumulative Closed Sessions	The total number of sessions that have been opened and then closed on this real service.	closedSessions *integer*
Current Open Sessions	The number of Layer 5 through Layer 7 sessions currently open on this real service. This number includes sessions that are in the TIME_WAIT state.	currentSessions *integer*

Field name	Description	Filter name
Peak Active Sessions	The greatest number of Layer 5 through Layer 7 sessions active at one time on this real service. This count does not include sessions in the TIME_WAIT state.	serverPeakSessions *integer*
Requests Forwarded	The total number of HTTP requests forwarded to the real service.	serverObjectsSend *integer*
Responses Received	The total number of successful responses to Layer 5 through Layer 7 HTTP requests.	serverObjectsReceived *integer*
Connect Failures	The total number of Layer 4 failures that occurred when connecting to this real service.	serverConnectFailures *integer*
Write Failures	The total number of Layer 4 failures that occurred when writing to this real service.	serverWriteFailures *integer*
Read Failures	The total number of Layer 4 failures that occurred when reading from this real service.	serverReadFailures *integer*
Invalid HTTP Responses	The total number of invalid HTTP responses from this real service.	serverInvalidHTTPResponses *integer*
Current Pooled	The total number of currently open pooled connections on this real service.	currentPooled *integer*
Number Pooled	The total number of pooled connections used per client connection for this real service.	numPooled *integer*

Associated MIB

realSvc.mib

Web path

- vSwitch → *name* → LoadBalance → realService → statistics

show requestPolicy

Purpose

Displays the parameters for a named request policy or all configured request policies on the system. These settings were configured with the `requestPolicy` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance requestPolicy`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch SVS1
sun(config-vSwitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# show requestPolicy
Name:                    qp1
Action:                  forward
Service Group:           sg1
Object Rule:             or-all
Precedence:              2
Persist Type:            none
Source Address Mask:     255.255.255.255
Optimize Last Response: disabled
Use Pooled Connections: disabled
First Object Switching: disabled
Oper Status:             active
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the request policy.	`name` *`text`*
`Action`	The action the load balancer should take when it finds a match to the named object policy. Possible actions are: • `forward` • `sorry`: Possible sorry actions include: - `Close` - `Page` - `Redirect`: - `Reset`	`action [forward | sorry]`
`Object Rule`	The name of the object rule associated with this policy.	`objectRule` *`text`*
`Sorry Data`	Sorry data associated with this policy; empty string for none.	`sorryData` *`text`*
`Precedence`	The precedence of the request policy, that is, the order in which the system considers the actions configured within it. If multiple policies exist with the same precedence, they are evaluated in the order in which they were configured. The default value is 1 (highest precedence).	`precedence` *`integer`*
`Persistence Type`	The algorithm used when the system makes session persistence decisions when forwarding HTTP requests. The options are `none`, `fieldHash`, `fieldMatchName`, `srcAddress`, and `switchCookie`.	`persistType {none | fieldHash | fieldMatchName | srcAddress |switchCookie}`
`Cookie Persistence`	The name of the cookie to be inserted into forwarded packets, from the cookie persistence table. You configure the parameters of the cookie with the `cookiePersistence` command.	`cookiePersist` *`text`*
`Field Prefix`	The string used to delineate the field in the HTTP request when the persistence type is `fieldHash` or `fieldMatchName`. If the URI_QUERY is expected to be of the format `ide=abcdefg01` then `Field Prefix` would be `ide`.	`fieldPrefix` *`text`*

Field name	Description	Filter name
Source Address Mask	The mask to apply to the IP address retrieved with the `srcAddress` value of the `persistType` argument. The default mask is 255.255.255.255.	`srcAddressMask` *`ipAddress`*
`Optimize Last Response`	The parameter that indicates whether the data path is optimized after the client sends a close request. The default value is `enabled`.	`optimizeLastResponse {enabled | disabled}`
`Use Pooled Connections`	The setting that enables the use of a pool of connections to real services.	`usePooledConnections {enabled | disabled}`
`Oper Status`	The operational state of this request policy. Possible values are: `active`: The request policy is operational and traffic is being load balanced across it. `inactive`: The request policy is not currently operational. `pending`: The request policy is transitioning to the operational state. `unavailable`: Health checks have determined that the real services in this request policy are down.	`operStatus {active | inactive | pending |unavailable}`

Associated MIB

`SUN-APP-SWITCH-REQUEST-POLICY-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → requestPolicy

show requestPolicy statistics

Purpose

Displays the number of requests that were redirected as a result of an object rule match and action.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance requestPolicy statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show requestPolicy images
statistics
Name:                       images
Number Policy Hits:         12
Number Sorry Service Hits:  4
Number Pooled:              5
Number Cookie Forwards:     3
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the request policy.	`name` *`text`*
`Number Policy Hits`	The number of times an incoming request has been evaluated and found to match this policy.	`PolicyHits` *`integer`*
`Number Sorry Service Hits`	The number of times that an incoming request matched this policy but there was a failure in reaching the real service to process the request.	`sorryServiceHits` *`integer`*
`Number Pooled`	The number of requests that use pooled connections.	`numPooled` *`integer`*
`Number Cookie Forwards`	The number of times a cookie is forwarded. This statistic applies only when the request policy `action` is set to `forward`.	`cookieForwards` *`integer`*

Associated MIB

`SUN-APP-SWITCH-REQUEST-POLICY-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → requestPolicy→ statistics

show requestTransform

Purpose

Displays attributes of one or more request transforms.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance requestTransform`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show requestTransform rqt1
Name:                    rqt1
Object Rule:             matchall
Custom HTTP Header:      N/A
Insert SSL Cipher:       enabled
SSL Cipher Header Name: X-SslClientCipher
Insert Source IP:        disabled
Source IP Header Name:   X-IP
Delete HTTP Header:      N/A
Oper Status:             inactive
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the request transform policy.	`name` *`text`*
`Object Rule`	The name of the object rule associated with this request transform.	`objectRule` *`text`*
`Custom HTTP Header`	An HTTP header that will be inserted into all requests.	`customHTTPHeader` *`text`*
`Insert SSL Cipher`	The setting that enables the insertion of an independent header line, which will be used to send SSL cipher information.	`SSLCipher {enabled | disabled}`
`SSL Cipher Header Name`	The name of the HTTP header used to send the SSL cipher information.	`SSLCipherHeader` *`text`*
`Insert Source IP`	The setting that enables the insertion of the named source IP header, an independent header line conveying the source IP address of the client into each forwarded request.	`insertSourceIP {enabled | disabled}`
`Source IP Header Name`	When `insertSourceIP` is enabled, this is the name of the HTTP header used to send the client source IP address field.	`sourceIPHeader` *`text`*
`Delete HTTP Header`	An HTTP header that will be deleted from all forwarded requests.	`deleteHTTPHeader` *`text`*
`Oper Status`	The operational state of this request transform. Possible values are: `active`: The request transform is operational. `inactive`: The request transform is not currently operational. `pending`: The request transform is transitioning to the active state.	`operStatus {active | inactive | pending}`

Associated MIB

`SUN-APP-SWITCH-REQUEST-TRANSFORM-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → requestTransform

show responsePolicy

Purpose

Displays the attributes of one or more response policies.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance responsePolicy`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show responsePolicy resp1
Name:                   resp1
Action:                 retry
Object Rule:            matchImages
Precedence:             1
Override Cookie Insert: N/A
Inline Health Check:    disabled
Operational Status:     inactive
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the response policy.	`name` *`text`*
`Action`	The action the load balancer should take when it finds a match to the named object policy. Possible actions are: • `return` • `retry` • `sorry`	`action [retry | return | sorry]`
`Object Rule`	The name of the object rule associated with this policy.	`objectRule` *`text`*
`Precedence`	The precedence of the response policy, that is, the order in which the system considers the actions configured within it. If multiple policies exist with the same precedence, they are evaluated in the order in which they were configured. The default value is 1 (highest precedence).	`precedence` *`integer`*
`Override Cookie Insert`	When enabled, the `set-Cookie` field will not be inserted into the reponse.	`overrideCookieInsert {enabled | disabled}`
`Inline Health Check`	When enabled, this setting treats responses as an in-line health check failure.	`inlineHealthCheck {enabled | disabled}`
`Oper Status`	The operational state of this response policy. Possible values are: `active`: The response policy is operational. `inactive`: The response policy is not currently operational. `pending`: The response policy is transitioning to the active state.	`operStatus {active | inactive | pending}`

Associated MIB

`SUN-APP-SWITCH-RESPONSE-POLICY-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → responsePolicy

show responsePolicy statistics

Purpose

Displays the number of responses that were initiated as a result of an object rule and action match.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance responsePolicy statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show responsePolicy
statistics resp1
Name:                       resp1
Number Policy Hits:         12
Number Sorry Service Hits:  4
Number Cookie Inserts:      3
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the response policy.	`name` *`text`*
`Number Policy Hits`	The number of times an outgoing response was evaluated and found to match this policy.	`PolicyHits` *`integer`*
`Number Sorry Service Hits`	The number of times that an outgoing response matched this policy but there was a failure in reaching the real service to process the response.	`sorryServiceHits` *`integer`*
`Number Cookie Inserts`	The number of times a cookie was inserted. This statistic applies only when the response policy `action` is set to `return`.	`cookieInserts` *`integer`*

Associated MIB

`SUN-APP-SWITCH-RESPONSE-POLICY-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → responsePolicy → statistics

show responseTransform

Purpose

Displays the attributes for one or more response transforms.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance responseTransform`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show responseTransform
rst1
Name:                    rst1
Object Rule:             matchImages
Rewrite Redirects:       enabled
Rewrite Port:            N/A
Custom HTTP Header:      N/A
Delete HTTP Header:      N/A
Operational Status:     active
```

Output description

Field name	Description	Filter name
`Name`	The text string identifying the response transform policy.	`name` *`text`*
`Object Rule`	The name of the object rule associated with this policy.	`objectRule` *`text`*
`Rewrite Redirects`	If traffic for an SSL session is redirected, indicates whether the header is rewritten to keep the client session private.	`rewriteRedirects {enabled | disabled}`
`Rewrite Port`	The parameter that indicates the ports on which the system performs redirect rewrites.	`rewritePort {`*`integer`*` | all}`
`Custom HTTP Header`	An HTTP header that will be inserted into each returned response.	`customHTTPHeader` *`text`*
`Delete HTTP Header`	An HTTP header that will be deleted from each returned response.	`deleteHTTPHeader` *`text`*
`Oper Status`	The operational state of this response transform. Possible values are: `active`: The response transform is operational. `inactive`: The response transform is not currently operational. `pending`: The response transform is transitioning to the active state.	`operStatus {active | inactive | pending}`

Associated MIB

`SUN-APP-SWITCH-RESPONSE-TRANSFORM-MIB.mib`

Web path

- vSwitch → *name* → LoadBalance → responseTransform

show serviceGroup

Purpose

Displays load-balancing algorithm settings and metrics for the service group. These settings were configured with the `serviceGroup` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup`

Sample output

```
sun(config-vSwitch-SVS1 loadBalance)# show serviceGroup
Name:                          sg1
Load Balance Type:             roundRobin
Configured Real Services:      rs1; rs2
Standby Real Services:         rs3
Active Real Services:          rs1; rs2
Admin State:                   enabled
Virtual Services:              vs1
Health Check Name:             first
In-Line Health Check Failure: enabled
Failover Retry Count:          1
List of Response Policies:     pp-500; pp-400
Oper Status:                   active
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the service group.	`name` *`text`*
`Load Balance Type`	The algorithm the system is using to make load-balancing decisions across this service group. See "Choosing a load-balance type" at the beginning of the `serviceGroup` command description for a full explanation of the types.	`loadBalanceType {roundRobin | weightedRandom | weightedHash | leastConnections}`
`Configured Real Services`	The list of real services (servers) that you configured as part of this service group.	`cfgRealServices` *`text`*
`Standby Real Services`	The list of real services (servers) that will be used for load balancing if any of the `cfgRealServices` are unavailable.	`standbyRealServices` *`text`*
`Active Real Services`	The list of real services (servers) that are determined to be active and which the system will use to make load-balancing decisions.	`realServices` *`text`*
`Admin State`	The administrative state of this service group, either `enabled` or `disabled`.	`adminState {enabled | disabled}`
`Virtual Service`	The virtual services to which the service group belongs.	`virtualService` *`text`*
`Health Check Name`	The name of the health check profile the system is using with the real services (servers) of this service group.	`healthName` *`text`*
`Inline Health Check`	Indicates whether the system uses in-band health checks to determine server health without waiting for the next out-of-band polling interval.	`inlineHealthCheck {enabled | disabled}`
`Failover Retry Count`	The number of tries the system will make to connect to a different real service (server) within the same service group before failover.	`retryCount {0 | 1 | 2}`
`List of Response Policies`	List of response policies to control responses returned to clients.	`responsePolicyList` *`text`*

Field name	Description	Filter name
List of Response Transforms	List of response transforms to control responses returned to clients.	`responseTransformList` *text*
Oper Status	The operational state of this service group. Possible values are: `active`: The service group is operational and traffic is being load balanced across it. `inactive`: The service group is not currently operational. Traffic destined for this real service is being dropped. `pending`: The service group is transitioning to the active state. `unavailable`: Health checks have determined that the real services in this service group are down. `disabled`: The service group `adminState` is set to `disabled`. `quiescing`: The service group is disabled but sessions have not been terminated.	`operStatus {active | inactive | pending |unavailable | disabled | quiescing}`

Associated MIB

`svcGrp.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup

show serviceGroup slbInfo

Purpose

Displays statistics resulting from information that in-band health check probes returned for the real services belonging to the named service group. The command output displays all real services configured for the service group, or can be further filtered to display only a specific real service.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup slbInfo`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup slbInfo

Service Group   Real Service     Status Last Latency Average Latency     Weight
bugCounter_SG   sun91:80             UP            0               0      65535
vlan20_SG       linux-110:80         UP            0               0      16383
vlan20_SG       linux-111:80         UP            0               0      16383
vlan20_SG       w2k-83:80            UP            0               0      16383
vlan20_SG       w2k-86:80            UP            0               0      16386
```

Output description

Field name	Description	Filter name
`Service Group`	The text string that identifies the service group.	`serviceGroupName` *`text`*
`Real Service Name`	The text string that identifies the real service within the service group. If you do not enter a `realServiceName`, the command displays statistics for all real services in the named service group.	`realServiceName` *`text`*
`Status`	The status of the named real service, either: `down`: There was no response to the probe. `dying`: The most recent probe(s) were unanswered, but the retry count has not yet been exceeded. `up`: Probe returned and confirmed availability. `inProgress`: Probe returned and the real service is configuring.	`status {down | dying | up | inprogress}`
`Last Latency`	The most recent probe result reported to the named real service.	`lastLatency` *`integer`*
`Average Latency`	The average latency to the named real service, which the switch recalculates with each returned probe.	`averageLatency` *`integer`*
`Weight`	The configured weight (static) or calculated weight (dynamic) for the named real service. The weight attribute is set with the `realService` command.	`weight` *`integer`*

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup → slbInfo

show serviceGroup slbinfo activation

Purpose

Displays information about standby activation.

Access mode

user

Syntax

```
show vSwitch-name loadBalance serviceGroup slbInfo activation
```

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup slbInfo
activation
Service Group Real Service      State      Initial Cause CurrentCause
-------------------------------------------------------------------------
sg1           rsPC-80           active       N/A         N/A
sg1           rsPC-81           active       N/A         N/A
sg1           rsPC-82           active       N/A         N/A
sg1           rsPC-83        notActive       none        none
sg1           rsPC-84        notActive       none        none
sg1           rsPC-85        notActive       none        none
```

Output description

Field name	Description	Filter name
Name	The text string that identifies the service group.	name *text*
Real Service Name	The text string that identifies the real service.	realService *text*
State	The activity state of the real service.	state *text*
Initial Cause	The initial cause of the standby activation.	initialCause *text*
Current Cause	The current cause of the standby activation.	currentCause *text*

Associated MIB

`serviceGroup.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup slbInfo activation

show serviceGroup slbInfo advanced counters

Purpose

Displays counters indicating the results of probes sent to the real services (servers) that are part of a service group.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup slbInfo advanced counters`

Sample output

```
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup slbInfo advanced
counters
Service Group Real Service     Probes    Successful  Unsuccessful       Passive
imageServers  rs1                   0             0             0             0
imageServers  rs2                   0             0             0             0
```

Output description

Field name	Description	Filter name
`Service Group`	The text string that identifies the service group.	`serviceGroupName` *`text`*
`Real Service`	The name of the real service, associated with this service group, that was the target of a probe.	`realServiceName` *`text`*
`Probes`	Total number of times the identified real service was probed.	`probes` *`integer`*
`Successful`	The number of times the targeted probe received a good response.	`successful` *`integer`*
`Unsuccessful`	The number of times the targeted probe failed to get a good response.	`unsuccessful` *`integer`*
`Passive`	The number of times the targeted probe was determined to be good using in-line statistics.	`passive` *`integer`*

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup → slbInfo → advanced → counters

show serviceGroup slbInfo advanced history

Purpose

Displays a summary of the probe history that the N2000 Series maintains for a specific target. The target is a real service (server) member of the service group. You can display the summary for all service groups or a specific service group.

Access mode

user

Syntax

```
show vSwitch-name loadBalance serviceGroup slbInfo advanced
history
```

Sample output

In the following sample, the previous poll cycles are represented as + for successful probes and - for unsuccessful probes.

```
sun(config-vSwitch-vsw1 loadBalance)# show serviceGroup slbInfo
advanced history

Service Group Real Service                        Probe History    Percent Good
------------- ------------     ---------------------------------   ------------
sg1           rs80            ++++++++++-++++++++++++++++++++             96
sg1           rs81            ---------------------------------           0
```

Output description

Field name	Description	Filter name
`Service Group`	The text string that identifies the service group.	`serviceGroupName` *`text`*
`Real Service`	The name of the real service, associated with this service group, that was the target of a probe.	`realServiceName` *`text`*
`Probe History`	A summary of the probe history window that the N2000 Series maintains.	`probeHistory` *`text`*
`Percent Good`	The percentage of good polls in the poll history window.	`percentGood` *`integer`*

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup → slbInfo → advanced → history

show serviceGroup slbInfo inline

Purpose

Displays statistics regarding in-line server health check results for the real services belonging to the named service group only when in-line health checks are enabled in the service group. The command output displays all real services configured for the service group, or can be further filtered to display only a specific real service.

In-line server health checks determine server health without waiting for the next out-of-band polling interval. If a server is unresponsive due to a failed TCP connection or HTTP request, the N2000 Series immediately removes the server as a participating load balancer within the service group, leaving the remaining active servers in the group to handle the traffic load or perform the configured forwarding action.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup slbInfo inline`

Sample output

```
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup slbInfo
inline
Service Group: imageServers
Real Service:  rs2
Il Status:     UP
Updates:       0
Adjusts:       0
Downs:         0
Successes:     0
Failures:      0
R S Activity:  0
```

Output description

Field name	Description	Filter name
`Service Group`	The text string that identifies the service group.	`serviceGroupName` *`text`*
`Real Service`	The text string that identifies the real service within the service group. If you do not enter a `realServiceName`, the command displays statistics for all real services in the named service group.	`realServiceName` *`text`*
`Il Status`	The in-line status of the named real service. Possible values are: `down`: There was no response to the probe. `dying`: The most recent probe(s) were unanswered, but the retry count has not yet been exceeded. `up`: Probe returned and confirmed availability. `ilshcDown`: Inline health check returned down status. `inProgress`: Probe returned and the real service is configuring.	`Status {down | dying | up | ilshcDown |inProgress}`
`Updates`	The number of in-line health check updates that the real service received.	`ilUpdates` *`integer`*
`Adjusts`	The number of real service weight adjustments that the real service received.	`ilAdjusts` *`integer`*
`Downs`	The number of in-line health check Out of Service updates that the real service received.	`ilDowns` *`integer`*
`Successes`	The number of reported successes in all in-line updates.	`successes` *`integer`*
`Failures`	The number of server health failures that the real service received.	`IlFailures` *`integer`*

Field name	Description	Filter name
R S Activity	The number of in-line updates that showed real service activity.	RSActivity *integer*

Additional filters

You can use the following additional filters with this command.

```
lastLatency integer
averageLatency integer
weight integer
probes integer
Time integer
IlRepWeight integer
IlRepStatus integer
LastInt integer
AvgInt integer
MinInt integer
MaxInt integer
successful integer
unsuccessful integer
probeHistory text
PercentGood text
Passive text
ActivityTime integer
UpdateTime integer
CurrPassive integer
CommErrors integer
ActMinTime integer
```

Associated MIB

shc.mib

Web path

- vSwitch → *name* → LoadBalance → serviceGroup → slbInfo → inline

show serviceGroup slbInfo interval

Purpose

Displays the operational probe intervals and statistics associated with the interval. The command output displays all real services configured for the service group, or can be further filtered to display only a specific real service.

Access mode

user

Syntax

```
show vSwitch-name loadBalance serviceGroup slbInfo interval
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup slbInfo
interval
```

Service Group	Real Service	Probes	Last Int	Avg Int	Min Int	Max Int
imageServers	rs1	0	0	0	0	0
imageServers	rs2	0	0	0	0	0

Output description

Field name	Description	Filter name
`Service Group`	The text string that identifies the service group.	`serviceGroupName` *`text`*
`Real Service`	The text string that identifies the real service within the service group. If you do not enter a `realServiceName`, the command displays statistics for all real services in the named service group.	`realServiceName` *`text`*
`Probes`	The number of health check probes that were sent to the real service.	`probes` *`integer`*
`Last Int`	The last probe interval.	`lastInt` *`integer`*
`Avg Int`	The average probe interval recorded .	`avgInt` *`integer`*
`Min Int`	The minimum probe interval recorded.	`minInt` *`integer`*
`Max Int`	The maximum probe interval recorded.	`maxInt` *`integer`*

Additional filters

You can also use the following filters with this command:.

```
lastLatency integer
averageLatency integer
weight integer
time integer
ilRepWeight integer
ilRepStatus integer
successful integer
unsuccessful integer
probeHistory text
percentGood text
passive text
activityTime integer
updateTime integer
currPassive integer
RSActivity integer
commErrors integer
actMinTime integer
```

Associated MIB

`shc.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup → slbInfo → interval

show serviceGroup slbinfo script status

Purpose

Displays the status of the scripted health check probe.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup slbInfo script status`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup slbInfo
script status
```

```
Service Group   Real Service              Script Status
sg1             rs80                           scriptOK
sg1             rs81                                  1
```

In the above sample, the defined script status, `scriptOK`, indicates that the script exited without errors or had the error code explicitly set to `success-0`. For the second real service, the script status, `1`, indicates that the script encountered an error.

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the service group.	`name` *`text`*
`Real Service Name`	The text string that identifies the real service	`realService` *`text`*
`Script Status`	The error code returned from the script.	`scriptStatus` *`text`*

Associated MIB

`serviceGroup.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup slbInfo script status

show serviceGroup slbinfo standby

Purpose

Displays the activity of standby devices.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup slbInfo standby`

Sample output

```
sun> enable
sun# configure
sun(config)# vSwitch SVS1
sun(config-vSwitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# show serviceGroup slbInfo
standby
```

```
Service Group   Real Service     RS Role   Status        State   Standby Seconds
sg-lnx1-HT-0    rs1 UP           primary       UP       active                 -
sg-lnx1-HT-0    rs2 UP           primary       UP       active                 -
sg-lnx1-HT-0    rs3 UP           primary       UP    notactive                 0
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the service group.	`name` *`text`*
`Real Service Name`	The text string that identifies the real service.	`realService` *`text`*
`RS Role`	The role of the real service.	`RSRole` *`text`*
`Status`	Status of the real service. Status is reported as: `down`: There was no response from to the probe. `dying`: The most recent probes were unanswered, but the retry count has not yet been exceeded. `up`: The probe returned and confirmed availability.	`status {down | dying | up}`
`State`	The activity state of the real service.	`state` *`text`*
`Standby Seconds`	The duration of the current activity state for standby devices.	`standbySeconds` *`integer`*

Associated MIB

`serviceGroup.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup slbInfo standby

show serviceGroup statistics realServiceSummary

Purpose

Displays statistics associated with the named real service.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup statistics realServiceSummary`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup
statistics realServiceSummary
Name:                   group1TCP
Real Service Name:      IxiaRSTCP2
Peak Active Sessions:   1
Requests Forwarded:     0
Responses Received:     0
Connect Failures:       0
Write Failures:         0
Read Failures:          0
Invalid HTTP Responses: 0
Current Pooled:         0
Num Pooled:             0
```

Output description

Field name	Description	Filter name
`name`	The text string that identifies the service group.	`name` *`text`*
`Real Service Name`	The text string that identifies the real service.	`realService` *`text`*
`Peak Active Sessions`	The greatest number of sessions open at one time on this real service. The count does not include sessions in the Time-Wait state, which are included in the Current Open Sessions count.	`peakActiveSessions` *`integer`*
`Requests Forwarded`	The number of HTTP requests sent to this real service.	`requestsForwarded` *`integer`*
`Responses Received`	The number of HTTP responses received from this real service.	`responsesReceived` *`integer`*
`Connect Failures`	The number of L4 failures that occurred when connecting to this real service.	`connectFailures` *`integer`*
`Write Failures`	The number of L4 failures when writing to this real service.	`writeFailures` *`integer`*
`Read Failures`	The number of L4 failures when reading from this real service.	`readFailures` *`integer`*
`Invalid HTTP Responses`	The number of invalid HTTP responses received from this real service.	`invalidHTTPResponses` *`integer`*
`Current Pooled`	The total number of currently opened pooled connections on this real service.	`currentPooled` *`integer`*
`Num Pooled`	The total number of pooled connections used (per client connection) for this real service.	`numPooled` *`integer`*

Associated MIB

`serviceGroup.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup statistics real service summary

show serviceGroup statistics summary

Purpose

Displays statistics about the number of requests handled by this service group in which there were no real services available.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance serviceGroup statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup
statistics summary
Name:                          group1TCP
slb Table Empty Counts:        0
```

Output description

Field name	Description	Filter name
`name`	The text string that identifies the service group.	`name` *`text`*
`slb Table Empty Counts`	The total number of times that a real service could not be found in the SLB table to perform load balancing.	`slbTableEmpty` *`integer`*

Associated MIB

`serviceGroup.mib`

Web path

- vSwitch → *name* → LoadBalance → serviceGroup statistics summary

show sorryData

Purpose

Displays the attributes of a named sorry data definition.

Access mode

user

Syntax

```
show vSwitch-name loadBalance sorryData
```

Sample output

```
sun(config-vSwitch-e-commerce loadBalance)# show sorryData so_1
Name:              so_1
Action:            redirect
Action String:     http://www.e-commerce.com/default/contact.htm/
```

Output description

Field name	Description	Filter name
Name	The text string assigned to the sorry data definition.	name *text*
Action	The action the load balancer should take when it finds a match to an associated request policy or response policy. Possible actions are: • close • redirect • reset • page	action {close | redirect | reset | page}
Action String	A text string that specifies information to return to the client depending on the action parameter.	actionString *text*

Associated MIB

`sorryData.mib`

Web path

- vSwitch → *name* → LoadBalance → sorryData

show summary

Purpose

Displays the active and inactive virtual services configured on the vSwitch.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance summary`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show summary
Total Virtual Services:    4
Active Virtual Services:   4
Inactive Virtual Services: 0
```

Output description

Field name	Description	Filter name
`Total Virtual Services`	The total number of virtual services configured on this vSwitch.	`totalVirtualServices` *`integer`*
`Active Virtual Services`	The number of virtual services configured on this vSwitch with an operational status of `active`.	`activeVirtualServices` *`integer`*
`Inactive Virtual Services`	The number of virtual services configured on this vSwitch with an operational status of `misconfigured`, `pending`, `unavailable`, `disabled`, or `active backup`.	`inactiveVirtualServices` *`integer`*

Associated MIB

`virtsvc.mib`

Web path

- vSwitch → *name* → LoadBalance → summary

show tideRunner congestion status

Purpose

Displays congestion statistics on a specific function card (TideRunner). Use this information to determine whether the vSwitch needs to be reconfigured with additional resources, or whether the vSwitch is under TCP attack.

All statistics are collected at each TideRunner statistics poll. At each poll interval, the current and previous values are compared to detect transitions between usage levels. Events are issued as the usage levels increase and decrease and the severity of the events change depending on absolute current usage level.

In the "Sample output," the first set of displayed fields (`clTcb` through `svrStrm`) shows the resource usage level statistics as a range. The second set of displayed fields (`clMaxTCBPct` through `svrMaxStrmPct`) shows the resource usage level statistics as integers that allow the NMON alarm group to monitor the TideRunner congestion table.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance tideRunner congestion status`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show tideRunner congestion
status
Card:          functionCard1
clTcb:         0-50%
clPage:        0-50%
clStrm:        0-50%
svrTcb:        0-50%
svrPage:       0-50%
svrStrm:       0-50%
clMaxTcbPct:   50
clMaxPagePct:  50
clMaxStrmPct:  50
svrMaxTcbPct:  50
svrMaxPagePct: 50
svrMaxStrmPct: 50
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`

`clTcb`	Client Transmission Control Block (TCB). The TCB is a 256-byte structure that TideRunner uses to maintain TCP connection state; one TCB is used per TCP connection. This statistic reflects the TCB resource usage level for this class (vSwitch, client resources). The usage level is represented as a range, using a logarithmic scale, and can take one of the following values:
	
	`0-50%`
	`50-75%`
	`75-88%`
	`88-94%`
	`94-97%`
	`97-98%`
	`>= 99%`
		`clTcb` *`text`*

`clPage`	Client Page. A page is a unit of physical memory that is used to hold TCP stream data from clients. The `clPage` statistic reflects the current usage of pages set aside for client data by this vSwitch.	`clPage` *`text`*

Field Name	Description	Filter Name
`clStrm`	Client Stream. A stream resource is a data structure referring to 0 or more pages used to persist received terminated TCP application data from clients. The `clStrm` statistic reflects the current usage of TideRunner stream resources that are used for TCP connections to clients.	`clStrm` *`text`*

`svrTcb`	Server TCB. The TCB is a 256-byte structure that TideRunner uses to maintain TCP connection state; one TCB is used per TCP connection. This statistic reflects the TCB resource usage level for this class (vSwitch, server resources). The usage level is represented as a range, using a logarithmic scale, and can take one of the following values:
	
	`0-50%`
	`50-75%`
	`75-88%`
	`88-94%`
	`94-97%`
	`97-98%`
	`>= 99%`
		`svrTcb` *`text`*

`svrPage`	Server Page. A page is a unit of physical memory used to hold TCP stream data from servers. The default page size is 2 Kbytes. The `svrPage` statistic reflects the current usage of pages set aside for server data by this vSwitch.	`svrPage` *`text`*
`svrStrm`	Server Stream. A stream resource is a data structure referring to 0 or more pages that are used to persist received terminated TCP application data from servers. The `svrStrm` statistic reflects the current usage of TideRunner stream resources that are used for TCP connections to servers.	`svrStrm` *`text`*

Field Name	Description	Filter Name

`clMaxTcbPct`	Client Maximum TCB Percentage. This resource usage level is represented by the `clTcb` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`clMaxTcbPct` *`integer`*

`clMaxPagePct`	Client Maximum Page Percentage. This resource usage level is represented by the `clPage` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`clMaxPagePct` *`integer`*

Field Name	Description	Filter Name

`clMaxStrmPct`	Client Maximum Stream Percentage. This resource usage level is represented by the `clStrm` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`clMaxStrmPct` *`integer`*

`svrMaxTcbPct`	Server Maximum TCB Percentage. This resource usage level is represented by the `svrTcb` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`svrMaxTcbPct` *`integer`*

Field Name	Description	Filter Name
svrMaxPagePct	Server Maximum Page Percentage. This resource usage level is represented by the svrPage end-of-range value and can take on one of the following values: 0 50 75 88 94 97 98 100 This integer field can be monitored by NMON.	svrMaxPagePct *integer*
svrMaxStrmPct	Server Maximum Stream Percentage. This resource usage level is represented by the svrStrm end-of-range value and can take on one of the following values: 0 50 75 88 94 97 98 100 This integer field can be monitored by NMON.	svrMaxStrmPct *integer*

Associated MIB

tteCounters.mib

Web path

- vSwitch → *name* → LoadBalance → tideRunner → congestion→ status

show tideRunner realService statistics

Purpose

Displays statistics for real service operations on a specific function card (TideRunner). These statistics are equivalent to the statistics you can view with the `show statistics summary` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance tideRunner realService statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show tideRunner
realService statistics
Card:              functionCard1
ConnsActive:       1
ActiveOpens:       16
ActiveDrops:       14
PassiveOpens:      0
PassiveDrops:      1
ConnFailures:      0
RcvTotalSegs:      80
RcvCtrl:           35
RcvOOSegs:         0
RcvBytes:          53193
RcvDrop:           0
RcvAfterWin:       0
RcvWinProbe:       0
RcvAckTooMuch:     0
RcvWinUpd:         0
SndTotalSegs:      81
SndCtrl:           62
SndRexmtSegs:      0
SndRttSegs:        0
SndBytes:          4896
SndRstSegs:        14
SndDelAck:         11
```

```
RexmtTimeout:      0
PersistTimeout:    0
InActivityTimeout: 1
RcvOrphanDrops:    0
CngSynDiscard:     0
CngAccTimeOut:     0
CngConnAborts:     0
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`
`ConnsActive`	The number of connections currently active on the function card.	`connsActive` *`integer`*
`ActiveOpens`	The number of active connections that the function card initiated.	`activeOpens` *`integer`*
`ActiveDrops`	The number of connections that the function card closed. The closed connections are ones that the function card initiated.	`activeDrops` *`integer`*
`PassiveOpens`	The number of connections that a remote client or server initiated.	`passiveOpens` *`integer`*
`PassiveDrops`	The number of HTTP connections that a remote server closed.	`passiveDrops` *`integer`*
`ConnFailures`	The total number of connection attempts that failed.	`connFailures` *`integer`*
`RcvTotalSegs`	The total number of TCP segments that the function card received.	`rcvTotalSegs` *`integer`*

Field Name	Description	Filter Name
RcvCtrl	The total number of control segments that the function card received. A control segment has no data payload.	rcvCtrl *integer*
RcvOOSegs	The total number of TCP segments that the function card received out of order or as a duplicate segment.	rcvOOSegs *integer*
RcvBytes	The total number of bytes that the function card received that contain the data payload, excluding TCP headers.	rcvBytes *integer*
RcvDrop	The total number of TCP segments that the function card dropped. This value does not include the number of received error segments (duplicate illegal, or orphan segments that the function card cannot process).	rcvDrop *integer*
RcvAfterWin	The total number of TCP packets that the function card received after a TCP receive window closed (the function card cannot receive additional data until it reopens the window). The function card drops these packets.	rcvAfterWin *integer*
RcvWinProbe	The total number of window probe packets that the function card received.	rcvWinProbe *integer*
RcvAckTooMuch	The total number of acknowledgements (ACKs) that the function card received for data that it has not yet received.	rcvAckTooMuch *integer*
RcvWinUpd	The total number of TCP window update packets that the function card received.	rcvWinUpd *integer*
SndTotalSegs	The total number of TCP segments that the function card transmitted to a remote client or server, excluding retransmitted segments.	sndTotalSegs *integer*
SndCtrl	The total number of control frames sent to a remote client or server. A control segment does not include a TCP message with data.	sndCtrl *integer*

Field Name	Description	Filter Name
SndRexmtSegs	The total number of TCP segments that the function card retransmitted. Segment retransmission occurs if the remote client or server does not send an acknowledgement (ACK) to the function card.	sndRexmtSegs *integer*
SndBytes	The total number of data payload bytes that the function card transmitted to a remote client or server, including retransmission. This value does not include TCP header bytes.	sndBytes *integer*
SndRstSegs	The total number of reset segments that the function card transmitted to a remote client or server.	sndRstSegs *integer*
SndRttSegs	The total number of round-trip time segments that the function card segments to a remote client or server. The function card uses the round-trip time measurements to avoid traffic congestion.	sndRttSegs *integer*
SndDelAck	The total number of delayed acknowledgements (ACKs) that the function card sent to a remote client or server. Using delayed ACKs allows the system to send a packet acknowledgement and response at the same time, increasing performance.	sndDelAck *integer*
RexmtTimeout	The total number of times that the function card reset a connection because it exceeded the number of times it could try to retransmit a TCP segment.	rexmtTimeout *integer*
PersistTimeout	The total number of times that the function card sent window probes to a remote client or server to determine whether a TCP window was open and could receive data.	persistTimeout *integer*

Field Name	Description	Filter Name
`InActivityTimeout`	The total number of times that a connection timed out because of inactivity. The function card closes a connection if it does not receive an acknowledgement from a connection after sending multiple keep-alive probes.	`inActivityTimeout` *`integer`*
`RcvOrphanDrops`	The total number of orphan segments that the function card dropped because it did not have the space to store these segments.	`rcvOrphanDrops` *`integer`*
`CngSynDiscard`	The number of SYN packets that were dropped due to congestion.	`CngSynDiscard` *`integer`*
`CngAccTimeOut`	The number of retransmit or inactivity time-out accelerations that were due to congestion.	`CngAccTimeOut` *`integer`*
`CngConnAborts`	The number of connections that were aborted due to congestion.	`CngConnAborts` *`integer`*

Associated MIB

`tteCounters.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → realService → statistics

show tideRunner realService sslStatistics

Purpose

Displays Secure Sockets Layer (SSL) statistics for real service operations on a specific function card (TideRunner). The statistics that this command displays are equivalent to the statistics you can display with the `show statistics group sslRecord` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance tideRunner realService sslStatistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show tideRunner
realService sslStatistics
Card:                                 functionCard1
Number Of Records Encrypted:          23977428
Number Of Bytes Encrypted:            34355378230
Number Of Errored Records Encrypted: 0
Number Of Records Decrypted:          3307083
Number Of Bytes Decrypted:            185858130
Number Of Errored Records Decrypted: 0
```

Output description

Field name	Description	Filter name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1|functionCard2}`
`Number of Records Encrypted`	The total number of SSL records that the function card encrypted for real services.	`encRecords` *`integer`*
`Number of Bytes Encrypted`	The total number of SSL bytes that the function card encrypted for real services.	`encBytes` *`integer`*
`Number of Records Decrypted`	The total number of SSL records that the function card decrypted for real services.	`decRecords` *`integer`*
`Number of Bytes Decrypted`	The total number of SSL bytes that the function card decrypted for real services.	`decBytes` *`integer`*
`Number of Encrypt Errors`	The total number of SSL records with errors that the function card encrypted for real services.	`encRecordsErr` *`integer`*
`Number of Decrypt Errors`	The total number of SSL records with errors that the function card decrypted for real services.	`decRecordsErr` *`integer`*

Associated MIB

`tteCounters.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → realService → sslStatistics

show tideRunner virtualService statistics

Purpose

Displays statistics for virtual service operations on a specific function card (TideRunner). These statistics are equivalent to the statistics you can display with the `show statistics summary` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance tideRunner virtualService statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show tideRunner
virtualService statistics
Card:              functionCard1
ConnsActive:       0
ActiveOpens:       0
ActiveDrops:       25
PassiveOpens:      30
PassiveDrops:      5
ConnFailures:      0
RcvTotalSegs:      290
RcvCtrl:           245
RcvOOSegs:         0
RcvBytes:          10970
RcvDrop:           0
RcvAfterWin:       0
RcvWinProbe:       0
RcvAckTooMuch:     0
RcvWinUpd:         0
SndTotalSegs:      370
SndCtrl:           119
SndRexmtSegs:      0
```

(continued)

```
SndRttSegs:         0
SndBytes:           341612
SndRstSegs:         23
SndDelAck:          6
RexmtTimeout:       0
PersistTimeout:     0
InActivityTimeout:  3
RcvOrphanDrops:     0
CngSynDiscard:      0
CngAccTimeOut:      0
CngConnAborts:      0
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`
`ConnsActive`	The number of connections currently active on the function card.	`connsActive` *`integer`*
`ActiveOpens`	The number of active connections that the function card initiated.	`activeOpens` *`integer`*
`ActiveDrops`	The number of connections that the function card closed. The closed connections are ones that the function card initiated.	`activeDrops` *`integer`*
`PassiveOpens`	The number of connections that a remote client or server initiated.	`passiveOpens` *`integer`*
`PassiveDrops`	The number of HTTP connections that a remote server closed.	`passiveDrops` *`integer`*
`ConnFailures`	The total number of connection attempts that failed.	`connFailures` *`integer`*

Field Name	Description	Filter Name
RcvTotalSegs	The total number of TCP segments that the function card received.	rcvTotalSegs *integer*
RcvCtrl	The total number of control segments that the function card received. A control segment has no data payload.	rcvCtrl *integer*
RcvOOSegs	The total number of TCP segments that the function card received out of order or as a duplicate segment.	rcvOOSegs *integer*
RcvBytes	The total number of bytes that the function card received that contain the data payload, excluding TCP headers.	rcvBytes *integer*
RcvDrop	The total number of TCP segments that the function card dropped. This value does not include the number of received error segments (duplicate illegal, or orphan segments that the function card cannot process).	rcvDrop *integer*
RcvAfterWin	The total number of TCP packets that the function card received after a TCP receive window closed (the function card cannot receive additional data until it reopens the window). The function card drops these packets.	rcvAfterWin *integer*
RcvWinProbe	The total number of window probe packets that the function card received.	rcvWinProbe *integer*
RcvAckTooMuch	The total number of acknowledgements (ACKs) that the function card received for data that it has not yet received.	rcvAckTooMuch *integer*
RcvWinUpd	The total number of TCP window update packets that the function card received.	rcvWinUpd *integer*
SndTotalSegs	The total number of TCP segments that the function card transmitted to a remote client or server, excluding retransmitted segments.	sndTotalSegs *integer*
SndCtrl	The total number of control frames sent to a remote client or server. A control segment does not include a TCP message with data.	sndCtrl *integer*

Field Name	Description	Filter Name
SndRexmtSegs	The total number of TCP segments that the function card retransmitted. Segment retransmission occurs if the remote client or server does not send an acknowledgement (ACK) to the function card.	sndRexmtSegs *integer*
SndBytes	The total number of data payload bytes that the function card transmitted to a remote client or server, including retransmission. This value does not include TCP header bytes.	sndBytes *integer*
SndRstSegs	The total number of reset segments that the function card transmitted to a remote client or server.	sndRstSegs *integer*
SndRttSegs	The total number of round-trip time segments that the function card segments to a remote client or server. The function card uses the round-trip time measurements to avoid traffic congestion.	sndRttSegs *integer*
SndDelAck	The total number of delayed acknowledgements (ACKs) that the function card sent to a remote client or server. Using delayed ACKs allows the system to send a packet acknowledgement and response at the same time, increasing performance.	sndDelAck *integer*
RexmtTimeout	The total number of times that the function card reset a connection because it exceeded the number of times it could try to retransmit a TCP segment.	rexmtTimeout *integer*
PersistTimeout	The total number of times that the function card sent window probes to a remote client or server to determine whether a TCP window was open and could receive data.	persistTimeout *integer*

Field Name	Description	Filter Name
`InActivityTimeout`	The total number of times that a connection timed out because of inactivity. The function card closes a connection if it does not receive an acknowledgement from a connection after sending multiple keep-alive probes.	`inActivityTimeout` *`integer`*
`RcvOrphanDrops`	The total number of orphan segments that the function card dropped because it did not have the space to store these segments.	`rcvOrphanDrops` *`integer`*
`CngSynDiscard`	The number of SYN packets that were dropped due to congestion.	`CngSynDiscard` *`integer`*
`CngAccTimeOut`	The number of retransmit or inactivity time-out accelerations that were due to congestion.	`CngAccTimeOut` *`integer`*
`CngConnAborts`	The number of connections that were aborted due to congestion.	`CngConnAborts` *`integer`*

Associated MIB

`tteCounters.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → virtualService → statistics

show tideRunner virtualService sslStatistics

Purpose

Displays Secure Sockets Layer (SSL) statistics for virtual service operations on a specific function card (TideRunner).

Access mode

user

Syntax

```
show vSwitch-name loadBalance tideRunner virtualService
sslstatistics
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show tideRunner
virtualService sslStatistics
Card:                                functionCard1
Number Of Records Encrypted:         23977428
Number Of Bytes Encrypted:           34355378230
Number Of Records Decrypted:         0
Number Of Bytes Decrypted:           3307083
Number Of Encrypt Errors:            185858130
Number Of Decrypt Errors:            0
```

Output description

Field name	Description	Filter name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1| functionCard2}`
`Number of Records Encrypted`	The total number of SSL records that the function card encrypted for virtual services.	`encRecords` *`integer`*
`Number of Bytes Encrypted`	The total number of SSL bytes that the function card encrypted for virtual services.	`encBytes` *`integer`*
`Number of Records Decrypted`	The total number of SSL records that the function card decrypted for virtual services.	`decRecords` *`integer`*
`Number of Bytes Decrypted`	The total number of SSL bytes that the function card decrypted for virtual services.	`decBytes` *`integer`*
`Number of Encrypt Errors`	The total number of SSL records with errors that the function card encrypted for virtual services.	`encRecordsErr` *`integer`*
`Number of Decrypt Errors`	The total number of SSL records with errors that the function card decrypted for virtual services.	`decRecordsErr` *`integer`*

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → virtualService → sslStatistics

show virtualService

Purpose

Displays the configuration of the virtual services on the system. These are either default values or values set with the `virtualService` command. In addition, there are two fields, `Oper Status` and `Oper Message`, that report configuration status. The fields that are displayed are dependent on the `virtualService appServiceType` argument as certain fields are only relevant for particular `appServiceTypes`. See the "Arguments" table for the `virtualService` command for the relevant arguments for each `appServiceType`.

The following operational messages indicate specific configuration problems:

- `Virtual Service disabled`
- `Request Policy not configured`
- `Service Group not configured`
- `Object Rule not configured`
- `Real Service not configured`
- `Host not configured`
- `H/W Config error`
- `Object Rule compilation error`
- `Invalid vRouter`
- `Configuration error (this indicates an undetermined error)`
- `Health check profile not configured`
- `Proxy IP pool not configured`
- `Disabled vRouter`
- `vSwitch disabled`
- `Cookie Persistence not configured`
- `Sorry Data not configured`
- `SE Bandwidth not configured`
- `SSL certificate unavailable`

Note: An operational status of `active` (displayed in the `Oper Status` field) does not ensure end-to-end connectivity. The status is only reporting the validity of the hierarchical configuration.

Access mode

user

Syntax

```
show vSwitch-name loadBalance virtualService
```

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch SVS1
sun(config-vswitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# show virtualService
Name:                                vs1
Service Type:                        HTTP
IP Address:                          11.8.201.19
Request Policy List:                 qp1; qp2
Request Transform List:              rqt1; rqt2
Description:                         vs1
Admin State:                         enabled
Disable Delay:                       0
Protocol:                            TCP
Port:                                80
vRouter:                             system:shared
Host Name:                           web_svr
Client Source IP Address Range:      0.0.0.0-255.255.255.255
SYN Rate Limit:                      unlimited
Oper Status:                         active
Oper Message:                        Operational
Function Card:                       1
Total Real Services:                 4
Active Real Services:                4
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the virtual service.	`name` *`text`*
`Service Type`	The application service type for the virtual service. Possible values are: `L4SLB`: Layer 4 server load balancing (stateless). This is the only non-TCP-terminated service type. The only load-balancing algorithm allowed with this service type is `weightedHash`, set with the `serviceGroup` command. `L4SLB_ADV`: TCP-terminated Layer 4 server load balancing. This application service type exploits advanced functionality implemented in the FX-SLB and FX-SSL function cards. This service type can take advantage of more sophisticated service selection algorithms leading to improved overall site availability and performance. (*Requires FX-SLB or FX-SSL function card*.) `L4SLB_SSL`: Provides the same hardware advantages as `L4SLB_ADV`, but with the added capability of SSL termination and re-origination. If selecting this type, you must also provide the `ckmKeyName`. (*Requires the FX-SSL function card*.) `HTTP`: Provides hardware TCP-termination to capture the entire HTTP application byte stream for Layer 5 through Layer 7 switching decisions based on objects within the session. (*Requires FX-SLB or FX-SSL function card*.) `HTTPS`: Adds SSL functionality to the HTTP capability. If selecting this type, you must also provide the `ckmKeyName`. (*Requires FX-SSL function card*.) `TDLB`: Provides Transparent Device load balancing. `FTPLB`: Provides FTP load balancing. `RTSPLB`: Provides RTSP load balancing.	`appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB | FTPLB | RTSPLB}`

Field name	Description	Filter name
IP Address or IP Address Range	The virtual IP (VIP) address that the NAT configuration is assigning to the hosts within this virtual service. The VIP address range assigned to the hosts if the appServiceType is TDLB.	ipAddress *ipAddress* ipAddress{*ipAddress-ipAddress*}
Service Group Name	The name of the service group associated with this virtual service.	serviceGroupName *text*
Request Policy List	The request policies the system is applying to this virtual service.	requestPolicyList *text*
Request Transform List	List of request transforms to control requests from clients.	requestTransformList *text*
Description	A text description of the virtual service.	description *text*
Admin State	The administrative state of the named virtual service, either enabled or disabled.	adminState {enabled | disabled}
Disable Delay	The time, in seconds, after a virtual service is disabled in which new sessions cannot be established but existing sessions remain up and are allowed to gracefully terminate. At the end of the disableDelay time, all connections will be reset.	disableDelay {*seconds* | unlimited}
Protocol	The protocol used by this virtual service, either TCP or UDP.	protocol {TCP | UDP}
Port	The port that the virtual service is using to listen for connections of the type specified with the protocol argument.	port *portNumber*
vRouter	The vRouter that traffic is coming in on to reach this virtual service.	vRouter *text*
Client Source IP Address Range	The specified group of clients allowed access to the virtual service.	clientSrcIPRange *ipAddress*

Field name	Description	Filter name
SYN Rate Limit	The number of TCP SYN packets that can be received in a period of 1 second. SYN packets signal the start of a TCP connection, and by limiting them you can institute a form of congestion control. When the limit is reached, subsequent packets are dropped. Valid values for the rate limit are either `unlimited`, which takes no action regardless of the number of packets received, or a static integer between 10 and 300000. The default setting is `unlimited`.	`synRateLimit {`*`integer`*` | unlimited}`
Persist Type	The algorithm used for maintaining session persistence.	`persistType {none | srcAddress}`
SRC Address Persist Mask	Mask for source IP address-based persistence.	`srcAddressPersistMask` *`ipAddress`*
FTP Data Port Range	The range of ports to be used for the FTP data service.	`ftpDataPortRange` *`portNumber`*
FTP Data RX Timer Short Timeout Value	The time to wait to receive the first data packet from a real service after a session has been established. If the timer expires, the switch terminates the session.	`ftpDataRxTimerShortTimeoutValue` *`integer`*`_seconds`
FTP Data RX Timer Long Timeout Value	The time to wait to receive subsequent data packets from a client after the initial packet is received. If the timer expires, the switch terminates the session.	`ftpDataRxTimerLongTimeoutValue` *`integer`*`_seconds`

Field name	Description	Filter name
`Oper Status`	The operational status of the virtual service configuration. Possible values are: `active`: The virtual service has a complete configuration, which means that all lower levels also have been properly configured. `misconfigured`: The configuration is not complete and therefore the virtual service cannot become active. In this case, the `Oper Message` field lists the configuration components that are still required or invalid. `pending`: The virtual service is being configured and is transitioning to the active state. `activeBackup`: The virtual service is a redundant virtual service, and it is in backup mode. `unavailable`: All real services that are part of this virtual service are unavailable as detected by health checks. `disabled`: The virtual service `adminState` has been set to `disabled`.	`operStatus {active | misconfigured | pending | activeBackup | unavailable | disabled}`
`Oper Message`	The message(s) identifying the pieces of the configuration that are either missing or invalid. If the `operStatus` is `active`, this field displays `operational`. See the list of messages in the "Purpose" section of this command description.	`operMessage` *`text`*
`VS Group`	The named virtual service group in which this virtual service resides. The named virtual services included in the group can share their request policies, service groups, and real services with each other. You can create up to 64 virtual service groups per vSwitch. However, a named virtual service can only be included in one virtual service group.	`vsGroup` *`text`*

Field name	Description	Filter name
`Function Card`	The number of the slot that holds the optional module on which the virtual service configuration is stored.	`functionCard {1 | 2}`
`VSRP Redirect Address`	The IP address to which packets are forwarded when the virtual service is configured as a VSRP backup. All packets addressed to the backup virtual service's IP address are redirected to this IP address, and forwarded to the virtual service in the active switch.	`vsrpRedirectAddress` *`ipAddress`*
`Total Real Services`	The total number of real services configured on this virtual service.	`totalRealServices` *`integer`*
`Active Real Services`	The number of real services configured on this virtual service with an operational status of `active`.	`activeRealServices` *`integer`*
`Inactive Real Services`	The number of real services configured on this virtual service with an operational status of `inactive` or `unavailable`.	`inactiveRealServices` *`integer`*

Associated MIB

`virtSvc.mib`

Web path

- vSwitch→ *name* → virtualService

show virtualService advanced

Purpose

Displays specific TCP settings for the client-to-switch connection. These settings were configured with the `virtualService advanced` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance virtualService advanced`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show virtualService
advanced
Name:                          e-commerceNet
TCB Table Template Key:        0
IP Type Of Service:            Normal
Retransmit Retry Limit:        4
Establishment Retry Limit:     4
RX Timer Short Timeout Value: 32_seconds
RX Timer Long Timeout Value:  64_seconds
Receive Window Size:           20480
Round Trip Time:               1500_msec
SMM Stream Limit:              1xRcvWnd
Est Short Timeout:             ExpRetr
Rx Window Disable:             false
rcvMss:                        1460
xmtMss:                        1460
enableHttpMode:                false
initParseWithData:             false
Use Long RX Timer:             false
```

Output description

Field name	Description	Filter name
`Name`	The name of the configured virtual service whose advanced TCP connections you are viewing.	`name` *`text`*
`TCB Table Template Key`	The TCB table template key in use by this connection. The default, and only acceptable value, is 0.	`tcbTemplateKey` *`integer`*
`IP Type Of Service`	The bit set in the type of service (TOS) field of IP headers for traffic traversing this connection. This informs intervening routers how they should handle the packets. Possible values are: `Normal`: Used for ICMP, BOOTP, DNS, TCP queries traffic `MinCost`: Used for Usenet traffic `MaxReliablity`: Used for SNMP and routing traffic `MaxThroughput`: Used for FTP traffic `MinDelay`: Used for interactive login applications, such as Telnet and rlogin	`ipTOS {Normal | MinCost | MaxReliablity | MaxThroughput | MinDelay}`
`Retransmit Retry Limit`	The base interval for the retransmit retry limit within an established connection. That is, the time between retransmits of data sent and not acknowledged.	`xmtRetryLimit` *`integer`*
`Establishment Retry Limit`	The base interval for the establishment retry limit, which is how long you are willing to attempt a connection.	`estRetryLimit` *`integer`*
`RX Timer Short Timeout Value`	The receive timer short time-out value, which defines how long you are willing to wait to receive the first data packet from a real service after a session has been established.	`shortRxTimer` *`integer`*
`RX Timer Long Timeout Value`	The receive timer long time-out value, which defines how long you are willing to wait to receive subsequent data from the real service after the initial packet.	`longRxTimer` *`integer`*
`Receive Window Size`	The maximum amount of memory allocated to received data on a session.	`rcvWnd` *`integer`*

Field name	Description	Filter name
`Round Trip Time`	The maximum value for the packet round-trip transmit time in seconds or milliseconds.	`xmtRTT` *`roundTripValue`*
`SMM Stream Limit`	The Stream Memory Manager multiplier from the `Receive Window Size` field.	`smmStreamLimit` *`streamLimitValue`*
`Est Short Timeout`	The type of Establishment Time time-out; either the value from ExpRFC793 or ExpRetr.	`estShortTimout {ExRFC793 | ExpRetr}`
`Receive Window Disable`	Specifies whether TCP receive flow control is performed on a per-connection basis and advertised to the sender. The default setting is `false`.	`rcvWndDisable {true | false}`
`Rcv Mss`	Specifies the receive maximum segment size; broadcasted with a TCP SYN segment. The default setting is 1460.	`rcvMss` *`integer`*
`Xmt Mss`	Specifies the maximum segment size; advertised during the SYN handshake. The default setting is 1460	`xmtMss` *`integer`*
`Enable HTTP Mode`	Indicates whether HTTP mode is enabled, either `true` or `false`. This is a performance-tuning parameter that optimizes HTTP. The default setting is `false`.	`enableHttpMode {true | false}`
`initParseWith Data`	For client side connections, this setting affects HTTP and HTTPS performance by delaying processing until data is received. The default setting is `false`.	`initparseWithData {true | false}`
`Rx Use Long Time`	Specifies whether the internal receive progress timer is used to recover dormant connections. The default setting is `false`.	`rxUseLongTime {true | false}`
`Disable Syn Cookies`	Disable SYN cookie processing for this virtual service.	`disableSynCookies {true | false}`
`Client First Protocol`	All protocols using this virtual service will be client-initiated.	`clientFirstProtocol {true | false}`

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService → advanced

show virtualService ssl

Purpose

Displays the Secure Sockets Layer (SSL) configurations for the virtual services. These are either default values or values set with the `virtualService` command. In addition, there are two fields, `Oper Status` and `Oper Message`, that report configuration status.

Certain operational messages indicate specific configuration problems. See `show virtualService` for a listing of messages.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance virtualService ssl`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show virtualService ssl
Name:                        test
IP Address:                  10.10.30.1
Service Type:                HTTPS
Fully Qualified Host Name: tester
Oper Status:                active
Oper Message:                Request Policy not configured
Certificate And Key Name:   N/A
SSL Protocols:               SSLv3; TLSv1
IE Export Ciphers Support: enabled
Configured SSL Ciphers:     RSA_WITH_RC4_128_MD5; RSA_WITH_RC4_128_SHA;
                             RSA_WITH_AES_128_CBC_SHA;
RSA_WITH_3DES_EDE_CBC_SHA
SSL Ciphers In Use:          RSA_WITH_RC4_128_MD5; RSA_WITH_RC4_128_SHA;
                             RSA_WITH_AES_128_CBC_SHA;
                             RSA_WITH_3DES_EDE_CBC_SHA;
RSA_WITH_DES_CBC_SHA;
                             RSA_EXPORT1024_WITH_RC4_56_SHA;
                             RSA_EXPORT1024_WITH_DES_CBC_SHA;
                             RSA_EXPORT_WITH_RC4_40_MD5;
                             RSA_EXPORT_WITH_DES40_CBC_SHA
SSL Renegotiation Support: true
SSL SGC Support:             true
SSL Resumption Support:     true
```

Output Description

Field name	Description	Filter name
`Name`	The text string that identifies the virtual service.	`name` *`text`*
`IP Address`	The virtual IP (VIP) address that the NAT configuration is assigning to the hosts within this virtual service.	`ipAddress` *`ipAddress`*

Field name	Description	Filter name
`Service Type`	The application service type for the virtual service. Possible values are: `L4SLB`: Layer 4 server load balancing (stateless). This is the only non-TCP-terminated service type. The only load-balancing algorithm allowed with this service type is `weightedHash`, set with the `serviceGroup` command. `L4SLB_ADV`: TCP-terminated Layer 4 server load balancing. This application service type exploits advanced functionality implemented in the FX-SLB and FX-SSL function cards. This service type can take advantage of more sophisticated service selection algorithms leading to improved overall site availability and performance. (*Requires FX-SLB or FX-SSL function card*.) `L4SLB_SSL`: Provides the same hardware advantages as `L4SLB_ADV`, but with the added capability of SSL termination and re-origination. If selecting this type, you must also provide the `ckmKeyName`. (*Requires the FX-SSL function card*.) `HTTP`: Provides hardware TCP-termination to capture the entire HTTP application byte stream for Layer 5 through Layer 7 switching decisions based on objects within the session. (*Requires FX-SLB or FX-SSL function card*.) `HTTPS`: Adds SSL functionality to the HTTP capability. If selecting this type, you must also provide the `ckmKeyName`. (*Requires FX-SSL function card*.) `TDLB`: Provides Transparent Device load balancing. `FTPLB`: Provides FTP load balancing. `RTSPLB`: Provides RTSP load balancing.	`appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB | FTPLB | RTSPLB}`
`Fully Qualified Host Name`	The DNS (fully qualified) host name identifying the virtual service.	`hostname` *`text`*

Field name	Description	Filter name
`Oper Status`	The operational status of the virtual service configuration. Possible values are: `active`: The virtual service has a complete configuration, which means that all lower levels also have been properly configured. `misconfigured`: The configuration is not complete and therefore the virtual service cannot become active. In this case, the `operMessage` field lists the configuration components that are still required or invalid. `pending`: The virtual service is being configured and is transitioning to the active state. `activeBackup`: The virtual service is a redundant virtual service, and it is in backup mode. `unavailable`: All real services that are part of this virtual service are unavailable as detected by health checks. `disabled`: The virtual service `adminState` has been set to `disabled`.	`operStatus {active | misconfigured | pending | activeBackup | unavailable | disabled}`
`Oper Message`	The message(s) identifying the pieces of the configuration that are either missing or invalid. If the `operStatus` is `active`, this field displays `operational`. See the list of messages in the "Purpose" section of `show virtualService`.	`operMessage` *`text`*
`Certificate And Key Name`	The name (which serves as an index entry to the system) for the SSL key and certificate the system is using on the frontend connection.	`ckmKeyName` *`text`*

Field name	Description	Filter name
SSL Protocols	The SSL protocols supported by this virtual service. Possible values are: SSLv3: Secure Sockets Layer Protocol, Version 3.0 TLSv1: Transport Layer Security (TLS) Version 1.0, as defined in RFC 2246. (TLS is also known as SSL, Version 3.1.)	sslProto {SSLv3 | TLSv1}
IE Export Cipher Support	Indicates whether weak ciphers for import/export compatibility are enabled or disabled.	ieExportCiphersSupport {enabled | disabled}
Configured SSL Ciphers	The encryption methods (cipher suites) used on traffic passing through this virtual service. (See RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*, for detailed descriptions of each cipher.) Possible supported cipher suites include: RSA_EXPORT_WITH_RC4_40_MD5 RSA_WITH_RC4_128_MD5 RSA_WITH_RC4_128_SHA RSA_EXPORT_WITH_DES40_CBC_SHA RSA_WITH_DES_CBC_SHA RSA_WITH_3DES_EDE_CBC_SHA RSA_WITH_AES_128_CBC_SHA RSA_EXPORT1024_WITH_DES_CBC_SHA RSA_EXPORT1024_WITH_RC4_56_SHA	sslCiphers *ciphers*

Field name	Description	Filter name

`SSL Ciphers In Use`	List of SSL ciphers currently in use. This list differs from the list of configured ciphers if you have enabled `IE Export Cipher Support`; in that case the IE Export ciphers are listed. Possible values are:
	
	`RSA_EXPORT_WITH_RC4_40_MD5`
	
	`RSA_WITH_RC4_128_MD5`
	
	`RSA_WITH_RC4_128_SHA`
	
	`RSA_EXPORT_WITH_DES40_CBC_SHA`
	
	`RSA_WITH_DES_CBC_SHA`
	
	`RSA_WITH_3DES_EDE_CBC_SHA`
	
	`RSA_WITH_AES_128_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_DES_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_RC4_56_SHA`
		`sslCiphersInUse` *`ciphers`*

`SSL Renegotiation Support`	This virtual service's support for SSL renegotiation, either `true` or `false`. When `true`, the system allows SSL renegotiation (multiple handshakes to occur over a single SSL connection). When `false`, the system ignores SSL renegotiation attempts from the SSL client. The default is `true`. SSL renegotiation may occur when dealing with SSL clients that would like to rekey periodically or when dealing with Web browsers attempting to use Netscape's Step-Up (a technique for upgrading from weak cryptography to strong cryptography over an existing SSL connection).	`reneg {true | false}`

Field name	Description	Filter name
`SSL SGC Support`	This virtual service's support for Microsoft's Server Gated Cryptography (SGC). SGC, like Netscape's Step-Up, is a technique for upgrading from weak cryptography to strong cryptography over an existing SSL connection. This proprietary variation on the SSL handshake violates the SSL draft and the TLS RFC, and it is only done by Microsoft Web browsers when speaking to SSL servers with special SGC certificates. When `sgcSupport` is `true`, the system allows clients to do SGC handshakes. When `false`, the system does not allow SGC, and will drop the connection of any client attempting to do SGC. SGC support should only be set to `true` if the virtual service is configured with an X.509 certificate with the special Server Gated Cryptography extension. Consult your CA for details on obtaining such a certificate.	`sgcSupport {true | false}`
`SSL Resumption Support`	The virtual service's capability for doing SSL session resumption. When set to `true`, the system will cache SSL sessions and allow clients to resume those sessions on subsequent handshakes. The resumption handshakes are less computationally intensive, yielding better performance on resumed connections. When set to `false`, the system keeps no cache and requires a full SSL handshake on all connections to the virtual service. The default setting is `false`.	`resume {true | false}`

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService

show virtualService ssl statistics

Purpose

Display Secure Sockets Layer (SSL)-specific connection data.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance virtualService ssl statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show virtualService ssl
statistics
Name:                           vlan40
Current Open SSL Connections:   5
Peak Open SSL Connections:      19
Total Successful SSL Handshakes: 1250154
Total Failed SSL Handshakes:    1
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the virtual service.	`name` *`text`*
`Current Open SSL Connections`	The number of SSL connections currently open on this virtual service.	`sslCurrentConnections` *`integer`*
`Peak Open SSL Connections`	The greatest number of SSL connections open at one time on this virtual service.	`sslPeakConnections` *`integer`*
`Total Successful SSL Handshakes`	The total number of completed SSL handshakes on this virtual service. (Each connection can require more than a single handshake.)	`sslHandshakesSuccess` *`integer`*
`Total Failed SSL Handshakes`	The total number of attempted but failed SSL handshakes on this virtual service.	`sslHandshakesFailure` *`integer`*

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService → ssl → statistics

show virtualService statistics

Purpose

Displays the total number of transmitted and received bytes and packets on a named virtual service, as well as session information.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance virtualService statistics`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show virtualservice
statistics
Name:                           vlan30
Bytes Transmitted to clients:   30809518527
Bytes Received from clients:    1655959736
Packets Transmitted to clients: 15500241
Packets Received from clients:  4918255
Cumulative Open Sessions:       1087991
Cumulative Closed Sessions:     1003193
Current Open Sessions:          272
Peak Active Sessions:          13
Responses Returned:             1082001
Requests Processed:             1082455
Write Failures:                 0
Read Failures:                  0
Tunneling Decisions:            0
No Predicate Match Counts:      0

Name:                           vlan40
Bytes Transmitted to clients:   6530754212
Bytes Received from clients:    856828177
Packets Transmitted to clients: 12916467
Packets Received from clients:  11494354
Cumulative Open Sessions:       1219186
Cumulative Closed Sessions:     1123449
Current Open Sessions:          5
Peak Active Sessions:          13
Responses Returned:             1218400
Requests Processed:             1218403
Write Failures:                 0
Read Failures:                  1
Tunneling Decisions:            0
No Predicate Match Counts:      0
```

Output description

Field name	Description	Filter name
Name	The text string that identifies the virtual service.	name *text*
Bytes Transmitted to clients	The total number of bytes transmitted over this virtual service.	bytesSent *integer*
Bytes Received from clients	The total number of bytes received over this virtual service.	bytesReceived *integer*
Packets Transmitted to clients	The total number of packets transmitted over this virtual real service.	packetsSent *integer*
Packets Received from clients	The total number of packets received over this virtual service.	packetsReceived *integer*
Cumulative Open Sessions	The total number of sessions open on this virtual service.	openSessions *integer*
Cumulative Closed Sessions	The total number of sessions that have been opened and then closed on this virtual service.	closedSessions *integer*
Current Open Sessions	The number of sessions currently open on this virtual service. This number does not include sessions that are in TIME_WAIT state.	currentSessions *integer*
Peak Active Sessions	The greatest number of Layer 5 through Layer 7 sessions open at one time on this virtual service. This count includes sessions in the TIME_WAIT state, which are included in the Current Open Sessions count.	clientPeakSessions *integer*
Responses Returned	The total number of successful responses to Layer 5 through Layer 7 HTTP requests.	clientObjectsSent *integer*
Requests Processed	The total number of requests received from the client and successfully processed.	clientObjectsReceived *integer*
Write Failures	The total number of Layer 4 client requests that the system failed to write a response to due to a TCP failure.	clientWriteFailures *integer*
Read Failures	The total number of Layer 4 through Layer 7 read failures, commonly due to a TCP read failure or invalid HTTP request.	clientReadFailures *integer*

Field name	Description	Filter name
`Tunneling Decisions`	The total number of times that the virtual service has initiated a tunnel due to enabling `requestPolicy firstObjectSwitching`. When a tunnel is initiated, all requests in a single TCP session are sent to the same real service.	`tunnelingDecisions` *`integer`*
`No Predicate Match Counts`	The total number of instances in which the object received did not match any of the defined object rules.	`noPredicateMatchCounts` *`integer`*

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService → statistics

show vsGroup

Purpose

Displays the current virtual service group(s) on a vSwitch. The named virtual services included in the group can share their request policies, service groups, and real services with each other.

Access mode

user

Syntax

`show vSwitch-`*`name`* `loadBalance vsGroup`

Sample output

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# show vsGroup
Name:             group_1
Virtual Services: vs1; vs2; vs3
```

Output description

Field name	Description	Filter name
`Name`	The text string that identifies the virtual service group name.	`name` *`text`*
`Virtual Services`	The list of named virtual services included in the virtual service group.	`virtualServices` *`text`*

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService → vsGroup

sorryData

Purpose

Defines the action the load balancer should take when a sorry action is specified for a request policy or response policy.

Access mode

config

Syntax

```
vSwitch-name loadBalance sorryData
   name text
   action {close | redirect | reset | page}
   [actionString text]
```

Arguments

Argument name	Description
`name`	The text string assigned to the sorryData definition. The name can be up to 64 alphanumeric characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`action {close | redirect | reset | page}`	Indicates the action the load balancer should take when the sorry action is set in a request policy or response policy. Possible values are: `close`: Closes the HTTP connection. It sends an HTTP 500 Internal Error status code and closes the connection using a four-way handshake and FIN instead of a reset. `redirect`: Returns an HTTP 302 redirect response to the client, redirecting the request to a different URI. The target of the redirection is set with the `actionString` argument. `reset`: Sends a TCP reset to the client, ending the connection. `page`: Returns an HTML page to the client. The HTML page is specified with the `actionString` argument.
`actionString` *`text`*	**Optional**: Specifies the string to use to complete a `redirect` or `page` action. If `action` is `page`, enter an HTML fully qualified path name. If `action` is `redirect`, enter a valid URI.

Delete filters

```
no vSwitch-name loadBalance sorryData
   name text
   action {close | redirect | reset | page}
   [actionString text]
```

Example

```
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# sorryData so_1 redirect
“http://www.e-commerce.com/default/contact.htm/”
```

Associated MIB

```
sorryData.mib
```

Web path

- vSwitch → *name* → LoadBalance → sorryData

virtualService

Purpose

Defines configuration attributes for the server load balancer.

The command specifies the application service type for the service group and the vRouter on which client traffic will be received. In addition, the `virtualService` command assigns a virtual IP (VIP) address to the load balancer. This is the address to which DNS resolves URIs. Essentially, it is the address of the load balancer and masks the individual addresses of the servers behind it. Network address translation (NAT) converts, on the outbound transmission, the server's IP address in response headers to the VIP when responding to the client. If a request arrives and does not match the defined VIP and port combination, it is dropped. You cannot configure the same
VIP/router port combination for more than one virtual service.

You can configure up to 512 virtual services for each vSwitch. Each virtual service can support one 1024-byte certificate.

If you enter the `virtualService` command with a name and press [Return], the command enters you into the `virtualService` command mode.

The `no` form of the command deletes the named virtual service from the virtualService table.

Access mode

config

Syntax

Certain arguments are relevant for only specific service types. See the `virtualService` "Arguments" table for information about which arguments pertain to a specific service type. To create a virtual service:

```
vSwitch-name loadBalance virtualService
   name text
   appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB
      | FTPLB | RTSPLB}
   ipAddress ipAddress
   ipAddressRange {ipAddress-ipAddress}
   requestPolicyList text
   [requestTransformList text]
   [serviceGroupName text]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {integer | unlimited}]
   [protocol TCP]
   [port portNumber]
   [vRouter vSwitch:vRouter]
   [hostname text]
   [clientSrcIPRange ipAddress]
   [synRateLimit {unlimited | integer}]
   [persistType {none | srcAddress}]
   [srcAddressPersistMask ipAddress]
   [ftpDataPortRange portNumber]
   [ftpDataRxTimerShortTimeoutValue integer_seconds]
   [ftpDataRxTimerLongTimeoutValue integer_seconds]
   [ckmKeyName text]
   [sslProto {SSLv3 | TLSv1}]
   [ieExportCiphersSupport {enabled | disabled}]
   [sslCiphers ciphers]
   [reneg {true | false}]
   [sgcSupport {true | false}]
   [resume {true | false}]
```

To modify a virtual service:

```
vSwitch-name loadBalance virtualService name text
   [appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS |
      TDLB | FTPLB | RTSPLB}]
   [ipAddress ipAddress]
   [ipAddressRange {ipAddress-ipAddress}]
   [requestPolicyList text]
   [requestTransformList text]
   [serviceGroupName text]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {integer | unlimited}]
   [protocol TCP]
   [port portNumber]
   [vRouter vSwitch:vRouter]
   [hostname text]
   [clientSrcIPRange ipAddress]
   [synRateLimit {unlimited | integer}]
   [persistType {none | srcAddress}]
   [srcAddressPersistMask ipAddress]
   [ftpDataPortRange portNumber]
   [ftpDataRxTimerShortTimeoutValue integer_seconds]
   [ftpDataRxTimerLongTimeoutValue integer_seconds]
   [ckmKeyName text]
   [sslProto {SSLv3 | TLSv1}]
   [ieExportCiphersSupport {enabled | disabled}]
   [sslCiphers ciphers]
   [reneg {true | false}]
   [sgcSupport {true | false}]
   [resume {true | false}]
```

Arguments

Argument name	Description
The following argument is required when creating or modifying a virtual service.	
`name` *`text`*	Creates a virtual service with the specified name, and associates the request policies identified by `requestPolicyList`. If the name specified already exists, the system modifies the configuration for that service group, as specified by subsequent arguments you enter. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)

Argument name	Description
The following arguments are required for all service types when creating a virtual service but optional when modifying a virtual service.	
`appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB | FTPLB | RTSPLB}`	Specifies the application service type for the virtual service. This is the criteria on which the system bases load-balancing decisions. The following service types can be set: `L4SLB`: Layer 4 server load balancing (stateless). This is the only non-TCP-terminated service type. The only load-balancing algorithm allowed with this service type is `weightedHash`, set with the `serviceGroup` command. `L4SLB_ADV`: TCP-terminated Layer 4 server load balancing. This application service type provides advanced functionality implemented in the FX-SLB and FX-SSL function cards. This service type can take advantage of more sophisticated service selection algorithms leading to improved overall site availability and performance. (*Requires FX-SLB or FX-SSL function card*.) `L4SLB_SSL`: Provides the same hardware advantages as `L4SLB_ADV`, but with the added capability of SSL termination and re-origination. If selecting this type, you must also provide the `ckmKeyName`. (*Requires the FX-SSL function card*.) `HTTP`: Provides hardware TCP-termination to capture the entire HTTP application byte stream for Layer 5 through Layer 7 switching decisions based on objects within the session. (*Requires FX-SLB or FX-SSL function card*.) `HTTPS`: Adds SSL functionality to the HTTP capability. If selecting this type, you must also provide the `ckmKeyName`. (*Requires FX-SSL function card*.) `TDLB`: Provides Transparent Device load balancing. `FTPLB`: Provides FTP load balancing. `RTSPLB`: Provides RTSP load balancing.
`ipAddress` *`ipAddress`* `or` `ipAddressRange` *`ipAddress-ipAddress`*	Assigns the virtual IP (VIP) address to the load balancer. (This argument is used for all service types except TDLB.) Assigns the virtual IP (VIP) address range for the load balancer. (This argument is relevant to service type TDLB.)

Argument name	Description
The following argument is required if the `appServiceType` **is set to** `HTTP`**,** `HTTPS`**, or** `RTSPLB`**.**	
`requestPolicyList` *`text`*	Specifies the request policies to be applied to this virtual service. You cannot assign the same request policy to more than one virtual service. Request policies (configured with the `requestPolicy` command) associate an object rule with a service group, to inform the virtual service of the match criteria and action, and forwarding instructions. Enter the request policies in a space-separated list enclosed in braces { }.
The following optional arguments are relevant if the `appServiceType` **is set to** `HTTP`**,** `HTTPS`**, or** `RTSPLB`**.**	
`requestTransformList` *`text`*	**Optional.** Assigns a list of request transforms to control requests from clients.
`hostName` *`text`*	**Optional**. Specifies the DNS (fully qualified) host name identifying the virtual service.

Argument name	Description
The following optional argument is relevant if the appServiceType is set to L4SLB**,** L4SLB_ADV**,** L4SLB_SSL**,** FTPLB**, or** TDLB**. See the following listings for additional optional arguments for these service types.**	
serviceGroupName *text*	**Optional.** Specifies the name of the service group associated with this virtual service. This argument is required when creating a virtual service but optional when modifying a virtual service.

Argument name	Description
The following optional arguments are relevant for all service types.	
`description` *`text`*	**Optional.** Assigns a text description to the virtual service. This description is displayed with output from the `show virtualService` command.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named virtual service, either `enabled` or `disabled`. Set a status of `disabled` if you want to preconfigure the service before bringing it online. The default administrative status is `enabled`.
`disableDelay {`*`integer`* `| unlimited}`	**Optional**. The time, in seconds, that must elapse before the system disables a virtual service when `adminState` is set to `disabled`. Possible values are 0 to 600 seconds or `unlimited`. The default setting is 0 seconds. If this value is set to non-zero or `unlimited` and the `adminState` is set to `disabled`, the virtual service does not accept any new connections but current connections are not terminated. At the end of the `disableDelay` time, the system terminates all current connections to the virtual service.
`protocol {TCP | UDP}`	**Optional.** Specifies the protocol that the virtual service takes action on. When traffic of that protocol type arrives over the port specified with the `port` argument, it is forwarded according to the load-balancing configuration. The supported protocols are TCP and UDP.
`port` *`portNumber`*	**Optional.** Specifies the port that the virtual service uses to listen for connections of the type specified with the `protocol` argument. Port 0 is not a valid configuration. If the `appServiceType` is set to `L4SLB`, `L4SLB_ADV`, or `TDLB`, the default port is port 80. If the `appServiceType` is set to `FTPLB`, the default port is port 21. If the `appServiceType` is set to `L4SLB_SSL`, the default port is port 443.
`vRouter` *`vSwitch:vRouter`*	**Optional.** Defines the vRouter that traffic comes in on to reach this virtual service. Enter the vRouter name in the format *`vSwitch:vRouter`*. The default vRouter is `system:shared`.
`clientSrcIPRange` *`ipAddress-ipAddress`*	**Optional.** Limits access to the virtual service to a specified group of clients, for example to limit service to a private server. Clients are identified by source IP address. Their address must fall within the range you specify. Enter the ends of the range in dotted-decimal format, separated by a hyphen: *`lowEndIPAddress-highEndIPAddress`* The default range is 0.0.0.0-255.255.255.255.

Argument name	Description
`synRateLimit {unlimited | integer}`	**Optional.** Limits the number of TCP SYN packets that can be received in a period of 1 second. SYN packets signal the start of a TCP connection, and by limiting them you can institute a form of congestion control. When the limit is reached, subsequent packets are dropped. Valid values for the rate limit are either `unlimited`, which takes no action regardless of the number of packets received or a static integer between 10 and 300000. The default setting is `unlimited`.

Argument name	Description
The following optional arguments are relevant if the `appServiceType` **is set to** `L4SLB_ADV`**,** `L4SLB_SSL`**, or** `FTPLB`**. See the following listings for additional optional arguments for service types** `L4SLB_SSL`**,** `FTPLB`**, and** `TDLB`	
`persistType {none | srcAddress}`	**Optional.** Sets the algorithm to be used for maintaining session persistence. The default setting is `none`.
`srcAddressPersistMask` *`ipAddress`*	**Optional.** Specifies the mask for source IP address-based persistence. The default setting is `255.255.255.255`.

Argument name	Description
The following optional argument is relevant if the `appServiceType` **is set to** `TDLB`**.**	
`ipAddressRange` *`ipAddress-ipAddress`*	**Optional.** Specifies the range of IP addresses to be used for Transparent Device load balancing. Enter the ends of the range in dotted-decimal format, separated by a hyphen: *`lowEndIPAddress-highEndIPAddress`*

Argument name	Description
The following optional arguments are relevant if the `appServiceType` **is set to** `FTPLB`**.**	
`ftpDataPortRange`	**Optional.** Specifies the range of ports to be used for the FTP data service. The valid range is 10001 through 12048.
`ftpDataRxTimerShortTimeoutValue`	**Optional.** Defines the time to wait to receive the first data packet from a real service after a session has been established. If the timer expires, the switch terminates the session. The default setting is `32_seconds`.
`ftpDataRxTimerLongTimeoutValue`	**Optional.** Defines the time to wait to receive subsequent data packets from a client after the initial packet is received. If the timer expires, the switch terminates the session. The default setting is `64_seconds`.

Argument name	Description
The following optional arguments are relevant if `appServiceType` **is set to** `L4SLB_SSL` **or** `HTTPS`**.**	
`ckmKeyName` *`text`*	**Optional.** Specifies the SSL key and certificate the system uses on the frontend connection. If the `appServiceType` is an SSL type—`L4SLB_SSL` or `HTTPS` —you must supply this argument. The name you specify is the name you entered when you created the key in the CKM utility. If the key name you enter is not stored in the CKM table, the command will fail. In addition, the key must be in RSA format, and must be a certified key.
`sslProto {SSLv3 | TLSv1}`	**Optional.** Specifies the SSL protocols supported by the virtual service (SSL at the frontend). Possible values are: `SSLv3`: Secure Sockets Layer Protocol, Version 3.0 `TLSv1`: Transport Layer Security (TLS) Version 1.0, as defined in RFC 2246. (TLS is also known as SSL, Version 3.1.) The default is to support both `SSLv3` and `TLSv1`.
`ieExportCiphersSupport {enabled | disabled}`	**Optional**. Enables or disables weak ciphers for import/export compatibility. The default setting is `disabled`.

Argument name	Description

`sslCiphers` *`cipher`*	**Optional.** Specifies the encryption methods (cipher suites) used on traffic passing through this virtual service. (See RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*, for detailed descriptions of each cipher.) Enter the cipher suites in a space-separated list enclosed in braces { }. Possible supported cipher suites include:
	
	`RSA_EXPORT_WITH_RC4_40_MD5`
	
	`RSA_WITH_RC4_128_MD5`
	
	`RSA_WITH_RC4_128_SHA`
	
	`RSA_EXPORT_WITH_DES40_CBC_SHA`
	
	`RSA_WITH_DES_CBC_SHA`
	
	`RSA_WITH_3DES_EDE_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_DES_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_RC4_56_SHA`
	
	The default list of cipher suites, and those highly recommended for use, is `RSA_WITH_RC4_128_MD5, RSA_WITH_RC4_128_SHA,` and `RSA_WITH_3DES_EDE_CBC_SHA`.

`reneg {true | false}`	**Optional.** Sets the virtual service's support for SSL renegotiation, either `true` or `false`. When `true`, the system allows SSL renegotiation (multiple handshakes to occur over a single SSL connection). When `false`, the system ignores SSL renegotiation attempts from the SSL client. SSL renegotiation may occur when dealing with SSL clients that would like to rekey periodically or when dealing with Web browsers attempting to use Netscape's Step-Up (a technique for upgrading from weak cryptography to strong cryptography over an existing SSL connection). The default is `true`.

Argument name	Description
`sgcSupport {true | false}`	**Optional.** Sets the virtual service's support for Microsoft's Server Gated Cryptography (SGC). SGC, like Netscape's Step-Up, is a technique for upgrading from weak cryptography to strong cryptography over an existing SSL connection. This proprietary variation on the SSL handshake violates the SSL draft and the TLS RFC, and it is only done by Microsoft Web browsers when speaking to SSL servers with special SGC certificates. When `sgcSupport` is `true`, the system allows clients to do SGC handshakes. When `false`, the system does not allow SGC, and will drop the connection of any client attempting to do SGC. SGC support should only be set to `true` if the virtual service is configured with an X.509 certificate with the special Server Gated Cryptography extension. Consult your CA for details on obtaining such a certificate. The default is `false`.
`resume {true | false}`	**Optional.** Sets the virtual service's capability for doing SSL session resumption. When set to `true`, the system caches SSL sessions, allowing clients to resume those sessions on subsequent handshakes. When set to `false`, the system keeps no cache and requires a full SSL handshake on all connections to the virtual service. Resumption handshakes are less computationally intensive, yielding better performance on resumed connections. However, the act of maintaining the SSL session cache slows down each full handshake. The default is `false`.

Delete filters

```
no vSwitch-name loadBalance virtualService name text
   [appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS |
      TDLB | FTPLB | RTSPLB}]
   [ipAddress ipAddress]
   [ipAddressRange {ipAddress-ipAddress}]
   [requestPolicyList text]
   [requestTransformList text]
   [serviceGroupName text]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {integer | unlimited}]
   [protocol TCP]
   [port portNumber]
   [vRouter vSwitch:vRouter]
   [hostname text]
   [clientSrcIPRange ipAddress]
   [synRateLimit {unlimited | integer}]
   [persistType {none | srcAddress}]
   [srcAddressPersistMask ipAddress]
   [ftpDataPortRange portNumber]
   [ftpDataRxTimerShortTimeoutValue integer_seconds]
   [ftpDataRxTimerLongTimeoutValue integer_seconds]
   [ckmKeyName text]
   [sslProto {SSLv3 | TLSv1}]
   [ieExportCiphersSupport {enabled | disabled}]
   [sslCiphers ciphers]
   [reneg {true | false}]
   [sgcSupport {true | false}]
   [resume {true | false}]
```

Example

The following example defines a virtual service (L4SLB_ADV type).

```
sun> enable
sun# configure
sun(config)# vswitch SVS1
sun(config-vswitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# virtualService vs2 l4SLB_ADV
ipAddress 2.2.2.2 serviceGroupName sg1
```

The following example defines a virtual service (HTTP type).

```
sun> enable
sun# configure
sun(config)# vswitch SVS1
sun(config-vswitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# virtualService vs1 HTTP
ipAddress 2.2.2.2 requestPolicyList {qp1 qp2} requestTransformList
{qt1 qt2}
```

The following example modifies a virtual service (SSL type).

```
sun> enable
sun# configure
sun(config)# vswitch SVS1
sun(config-vswitch-SVS1)# loadbalance
sun(config-vSwitch-SVS1 loadBalance)# virtualService vs1_ssl
ckmKeyName Key1
```

Associated MIB

virtSvc.mib

Web path

- vSwitch → *name* → LoadBalance → virtualService → add
- vSwitch → *name* → LoadBalance → virtualService → copy
- vSwitch → *name* → LoadBalance → virtualService → modify

virtualService advanced

Purpose

Configures Layer 4 properties of the specified virtual service's TCP/IP session. By defining the initial state between the client and the system, you can fine tune the TCP connection for performance improvement through resource management.

You must enter the `virtualService` command with an existing named virtual service to access the advanced command.

Caution: It is possible to stop all traffic flow to the system by misconfiguring these values. The system comes preconfigured with defaults that serve most TCP connections. You should not change these defaults unless you are confident that you understand TCP operations.

Access mode

config

Syntax

```
vSwitch-name loadBalance virtualService vsName advanced
   [tcbTemplateKey 0]
   [ipTOS {Normal | MinCost | MaxReliablity | MaxThroughput |
      MinDelay}]
   [xmtRetryLimit integer]
   [estRetryLimit integer]
   [shortRxTimer timeoutValue]
   [longRxTimer timeoutValue]
   [rcvWnd integer]
   [xmtRTT rttValue]
   [smmStreamLimit limitValue]
   [estShortTimeout {ExpRFC793 | ExpRetr}]
```

Arguments

Argument name	Description
`name` *`text`*	Specifies a name of a configured virtual service. It is the connection between the client and this virtual service that is affected by modifications in this command.
`tcbTemplateKey 0`	**Optional.** Specifies the TCB table template key in use by this connection. The default, and only acceptable value, is 0. This argument works in conjunction with troubleshooting commands that are only available through field service support.
`ipTOS {Normal | MinCost | MaxReliablity | MaxThroughput | MinDelay}`	**Optional.** Sets the bit in the type of service (TOS) field of IP headers for traffic traversing this connection. This informs intervening routers how they should handle the packets. Possible values are: `Normal`: Used for ICMP, BOOTP, DNS, TCP queries traffic `MinCost`: Used for Usenet traffic `MaxReliablity`: Used for SNMP and routing traffic `MaxThroughput`: Used for FTP traffic `MinDelay`: Used for interactive login applications, such as Telnet and rlogin. The default setting is `Normal`.
`xmtRetryLimit` *`integer`*	**Optional.** Sets the base interval for the retransmit retry limit within an established connection. That is, the time between retransmits of data sent and not acknowledged. The values are as follows: • `3_seconds` • `9_seconds` • `21_seconds` • `43_seconds` • `91_seconds` • `155_seconds` • `219_seconds` • `283_seconds` • `347_seconds` • `411_seconds` The default setting is `283_seconds`.

Argument name	Description
`estRetryLimit` *integer*	**Optional.** Sets the base interval for the establishment retry limit, which is how long you are willing to attempt a connection. The values are as follows: • `6_seconds` • `30_seconds` • `78_seconds` • `142_seconds` • `206_seconds` • `270_seconds` • `334_seconds` • `398_seconds` • `462_seconds` • `526_seconds` The default setting is `78_seconds`.
`shortRxTimer` *timeoutValue*	**Optional.** Sets the receive timer short time-out value, which defines how long you are willing to wait to receive the first data packet from a client after a session has been established. If the timer expires, the switch terminates the session. Values are defined to allow for argument completion at the command line. Possible values are: • `disable` • `250_msec` • `500_msec` • `1_second` • `2_seconds` • `4_seconds` • `8_seconds` • `16_seconds` • `32_seconds` • `64_seconds` • `128_seconds` • `256_seconds` • `512_seconds` • `1024_seconds` • `2048_seconds` • `4096_seconds` The default setting is `32_seconds`.

Argument name	Description
`longRxTimer` *`timeoutValue`*	**Optional.** Sets the receive timer long time-out value, which defines how long you are willing to wait to receive subsequent data from a client after the initial packet. If the timer expires, the switch terminates the session. Values are defined to allow for argument completion at the command line. Possible values are displayed in the `shortRxTimer` argument description, above. The default setting is `128_seconds`.
`rcvWnd` *`integer`*	**Optional.** Sets the maximum amount of memory allocated to received data from a client on a session. Valid range is 1024 through 8388608; the default value is 16384 bytes.
`xmtRTT` *`roundTripValue`*	**Optional.** Sets the maximum value for the packet round trip transmit time in seconds or milliseconds. Possible values are: • `375_msec` • `500_msec` • `625_msec` • `750_msec` • `875_msec` • `1_second` • `1125_msec` • `1250_msec` • `1375_msec` • `1500_msec` The default setting is `750_msec`.
`smmStreamLimit` *`streamLimitValue`*	**Optional.** Sets the Stream Memory Manager multiplier from the Receive Window Size field. Possible values are: • `1xRecvWnd` • `2xRecvWnd` • `4xRecvWnd` • `8xRecvWnd` The default setting is `1xRecvWnd`.
`estShortTimeout {ExRFC793 | ExpRetr}`	**Optional.** Sets the type of Establishment Time time-out; either the value from ExpRFC793 or ExpRetr. The default setting is `ExpRetr`.

Argument name	Description
`rcvWndDisabled {true | false}`	**Optional.** Specifies whether TCP receive flow control is performed on a per-connection basis and advertised to the sender. The default setting is `false`.
`rcvMss` *`integer`*	**Optional.** Specifies the receive maximum segment size; broadcasted with a TCP SYN segment. The default setting is 1460.
`xmtMss` *`integer`*	**Optional.** Specifies the maximum segment size; advertised during the SYN handshake. The default setting is 1460.
`enableHttpMode {true | false}`	**Optional.** Indicates whether HTTP mode is enabled, either `true` or `false`. This is a performance-tuning parameter that optimizes HTTP. The default setting is `false`.
`initParseWithData {true | false}`	**Optional.** For client side connections, this setting affects HTTP and HTTPS performance by delaying processing until data is received. The default setting is `false.`
`rxUseLongTime {true | false}`	**Optional.** Specifies whether the internal receive progress timer is used to recover dormant connections. The default setting is `false.`
`Disable Syn Cookies {true | false}`	**Optional.** Disables SYN cookie processing for this virtual service.
`Client First Protocol {true | false}`	**Optional.** Specifies all protocols using this virtual service will be client-initiated.

Example

The following example sets type of service (TOS) bits to change how intervening routers handle these packets.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# virtualService VS1
advanced ipTos mincost
```

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService → advanced

virtualService ssl

Purpose

Modifies the Secure Sockets Layer (SSL) properties for a configured virtual service. This command is equivalent to the `virtualService` command.

In addition to modifying the SSL properties, you can use this command to modify all of the virtual service properties set with the `virtualService` command. See the description of the `virtualService` command for more information.

The no form of this command deletes the specified virtual service.

Access mode

config

Syntax

```
vSwitch-name loadBalance virtualService VSname ssl
   ipAddress ipAddress
   appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB
      | FTPLB | RTSPLB}
   requestPolicyList text
   [ipAddressRange {ipAddress-ipAddress}]
   [FTPDataPortRange portNumber]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {integer | unlimited}]
   [protocol TCP]
   [port portNumber]
   [vRouter vSwitch:vRouter]
   [hostname text]
   [clientSrcIPRange ipAddress]
   [synRateLimit {unlimited | integer}]
   [ckmKeyName text]
   [sslProto {SSLv3 | TLSv1}]
   [ieCompatSSLCiphers {enabled | disabled}]
   [sslCiphers ciphers]
   [reneg {true | false}]
   [sgcSupport {true | false}]
   [resume {true | false}]
```

Arguments

Argument name	Description
`ipAddress` *`ipAddress`*	Assigns the virtual IP (VIP) address to the load balancer. If you have specified only a single IP address for the outbound range of the static NAT configuration (with the `staticNAT` command), be certain that the IP address you assign here is not the same, or static NAT will fail. This argument is required when creating a virtual service, but optional when modifying or deleting one.
`appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB | FTPLB | RTSPLB}`	Specifies the application service type for the virtual service. This is the criteria on which the system bases load-balancing decisions. The following service types can be set: `L4SLB`: Layer 4 server load balancing (stateless). This is the only non-TCP-terminated service type. The only load-balancing algorithm allowed with this service type is `weightedHash`, set with the `serviceGroup` command. `L4SLB_ADV`: TCP-terminated Layer 4 server load balancing. This application service type provides advanced functionality implemented in the FX-SLB and FX-SSL function cards. This service type can take advantage of more sophisticated service selection algorithms leading to improved overall site availability and performance. (*Requires FX-SLB or FX-SSL function card*.) `L4SLB_SSL`: Provides the same hardware advantages as `L4SLB_ADV`, but with the added capability of SSL termination and re-origination. If selecting this type, you must also provide the `ckmKeyName`. (*Requires the FX-SSL function card*.) `HTTP`: Provides hardware TCP-termination to capture the entire HTTP application byte stream for Layer 5 through Layer 7 switching decisions based on objects within the session. (*Requires FX-SLB or FX-SSL function card*.) `HTTPS`: Adds SSL functionality to the HTTP capability. If selecting this type, you must also provide the `ckmKeyName`. (*Requires FX-SSL function card*.) `TDLB`: Provides Transparent Device load balancing. `FTPLB`: Provides FTP load balancing. `RTSPLB`: Provides RTSP load balancing This argument is required when creating a virtual service, but optional when modifying or deleting one.

Argument name	Description
`requestPolicyList` *`text`*	Specifies the request policies to be applied to this virtual service. You cannot assign the same request policy to more than one virtual service. Request policies (configured with the `requestPolicy` command) associate an object rule with a service group, to inform the virtual service of the match criteria and action, and forwarding instructions. Enter the request policies in a space-separated list enclosed in braces { }. If the `appServiceType` is an L4 type — `L4SLB`, `L4SLB_ADV`, or `L4SLB_SSL` — you can only specify one request policy in this list. If it is *not* an L4 type, the request policy you select must contain at least one object rule. This argument is required when creating a virtual service, but optional when modifying or deleting one.
`ipAddressRange` *`ipAddress-ipAddress`*	**Optional.** Specifies the range of IP addresses to be used for Transparent Device load balancing. (Relevant for TDLB service type.) Enter the ends of the range in dotted-decimal format, separated by a hyphen: *`lowEndIPAddress-highEndIPAddress`*
`FTPDataPortRange`	**Optional.** Specifies the range of ports to be used for the FTP data service. (Relevant for FTP service type.)
`description` *`text`*	**Optional.** Assigns a text description to the virtual service. This description is displayed with output from the `show virtualService` command.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the named virtual service, either `enabled` or `disabled`. Set a status of `disabled` if you want to preconfigure the service before bringing it online. The default administrative status is `enabled`.
`disableDelay {`*`integer`* `| unlimited)]`	**Optional**. The time, in seconds, that must elapse before the system disables a virtual service when the `adminState` is set to `disabled`. Possible values are 0 to 600 seconds or `unlimited`. The default setting is 0 seconds. If this value is set to non-zero or `unlimited` and the `adminState` is set to `disabled`, the virtual service does not accept any new connections but current connections are not terminated. At the end of the `disableDelay` time, the system terminates all current connections to the virtual service.

Argument name	Description
`protocol TCP`	**Optional.** Specifies the protocol that the virtual service takes action on. When traffic of that protocol type arrives over the port specified with the `port` argument, it is forwarded according to the load-balancing configuration. The only supported protocol is TCP.
`port` *`portNumber`*	**Optional.** Specifies the port that the virtual service uses to listen for connections of the type specified with the `protocol` argument. Port 0 is not a valid configuration. The default port is port 80.
`vRouter` *`vSwitch:vRouter`*	**Optional.** Defines the vRouter that traffic comes in on to reach this virtual service. Enter the vRouter name in the format *`vSwitch:vRouter`*. The default vRouter is `system:shared`.
`hostname` *`text`*	**Optional**. The DNS host name of the virtual service.
`clientSrcIPRange` *`ipAddress`*	**Optional.** Limits access to the virtual service to a specified group of clients, for example to limit service to a private server. Clients are identified by source IP address. Their address must fall within the range you specify. Enter the ends of the range in dotted-decimal format, separated by a hyphen: *`lowEndIPAddress-highEndIPAddress`* The default range is 0.0.0.0-255.255.255.255.
`synRateLimit {unlimited |` *`integer`*`}`	**Optional.** Limits the number of TCP SYN packets that can be received in a period of 1 second. SYN packets signal the start of a TCP connection, and by limiting them you can institute a form of congestion control. When the limit is reached, subsequent packets are dropped. Valid values for the rate limit are either `unlimited`, which takes no action regardless of the number of packets received or a static integer between 10 and 300000. The default setting is `unlimited`.
`ckmKeyName` *`text`*	**Optional.** Specifies the SSL key and certificate the system uses on the frontend connection. If the `appServiceType` is an SSL type — `L4SLB_SSL` or `HTTPS` — you must supply this argument. The name you specify is the name you entered when you created the key in the CKM utility. If the key name you enter is not stored in the CKM table, the command will fail. In addition, the key must be in RSA format, and must be a certified key.

Argument name	Description
`sslProto {SSLv3 | TLSv1}`	**Optional.** Specifies the SSL protocols supported by the virtual service (SSL at the frontend). Possible values are: `SSLv3`: Secure Sockets Layer Protocol, Version 3.0 `TLSv1`: Transport Layer Security (TLS) Version 1.0, as defined in RFC 2246. (TLS is also known as SSL, Version 3.1.) The default is to support both `SSLv3` and `TLSv1`.
`ieExportCiphersSupport {enabled | disabled}]`	**Optional**. Enables weak ciphers for import/export compatibility. The default setting is that weak ciphers are disabled.

`sslCiphers` *`cipher`*	**Optional.** Specifies the encryption methods (cipher suites) used on traffic passing through this virtual service. (See RFC 2246, *The Transport Layer Security (TLS) Protocol Version 1.0*, for detailed descriptions of each cipher.) Enter the cipher suites in a space-separated list enclosed in braces { }. Possible supported cipher suites include:
	
	`RSA_EXPORT_WITH_RC4_40_MD5`
	
	`RSA_WITH_RC4_128_MD5`
	
	`RSA_WITH_RC4_128_SHA`
	
	`RSA_EXPORT_WITH_DES40_CBC_SHA`
	
	`RSA_WITH_DES_CBC_SHA`
	
	`RSA_WITH_3DES_EDE_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_DES_CBC_SHA`
	
	`RSA_EXPORT1024_WITH_RC4_56_SHA`
	
	The default list of cipher suites, and those highly recommended for use, is `RSA_WITH_RC4_128_MD5`, `RSA_WITH_RC4_128_SHA`, and `RSA_WITH_3DES_EDE_CBC_SHA`.

Argument name	Description
`reneg {true | false}`	**Optional.** Sets the virtual service's support for SSL renegotiation, either `true` or `false`. When `true`, the system allows SSL renegotiation (multiple handshakes to occur over a single SSL connection). When `false`, the system ignores SSL renegotiation attempts from the SSL client. SSL renegotiation may occur when dealing with SSL clients that would like to rekey periodically or when dealing with Web browsers attempting to use Netscape's Step-Up (a technique for upgrading from weak cryptography to strong cryptography over an existing SSL connection). The default is `true`.
`sgcSupport {true | false}`	**Optional.** Sets the virtual service's support for Microsoft's Server Gated Cryptography (SGC). SGC, like Netscape's Step-Up, is a technique for upgrading from weak cryptography to strong cryptography over an existing SSL connection. This proprietary variation on the SSL handshake violates the SSL draft and the TLS RFC, and it is only done by Microsoft Web browsers when speaking to SSL servers with special SGC certificates. When `sgcSupport` is `true`, the system allows clients to do SGC handshakes. When `false`, the system does not allow SGC, and will drop the connection of any client attempting to do SGC. SGC support should only be set to `true` if the virtual service is configured with an X.509 certificate with the special Server Gated Cryptography extension. Consult your CA for details on obtaining such a certificate. The default is `false`.
`resume {true | false}`	**Optional.** Sets the virtual service's capability for doing SSL session resumption. When set to `true`, the system will cache SSL sessions and allow clients to resume those sessions on subsequent handshakes. The resumption handshakes are less computationally intensive, yielding better performance on resumed connections. When set to `false`, the system keeps no cache and requires a full SSL handshake on all connections to the virtual service. The default setting is `false`.

Delete filters

See the show virtualService ssl command for argument descriptions.

```
no vSwitch-name loadBalance virtualService VSname ssl
   ipAddress ipAddress
   appServiceType {L4SLB | L4SLB_ADV | L4SLB_SSL | HTTP | HTTPS | TDLB
      | FTPLB | RTSPLB}
   requestPolicyList text
   [ipAddressRange ipAddress]
   [FTPDataPortRange portNumber]
   [description text]
   [adminState {enabled | disabled}]
   [disableDelay {integer | unlimited}]
   [protocol TCP]
   [port portNumber]
   [vRouter vSwitch:vRouter]
   [hostname text]
   [clientSrcIPRange ipAddress]
   [synRateLimit {unlimited | integer}]
   [ckmKeyName text]
   [sslProto {SSLv3 | TLSv1}]
   [ieExportCipherSupport {enabled | disabled}]
   [sslCiphers ciphers]
   [reneg {true | false}]
   [sgcSupport {true | false}]
   [resume {true | false}]
   [functionCard {1 | 2}]
   [operStatus {inactive | active | inactivePending | activePending}
   [operMessage text]
```

Example

The following example modifies the SSL properties for a virtual service.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# virtualService name VS1
ssl sslProto SSLv3 ieExportCipherSupport enabled
```

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → virtualService → ssl → modify
- vSwitch → *name* → LoadBalance → virtualService → ssl → delete

vsGroup

Purpose

Configures a virtual service group that contains multiple (up to 64) shared virtual service configurations. The named virtual services included in the group can share request policies, service groups, and real services with each other.

You can create up to 64 virtual service groups per vSwitch. However, a named virtual service can only be included in one virtual service group.

For detailed information on creating virtual service groups, refer to the *Sun N2000 Series Release 2.0 – System Configuration Guide*.

Access mode

enable

Syntax

For creating a virtual service group:

```
vSwitch-name loadBalance vsGroup
   name text
   virtualServices {name name name...}
```

For modifying a virtual service group:

```
vSwitch-name loadBalance vsGroup name
   [virtualServices {name name name...}]
```

Arguments

Argument name	Description
`name` *`text`*	Specifies the new or existing virtual service group name. If the name specified already exists, the system modifies the configuration for that virtual service group. The name can be up to 64 characters, including numerals and the following special characters: • underscore (_) • period (.) • at sign (@) • forward slash (/) • colon (:) • dash (-)
`virtualServices` *`text`*	Specifies the list of named virtual services to be included in a virtual service group. A group can include up to 64 named virtual service configurations, with up to 64 groups per vSwitch. However, a named virtual service can only be included in one group. Specify the list of named virtual services by enclosing them in braces ({}), separating the virtual service names with a blank space. Use the semi-colon (;) to separate virtual service names from specified name ranges (using the hyphen character (-)). **Example:** `vsgroup group_1 {vs1 vs2 vs3}` **Example:** `vsgroup group_2 {vs4; vs8-vs50}`

Delete filters

```
no vSwitch-name loadBalance vsGroup name
[virtualServices {name name name...}]
```

Example

The following example creates a vsGroup that includes three virtual service configurations.

```
sun> enable
sun# configure
sun(config)# vswitch e-commerce
sun(config-vswitch-e-commerce)# loadbalance
sun(config-vSwitch-e-commerce loadBalance)# vsGroup group_1 {vs1 vs2
vs3}
```

Associated MIB

`virtSvc.mib`

Web path

- vSwitch → *name* → LoadBalance → vsGroup

Chapter 30. TideRunner function card commands

TideRunner command description

The tideRunner commands allow you to view TideRunner operational statistics for the function cards. These are hardware-collected counts of terminated services in the switch. The statistics indicate both connection/session and segment activity. A *session* is an established connection between a TCP client and server. A *segment* is a TCP data unit.

TideRunner command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`switchServices tideRunner` *`commandname`*

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

TideRunner command summary

Table 30-1 lists and briefly describes the tideRunner commands.

Table 30-1. tideRunner Command Summary

Command Name	Description
`initKeys`	Modify TideRunner advanced tuning settings.
`show congestion summary`	Display congestion statistics for a function card.
`show initKeys`	Display TideRunner advanced tuning settings.
`show statistics summary`	Display TCP statistics for a function card.
`show statistics group`	Display TCP statistics for a statistics group associated with a vSwitch.
`show statistics group sslRecord`	Display SSL records for a statistics group associated with a vSwitch.

initKeys

Configures advanced N2000 Series TideRunner tuning parameters. These settings should only be modified with the assistance of Sun.

Access mode

config

Syntax

```
vSwitch-name tideRunner initKeys
   Card {functionCard1 | functionCard2}
   [R_PciInterfaceCount integer]
   [TcbTemplateMax integer]
   [StatPollPeriod integer]
   [SmmPageSize integer]
```

Arguments

Argument name	Description
`Card {functionCard1 | functionCard2}`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.
`R_PciInterfaceCount` *integer*	For Sun internal use only. Advanced tuning parameter. Default is 5000.
`TcbTemplateMax` *integer*	For Sun internal use only. Advanced tuning parameter. Default is 4096.
`StatPollPeriod` *integer*	For Sun internal use only. Advanced tuning parameter. Configures the poll period for sampling Tiderunner hardware-maintained statistics. Default is 2.

Argument name	Description
`SmmPageSize` *`integer`*	For Sun internal use only. Advanced tuning parameter. Configures the SMM page size (memory). Default is 4.

Associated MIB

`tteCounters.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → initKeys

show congestion summary

Purpose

Displays congestion statistics on a specific function card (TideRunner). Use this information to determine if the vSwitch needs to be reconfigured with additional resources, or if the vSwitch is under TCP attack.

All statistics are collected at each TideRunner statistics poll. At each poll interval, the current and previous values are compared to detect transitions between usage levels. Events are issued as the usage levels increase and decrease and the severity of the events change depending on absolute current usage level.

In the "Sample output," the first set of displayed fields (`clTcb` through `svrStrm`) shows the resource usage level statistics as a range. The second set of displayed fields (`clMaxTCBPct` through `svrMaxStrmPct`) shows the resource usage level statistics as integers that allow the NMON alarm group to monitor the TideRunner congestion table.

Access mode

user

Syntax

`show vSwitch-`*`name`* `tideRunner congestion summary`

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# show tideRunner congestion summary
Card:          functionCard1
clTcb:         0-50%
clPage:        0-50%
clStrm:        0-50%
svrTcb:        0-50%
svrPage:       0-50%
svrStrm:       0-50%
clMaxTcbPct:   50
clMaxPagePct:  50
clMaxStrmPct:  50
svrMaxTcbPct:  50
svrMaxPagePct: 50
svrMaxStrmPct: 50
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`

`clTcb`	Client Transmission Control Block (TCB). The TCB is a 256-byte structure that TideRunner uses to maintain TCP connection state; one TCB is used per TCP connection. This statistic reflects the TCB resource usage level for this class (vSwitch, client resources). The usage level is represented as a range, using a logarithmic scale, and can take one of the following values:
	
	`0-50%`
	`50-75%`
	`75-88%`
	`88-94%`
	`94-97%`
	`97-98%`
	`>= 99%`
		`clTcb` *`text`*

`clPage`	Client Page. A page is a unit of physical memory that is used to hold TCP stream data from clients. The `clPage` statistic reflects the current usage of pages set aside for client data by this vSwitch.	`clPage` *`text`*

Field Name	Description	Filter Name
`clStrm`	Client Stream. A stream resource is a data structure referring to 0 or more pages used to persist received terminated TCP application data from clients. The `clStrm` statistic reflects the current usage of TideRunner stream resources that are used for TCP connections to clients.	`clStrm` *`text`*

`svrTcb`	Server TCB. The TCB is a 256-byte structure that TideRunner uses to maintain TCP connection state; one TCB is used per TCP connection. This statistic reflects the TCB resource usage level for this class (vSwitch, server resources). The usage level is represented as a range, using a logarithmic scale, and can take one of the following values:
	
	`0-50%`
	`50-75%`
	`75-88%`
	`88-94%`
	`94-97%`
	`97-98%`
	`>= 99%`
		`svrTcb` *`text`*

`svrPage`	Server Page. A page is a unit of physical memory used to hold TCP stream data from servers. The default page size is 2 Kbytes. The `svrPage` statistic reflects the current usage of pages set aside for server data by this vSwitch.	`svrPage` *`text`*
`svrStrm`	Server Stream. A stream resource is a data structure referring to 0 or more pages that are used to persist received terminated TCP application data from servers. The `svrStrm` statistic reflects the current usage of TideRunner stream resources that are used for TCP connections to servers.	`svrStrm` *`text`*

Field Name	Description	Filter Name

`clMaxTcbPct`	Client Maximum TCB Percentage. This resource usage level is represented by the `clTcb` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`clMaxTcbPct` *`integer`*

`clMaxPagePct`	Client Maximum Page Percentage. This resource usage level is represented by the `clPage` end-of-range value, and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`clMaxPagePct` *`integer`*

Field Name	Description	Filter Name

`clMaxStrmPct`	Client Maximum Stream Percentage. This resource usage level is represented by the `clStrm` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`clMaxStrmPct` *integer*

`svrMaxTcbPct`	Server Maximum TCB Percentage. This resource usage level is represented by the `svrTcb` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`svrMaxTcbPct` *integer*

Field Name	Description	Filter Name

`svrMaxPagePct`	Server Maximum Page Percentage. This resource usage level is represented by the `svrPage` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`svrMaxPagePct` *`integer`*

`svrMaxStrmPct`	Server Maximum Stream Percentage. This resource usage level is represented by the `svrStrm` end-of-range value and can take on one of the following values:
	
	`0`
	`50`
	`75`
	`88`
	`94`
	`97`
	`98`
	`100`
	
	This integer field can be monitored by NMON.
		`svrMaxStrmPct` *`integer`*

Associated MIB

`tteCounters.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → congestion→ summary

show initKeys

Purpose

Displays advanced N2000 Series TideRunner tuning parameters. These settings should only be modified with the assistance of Sun.

Access mode

config

Syntax

`show vSwitch-`*`name`* `tideRunner initKeys`

Sample output

```
sun> enable
sun# switchServices
sun(switchServices)# show tideRunner initKeys
Card:                           functionCard1
R_Pci Interface Count:          5000
Tcb Template Max:               4096
Stat Poll Period:               2
SMM Page Size:                  4
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`

Field Name	Description	Filter Name
`R_PCI Interface Count`	For Sun use only. Advanced tuning parameter. Default is 5000.	`R_PciInterfaceCount <`*`integer`*
`TCB Template Max`	For Sun use only. Advanced tuning parameter. Default is 4096.	`TcbTemplateMax` *`integer`*
`Stat Poll Period`	For Sun use only. Advanced tuning parameter. Indicates the poll period for sampling Tiderunner hardware-maintained statistics. Default is 2.	`StatPollPeriod` *`integer`*
`SMM Page Size`	For Sun use only. Advanced tuning parameter. Indicates the SMM page size (memory). Default is 4	`SmmPageSize` *`integer`*

Associated MIB

`tteCounters.mib`

Web path

- vSwitch → *name* → LoadBalance → tideRunner → initKeys

show statistics summary

Purpose

Displays aggregated TCP statistics for all TCP sessions on a function card.

Access mode

user

Syntax

```
show switchServices tideRunner statistics summary
```

Sample output

```
sun> enable
sun# switchServices
sun(switchservices)# show tideRunner statistics summary
Card:              functionCard1
ConnsActive:       0
ActiveOpens:       0
ActiveDrops:       1
PassiveOpens:      1
PassiveDrops:      0
ConnFailures:      0
RcvTotalSegs:      6
RcvCtrl:           5
RcvOOSegs:         0
RcvBytes:          288
RcvDrop:           0
RcvAfterWin:       0
RcvWinProbe:       0
RcvAckTooMuch:     0
RcvWinUpd:         0
SndTotalSegs:      5
SndCtrl:           4
SndRexmtSegs:      0
SndRttSegs:        0
SndBytes:          128
SndRstSegs:        0
SndDelAck:         0
RexmtTimeout:      0
PersistTimeout:    0
InActivityTimeout: 0
RcvOrphanDrops:    0
CngSynDiscard:     0
CngAccTimeOut:     0
CngConnAborts:     0
```

```
SndSynCookie:     0
RcvSynCookie:     0
RcvSynCookieDrop: 0
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`
`ConnsActive`	The number of connections that are currently active.	`connsActive` *`integer`*
`ActiveOpens`	The number of connections opened where the function card initiated the open process.	`activeOpens` *`integer`*
`ActiveDrops`	The number of connections closed where the function card initiated the close.	`activeDrops` *`integer`*
`PassiveOpens`	The number of connections that a remote client or server initiated.	`passiveOpens` *`integer`*
`PassiveDrops`	The number of connections that a remote client or server closed.	`passiveDrops` *`integer`*
`ConnFailures`	The total number of connection attempts that failed.	`connFailures` *`integer`*
`RcvTotalSegs`	The total number of TCP segments that the function card received.	`rcvTotalSegs` *`integer`*
`RcvCtrl`	The total number of control segments that the function card received. A control segment contains only the TCP flags of SYN, FIN, RST, and ACK, or an ACK with no data.	`rcvCtrl` *`integer`*

Field Name	Description	Filter Name
RcvOOSegs	The total number of TCP segments that the function card received out of order.	rcvOOSegs *integer*
RcvBytes	The total number of bytes that the function card received that contain the data payload, excluding TCP headers.	rcvBytes *integer*
RcvDrop	The total number of TCP segments that the function card dropped. This value does not include the number of received error segments (for example, duplicate, illegal, or orphan segments that the function card cannot process).	rcvDrop *integer*
RcvAfterWin	The total number of TCP packets that the function card received after a TCP receive window closed (the function card cannot receive additional data until it reopens the window). The function card drops these packets.	rcvAfterWin *integer*
RcvWinProbe	The total number of window probe packets that the function card received.	rcvWinProbe *integer*
RcvAckTooMuch	The total number of acknowledgements (ACKs) that the function card accepted for data that it did not receive.	rcvAckTooMuch *integer*
RcvWinUpd	The total number of TCP window update packets that the function card received.	rcvWinUpd *integer*
SndTotalSegs	The total number of TCP segments that the function card transmitted to a remote client or server, excluding retransmitted segments.	sndTotalSegs *integer*
SndCtrl	The total number of control frames sent to a remote client or server. A control segment from the function card does not include data. It can include TCP flags such as SYN, FIN, RST, or ACK.	sndCtrl *integer*
SndRexmtSegs	The total number of TCP segments that the function card retransmitted. Segment retransmission occurs if the remote client or server does not send an acknowledgement (ACK) to the function card on the first attempt at transmission.	sndRexmtSegs *integer*

Field Name	Description	Filter Name
SndBytes	Total number of data payload bytes that the function card transmitted to a remote client or server, including retransmission. This value does not include TCP header bytes.	sndBytes *integer*
SndRstSegs	The total number of reset segments that the function card transmitted to a remote client or server.	sndRstSegs *integer*
SndDelAck	Number of delayed acknowledgements (ACKs) that the function card sent to a remote client or server. Using delayed ACKs allows the system to send a packet acknowledgement and response at the same time, increasing performance.	sndDelAck *integer*
RexmtTimeout	The total number of times that the function card reset a connection because it exceeded the number of times it could try to retransmit a TCP segment.	rexmtTimeout *integer*
PersistTimeout	The total number of times the function card sent window probes to a remote client or server to determine if a TCP window was open and could receive data.	persistTimeout *integer*
InActivityTimeout	The total number of times a connection timed out because of inactivity. The function card closes a connection if it does not receive an acknowledgement from a connection after sending multiple keep alive probes.	inActivityTimeout *integer*
RcvOrphanDrops	The total number of orphan segments that the function card dropped because it did not have the space to store these segments.	rcvOrphanDrops *integer*
CngSynDiscard	The number of passive open requests dropped due to congestion.	CngSynDiscard *integer*
CngAccTimeOut	The number of retransmit or inactivity timeout accelerations due to congestion.	CngAccTimeOut *integer*

Field Name	Description	Filter Name
`CngConnAborts`	The number of connections that were aborted due to congestion.	`CngConnAborts` *`integer`*
`sndSynCookie`	The total number of SYN.ACK cookies sent.	`sndSynCookie`
`rcvSynCookie`	The total number of received frames that required a cookie to be validated.	`rcvSynCookie`
`rcvSynCookieDrop`	The total number of received frames that required a cookie to be validated and was rejected.	`rcvSynCookieDrop`

Associated MIB

`tteCounters.mib`

Web path

- SwitchServices → tideRunner → statistics → summary

show statistics group

Purpose

Displays TCP statistics for a specific statistics group associated with a function card. The system associates all connections for a vSwitch with two statistics groups: one group for connections to virtual services and another group for real service connections per vSwitch. The system clears all counters when you reboot the system or the function card.

The system assigns all TCP connections for a vSwitch to one of 128 statistics groups. This command displays statistics for statistics group 0. Statistics group 0 contains an aggregate of the statistics from all other statistics groups.

Access mode

user

Syntax

```
show switchServices tideRunner statistics group
```

Sample output

```
sun> enable
sun# switchServices
sun(switchservices)# show tideRunner statistics group
Card:              functionCard1
Group Id:          1
vSwitch            e-commerce
Type               VirtualService
ConnsActive:       0
ActiveOpens:       0
ActiveDrops:       0
PassiveOpens:      0
PassiveDrops:      0
ConnFailures:      0
RcvTotalSegs:      0
RcvCtrl:           0
RcvOOSegs:         0
RcvBytes:          0
RcvDrop:           0
RcvAfterWin:       0
RcvWinProbe:       0
RcvAckTooMuch:     0
RcvWinUpd:         0
```

```
SndTotalSegs:      0
SndCtrl:           0
SndRexmtSegs:      0
SndRttSegs:        0
SndBytes:          0
SndRstSegs:        0
SndDelAck:         0
RexmtTimeout:      0
PersistTimeout:    0
InActivityTimeout: 0
cngSynDiscard      0
cngAccTimeOut      0
cngConnAborts      0
RcvOrphanDrops:    0

Card:              functionCard1
Group Id:          2
vSwitch            e-commerce
Type               RealService
ConnsActive:       0
ActiveOpens:       0
ActiveDrops:       0
PassiveOpens:      0
PassiveDrops:      0
ConnFailures:      0
RcvTotalSegs:      0
RcvCtrl:           0
RcvOOSegs:         0
RcvBytes:          0
RcvDrop:           0
RcvAfterWin:       0
RcvWinProbe:       0
RcvAckTooMuch:     0
RcvWinUpd:         0
SndTotalSegs:      0
SndCtrl:           0
SndRexmtSegs:      0
SndRttSegs:        0
SndBytes:          0
SndRstSegs:        0
SndDelAck:         0
RexmtTimeout:      0
PersistTimeout:    0
InActivityTimeout: 0
cngSynDiscard      0
cngAccTimeOut      0
cngConnAborts      0
RcvOrphanDrops:    0
```

Output description

Field Name	Description	Filter Name
`Card`	The function card for which statistics are being reported. Possible values are: • `functionCard1`: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. • `functionCard2`: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	`card {functionCard1 | functionCard2}`
`Group ID`	The counter index for a statistics group. The system maps all TCP connections for a vSwitch to a specific statistic group. Possible values are 1 through 128.	`groupId` *`integer`*
`vSwitch`	The name of the vSwitch associated with the statistics group.	`vSwitch` *`name`*
`Type`	The type of statistic counter, either `VirtualService` or `RealService`.	`type {realService | virtualService}`
`ConnsActive`	The number of connections that are currently active.	`connsActive` *`integer`*
`ActiveOpens`	The number of connections opened where the function card initiated the open process.	`activeOpens` *`integer`*
`ActiveDrops`	The number of connections closed where the function card initiated the close.	`activeDrops` *`integer`*
`PassiveOpens`	The number of connections that a remote client or server initiated,	`passiveOpens` *`integer`*
`PassiveDrops`	The number of connections that a remote server client or server closed.	`passiveDrops` *`integer`*
`ConnFailures`	The total number of connection attempts that failed.	`connFailures` *`integer`*
`RcvTotalSegs`	The total number of TCP segments that the function card received.	`rcvTotalSegs` *`integer`*

Field Name	Description	Filter Name
RcvCtrl	The total number of control segments that the function card received. A control segment contains only the TCP flags of SYN, FIN, RST, and ACK, or an ACK with no data.	rcvCtrl *integer*
RcvOOSegs	The total number of TCP segments that the function card received out of order.	rcvOOSegs *integer*
RcvBytes	The total number of bytes that the function card received that contain the data payload, excluding TCP headers.	rcvBytes *integer*
RcvDrop	The total number of TCP segments that the function card dropped. This value does not include the number of received error segments (for example, duplicate, illegal, or orphan segments that the function card cannot process).	rcvDrop *integer*
RcvAfterWin	The total number of TCP packets that the function card received after a TCP receive window closed (the function card cannot receive additional data until it reopens the window). The function card drops these packets.	rcvAfterWin *integer*
RcvWinProbe	The total number of window probe packets that the function card received.	rcvWinProbe *integer*
RcvAckTooMuch	The total number of acknowledgements (ACKs) that the function card accepted for data that it did not receive.	rcvAckTooMuch *integer*
RcvWinUpd	The total number of TCP window update packets that the function card received.	rcvWinUpd *integer*
SndTotalSegs	The total number of TCP segments that the function card transmitted to a remote client or server, excluding retransmitted segments.	sndTotalSegs *integer*
SndCtrl	The total number of control frames sent to a remote client or server. A control segment from the function card does not include data. It can include TCP flags such as SYN, FIN, RST, or ACK.	sndCtrl *integer*

Field Name	Description	Filter Name
SndRexmtSegs	The total number of TCP segments that the function card retransmitted. Segment retransmission occurs if the remote client or server does not send an acknowledgement (ACK) to the function card on the first attempt at transmission.	sndRexmtSegs *integer*
SndBytes	Total number of data payload bytes that the function card transmitted to a remote client or server, including retransmission. This value does not include TCP header bytes.	sndBytes *integer*
SndRstSegs	The total number of reset segments that the function card transmitted to a remote client or server.	sndRstSegs *integer*
SndRttSegs	The total number of round trip time segments that the function card segments to a remote client or server. The function card uses the round trip time measurements to avoid traffic congestion.	sndRttSegs *integer*
SndDelAck	Number of delayed acknowledgements (ACKs) that the function card sent to a remote client or server. Using delayed ACKs allows the system to send a packet acknowledgement and response at the same time, increasing performance.	sndDelAck *integer*
RexmtTimeout	The total number of times that the function card reset a connection because it exceeded the number of times it could try to retransmit a TCP segment.	rexmtTimeout *integer*
PersistTimeout	The total number of times the function card sent window probes to a remote client or server to determine if a TCP window was open and could receive data.	persistTimeout *integer*

Field Name	Description	Filter Name
`InActivityTimeout`	The total number of times a connection timed out because of inactivity. The function card closes a connection if it does not receive an acknowledgement from a connection after sending multiple keep alive probes.	`inActivityTimeout` *`integer`*
`CngSynDiscard`	The number of passive open requests dropped due to congestion.	`CngSynDiscard` *`integer`*
`CngAccTimeOut`	The number of retransmit or inactivity timeout accelerations due to congestion.	`CngAccTimeOut` *`integer`*
`CngConnAborts`	The number of connections that were aborted due to congestion.	`CngConnAborts` *`integer`*
`RcvOrphanDrops`	The total number of orphan segments that the function card dropped because it did not have the space to store these segments.	`rcvOrphanDrops` *`integer`*

Associated MIB

`tteCounters.mib`

Web path

- SwitchServices → tideRunner → statistics → group

show statistics group sslRecord

Purpose

Displays SSL statistics for a specific statistics group associated with a function card. The system associates all connections for a vSwitch with two statistics groups; one group for connections to virtual services and another group for real service connections per vSwitch.

Access mode

user

Syntax

```
show switchServices tideRunner statistics group sslRecord
```

Sample output

```
sun> enable
sun# switchServices
sun(switchservices)# show tideRunner statistics group sslRecord
Card:                           functionCard1
Group Index:                    1
VSwitch:                        SVS1
Type:                           RealService
Number Of Records Encrypted:    0
Number Of Bytes Encrypted:      0
Number Of Records Decrypted:    0
Number Of Bytes Decrypted:      0
Number Of Encrypt Errors:       0
Number Of Decrypt Errors:       0

Card:                           functionCard1
Group Index:                    2
VSwitch:                        SVS1
Type:                           VirtualService
Number Of Records Encrypted:    0
Number Of Bytes Encrypted:      0
Number Of Records Decrypted:    0
Number Of Bytes Decrypted:      0
Number Of Encrypt Errors:       0
Number Of Decrypt Errors:       0
```

Field Name	Description	Filter Name
Card	The function card for which statistics are being reported. Possible values are: functionCard1: The card installed in the front of the chassis. Use the top row of LEDs on the front of the chassis for status. functionCard2: The card installed in the back of the chassis. Use the bottom row of LEDs for status.	card (functionCard1| functionCard2)
GroupIndex	The counter index for a statistics group. The system maps all TCP connections for a vSwitch or vRouter to a specific statistics group. Possible values are from 1 through 128.	groupId *integer*
vSwitch	The name of the vSwitch associated with the statistics group.	vSwitch *name*
Type	The type of statistic counter, either VirtualService or RealService.	type {realService | virtualService}
Number of Records Encrypted	The total number of SSL records that the function card encrypted.	encRecords *integer*
Number of Bytes Encrypted	The total number of SSL bytes that the function card encrypted.	encBytes *integer*
Number of Errored Records Encrypted	The total number of SSL records with errors that the function card encrypted.	encRecordsErr *integer*
Number of Records Decrypted	The total number of SSL records that the function card decrypted.	decRecords *integer*
Number of Bytes Decrypted	The number of SSL bytes that the function card decrypted.	decBytes *integer*
Number of Errored Records Decrypted	The number of SSL records with errors that the function card decrypted.	decRecordsErr *integer*

Associated MIB

`srpCounters.mib`

Web path

- SwitchServices → tideRunner → statistics → group → sslRecord

Part VIII. Redundancy and Failover

The chapters in Part VIII describe the commands for implementing redundancy and failover in the system.

Chapter 31. Virtual Service Redundancy Protocol commands

Virtual Service Redundancy Protocol description

This chapter describes the commands for configuring the Virtual Services Redundancy Protocol (VSRP) on an N2000 Series. VSRP is a protocol that provides redundancy support for Layer 4 through 7 services. When you configure VSRP on the N2000 Series, two systems can exchange state and health information about each other and their redundant vSwitches. An election occurs, and one switch becomes the master system. If the master system has a failure, failover can occur and service traffic switches to the backup, or peer node.

For additional information about configuring VSRP and how it works with the Virtual Router Redundancy Protocol (VRRP), see the *Sun N2000 Series Release 2.0 – System Configuration Guide*.

VSRP elections

VSRP uses an election preference to determine which node should be the master node. The node with the higher election preference becomes the master node. If the local node and the peer node have the same election preference value, VSRP uses the node identifier as a tie breaker. In this situation, the node with the higher node identifier becomes the master system. If the master node is no longer available, VSRP elects a new master from one of the configured VSRP peers.

When VSRP is first configured, it remains in a backup state for a hold-down period until the first election is held. During this time, the configured virtual services are deactivated. Once the election identifies the master, the virtual services are activated on that system. Because of this, the VSRP should be configured prior to configuring virtual services.

VSRP sessions

At startup, a VSRP node tries to connect to its peer. Once they establish this session, the nodes exchange messages about their state and the services registered with VSRP. You can configure multiple VSRP sessions for redundancy purposes, so failure of a single VSRP session does not result in service failover.

vsrp command path

The command names in this chapter show you how to execute the commands from within the following command mode:

```
redundancy vsrp
```

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output step through the hierarchy from the top level to the appropriate command mode.

vsrp command summary

Table 31-1 lists and briefly describes the vsrp commands.

Table 31-1. VSRP command summary

Command name	Description
`node`	Defines attributes of the local node participating in a VSRP configuration.
`node peer`	Defines attributes of the peer node participating in the VSRP configuration.
`node peer session`	Defines a session that the local and peer nodes use to communicate with each other.

Table 31-1. VSRP command summary (continued)

Command name	Description
`show node`	Displays the local node's properties.
`show node peer`	Displays the configuration for a peer node.
`show node peer session`	Displays the configuration for the session that that the local and peer nodes use to communicate.

Virtual service redundancy protocol (VSRP) basic configuration

Table 31-2 shows the procedure for configuring a VSRP..

Table 31-2. Steps for configuring VSRP

Step	Action
1.	Configure the local node (using the `node` command)
2.	Configure the peer node for the local node (using the `node peer` command).
3.	Configure one or more sessions that the local node and peer use to exchange VSRP messages (using the `node peer session` command). The two nodes use these sessions to communicate with each other.
4.	Optionally, specify an IP address to use for redirected virtual service traffic used by the backup switch to redirect service traffic to the master switch (using the IP `address` command).
5.	Repeat steps 1 through 4 on the peer node.

node

Purpose

Defines the local node used in a VSRP configuration. The local node uses one or more VSRP sessions to exchange information with a peer node. This information exchange determines which node should handle service traffic.

The no form of the command deletes the node, in which case it operates as if it was the master node.

Access mode

config

Syntax

For creating or modifying a VSRP node:

```
redundancy node
   nodeId integer
   [port number]
   [electionPreference integer]
   [electedIncrease integer]
   [helloTime integer]
   [adminState {enabled | disabled}]
```

Arguments

Argument name	Description
`nodeId` *`integer`*	The numeric identifier associated with the local node. If you do not configure an `electionPreference`, the local and peer nodes use this value to determine which system should be the master. The system with the higher `nodeID` value becomes the master system.
`port` *`number`*	**Optional**. The number of the TCP port that the local node listens on for communication from the peer node. Valid values are 1 through 65535. The default value is 6666.
`electionPreference` *`integer`*	**Optional**. The preference assigned to the local node. The local and peer nodes use this value to determine which system should be the master. The system with the higher electionPreference becomes the master system. Valid values are 0 through 65535. The default value is 100.
`electedIncrease` *`integer`*	**Optional**. The amount to increase the advertised `electionPreference` if the local node becomes the master node. Valid values are 0 through 65535. The default value is 100. This feature can prevent unnecessary failover when a failed switch becomes operational again.
`helloTime` *`integer`*	**Optional**. The time, in seconds, that elapses before the system sends a hello message to its peer node. The shorter the time, the faster a failure is detected, requiring additional network bandwidth. Valid values are 0 through 65535. The default value is 1.
`missingHelloCount` *`integer`*	**Optional**. The number of hellos that a peer node can ignore before the system determines that the peer is down. Valid values are 1 through 10; the default value is 3.
`adminState (disabled | enabled)`	**Optional.** The administrative state of the node configuration. If `enabled`, the node participates in the VSRP configuration and VSRP establishes a TCP connection to the peer node. If `disabled`, VSRP is turned off. In the `disabled` state, the VSRP configuration remains on the system; however, the system terminates any existing TCP connections to a peer node. In this situation, the local node becomes the master node.The default value is `enabled`.

Delete filters

```
no redundancy vsrp node
   nodeId integer
   [port number]
   [electionPreference integer]
   [electedIncrease integer]
   [helloTime integer]
   [adminState {enabled | disabled}]
```

Example

The following example configures a local node to have a high electionPreference to specify that you want this system to become the master system.

```
sun> enable
sun# config
sun(config)# redundancy
sun(config-redundancy)# vsrp
sun(config-redundancy vsrp)# node nodId1 port 535 electionPreference
1500 electedIncrease 200 helloTime 5
```

Associated MIB

```
vsrp.mib
```

Web path

- Redundancy → vsrp → node → add
- Redundancy → vsrp → node → copy
- Redundancy → vsrp → node → modify
- Redundancy → vsrp → node → delete

node peer

Purpose

Defines the peer node in the VSRP configuration. The local and peer nodes use a VSRP session to exchange information that determines which node should handle service traffic.

The no form of the command deletes the peer node configuration.

Access mode

config

Syntax

For creating a peer:

```
redundancy node nodeId integer peer
   peerId integer
   [port portNumber]
   [adminState {disabled | enabled}]
```

For modifying a peer:

```
redundancy node nodeId integer peer peerId integer
   [port portNumber]
   [adminState {disabled | enabled}]
```

Arguments

Argument name	Description
`peerID` *`integer`*	The numeric identifier associated with the peer node.
`port` *`portNumber`*	**Optional.** Identifies the TCP port number associated with the peer node. Valid values are 1 through 65535. The default value is 4121. If the local node ID is greater than the peer ID, the local node initiates a connection to the peer using this port. If the local node ID is less than the peer ID, the local node uses this port to listen for a connection from the peer node.
`adminState {enabled | disabled}`	**Optional.** Sets the administrative state of the peer node configuration. If `enabled`, the peer node participates in the VSRP configuration and VSRP establishes a TCP connection between the local and peer nodes. If `disabled`, the local node does not exchange VSRP messages with the peer node. The default value is `enabled`.

Delete filters

See the Arguments section for this command for a description of the following arguments.

```
no redundancy node nodeId integer peer
   peerId integer
   [port number]
   [adminState {disabled | enabled}]
```

Example

The following example defines a peer node for the local node.

```
sun> enable
sun# config
sun(config)# redundancy
sun(config-redundancy)# vsrp
sun(config-redundancy vsrp)# node nodeId 1 peer peerId 2 port 4535
```

Associated MIB

`vsrp.mib`

Web path

- Redundancy → vsrp → node → peer → add
- Redundancy → vsrp → node → peer → copy
- Redundancy → vsrp → node → peer → modify
- Redundancy → vsrp → node → peer → delete

node peer session

Purpose

Defines a session that the local and peer nodes in the VSRP configuration use to communicate with each other. You can define multiple sessions (in the same or different vRouters) to provide resilient communications between the local and peer nodes.

The no form of the command deletes the session.

Access mode

config

Syntax

For creating or modifying a peer session:

```
redundancy node nodeId integer peer peerId integer session
   vRouter vswitch:vrouter
   peerIpAddress ipAddress
   [adminState {enabled | disabled}]
```

Arguments

Argument name	Description
`nodeId` *`integer`*	Specifies the numeric identifier associated with the local node.
`peerId` *`integer`*	Specifies the numeric identifier associated with the peer node.
`vRouter` *`vSwitch:vRouter`*	Specifies the vRouter associated with this session used for VSRP communications. The format of this argument is *`vSwitch:vRouter`*.
`peerIpAddress` *`ipAddress`*	Specifies the IP address configured on the peer node for VSRP session communication.
`adminState {enabled | disabled}`	**Optional**. Sets the administrative state of the session configuration. If `enabled`, the local and peer nodes use this connection to exchange VSRP messages. If `disabled`, the local and peer nodes do not use this session to exchange VSRP messages. If no other sessions are available, VSRP elects the local node to be the master. The default value is `enabled`.

Delete filters

See the Arguments section for this command for a description of the following arguments.

```
no redundancy node-ID peer
   nodeId integer
   peerId integer
   vRouter vswitch:vrouter
   ipAddress ipAddress
   [adminState {disabled | enabled}]
```

Example

The following example defines a session on the system vSwitch that the systems uses for VSRP communication with a peer node.

```
sun> enable
sun# config
sun(config)# redundancy
sun(config-redundancy)# vsrp
sun(config-redundancy vsrp)# node nodeId 1 peer peerId 2 session
vRouter system:shared ipAddress 10.10.45.60
```

Associated MIB

```
vsrp.mib
```

Web path

- Redundancy → vsrp → node → peer → session → add
- Redundancy → vsrp → node → peer → session →copy
- Redundancy → vsrp → node → peer → session →modify
- Redundancy → vsrp → node → peer → session →delete

show node

Purpose

Displays the properties of the local VSRP node. The `node` command configures these properties.

Access mode

user

Syntax

```
show redundancy vsrp node
```

Sample output

```
sun> redundancy
sun(redundancy)> vsrp
sun(redundancy vsrp)> show node
Node ID:                       1
Port:                          6666
Election Preference:           100
Elected Increase:              100
Hello Time:                    2
Admin State:                   enabled
Oper Status:                   down
Elected State:                 master
Current Election Preference: 200
Election Changes:              1
```

Output description

Field name	Description	Filter name
`Node ID`	The numeric identifier associated with the local node.	`nodeId` *`integer`*
`Port`	The port the local node uses to listen for VSRP communications from a peer node.	`port` *`number`*

Field name	Description	Filter name
`Election Preference`	The election preference assigned to the local node. The node with the higher election preference becomes the master node.	`electionPreference` *`integer`*)
`Elected Increase`	The amount the electionPreference increases when a VSRP node is elected the master node.	`electedIncrease` *`integer`*
`Hello Time`	The amount of time (in seconds) that elapse before the system sends a hello message to the peer node.	`helloTime` *`integer`*
`Missing Hello Count`	The number of hellos that a peer node can ignore before the system determines that the peer is down.	`missingHelloCount` *`integer`*
`Admin State`	The administrative state of the node configuration. If `enabled`, the node participates in the VSRP configuration and VSRP establishes a TCP connection to the peer node. If `disabled`, VSRP is turned off. In the `disabled` state, the VSRP configuration remains on the system; however, the system terminates any existing TCP connections to a peer node. In this situation, the local node becomes the master node.	`adminState {disabled | enabled}`
`Oper Status`	The current operational status of communication between the local and peer nodes. If `up`, at least one TCP session exists that the local and peer nodes can use to exchange VSRP messages. If `down`, no sessions exist between the local and peer nodes.	`operStatus {down | up}`

Field name	Description	Filter name
`Elected State`	The current elected state of the local node. Possible values are: `master`: The system directs virtual service traffic to the local node. `backup`: The local node is in standby mode. The system directs virtual services traffic to the VSRP peer node. `incapable`: The local node is unable to participate in VSRP configuration because it is not healthy.	`electedState {master | backup | incapable}`
`Current Election Preference`	The current value of the election preference. This value can differ from the value set with the `node` command if the local node elected state changes.	`currentElection Preference` *`integer`*
`Election Changes`	The number of times the election changed in this VSRP configuration.	`electionChanges` *`integer`*

Associated MIB

`vsrp.mib`

Web path

- Redundancy → vsrp → node

show node peer

Purpose

Displays the settings for the peer node in the VSRP configuration. The `node peer` configures these settings.

Access mode

user

Syntax

`show redundancy vsrp node peer`

Sample output

```
sun> redundancy
sun(redundancy)> vsrp
sun(redundancy vsrp)> show node peer
Node ID:                            1
Peer ID:                            2
Port:                               6666
Admin State:                        enabled
Oper Status:                        up
Last Reported Elected State:        backup
Last Reported Election Preference: 100
Locally Determined Elected State:   backup
```

Output description

Field name	Description	Filter name
`Node ID`	The numeric identifier associated with the local node.	`nodeId` *`integer`*
`Peer ID`	The numeric identifier associated with the peer node.	`peerId` *`integer`*
`Port`	The number of the port the peer node uses to listen for VSRP messages.	`port` *`number`*
`Admin State`	The administrative state of the VSRP peer node configuration.	`adminState {disabled | enabled}`

Field name	Description	Filter name
`Oper Status`	The current operational status communication between the local and peer nodes. If `up`, at least one TCP session exists that the local and peer nodes can use to exchange VSRP messages. If `down`, no sessions exist between the local and peers nodes.	`operStatus {down | up}`
`Last Reported Elected State`	The peer's election state the last time the peer sent a VSRP message to the local node. Possible values are: `active`: The peer node handles the virtual services traffic. `backup`: The peer node is in standby mode and directs the virtual services traffic only if the state of the active node changes. `incapable`: The peer node is not available or cannot participate in VSRP because it is not healthy.	`lastReportedElectedState {active | backup| incapable}`

Field name	Description	Filter name
`Last Reported Election Preference`	The peer's election preference value the last time the peer sent a VSRP message to the local node.	`lastReportedElection Preference` *integer*
`Locally Determined Elected State`	The current state of the peer node reported to the local node. Possible values are: `active`: The peer node handles the virtual services traffic. `backup`: The peer node is in standby mode and directs the virtual services traffic only if the state of the active node changes. `incapable`: The peer node is not available or cannot participate in VSRP elections.	`locallyDeterminedElected State {active | backup | incapable}`

Associated MIB

`vsrp.mib`

Web path

Redundancy → vsrp → node → peer

show node peer session

Purpose

Displays the configuration for the session that the local and peer node use for VSRP communications. The `node peer session` configures these settings.

Access mode

user

Syntax

`show redundancy vsrp node peer session`

Sample output

```
sun> redundancy
sun(redundancy)> vsrp
sun(redundancy vsrp)> show node peer session
Node ID:                    1
Peer ID:                    1
vRouter:                    system:shared
IP Address:                 10.10.10.10
Admin State:                enabled
Oper Status:                up
Peer Hello Time:            5
Seconds Since Peer Hello:   2
Hellos Sent:                25
Valid Hellos Received:      25
Invalid Hellos Received:    15
```

Output description

Field name	Description	Filter name
`Node ID`	The numeric identifier associated with the local node.	`nodeId` *`integer`*
`Peer ID`	The numeric identifier associated with the peer node.	`peerId` *`integer`*
`vRouter`	The vRouter on the local node associated with the session.	`vRouter` *`vSwitch:vRouter`*

Field name	Description	Filter name
`Peer IP Address`	The IP address of the peer node for this session.	`ipAddress` *`IPAddress`*
`Admin State`	The administrative state of the session. If `enabled`, the local node can try to use this session to exchange VSRP messages with a peer node. If `disabled`, the node cannot use this session to communicate with a peer node.	`adminState {disabled | enabled}`
`Oper Status`	The current status of the session. If `up`, the local and peer nodes can exchange messages. If `down`, the local and peer nodes cannot use this session to exchange VSRP messages.	`operStatus {down | up}`
`Peer Hello Time`	The amount of time (in seconds) between receipt of hello messages from the peer.	`peerHelloTime` *`integer`*
`Seconds Since Peer Hello`	The number of seconds since the local node received a hello message from the peer node.	`secondsSincePeerHello` *`integer`*
`Hellos Sent`	The number of hello messages that the local node sent to the peer node.	`hellosSent` *`integer`*
`Valid Hellos Received`	The number of valid hello messages that the local node received from the peer node.	`validHellosReceived` *`integer`*
`Invalid Hellos Received`	The number if invalid hello messages that the local node received from the peer node.	`invalidHellosReceived` (*`integer`*

Associated MIB

`vsrp.mib`

Web path

- Redundancy → vsrp → node → peer → session

Chapter 32. Virtual Router Redundancy Protocol commands

Virtual Router Redundancy Protocol description

This chapter describes the commands for configuring the Virtual Router Redundancy Protocol (VRRP) on an N2000 Series. VRRP provides default gateway redundancy between N2000 Series switches should a link failure occur. See RFC 2338, *Virtual Router Redundancy Protocol*, for details about this protocol.

For additional information about configuring VRRP on an N2000 Series and how it works with the Virtual Service Redundancy Protocol (VSRP), see the *Sun N2000 Series Release 2.0 – System Configuration Guide*.

VRRP on the N2000 Series

A VRRP configuration consists of N2000 Series shared and default vRouters that function as VRRP routers. When you configure VRRP on a vRouter interface, you create a VRRP router that operates as either a master or backup router in a redundant N2000 Series configuration.

VRRP election

VRRP uses an election protocol that dynamically assigns master or backup responsibility for each IP interface running VRRP. The router responsible for forwarding traffic on that interface is the master VRRP router. The other VRRP routers in the configuration function as VRRP backup routers.

If the master VRRP router becomes unavailable, the election process enables a different VRRP router to become the master, thus providing dynamic failover for IP traffic forwarding.

VRRP elections are run individually on each IP interface. A VRRP router may be master for some interfaces, and backup for others.

VRRP command path

The command names in this chapter show you how to execute the commands from within the following command mode:

`vSwitch-`*`name`* `vRouter-`*`name`* `vrrp`

The syntax shows you how to enter the command from the top level of the CLI hierarchy.

Examples and output descriptions step through the hierarchy from the top level to the appropriate command mode.

VRRP command summary

Table 32-1 lists and briefly describes the VRRP commands.

Table 32-1. VRRP command summary

Command name	Description
`interface`	Define a VRRP interface.
`show interface`	Display the settings for the VRRP router interface.
`show interface stats`	Display operational statistics for the VRRP router interface.
`show stats`	Display statistics for VRRP communications.

Table 32-1. VRRP command summary (continued)

Command name	Description
`show vrrp`	Display basic settings for VRRP configuration.
`show VRRP summary`	Display detailed information about the virtual router redundancy protocol (VRRP) currently in use.
`vrrp (root)`	Enable or disable trap generation for VRRP events. Set the VSRP preference.

VRRP basic configuration

Table 32-2. Steps for configuring VRRP

Step	Action
1.	Configure the VRRP interface (using the `interface` command).
2.	Set trap generation and VSRP interaction (using the `vrrp (root)` command).

interface

Purpose

Creates a VRRP interface on an N2000 Series vRouter and sets operational VRRP settings. Once configured, the N2000 Series vRouter functions as an elected VRRP master or backup VRRP router on that interface.

The no form of the command deletes the VRRP interface configuration.

Access mode

config

Syntax

For creating a VRRP router interface:

```
vSwitch-name vRouter-name vrrp interface
   ifName ifName
   vrid integer
   ipAddresses IPAddressList
   [primaryIpAddress IPAddress]
   [adminState {enabled | disabled}]
   [priority integer]
   [interval integer]
   [preempt {true | false}]
   [icmpEcho {enabled | disabled}]
```

For modifying a VRRP router interface:

```
vSwitch-name vRouter-name vrrp interface ifName vrid integer
   [ipAddresses IPAddressList]
   [primaryIpAddress IPAddress
   [adminState {enabled | disabled}]
   [priority integer]
   [interval integer]
   [preempt {true | false}]
   [icmpEcho {enabled | disabled}]
```

Arguments

Argument	Description
`ifName` *`IfName`*	Specifies the interface name of the lower layer associated with the VRRP router. This could be either `vlan.`*`x`*, `lag.`*`x`*, or `eth.`*`x.x`*. If the interface does not already exist, the system returns an error.
`vrid` *`IDNumber`*	Specifies the numeric identifier for the VRRP router. This identifier must match the ID on the redundant VRRP router interface. Valid values are 1 through 12.
`ipAddresses` *`IPAddressList`*	Specifies one or more VRRP IP addresses. These addresses cannot be IP addresses associated with a vRouter IP interface on the N2000 Series. Separate each address with a semicolon and enclose the list within parentheses.
`primaryIpAddress` *`IPAddress`*	**Optional**. Defines the IP address that the VRRP router uses to transmit VRRP advertisements when it is the master VRRP router. This address is an IP address associated with a vRouter IP interface. The default value is 0.0.0.0. One of the following occurs when you use the default value of 0.0.0.0: • If the vRouter interface has only one IP address, VRRP uses that address as the primary IP address. • If the vRouter interface has multiple IP addresses, VRRP uses the lowest IP address as the primary IP address.
`adminState {enabled | disabled}`	**Optional**. Specifies the administrative state of VRRP on the vRouter. If `enabled`, the N2000 vRouter participates as a VRRP router. If `disabled`, the vRouter does not participate as a VRRP router.
`priority` *`integer`*	**Optional**. Specifies the priority value of the VRRP router. The higher the value, the higher the priority. Valid values are from 1 through 255. The default value is 100.
`interval` *`integer`*	**Optional**. Specifies the time, in seconds, between transmission of VRRP advertisement messages. Valid values are 1 through 60 seconds. The default value is 1 second. You must configure the same interval for all VRRP routers associated with a single `vrid`. The size of the interval time affects the failover time. Specifying a smaller advertisement interval shortens the failover time if the master VRRP router fails.

Argument	Description
`preempt {true | false}`	**Optional**. Controls whether a backup VRRP router with a higher priority can preempt a lower priority master VRRP router. If set to `true`, the preemption can occur. If set to `false`, the backup VRRP router cannot preempt a master VRRP router with a lower priority. The default setting is `true`.
`ICMP Echo {true | false}`	**Optional**. Indicates whether the virtual router will respond to ICMP echo requests; either `enabled` or `disabled`. The default is `disabled`.

Delete filters

```
no vSwitch-name vRouter-name vrrp interface
   ifName ifName
   vrid integer
   ipAddresses IPAddressList
   [primaryIpAddress IPAddress
   [adminState {enabled | disabled}]
   [priority integer]
   [interval integer]
   [preempt {true | false}]
   [macAddr MACAddress]
   [operState {initialize | backup | master}]
   [ipAddrCount integer]
   [masterIpAddr IPAddress]
   [uptime date:time]
```

Example

The following example defines the default vRouter in the e-commerce vSwitch as a VRRP router. It is associated with VRRP identifier (VRID 1) and, if it is the VRRP master, acts as a default gateway for any of the IP addresses in the `ipAddresses` list. These VRRP IP addresses (10.10.11.40, 10.10.11.41, and 10.10.11.42) cannot be configured on an IP interface. The primary IP address (10.10.10.40) is used as the source address in VRRP updates. Setting preemption to `false` means that if this VRRP router comes online and the VRRP master is running at lower priority (or this VRRP router is running as the master and a backup has its priority increased), the current master will not be preempted.

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce vRouter default
sun(config-vSwitch-e-commerce vRouter-default vrrp)# eth.1.6 vrid 1
ipAddresses (10.10.11.40;10.10.11.41;10.10.11.42) primaryIPAddress
10.10.10.40.1 adminState enabled priority 200 interval 3 preempt false
```

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp → interface → add
- vSwitch → *name* → vRouter → *name* → vrrp → interface → copy
- vSwitch → *name* → vRouter → *name* → vrrp → interface → modify
- vSwitch → *name* → vRouter → *name* → vrrp → interface → delete

show interface

Purpose

Displays the administrative and operational status of the VRRP interface for the vRouter. The N2000 Series vRouter interfaces configured to use VRRP are called VRRP routers. These settings were configured with the `interface` command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vrrp interface`

Sample output

```
sun> enable
sun# vSwitch e-commerce vRouter default
sun(vSwitch-e-commerce vRouter-default)# vrrp
sun(vSwitch-e-commerce vRouter-default vrrp)# show interface
IfName:              lag.10
VRID:                1
IP Addresses:        10.10.10.11
Primary IP Address:  0.0.0.0
Admin State:         enabled
Oper State:          initialize
Priority:            100
Actual Priority     100
Interval:            1
Preempt Mode:        true
MAC Address:         00:00:5e:00:01:01
IP Address Count:    1
Master's IP Address: 0.0.0.0
Up Time:             1/0/1900-00:00:00
ICMP Echo:           enabled
```

Output description

Field name	Description	Filter name
`IfName`	The interface name of the lower layer associated with the vRouter.	`ifName` *`ifName`*
`VRID`	The identifier of the VRRP router. This VRID must have a matching VRID on the redundant VRRP router. The VRID range is 1 through 12.	`vrid` *`integer`*
`IP Addresses`	The VRRP IP addresses. These addresses cannot be the IP addresses associated with a vRouter IP interface on the N2000 Series.	`ipAddresses` *`IPAddressList`*
`Primary IP Address`	The IP address that the VRRP router uses to transmit VRRP advertisements when it is the master VRRP router. This address is an IP address associated with a vRouter IP interface. One of the following occurs if you use the default value of 0.0.0.0: • If the vRouter interface has only one IP address, VRRP uses that address as the primary IP address. • If the vRouter interface has multiple IP addresses, VRRP uses the lowest IP address as the primary IP address.	`primaryIpAddress` *`IPAddress`*
`Admin State`	The administrative state of VRRP on the vRouter. If `enabled`, the vRouter participates as a VRRP router. If `disabled`, the vRouter does not participate as a VRRP router.	`adminState {enabled | disabled}`

Field name	Description	Filter name
`Oper Status`	The current state of the VRRP router. Possible values are: `initialize`: The VRRP router is waiting for a startup event. Once the router receives this event, it broadcasts an ARP request that contains the virtual router MAC address, and transitions to the `master` state. If the VRRP router does not receive a startup event, it transitions to the `backup` state. `backup`: The VRRP router is the elected backup, and it monitors the availability and state of the master VRRP router. In this state, the VRRP backup is elected to master status if a failure is detected. `master`: The VRRP router is the master router. It functions as the active forwarding router.	`operState {initialize | backup | master}`
`Priority`	The priority value of the master VRRP router. The higher the value, the higher the priority.	`priority` *`integer`*
`Actual Priority`	The priority that is being sent in advertisements. If the vsrpPreference is greater than 0 and the vsrpState is master, actualPriority = priority + vsrpPreference.	`actualPriority` *`integer`*
`Interval`	The time, in seconds, between transmission of VRRP advertisement messages.	`interval` *`seconds`*
`Preempt Mode`	Specifies whether a backup VRRP router with a higher priority can preempt a lower priority master VRRP router. If set to `true`, the preemption can occur. If set to `false`, the backup VRRP router cannot preempt a master VRRP router with a lower priority.	`preempt {true | false}`

Field name	Description	Filter name
`MAC Address`	The virtual MAC address of the VRRP router.	`macAddr` *`MACAddress`*
`IP Address Count`	The number of IP addresses associated with a VRRP router.	`ipAddrCount` *`integer`*
`Master's IP Address`	The master VRRP router's real (primary) IP address. This is the IP address that the master VRRP router uses to transmit VRRP advertisements. If this vRouter is the master VRRP router, this address is the address associated with the vRouter's IP interface.	`masterIpAddr` *`IPAddress`*
`Up Time`	The date and time when this VRRP router transitioned out of an initialized state. The format of this field is: `mm/dd/yyyy-hh:mn:ss`	`uptime` *`dateAndTime`*
`ICMP Echo`	Indicates whether the virtual router will respond to ICMP echo requests.	`icmpEcho {enabled | disabled}`

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp → interface

show interface stats

Purpose

Displays operational statistics for the VRRP interface.

Access mode

user

Syntax

show vSwitch-*name* vRouter-*name* vrrp interface stats

Sample output

```
sun> enable
sun# vSwitch e-commerce vRouter default
sun(vSwitch-e-commerce vRouter-default)# vrrp
sun(vSwitch-e-commerce vRouter-default vrrp)> show interface stats
VRID:                    1
Times Master:            1
Advertisements Rcvd:     20
Ad Interval Errors:      0
IP TTL Errors:           0
Pri Zero Pkts Rcvd:      0
Pri Zero Pkts Sent:      0
Invalid Type Received:   0
IP Addr List Errors:     0
Pkt Len Errors:          0
```

Output description

Field name	Description	Filter name
VRID	The numeric identifier for the VRRP router with which this VRRP router is associated.	vrid *integer*
Times Master	The number of times this vRouter was the master VRRP router.	timesMaster *integer*
Advertisements Rcvd	The number of advertisements the VRRP router received.	adsRcvd *integer*

Field name	Description	Filter name
`Ad Interval Errors`	The number of VRRP advertisements that the VRRP router received that have an advertisement interval that is different from the interval configured for this VRRP router.	`adIntErrors` *`integer`*
`IP TTL Errors`	The total number of VRRP packets that the VRRP router received that had an IP Time-To-Live that was not equal to 255.	`ipTtlErrors` *`integer`*
`Pri Zero Pkts Rcvd`	The total number of VRRP packets that the VRRP router received that did not have a priority of 0.	`priZeroPktsRcvd` *`integer`*
`Pri Zero Pkts Sent`	The total number of VRRP packets sent by the VRRP router with a priority of 0.	`priZeroPktsSent` *`integer`*
`Invalid Type Received`	The number of VRRP packets that the VRRP router received with an invalid type in the VRRP message type field.	`invalidTypePktsRcvd` *`integer`*
`IP Addr List Errors`	The total number of packets that the VRRP router received for which the address list did not match the locally configured IP address list.	`addrListErrors` *`integer`*
`Pkt Len Errors`	The number of VRRP packets that the VRRP router received with a length that is less than the size of the VRRP header.	`pktLenErrors` *`integer`*

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp → virtualRouter → stats

show stats

Purpose

Displays statistics for VRRP communications. These statistics are kept per vRouter. If multiple VRRP routers were defined in a single vRouter, the statistics would cover all VRRP routers defined in that vRouter. Field filtering is not available for this command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vrrp stats`

Sample output

```
sun> enable
sun# vSwitch e-commerce vRouter default
sun(vSwitch-e-commerce vRouter-default)# vrrp
sun(vSwitch-e-commerce vRouter-default vrrp)> show stats
Checksum Errors:  0
Version Mismatch: 0
VRID Errors:      0
```

Output description

Field name	Description
`Checksum Errors`	The number of VRRP packets with checksum errors that the VRRP protocol stack on this vrouter received. The checksum detects data corruption in a VRRP message.
`Version Mismatch`	The total number of VRRP packets received that contained a version that is different from the VRRP version that the VRRP router uses.
`VRID Errors`	The total number of VRRP packets received that contained a VRRP router identifier (VRID) that did not match the ID for any VRRP router in this vRouter

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp → stats

show vrrp

Purpose

Displays the basic VRRP configuration settings. These values were set with the `vrrp (root)` command. Field filtering is not available for this command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vrrp`

Sample output

```
sun> enable
sun# vSwitch e-commerce vRouter default
sun(vSwitch-e-commerce vRouter-default)> show vrrp
Version:         2
VSRP State:      master
VSRP Preference: 0
Generate Traps:  disabled
```

Output description

Field name	Description
`Version`	The VRRP version in use.
`VSRP State`	The current state of the VSRP router; either master, backup, initialized, or unknown.
`VSRP Preference`	The value added to the VRRP priority if the VSRP state is master. The VRRP actual priority is calculated as vrrpPriority + vsrpPreference when the VSRP state is master. If the vrrpPriority + vsrpPreference total is greater that 254, then the vrrpPriority is set to 254. A VSRP preference value of 0 indicates that VRRP should not accept VSRP hints and should monitor the VSRP state.
`Generate Traps`	Indicates whether the system generates VRRP traps. If `enabled`, the system generates the traps; if `disabled`, it does not generate VRRP traps.

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp

show VRRP summary

Purpose

Displays detailed information about the Virtual Router Redundancy Protocol (VRRP) currently in use. Field filtering is not available for this command.

Access mode

user

Syntax

`show vSwitch-`*`name`* `vRouter-`*`name`* `vrrp summary`

Sample output

```
sun> enable
sun# vSwitch e-commerce vRouter default
sun(vSwitch-e-commerce vRouter-default)# show vrrp summary
VRouter ID:         2
IF Name:            master
VRID:               0
Priority:           disabled
Actual Priority:
Admin State:
Oper Status:
IP Addresses:
```

Output description

Field name	Description
`vRouterID`	The numeric identifier for the vRouter with which this VRRP router is associated.
`vRouter`	The vRouter on the local node associated with the VRRP interface. The format of this argument is vSwitch:VRouter.
`IfName`	The interface index of the lower layer associated with the vRouter.
`VRID`	The numeric identifier for the VRRP router with which this VRRP router is associated. Possible values are 1 through 12.
`Priority`	The priority value of the master VRRP router. The higher the value, the higher the priority.

Field name	Description
`Actual Priority`	The priority that is being sent in advertisements. If the vsrpPreference is greater than 0 and the vsrpState is master, actualPriority = priority + vsrpPreference.
`Admin State`	The administrative state of VRRP on the vRouter. If `enabled`, the Sun Application Switch vRouter participates as a VRRP router. If `disabled`, the vRouter does not participate as a VRRP router.
`Oper Status`	The current state of the VRRP router. Possible values are: `initialize`: The VRRP router is waiting for a startup event. Once the router receives this event, it broadcasts an ARP request that contains the virtual router MAC address, and transitions to the `master` state. If the VRRP router does not receive a startup event, it transitions to the `backup` state. `backup`: The VRRP router is the elected backup and it monitors the availablity and state of the master VRRP router. In this state, the VRRP backup is elected to master status if a failure is detected. `master`: The VRRP router is the master router. It functions as the active forwarding router.
`IP Addresses`	One or more VRRP IP addresses. These addresses cannot be the IP addresses associated with a vRouter IP interface on the Sun Application Switch.

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp summary

vrrp (root)

Purpose

Enables or disables the generation of Virtual Router Redundancy Protocol (VRRP) traps. Also enters the `vrrp` command mode.

Access mode

config

Syntax

```
vSwitch-name vRouter-name vrrp
   [traps {enabled | disabled}]
   [vsrpPreference integer]
```

Arguments

Argument name	Description
`traps {enabled | disabled}`	Specifies whether the system generates traps for VRRP events. `Enabled` specifies the system generates the traps; `disabled` stops generation of VRRP traps. The default value is `disabled`.
`vsrpPreference`	The value added to the vrrpPriority if the VSRP state is master. The VRRP actual priority is calculated as vrrpPriority + vsrpPreference when the VSRP state is master. If the vrrpPriority + vsrpPreference total is greater that 254, then the vrrpPriority is set to 254. The purpose of this is to favor the VSRP master switch so that it is more likely to be elected the VRRP master, generally resulting in more efficient data flow. A VSRP preference value of 0 indicates that VRRP should not accept VSRP hints and monitor the VSRP state. Valid values are 0 through 254.

Examples

The following example enables trap generation for VRRP events.

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce vRouter default vrrp
sun (config-vSwitch-e-commerce vRouter-default vrrp)# traps enabled
```

The following example sets the VSRP preference:

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce vRouter default vrrp
sun (config-vSwitch-e-commerce vRouter-default)# vsrpPreference 5
```

Associated MIB

`vrrp.mib`

Web path

- vSwitch → *name* → vRouter → *name* → vrrp → modify

Appendix A. Advanced CLI use

About this appendix

This appendix describes advanced methods that can help you enter command-line interface (CLI) commands more quickly.

Topics

This appendix includes the following topics:

TCL information

The CLI includes a Tool Command Language (TCL) interpreter that allows you to program functions in the CLI. TCL commands are fully integrated with the CLI, so that you do not have to change environments or shells. The CLI supports TCL Version 8.3.3. The CLI includes only basic TCL built-in commands. The GUI Toolkit, Safe TCL and other TCL extensions, and command reference documentation (man pages) are not included.

For information about using TCL with the CLI, see Appendix B, “TCL usage.”

Advanced methods for entering commands

This section describes methods that help you enter commands more quickly. You can use one of the following methods:

- Enter commands with argument values only.
- Enter commands using a mixture of argument values and argument names and values.
- Change the order of arguments when entering commands.

Entering commands with argument values only

A quick method of entering executable commands is to enter only the argument values without entering the corresponding arguments. To do this, you must enter the values in the correct positional order. You can display correct order from the command line by pressing [?]. Once you disrupt the order, for example, by excluding some of the arguments, you must begin entering both the argument names and values.

Example: A command with values and no argument names

The order of the argument names and values for the command to configure an NTP (Network Time Protocol) server is:

```
ntp
    id {1|2|3|4|5}
    ipAddress ipAddress
    [prefer {true|false}]
    [burst {true|false}]
    [minPoll seconds]
    [maxPoll seconds]
    [version {1|2|3|4}]
```

To configure the primary NTP server with an IP address and non-default polling intervals, you can omit the arguments if you enter the argument values in the following order:

```
sun(config-system)# ntp 3 172.26.3.10 false false 64 4096
```

Entering commands with a mixture of argument names and values

To execute a command by entering a mixture of argument values with argument names and values, you must enter the values-only entries in the expected command syntax order.

If you begin by entering only values and then omit some arguments from the syntax, you must specify the next value using the argument name and the value. After this point, you can return to entering only values, as long as you enter the values in the expected command syntax order.

Examples: A command with a mixture of argument names and values

In this example, you enter values for the first two arguments but do not enter the names. Because you skip the `prefer` and `burst` arguments, you continue the command by entering argument names *and* values for the `minpoll` and `maxpoll` arguments.

```
sun(config-system)# ntp 3 172.26.3.42 minpoll 64 maxpoll 4096
```

In this example, you enter values for the first two arguments but do not enter the names. Because you skip the `prefer` and `burst` arguments, you continue the command by entering an argument name *and* value for the `minpoll` argument. You can then enter only the value for the `maxpoll` argument because `maxpoll` is the next argument in the expected command syntax order.

```
sun(config-system)# ntp 3 172.26.3.42 minpoll 64 4096
```

Changing the order of arguments

The CLI lets you enter arguments and values in any order. In this case, however, you must use the argument names.

Example

```
sun(config-system)# ntp minpoll 64 ipAddress 172.26.3.10 maxpoll 4096
version 4 burst false prefer false id 3
```

Using filters

This section describes how to use filtering when working with configuration entries. You can use filtering to perform the following functions:

- To display specific entries that match filter criteria
- To modify specific multiple configuration entries at one time
- To delete specific multiple configuration entries at one time
- To clear counters

Parentheses indicate a set of values to use for a query followed by the value you want to change. This action changes values for all configurations that match the values in the parentheses. For example, `command (argument` *`value`* `) argument` *`value`*.

When using filters in the CLI, you can use the following methods to indicate operations or values:

- A pipe or vertical bar symbol (|) indicates an OR operation (for example, http | https).
- The range symbol (..) indicates a range of numbers (for example 1..100).
- An asterisk (*) indicates the wildcard symbol, which matches zero or more characters (for example 10.10.1.*).

Note: In the Web interface, filtering is available for `show` commands only. When using filtering in the Web interface, you can enter filtering symbols only in text boxes. If an argument has a drop-down list, you can only filter by using the selections from the drop-down list.

Filtering display output

You can specify filters with the `show` command to control the output that the system displays.

Note: The system displays only the configuration entries that match *all* of the specified filters.

Example

The following example illustrates how to use the `show` command to display all vRouter configurations that have a `vRouterName` starting with the letter “r” and an `adminState` set to `disabled`.

```
sun(config-vSwitch-vs1)# show vRouter vRouterName r* adminState
disabled
ID:                 4
Name:               r1
Description:        N/A
Admin State:        disabled
Operational Status: N/A

ID:                 5
Name:               r2
Description:        N/A
```

```
Admin State:        disabled
Operational Status: N/A
sun(config-vswitch-vs1)#
```

Using filters and wildcards for modify operations

You can use filters and wildcards when modifying existing configurations. Using these methods allows you to modify multiple configuration entries that have some of the same characteristics at one time.

Examples:

Modify configurations using wildcards and filters

In the following example, using the wildcard symbol (*) and filters allows you to configure multiple entries for all existing user entries with a priority of 1 or 2 so that they use the same vSwitch name. You can use the pipe or vertical bar character (|) to indicate an OR operation.

```
sun(switchServices userAdministration)# user (userName * priority 1|2)
vSwitchName system
6 entries were modified.
sun(switchServices userAdministration)#
```

In the following example, using the range symbol (..) allows you to modify user entries that have a priority within a specific range.

```
sun(switchServices userAdministration)# user (userName * priority
1..5) vSwitchName system
6 entries were modified.
sun(switchServices userAdministration)#
```

In the following example, using parentheses to enclose the query values allows you to change a value for all user entries that match the query values. In this situation, the system changes the `authenticationMethod` value to `alwaysAccept` for all records where `authenticationMethod` is `tacacs`.

```
sun(switchServices userAdministration)# user (userName user* priority
* authenticationMethod tacacs) authenticationMethod alwaysAccept
3 entries were modified.
sun(switchServices userAdministration)#
```

Using filters for delete operations

You can use filters to delete one or more configurations that have specific characteristics.

Example

The following example shows how to use filters to delete multiple configuration entries at one time. The system deletes only the configuration entries that match all of the specified filters. When you delete user entries, the system prompts you to confirm that you want to delete the entries. Enter `y` to continue the delete operation; enter `n` to cancel the delete operation.

```
sun(switchServices userAdministration)# no user userName user*
priority *
Do you really want to delete all entries: (y or n)? y

3 entries were deleted.
```

Appendix B. TCL usage

About this appendix

This appendix describes the type of Tool Command Language (TCL) support that is available in the N2000 Series command-line interface (CLI). See Chapter 1, “Using the management interfaces” and Appendix A, “Advanced CLI use” for additional information about using the CLI.

Topics

This appendix includes the following topics:

TCL overview

TCL (pronounced “tickle”) is a simple yet powerful scripting language that lets you program functions in the N2000 Series CLI. The CLI includes a TCL interpreter and supports TCL Version 8.3.3

TCL commands are fully integrated with the CLI, so that you do not have to change environments or shells. The CLI includes only basic TCL built-in commands; it does not include TK (GUI Toolkit), Safe TCL, other TCL extensions, or command reference documentation (man pages).

TCL references

This document does not explain how to use TCL or describe the syntax for the supported TCL commands. If you are unfamiliar with using TCL, refer to the following information sources:

- Complete TCL documentation is at `http://www.scriptics.com/doc/`.
- For a TCL tutorial, *Getting Started with TCL*, see `http://www.scriptics.com/scripting/`.
- For a TCL introduction and overview, see `http://www.scriptics.com/scripting/primer.html`.
- For an introduction to TCL syntax, see `http://www.scriptics.com/scripting/syntax.html`.
- For TCL command reference documentation (man pages), see `http://www.scriptics.com/man/tcl8.3/TclCmd/contents.htm`.
- *TCL and the TK Toolkit* by Dr. John Ousterhout (Addison-Wesley, ISBN 0-201-63337-X).

TCL commands

The following table lists the TCL commands that the CLI supports. For details about command syntax, refer to your TCL documentation.

Table B-1. Supported TCL commands

Command	Description
`append`	Append to a variable.
`array`	Manipulate array variables.
`auto_import`	Determine whether specified imported commands reside in an autoloaded library and load the commands.
`auto_load`	Attempt to load the definition for a command.
`auto_qualify`	Compute a list of fully qualified names for a command.
`binary`	Insert and extract fields from binary strings.
`break`	Abort a looping command.
`case`	Evaluate one of several scripts, depending on a pattern match. (This command is deprecated. Use the `switch` command instead.)
`catch`	Evaluate a TCL script and trap exceptional returns.
`cd`	Change working directory.
`close`	Close an open channel.
`concat`	Join (concatenate) lists.
`continue`	Skip to the next iteration of a loop.
`cp`	Copy a file to a specified destination.
`eof`	Check for end-of-file condition on a channel.
`error`	Generate an error and display a message.
`eval`	Evaluate a TCL script.
`expr`	Evaluate an expression.
`fblocked`	Determine whether the last input operation exhausted all available input.
`fconfigure`	Set and retrieve options on a channel.

Table B-1. Supported TCL commands (continued)

Command	Description
`fcopy`	Copy data from one channel to another.
`file`	Manipulate a file and its attributes.
`for`	Execute a loop.
`foreach`	Iterate over all elements in one or more lists.
`format`	Format a string in the same way as the ANSI C `sprintf` command.
`gets`	Read the next line from a channel.
`glob`	Return names of files that match patterns.
`global`	Access global—rather than local—variables.
`if`	Execute a TCL script conditionally.
`incr`	Increment the value of an integer variable.
`info`	Return information about the state of the TCL interpreter (for example, variables and procedures).
`join`	Create a string by joining the elements of a list.
`lappend`	Append elements of a list to a variable.
`lindex`	Retrieve an element from a list.
`linsert`	Insert elements into a list.
`list`	Create a list.
`llength`	Count the number of elements in a list.
`lrange`	Return a new list containing sequential elements from an existing list.
`lreplace`	Return a new list formed by replacing elements of a list with new elements.
`ls`	List the contents of a directory. A Sun addition.
`lsearch`	Determine whether a list contains a particular element.
`lsort`	Sort the elements of a list and return a new list in sorted order.
`mkdir`	Create a directory.
`mv`	Move a file to a specified destination.

Table B-1. Supported TCL commands (continued)

Command	Description
`namespace`	Create and manipulate contexts for commands and variables.
`open`	Open a file, serial port, or command pipeline and return a channel identifier.
`package`	Manipulate a database of packages available for use by the current TCL interpreter.
`pause`	Pause a TCL script for a specified number of seconds. This command is a Sun addition. See "Using the pause command" on page B-9 for additional details.
`pid`	Retrieve process id(s).
`proc`	Create a TCL procedure.
`puts`	Write characters to a channel.
`pwd`	Return the path name of the current working directory.
`read`	Read characters from a channel.
`regexp`	Determine whether a regular expression matches a string.
`regsub`	Substitute strings in variables using regular expression pattern matching.
`rename`	Rename or delete a TCL command.
`return`	Return from the current procedure.
`rm`	Removes a file or directory.
`scan`	Parse a string using conversion specifiers in the same way as the ANSI C `sscanf` command.
`seek`	Change the current access position for an open channel.
`set`	Create, read, or write a variable.
`source`	Pass a file to the TCL interpreter as a text script.
`split`	Split a string into a proper TCL list.
`string`	Manipulate a string in various ways, for example, compare, search, and character type.

Table B-1. Supported TCL commands (continued)

Command	Description
`subst`	Substitute variables, commands, and backslashes in a string.
`switch`	Evaluate one of several scripts, depending on a pattern match.
`tell`	Return the current access position in an open channel.
`tclflush`	Flush any output that has been buffered for a channel.
`trace`	Execute a TCL command whenever a given variable is accessed.
`unknown`	Handle attempts to invoke non-existent TCL commands.
`unset`	Delete one or more variables.
`uplevel`	Execute a script in a different level of the procedure calling stack.
`upvar`	Create a link to a variable in a different level of the procedure calling stack.
`variable`	Create and initialize a namespace variable.
`while`	Execute a script repeatedly as long as a Boolean condition is true.

TCL use guidelines

Consider the following guidelines when you use TCL:

- You can use the TCL interpreter in all CLI access and command modes.
- TCL commands are case-sensitive. Use lowercase characters for all commands. For example, `expr` is a valid command, but `EXPR` is not.
- You can use the same editing and navigation keys with TCL as the ones you use with the CLI. See Chapter 1, “Using the management interfaces” and Appendix A, “Advanced CLI use” for details.

- The TCL interpreter does not include any of the CLI Help functions. For example, the CLI `?` query does not return any TCL commands or keywords and pressing the [Tab] key does not complete any TCL commands or keywords. TCL error messages, however, do provide some syntax help. For example:

  ```
  sun> expr [Enter]
  wrong # args: should be "expr arg ?arg ...?"
  ```

- You can install a TCL startup script as file `.tclshrc` in directory `/ftl0`. If this file is present, the TCL interpreter evaluates it when the CLI first starts.

Using numbers

TCL commands accept both decimal and hexadecimal numbers, and return decimal numbers. For example:

```
sun> set num 0xf
0xf
sun> incr num
16
sun> puts $num
16
sun> expr pow($num,4)
65536.0
sun> expr 0xa * 10
100
```

Using the ls command

The `ls` command lists the contents of a directory. This command is a Sun addition, and it functions like most UNIX shell `ls` commands.

Syntax

`ls [-a] [-l] [`*`name ...`*`]`

Arguments

Argument	Description
`-a`	List all files, including hidden files beginning with a period (.).
`-l`	List contents in long format, which includes directory designator, user/group/other permissions (read, write, execute), size in bytes, date and time last modified, and name. Subdirectory names end with a slash (/).
`name ...`	Display information about the named files or directories. You must enter the full file or directory name(s). Wildcards are not allowed. Without this argument, the `ls` command lists the contents of the current directory.

Examples

In this example, the `ls` command, without any arguments, shows a listing of the current directory.

```
sun> ls
Directory '/ftl0/'
nosConfig
errhnd
evtMgr.elf.stub
halagent.elf.stub
logMgr.elf.stub
ui.elf.stub
lib/
cdb.bak
cdb.dat
```

In this example, the `ls -l` command shows a detailed listing of the files in the current directory.

```
sun> ls -l
Directory '/ftl0/'
-rw-rw-rw-       180   1 Jan 2002 00:10:35 nosConfig
-rw-rw-rw-      9999  14 Jan 2002 16:48:45 errhnd
-rw-rw-rw-      4616  14 Jan 2002 16:50:06 evtMgr.elf.stub
-rw-rw-rw-      4616  14 Jan 2002 16:50:08 halagent.elf.stub
-rw-rw-rw-      4616  14 Jan 2002 16:50:10 logMgr.elf.stub
-rw-rw-rw-      4616  14 Jan 2002 16:50:44 ui.elf.stub
drwxrwxrwx       512   1 Jan 2002 00:04:12 lib/
-rw-rw-rw-       649   9 Jan 2002 19:08:55 cdb.bak
-rw-rw-rw-       685  14 Jan 2002 16:51:25 cdb.dat
sun>
```

In this example, the `ls -l` command with a file name shows a detailed listing for the specified file.

```
sun> ls -l cdb.dat
-rw-rw-rw-       685  14 Jan 2002 16:51:25 cdb.dat
```

Using the pause command

The `pause` command lets you pause a TCL script or the CLI. This command is a Sun addition.

Syntax

`pause` *`seconds`*

Argument

Argument	Description
seconds	The number of seconds to pause. The valid range is 0 through 4294967295.

Example

In this example, the `pause` command inserts a 30-second pause in the script.

```
sun# proc test args {
vSwitch create
pause 30
puts “vSwitch is fully up and working.”
}
```

TCL examples

This section contains simple examples of using TCL with the CLI to quickly complete configuration tasks.

Configuring multiple vSwitches with a script

```
sun(config)# foreach i {eng pubs fin} {vSwitch $i;
  exit;
};
create new vSwitch "eng" ? (y or n): y
.....................................

create new vSwitch "pubs" ? (y or n): y
.....................................

create new vSwitch "fin" ? (y or n): y
.....................................
sun(config)# show vSwitch
Name:               eng
ID:                 2
Description:        N/A
Admin State:        enabled
Operational Status: up

Name:               fin
ID:                 4
Description:        N/A
Admin State:        enabled
Operational Status: up

Name:               pubs
ID:                 3
Description:        N/A
Admin State:        enabled
Operational Status: up

Name:               system
ID:                 0
Description:        System vSwitch
Admin State:        enabled
Operational Status: up
```

Creating variables for CLI commands

This example shows how to use TCL to create a variable that you can use in CLI commands. The variable in this example represents an IP address for an NTP server. You can use the variable in CLI commands instead of entering the server's IP address.

Note: Variables persist only for the duration of the current CLI session. If you restart the CLI, you must re-create any TCL variables.

```
sun(switchServices)# set ip 172.26.3.10
172.26.3.10

sun(switchServices)# ntp primary $ip minpoll 256 maxpoll 2048
sun(switchServices)# show ntp * $ip
Server type:        Primary
Server IP Address:  172.26.3.10
Server Preference:  N/A
Burst Mode:         N/A
Min. Poll Interval: 256
Max. Poll Interval: 2048
Version:            N/A

sun# event
sun(event)# syslog $ip loglevel warning
sun(event)# show syslog $ip
SysLog           Syslog Log
Host             Port   Level
172.26.3.10      N/A    warning
```

TCL error messages

The TCL interpreter displays descriptive error messages if you enter invalid commands or keywords, or if an operation fails. This section lists the common error messages.

Table B-2. TCL error messages

Command and error	Description and solution

`sun# append [Enter]`
 	
`wrong # args: should be "append varName ?value value ...?"`
	The command is incomplete, and the error message displays the correct syntax. Enter the command with its required arguments.

`sun# ntp sec $pi`
 	
`can't read "pi": no such variable`
	The variable does not exist. Create a variable before you use it. Variables persist only for the duration of the current CLI session. If you restart the CLI, you must re-create any TCL variables. Use the TCL `info vars` command to display current variables. Variables are case-sensitive. Correct any typing errors.

`sun# ls -l foo.txt`
 	
`Could not stat 'foo.txt': No such file or directory.`
	The file does not exist in the current directory. This result is not necessarily an error; the file may be in a different directory. Check the file name and correct any typing errors.

`sun# Expr 64 ^ 8`
 	
`ERROR: "Expr" is not a recognized command`
	TCL commands are case-sensitive and are lowercase. Enter the command in all lowercase characters.

`sun# set pi 3.14159`
 	
`3.14159`
 	
`sun# incr pi`
 	
`expected integer but got "3.14159"`
	You can increment only integer variables. Use `expr` to add to a real variable, or change the variable to an integer.

`sun# expr pi * 5`
 	
`syntax error in expression "pi * 5"`
	Prefix a variable with $ to substitute its value in a command. In this example, use `$pi`.

TCL and scripted server health checks

Using TCL scripts in conjunction with the SCRIPT health check probe type enables you to write customized server health checks to verify real service availability. Executed within the virtual switch that is running the server health check, the script can range from simple TCL commands to complicated protocol-specific server health checks. Sharing many of the same arguments that define the other types of health checks, scripted health checks also have specific parameters such as `scriptFile`, `scriptCommands`, and `scriptArgs`. For information about the specific scripted health check profile arguments and the other health check probe types, see Chapter 29, "Load-balancing commands."

The most significant difference between scripted health checks and the other types of health checks is the higher volume of system resources required by scripted health checks. Rather than performing the protocol-level exchange directly from a single application like the other health check types, the scripted health check process creates a task to run a new, freshly initiated TCL interpreter each time a script is executed. Although a new, separate interpreter ensures the same predictable starting point for each server probe, this task places a great demand on system resources.

The execution of a scripted health check begins with the evaluation of the TCL `scriptFile` (when specified). Although the script file can contain TCL commands, it is recommended that the file contain TCL procedures. These procedures can then be launched by the `scriptCommands` argument. This is an easier way of writing scripts and provides an additional method for passing arguments.

If no errors are detected during the evaluation, the script commands, which can be simple TCL commands or process execution statements, are executed. During command execution, TCL errors, indicating a failure, are reported. To ensure that a success status is reported if there are no TCL errors, you can use script logic to set a particular error code as an "exit code." An exit code of 0 indicates success; a non-zero exit code indicates failure. If an exit code is not set, and no TCL errors are reported, successful execution is assumed.

The following topics describe:

- TCL global variables that are defined within the procedures contained in a .tcl file, which is stored in flash memory.
- Health check profile arguments that you use to define the SCRIPT health check profile. You then specify the stored TCL script (.tcl file) in the SCRIPT health check profile.

An example is also provided to show how to use TCL scripts with scripted server health checks.

TCL global variables

Real service target information is passed by the following global variables used within a defined TCL procedure.

- `global shc_real_service_ip`
- `global shc_real_service_port`

Optional global arguments may be available, depending on the configuration of the scripted health check profile.

- `global shc_arg1`
- `global shc_arg2`
- `global shc_arg3`
- `global shc_arg4`
- `global shc_state_info` — Can be used to return state information about the last execution of the script. This value will then be available for the next script execution of the probe. The state information can include values such as the status of the previous script run, an incrementing counter, or a history summary. This variable can contain string data between 0 and 32 characters in length.
- `global shc_exit_code` — Can be used to indicate the successful execution of the script. The return of 0 indicates a successful execution; the return of non-zero indicates failure. The exit code can also be used to signal different types of errors to give more visibility into the cause of the error. For example, you could set an exit code of 4 to indicate a “page not found” failure.

Health check profile arguments

The following are optional arguments for the SCRIPT health check profile. You set the values using the `healthCheckProfile` command. For descriptions of these arguments, see Chapter 29, "Load-balancing commands."

- scriptMode
- `scriptFile` *`text`*
- `scriptCommands` *`text`*
- `scriptArg1` *`text`*
- `scriptArg2` *`text`*
- `scriptArg3` *`text`*
- `scriptArg4` *`text`*
- profileDescription *`text`*

Note: If you specify a `scriptFile` in the health check profile, it is recommended that the file contain defined TCL procedures (`proc`). You should then use the `scriptCommands` argument to launch the procedures with calling arguments if required. Optionally, you can specify up to four arguments in the health check profile.

The following are the arguments that are shared by all health check probe types. For descriptions of these arguments, see Chapter 29, "Load-balancing commands."

- interval *`seconds`*
- retries *`probes`*
- successRate *`percentage`*
- timeout *`seconds`*
- count *`probes`*

TCL and scripted server health check example

The following example shows a test script file for use in scripted server health checks. Comments (#) are included in the file to describe the variables and arguments that are available.

```
# ------------------------------------------------------------------------
#  Description: this is a test script file that demonstrates some of the
#  capabilities of the scripted health checks.  The script opens a socket to
#  a target, and sends an http request.  Note that the target url does not have
#  to be valid. Completion of opening and sending the request indicates that the
#  server is at least available.
#
#  The TCL test script (shc_script.tcl) uses the following arguments:

#  probeLevel - defines the depth (0 to 5 and above) of the probe for
#               performing a full http get of the root file.
#
#  debugMode - defines whether the proc will log output to a file.
#              Since this operation is quite slow, the debug
#              mode should be used only to verify script operation.
#              The debugMode argument should not normally be enabled.
#
#  Additionally, the proper response can be retrieved and verified, or
#  simply ignored.
#
#  This example uses the following displayed health check profile (hc1) and
#  the test script file shc_script.tcl, both of which reside in flash memory
#  in directory /ftl0.

#    sun(config-vSwitch-vsw1 loadBalance)# show healthCheckProfile
#    Name:                        hc1
#    Type:                        SCRIPT
#    Interval:                    5
#    Retries:                     3
#    Success Rate:                0
#    Timeout:                     2
#    Count:                       3
#    Script File:                 /ftl0/shc_script.tcl
#  **Script Commands:             shc-script-http 5
```

```
#    Script Arg1:                value_for_arg1
#    Script Arg2:                value_for_arg2
#    Script Arg3:                value_for_arg3
#    Script Arg4:                value_for_arg4
#
# ** The specified Script Commands indicates that this procedure is called
#    in ‘quiet’ mode with debugMode OFF(0) by default. If debugMode
#    were enabled, individual host/port files would be written
#    to the file system.  To enable debugMode, set the value to ON(1) as the
#    second calling argument as shown below.
#
#    Script Commands:            shc-script-http 5 1
#
#    With debugMode set to 1, individual real service activity is logged
#    in a file, shc_<ipAddress>p<port>.txt.
#
#    If the hostIp value is 192.168.124.238, port 80, the output file would
#    be named:
#    shc_192.168.124.238p80.txt
# ------------------------------------------------------------------------
#
proc shc-script-http { { probeLevel 0 } { debugMode 0 } } {

   # In the TCL proc, real service target information is passed using the
   # following global variables.

   global shc_real_service_ip
   global shc_real_service_port
   #
   # The following optional arguments may be used, depending on the
   # configuration of the SHC Profile for the scripted health check.
   #
   global shc_arg1
   global shc_arg2
   global shc_arg3
   global shc_arg4

   # The global variable shc_state_info stores script state,
   # which can be used to pass state information regarding the last
   # execution of the script.  The variable can contain string data
   # between 0-32 chars in length.  This value will be available for
   # the next probe’s script exection.  The script state can include
```

```
# information such as the status of the previous run, an
# incrementing counter, a history summary, or average successes. It
# can be a single value, a list of values, or just a summary.
#
global shc_state_info


# The global variable 'shc_exit_code' can be used to indicate script
# execution success or failure.  If it is not set, and there are no
# other TCL errors, success is assumed. The possible values of the
# shc_exit_code are 0 = success, non-zero = error.
#
global shc_exit_code


# Initialize the returned error first, and clear it upon success.
#
set shc_exit_code 10

# Open a file to catch the output and dump the internal
# variables that are available to the script.
#
# set dyn_output_file "/ftl0shc_$shc_real_service_ip_
# $shc_real_service_port.txt"
#
if { $debugMode != 0 } {
  set dyn_output_file "/ftl0/shc_"
  append dyn_output_file "$shc_real_service_ip"
  append dyn_output_file "p"
  append dyn_output_file "$shc_real_service_port"
  append dyn_output_file ".txt"

  set outputName  "$dyn_output_file"

  # Open the file and rewrite it each time (w+) rather than appending
  # to it (a+) since appending will cause the file to grow rapidly.

  set gotError [ catch { set fname [ open "$outputName" w+ ] } ]

  catch { puts $fname "  shc-script-http...                                  " }
  catch { puts $fname "                                                      " }
  catch { puts $fname "           Dynamic FileName: $dyn_output_file         " }
  catch { puts $fname "               calling arg :  $calling_arg            " }
  catch { puts $fname "              RS IP Address:  $shc_real_service_ip    " }
  catch { puts $fname "              RS Port Num:  $shc_real_service_port    " }
  catch { puts $fname "                                                      " }
  catch { puts $fname "               State_Info:   $shc_state_info          " }
  catch { puts $fname "                     arg_1:  $shc_arg1                " }
  catch { puts $fname "                     arg_2:  $shc_arg2                " }
  catch { puts $fname "                     arg_3:  $shc_arg3                " }
  catch { puts $fname "                     arg_4:  $shc_arg4                " }
```

```
  catch { puts $fname "                                                    " }
  catch { puts $fname "                                                    " }
}

# Initialize the errors to zero
#
set sockError 0
set fconError 0
set putsError 0
set recvError 0
set rsSock "not-set"
set stagesSet ""
set inLIne ""

if { $probeLevel > 1 } {
  append stagesSet "socket"
  set sockError [ catch { set rsSock [ socket $shc_real_service_ip
       $shc_real_service_port ] } ]
} else {
  set rsSock "not-set and not-required"
}

if { $probeLevel > 2 } {
  append stagesSet " fconfig"
  set fconError [ catch { fconfigure $rsSock -buffering none -eofchar {} } ]
}

if { $probeLevel > 3 } {
  append stagesSet " get"
  set putsError [ catch { puts -nonewline $rsSock "GET / HTTP/1.0\n\n" } ]
}

if { $probeLevel > 4 } {
  append stagesSet " receive"
  set recvError [ catch { set inLine [ read $rsSock ] } ]
}

# Clean up the socket, if it was opened
#
if { $rsSock != "not-set" } {
  append stagesSet " closed-socket"
  catch { close $rsSock }
}

# Gather updated info about the current probe count, which will be used
# to provide additional state information about the probe success or
# failure.  It will also provide a probe 'count' value in case it
# is needed.
#
set l_len 0
catch { set l_len [ llength $shc_state_info ] }
```

```
   if { $l_len >= 5 } {
     set prev_probe [ lindex $shc_state_info 4 ]
   } else {
     set prev_probe 0
   }
   set probeCount [ expr $prev_probe + 1 ]

   # Summarize the errors, and set the appropriate return code and state info.
   #
   if { $sockError == 0 && $fconError == 0 && $putsError == 0 &&
           $recvError == 0 }{
      set shc_exit_code 0
      set shc_state_info "prev probe was OK $probeCount"
   } else {
      set shc_exit_code 1
      set shc_state_info "prev probe was NotOK $probeCount"
   }

   if { $debugMode != 0 } {
      catch { puts $fname "                  rsSock: $rsSock                  " }
      catch { puts $fname "                  inLine: $inLine                  " }
      catch { puts $fname "                  stages: $stagesSet               " }
      catch { puts $fname "                                                   " }
      catch { puts $fname "                 exit code:  $shc_exit_code        " }
      catch { puts $fname "                                                   " }

      if { $shc_exit_code == 0 } {
         catch { puts $fname " Success! sock: $sockError,  fconf: $fconError" }
         catch { puts $fname "          puts: $putsError,   recv: $recvError" }
      } else {
         catch { puts $fname " ERROR!   sock: $sockError,  fconf: $fconError" }
         catch { puts $fname "          puts: $putsError,   recv: $recvError" }
      }

      catch { close $fname }
   }

}
```

Appendix C. Object rule predicate statements

About this appendix

This appendix lists the predefined field names, operators, and keywords, and keywordSets that are available for writing predicate statements for object rules.

Topics

This appendix includes the following topics and tables:

Terminology

Before configuring L5 to L7 load balancing, you should be familiar with the following terminology.

Object

An *object* is a message with a defined start point and end point within an application protocol stream layered over TCP. An HTTP request (client to server) and an HTTP response (server to client) are both objects.

Predicate

A *predicate* is a filter that uses one or more match expressions (joined by logical operators), which the load balancer uses to match inbound or outgoing traffic. A predicate can examine all or parts of the Uniform Resource Identifier (URI), HTTP header fields, cookies, and other request data, and can perform integer and string comparisons with prefix, suffix, and substring type operations. The predicate is true if the expression matches the object, or false if the expression does not match.

Object rule

An *object rule* is a named statement that describes how the switch must evaluate an object such as an HTTP request, HTTP response, or Uniform Resource Identifier (URI). An object rule defines a predicate that is compared to incoming or outgoing traffic. If the traffic matches the object rule, a request or response policy will execute a specific action such as forward or sorry, or apply specific header transformations.

Host

A *host* is a machine, such as a backend server, with an assigned IP address. The IP address configured for load balancing is the IP address of the actual server to be load balanced. Hosts of interest for load balancing are those machines running server applications.

Real service

A *real service* is a server application executing on a host. A real service is identified by its host name and the port on which the server application is running. Real services are grouped into service groups for load balancing.

Request policy

A *request policy* is a virtual service policy that is evaluated to make a forwarding decision to a service group or to perform the action defined by a sorry service if the HTTP request received from a client matches the object rule. The request policy specifies an object rule, the action data required to forward the HTTP traffic, the sorry service, and the service group that will receive the request. The request policy must be configured in the virtual service definition. All L5 to L7 load-balancing applications require a request policy for each service group.

Request transform

A *request transform* defines any changes to the HTTP header text of an HTTP request if the traffic matches the object rule. Changes that can be defined include header insertions, such as source IP address, cipher strength, and customer headers. The request transform is optional, but if used, the request transform must be configured in the service group definition.

Response policy

A *response policy* is a service group policy that defines how to process an HTTP response. If the traffic matches the object rule, the options are to return the HTTP object to the client; retry the original request to the host server; or apply a sorry service action. A response policy is usually used to handle error returns from the servers. A response policy is optional, but if used, the response policy must be configured in the service group definition.

Response transform

A *response transform* defines changes to the HTTP header text of the HTTP response if the traffic matches the object rule. Changes that can be defined include server cloaking, which is the removal of server signature from HTTP headers. The response transform is optional, but if used, the response transform must be configured in the service group definition.

Service group

A *service group* is a collection of real services, all capable of fulfilling a classified request, distinguished only by relative health or load capacity (load-balancing algorithm). When a request policy indicates that an object will be forwarded, it will specify a service group across which requests will be distributed.

Sorry service

A *sorry service* is a named component that specifies the action to take when the action field in a request or response policy is set to sorry. A sorry service can be defined to close a TCP connection, return an HTML page to a client, or redirect a request to a different URI. The sorry service is configured in the definition of a request or response policy.

Virtual service

The *virtual service* configures the client side of the configuration for the server load balancer. When a request comes in from the client, the virtual service compares the request to the request policies in the list based on the precedence of the request policy. When a match is found, the request policy has a service group associated with it, and the request can be forwarded to that service group. The system then load balances across the service group. The virtual service defines the load-balancing application type, the virtual service IP address (VIP) and the vSwitch:vRouter for network traffic. The virtual service links one or more request policies to the VIP address. A virtual service is configured for each vSwitch.

Object rule details — predicates and actions

Table C-1 lists the HTTP Request and HTTP Response header field names that you can supply in an object predicate, along with some examples of their usage.

Table C-1. HTTP Request and Response header predicates

Field name	Description

ACCEPT	HTTP Request header; client specifies the content type it can accept in the message body of the HTTP response.
	
	Type: string
	
	Example:
	`{ACCEPT matches "*/*"}`
	
	Example:
	`{ACCEPT matches "text/*"}`

ACCEPT_LANGUAGE	HTTP Request header; client specifies the preferred language to be supplied in the HTTP response. The first two letters are the ISO 639 language designation; the second two letters are the ISO 3166 country code. Type: string **Example:** `{ACCEPT_LANGUAGE eq "ja-jp"}`
ACCEPT_ESI (Edge Side Includes)	HTTP Request header; client specifies an Akamai-sourced HTTP request. Type: string **Example:** `{ACCEPT_ESI present}`

CONNECTION	General; supports persistent and non-persistent connections. CONNECTION informs the client whether the server will close a connection after sending a response, or whether it will keep the connection persistent.
	
	Type: keywordSet (See Table C-5.)
	
	Example:
	`{CONNECTION contains close}`
	
	Example:
	`{CONNECTION contains keep-alive}`

Table C-1. HTTP Request and Response header predicates (continued)

Field name	Description
CONTENT_LENGTH	Allows examination of the size of the message body in bytes. Type: integer **Example:** {CONTENT_LENGTH < 40000} **Note:** Valid with HTTP Request Method of POST. See METHOD.
COOKIE	HTTP Request; client includes any preferred cookies that it has received from a server (Set-Cookie in an HTTP response) in subsequent requests to that server using the cookie header. Type: string **Example:** {COOKIE has “session-id” eq “105”}
HOST	HTTP Request; client includes the host URL of the Web server. Type: string **Example:** {HOST eq “www.e-commerce.com”} **Note:** Derived from HOST_HEADER or URI_HOST. If you specify the HOST field name, the switch first checks for the URI_HOST field definition. If URI_HOST does not exist, then the switch checks for the HOST_HEADER field.
HOST_HEADER	HTTP Request; client includes the host URL of the Web server. Type: string **Example:** {HOST_HEADER eq “www.e-commerce.com”}
HOST_HEADER_PORT	HTTP Request; client includes the TCP port that the Web sever application protocols should use. TCP Port 80 is the expected port for HTTP requests. Type: integer **Example:** {HOST_HEADER_PORT == 80}

Table C-1. HTTP Request and Response header predicates (continued)

Field name	Description
REFERER	HTTP Request (optional); client specifies where it got the URL specified in the HTTP request. Web sites that provide links to other sites are the “referral” sites. Type: string **Example:** `{REFERER eq “http://www.e-commerce.com/default/relatedlinks.htm”}`

TRANSFER_ENCODING	General; indicates the transfer encoding format applied to the HTTP message body.
	
	Type: keywordSet (See Table C-5.)
	
	Example:
	`{TRANSFER_ENCODING contains chunked}`
	
	Chunked encoding breaks up the message body into chunks to improve Web server performance. The server begins sending the response as soon as it begins composing the response. The last chunk has a size of 0 bytes.
	
	Example:
	`{TRANSFER_ENCODING contains gzip}`
	
	Note: The `gzip` keyword indicates that the message body is compressed and reduces transmission time.

METHOD	HTTP Request; client specifies the method to be performed on the object identified by the URL. The `METHOD` is the first field name in the HTTP request line. Type: keyword (See Table C-4.) **Example:** `{METHOD is GET}` **Note:** `Methods` are `GET`, `HEAD`, `POST`, `PUT`, `DELETE`, `CONNECT`, `TRACE`, `OPTIONS`, `PROPFIND`, `PROPPATCH`, `MKCOL`, `COPY`, `MOVE`, `LOCK`, `UNLOCK`

Table C-1. HTTP Request and Response header predicates (continued)

Field name	Description
HTTP_VERSION	HTTP Response; specifies the HTTP protocol version that the client or servers are able to support. The HTTP_VERSION follows the URI field name in the HTTP request line. Type: string Sample HTTP request line: GET / HTTP/2.0 200 OK **Example:** {HTTP_VERSION eq “HTTP/2.0”}
PORT	HTTP Request; client includes the TCP port that the Web sever application protocols should use. TCP Port 80 is the expected port for HTTP requests. Type: integer **Example:** {PORT == 80} **Note:** Derived from HOST_HEADER_PORT or URI_PORT. If you specify the PORT field name, the switch first checks for the URI_PORT field. If URI_PORT does not exist, then the switch checks for the HOST_HEADER_PORT field.
UPGRADE	General; client requests and negotiates an HTTP protocol upgrade with the server. Type: string **Example:** {UPGRADE eq “HTTP/2.0”}
USER_AGENT	General; identifies the client or browser implementation of HTTP. Type: string **Example:** {USER_AGENT matches “*Mozilla/4.0*”}

Table C-1. HTTP Request and Response header predicates (continued)

Field name	Description

RANGE	HTTP Request; client requests only partial content instead of the entire resource (in bytes).
	
	Type: string
	
	Example: Byte range
	`{RANGE eq “bytes 100-200”}`
	
	Example: Final 200 bytes
	`{RANGE eq “-200”}`
	
	HTTP Response: 206 Partial Content, 416 Request range not satisfiable

RESPONSE_CODE	HTTP Response; response status codes returned to the client. Used only with `enableRetryServices` and `retryServicePredicate` forwarding actions (see Table C-6). Type: integer **Example:** `objectRule OR1 predicate {URI_SUFFIX eq “org”} action forward enableRetryServices true retryServicePredicate {RESPONSE_CODE != 404} sorryServiceType page sorryString “/ft0/sorrypage.html”` In this example, if a backend server returns a response code not equal to 404 (NOT FOUND), the switch attempts a retry to the backend server. If the retry fails, the sorryServices Web page is returned to the client. Status codes: • 100-199: Informational; final result not available. • 200-299: Success; the HTTP request was successful. • 300-399: Redirection; the client should redirect the HTTP request to a different server. • 400-499: Client error; the HTTP request contained an error and the server was unable to complete the request. • 500-599: Server error; the server failed to act on the HTTP request, even if the request was valid.

Predicates with URI field names

Figure C-1 illustrates the Uniform Resource Identifier (URI) structure.

Figure C-1. Uniform Resource Identifier structure

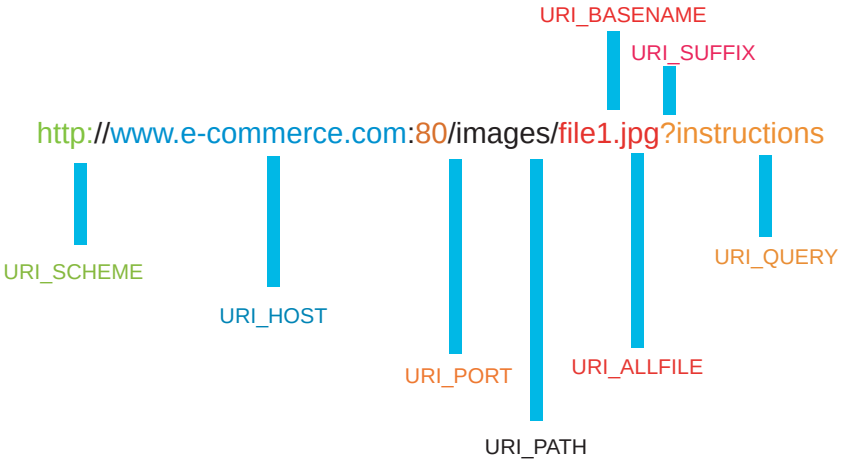


Table C-2 lists the supported URI field names.

Table C-2. HTTP Uniform Resource Identifier predicates

Field name	Description
URI	HTTP Request; specifies the complete Uniform Resource Identifier (URI) string to the Web server resource. Type: string **Example:** `{URI eq “http://www.e-commerce.com:80/images/file.jpg?instructions”}`
URI_SCHEME	Within URI; identifies the application protocol (HTTP) used to access the Web server(s). Type: string **Example:** `{URI_SCHEME ne “http”} action reset`
URI_HOST	Within URI; client specifies the host URL of the Web server. Type: string **Example:** `{URI_HOST eq “www.e-commerce.com”}`
URI_PORT	Within URI; client includes the TCP port that the Web sever application protocols should use. TCP Port 80 is the expected port for HTTP requests. Type: integer **Example:** `{URI_PORT != 80}`
URI_PATH	Within URI; client specifies the directory path to a resource on the Web server. Type: string **Example:** `{URI_PATH matches “/images/*”}`

Table C-2. HTTP Uniform Resource Identifier predicates (continued)

Field name	Description
URI_ALLFILE	Within URI; client specifies the complete resource (basename and suffix) to access on the Web server. Type: string **Example:** `{URI_ALLFILE eq “file1.jpg”}`
URI_BASENAME	Within URI; client specifies the basename resource to access on the Web server. The suffix is not specified. Type: string **Example:** `{URI_BASENAME matches “file1”}`
URI_SUFFIX	Within URI; client specifies the resource suffix or file extension. Type: string **Example:** `{URI_SUFFIX matches “jpg”}`
URI_QUERY	Within URI: client specifies or requests additional information from the server. Type: string **Example:** `{URI_QUERY eq “instructions”}`
CLIENT_ADDRESS	N2000 Series initiates the configured action (forward, redirect, or reset) based on the specified client IP address. Type: IP address in dotted-decimal notation (no quotes allowed) and network mask in CIDR or dotted-decimal notation. **Example:** `{(URI_PATH matches “/images/*”) and (CLIENT_ADDRESS matches 192.168.124.6/24)}`

Using the predicate operators

Table C-3 lists and describes the operators associated with object rule predicate statements. Within a predicate statement, operators determine how text strings and integers perform with the action specified in a request policy.

Table C-3. Object rule predicate operators

Operator	Purpose	Example
{ } braces	Encloses a predicate statement created in the CLI. (Not used in the Web interface).	**{**URI_QUERY matches “information*”**}**
“ ” quotes	Encloses text strings.	{URI_SUFFIX matches **“**jpg**”**}
* asterisk wildcard	Operates with the matches predicate operators only. When used with other operators, the asterisk is taken literally and is evaluated like any other character.	See the matches operator in this table.
? question mark	Operates with the matches predicate operator only. When used with other operators, the asterisk is taken literally and is evaluated like any other character.	See the matches operator in this table.
eq	Equal to (string)	{HTTP_VERSION **eq** “HTTP/2.0”}
==	Equal to (integer)	{URI_PORT **==** 80}
ne	Not equal to (string)	{URI_SCHEME **ne** “http”}
!=	Not equal to (integer)	{URI_PORT **!=** 80}
lt	Less than (string)	{ACCEPT **lt** “200”}
<	Less than (integer)	{CONTENT-LENGTH < 40000}
gt	Greater than (string)	{ACCEPT **gt** “100”}
>	Greater than (integer)	{CONTENT-LENGTH **>** 40000}

Table C-3. Object rule predicate operators (continued)

Operator	Purpose	Example
le	Less than or equal to (string)	{ACCEPT **le** “350”}
<=	Less than or equal to (integer)	{CONTENT-LENGTH **<=** 40000}
ge	Greater than or equal to (string)	{ACCEPT **ge** “350”}
>=	Greater than or equal to (integer)	{CONTENT-LENGTH **>=** 40000}
( ) grouping in parentheses	Encloses a predicate statement when multiple operators (such as “and”, “or”) are used within an object rule.	{**(**CONTENT-LENGTH > 500**)** or **(**CONTENT-LENGTH == 500**)**}
not	**not** operator	{not METHOD is GET}
!	See != in this table	
and	**and** operator	{(METHOD is GET) **and** (URI matches “http://www.e-commerce.com:80/images/*”)}
&&	Same as **and**	{(METHOD is GET) **&&** (URI matches “http://www.e-commerce.com:80/images/*”)}
or	**or** operator	{(METHOD is GET) **or** (METHOD is HEAD)}
||	Same as **or**	{(METHOD is GET) **||** (METHOD is HEAD)}
and or	Combination of AND and OR in a single predicate statement.	{(METHOD is GET) **or** (METHOD is HEAD) **and** (URI_PATH matches “/images/*”)}

Table C-3. Object rule predicate operators (continued)

Operator	Purpose	Example
`matches`	String matching This operator supports the * and ? characters for wildcard string matching. The asterisk (*) matches all or no characters; the ? matches a single character. Examples: *M* - matches all or no characters before and after the letter M. M? - matches a single character following the letter M. When used with other operators, the asterisk is taken literally and is evaluated like any other character.	`{USER_AGENT` **`matches`** `"*Mozilla/ 4.0*"}`
`contains`	KeywordSet matching	`{TRANSFER_ENCODING` **`contains`** `chunked}`
`is`	Keyword matching	`{METHOD` **`is`** `POST}`
`has`	String matching (used with `COOKIE` field name only)	`{COOKIE` **`has`** `"value" > 300}`
`present`	HTTP header exists.	`{ACCEPT_ESI` **`present`**`}`
`absent`	HTTP header does not exist.	`{ACCEPT_ESI` **`absent`**`} action reset`

Using predicate keywords

Table C-4 lists and describes the keywords associated with the `METHOD` object rule predicate statement

Table C-4. Object rule predicate keywords

Keyword	Used with; Description	Example
`GET`	`METHOD`; The client requests a specific resource from the server. Sample request: `GET http://www.e-commerce.com/images/file1.jpg`	`{METHOD is GET}`
`HEAD`	`METHOD`; The client requests that the server not include the resource in the response. Sample request: `HEAD http://www.e-commerce.com/images/file1.jpg`	`{METHOD is HEAD}`
`OPTIONS`	`METHOD`; The client requests the server to provide the options it supports for the indicated response. Sample request: `OPTIONS http://www.e-commerce.com/images/file1.jpg`	`{METHOD is OPTIONS}`
`POST`	`METHOD`; The client requests the server to pass the message body to the indicated resource. Sample request: `POST http://www.e-commerce.com/cgi-bin/file.cgi HTTP/2.0`	`{METHOD is POST}`

Table C-4. Object rule predicate keywords (continued)

Keyword	Used with; Description	Example
PUT	METHOD; The client requests the server to accept the message body as the resource. Sample request: `PUT http://www.e-commerce.com/images/file2.jpg`	{METHOD is **PUT**}
DELETE	METHOD; The client requests the server to delete the indicated resource. Sample request: `DELETE http://www.e-commerce.com/images/file1.jpg`	{METHOD is **DELETE**}
TRACE	METHOD; The client requests the server to acknowledge the request only. Sample request: `TRACE http://www.e-commerce.com`	{METHOD is **TRACE**}
CONNECT	METHOD; The client requests the server to establish a tunnel. Sample request: `CONNECT http://www.e-commerce.com/home.htm`	{METHOD is **CONNECT**}
PROPFIND	METHOD; The client requests the server to search for object properties.	{METHOD is **PROPFIND**}
PROPPATCH	METHOD; The client requests the server to update the object properties.	{METHOD is **PROPPATCH**}
MKCOL	METHOD; The client requests the server to update the object properties.	{METHOD is **MKCOL**}
COPY	METHOD; The client requests the server to copy a file or object to a specified location.	{METHOD is **COPY**}
MOVE	METHOD; The client requests the server to move a file or object to a specified location.	{METHOD is **MOVE** }

Table C-4. Object rule predicate keywords (continued)

Keyword	Used with; Description	Example
`LOCK`	`METHOD`; The client requests the server to lock a file or object.	`{METHOD is **LOCK**}`
`UNLOCK`	`METHOD`; The client requests the server to unlock a file or object.	`{METHOD is **UNLOCK**}`

Using predicate keywordSets

Table C-5 lists and describes the keywordSets associated with the specific object rule predicate statements `CONNECTION` and `TRANSFER-ENCODING`.

Table C-5. Object rule predicate keywordSets

Keyword	Used with; Description	Example
`keep-alive`	`CONNECTION`; The client is informed that the server will keep a persistent connection with the client after sending a response.	`{CONNECTION contains **keep-alive**}`
`close`	`CONNECTION`; The client is informed that the server will close the connection after sending a response.	`{CONNECTION contains **close**}`
`chunked`	`TRANSFER-ENCODING`; Chunked encoding breaks up the message body into chunks to improve Web server performance. The server begins sending the response as soon as it begins composing the response. The last chunk has a size of 0 bytes.	`{TRANSFER_ENCODING contains **chunked**}`
`gzip`	`TRANSFER-ENCODING`; The `gzip` keyword compresses the message body and reduces transmission time.	`{TRANSFER_ENCODING contains **gzip**}`

Specifying request policy actions

A request policy requires one of the following actions and associates an object rule with the named action:

- Forward
- Sorry

Forward

The forward action passes the HTTP request to a service group if the traffic matches the object rule.

Table C-6 lists and describes the `firstObjectSwitching` argument, a request policy option that you can use to refine how the traffic is forwarded. For more information about other request policy options, refer to the `requestPolicy` command in Chapter 29, "Load-balancing commands."

Table C-6. Forwarding option setting

Forwarding option	Description
`firstObjectSwitching`	Sets the method of load-balance processing of client requests in a single TCP session. When disabled, the system makes a load-balancing decision on each client request in a persistent HTTP session. If the request results in a different service group assignment, the system initiates a new TCP session. When enabled, all requests in a single TCP session are sent to the same real service. This lessens the granularity of the load-balancing function, but can speed processing by simplifying load-balancing decisions. The default setting is `disabled`. **Note:** Enabling the `firstObjectSwitching` argument stops the following operations: • Switch cookie insertion • Custom HTTP header rewriting • SSL cipher additions in the HTTP header • All response policy processing • All request transform processing • All response transform processing

Sorry

The `sorryData` command defines the action the load balancer should take when a sorry action is specified for a request or response policy. The command specifies the sorry action and the associated redirect string if the sorry action is `redirect` or the HTML page if the sorry action is `page`.

For more information about the `sorryData` command, see Chapter 29, "Load-balancing commands."

Table C-7 lists and describes the sorry actions that can be configured with the `sorryData` command.

Table C-7. Sorry action settings

Sorry action	Description
`action`	Specifies the action to take when the system has exceeded the number of retries allowed for connection to a different real service within a service group. Possible actions are: • `page`: Returns an HTML page to the client. You specify the page returned with the `actionString` argument. • `close`: Gracefully ends the TCP connection to the client. It sends an HTTP 500 Internal Error status code and closes the connection using a four-way handshake and FIN instead of a reset. • `redirect`: Returns an HTTP 302 redirect response to the client, redirecting the request to a different URI. The target of the redirection is set with the `actionString` argument. • `reset`: Sends a TCP reset to the client, ending the connection. The default action is `close`.
`actionString`	Specifies information to return to the client, to complete a `redirect` or `page` action. If `action` is `page`, enter an HTML fully qualified path name. If `action` is `redirect`, enter a valid URI.

Appendix D. About authentication and authorization services

About this appendix

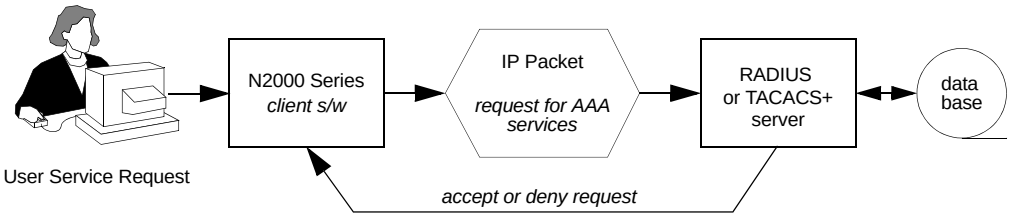
This appendix contains information about authentication and authorization services and describes the tasks involved in making the N2000 Series work properly in a RADIUS or TACACS+ environment.

Topics

This appendix includes the following topics and tables:

About authentication and authorization services

In addition to supporting authentication, authorization, and accounting (AAA) services through its own internal tables, the N2000 Series supports two of the most widely used AAA services: TACACS+ and RADIUS. With both these services, the N2000 Series functions as a client, sending authentication requests to one or more RADIUS or TACACS+ servers. In a broad view, the process works as shown in the figure below.



There are two separate but interrelated tasks involved in making the N2000 Series work properly in a RADIUS or TACACS+ environment.

1. Setting up the RADIUS or TACACS+ client side software on the N2000 Series
2. Setting up the RADIUS or TACACS+ server so that it “knows” how to interpret any vendor-specific data received in the service request packets. If you want to rely on the servers for simple authentication only, it is possible that the RADIUS or TACACS+ server databases and configuration files require no changes.

As an N2000 Series administrator, you need to work closely with the network administrator responsible for the RADIUS or TACACS+ servers in your environment. In this way, you can ensure that the client-side software is properly configured. Client-side configuration is covered in the *Sun N2000 Series Release 2.0 – System Administration Guide*.

The RADIUS or TACACS+ network administrator needs to understand the vendor-specific data that the N2000 Series is capable of sending so as to ensure that the RADIUS or TACACS+ servers properly handle any authentication requests originating from the N2000 Series. See Chapter 13, “User administration commands” for information about specifying authentication and authorization attributes for the N2000 Series.

You can view a complete list of AAA attributes used by the system by typing the following commands from the N2000 Series CLI.

```
sun> enable
sun# switchServices
sun(switchServices)# userAdministration
sun(switchServices userAdministration)# show advanced attributes
```

See Chapter 13, "User administration commands" for a sample display of the types of authentication and authorization attributes that the N2000 Series exchanges with an AAA server. The vendor-specific attributes appear near the end of the list, starting with ID 301.

Configuring the AAA servers for TACACS+

As the N2000 Series administrator, you should consult with the network administrator responsible for the TACACS+ servers to ensure that the client software uses the correct "secret" for encryption and decryption.

The network administrator may need to edit the configuration files on the TACACS+ server to accommodate the N2000 Series specific extensions. Be aware that any authentication or authorization attributes specified in the server configuration files override the local authentication and authorization attributes provided on the N2000 Series system.

A sample section of a customized TACACS+ configuration file follows.

```
user = sysopOne {
           global = cleartext 2obvious

[... text omitted ...]

service = telnetLogin {
 vswitchName = system
 profileName = systemOperator
}
```

Note: The exact syntax that the TACACS+ server expects may vary slightly from this example.

The sample section of the TACACS+ configuration file has been customized to authenticate a user named `sysopOne` and to authorize her to log in to the system vSwitch using Telnet. User `sysopOne` is granted read-only access by virtue of being assigned the `systemOperator` profile.

Configuring the AAA servers for RADIUS

As the N2000 Series administrator, you should consult with the network administrator responsible for the RADIUS servers to ensure that the client software uses the correct "secret" for encryption and decryption.

The N2000 Series can transmit vendor-specific information either by using Code 26 as specified in RFC 2865, or by using a vendor ID offset to provide compatibility with servers running older RADIUS software that does not support RFC-specified vendor encoding. The vendor ID offset is used to handle vendor-specific attributes within the standard RADIUS attribute numbering space.

You should consult with the network administrator to determine whether your network requires the use of a vendor ID offset to ensure that the same offset is configured on both the client and server sides.

The vendor-specific information that the N2000 Series transmits takes the following form.

Code	Length	Data			
		Vendor ID	Vendor Type (subtype)	Vendor Length	Vendor Value
26	*# bytes*	8857	1	*# bytes*	`telnetLogin`

The Vendor ID is 8857. The Vendor Type (also called *subtype*) codes used by the N2000 Series are as shown in the following table.

Subtype	Definition	Allowed values
1	service—The type of application or service being requested.	• `consoleLogin` • `httpLogin` • `httpsLogin` • `sshSession` • `sshdLogin` • `telnetLogin`
2	vSwitchName—The ID associated with the user for access.	1–64 characters matching specifically configured vSwitch users
3	profileName—The name associated with the vSwitch user for CLI or GUI access	1–32 characters identifying an existing profile, typically: • `systemAdmin` • `systemOperator` • `vSwitchAdmin` • `vSwitchOperator`
4	SSH privileges—The numerical value for the secure shell privileges associated with a user for the sshdSession service.	• 1 (none) • 2 (session) • 3 (sftpRead) • 4 (sftpReadWrite)

Configuration files on the RADIUS server

The network administrator may need to edit the dictionary file on the RADIUS server to accommodate the N2000 Series specific extensions. A sample section of a dictionary file follows.

```
## Sun-Nauticus Specific extensions

VENDOR   Sun-Nauticus    8857

ATTRIBUTE    service         1    string     Sun-Nauticus
ATTRIBUTE    vswitchName     2    string     Sun-Nauticus
ATTRIBUTE    profileName     3    string     Sun-Nauticus
ATTRIBUTE    profileName     4    integer    Sun-Nauticus

VALUE  sshdPrivs    none           1
VALUE  sshdPrivs    session        2
VALUE  sshdPrivs    sftpRead       3
VALUE  sshdPrivs    sftpReadWrite  4
```

In addition, depending on how you want authorization to be handled, the network administrator may need to edit the user file on the RADIUS server. Be aware that any authentication or authorization attributes specified in these files override the local authentication and authorization attributes provided on the N2000 Series system itself.

A sample section of a RADIUS user file appears below.

```
adminOne  Auth-Type := Local, User-Password = "admin123"
      service = httpsLogin,
      service = consoleLogin,
      vswitchName = system,
      profileName = systemAdmin

sysopOne  Auth-Type := Local, User-Password = "op123"
      service = telnetLogin,
      vswitchName = system,
      profileName = systemOperator
```

Note: The exact syntax the RADIUS expects may vary slightly from this example.

The sample section of the user file has been customized to authenticate two users, `adminOne` and `sysopOne`. User `adminOne` is authorized to log in only via the secure methods of HTTPS and console. This prevents her password from being compromised. User `sysopOne` is authorized to log in via the less-secure Telnet, but because his profile is defined as `systemOperator`, he is restricted to read-only access.

Command Index