



Sun™ N2000 Series Release 2.0— System Configuration Guide

Sun Microsystems, Inc.
www.sun.com

Part No. 817-7637-12
November 2004, Revision A

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

Copyright 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, AnswerBook2, docs.sun.com, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and in other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a.

Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, AnswerBook2, docs.sun.com, et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciées de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



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.

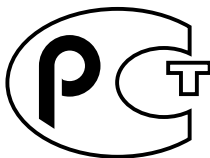
以下声明适用于运往中国且其认证标志上注有 "Class A" 字样的产品。

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

Preface

About this manual.....	xix
What is in this manual?	xx
Related documentation	xxi
Conventions	xxii
Typographical conventions.....	xxii
CLI commands	xxii
Data formats	xxiii
IP addresses.....	xxiii
Subnet masks and wildcard masks	xxiii
MAC addresses	xxiii
Text strings	xxiii
Port numbers	xxiii
Hexadecimal numbers.....	xxiii
Notes, cautions, warnings	xxiv
Accessing Sun documentation	xxiv
Third-party Web sites	xxiv
Contacting Sun technical support.....	xxv
Sun welcomes your comments	xxv
Abbreviations and acronyms	xxv

Chapter 1. N2000 Series overview

Introduction	1-1
Topics	1-1
N2000 Series overview	1-2
Figure 1-1. Sun N2040 and N2120 hardware platforms.....	1-2
Figure 1-2. N2000 Series basic Internet network	1-3
N2000 Series functionality	1-3
Application switching	1-4

Figure 1-3. N2000 Series application-switched network.....	1-5
Virtual switching and routing using vSwitches	1-5
System vSwitch	1-6
Operator-defined vSwitches	1-7
Figure 1-4. N2000 Series virtual network	1-7
Load balancing	1-8
Layer 4 Server Load Balancing (L4SLB)	1-8
Layer 4 Server Load Balancing Advanced (L4SLB_ADV)	1-9
Layer 4 Server Load Balancing with Secure Sockets Layer (L4SLB_SSL)	1-9
Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS) object switching	1-9
Bridge Mode Load Balancing	1-9
Transparent Device Load Balancing (TDLB)	1-10
File Transfer Protocol Load Balancing (FTPLB)	1-10
Real Time Streaming Protocol Load Balancing (RTSPLB)	1-10
Simple Mail Transfer Protocol Load Balancing	1-10
Internet Message Access Protocol Load Balancing	1-10
Figure 1-5. N2000 Series load balancing across Web server groups ..	1-11
Client address translation	1-11
Figure 1-6. N2000 Series CAT network.....	1-12
Static and dynamic network address translation	1-13
Figure 1-7. Static NAT and dynamic NAT network.....	1-14
SSL acceleration and regeneration	1-15
Figure 1-8. N2000 Series as an SSL client and an SSL server.....	1-16
Server health checks	1-17
Figure 1-9. N2000 Series server health checks.....	1-17
Access control lists	1-18
N2000 Series redundancy	1-18
N2000 Series management tools	1-18
Figure 1-10. N2000 Series management tools.....	1-19
Command-line interface	1-19
Web interface	1-20

SNMP	1-20
Before you start the N2000 Series configuration	1-21
N2000 Series configuration steps	1-21
Standard configuration steps	1-21
Optional configuration steps	1-22

Chapter 2. Using the system vSwitch

Introduction	2-1
Topics	2-1
System vSwitch overview	2-2
Figure 2-1. N2000 Series system vSwitch	2-2
How to Display system vSwitch information	2-2
Configuring the management vRouter	2-3
Figure 2-2. Management vRouter network.....	2-4
How to assign an IP address to the management vRouter	2-4
How to configure an Ethernet port as the management interface	2-5
How to set up a default route to the local gateway	2-5
Configuring shared vRouters	2-6
Figure 2-3. System vSwitch using one shared vRouter	2-6
Figure 2-4. System vSwitch with two shared vRouters	2-7
How to configure multiple shared vRouters	2-7

Chapter 3. Creating vSwitches

Introduction	3-1
Topics	3-1
Operator-defined vSwitch overview	3-2
Figure 3-1. N2000 Series network with operator-defined vSwitches.....	3-3
Default vRouters	3-3
Load-balancer application	3-4
Network address translation	3-4
Static NAT	3-4
Dynamic NAT	3-5
Creating the first operator-defined vSwitch	3-5

Figure 3-2. Network with one operator-defined vSwitch	3-5
How to create an operator-defined vSwitch	3-6
Allocating port and service bandwidth to a vSwitch	3-6

Chapter 4. Configuring the L2/L3 network

Introduction	4-1
Topics	4-1
L2/L3 Network Configuration Overview	4-2
Configuring IP interfaces	4-3
Figure 4-1. N2000 Series Ethernet, LAG, and VLAN interfaces	4-3
How to enter the vRouter context	4-4
How to configure IP over Ethernet interfaces	4-5
Figure 4-2. Default vRouter interfaces and IP addresses	4-6
How to configure IP over VLAN over Ethernet interfaces	4-6
Figure 4-3. VLAN configuration	4-7
How to configure IP over LAG over Ethernet interfaces	4-8
Figure 4-4. LAG configuration.....	4-9
How to configure IP over VLAN over LAG interfaces	4-10
Figure 4-5. IP/VLAN/LAG over Ethernet network	4-10
Spanning Tree Protocol overview	4-11
Terminology and concepts	4-11
Bridge Protocol Data Units	4-11
Root bridge (designated root)	4-11
Root port	4-12
Designated port	4-12
Path costs	4-12
Table 4-1. 802.1d standard path costs	4-12
Configuring the Spanning Tree Protocol	4-13
How to configure the Spanning Tree bridge	4-13
Figure 4-6. Spanning Tree Protocol network	4-13
How to configure Spanning Tree ports	4-14
Configuring IP static routes	4-15

How to configure an IP static route	4-15
Figure 4-7. Network with a static route	4-16
Configuring the Routing Information Protocol	4-16
How to configure RIP interfaces	4-17
Figure 4-8. Network with RIP interfaces.....	4-17
Configuring Open Shortest Path First	4-18
How to display OSPF command options	4-18
Configuring the OSPF areas and interfaces	4-19
Stub and non-stub areas	4-19
Not-so-stubby areas	4-19
Figure 4-9. OSPF network.....	4-20
How to create OSPF areas and interfaces	4-20
Configure the OSPF backbone area and interfaces	4-21
Configure the OSPF non-backbone area and interfaces	4-21
Display the OSPF area settings	4-21

Chapter 5. Load balancing L4 network traffic

Introduction	5-1
Topics	5-2
L4 load-balancing overview	5-2
L4SLB	5-2
L4SLB_ADV	5-3
L4SLB_SSL	5-3
Terminology and concepts	5-4
Host	5-4
Real service	5-4
Service group	5-4
Virtual service	5-4
L4 load-balancing configuration steps	5-4
L4 load-balancing configuration hierarchy	5-5
Figure 5-1. N2000 Series L4 load-balancing configuration hierarchy	5-6
Sample network	5-7

Figure 5-2. E-commerce network	5-7
How to add a host to the operator-defined vSwitches	5-7
Figure 5-3. Configuration with backend hosts	5-8
How to create a real service	5-9
Figure 5-4. Real service definitions on backend hosts	5-12
How to create service groups	5-12
Load-balancing algorithms	5-13
Weighted hash	5-14
Weighted random	5-14
Round-robin	5-14
Least connections	5-14
Load-balancing metrics	5-14
Figure 5-5. L4SLB weighted hash configuration across three servers .	5-16
How to create the virtual service for L4 load balancing	5-16
How to create virtual service groups	5-18

Chapter 6. Load balancing L5 to L7 network traffic

Introduction	6-1
Topics	6-1
L5 to L7 load-balancing overview	6-2
Terminology and concepts	6-3
Host	6-3
Object	6-3
Object rule	6-3
Policy	6-3
Predicate	6-4
Real service	6-4
Response policy	6-4
Response transform	6-4
Request policy	6-5
Request transform	6-5
Service group	6-5

Sorry data	6-5
Virtual service	6-6
L5 to L7 load-balancing configuration steps	6-6
L7 load-balancing configuration hierarchy	6-8
Figure 6-1. L5 to L7 load-balancing configuration hierarchy	6-9
Sample network	6-10
Figure 6-2. E-commerce network	6-10
How to create an object rule	6-10
How to create a request policy	6-13
How to create a proxy IP pool	6-15
How to create a request transform	6-15
How to create a response policy	6-17
How to create a response transform	6-19
How to define a sorry data	6-20
How to create the virtual service for L5 to L7 load balancing	6-21
How to create virtual service groups	6-22
Object rule details — predicates	6-23
HTTP Request and HTTP Response header field names	6-23
Table 6-1. HTTP Request and Response header predicates - HTTP fields	6-23
Predicates with URI field names	6-28
Figure 6-3. Uniform Resource Identifier structure	6-28
Table 6-2. HTTP Uniform Resource Identifier predicates	6-29
Using the predicate operators	6-31
Table 6-3. Object rule predicate operators.....	6-31
Using predicate keywords	6-34
Table 6-4. Object rule predicate keywords.....	6-34
Using predicate keywordSets	6-36
Table 6-5. Object rule predicate keywordSets	6-36
Forwarding actions	6-37
Forward	6-37
Table 6-6. Forwarding option setting.....	6-37
Sorry	6-38

Table 6-7. Sorry Settings	6-38
HTTP load-balancing example	6-39
Figure 6-4. Network with two e-commerce service groups	6-39
Table 6-8. Policies for the HTTP load-balancing example	6-40
Host definitions	6-41
Real service definitions	6-41
Service group definitions	6-41
Object rule definitions	6-41
sorryData definitions	6-42
Request and response policy definitions	6-43
Virtual service definitions	6-43

Chapter 7. Additional load-balancing applications

Introduction	7-1
Topics	7-1
Bridge Mode Load Balancing	7-2
Configuration guidelines	7-2
How to configure Bridge Mode Load Balancing	7-3
Figure 7-1. Bridge Mode Load Balancing configuration.....	7-3
FTP Load Balancing	7-9
How to configure FTP Load Balancing	7-9
IMAP Load Balancing	7-13
How to Configure IMAP Load Balancing	7-13
RTSP Load Balancing	7-13
How to configure RTSP Load Balancing	7-13
SMTP Load Balancing	7-15
How to configure SMTP Load Balancing	7-15
Transparent Device Load Balancing	7-15
How to configure Transparent Device Load Balancing	7-15
Figure 7-2. Configuration for Transparent Device Load Balancing.....	7-16

Chapter 8. Configuring client address translation

Introduction	8-1
Topics	8-1
N2000 Series client address translation overview	8-2
Comparing load balancing with and without client address translation	8-2
Client address and port translations	8-3
Table 8-1. Client address translation on inbound/outbound SLB traffic	8-3
Client address translation configuration steps	8-4
Figure 8-1. Sample N2000 Series CAT network	8-4
How to create the proxy IP pool	8-4
How to enable client address translation	8-5
How to display proxy IP pool statistics	8-5

Chapter 9. Configuring cookie persistence

Introduction	9-1
Topics	9-2
N2000 Series cookie persistence overview	9-2
Figure 9-1. N2000 Series network with cookie persistence transactions	9-4
Creating cookie persistence rules	9-5
cookieName	9-5
cookieDomain	9-5
cookiePath	9-6
cookieExpires	9-6
secure	9-6
Cookie persistence limitations	9-6

Chapter 10. Configuring network address translation

Introduction	10-1
Topics	10-1
N2000 Series outbound NAT overview	10-2
Figure 10-1. N2000 NAT network.....	10-2
Static NAT	10-3

Dynamic NAT	10-3
Configuring static and dynamic NAT	10-3
Figure 10-2. Static and dynamic NAT network	10-4
How to configure static NAT	10-5
How to configure dynamic NAT	10-6
How to display NAT statistics	10-7

Chapter 11. Configuring CKM and SSL acceleration

Introduction	11-1
Topics	11-1
SSL acceleration overview	11-2
Figure 11-1. N2000 Series SSL server and client	11-3
Certificate and key management for SSL	11-3
For new keys and certificates	11-4
For existing keys and certificates	11-4
CKM and SSL configuration steps	11-5
For new SSL deployments	11-5
For existing SSL deployments	11-6
Generating the SSL key and certificate request	11-6
How to create a cryptographic key pair	11-6
How to create the Certificate Signing Request	11-7
How to issue temporary certificates	11-8
How to submit the certificate request to the CA	11-8
How to install the certificate on the N2000 Series	11-8
Working with existing keys and certificates	11-9
SSL on a vSwitch configuration steps	11-10
How to configure SSL in the serviceGroup definition	11-10
How to configure SSL in the virtualService definition	11-11

Chapter 12. Configuring server health checks

Introduction	12-1
Topics	12-1
Server health checks overview	12-2

Out-of-band server health checks	12-2
Figure 12-1. N2000 Series out-of-band server health checks.....	12-3
In-line server health checks	12-3
Passive health checks	12-4
Server health check considerations	12-4
realService weight	12-4
Server memory and resources	12-4
Configuring server health checks	12-5
How to configure out-of-band server health checks	12-5
Naming the profile	12-5
Defining the health check profile	12-6
Table 12-1. Server health check probes and profile settings.....	12-7
Applying the health check profile to a serviceGroup	12-10
How to configure in-line server health checks	12-11
Configuring in-line server health checks for HTTP	12-12
How to configure passive server health checks	12-12

Chapter 13. Configuring SSL regeneration

Introduction	13-1
Topics	13-1
SSL regeneration overview	13-2
Figure 13-1. N2000 Series SSL regeneration to backend servers.....	13-2
SSL regeneration configuration steps	13-3
How to configure SSL in the serviceGroup definition	13-3

Chapter 14. Configuring access control lists and groups

Introduction	14-1
Topics	14-1
Access control lists (ACL) configuration steps	14-2
Basic access control list configuration	14-2
Figure 14-1. Network using an ACL to block an HTTP request	14-3
How to create and edit the ACL	14-3
Setting ACL rule precedence, actions, and protocols	14-4

Precedence	14-4
Actions	14-4
Protocols	14-4
Table 14-1. Well-known protocols for ACL traffic matching.....	14-5
Required and optional ACL settings	14-5
How to create the access group	14-6

Chapter 15. Configuring redundancy

Introduction	15-1
Topics	15-1
VRRP overview	15-2
Figure 15-1. N2000 Series VRRP network	15-3
VRRP configuration steps	15-4
How to configure VRRP on redundant switches	15-5
Figure 15-2. N2000 Series VRRP network configuration	15-5
VSRP overview	15-11
Figure 15-3. N2000 Series VSRP network	15-11
VSRP configuration requirements	15-12
VSRP configuration steps	15-12
How to configure VSRP peers	15-13
Figure 15-4. N2000 Series VSRP network configuration.....	15-13
Exporting configurations to redundant switches	15-17
Redundancy configuration steps	15-17
How to export and import redundant configurations	15-17

Index

Preface

About this manual

The *Sun N2000 Series Release 2.0 — System Configuration Guide* 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 installers and system administrators who are responsible for configuring, monitoring, and maintaining the Sun N2000 Series system. You should be familiar with switch and router configuration, load balancing, SSL acceleration, and virtualization.

What is in this manual?

This manual includes the following topics.

For information about:	See:
N2000 Series application switches and functionality	Chapter 1
Using the system virtual switch	Chapter 2
Creating operator-defined virtual switches	Chapter 3
Configuring the L2/L3 network	Chapter 4
Load balancing L4 network traffic	Chapter 5
Load balancing L5 to L7 network traffic	Chapter 6
Additional load-balancing applications	Chapter 7
Configuring client address translation	Chapter 8
Configuring cookie persistence	Chapter 9
Configuring network address translation	Chapter 10
Configuring CKM and SSL acceleration	Chapter 11
Configuring server health checks	Chapter 12
Configuring SSL regeneration	Chapter 13
Configuring access control lists and groups	Chapter 14
Configuring redundancy	Chapter 15

Related documentation

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

Title	Document Number	Location
<i>Sun N2000 Release 2.0 — Introduction Guide</i>	817-7641-10	Documentation CD
<i>Sun N2000 Series Release 2.0 — Quick Installation</i>	817-7640-10	Printed, in ship kit Documentation CD
<i>Sun N2000 Series Release 2.0 — Hardware Installation and Startup Guide</i>	817-7638-10	Printed, in ship kit Documentation CD
<i>Sun N2000 Series Release 2.0 — System Configuration Guide</i>	817-7637-11	Documentation CD
<i>Sun N2000 Series Release 2.0 — System Administration Guide</i>	817-7635-10	Documentation CD
<i>Sun N2000 Series Release 2.0 — Command Reference</i>	817-7636-11	Documentation CD
<i>Sun N2000 Series Release 2.0 — Release Notes</i>	817-7639-11	Printed, in ship kit

Conventions

Typographical conventions

This manual uses the following typographical conventions.

Convention	Function	Example
Ctrl+x	Indicates a control key combination.	Press Ctrl+C
[key name]	Identifies the name of a key to press.	Type <code>xyz</code> , then press [Enter]
brackets []	Indicates an optional argument.	<code>show protocol telnet sessions [ipAddress ipAddress]</code>
braces { }	Indicates a required argument with a choice of values; choose one. Encloses an object rule predicate or a list within an object rule created with the CLI.	<code>ckm import paste pairHalf {privateKey certificate}</code> <code>objectRule rule1 predicate {URI_QUERY matches "information*"}</code>
vertical bar	Separates parameter values. Means "or."	<code>format {pem der iis4 pkcs12 sun}</code>
Monospaced regular	Screen output, argument keywords, and defined argument values.	<code>protocol telnet adminState enabled</code>
Monospaced italic	Variable; generic text for which you supply a value.	<code>ntp id number</code>
Monospaced bold	User input.	<code>sun> show vSwitch</code>

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, uppercase and lowercase. 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

This manual uses the following formats to highlight notes, cautions, and 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 — System Configuration Guide, 817-7637-10

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	Certificate 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 Sockets Layer
LAG	link aggregation group
LAN	local area network
LB	load balancer application on the N2000 Series
MD5	Message Digest 5
MIB	management information base
N2000 Series	Sun N2000 Series application switch
N2040	N2000 Series model that provides 40 10/100-Mbps ports and 4 SFF pluggable Gigabit Ethernet ports
N2120	N2000 Series model that provides 12 SFF pluggable Gigabit Ethernet ports
NAT	network address translation
NMON	network monitor
NTP	Network Time Protocol
OID	object identifier
OSPF	Open Shortest Path First
PEM	Privacy Enhanced Mail format
PKCS12	Public Key Cryptography Standard #12 format
QoS	Quality of Service
RIP	Routing Information Protocol
SFF	small form factor

SFTP	Secure Shell File Transfer Protocol
SLB	server load balancing
SNMP	Simple Network Management Protocol
SSH	Secure Shell
SSL	Secure Sockets 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
VPN	virtual private network
vRouter	virtual router on the N2000 Series
VRRP	Virtual Router Redundancy Protocol
VSRP	Virtual Service Redundancy Protocol
vSwitch	virtual switch on the N2000 Series

Chapter 1. N2000 Series overview

Introduction

This chapter provides a high-level overview of the Sun™ N2000 Series application switch, as well as the terminology and information with which you should be familiar before planning and creating an N2000 Series configuration.

Topics

This chapter includes the following topics:

Topic	Page
N2000 Series overview	1-2
N2000 Series functionality	1-3
Application switching	1-4
Virtual switching and routing using vSwitches	1-5
Load balancing	1-8
Client address translation	1-11
Static and dynamic network address translation	1-13
SSL acceleration and regeneration	1-15
Server health checks	1-17
Access control lists	1-18
N2000 Series redundancy	1-18
N2000 Series management tools	1-18
Before you start the N2000 Series configuration	1-21
N2000 Series configuration steps	1-21

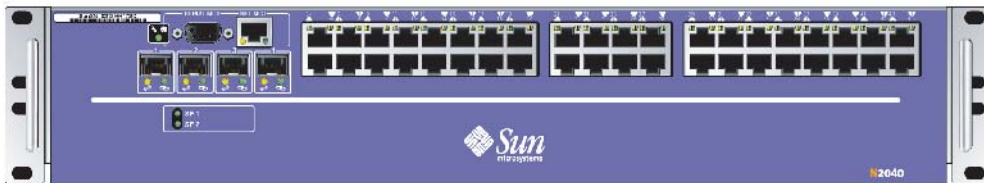
N2000 Series overview

The Sun N2000 Series is a gigabit-scaled hardware platform that provides application switching, L4 to L7 load balancing, and high-speed Transmission Control Protocol (TCP) termination and Secure Sockets Layer (SSL) acceleration for network data centers. The N2000 Series implements virtual switching technology that allows enterprises and Internet service providers (ISPs) to create multiple switches that are partitioned from one another within a single N2000 Series platform. Each virtual switch is a logical N2000 Series device with its own switching and routing capabilities, traffic load balancer, and SSL accelerator.

Figure 1-1 illustrates the Sun N2040 and the Sun N2120 platforms. The N2040 application switch provides 40 10/100-Mbps Ethernet ports with four small form factor pluggable (SFP) Gigabit Ethernet (GE) ports, and the N2120 application switch provides 12 SFP GE ports. Both platforms have a 10/100-Mbps Ethernet management port and an RS-232 local console port for system management.

Figure 1-1. Sun N2040 and N2120 hardware platforms

N2040



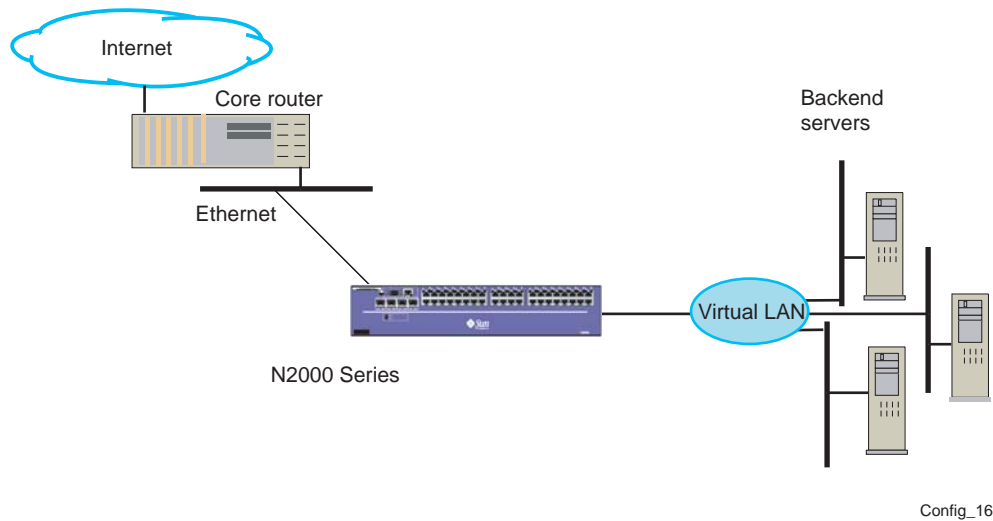
N2120



Config_15

Figure 1-2 illustrates the N2000 Series application switch in a basic network.

Figure 1-2. N2000 Series basic Internet network



For detailed information on installing the N2000 Series hardware in your network, refer to the *Sun N2000 Series Release 2.0 – Hardware Installation and Startup Guide*.

N2000 Series functionality

This section provides an overview of the N2000 Series functionality, including:

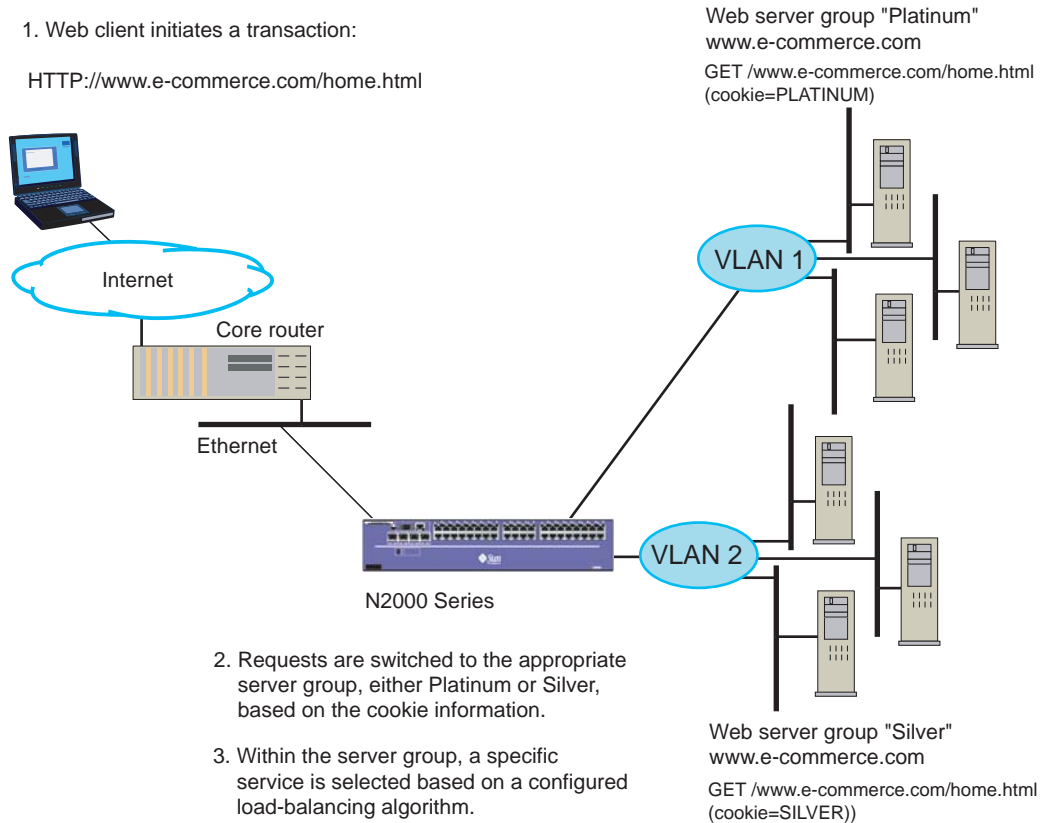
- Application switching
- Virtual switching and routing using vSwitches
- Load balancing with and without client address translation
- Static and dynamic network address translation
- SSL acceleration and regeneration
- Server health checks
- Access control lists
- N2000 Series redundancy

Application switching

As an application switch, the N2000 Series inspects client requests and makes forwarding decisions based on embedded content (terminated TCP) or the TCP packet header (non-terminated TCP). The switch applies one or more object rules and policies (such as levels of service, HTTP headers, cookies, etc.) and a load-balancing algorithm before forwarding the packets to their Web server destinations.

Figure 1-3 illustrates a sample N2000 Series network using two Web server groups. In this example, the N2000 Series switches traffic between the two server groups using the cookie information (PLATINUM and SILVER) passed in the HTTP header.

Figure 1-3. N2000 Series application-switched network



Config_17

Virtual switching and routing using vSwitches

The N2000 Series uses virtualization to partition the N2000 Series platform into multiple logical domains called *vSwitches*. Creating multiple *vSwitches* allows you to partition the data center among multiple customers and organizations based on the network services and the applications they are running.

The N2000 Series supports two types of *vSwitches*: the *system vSwitch* and the *operator-defined vSwitches*.

System vSwitch

Each N2000 Series comes preconfigured with the vSwitch called *system*. The system vSwitch provides the interface to Internet routers using one or more physical N2000 Series Ethernet ports and a virtual router called *shared*. The shared vRouter supports the IP routing protocols running on the switch, and connects to the operator-defined vSwitches. All physical Internet connections occur in the shared vRouter, which isolates vRouter routing tables and Ethernet ports from other operator-defined vSwitches

For system management, the system vSwitch also comes equipped with an independent virtual router called *management*. The management vRouter uses a configured physical N2000 Series Ethernet port for dedicated local or remote system management traffic where it isolates management traffic from data traffic on the system. This keeps all other Ethernet ports available for data connections to the backend servers.

As a separate vRouter, the management vRouter runs the management protocols and the SNMP agent for local and remote configuration and monitoring using the command-line interface (CLI), the Sun Application Switch Manager Web interface, or third-party SNMP application. It supports SNMP, TFTP, Telnet, SSH, HTTP, syslogger, trapd, and NTP.

The N2000 Series system vSwitch supports up to ten vRouters, including the shared vRouter and the management vRouter.

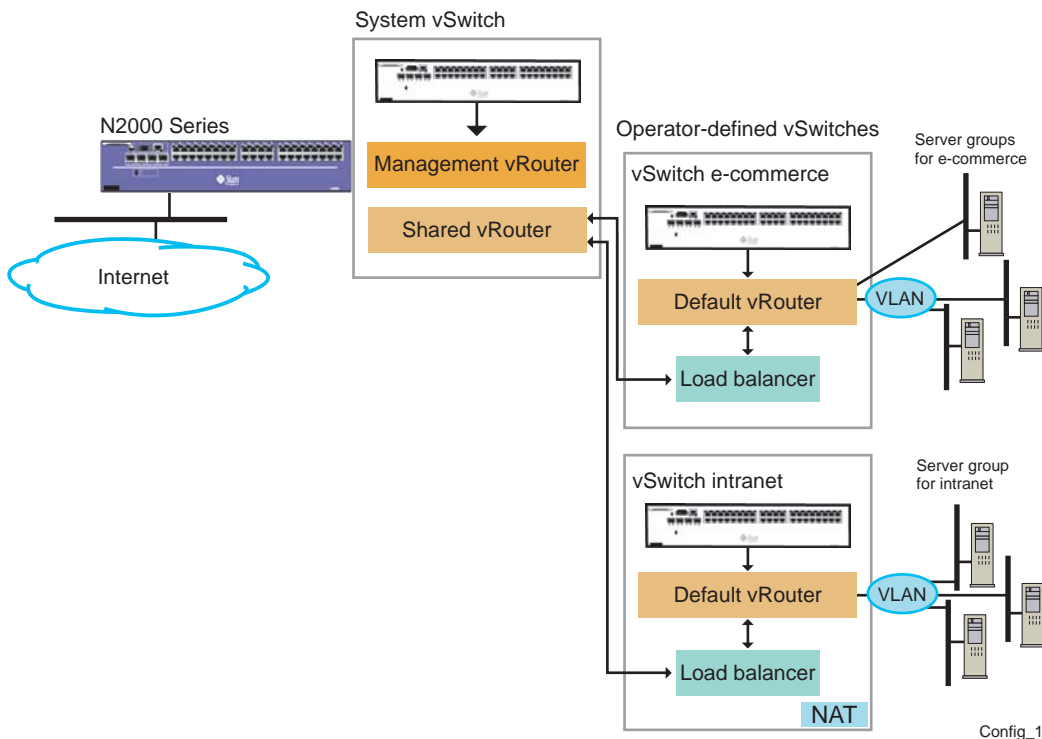
Operator-defined vSwitches

An *operator-defined vSwitch* is an independent and uniquely named logical N2000 Series system supporting L2/L3 switching and IP routing, L4 to L7 load balancing, TCP traffic termination, and SSL acceleration.

When you create an operator-defined vSwitch, the system creates a single virtual router called *default*. The default vRouter switches load-balanced traffic between the backend Web servers, the shared vRouter on the system vSwitch, and the Internet clients who are requesting and accessing resources on the Web servers.

Figure 1-4 illustrates a sample N2000 Series network showing the virtual components described in this section.

Figure 1-4. N2000 Series virtual network



Load balancing

The N2000 Series uses traditional L4 load balancing and L5 to L7 object switching to distribute traffic across server groups. Load balancing improves network performance by distributing traffic so that individual servers are not overwhelmed with network traffic. These servers appear as a single device to the network clients. Load-balancing algorithms include round-robin, weighted round-robin, weighted random selection, and weighted hash selection.

The L4 to L7 load balancer application defines the relationship between virtual services and real services. Each load balancer is assigned one or more virtual IP addresses, called a VIP, which is the address of the virtual service known to external networks. When the VIP receives a client request (such as an HTTP request), the load balancer selects an appropriate server to fulfill the request using an algorithm such as round robin then rewrites the destination IP address to that of the selected Web server. When the Web server responds to the request, the N2000 Series rewrites the source IP address to that of the VIP before forwarding the traffic back to the client. Rules can be configured to provide more control over the forwarding of requests based on HTTP headers, cookies and URIs.

The N2000 Series incorporates a load balancer on each operator-defined vSwitch. See the section, “Virtual switching and routing using vSwitches” (page 1-5) in this chapter for more information.

The N2000 Series supports the following load-balancing application types.

Layer 4 Server Load Balancing (L4SLB)

L4SLB provides Layer 4 TCP traffic balancing based on IP source address, and TCP/UDP source port using a weighted-hash algorithm. L4SLB supports the Transmission Control Protocol (TCP)/User Datagram Protocol (UDP), Domain Name Server (DNS), and Remote Authentication Dial-In User Service (RADIUS) traffic.

Refer to Chapter 5, “Load balancing L4 network traffic” for more information.

Layer 4 Server Load Balancing Advanced (L4SLB_ADV)

L4SLB_ADV uses the Server Load Balancing with SSL Function Card (Fx-SSL) and provides the following advanced functionality:

- High-speed inbound TCP termination on the system hardware and regeneration into new outbound sessions to upper layer protocols
- Load-balancing algorithms: round-robin, weighted round-robin, and weighted random selection

Refer to Chapter 5, “Load balancing L4 network traffic” for more information.

Layer 4 Server Load Balancing with Secure Sockets Layer (L4SLB_SSL)

L4SLB_SSL uses the Server Load Balancing with SSL Function Card (Fx-SSL) and provides L4SLB_ADV functionality with SSL, allowing clients to communicate over encrypted sessions.

Refer to Chapter 11, “Configuring CKM and SSL acceleration” for more information.

Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS) object switching

HTTP and HTTPS use the Server Load Balancing with SSL Function Card (Fx-SSL). L5 to L7 HTTP and HTTPS policy-based matching uses HTTP headers, cookies, URLs, or other headers over inbound and outbound sessions.

Refer to Chapter 6, “Load balancing L5 to L7 network traffic” for more information.

Bridge Mode Load Balancing

Bridge Mode Load Balancing allows Load Balancing of traffic between clients and servers on the same IP-subnet. Bridge Mode load balancing is supported on all application service types and is configured on a real service basis.

Refer to Chapter 7, “Additional load-balancing applications” for more information.

Transparent Device Load Balancing (TDLB)

Transparent Device Load Balancing allows load balancing of transparent devices, such as transparent caches.

Refer to Chapter 7, “Additional load-balancing applications” for more information.

File Transfer Protocol Load Balancing (FTPLB)

FTP Load Balancing supports load balancing in both the active and passive FTP roles, as well as client address translation and outbound FTP load balancing. FTP Load Balancing is configured as a Layer 4 server load balancing application.

Refer to Chapter 7, “Additional load-balancing applications” for more information.

Real Time Streaming Protocol Load Balancing (RTSPLB)

RTSP Load Balancing provides support for load balancing of streaming media content (video stream, audio stream, text) stored on servers. RTSP is configured as a Layer 7 load balancing application.

Refer to Chapter 7, “Additional load-balancing applications” for more information.

Simple Mail Transfer Protocol Load Balancing

SMTP Load Balancing provides support for load balancing of SMTP traffic. SMTP Load Balancing is configured as a Layer 4 server load balancing application.

Refer to Chapter 7, “Additional load-balancing applications” for more information.

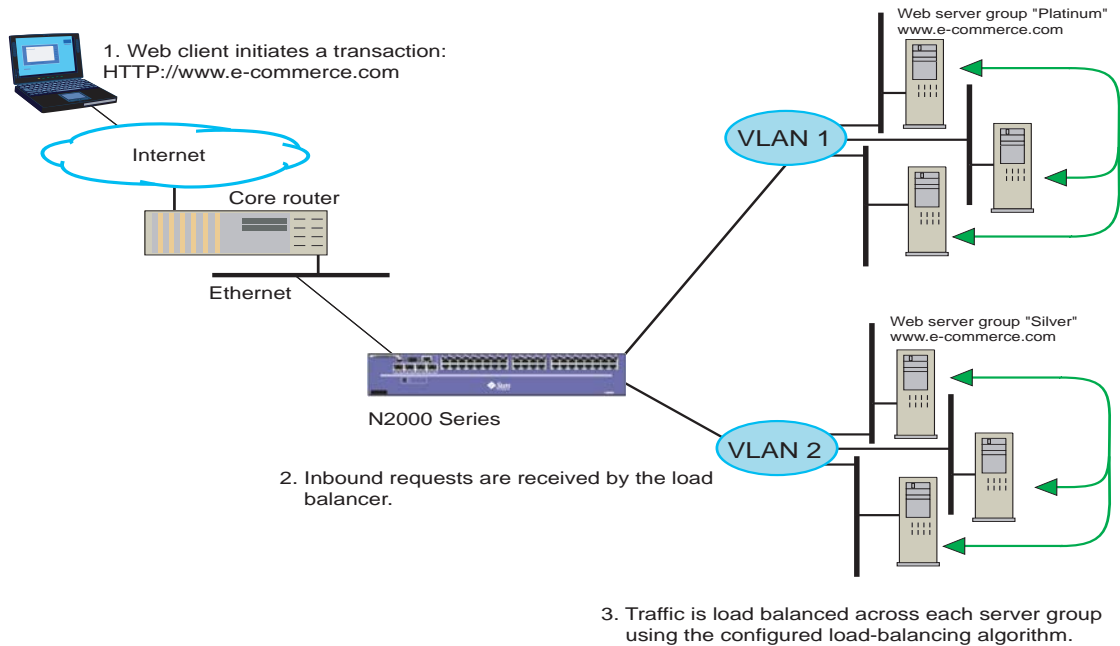
Internet Message Access Protocol Load Balancing

IMAP Load Balancing provides support for load balancing of IMAP traffic. IMAP load balancing is configured as a Layer 4 server load balancing application.

Refer to Chapter 7, “Additional load-balancing applications” for more information.

Figure 1-5 illustrates a simple load-balancing application in an N2000 series network.

Figure 1-5. N2000 Series load balancing across Web server groups



Config_19

Client address translation

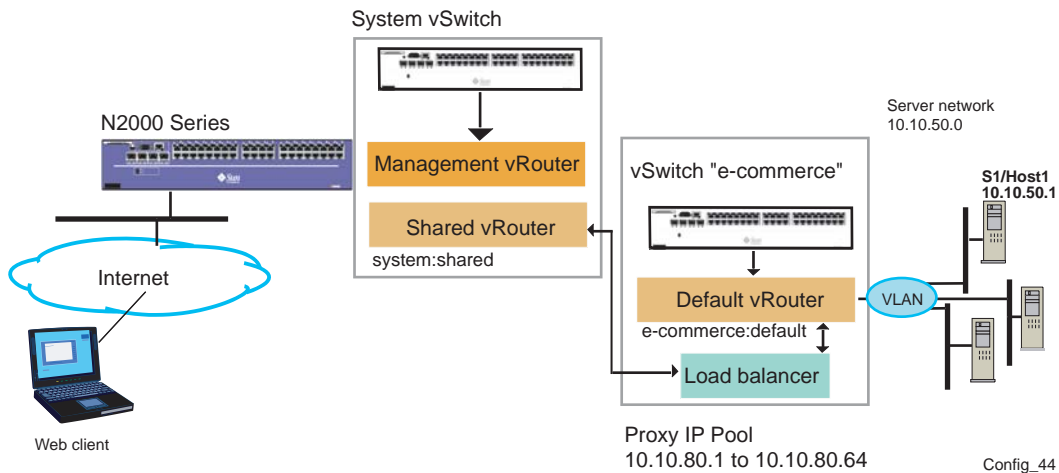
The Sun N2000 Series provides server load balancing (SLB) with client address translation (CAT). On an inbound request, the Internet client's source address and source port information contained in the IP header are translated using a pool of proxy IP addresses. With CAT enabled, the load-balancer application converts the client address and TCP port to a proxy address and port before forwarding the traffic to the selected server.

Typically, the N2000 Series is configured as the default gateway for each host. When the server sends traffic back to the client's global IP address, the server will send it to the N2000 Series default gateway. In some networks, the N2000 Series may not be the default gateway for the backend servers. In this case, using CAT ensures that traffic from the servers destined for clients is first sent to the N2000 Series. There are two network topologies where CAT may be useful:

- Where the N2000 Series is installed in front of a server network that already has an existing load-balancing device that is operating as the default gateway
- Where the real services are located elsewhere in the Internet, with possibly many intervening gateways

Figure 1-6 illustrates a sample network using CAT.

Figure 1-6. N2000 Series CAT network



Refer to Chapter 8, “Configuring client address translation” for more information.

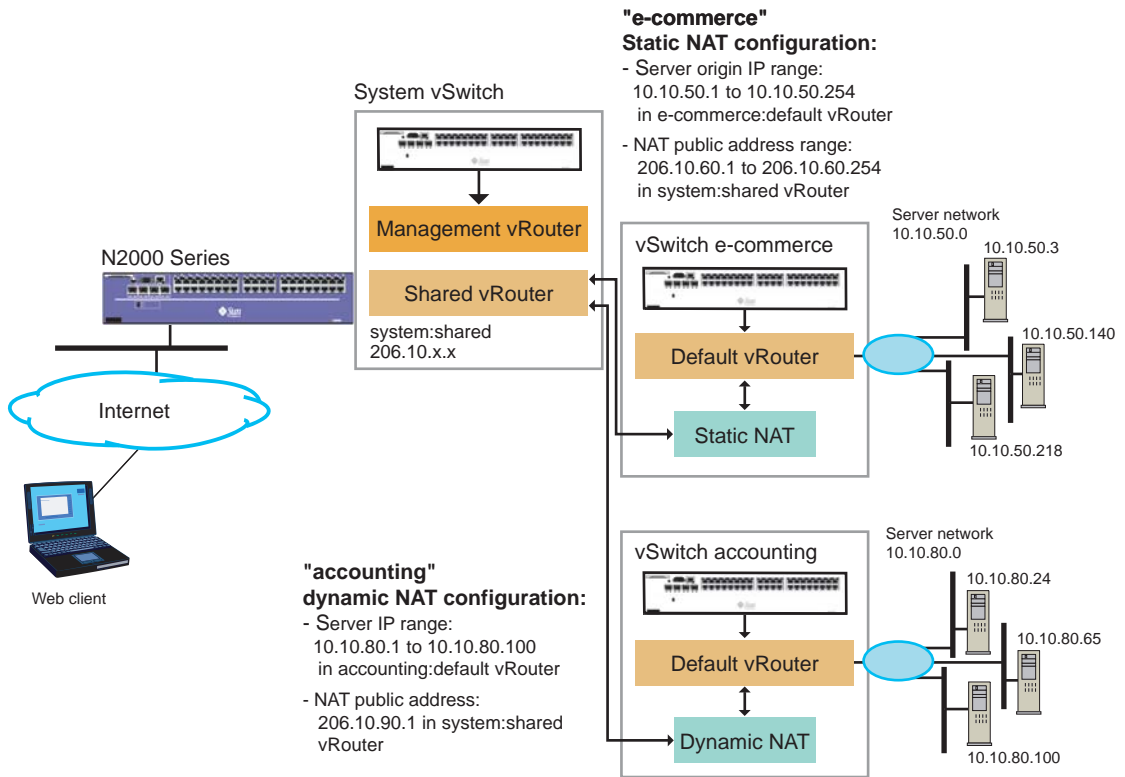
Static and dynamic network address translation

Static network address translation (NAT) and dynamic NAT allow backend servers to access the Internet (acting as clients) while not exposing the backend server private IP addressing information to public Internet. Static NAT translates private IP addresses to unique public addresses by taking outbound traffic initiated from the backend Web server (such as email) and mapping the traffic to global public IP addresses, masking the host IP 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.

Figure 1-7 illustrates a sample network using static and dynamic NAT. The “e-commerce” vSwitch is using a static NAT configuration where the host IP address range has a corresponding one-to-one NAT public address range for outbound traffic to the Internet. The “accounting” vSwitch is using a dynamic NAT configuration where the hosts are using a single public address.

Figure 1-7. Static NAT and dynamic NAT network



Config_43

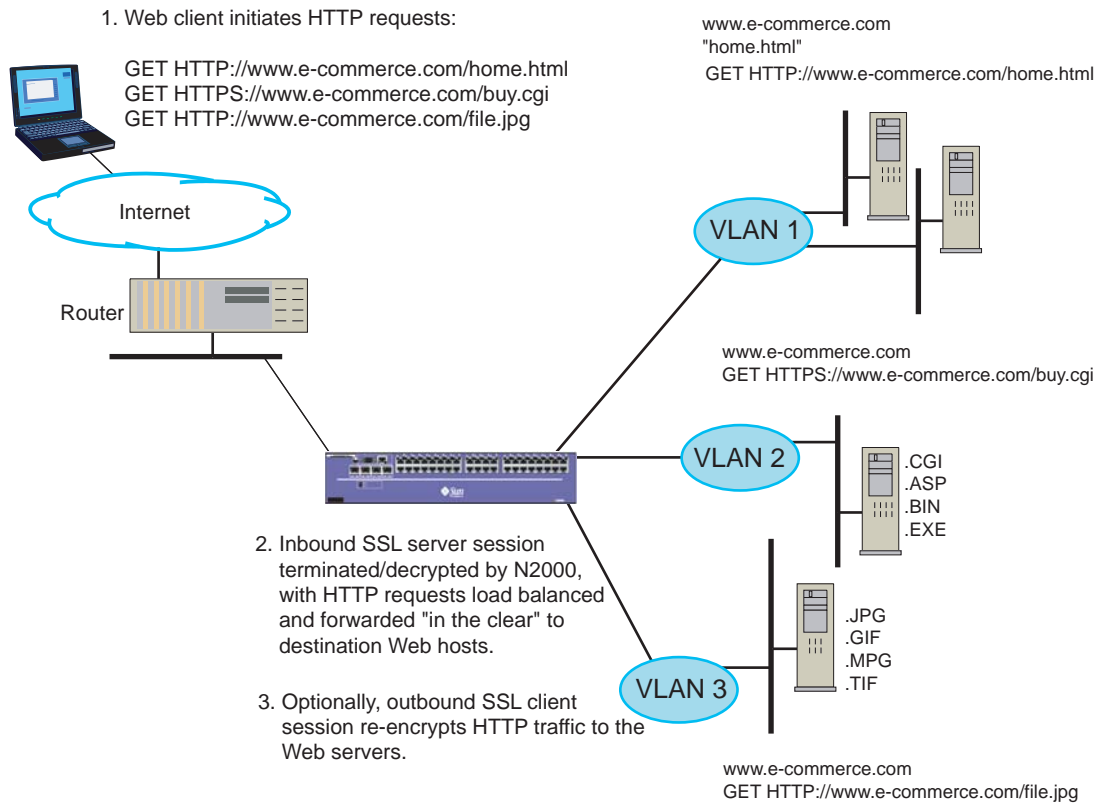
Refer to Chapter 10, "Configuring network address translation" for more information.

SSL acceleration and regeneration

The N2000 Series uses Secure Sockets Layer (SSL) to terminate and decrypt secure requests from Web clients. The N2000 Series off loads the SSL processing responsibilities away from the Web hosts, keeping the servers free for other processing tasks.

The N2000 Series functions as both an SSL client and an SSL server. As an SSL server, the N2000 Series terminates and decrypts client requests from browsers on the Internet, forwarding the traffic in the clear to the destination Web servers. Optionally, as an SSL client, the N2000 Series uses SSL regeneration to re-encrypt the data destined to the backend Web servers.

Figure 1-8 illustrates a sample network where the N2000 Series functions as both an SSL server and an SSL client.

Figure 1-8. N2000 Series as an SSL client and an SSL server

Config_20

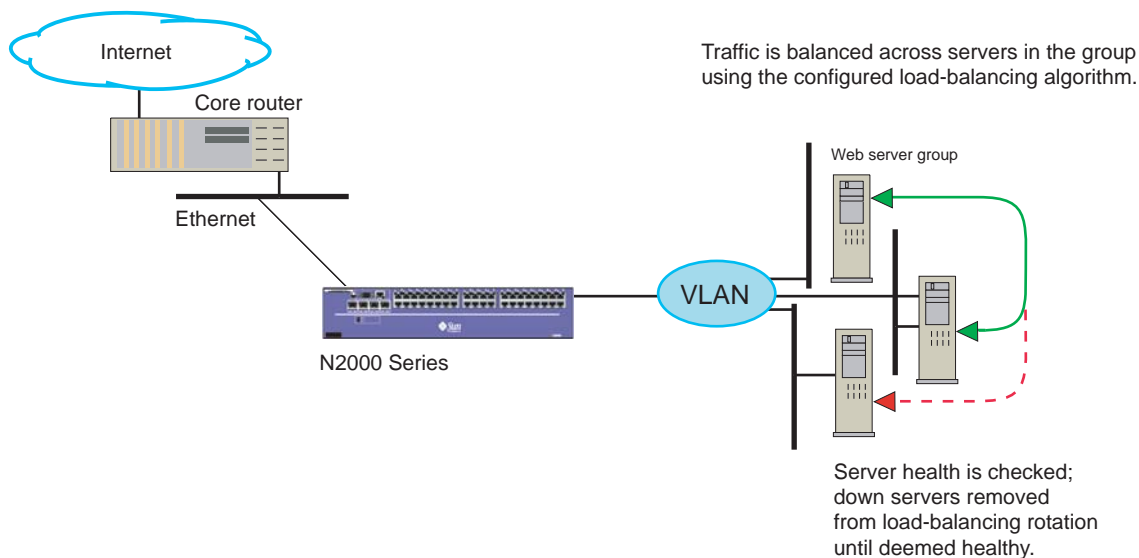
For detailed information on configuring SSL, refer to Chapter 11, “Configuring CKM and SSL acceleration” and Chapter 13, “Configuring SSL regeneration.”

Server health checks

The N2000 Series monitors the state of application servers in a real service group to ensure their availability for load balancing. If a server in the group goes down, the N2000 Series removes it from the load-balancing algorithm, and can dynamically adjust the load preferences. When the server becomes operational, the N2000 Series places the server back into the load-balancing algorithm. The N2000 Series uses TCP, ICMP, LIST, DNS_UDP, DNS_TCP, RADIUS, RAW_UDP, RAW_TCP, HTTP, FTP, RTSP, POP3, IMAP4, SMTP or scripted probes to monitor servers at set intervals using operator-defined settings in the configuration.

Figure 1-9 illustrates a simple network where the N2000 Series detects a failed server and removes the server from the load-balancing rotation.

Figure 1-9. N2000 Series server health checks



Config_21

For information on configuring server health checks, see Chapter 12, “Configuring server health checks.”

Access control lists

In addition to SSL acceleration, the N2000 Series uses access control lists (ACLs) to permit or deny inbound and outbound traffic on vRouter interfaces. An ACL consists of one or more rules that define a traffic profile. The N2000 Series use this profile to match traffic, permitting or denying traffic forwarding to resources on the backend servers.

For detailed information on configuring access control lists, refer to Chapter 14, “Configuring access control lists and groups.”

N2000 Series redundancy

N2000 Series redundancy uses the Virtual Router Redundancy Protocol (VRRP), as described in RFC 2338, and the Sun proprietary Virtual Service Redundancy Protocol (VSRP).

Using two N2000 Series application switches, the redundancy protocols provide link-level and service-level failover capabilities between the two switches. VRRP provides continued service to a redundant N2000 Series switch when a link goes down, while VSRP guarantees continued operation for the N2000 Series virtual services using a configured VSRP peer. Both VRRP and VSRP use an election protocol to decide which N2000 Series switch becomes the master in the event of a service or link failure.

Redundant configurations can be copied and imported to redundant peers to simplify and speed up configuration tasks.

For detailed information on configuring redundancy, see Chapter 15, “Configuring redundancy.”

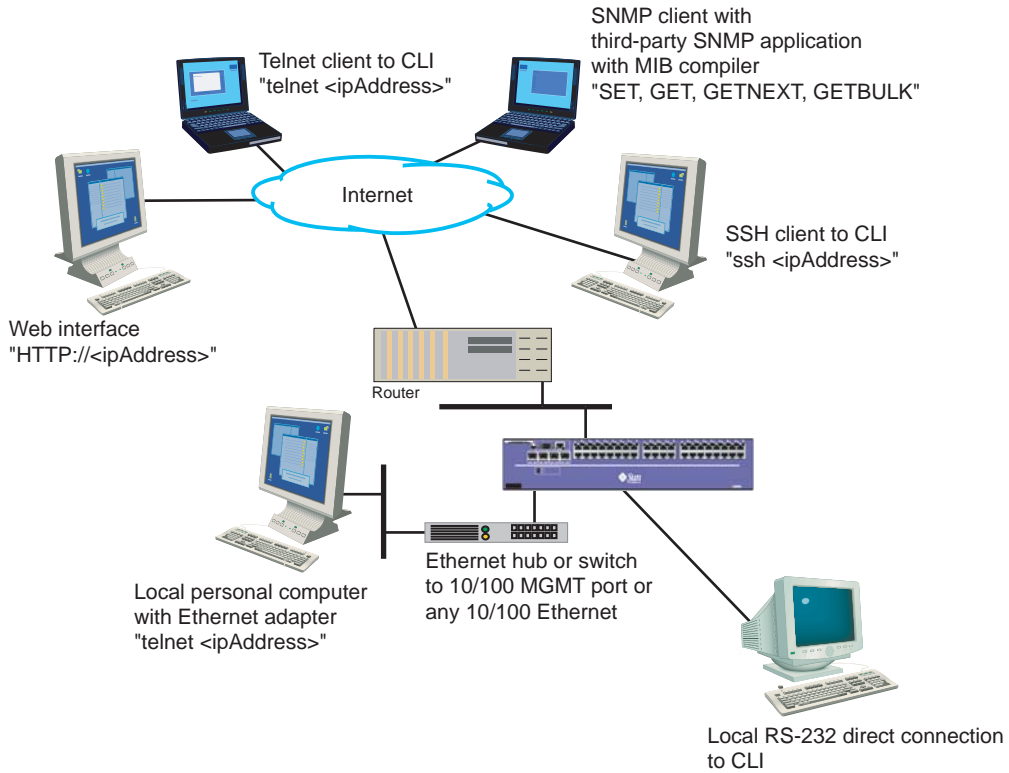
N2000 Series management tools

You can manage the N2000 Series with the following management tools:

- Command-line interface (over local console, Telnet, and SSH connection)
- Sun Application Switch Manager Web interface
- SNMP

Figure 1-10 illustrates these tools in an N2000 Series network.

Figure 1-10. N2000 Series management tools



Config_23

Command-line interface

The command-line interface (CLI) is based on an industry-standard model that allows you to configure and manage the N2000 Series with a computer keyboard. You access the CLI over a direct console connection to the RS-232 port on the front of the system, or over a Telnet or SSH connection. A connection to the CLI is indicated by the `sun>` or operator-defined prompt on your screen.

The CLI design is hierarchical, allowing you to move deeper into the hierarchy as you build the configuration. The CLI command prompt displays your current position in the hierarchy. Simple commands allow you to navigate to the appropriate context.

For details on using the CLI, refer to the following manuals:

- *Sun N2000 Series Release 2.0 – Command Reference*
- *Sun N2000 Series Release 2.0 – System Administration Guide*

Web interface

The Sun Application Switch Manager Web interface is a graphical user interface (GUI) that allows you to configure and manage the N2000 Series using popular Web browsers. The Web interface supports all management capabilities provided by the CLI. Instead of entering information on a command line, you navigate menus and supply information in menu fields. The Web interface also has advanced editors that help you build the configuration.

For details on using the Web interface, refer to the *Sun N2000 Series Release 2.0 – Introduction Guide*

SNMP

The Simple Network Management Protocol (SNMP) allows you to communicate with the SNMP agent on the N2000 Series from a remote management station. This allows you to query information about managed objects on the switch as well as change configuration settings.

The N2000 Series supports the following SNMP versions:

- SNMPv1
- SNMPv2c
- SNMPv3

The N2000 Series supports the standard SNMP commands: GET, GETNEXT, GETBULK, SET. It does not support any of the INFORM commands.

For detailed information on using SNMP to manage the N2000 Series, refer to the following manuals:

- *Sun N2000 Series Release 2.0 – Command Reference*
- *Sun N2000 Series Release 2.0 – System Administration Guide*

Before you start the N2000 Series configuration

If you are configuring the N2000 Series over the Web, Telnet, or SSH, ensure that your N2000 Series hardware is properly installed and connected to the network.

For information on installing, cabling, and assigning the first IP address to the system, refer to the *Sun N2000 Series Release 2.0 – Hardware Installation and Startup Guide*.

N2000 Series configuration steps

There are several steps that you must perform to configure the N2000 Series application switch after you have installed the hardware properly in your data center.

Standard configuration steps

1. **Set up the system vSwitch.** (See Chapter 2.)
 - Configure the management vRouter.
 - Configure the shared vRouter.
2. **Create the operator-defined vSwitches.** (See Chapter 3.)
 - Configure the default vRouter.
 - Allocate port and memory resources.
3. **Configure the L2/L3 network on each vSwitch.** (See Chapter 4.)
 - Configure vRouter interfaces and the network topology:
 - IP over Ethernet
 - IP over virtual LAN (VLAN)
 - IP over link aggregation group (LAG)

- IP over VLAN over LAGs
 - Configure the Spanning Tree Protocol (STP) settings on VLANs.
 - Configure IP routing, static routes, and Routing Information Protocol (RIP) interfaces.
- 4. Configure the L4 to L7 load balancing on each operator-defined vSwitch.** (See Chapter 5, “Load balancing L4 network traffic” and Chapter 6, “Load balancing L5 to L7 network traffic.”)
- Create hosts, real services, service groups, and virtual services for L4SLB, L4SLB_ADV, HTTP, and HTTPS applications.
 - Create request policies and optionally request transforms, response policies and response transforms for HTTP and HTTPS applications.
 - Select the load-balancing algorithm to use in each service group.
 - Create object rules for HTTP and HTTPS applications.

Optional configuration steps

- Configure additional load balancing applications, such as Bridge Mode Load Balancing, Transparent Device Load Balancing (TDLB,) FTP Load Balancing, RTSP Load Balancing, SMTP Load Balancing, IMAP Load Balancing. (See Chapter 7.)
- Configure client address translation. (See Chapter 8.)
- Set up cookie persistence on the load balancer. (See Chapter 9.)
- Configure static and dynamic NAT to shield private IP addresses from the Internet. (See Chapter 10.)
- Create server health check profiles to ensure server availability in the load-balancing algorithm. (See Chapter 12.)
- Configure CKM and SSL acceleration and regeneration for secure applications. (See Chapter 13.)
- Configure access control lists to permit or deny traffic forwarding to the backend Web servers. (See Chapter 14.)
- Configure system redundancy. (See Chapter 15.)

Chapter 2. Using the system vSwitch

Introduction

This chapter describes the resident system vSwitch on the Sun N2000 Series application switch.

Topics

This chapter includes the following topics:

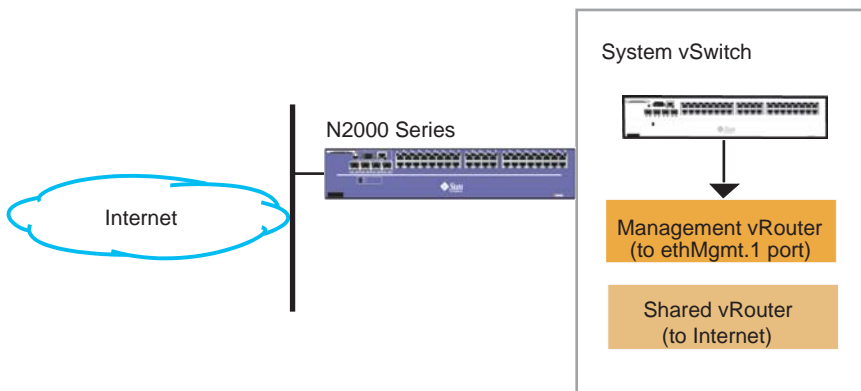
Topic	Page
System vSwitch overview	2-2
How to Display system vSwitch information	2-2
Configuring the management vRouter	2-3
How to assign an IP address to the management vRouter	2-4
How to configure an Ethernet port as the management interface	2-5
How to set up a default route to the local gateway	2-5
Configuring shared vRouters	2-6
How to configure multiple shared vRouters	2-7

System vSwitch overview

As described in Chapter 1, the system vSwitch is the predefined virtual switch on the N2000 Series. It supports the resident *management* vRouter and one preconfigured *shared* vRouter (by default), as illustrated in Figure 2-1.

The system vSwitch provides the management, switching, and routing interface(s) to the Internet (or intranet).

Figure 2-1. N2000 Series system vSwitch



Config_2

How to Display system vSwitch information

The following CLI session displays default settings and operational status associated with the system vSwitch.

CLI Session

```
sun> enable
sun# show vswitch
Name:                system
ID:                  0
Description:         System vSwitch
Admin State:         enabled
Operational Status: up
sun#
```

Configuring the management vRouter

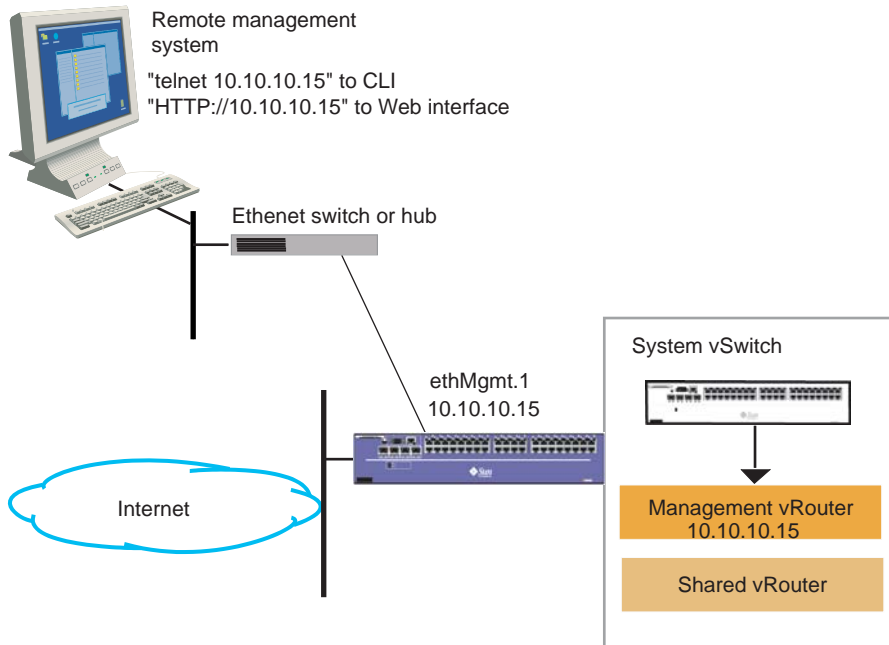
By default, the management vRouter uses the 10/100-Mbps Ethernet management port called ethMgmt.1. This is an independent out-of-band port specifically designed for management traffic. It has its own connection to the N2000 Series, so that it does not compete with data traffic for bandwidth.



Note: You do not have to use the 10/100-Mbps ethMgmt.1 port for system management applications. You can use any Ethernet port (or ports) on the N2000 Series switch. You can also use VLAN interfaces for management. However, both management traffic on these interfaces may have to compete with data traffic.

Figure 2-2 illustrates a sample management vRouter configuration where the system ethMgmt.1 port uses the sample IP address 10.10.10.15.

Figure 2-2. Management vRouter network



Config_3

How to assign an IP address to the management vRouter

If you have not already assigned a management IP address, as described in the *Sun N2000 Series Release 2.0 – Hardware Installation and Startup Guide*, assign an IP address to the management vRouter using the following sample CLI session as a guide.

The following CLI session uses the 10/100-Mbps Ethernet management port called *ethMgmt.1*. The IP address is specified in dotted decimal format with an optional network mask.

CLI Session

```
sun> enable
sun# config
```

```
sun<config># vswitch system
sun<config-vswitch-system># vrouter management
sun<config-vswitch-system-vrouter-management># ip
sun<config-vswitch-system-vrouter-management ip># address ethMgmt.1
10.10.10.15 255.255.255.0
sun<config-vswitch-system-vrouter-management>#
```

How to configure an Ethernet port as the management interface

The following CLI session configures a standard Ethernet 10/100 Mb/s port called *eth.1.5* as the management interface. The IP Address is specified in decimal format with an optional network mask.

CLI Session

```
sun> enable
sun# config
sun<config># vswitch system
sun<config-vswitch-system># vrouter management
sun<config-vswitch-system-vrouter-management># ip
sun<config-vswitch-system-vrouter-management ip># interface eth.1.5
sun<config-vswitch-system-vrouter-management ip># address eth.1.5
10.10.10.15 255.255.255.0
```

How to set up a default route to the local gateway

After configuring the management port, you should configure a static IP default route to the default gateway for the management network. A default route allows access to the N2000 Series management port from management stations on different subnets.

The following CLI session creates a default route to a default gateway with an IP address of 10.10.10.10.

CLI Session

```
sun(config-vSwitch-system vRouter-management)# ip  
...vRouter-management ip)# route static destAddr 0.0.0.0 mask 0.0.0.0  
nextHop 10.10.10.10
```

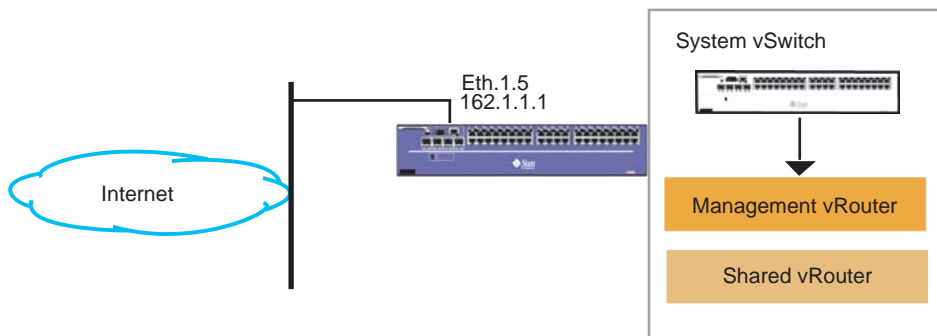
Configuring shared vRouters

The N2000 Series supports up to four “shared” vRouters, with each shared vRouter providing access to the Internet or other shared networks. This means that one shared vRouter can be used by multiple operator-defined vSwitches.

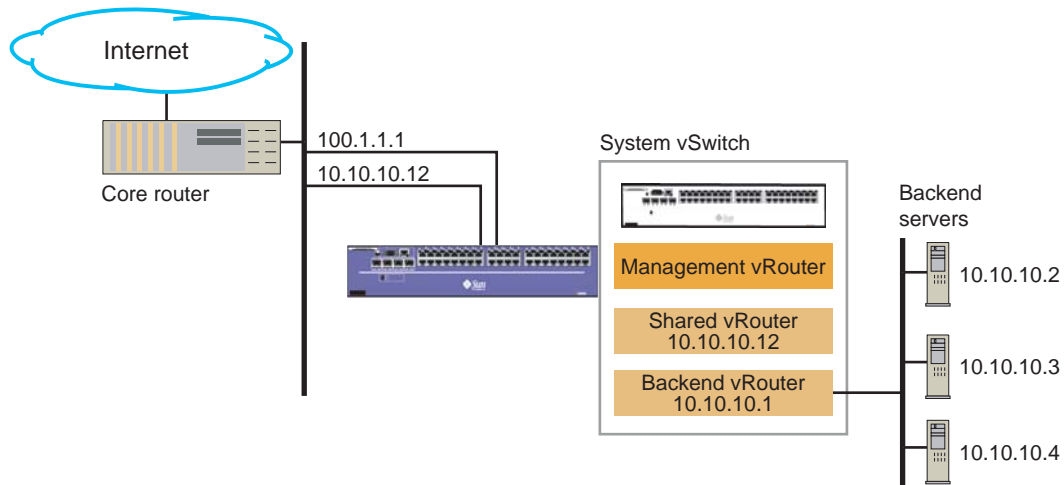
By default, the N2000 Series software automatically creates one vRouter named *shared*. Other shared vRouters that you create must each have a unique name.

Figure 2-3 illustrates a sample network with one shared vRouter for Internet access. Figure 2-4 illustrates the same network using multiple shared vRouters, one for Internet access and one for a shared backend network. Note that these examples show a single IP interface per vRouter, even though a vRouter can have many interfaces.

Figure 2-3. System vSwitch using one shared vRouter



Config_4

Figure 2-4. System vSwitch with two shared vRouters

Config_24

How to configure multiple shared vRouters

The following configuration session configures two shared vRouters on the system vSwitch: *shared* and *backend*. In this example, the *shared* vRouter routes traffic over interface eth.1.12 (Ethernet port 12, 10.10.10.12), and the *backend* vRouter routes traffic over interface eth.1.16 (Ethernet port 16, 10.10.10.1).

CLI Session

```
sun(config)# vSwitch system
sun(config-vSwitch-system)# vRouter shared
sun(config-vSwitch-system vRouter-shared)# ip interface eth.1.12
...vRouter-shared)# ip address eth.1.12 10.10.10.12 255.255.255.0
sun(config-vSwitch-system vRouter-shared) exit

sun(config-vSwitch-system) vRouter backend
...vRouter-e-commerce)# ip interface eth.1.16
...vRouter-e-commerce)# ip address eth.1.16 10.10.10.1 255.255.255.0
sun(config-vSwitch-system vRouter-e-commerce) exit
```

See Chapter 4, “Configuring the L2/L3 network” for information on configuring VLAN interfaces on a vRouter.

Chapter 3. Creating vSwitches

Introduction

This chapter describes how you create the operator-defined vSwitches.

Topics

This chapter includes the following topics:

Topic	Page
Operator-defined vSwitch overview	3-2
Default vRouters	3-3
Load-balancer application	3-4
Network address translation	3-4
Creating the first operator-defined vSwitch	3-5
How to create an operator-defined vSwitch	3-6
Allocating port and service bandwidth to a vSwitch	3-6

Operator-defined vSwitch overview

In the past, multiple switches were often required to support the requirements of multiple Web tiers, business units, or organizations in the Web data center. A Sun N2000 Series vSwitch allows these resources to be allocated to separate users, all within a single hardware platform.

Each vSwitch can contain:

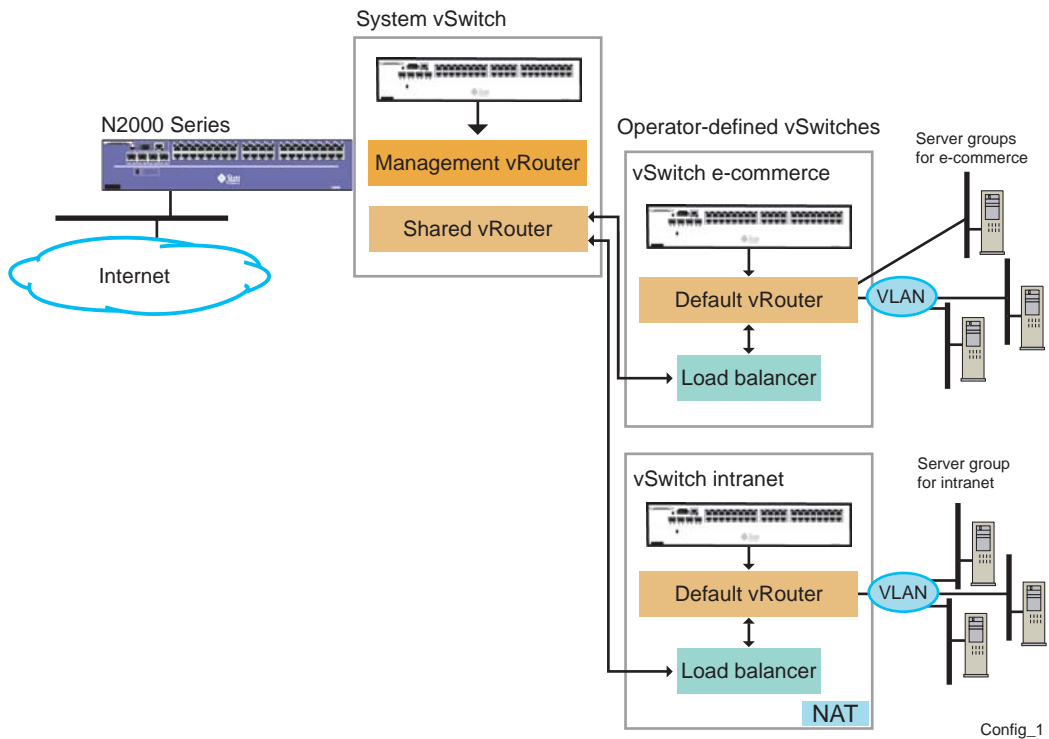
- A full-functioned virtual IP router
- Provisioned bandwidth on various Ethernet ports
- Application services, including
 - Server load balancing
 - Secure Sockets Layer (SSL) termination
 - Network address translation (NAT)
- A consistent management view

The specific Sun virtualization option that you are running controls the maximum number of operator-defined vSwitches that you can have in your network. Refer to the current *Sun N2000 Series Release 2.0 – Release Notes* for information on the available virtualization options.

Creating multiple vSwitches allows you to partition customers and backend servers based on the applications and services they are running.

Figure 3-1 illustrates a sample network using two operator-defined vSwitches, one called “e-commerce” and another called “intranet.” These vSwitches use the *shared* vRouter on the *system* vSwitch for Internet access, and a backend vRouter for the Web servers.

Figure 3-1. N2000 Series network with operator-defined vSwitches



Default vRouters

Each operator-defined vSwitch supports a single vRouter called *default* that typically provides connections to the servers in the backend networks. On a default vRouter, you configure the actual N2000 Series Ethernet or VLAN interfaces, their associated IP addresses, IP routing protocols, and other L3 configuration settings.

Load-balancer application

The L4 to L7 load-balancer (LB) application is assigned one or more virtual IP addresses, called a VIP, which is the address known to external clients. When the load balancer receives a client request (such as an HTTP request) to the VIP, the load balancer forwards the traffic to a selected Web server using a load-balancing algorithm (such as round-robin). When the server responds to the request, the N2000 Series redirects the traffic to the client.

The N2000 Series supports the following types of load balancers:

- Layer 4 Server Load Balancing (L4SLB)
- Layer 4 Server Load Balancing Advanced (L4SLB_ADV)
- Layer 4 Server Load Balancing with Secure Sockets Layer (L4SLB_SSL)
- Hypertext Transfer Protocol (HTTP) and HTTP Secure (HTTPS) object switching
- FTP Load Balancing
- RTSP Load Balancing
- Transparent Device Load Balancing
- SMTP Load Balancing
- IMAP Load Balancing

For an overview of the load-balancing applications, refer to Chapter 1, “N2000 Series overview”. For information on configuring N2000 Series server load balancing, refer to Chapter 5, “Load balancing L4 network traffic”, Chapter 6, “Load balancing L5 to L7 network traffic”, and Chapter 7, “Additional load-balancing applications”.

Network address translation

Network address translation (NAT) is an application that allows hosts in backend networks to access services over the Internet. NAT is used to translate private backend addresses into public addresses that can be used over the Internet. The N2000 Series supports the following NAT applications.

Static NAT

With static NAT, the N2000 Series maintains a one-to-one mapping of private IP addresses to public IP addresses. No TCP/UDP port translation is done.

Dynamic NAT

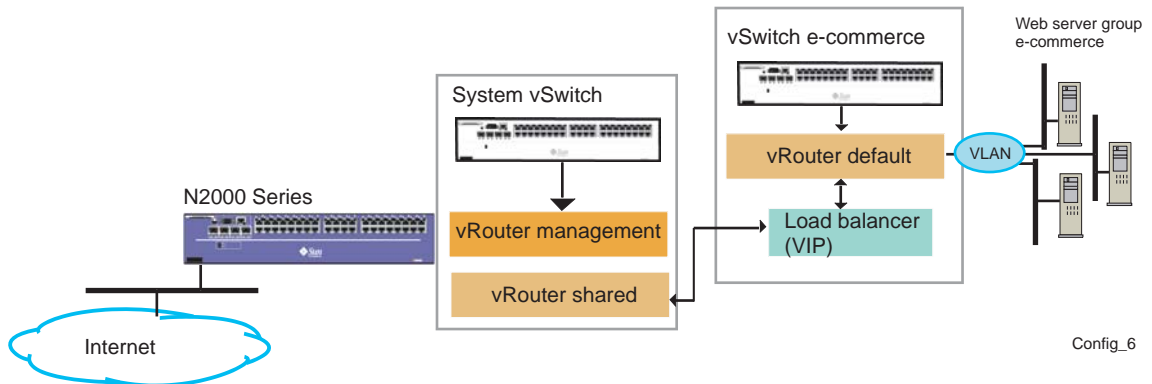
With dynamic NAT, private IP addresses are mapped to a single public IP address. Port translation must also be performed to avoid collisions.

For detailed information on static and dynamic NAT, refer to Chapter 10, “Configuring network address translation.”

Creating the first operator-defined vSwitch

Figure 3-2 illustrates a sample network showing the creation of a vSwitch named *e-commerce*. Within the *e-commerce* vSwitch, the software creates a vRouter named *default*.

Figure 3-2. Network with one operator-defined vSwitch



How to create an operator-defined vSwitch

The following configuration session creates the operator-defined vSwitch named *e-commerce*. The `show vswitch` command displays the current vSwitches configured on the N2000 Series.

CLI Session

```
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# exit
sun(config)# show vswitch
Name:                e-commerce
ID:                  2
Description:         e-commerce
Admin State:         enabled
Operational Status: up
```

Allocating port and service bandwidth to a vSwitch

The N2000 Series software allows you to customize the port and service bandwidth on each configured vSwitch. Port bandwidth allocation is associated with each N2000 Series Ethernet port, while service bandwidth allocation is associated with function card resources (memory) and a configured percentage to allocate (or guarantee) to each vSwitch. You should allocate resources based on the predicted traffic load and real services running on the backend servers. For example, if you are using HTTP virtual services implemented on the function card, you need to configure service bandwidth to the vswitch.

For detailed information on allocating vSwitch resources, see the *Sun N2000 Series Release 2.0 – System Administration Guide*.

Chapter 4. Configuring the L2/L3 network

Introduction

This chapter describes how to configure the basic functions for L2/L3 networks.

Topics

This chapter includes the following topics:

Topic	Page
L2/L3 Network Configuration Overview	4-2
Configuring IP interfaces	4-3
How to enter the vRouter context	4-4
How to configure IP over Ethernet interfaces	4-5
How to configure IP over VLAN over Ethernet interfaces	4-6
How to configure IP over LAG over Ethernet interfaces	4-8
How to configure IP over VLAN over LAG interfaces	4-10
Spanning Tree Protocol overview	4-11
Terminology and concepts	4-11
Spanning Tree Protocol overview	4-11
How to configure the Spanning Tree bridge	4-13
How to configure Spanning Tree ports	4-14
Configuring IP static routes	4-15
How to configure an IP static route	4-15

Topic	Page
Configuring the Routing Information Protocol	4-16
How to configure RIP interfaces	4-17
Configuring Open Shortest Path First	4-16
How to display OSPF command options	4-18
Configuring the OSPF areas and interfaces	4-19
How to create OSPF areas and interfaces	4-20

L2/L3 Network Configuration Overview

Each operator-defined vSwitch supports a single default vRouter. Typically, you configure the actual N2000 Series IP interfaces and other L2/L3 configuration settings on the default vRouter. However, your specific network design determines which objects you need to configure and which vRouter requires these configurations.

For example, if you have a physical router between the N2000 Series and your connection to the Internet, you may want to configure a static route and Routing Information Protocol (RIP), or Open Shortest Path First (OSPF) interfaces for the shared vRouter in the system vSwitch.

Based on your network configuration, you should perform some or all of these tasks to configure the L2/L3 network on the N2000 Series:

For each vRouter:

- Configure virtual router interfaces on Ethernet ports, link aggregation groups (LAGs), and virtual LANS (VLANs).
- Enable the Spanning Tree Protocol (STP) on VLANs.
- Configure static routes, Routing Information Protocol (RIP), and Open Shortest Path First (OSPF) interfaces on vRouters that connect to external routers.

Configuring IP interfaces

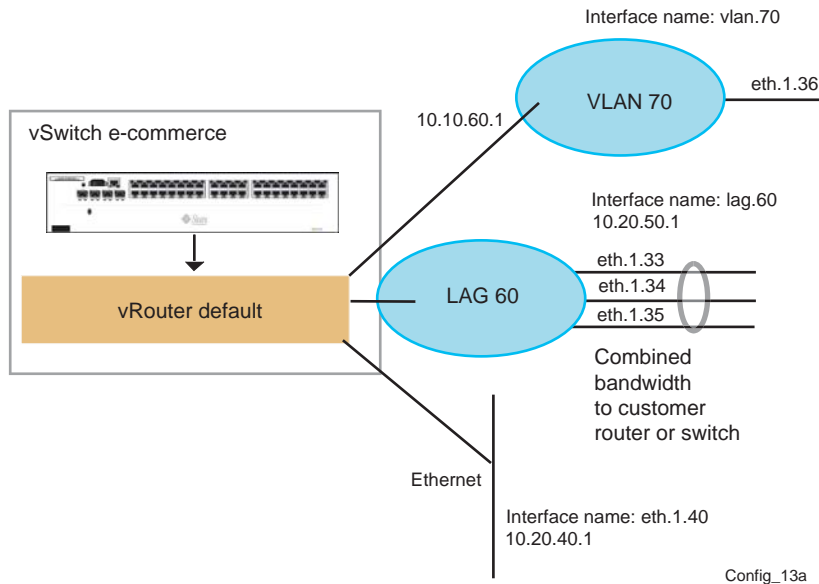
Depending on your network configuration, you may need to configure IP interfaces on the shared vRouters and the default vRouters.

This section uses the *default vRouter* in an operator-defined vSwitch to illustrate how to configure IP interfaces. The *default vRouter* is a virtual IP router for the routing domain that typically contains the backend Web servers.

The IP routing function supports three types of interfaces: 10/100-Mbps or Gigabit Ethernet ports, link aggregation groups (LAGs), and virtual LANs (VLANs).

Figure 4-1 illustrates a sample network using these interfaces.

Figure 4-1. N2000 Series Ethernet, LAG, and VLAN interfaces



The following CLI sessions show you how to create a basic default vRouter configuration on a vSwitch and how to create the physical L2/L3 network:

- How to enter the vRouter context
- How to configure IP over Ethernet interfaces

- How to configure IP over VLAN over Ethernet interfaces
- How to configure IP over LAG over Ethernet interfaces
- How to configure IP over VLAN over LAG interfaces

How to enter the vRouter context

The following configuration session enters the default vRouter context and displays the available command mode arguments, executable commands, and command modes.

CLI Session

```
sun> enable
sun# config
sun(config)# vSwitch e-commerce
sun(config-vSwitch-e-commerce)# vRouter default
sun(config-vSwitch-e-commerce vRouter-default)# ?

      description <text>                Description of the vRouter
      adminState (enabled|disabled)    State of the vRouter
      vRouter                          Create a new vRouter

GLOBAL COMMANDS                          (can be run from anywhere)
  mode                                    Return the current session user mode
  no                                       Negate a command or set its defaults
  show                                     Show running system information

EXECUTABLE COMMANDS
  interfaces                              Modify existing vRouter interfaces
  ip                                       Modifies the instance of IP on the
                                           vRouter
  ping                                     Sends a packet and listens for an answer
  traceroute                              Trace the route of a packet from the
                                           source host to a remote destination.

  vlan                                     Create new VLAN configuration on vRouter
  vrrp

COMMAND MODES
  ip                                       enter ip command mode
  irdp                                     irdp command mode
  ospf                                    ospf command mode
  rip                                     rip command mode
  vlan                                    enter vlan command mode
  vrrp                                    enter vrrp command mode
```

How to configure IP over Ethernet interfaces

The following configuration session adds Layer 3 IP interfaces to the default vRouter directly over an Ethernet interface. This allows an IP interface to be created directly on an Ethernet interface for cases when the IP subnet only exists on one port.

Ethernet ports are numbered according to the system slot and their position in the slot. Since the N2000 Series is a single-slot device, all Ethernet ports are in slot 1.

Interfaces such as eth.1.20 are Ethernet interfaces to customer networking equipment in the data center, such as other switches, routers, or hosts. On the N2000 Series, the interface labeled eth.1.20 represents the 20th physical Ethernet interface on the switch.

Figure 4-2 illustrates a sample default vRouter with assigned interfaces and IP addresses. Use the `show` command to verify the interfaces and addresses.

CLI Session

```
sun(config-vSwitch-e-commerce-vRouter-default) ip
...vRouter-default ip)# interface eth.1.20
...vRouter-default ip)# address eth.1.20 10.10.20.1 255.255.255.0

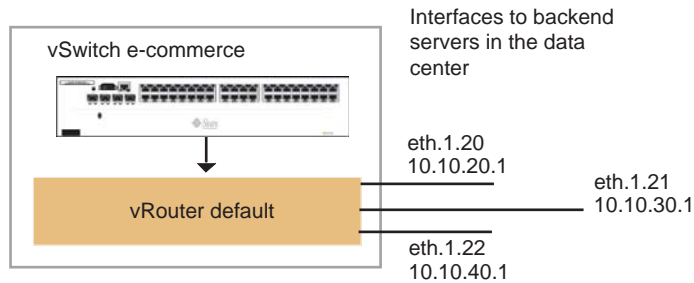
...vRouter-default ip)# interface eth.1.21
...vRouter-default ip)# address eth.1.21 10.10.30.1 255.255.255.0

...vRouter-default ip)# interface eth.1.22
...vRouter-default ip)# address eth.1.22 10.10.40.1 255.255.255.0

sun(config-vSwitch-e-commerce vRouter-default ip)# show interface
IfName      Admin State  Oper Status  MTU      Phys Addr
eth.1.20    enabled      down         1500     00:07:82:00:03:c1
eth.1.21    enabled      down         1500     00:07:82:00:03:c2
eth.1.21    enabled      down         1500     00:07:82:00:03:c3

sun(config-vSwitch-e-commerce vRouter-default ip)# show address
If Index    IP Address    Subnet Mask   VSRP Redirect
eth.1.20    10.10.20.1    255.255.255.0 disabled
eth.1.21    10.10.30.1    255.255.255.0 disabled
eth.1.22    10.10.40.1    255.255.255.0 disabled
```

Figure 4-2. Default vRouter interfaces and IP addresses



Config_8

How to configure IP over VLAN over Ethernet interfaces

The N2000 Series supports virtual LANs (VLANs), allowing the vSwitches to be partitioned at Layer 2 for default vRouter connections to the backend servers. A vRouter can include multiple VLANs, and a VLAN can only exist within a single vRouter. Creating one or more VLANs allows you to group separate LAN segments so that they appear to be on the same Layer 2 network.

Each VLAN is identified by a VLAN ID, and each ID must be unique within the physical switch. This means that multiple vRouters cannot use the same VLAN IDs. VLAN IDs can be in the range of 1 to 4094.



Note: Refer to the *Sun N2000 Series Release 2.0 – Release Notes* for the current number of supported VLANs per N2000 Series system, vSwitch, and vRouter.

Because the VLAN belongs to a vRouter, you must be in a vRouter command mode to create a VLAN.

The following configuration session adds *vlan 10* to the *default* vRouter. The session creates a VLAN named *vlan 10* with three physical interfaces, eth.1.30, eth.1.31, and eth.1.32, and assigns an IP address (10.10.50.1) and network mask to the VLAN.

Figure 4-3 illustrates a sample VLAN configuration.



Note: If you are connecting directly to Web servers that are not running VLAN tagging, you need to disable tagging on the Ethernet interfaces for the VLAN. You also need to make VLAN 10 the default VLAN for the Ethernet port.

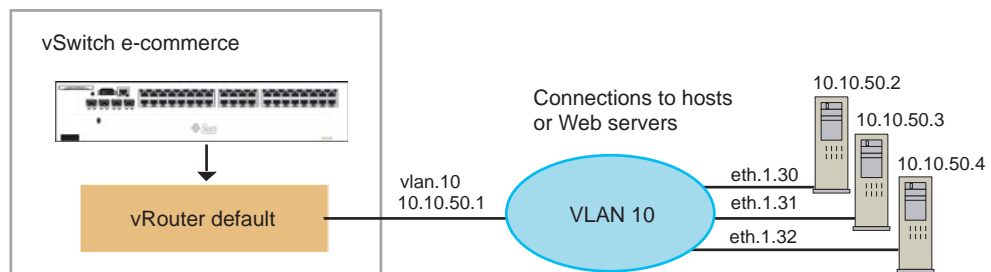
CLI Session

```
sun(config-vSwitch-e-commerce vRouter-default)# vlan 10
...vRouter-default vlan-10)# interface eth.1.30
...vRouter-default vlan-10)# interface eth.1.31
...vRouter-default vlan-10)# interface eth.1.32
...vRouter-default vlan-10)# show interface
```

Vlan Id	Port	Interface	Admin State	Oper Status	Tagging
10	eth.1.30	vlan.10.1	up	down	disabled
10	eth.1.31	vlan.10.2	up	down	disabled
10	eth.1.32	vlan.10.3	up	down	disabled

```
...vRouter-default vlan-10)# exit
sun(config-vSwitch-e-commerce vRouter-default) ip
...vRouter-default ip)# interface vlan.10
...vRouter-default ip)# address vlan.10 10.10.50.1 255.255.255.0
```

Figure 4-3. VLAN configuration



Config_11

How to configure IP over LAG over Ethernet interfaces

Link aggregation groups (LAGs), as defined by the IEEE 803.2ad/D3.0 specification, allow you to configure multiple interfaces so that they appear as a single Media Access Control (MAC) address, or logical interface, to upper layer network protocols.

A LAG provides increased network capacity by totaling the bandwidth of all ports defined by the LAG. The LAG carries traffic at the higher data rate because the traffic is distributed across the physical ports. All ports in a LAG must connect to an adjacent switch also configured with a LAG.

Because a LAG consists of multiple ports, the software load balances inbound and outbound traffic across the LAG ports. If a port fails, the N2000 Series reroutes the traffic to the other available ports.

The N2000 Series supports the following LAG capacities:

- A maximum of 22 LAGs per N2000 Series
- A maximum of 16 Ethernet interfaces per LAG

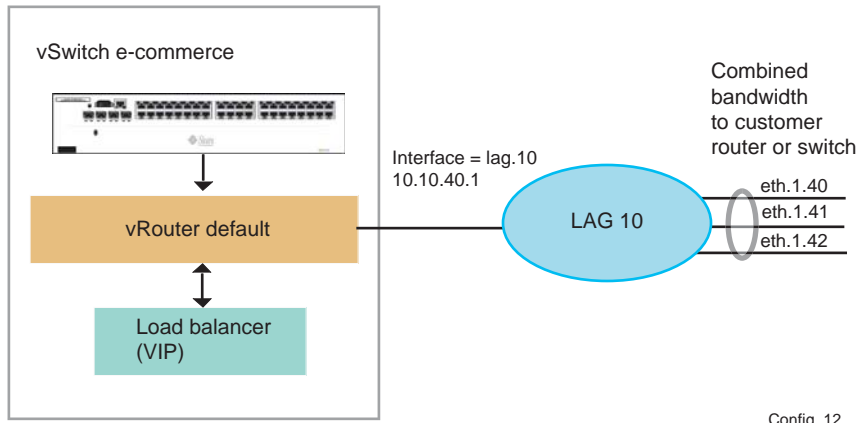


Note: A single LAG definition can comprise mixed Ethernet media, including 10/100-Mbps Ethernet and Gigabit Ethernet. Weighting is used to distribute the traffic load according to link bandwidth.

A LAG is treated as an interface by the higher layer network protocols and is created in the top level system context.

Figure 4-4 illustrates a sample LAG configuration on the default vRouter. The figure is followed by the configuration session that creates this network. Note that each interface has a configured weight, where higher numbers indicate preferred interfaces that will be used more frequently.

Figure 4-4. LAG configuration



Config_12

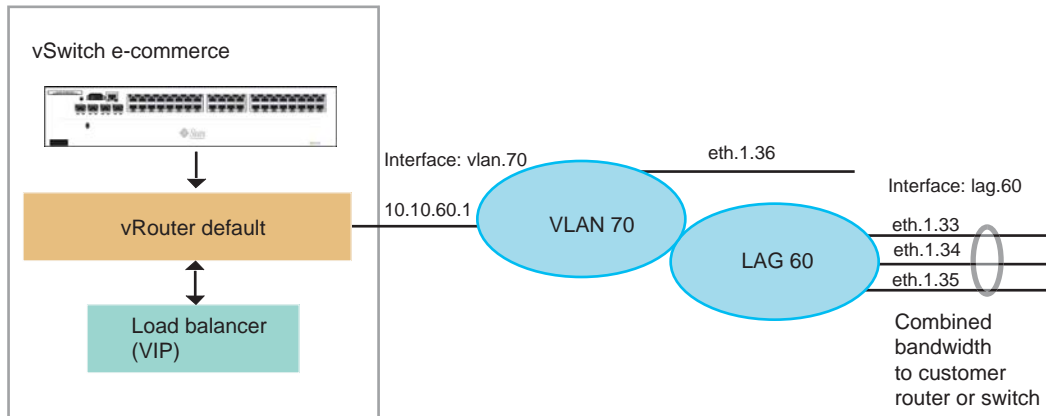
CLI Session

```
sun(config)# lag 10
sun(config-lag-10)# interface eth.1.40 weight 50
sun(config-lag-10)# interface eth.1.41 weight 100
sun(config-lag-10)# interface eth.1.42 weight 200
sun(config-lag-10)# exit
sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
..e-commerce vRouter-default)# ip
..e-commerce vRouter-default ip)# interface lag.10
..vRouter-default ip)# address lag.10 10.10.40.1 255.255.255.0
sun(config-vSwitch-e-commerce vRouter-default ip)#
```

How to configure IP over VLAN over LAG interfaces

Figure 4-5 illustrates a sample network that interconnects a VLAN and LAG to the default vRouter. The configuration session that follows shows how to create a LAG and connect it to the VLAN and the default vRouter.

Figure 4-5. IP/VLAN/LAG over Ethernet network



Config_13

CLI Session

```
sun(config)# lag 60
sun(config-lag-60)# interface eth.1.33
sun(config-lag-60)# interface eth.1.34
sun(config-lag-60)# interface eth.1.35
sun(config-lag-60)# exit

sun(config)# vswitch e-commerce
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 70
...vRouter-default vlan-70)# interface lag.60
...vRouter-default vlan-70)# exit
sun(config-vSwitch-e-commerce vRouter-default) ip
...vRouter-default ip)# interface vlan.70
...vRouter-default ip)# address vlan.70 10.10.60.1 255.255.255.0.
```

Spanning Tree Protocol overview

In order for a bridging network to function properly, there can be no loops in the topology. You can ensure a loop-free configuration by using the Spanning Tree Protocol (STP).

STP detects and eliminates loops in a bridged network by disabling bridged links that it determines are redundant. It can also revive a disconnected network when a link fails by enabling alternate links.

Terminology and concepts

The following terminology may be useful when configuring the Spanning Tree parameters.

Bridge Protocol Data Units

All switches in an N2000 Series VLAN that participate in an STP network exchange messages, called *Bridge Protocol Data Units* (BPDUs), with other switches in the VLAN. The switches use these messages to learn about the other switches in the network and to learn the network topology. The switches in the VLAN transmit BPDUs on all ports at regular intervals. If a switch receives BPDUs on multiple ports, redundant paths exist in the network. STP uses a path cost value to determine which ports can forward data and which port blocks data. A port that blocks data continues to receive BPDUs.

Root bridge (designated root)

The *root bridge* is the main reference point of the Spanning Tree topology and is typically a stable system that does not change frequently. Non-root switches use the root bridge placement in the network to determine whether loops exist. STP uses ports that have the lowest cost path between a non-root bridge and the root bridge to forward data. For paths that are not required to reach the root bridge, STP places the associated port in a standby mode. In this mode, the port does not send or receive data. It does, however, continue to receive BPDUs.

Root port

The *root port* is a port on a switch that has the lowest cost path to the root bridge. Each switch participating in a Spanning Tree group, other than the root bridge, has one root port.

Designated port

A *designated port* is a port that sends and receives data for a specific network segment; designated ports are always in a forwarding state. In the Spanning Tree network, each network segment has only one designated port, which prevents loops. The lowest path cost from a network segment to the root bridge determines which port is the designated port.

For the root bridge, any ports on the switch that are connected to network segments become designated ports; the root bridge has a path cost of 0. For all other switches, the designated port is determined by comparing the path costs of the ports on the switch.

Path costs

As a port prepares to send a BPDU to other switches, STP applies a port cost to the port. The sum of all port costs between a switch and the root bridge is the *path cost*. The 802.1d standards provide guidelines for calculating path costs. You can calculate path costs by dividing the LAN speed in megabits per second by 1000. Typical path costs, based on the 802.1d standard guidelines are listed in Table 4-1.

Table 4-1. 802.1d standard path costs

Bandwidth	Path cost
10 Mbps	100
100 Mbps	19
1 Gbps	4
10 Gbps	2

You can use the interface `spanningTree` command to set path costs for specific ports in a VLAN. By setting the path costs, you can influence whether a port transitions to a forwarding or blocked state. The lower the path cost, the more likely the port will transition to a forwarding state.

Configuring the Spanning Tree Protocol

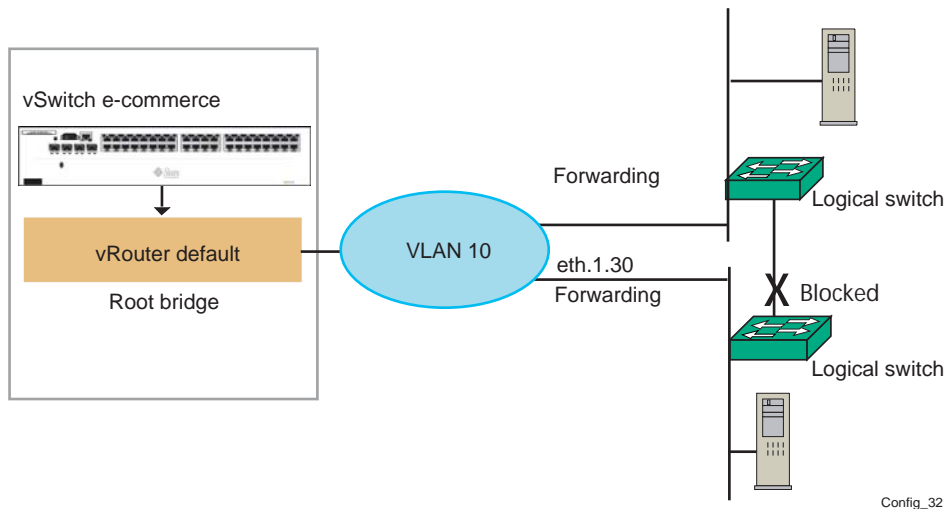
For each virtual LAN (VLAN), you can establish a single instance of a Spanning Tree network. Each instance of the Spanning Tree Protocol ensures that the network does not have loops for a particular VLAN. To configure the Spanning Tree Protocol for a VLAN, you configure bridge and port parameters for the VLAN.

How to configure the Spanning Tree bridge

The following CLI session configures the bridge parameters for *vlan 10*. After enabling the Spanning Tree Protocol for the VLAN, the N2000 Series performs the bridging functions for the VLAN in the Spanning Tree network. Enabling the Spanning Tree Protocol on individual ports determines which ports participate in the Spanning Tree network.

Figure 4-6 shows a sample network where the VLAN bridge (the *default* vRouter) is the root bridge in this Spanning Tree network.

Figure 4-6. Spanning Tree Protocol network



CLI Session

In this session, the bridge priority is set to a value that is lower than the default value of 32768. If all other switches participating in the Spanning Tree Protocol network have a higher bridge ID than the priority configured for the VLAN, it is likely that the VLAN becomes the root bridge in the Spanning Tree network.

```
sun(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 10
..vlan-10)# spanningTree adminState enabled priority 1000

..vlan-10)# show spanningTree
Vlan ID:                10
Admin State:            enabled
Priority:                1000
Time Since Topology Change: 1253471
Topology Changes:      0
Designated Root:       03:e8:00:07:82:0e: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
```

How to configure Spanning Tree ports

This configuration session enables Spanning Tree for the ports on *vlan 10*. The `show` command displays the Spanning Tree port configuration for the VLAN ports.

CLI Session

```
sun(config-vSwitch-e-commerce vRouter-default vlan-10)#

(config-vSwitch-e-commerce)# vrouter default
sun(config-vSwitch-e-commerce vRouter-default)# vlan 10
..vRouter-default vlan-10)# spanningTree enabled
..vRouter-default vlan-10)# interface
..vRouter-default vlan-10 interface)# spanningTree port eth.1.30
adminState enabled
..vRouter-default vlan-10 interface)# spanningTree port eth.1.31
adminState enabled
..vRouter-default vlan-10 interface)# show spanningTree
Interface:                eth.1.30
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:      03:e8:00:07:82:0e:0c:29
Designated Cost:      0
Designated Bridge:    03:e8:00:07:82:0e:0c:29
Designated Port:      80:01
Forwarding Transitions: 0

Interface:             eth.1.31
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:      03:e8:00:07:82:0e:0c:29
```

Configuring IP static routes

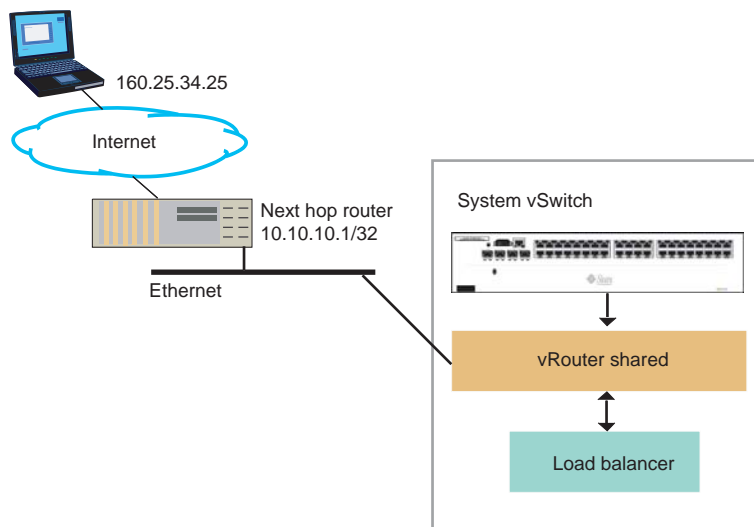
If your network has a single Internet connection or if you want to use a constant route to a specific network destination, configure a static route to the appropriate vRouter. A static route takes precedence over dynamically learned routes and is not overwritten by dynamic routing protocols (such as RIP and OSPF) running in your network.

How to configure an IP static route

The following configuration session configures a static route to a specific host on a remote network.

Figure 4-7 illustrates a sample network with a static route configured from the N2000 Series to a remote host.

Figure 4-7. Network with a static route



Config_30

CLI Session

```
sun(config-vSwitch-system vRouter-shared)# ip
...vRouter-shared ip)# route
...vRouter-shared ip route)# static destAddr 160.25.34.25 mask
255.255.255.255 nextHop 10.10.10.1
```

Configuring the Routing Information Protocol

Routing Information Protocol (RIP) is a dynamic routing protocol that allows routers to broadcast and share learned network routes among other RIP routers in the network. When a RIP router receives a routing update from a configured RIP neighbor, it broadcasts this information to other neighboring routers so that they also synchronize their routing tables. If a failure in the network causes a learned route to become unavailable, RIP calculates a new route around the failure until the original route is reestablished and released.

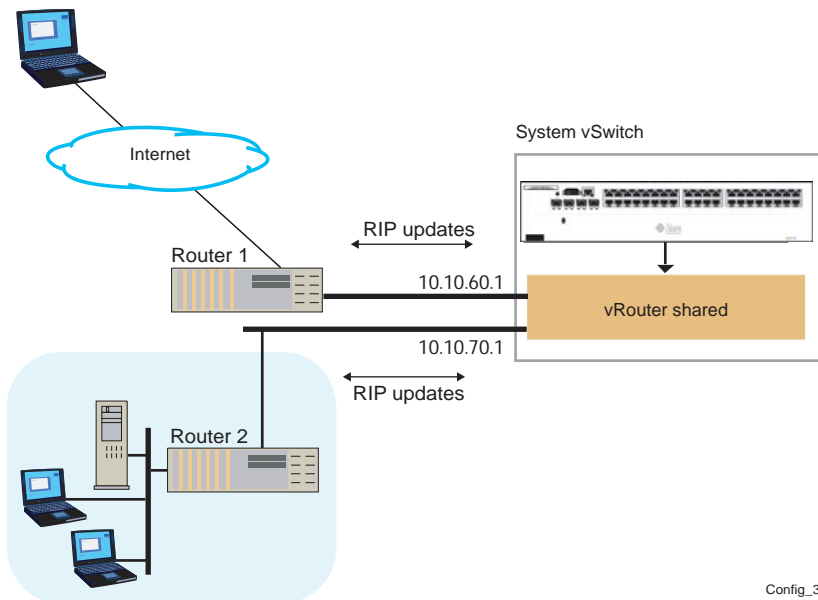
Static routes (predefined) to destinations will take precedence over learned RIP routes.

How to configure RIP interfaces

The following configuration session configures RIP version 1 interfaces for the shared vRouter in the system vSwitch. You must also configure the appropriate IP interfaces for the vRouter; the RIP configuration is not functional until you create the required IP interfaces. The N2000 Series supports both RIP version 1 and version 2, as well as versions 1-2 compatibility mode.

Figure 4-8 illustrates a sample network with RIP interfaces configured for the shared vRouter in the system vSwitch.

Figure 4-8. Network with RIP interfaces



Config_31

CLI Session

```
sun(config-vSwitch-system-vRouter-shared) ip
..vRouter-shared ip)# interface eth.1.43
..vRouter-shared ip)# address eth.1.43 10.10.60.1 255.255.255.0
..vRouter-shared ip)# interface eth.1.30
..vRouter-shared ip)# address eth.1.30 10.10.70.1 255.255.255.0
..vRouter-shared ip)# exit
```

```
sun(config-vSwitch-system vRouter-shared)# rip
```

```
..vRouter-shared rip)# interface address 10.10.10.60.1 send
ripVersion1 receive rip1OrRip2
..vRouter-shared rip)# interface address 10.10.10.70.1 send
ripVersion1 receive rip1OrRip2

..vRouter-shared rip)# show interface
Address:          10.10.60.1
Auth Type:       noAuthentication
Auth Key:
Send:            ripVersion1
Receive:         rip1OrRip2
Default Metric:  1
```

Configuring Open Shortest Path First

Open Shortest Path First (OSPF) is a link-state routing protocol that allows the N2000 Series to maintain least-cost routing information to destinations on the network. The OSPF protocol runs within an autonomous system (AS), where each OSPF router in the AS maintains and advertises the shortest path information (including direct routes, static routes, or RIP routes) with other OSPF routers.

OSPF calculates the total cost path between source and destination based on the speed of an interface, distance to the destination router, and the number of routers (next hops) needed to reach a destination. The path with the lowest cost takes precedence over paths with higher routing costs. If a router on the preferred path is unavailable, OSPF selects the path with the next lowest cost to reach the destination. OSPF routers regularly recalculate the routes to ensure that its database reflects topological changes.

OSPF *areas* minimize routing tables by grouping networks into a single entity. Areas are described by their area ID (specified in IP address format), where area 0.0.0.0 is reserved for the OSPF backbone. OSPF also uses router interfaces (IP addresses) to uniquely identify the routers in an area.

How to display OSPF command options

The following CLI session shows the OSPF command options.

CLI Session

```
sun(config-vSwitch-system vRouter-shared)# ospf
sun(config-vSwitch-system vRouter-shared ospf)# ?
GLOBAL COMMANDS                (can be run from anywhere)
```

mode	Return the current session user mode
no	Negate a command or set its defaults
show	Show running system information
EXECUTABLE COMMANDS	
advertise-ase	Modify OSPF ASE advertise settings
advertise-nssa	Modify OSPF NSSA advertise settings
area	Create OSPF area
areaAggregate	Create OSPF area aggregate
globalSettings	Modify OSPF global settings
host	Create OSPF host
interface	Create OSPF interface
virtualInterface	Create OSPF virtual interface
sun(config-vSwitch-system vRouter-shared ospf)#	

Configuring the OSPF areas and interfaces

The OSPF backbone area distributes routing information between OSPF areas. The backbone is made up of all networks and routers in area ID 0.0.0.0. A router that is directly connected to the backbone is an internal router, with all routers on the backbone being physically adjacent. For non-adjacent routers, you must configure virtual interfaces so that the routers have the appearance of being connected to the backbone.

Stub and non-stub areas

Stub areas are areas where external advertisements are prevented from flooding an OSPF area. A stub area reduces the size of the OSPF route database and memory resources on the internal routers in the stub area.

Non-stub areas (default) flood external routes throughout the network.

Not-so-stubby areas

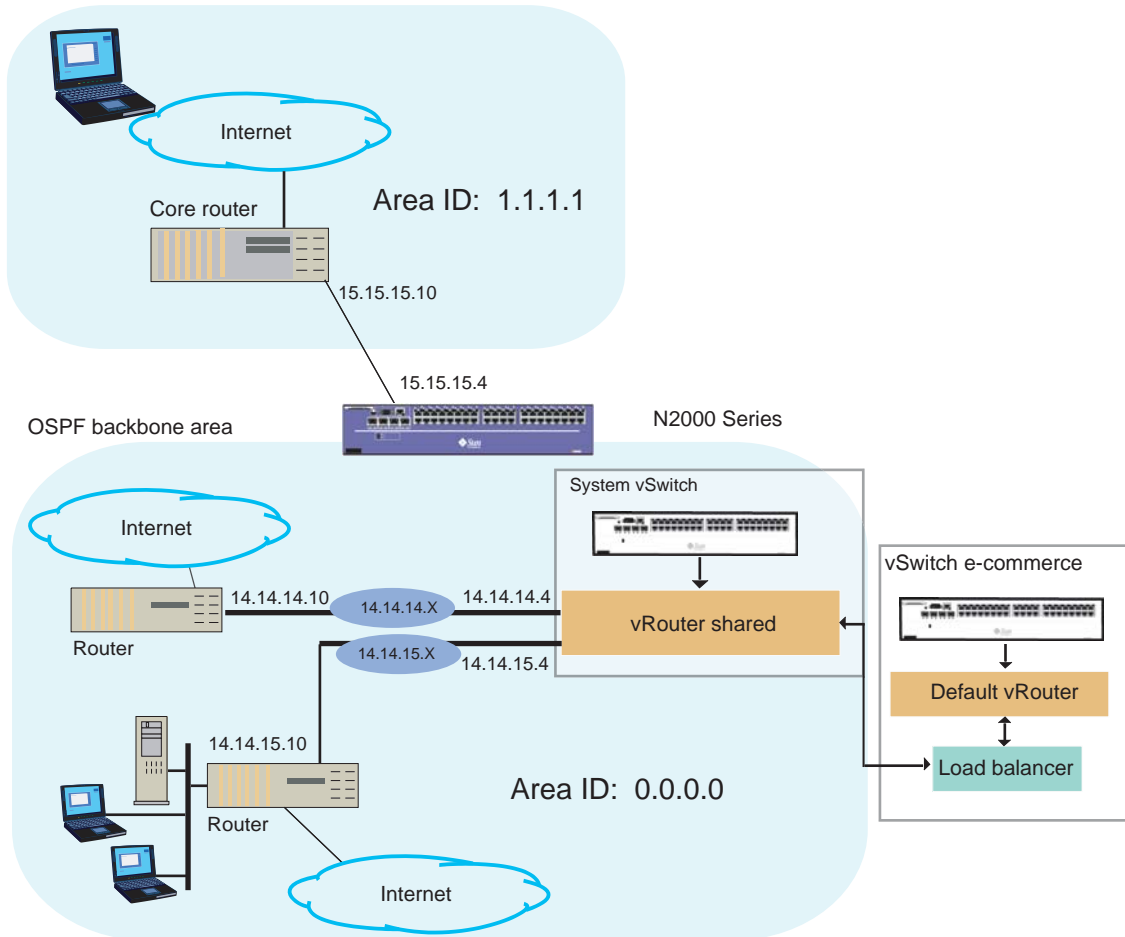
A not-so-stubby area (NSSA) allows external routes to be flooded within the area. These routes are then leaked into other areas. However, external routes from other areas do not enter the NSSA.

Figure 4-9 illustrates a sample OSPF network. The OSPF backbone area (0.0.0.0) consists of the *shared* vRouter and two adjacent routers (each on their own subnets). The non-backbone area (1.1.1.1) represents an external OSPF area.

Creating a basic OSPF configuration consists of creating areas and interfaces on an N2000 Series vRouter.

Figure 4-9. OSPF network

OSPF non-backbone area



Config_47

How to create OSPF areas and interfaces

The following configuration session creates the OSPF areas and interfaces in the sample OSPF network (Figure 4-9).

CLI Session

Configure the OSPF backbone area and interfaces

```
sun(config-vSwitch-system vRouter-shared)# ospf area 0.0.0.0 areaType
nonStub
sun(config-vSwitch-system vRouter-shared)# ospf interface 14.14.14.4
areaid 0.0.0.0
sun(config-vSwitch-system vRouter-shared)# ospf interface 14.14.14.10
areaid 0.0.0.0
sun(config-vSwitch-system vRouter-shared)# ospf interface 14.14.15.4
areaid 0.0.0.0
sun(config-vSwitch-system vRouter-shared)# ospf interface 14.14.15.10
areaid 0.0.0.0
```

Configure the OSPF non-backbone area and interfaces

```
sun(config-vSwitch-system vRouter-shared)# ospf area 1.1.1.1 areaType
nssa
sun(config-vSwitch-system vRouter-shared)# ospf interface 15.15.15.4
areaid 1.1.1.1
sun(config-vSwitch-system vRouter-shared)# ospf interface 15.15.15.10
areaid 1.1.1.1
sun(config-vSwitch-system vRouter-shared)#
```

Display the OSPF area settings

```
sun(config-vSwitch-system vRouter-shared)# show ospf area
Id:                                0.0.0.0
AreaType:                           nonStub
SpfRuns:                             7
BdrRtrCount:                         0
AsBdrRtrCount:                       0
LsaCount:                             1
LsaChecksumSum:                      27298
Summary:                             noAreaSummary
StubMetric:                           1
NssaDefaultRouteType:               type1
StubDefaultRoute:                    enabled
AuthType:                             none
AuthKey:                             N/A
AuthId:                               0

Id:                                1.1.1.1
AreaType:                           nonStub
SpfRuns:                             3
BdrRtrCount:                         0
AsBdrRtrCount:                       0
LsaCount:                             1
```

```

LsaChecksumSum:          27298
Summary:                 noAreaSummary
StubMetric:              1
NssaDefaultRouteType:   type1
StubDefaultRoute:       enabled
AuthType:                none
AuthKey:                 N/A
AuthId:                  0

```

Display the OSPF interface settings

```

sun(config-vSwitch-system vRouter-shared)# show ospf interface
IpAddress:               14.14.14.4
AreaId:                  0.0.0.0
AdminStat:               enabled
RtrPriority:              1
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:                none
AuthKey:                 N/A
AuthId:                  0

IpAddress:               14.14.14.10
AreaId:                  0.0.0.0
AdminStat:               enabled
RtrPriority:              1
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:                none
AuthKey:                 N/A
AuthId:                  0

IpAddress:               14.14.15.4
AreaId:                  0.0.0.0

```

```
AdminStat:          enabled
RtrPriority:         1
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:            none
AuthKey:              N/A
AuthId:              0

IpAddress:           14.14.15.10
AreaId:              0.0.0.0
AdminStat:          enabled
RtrPriority:         1
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:            none
AuthKey:              N/A
AuthId:              0

IpAddress:           15.15.15.4
AreaId:              1.1.1.1
AdminStat:          enabled
RtrPriority:         1
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:            none
AuthKey:              N/A
AuthId:              0
```

```
IpAddress:          15.15.15.10
AreaId:             1.1.1.1
AdminStat:          enabled
RtrPriority:         1
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:           none
AuthKey:            N/A
AuthId:             0
```

```
sun(config-vSwitch-system vRouter-shared)#
```

For detailed information on all OSPF commands, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

Chapter 5. Load balancing L4 network traffic

Introduction

The N2000 Series allows you to load balance Layer 4 to Layer 7 traffic over the operator-defined vSwitches and the Web hosts connected to them.

This chapter describes N2000 Series L4 load-balancing configurations, including:

- Basic L4 (L4SLB) packet-by-packet TCP traffic load balancing based on IP source address and TCP/UDP source port, and a weighted hash algorithm.
- Advanced L4 (L4SLB_ADV) terminated TCP traffic load balancing based on IP source and destination address, L4 source and destination port, and a selected algorithm.

L5 to L7 load balancing differs from L4 load balancing in that the content of the HTTP requests (the “objects”) determine how the requests are handled. For information about L5 to L7 load balancing see Chapter 6, “Load balancing L5 to L7 network traffic” for more information.

For information about FTP Load Balancing, Bridge Mode Load Balancing, Transparent Device Load Balancing, RTSP Load Balancing, SMTP Load Balancing, and IMAP Load Balancing, see Chapter 7, “Additional load-balancing applications” for more information.

Other load-balancing features, such as SSL acceleration, and server health checking associated with L4 to L7 traffic, are covered in other chapters in this guide.

Topics

This chapter includes the following topics:

Topic	Page
L4 load-balancing overview	5-2
L4SLB	5-2
L4SLB_ADV	5-3
L4SLB_SSL	5-3
Terminology and concepts	5-4
L4 load-balancing configuration steps	5-4
L4 load-balancing configuration hierarchy	5-5
How to add a host to the operator-defined vSwitches	5-7
How to create a real service	5-9
How to create service groups	5-12
Load-balancing algorithms	5-13
Load-balancing metrics	5-14
How to create the virtual service for L4 load balancing	5-16
How to create virtual service groups	5-18

L4 load-balancing overview

L4SLB

Layer 4 load balancing (L4SLB) is packet-by-packet TCP traffic load balancing based on the source and destination IP address, source and destination TCP port, and a weighted hash algorithm.

When the N2000 Series receives a TCP SYN packet addressed to a virtual service, it selects a real service to which the SYN will be forwarded. The real service is selected using a weighted hash of the source and destination IP address and the source and destination TCP port. Real services are weighted according to their health, response time, or can be weighted statically.

Once a real service has been selected, the N2000 Series creates a session table entry that directs the client traffic to the selected server, and likewise directs the return server traffic to the client. In each direction, TCP and IP address information must be mapped between virtual service and real service addresses.

If client address translation (CAT) is enabled, the N2000 Series converts the client address and TCP port to a proxy address and port before forwarding the traffic to the selected server. See Chapter 8, “Configuring client address translation” for information about CAT.

L4SLB_ADV

Advanced Layer 4 (L4SLB_ADV) terminated TCP traffic load balancing is based on IP source and destination address, L4 source and destination port, and a selected algorithm: round robin, weighted hash, weighted random, source address, and least connections. L4SLB_ADV uses the Server Load Balancing Function Card (Fx-SSL) that supports high-speed TCP traffic termination and multiple load balancing algorithms.

L4SLB_ADV provides more load balancing options than L4SLB, defeats spoofing, and provides server session management with persistent TCP sessions.

L4SLB_ADV uses the full TCP termination capability of the N2000 Series to implement session load balancing. The N2000 Series completes the three-way TCP handshake before selecting a real service. Completion of the three-way handshake alone defeats Denial-of-Service (DoS) attacks that rely on source address spoofing. Once the handshake is completed, the N2000 Series selects a real service based on a variety of criteria, including the load each server is under, its response time, and response to health checks.

The N2000 Series then typically opens a new, independent TCP session to the selected service (persistent TCP sessions can also be used with real services, see Chapter 8 for details.) and the request is delivered to the selected server. Since a separate TCP session is used with the real service (rather than the one from the client), TCP-based DoS attacks can be eliminated.

L4SLB_SSL

L4SLB_SSL is similar to L4SLB_ADV except that it terminates or regenerates the SSL connection with the client.

Terminology and concepts

You should be familiar with the following terminology and concepts before configuring load balancing on the N2000 Series.

Host

A *host* is a machine, such as a backend server, with an assigned IP address. The IP address configured here is the IP address of an actual server to be load balanced. Hosts of interest here are those 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 TCP or UDP port on which the server application is running. Real services are grouped into service groups for load balancing.

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.

Virtual service

A *virtual service* is the client visible configuration for the server load balancer. A virtual service contains the virtual IP address (VIP) and TCP/UDP port to the load balancer and specifies the application service type for the service group. The virtual service also defines the vRouter that the traffic will come in on.

L4 load-balancing configuration steps

When configuring L4 load balancing, keep in mind that all components need unique names. Components are modified or deleted by using their names. Components are also linked together by name, for example, a service group contains a list of real service names.

For detailed information about the syntax, parameters, and default settings for the load balancing commands mentioned in this chapter, refer to the *Sun N2000 Series Release 2.0 – Command Reference* guide or the online help available with the Web interface.

To configure L4 load-balancing on the N2000 Series, perform the following steps:

1. Add hosts (Web and applications servers) to the vSwitches.

A host typically resides on the backend network and is identified by a host name and an IP address. These IP addresses are kept private and are not exposed to the Internet. See “How to add a host to the operator-defined vSwitches” (page 5-7).

2. Define the real services that are running on these hosts.

A real service, associated with a host, defines the expected type of inbound and outbound traffic processed by the host, as identified by the application port. Real services have assigned weights when they participate in load-balancing groups. See “How to create a real service” (page 5-9).

3. Create service groups for fulfilling Web service requests.

A service group combines one or more real service definitions into a group. A service group assigns a particular load-balancing algorithm to the services in the group, along with other configurable characteristics. See “How to create service groups” (page 5-12).

4. Configure virtual services.

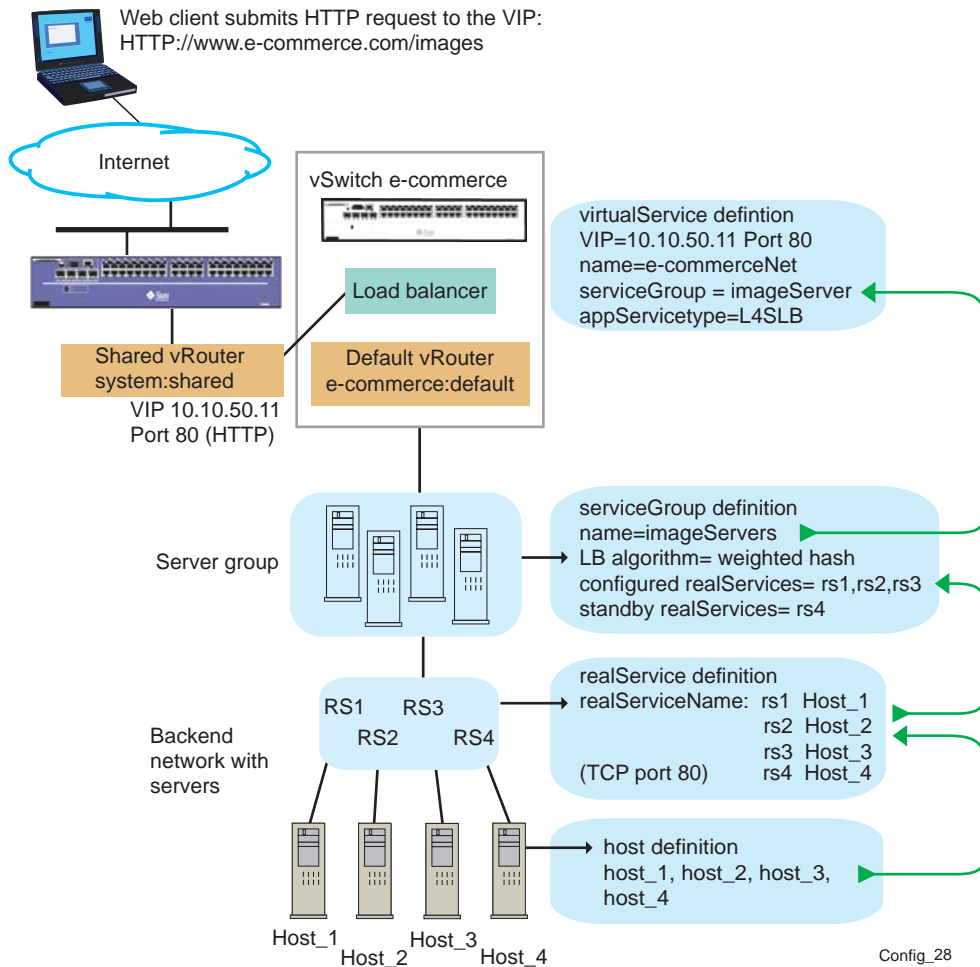
A virtual service contains the client visible configuration attributes for the load balancer. A virtual service defines the virtual IP address (VIP) and specifies the application service type (L4SLB, L4SLB_ADV, L4SLB_SSL). A virtual service also defines the vRouter for client traffic. See “How to create the virtual service for L4 load balancing” (page 5-16).

Additional application service types are available for L5 to L7 load balancing. For a complete list of the application service types available for load balancing, refer to the *Sun N2000 Series Release 2.0 — Release Notes*.

L4 load-balancing configuration hierarchy

Figure 5-1 illustrates the Layer 4 load-balancing configuration hierarchy, starting with the host definitions at the bottom of the illustration. The load balancer uses the service and policy definitions to balance and forward traffic from the client according to the configured load-balancing algorithm

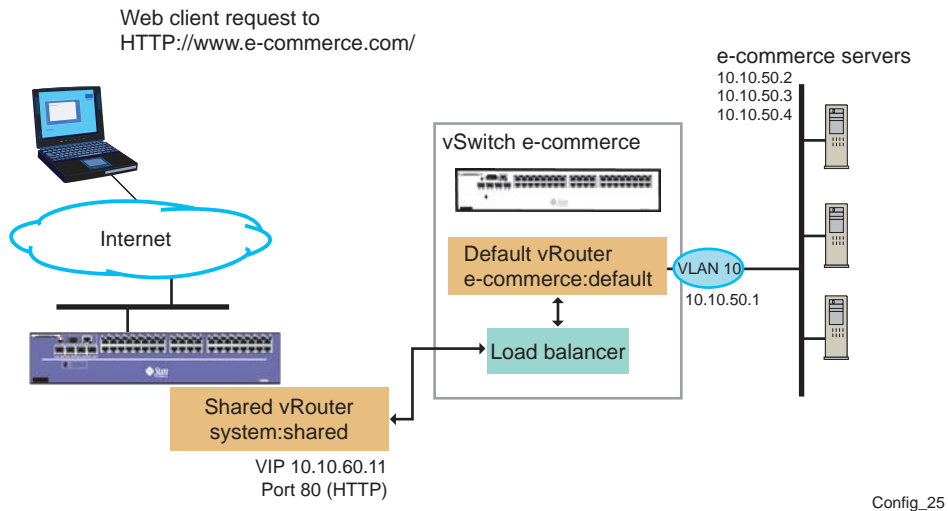
Figure 5-1. N2000 Series L4 load-balancing configuration hierarchy



Sample network

Figure 5-2 illustrates a sample e-commerce network using a Web server group on network 10.10.50.x. A remote client is using a Web browser to access the hosts. This sample network is used throughout this chapter to build an L4 load-balancing configuration.

Figure 5-2. E-commerce network



How to add a host to the operator-defined vSwitches

The host is a machine such as a backend server with an assigned IP address. Each host requires a name and an IP address. The IP address defined here is the IP address of the actual server to be load-balanced.

The host definition includes a set of default settings that you can customize, including:

- A description of the host.
- Administrative setting (enabled, disabled).
- The vRouter on which the server will be receiving traffic. The default is the vRouter of the current vSwitch.

Required elements

Each host definition requires the following elements:

- Name
- IP Address

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of the parameters available for the `host` command.

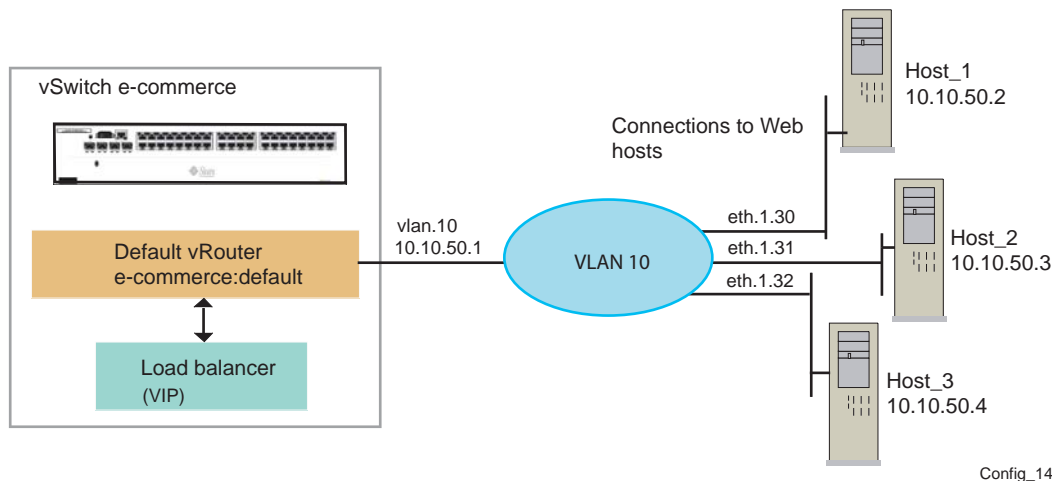
Configuration steps

To configure a host, perform the following steps:

1. **Select Host from the Load Balance menu in the Web interface or use the `Host` command in the CLI.**
2. **Enter a Host-Name and the IP address of the host.**
Note: Two hosts cannot be defined with the same IP address.
3. **Change the default settings of the other optional parameters if appropriate.**

Figure 5-3 illustrates a sample network with three Web hosts connected to `vlan10` on the vSwitch named `e-commerce`.

Figure 5-3. Configuration with backend hosts



CLI Session

The following CLI session creates this configuration:

- Adds *vlan10* and three Ethernet interfaces to the default vRouter
- Creates the interface named *vlan.10* and assigns an IP address to the vRouter interface
- Adds the Web servers named *host_1*, *host_2*, and *host_3* to the network

```
sun(config-vSwitch-e-commerce vRouter-default)# vlan 10
..vRouter-default vlan-10)# interface eth.1.30
..vRouter-default vlan-10)# interface eth.1.31
..vRouter-default vlan-10)# interface eth.1.32
..vRouter-default vlan-10)# show interface
Vlan ID      Port                Admin State Oper Status Tagging
10           eth.1.30            enabled     down      disabled
10           eth.1.31            enabled     down      disabled
10           eth.1.32            enabled     down      disabled

..vRouter-default vlan-10)# exit
sun(config-vSwitch-e-commerce vRouter-default) ip
..vRouter-default ip)# interface vlan.10
..vRouter-default ip)# address vlan.10 10.10.50.1 255.255.255.0
..vRouter-default ip)# exit
sun(config-vSwitch-e-commerce vRouter-default) exit

sun(config-vSwitch-e-commerce)# loadbalance
..vSwitch-e-commerce loadBalance)# host host_1 10.10.50.2
..vSwitch-e-commerce loadBalance)# host host_2 10.10.50.3
..vSwitch-e-commerce loadBalance)# host host_3 10.10.50.4

sun(config-vSwitch-e-commerce loadBalance)# show host
Name          IP Address          Admin State Description vRouter
host1         10.10.50.2          enabled     N/A        e-commerce:def
host_2        10.10.50.3          enabled     N/A        e-commerce:def
host_3        10.10.50.4          enabled     N/A        e-commerce:def
```

How to create a real service

Real service definitions are associated with each host that you are configuring on the network. Real service definitions (within a service group) provide some of the required input parameters for the load-balancing algorithm that you choose.

The real service definition uses a set of default settings that you can customize for your application, including:

- Protocol and port definitions (defaulting to TCP and HTTP, respectively)
The port does not need to match the virtual service destination.
- Real service weight (with the higher number accepting more load-balancing requests compared to other real service definitions with lower weights), or a dynamic weight setting derived from the server health checks
- Administrative settings (real service text description and the current enabled/disabled state)
- SSL settings (regeneration, authentication, SSL protocols, ciphers, renegotiation)
- Bridge mode load balancing, which supports all load-balancing applications without using CAT when clients reside in the same subnet as servers (disabled by default)

Required elements

Each real service definition requires the following elements:

- Name
- Host name

Each real service requires a host definition (illustrated in Figure 5-1). However, multiple real service definitions can be defined on the same host.

- Protocol (defaults to TCP)
- Port (defaults to 80)
- Weight (defaults to 1)

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters available for the `RealService` command.

Configuration steps

To create a real service, perform the following steps:

- 1. Select Real Service from the Load Balance menu in the Web interface or use the `realService` command in the CLI.**

2. Enter a `realService` name, a host name, a protocol, a port, and a weight.
3. Change the default settings of the other optional parameters, as appropriate.
4. Configure the name of the real service in the service group definition.

CLI Session

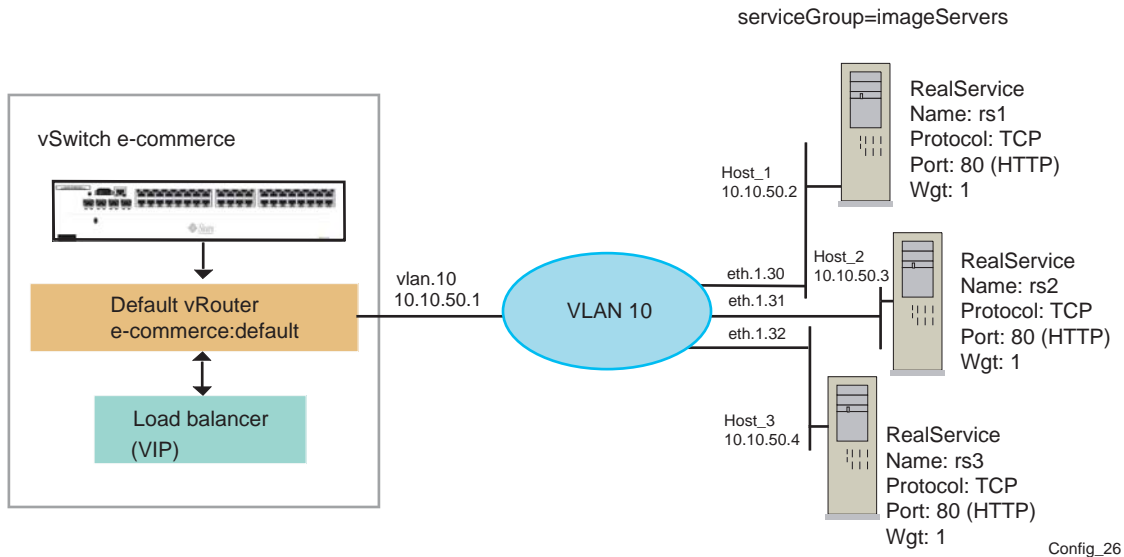
The following CLI session creates the real service named `rs1` on the host named `host_1`. In this example, `host_1` balances TCP traffic received on TCP port 80 using a load-balancing weight of 1. Use the `show` command to display the real service definitions for this host and others that you create.

```
sun(config-vSwitch-e-commerce loadBalance)# realService rs1 host_1  
protocol TCP port 80 weight 1
```

```
sun(config-vSwitch-e-commerce loadBalance)# show realService  
Name: rs1  
Host Name: host_1  
Protocol: TCP  
Port: 80  
Weight In ServiceGroup: 1  
Description: N/A  
Admin State: enabled  
Disable Delay: 0  
inLine SHC Failure Rate Threshold: 1  
Client Address Translation: disabled  
Bridge Mode: disabled  
Encryption: unencrypted  
Persistence Name: N/A  
Virtual Services:  
Oper Status: inactive
```

Figure 5-4 illustrates a sample network using a real service definition on each backend host.

Figure 5-4. Real service definitions on backend hosts



How to create service groups

A service group places real service definitions into one common grouping, and specifies the load-balancing algorithm to be used in making traffic-balancing decisions across the hosts. Optionally, you can configure standby real services that the system brings into service if one of the configured real services fails.

Each service group requires a name, a selected load-balancing algorithm, and the list of hosts that participate in the group (illustrated in Figure 5-1). The service group definition includes a set of default settings that you can customize, for example:

- Administrative settings (current enabled/disabled state)
- Load-balancing metrics
- Server health profile name
- Customized SSL settings for this service group (regeneration, authentication, SSL protocols, ciphers, renegotiation)
- Standby real services

Additional parameters are available for L5 to L7 application types, such as `ResponsePolicyList` and `ResponseTransformList`. See “Load balancing L5 to L7 network traffic” (page 6-1) for more information.

Required elements

Each service group definition requires the following elements:

- Name
- Load-balance type (`roundRobin|weightedRandom...`)
This is the algorithm used for load balancing decisions. See “Load-balancing algorithms” (page 5-13) for more information.
- Real services (list of real services to load balance across)

Refer to the *Sun N2000 Series Release 2.0 – Command Reference* or the online Help for detailed descriptions of all parameters that are available for the `ServiceGroup` command.

Configuration steps

To configure a service group, perform the following steps:

1. **Select Service Group from the Load Balance menu in the Web interface or use the `ServiceGroup` command in the CLI.**
2. **Configure a serviceGroup name, load balance type, and list of real services.**
3. **Change the default settings of the other optional parameters, if appropriate.**

Load-balancing algorithms

There are five load-balancing algorithms that you can specify in the service group:

- Weighted hash
- Weighted random
- Round-robin
- Least connections

For each weighted algorithm, you can assign static weights or a dynamic weight.

Weighted hash

The weighted hash algorithm attempts to distribute traffic evenly across a service group. The weighted hash algorithm uses the load-balancing weight setting associated with each host to see where it can distribute more or less traffic.

When configuring a real service with a static load-balancing weight (instead of using dynamic weight), you should consider that server's ability to handle more or less traffic than other servers in the group. If a server is capable of handling more traffic, then set the server weight to a higher weight (by number) than those weights assigned to other servers in the group. Hashing of the client IP address is used so that generally sessions from a given client will be balanced to the same server.

An L4SLB network supports the weighted hash algorithm only. No other algorithm can be specified with L4SLB.

Weighted random

The weighted-random algorithm distributes traffic to Web servers randomly using weight settings. Servers with higher weights receive more traffic than those configured with lower weight settings during the random selection.

Round-robin

The round-robin algorithm distributes traffic sequentially to the next server in the service group. All servers are treated equally, regardless of the number of inbound connections or the response time.

Least connections

The least connections algorithm dynamically directs traffic to the server with the least number of active connections.

Load-balancing metrics

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.) Dynamic weight is supported for both L4SLB and L4SLB_ADV services.

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 32, to the current average latency.

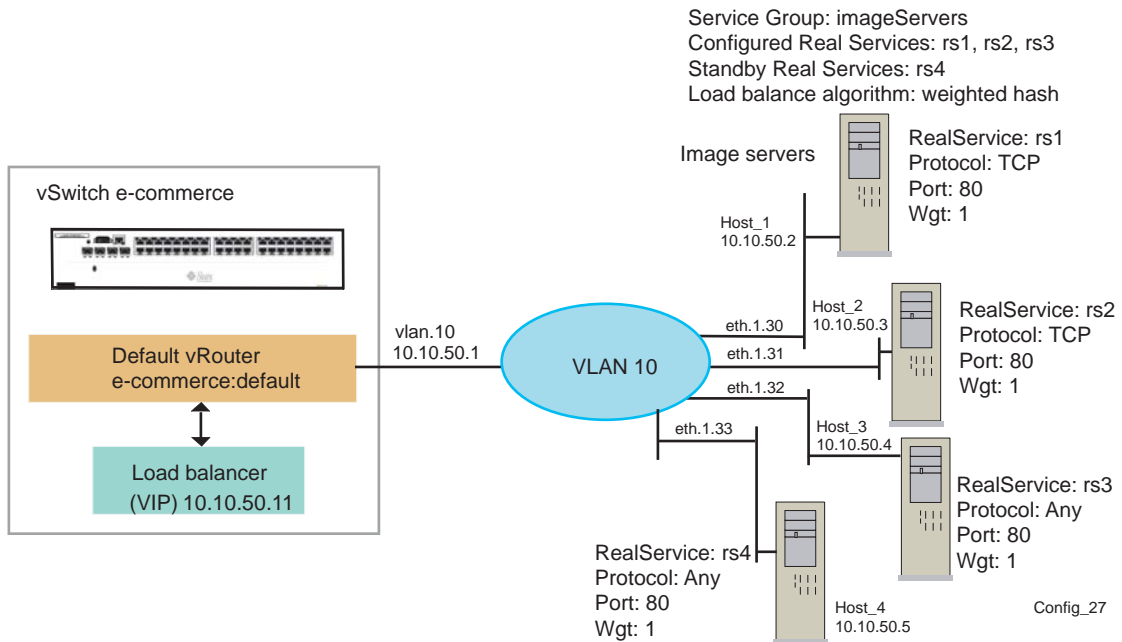
The following CLI session creates the service group named *imageServers*, specifies weighted hash load balancing, and adds the real services that will load balance traffic across this VLAN.

Figure 5-5 illustrates the sample network balancing that occurs across three equally weighted servers.

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# serviceGroup imageServers
weightedHash {rs1 rs2 rs3}
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup
Name:                               imageServers
Load Balance Type:                   weightedHash
Configured Real Services:            rs1; rs2; rs3
Standby Real Services:
Active Real Services:                rs1; rs2; rs3
Admin State:                         enabled
Virtual Services:                   e-commerceNet
Health Check Name:
In-Line Health Check:               enabled
Failover retry Count:               1
Oper Status:                         active
```

Figure 5-5. L4SLB weighted hash configuration across three servers



How to create the virtual service for L4 load balancing

A virtual service contains the client visible configuration attributes for the server load balancer. The virtual service provides the virtual IP address (VIP) to the load balancer and specifies the applications service type for the service group. A virtual service also defines the vRouter on which the client traffic will be received.

Required elements

Each virtual service definition requires the following elements:

- Application Service type:
 - L4SLB, L4SLB_ADV, L4SLB_SSL
 - Additional types are available for L5-L7 load balancing

- Virtual service name
- IP Address - VIP address
- TCP/UDP port
- Service group (For L4 load balancing only. L5-L7 load balancing requires a requestPolicyList.)

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters available for the `VirtualService` command.

Configuration steps

To configure a virtual service, perform the following steps:

1. **Select Virtual Service from the Load Balance menu in the Web interface or use the `virtualService` command in the CLI.**
2. **Configure a Virtual Service Name, Application Service Type, VIP address, and service group.**

The application type you select determines the parameters available for the `virtualService`.

3. **Change the default settings of the other optional parameters if appropriate.**

CLI Session

The following CLI session creates the virtual service named *e-commerceNet*, sets the VIP to *10.10.50.11*, specifies *L4SLB*, and the service group *imageServers*. The `show virtualService` command displays these settings and the default virtual service settings.

By default, the switch links the operator-defined e-commerce vSwitch to the *system:shared* vRouter on the system vSwitch.

```
sun(config-vSwitch-e-commerce loadBalance)# virtualService
e-commerceNet L4SLB 10.10.50.11 L4SLB imageServers
sun(config-vSwitch-e-commerce loadBalance)# show virtualService
```

```
Name:                e-commerceNet
Service Type:        L4SLB
IP Address:          10.10.50.11
Protocol:            TCP
Port:                80
```

```
Service Group:          imageServers
Admin State:            enabled
Disable Delay:         0
vRouter:               system:shared
Client Source IP Address Range: 0.0.0.0-255.255.255.255
SYN Rate Limit:        unlimited
Oper Status:           active
Oper Message:          Operational; Server Health Checking not
                       configured
```

```
sun(config-vSwitch-e-commerce loadBalance) show virtualservice
statistics
```

```
Name:                  e-commerceNet
Bytes Transmitted to clients: 1850
Bytes Received from clients: 1850
Packets Transmitted to clients: 20500
Packets Received from clients: 22456
Cumulative Open Sessions: 578
Cumulative Closed Sessions: 572
Current Open Sessions: 6
```

How to create virtual service groups

Virtual service groups (vsGroups) allow you to combine one or more named virtual services into a single group so that request policies and real services are shared among the virtual services. By creating vsGroups, you can distribute the processing load on N2000 Series systems that use function cards.

The N2000 Series will automatically create a named vsGroup called *default* if you create two or more virtual services that share the same request policy. If your system uses only one function card, then all vsGroups will operate on that function card.

A vsGroup can contain up to 64 named virtual services.

For information about request policies, refer to “Load balancing L5 to L7 network traffic” (page 6-1).

CLI Session

The following configuration 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}
```

Chapter 6. Load balancing L5 to L7 network traffic

Introduction

This chapter covers N2000 Series L5 to L7 server load balancing, object switching, and policy-based matching. The N2000 Series applies object rules to request and response policies that make load balancing decisions based on individual application requests.

Topics

This chapter includes the following topics:

Topic	Page
L5 to L7 load-balancing overview	6-2
Terminology and concepts	6-3
L5 to L7 load-balancing configuration steps	6-6
L7 load-balancing configuration hierarchy	6-8
Sample network	6-10
How to create an object rule	6-10
How to create a request policy	6-13
How to create a proxy IP pool	6-15
How to create a request transform	6-15
How to create a response policy	6-17
How to create a response transform	6-19

Topic	Page
How to define a sorry data	6-20
How to create the virtual service for L5 to L7 load balancing	6-21
How to create virtual service groups	6-22
Object rule details — predicates	6-23
Predicates with URI field names	6-28
Using the predicate operators	6-31
Using predicate keywords	6-34
Using predicate keywordSets	6-36
Forwarding actions	6-37
HTTP load-balancing example	6-39

L5 to L7 load-balancing overview

L5 to L7 load balancing differs from L4 load balancing in that the content of the HTTP requests (the objects) determine how the requests are handled.

L5 to L7 load balancing uses policy-based matching that operates on HTTP headers, cookies, URLs, or actual content over inbound and outbound sessions. In its simplest form, L5 to L7 load balancing operates from client to server and server to client as described below:

- A client sends an HTTP request to a virtual service configured on the N2000 Series. When the N2000 Series receives the request, request policies are applied to the request to determine how it should be processed. If the request policy indicates that the request should be forwarded to the backend servers, the N2000 Series selects a suitable server and forwards the request.
- An HTTP response is sent to the client through the N2000 Series switch. Response policies are applied to the response to determine how it should be processed. A response may be sent to the client, or retried to another real service in the server group. If a server response policy is not defined, the traffic is returned to the client.

L5 to L7 load balancing also provides request and response transforms that can be used to modify an HTTP header. For example, you can add the client's source address to traffic going to the server. You can also insert custom headers into the traffic from the client to the server or from the server to the client.

Terminology and concepts

Before configuring L5 to L7 load balancing, you should be familiar with the specific terminology and concepts used in this chapter.

Host

A *host* is a machine, such as a backend server, with an assigned IP address. The IP address configured here is the IP address of the actual server to be load balanced. Hosts of interest here are those machines running server applications.

Object

An *object* is a message with a defined start 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.

Object rule

An *object rule* is a named predicate. The predicate is named for ease of use in request and response policies and transforms.

Policy

A *policy* is created when an object rule is bound to a forwarding action. Policies that are applied to the traffic received from clients are called request policies. Policies that are applied to the traffic received from servers are called response policies.

Predicate

A *predicate* is a filter that uses one or more match expressions (joined by logical operators) that the load balancer uses to match inbound or outgoing traffic. A predicate can examine all or parts of the URI, HTTP header fields, cookies, and other request and response data, and can perform integer and string comparisons with prefix, suffix, and substring type operations. The predicate is true when the expression matches the object, or false when the expression does not match.

Real service

A *real service* is a server application executing on a host. A real service is identified by its host name and the TCP/UDP port on which the server application is running. Real services are grouped into service groups for load balancing.

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 client request to another real service in the service group; or apply a sorry 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 needs to 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. For example, the response transform supports server cloaking, which is the removal of server signature from HTTP headers. The response transform is optional, but if used, the response transform needs to be configured in the service group definition.

Request policy

A *request policy* is a virtual service policy that is evaluated to make a forwarding decision regarding a request received from a client. If the HTTP request received from a client matches the object rule, the request is forwarded to the service group or a sorry data is applied, depending on the configuration. The request policy specifies an object rule, the action data required to forward the HTTP traffic, the sorry data, and the service group that will receive the request. The request policy must be configured in the virtual service definition. L5 to L7 load-balancing applications require a request policy for each service group. If there is only one service group (and no sorry data), you can use a wildcard request policy (“*”), that will direct all requests to the 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. For example, the request transform defines header insertions, such as source IP address, cipher strength, and customer headers. The request transform is optional, but if used, the request transform needs to be configured in the virtual service 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 data

A *sorry data* 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 data can be defined to close a TCP connection, return an HTML page to a client, redirect a request to a different URI, and reset a TCP connection. The named sorry data is configured in the definition of the request or response policy.

Virtual service

The *virtual service* is the client visible configuration for the server load balancer. When a request comes in from the client, the virtual service evaluates it based on the precedence of the request policies that are listed in the requestpolicylist included in the virtual service definition. When a match is found, the request policy has a service group associated with it and the request is 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 vRouter that the traffic will come in on.

L5 to L7 load-balancing configuration steps

The required steps for configuring L5 to L7 load balancing on the N2000 Series are similar to the configuration steps for L4SLB and L4SLB_ADV in that you need to configure hosts, real services, service groups and virtual services. For L5-L7 load balancing, you also need to configure request policies and object rules.

When configuring L5 to L7 load balancing, keep in mind that all components need unique names. Components are modified or deleted by using their names. Components are also linked together by name, for example, a service group contains a list of real service names. Names can be used before the actual component is created, for example, a service group can refer to a real service that has not been created yet.

For detailed information about syntax, a description of each parameter, and the default settings for the load balancing commands mentioned in this chapter, refer to the *Sun N2000 Series Release 2.0 – Command Reference* guide or the online Help available with the Web interface.

This procedure explains some of what you need to do and points you to the appropriate section in this chapter or in Chapter 5 for more information.

Perform the following steps to configure L5 to L7 load balancing:

- 1. Add hosts (Web and applications servers) to the vSwitch.**

A host typically resides on the backend network and is identified by a host name and an IP address. These IP addresses are kept private and are not exposed to the Internet. See “How to add a host to the operator-defined vSwitches” (page 5-7).

- 2. Define the real services that are running on these hosts.**

A real service, associated with a host, is identified by a real service name and the TCP/UDP port on which the server application is running. The real service defines the expected type of inbound and outbound traffic processed by the host, defined by the IP address and application port. Real services have assigned weights when they participate in load-balancing groups. See “How to create a real service” (page 5-9).

3. Create service groups for load balancing.

A service group combines one or more real service definitions into a group. A service group assigns a particular load-balancing algorithm to the services in the group, along with other configurable characteristics. If defined, response policies and response transforms are configured with the service group. See “How to create service groups” (page 5-12).

4. Create an object rule for matching and forwarding HTTP traffic.

An object rule specifies the pattern matching predicate that is used in request policies. See “How to create an object rule” (page 6-10).

5. Create a request policy.

The request policy specifies the object rule, the action data, (forward or sorry) and the service group required to forward the HTTP traffic to that service group for load balancing. L5 to L7 load balancing applications require that you configure a single, named request policy with each service group. See “How to create a request policy” (page 6-13).

6. Define a sorry data if you are using a sorry action in a request or response policy.

A sorry data is a named component that defines a sorry action and the associated redirect string or filename for the HTML page. See “How to define a sorry data” (page 6-20).

7. (Optional) Create a request transform.

A request transform defines changes to the HTTP header text of the HTTP request from the client. These changes will be made to each request before it is sent to the server, if the request matches the object rule. See “How to create a request transform” (page 6-15).

8. (Optional) Create a response policy.

A response policy defines how the load balancer will process an HTTP response from a server. See “How to create a response policy” (page 6-17).

9. (Optional) Create a response transform.

A response transform defines any changes to the HTTP header of the outgoing HTTP response if the traffic matches the object rule. See “How to create a response transform” (page 6-19).

10. Configure the virtual service.

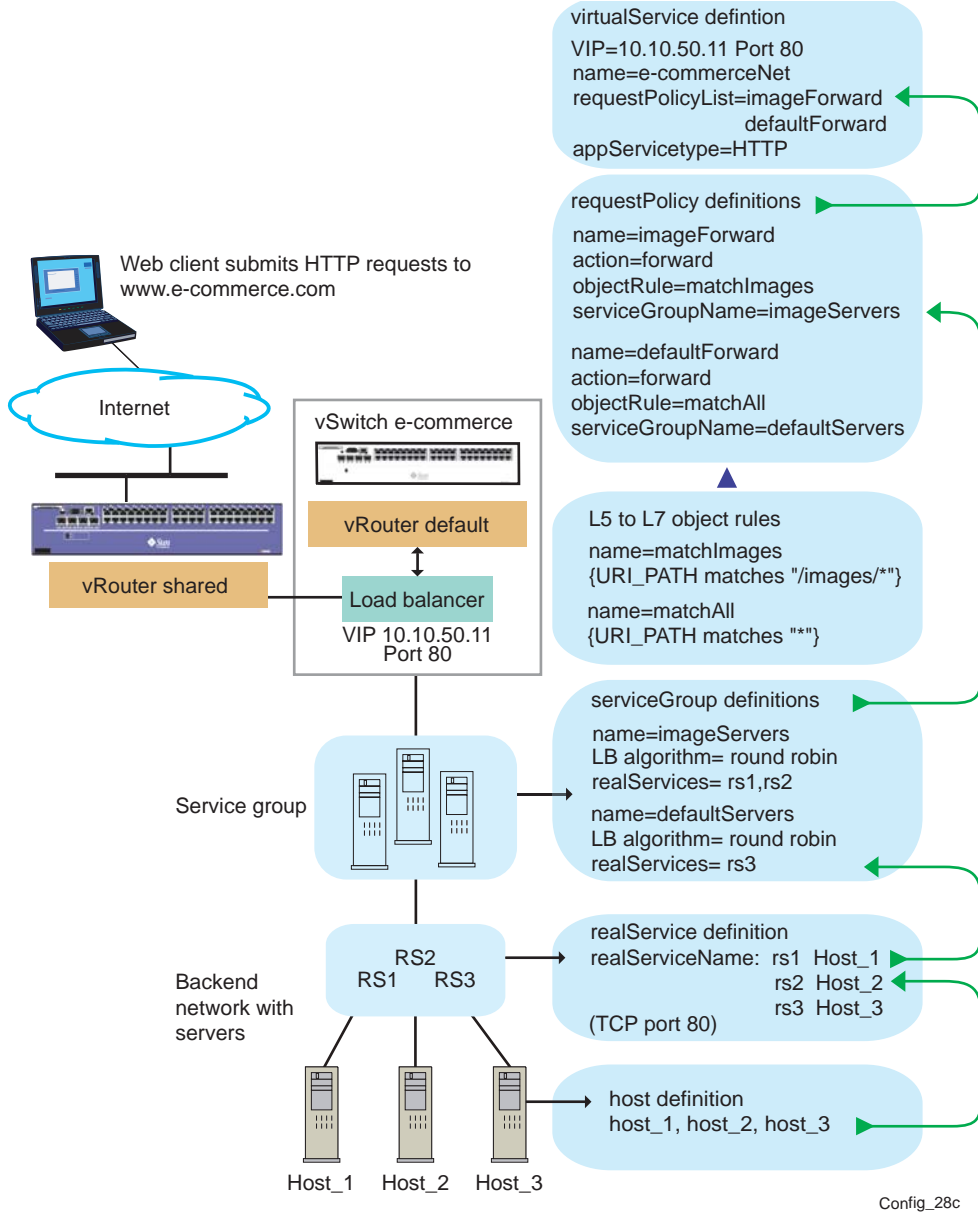
The virtual service links one or more request policies to the virtual IP address (VIP), specifies the application service type (HTTP, HTTPS, TDLB, FTPLB, RTSPLB), specifies the request policy and request transform lists, and also defines the vRouter for network traffic. See “How to create the virtual service for L5 to L7 load balancing” (page 6-21).

For the complete list of application service types available for load balancing, refer to the *Sun N2000 Series 2.0 — Release Notes*.

L7 load-balancing configuration hierarchy

Figure 6-1 illustrates the L7 load-balancing configuration hierarchy. In this hierarchy, the switch evaluates object rules, and associates the rules with a request policy and a service group.

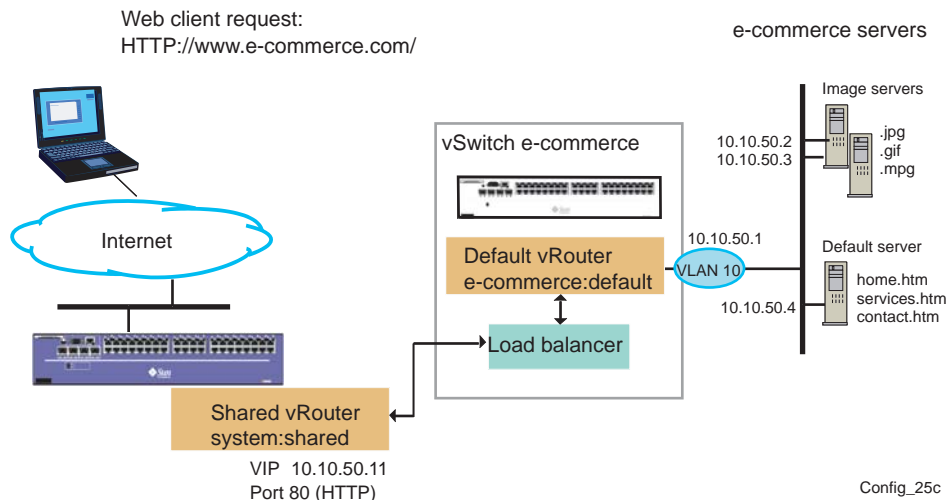
Figure 6-1. L5 to L7 load-balancing configuration hierarchy



Sample network

Figure 6-2 illustrates a sample e-commerce network using three servers on network 10.10.50.x. A Web client is using a Web browser to access the servers. This sample network is used throughout this chapter to build an object rule configuration.

Figure 6-2. E-commerce network



How to create an object rule

An object rule is a named statement that describes how the switch must evaluate an object, such as an HTTP Request or 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 policy or a response policy will execute a specific action such as forward or sorry or specific header transformations will be done. The name of an object rule is configured in the definition of the request policy and the response policy.

Required elements

Each object rule requires a name and a predicate.

A predicate consists of an HTTP field name or a URI field name, followed by a predicate operator, followed by a text string or keyword.

A predicate can examine all or parts of the URI, HTTP header fields, cookies, and other request and response data, and can perform integer and string comparisons with prefix, suffix, and substring type operations. The predicate is true when the expression matches the object, or false when the expression does not match.

CLI syntax

```
objectRule <objectRule_name> predicate  
{URI field_name: <operator> [integer|string|keyword]}
```

Where:

<objectRule_name> is any unique alphanumeric name with no blank spaces, followed by the predicate statement.

Configuration steps

To configure an object rule, perform the following steps:

- 1. Select `objectRule` from the `LoadBalance` menu in the Web interface and click `Add`.**
Note: You may use the `ObjectRule` command in the CLI; however, using the Object Rule Editor in the Web interface is highly recommended for easier configuration and to avoid syntax and logical errors.
- 2. When the `Add` screen is displayed, enter a name for the object rule.**
- 3. Click the `Editor` button on the `Add` screen to display the `Object Rule Editor Predicate Field` screen.**

4. Follow the instructions on the screen to build the predicate statement for the object rule.

Refer to the following tables in this chapter for descriptions of the fields and operators available for the predicate statements:

- Table 6-1, “HTTP Request and Response header predicates - HTTP fields,” on page 6-23.
- Table 6-2, “HTTP Uniform Resource Identifier predicates,” on page 6-29.
- Table 6-3, “Object rule predicate operators,” on page 6-31.
- Table 6-4, “Object rule predicate keywords,” on page 6-34.
- Table 6-5, “Object rule predicate keywordSets,” on page 6-36.

5. Configure the name of the object rule in the definition for a request or a response policy.

The following sample configuration session accomplishes the following:

- Creates an object rule that if matched can be used with a forward action to allow inbound HTTP requests to the e-commerce/images server group to be load balanced and forwarded to the appropriate image servers
- Creates a second object rule that if matched can be used with a forward action to forward all remaining HTTP requests to the default server

This configuration example uses the object rule names *matchImages* and *matchAll*, followed by a predicate statement.

CLI Session

```
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 "*"}
```

```
sun(config-vSwitch-e-commerce loadBalance)# show objectRule  
Name: matchAll  
Predicate Field: URI_PATH matches "*" 
```

```
Name: matchImages  
Predicate Field: URI_PATH matches "/images/*" 
```

How to create a request policy

A request policy is a virtual service policy that is evaluated to make a forwarding decision to a service group or respond with a sorry if the traffic matches the named object rule.

- If the action is set to forward, all HTTP traffic from the client that matches the object rules will be sent to the service group for load balancing. See Table 6-6 on page 6-37 for more information about forwarding options.
- If the action is set to sorry, all matching traffic can either close the TCP connection, reset the connection (with a TCP reset packet), redirect a request to another URI, or send back a predefined HTML page. See “How to define a sorry data” (page 6-20).

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. The request policy that matches the request and has the highest precedence, which means the precedence field in the request policy is set at the lowest numerical value (default is 1), is selected. When a request is received, the request policies are examined in decreasing-precedence order. The lower the value the earlier the request policy is examined. 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.

Required elements

Each request policy requires the following elements:

- Name
- Object rule - See “How to create an object rule” (page 6-10).
- Action [forward or sorry]
- A named service group when the action is forward; a named sorry data when the action the action is sorry

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters that are available for the RequestPolicy command.

Configuration steps

To define a request policy, perform the following steps:

1. **Select RequestPolicy from the Load Balance menu in the Web interface or use the RequestPolicy command in the CLI.**
2. **Configure a name for the request policy, an action, object rule and service group if the action is forward or named sorry data if the action is sorry.**
3. **Change the default settings of the other optional parameters if appropriate.**
4. **In the VirtualService, configure the name of the request policy with the requestPolicyList.**

The following configuration session creates a request policy named images and binds in to a service group named imageServers.

CLI Session

```
sun(config-vSwitch-e-commerce loadbalance)# requestpolicy images
forward matchImages imageServers
sun(config-vSwitch-e-commerce loadBalance)# show requestpolicy
```

```
Name: images
Action: Forward
Service Group: imageServers
Object Rule: matchImages
Precedence: 1
Persist Type: none
cookiePersist N/A
FieldPrefix N/A
Source Address Mask: 255.255.255.255
Optimize Last Response: enabled
Use Pooled Connections: disabled
First Object Switching: disabled
Oper Status: inactive
```

How to create a proxy IP pool

A proxy IP pool is a group of IP addresses that are owned by the N2000 Series. The proxy IP pool is used by one or more N2000 Series applications, such as client address translation and TCP pooled connections, that need to draw IP addresses from a pool when establishing TCP sessions with servers. Enabling the `usePooledConnections` parameter in the request policy configuration allows a single set of TCP session to the server to carry all client requests. This feature reduces the TCP processing burden on the servers.

The following CLI session creates the proxy IP Pool on the *e-commerce* vSwitch. The example creates the proxy IP pool named *ipPool_1* that uses an IP address range of 10.10.80.1 through 10.10.80.64 on the *e-commerce:default* vRouter, and specifies that the pool operate on HTTP traffic between the Internet and the real servers.

The IP address pool can have up to 64 addresses specified by a combination of contiguous ranges separated by a hyphen character (-), or individual IP addresses separated by a semicolon character (;).

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)#proxyIPPool ipPool_1  
{10.10.80.1.-10.10.80.64} e-commerce:default HTTP
```

```
sun(config-vSwitch-e-commerce loadBalance)# show proxyIPPool  
Name                               IpPool_1  
Proxy IP Address Pool:             10.10.80.1-10.10.80.64  
vRouter:                           e-commerce:default  
Service Type:                       HTTP
```

How to create a request transform

A request transform defines changes to an HTTP header in an HTTP request. The N2000 Series will apply a request transform to an incoming request from a client if the object rules match 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 strength, and customer headers.

Required elements

Each request transform requires the following elements:

- Name
- Object rule - See “How to create an object rule” (page 6-10).
- Optional parameters that describe the transform.

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters that are available for the `RequestTransform` command.

Configuration steps

To configure a request transform, perform the following steps:

- 1. Select Request Transform from the Load Balance menu in the Web interface or use the `RequestTransform` command in the CLI.**
- 2. Enter a request transform name, and the name of an object rule.**
- 3. Change the default settings of the other optional parameters, if appropriate.**
- 4. In the `VirtualService` definition, add the name of the request transform to the `requestTransformList`.**

The following CLI session creates a request transform to add the client source IP address to the HTTP header in all HTTP client traffic forwarded to the service group.

CLI Session

```
sun(config-vSwitch-e-commerce loadbalance)# requestTransform rqt1  
matchall insertSrcIP enabled srcIPHeaderName 198.168.1.0  
sun(config-vSwitch-e-commerce loadbalance)#show requestTransform
```

```
Name:                               rqt1  
Object Rule:                         matchall  
Custom HTTP Header:                 N/A  
Insert SSL Cipher                    disabled  
SSL Cipher Header name:              X-SslClientCipher  
Insert Source IP:                    enabled  
Source IP Header Name:               198.168.1.0  
Delete HTTP Header:                 N/A  
Oper Status:                         inactive
```

How to create a response policy

A response policy consists of an object rule and an action. A response policy determines how the load balancer will process an HTTP response if the response matches the object rule.

If the action is set to `return`, the response is returned to the client. If set to `retry`, the N2000 Series will retry the client request to another server. If set to `sorry`, the action defined in the `sorry` data will be applied.

A response policy is typically used to handle error returns from the servers. If not defined, all server response traffic is returned to the client.

Each response policy has an ordered precedence with respect to the other configured response policies. The response policy that matches the response and has the highest precedence, which means the precedence field in the response policy is set at the lowest numerical value (default is 1) is selected. When a response is received, the response policies are examined in decreasing-precedence order. The first policy with a matching predicate determines how the response will be processed. If no matching policy is found, the object is returned to the client.

Required elements

Each response policy requires the following elements:

- Name
- Action (`return|sorry|retry`)

- `return`: The response is returned to the client.
- `retry`: The client request is sent to another server.
- `sorry`: The action defined in the `sorry` data will be applied.
- Object rule - See “How to create an object rule” (page 6-10).

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters available for the `ResponsePolicy` command.

Configuration steps:

To configure a response policy, perform the following steps:

1. **Select Response Policy from the Load Balance menu in the Web interface or use the `responsePolicy` command in the CLI.**
2. **Enter a response policy name, an action, and an object rule.**
3. **Change the default settings of the other optional parameters if appropriate.**
4. **In the `ServiceGroup` definition add the name of the response policy to the `responsePolicyList`.**

The following CLI session matches all HTTP traffic with a response code greater than or equal to 500, and returns a server error message to the client.

CLI Session

```
sun(config-vSwitch-e-commerce loadbalance)# objectrule
rule5 predicate {RESPONSE_CODE >= 500}
```

```
sun(config-vSwitch-e-commerce loadbalance)# sorrydata so3
page actionstring /ft10/sorry.htm
```

```
sun(config-vSwitch-e-commerce loadbalance)#
responsePolicy rp1 rule5 sorry so3 precedence 1
```

Name	rp1
Action:	sorry
Object Rule:	rule5
Precedence:	1
sorryData:	so3
overrideCookieInsert	disabled

```
Inline Health Check Failure: disabled
Oper status:                 inactive
```

How to create a response transform

A response transform defines changes made to an HTTP header in an HTTP response. These changes will be made to an outgoing response from the server if the object rules match the response received from the server.

The response transform supports server cloaking, which is a removal of server signature from HTTP headers.

Required elements

Each response transform requires the following elements:

- Name
- Object rule - See “How to create an object rule” (page 6-10).
- Optional parameters that describe the transform.

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters available for the `ResponseTransform` command.

Configuration steps

To configure a response transform, perform the following steps:

1. **Select Response Transform from the Load Balance menu in the Web interface or use the Response Transform command in the CLI.**
2. **Enter a response transform name, and an objectRule.**
3. **Change the default settings of the other optional parameters, if appropriate.**
4. **In the ServiceGroup definition, add the name of the response policy to the responseTransformList.**

The following CLI session creates a response transform that will enable server cloaking. The object rule (`or2`) is a default object rule that accepts all traffic. The `deleteHeader` field is set to `Server` to prevent server-specific information from reaching the client. Use this response transform in the service group to cloak all servers from the clients accessing the virtual service.

CLI Session

```
sun(config-vSwitch-e-commerce loadbalance)#  
responseTransform resptrans1 or2 delete HTTP Header:  
Server
```

Name	resptrans1
Object Rule:	or2
Rewrite Redirects:	disabled
Rewrite Port:	all
Custom HTTP Header:	N/A
Delete HTTP Header:	Server
Oper status:	inactive

How to define a sorry data

A sorry data describes the action the load balancer should take when a sorry action is set in a request policy or a response policy.

Required elements

Each sorry data requires the following elements:

- SorryData name

Each sorry data has a unique name that you can associate with a request policy or a response policy. The software supports one sorry data per policy but the same sorry data can be used in more than one policy

- Action (close|reset|redirect|page)
 - `close`: Gracefully ends the TCP connection to the client
 - `reset`: Sends a TCP reset to the client ending the connection
 - `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`
 - `page`: Returns a specified HTML page to the client.
- Action string

The action string is a text string that specified information to return to the client, depending on the action. If the action is `page`, enter an HTML fully qualified path name. If the action is `redirect`, enter a valid URI.

Configuration steps

To configure a sorry data, perform the following steps:

1. Use the `sorryData` command in the CLI or select `SorryData` from the **Load Balance** menu in the Web interface.
2. Enter a name and an action for the sorry data.
3. When you configure a response policy or request policy, enter `sorry` for action and enter the name of the sorry data next to the `sorryData` field.

CLI Session

```
sun(config-vswitch-e-commercenet-loadbalance)# sorrydata so_1 reset
sun(config-vswitch-e-commercenet-loadbalance)# show sorrydata
```

```
Name:          so_1
Action:        reset
```

How to create the virtual service for L5 to L7 load balancing

A virtual service contains the client visible configuration attributes for the server load balancer. The virtual service provides the virtual IP address (VIP) to the load balancer and specifies the applications service type for the service group. A virtual service also defines the vrouter on which the client traffic will be received.

For L5 to L7 load-balancing applications, the virtual service evaluates an incoming HTTP request against the object rules listed in the request policies that you configure with the virtual service command. When a match is found, the request policy has a service group associated with it and the request is forwarded to that service group. The N2000 Series then load balances across the Web servers in the service group.

Required elements

Each virtual service definition requires the following elements:

- Application Service type (HTTP, HTTPS, TDLB, FTPLB, RTSPLB)
- Virtual service name
- IP Address - VIP address and TCP/UDP port
- Request Policy List

Refer to the *Sun N2000 Series Release 2.0 — Command Reference* or the online Help for detailed descriptions of all of the parameters that are available for the `VirtualService` command.

Configuration steps

To configure a virtual service, perform the following steps:

1. **Select Virtual Service from the Load Balance Menu in the Web interface or use the `VirtualService` command in the CLI.**
2. **Configure a Virtual Service Name, Application Service Type, VIP address, request policy**

The application type you select determines the parameters available for the `VirtualService`.

3. **Change the default settings of the other optional parameters if appropriate.**

How to create virtual service groups

Virtual service groups (vsGroups) allow you to combine one or more named virtual services into a single group so that request policies and real services are shared among the virtual services. By creating vsGroups, you can distribute the processing load on N2000 Series systems that use function cards.

The N2000 Series will automatically create a named vsGroup called *default* if you create two or more virtual services that share the same request policy. If your system uses only one function card, then all vsGroups will operate on that function card.

A vsGroup can contain up to 64 named virtual services.

CLI Session

The following configuration 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}
```

Object rule details — predicates

This section provides the HTTP header fields, URI field names, operators, and keywords that you can use in the predicate statement included in an object rule.

HTTP Request and HTTP Response header field names

Table 6-1 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 6-1. HTTP Request and Response header predicates - HTTP fields

Field name	Description
ACCEPT	<p>HTTP Request header; client specifies the content type it can accept in the message body of the HTTP response.</p> <p>Type: string</p> <p>Example: <code>{ACCEPT matches "*"/*"}</code></p> <p>Example: <code>{ACCEPT matches "text/*"}</code></p>
ACCEPT_LANGUAGE	<p>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.</p> <p>Type: string</p> <p>Example: <code>{ACCEPT_LANGUAGE eq "ja-jp"}</code></p>
ACCEPT_ESI (Edge Side Includes)	<p>HTTP Request header; client specifies an Akamai-sourced HTTP request.</p> <p>Type: string</p> <p>Example: <code>{ACCEPT_ESI present}</code></p>

Table 6-1. HTTP Request and Response header predicates - HTTP fields

Field name	Description
CONNECTION	<p>General; supports persistent and non-persistent connections. CONNECTION informs the client whether the server will close a connection after sending a response, or keep the connection persistent.</p> <p>Type: keywordset (See Table 6-5)</p> <p>Example: {CONNECTION contains close}</p> <p>Example: {CONNECTION contains keep-alive}</p>
CONTENT_LENGTH	<p>Allows examination of the size of the message body in bytes.</p> <p>Type: integer</p> <p>Example: {CONTENT_LENGTH < 40000}</p> <p>Note: Valid with HTTP Method of POST. See METHOD.</p>
COOKIE	<p>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.</p> <p>Type: string</p> <p>Example: {COOKIE has "session-id" eq "105"}</p>
HOST	<p>HTTP Request; client includes the host URL of the Web server.</p> <p>Type: string</p> <p>Example: {HOST eq "www.e-commerce.com"}</p> <p>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.</p>

Table 6-1. HTTP Request and Response header predicates - HTTP fields

Field name	Description
HOST_HEADER	<p>HTTP Request; client includes the host URL of the Web server.</p> <p>Type: string</p> <p>Example: <code>{HOST_HEADER eq "www.e-commerce.com" }</code></p>
HOST_HEADER_PORT	<p>HTTP Request; client includes the TCP port that the Web server application protocols should use. TCP port 80 is the expected port for HTTP requests.</p> <p>Type: integer</p> <p>Example: <code>{HOST_HEADER_PORT == 80 }</code></p>
REFERER	<p>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.</p> <p>Type: string</p> <p>Example: <code>{REFERER eq "http://www.e-commerce.com/default/relatedlinks.htm" }</code></p>
TRANSFER_ENCODING	<p>General; indicates the transfer encoding format applied to the HTTP message body.</p> <p>Type: keywordset (See Table 6-5)</p> <p>Example: <code>{TRANSFER_ENCODING contains chunked }</code></p> <p>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.</p> <p>Example: <code>{TRANSFER_ENCODING contains gzip }</code></p> <p>The <code>gzip</code> keyword indicates the message body is compressed and reduces transmission time.</p>

Table 6-1. HTTP Request and Response header predicates - HTTP fields

Field name	Description
METHOD	<p>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.</p> <p>Type: keyword (See Table 6-4)</p> <p>Example: {METHOD is GET}.</p> <p>Methods: GET, HEAD, POST, PUT, DELETE, CONNECT, TRACE, OPTIONS, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK</p>
HTTP_VERSION	<p>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.</p> <p>Type: string</p> <p>Sample HTTP request line: GET / HTTP/1.1 200 OK</p> <p>Example: {HTTP_VERSION eq "HTTP/1.1"}</p>
PORT	<p>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.</p> <p>Type: integer</p> <p>Example: {PORT == 80}</p> <p>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.</p>
UPGRADE	<p>General; client requests and negotiates an HTTP protocol upgrade with the server.</p> <p>Type: string</p> <p>Example: {UPGRADE eq "HTTP/1.1"}</p>

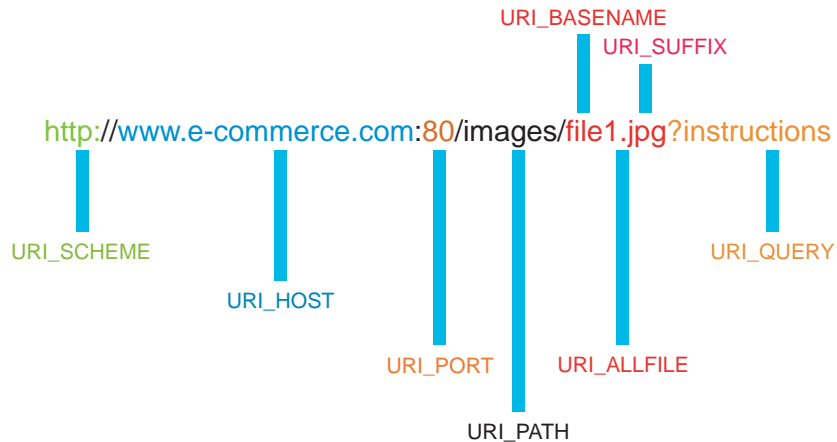
Table 6-1. HTTP Request and Response header predicates - HTTP fields

Field name	Description
USER_AGENT	<p>General; identifies the client or browser implementation of HTTP.</p> <p>Type: string</p> <p>Example: {USER_AGENT matches "*Mozilla/4.0*" }</p>
RANGE	<p>HTTP Request; client requests only partial content instead of the entire resource (in bytes).</p> <p>Type: string</p> <p>Example: Byte range {RANGE eq "bytes 100-200" }</p> <p>Example: Final 200 bytes {RANGE eq "-200" }</p> <p>HTTP Response: 206 Partial Content, 416 Request range not satisfiable</p>
RESPONSE_CODE	<p>HTTP Response; response status codes returned to client.</p> <p>Type: integer</p> <p>Status codes:</p> <ul style="list-style-type: none"> • 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 6-3 illustrates the Uniform Resource Identifier (URI) structure and Table 6-2 lists the supported URI field names.

Figure 6-3. Uniform Resource Identifier structure



Config_29

Table 6-2. HTTP Uniform Resource Identifier predicates

Field name	Description
URI	<p>HTTP Request; specifies the complete Uniform Resource Identifier (URI) string to the Web server resource.</p> <p>Type: string</p> <p>Example: <code>{URI eq "http://www.e-commerce.com:80/images/file.jpg?instructions" }</code></p>
URI_SCHEME	<p>Within URI; identifies the application protocol (HTTP) used to access the Web server(s).</p> <p>Type: string</p> <p>Example: <code>{URI_SCHEME ne "http"} action reset</code></p>
URI_HOST	<p>Within URI; client specifies the host URL of the Web server.</p> <p>Type: string</p> <p>Example: <code>{URI_HOST eq "www.e-commerce.com" }</code></p>
URI_PORT	<p>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.</p> <p>Type: integer</p> <p>Example: <code>{URI_PORT != 80}</code></p>
URI_PATH	<p>Within URI; client specifies the directory path to a resource on the Web server.</p> <p>Type: string</p> <p>Example: <code>{URI_PATH matches "/images/*" }</code></p>

Table 6-2. HTTP Uniform Resource Identifier predicates (continued)

Field name	Description
URI_ALLFILE	<p>Within URI; client specifies the complete resource (basename and suffix) to access on the Web server.</p> <p>Type: string</p> <p>Example: {URI_ALLFILE eq "file1.jpg"}</p>
URI_BASENAME	<p>Within URI; client specifies the basename resource to access on the Web server. The suffix is not specified.</p> <p>Type: string</p> <p>Example: {URI_BASENAME matches "file1"}</p>
URI_SUFFIX	<p>Within URI; client specifies the resource suffix or file extension.</p> <p>Type: string</p> <p>Example: {URI_SUFFIX matches "jpg"}</p>
URI_QUERY	<p>Within URI; client specifies or requests additional information from the server.</p> <p>Type: string</p> <p>Example: {URI_QUERY eq "instructions"}</p>
CLIENT_ADDRESS	<p>N2000 Series initiates the configured action (forward, redirect, or reset) based on the specified client IP address.</p> <p>Type: IP address in dotted-decimal notation (no quotes allowed) and network mask in CIDR or dotted-decimal notation.</p> <p>You may also use a single IP address without a netmask.</p> <p>Example: {(URI_PATH matches "/images/*") and (CLIENT_ADDRESS matches 192.168.124.6/24)}</p>

Using the predicate operators

Table 6-3 lists and describes the operators associated with object rule predicate statements. Within a predicate statement, operators determine how text strings and integers perform with a specified action (forward, redirect, reset).

Table 6-3. 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 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.
? question mark	Operates with the matches predicate operator only. When used with other operators, the question mark 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/1.1" }
==	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 6-3. 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	{ (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 6-3. 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 question mark (?) 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.	<pre>{USER_AGENT matches "*Mozilla/4.0*" }</pre>
contains	keywordSet matching	<pre>{TRANSFER_ENCODING contains chunked }</pre>
is	keyword matching	<pre>{METHOD is POST }</pre>
has	String matching (used with COOKIE field name only)	<pre>{COOKIE has "value" > 300 }</pre>
present	HTTP header exists.	<pre>{ACCEPT_ESI present }</pre>
absent	HTTP header does not exist.	<pre>{ACCEPT_ESI absent } action reset</pre>

Using predicate keywords

Table 6-4 lists and describes the keywords associated with the METHOD object rule predicate statement.

Table 6-4. Object rule predicate keywords

Keyword	Used with; Description	Example
GET	METHOD; client requests a specific resource from the server. Sample request: GET http://www.e-commerce.com/images/file1.jpg	{METHOD is GET }
HEAD	METHOD; 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; client requests that the server 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; client requests that the server pass the message body to the indicated resource. Sample request: POST http://www.e-commerce.com/cgi-bin/file.cgi HTTP/1.1	{METHOD is POST }

Table 6-4. Object rule predicate keywords (continued)

Keyword	Used with; Description	Example
PUT	METHOD; client requests that the server accept the message body as the resource. Sample request: PUT http://www.e-commerce.com/images/file2.jpg	{METHOD is PUT }
DELETE	METHOD; client requests that the server delete the indicated resource. Sample request: DELETE http://www.e-commerce.com/images/file1.jpg	{METHOD is DELETE }
TRACE	METHOD; client requests that the server acknowledge the request only. Sample request: TRACE http://www.e-commerce.com	{METHOD is TRACE }
CONNECT	METHOD; client requests that the server establish a tunnel. Sample request: CONNECT http://www.e-commerce.com/home.htm	{METHOD is CONNECT }
PROPFIND	METHOD; client requests that the server search for object properties.	{METHOD is PROPFIND }
PROPPATCH	METHOD; client requests that the server update the object properties.	{METHOD is PROPPATCH }
MKCOL	METHOD; client requests that the server update the object properties	{METHOD is MKCOL }
COPY	METHOD; client requests that the server copy a file or object to a specified location.	{METHOD is COPY }
MOVE	METHOD; client requests that the server move a file or object to a specified location.	{METHOD is MOVE }

(continued)

Table 6-4. Object rule predicate keywords (continued)

Keyword	Used with; Description	Example
LOCK	METHOD; client requests that the server lock a file or object.	{METHOD is LOCK }
UNLOCK	METHOD; client requests that the server unlock a file or object.	{METHOD is UNLOCK }

Using predicate keywordSets

Table 6-5 lists and describes the keywordSets associated with the specific object rule predicate statements CONNECTION and TRANSFER-ENCODING.

Table 6-5. Object rule predicate keywordSets

Keyword	Used with; Description	Example
keep-alive	CONNECTION; client is informed that the server will keep a persistent connection with the client after sending a response.	{CONNECTION contains keep-alive }
close	CONNECTION; 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; gzip keyword compresses the message body and reduces transmission time.	{TRANSFER_ENCODING contains gzip }

Forwarding actions

A request policy requires one of the following actions:

- Forward
- Sorry

Forward

The forward action passes the HTTP request to a load-balanced server in the service group.

Table 6-6 lists and describes the `firstObjectSwitching` option that you can use to refine how the traffic is forwarded. For more information on using these options, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

Table 6-6. Forwarding option setting

Forwarding option	Description
<code>firstObjectSwitching</code>	<p>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 <code>disabled</code>.</p> <p>Enabling the <code>firstObjectSwitching</code> argument stops the following operations:</p> <ul style="list-style-type: none">• Switch cookie insertion• Custom HTTP header rewriting• SSL cipher additions in the HTTP header• All request transform processing• Response policy and response transform processing

Sorry

The sorry data specifies a sorry action to take and the associated redirect string or HTML path for a page.

Table 6-7. Sorry Settings

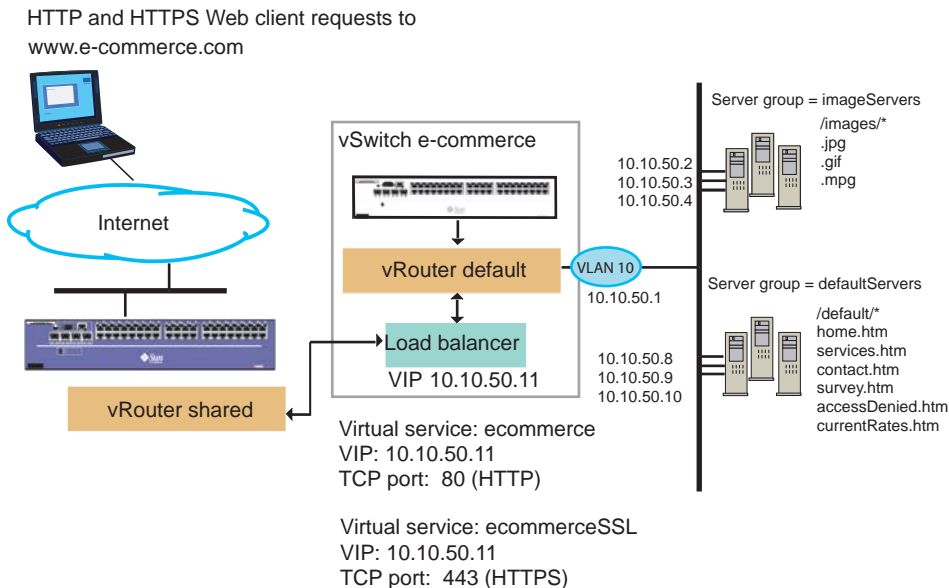
Forwarding option	Description
action	<p>Specifies the action to take when the system has exceeded the number of retries allowed for connection to a different real service within a service group. Possible actions are:</p> <ul style="list-style-type: none"> • <code>close</code>: Gracefully ends the TCP 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. • <code>redirect</code>: 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 <code>sorryString</code> argument • <code>page</code>: Returns an HTML page to the client. You specify the page returned with the <code>sorryString</code> argument. • <code>reset</code>: Sends a TCP reset to the client ending the connection. <p>The default action is <code>close</code>.</p>
actionString	<p>Specifies information to return to the client, depending on the configured page or redirect:</p> <p>If <code>sorryServiceType</code> is <code>page</code>, enter an HTML fully qualified path name.</p> <p>If <code>sorryServiceType</code> is <code>redirect</code>, enter a valid URI.</p>

HTTP load-balancing example

This example shows the CLI sessions for configuring five load-balancing scenarios.

Figure 6-4 illustrates the sample network with the two e-commerce server groups used in the example. Each server group supports specific Web content: images, secured content, and HTML files.

Figure 6-4. Network with two e-commerce service groups



Config_25a

The CLI session shows the current host, realService and serviceGroup configurations for the example followed by the object rules that govern HTTP and HTTPS requests to the e-commerce network. The CLI session also shows the request policies and the response policy that contain each object rule and action. The virtual service definitions follow the policies. Each request policy is linked to a named virtualService definition (*e-commerce1* for HTTP requests, and *e-commercesSSL* for HTTPS requests). The response policy is linked to the named ServiceGroup definition, *defaultServers*.

Table 6-8 describes the request policies and the response policy configured for the example.

Table 6-8. Policies for the HTTP load-balancing example

Policy Name	Description	Object rule	Action	SorryData
rqp1	This request policy matches and forwards all HTTP traffic to the service group named <code>defaultServers</code> , including all HTTP requests with the extension <code>.htm</code> ; files beginning with the string "survey"; and the file named <code>currentRates.htm</code> .	rule1	Forward	None
rqp2	This request policy matches all HTTP traffic from client 192.168.124.6 and returns the "Access Denied" HTML page.	rule2	Sorry	So1
rqp3	This request policy redirects HTTPS log-on and enroll requests to a specific Web page.	rule3	Sorry	So2
rqp4	This request policy matches and forwards all media requests (<code>.jpg</code> , <code>.gif</code> , <code>.mpg</code> , etc.) to the service group named <code>imageServers</code> .	rule4	Forward	None
rp1	This response policy matches all HTTP traffic with a response code greater than or equal to 500, and returns a server error message to the client.	rule5	Sorry	So3



Note: CLI `show` command displays are shortened for clarity in this session.

CLI Session

Host definitions

```
sun(config-vSwitch-e-commerce loadBalance)# host host_1 10.10.50.2
... (config-vSwitch-e-commerce2 loadBalance)# host host_2 10.10.50.3
... (config-vSwitch-e-commerce2 loadBalance)# host host_3 10.10.50.4
... (config-vSwitch-e-commerce2 loadBalance)# host host_4 10.10.50.5
... (config-vSwitch-e-commerce2 loadBalance)# host host_7 10.10.50.8
... (config-vSwitch-e-commerce2 loadBalance)# host host_8 10.10.50.9
... (config-vSwitch-e-commerce2 loadBalance)# host host_9 10.10.50.10
```

Real service definitions

```
sun(config-vSwitch-e-commerce loadBalance)# realService rs1 host_1
tcp 80
...vSwitch-e-commerce2 loadBalance)# realService rs2 host_2 tcp 80
...vSwitch-e-commerce2 loadBalance)# realService rs3 host_3 tcp 80
...vSwitch-e-commerce2 loadBalance)# realService rs4 host_4 tcp 443
...vSwitch-e-commerce2 loadBalance)# realService rs7 host_7 tcp 80
...vSwitch-e-commerce2 loadBalance)# realService rs8 host_8 tcp 80
...vSwitch-e-commerce2 loadBalance)# realService rs9 host_9 tcp 80
```

Service group definitions

```
sun(config-vSwitch-e-commerce loadBalance)# serviceGroup
defaultServers roundrobin {rs7 rs8 rs9} responsePolicyList {rp1}

sun(config-vSwitch-e-commerce loadBalance)# serviceGroup imageServers
roundrobin {rs1 rs2 rs3}
```

Object rule definitions

Rule1

```
sun(config-vSwitch-e-commerce loadBalance)# objectRule rule1
predicate {(HOST_HEADER_PORT != 80) or (HTTP_VERSION ne "HTTP/1.1")}
and (URI_PATH matches "/.htm/*" or "(URI_BASENAME eq "survey") or
(URI_ALLFILE eq "currentRates.htm"))}
```

Rule2

```
sun(config-vSwitch-e-commerce loadBalance)# objectRule rule2 predicate
{(URI_PATH matches "/imageServers/*") and (CLIENT_ADDRESS matches
192.168.124.6/24)}
```

Rule3

```
sun(config-vSwitch-e-commerce loadBalance)# objectRule rule3 predicate
{(URI_PATH matches "/cgi-bin/*") and
(URI_QUERY matches "cmd_DisplayLogon=*")}
```

Rule4

```
sun(config-vSwitch-e-commerce loadBalance)# objectRule rule4 predicate
{(URI_PATH matches "/images/*") or (URI_SUFFIX eq "jpg") or
(URI_SUFFIX eq "mpg") or (URI_SUFFIX eq "gif")}
```

Rule5

```
sun(config-vSwitch-e-commerce loadBalance)# objectRule rule5 predicate
{RESPONSE_CODE >=500}
```

sorryData definitions**So1**

```
sun(config-vSwitch-e-commerce loadBalance)# sorrydata so1 page
www.ecommerce.com/default/accessDenied.htm
```

So2

```
sun(config-vSwitch-e-commerce loadBalance)# sorrydata so2 redirect
"https://www.e-commerce.com/logon"
```

So3

```
sun(config-vSwitch-e-commerce loadBalance)# sorrydata so3 page
actionstring /ft10/sorry.html
```

Request and response policy definitions

rqp1

```
sun(config-vSwitch-e-commerce loadBalance)# requestpolicy rqp1 rule1  
forward defaultServers precedence 10
```

rqp2

```
sun(config-vSwitch-e-commerce loadBalance)# requestpolicy rqp2 rule2  
sorry so1
```

rqp3

```
sun(config-vSwitch-e-commerce loadBalance)# requestpolicy rqp3 rule3  
sorry so2 precedence 40
```

rqp4

```
sun(config-vSwitch-e-commerce loadBalance)# requestpolicy rqp4 rule4  
forward imageServers precedence 60
```

rp1

```
sun(config-vSwitch-e-commerce loadBalance)# responsepolicy rp1 rule5  
sorry so3 precedence 100
```

Virtual service definitions

```
sun(config-vSwitch-e-commerce loadBalance)# virtualService HTTP  
e-commerce 10.10.50.11 port 80 requestpolicyList "rqp1, rqp2, rqp4"
```

```
sun(config-vSwitch-e-commerce loadBalance)# virtualService HTTPS  
e-commerceSSL 10.10.50.11 port 443 requestpolicyList "rqp3"
```

Chapter 7. Additional load-balancing applications

Introduction

This chapter covers the following additional load-balancing applications supported by the Sun N2000 Series:

- Layer 4 applications: FTP Load Balancing (FTPLB), IMAP Load Balancing, and SMTP Load Balancing, Transparent Device Load Balancing (TDLB)
- Layer 7 applications: RTSP Load Balancing (RTSPLB)

This chapter also provides information about Bridge Mode Load Balancing.

A health check is available for each application. Refer to Chapter 12, “Configuring server health checks” for more information.

Topics

This chapter includes the following topics:

Topic	Page
Bridge Mode Load Balancing	7-2
Configuration guidelines	7-2
How to configure Bridge Mode Load Balancing	7-3
FTP Load Balancing	7-9
How to configure FTP Load Balancing	7-9
IMAP Load Balancing	7-13

Topic	Page
How to Configure IMAP Load Balancing	7-13
RTSP Load Balancing	7-13
How to configure RTSP Load Balancing	7-13
SMTP Load Balancing	7-15
How to configure SMTP Load Balancing	7-15
Transparent Device Load Balancing	7-15
How to configure Transparent Device Load Balancing	7-15

Bridge Mode Load Balancing

Bridge Mode Load Balancing allows the N2000 Series to load balance traffic between clients and servers residing in the same IP subnet and VLAN without the use of client address translation (CAT). Bridge Mode Load Balancing is available for all load-balancing applications and is configured on a real service basis.

Typically in an N2000 Series topology, the client and the server reside in different subnets. All traffic coming from the client and the server passes through the switch and load balancing policies are applied. If the client and the server reside in the same subnet, traffic from the client will pass through the switch, but traffic from the server will be addressed to the client's IP address and go directly to the client rather than to the switch. Normal load-balancing policies will not work in this scenario.

Enabling Bridge Mode Load Balancing will ensure that traffic from the server destined for the client reaches the N2000 Series load balancer where network address translation (RIP/port to VIP/port) can be performed.

Configuration guidelines

- Use Bridge Mode Load Balancing only when the servers and clients are on the same IP subnet.
- The servers and clients must be physically separated. Any connection between a server and a client is through the N2000 Series acting as an L2 bridge. You can configure this topology using VLANS.
- Physically place the N2000 Series in the L2/L3 network so that it is always in the path of all client-server traffic.

Figure 7-1 shows a sample topology for Bridge Mode Load Balancing

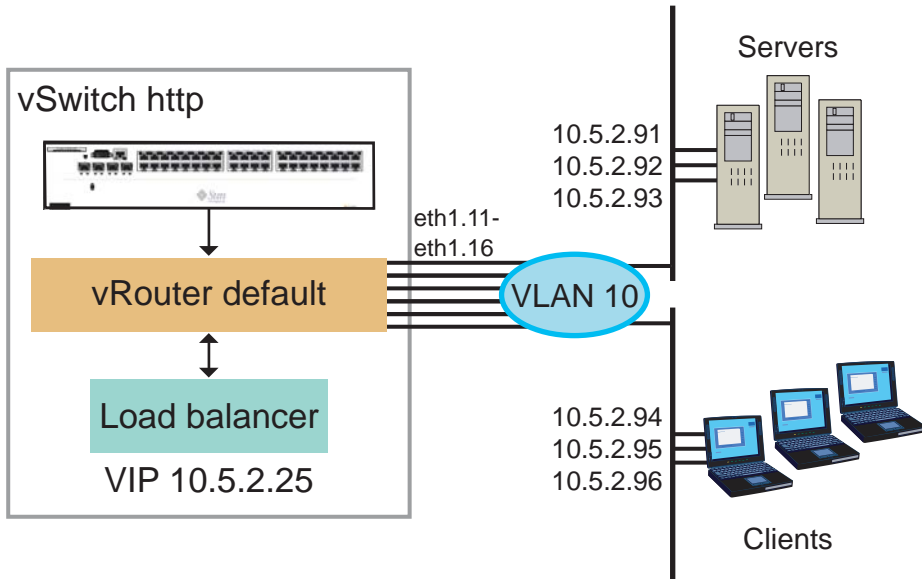
How to configure Bridge Mode Load Balancing

Configure Bridge Mode Load Balancing for each real service. In the `RealService` definition, set the `bridgeMode` setting to **enabled**.

This will ensure that traffic from this real service is available for load balancing. Bridge Mode Load Balancing is disabled by default.

Figure 7-1 shows a sample topology for Bridge Mode Load Balancing. A CLI session enabling Bridge Mode Load Balancing follows the illustration.

Figure 7-1. Bridge Mode Load Balancing configuration



Config_60

CLI Session

1. Create a vSwitch.

```
sun(config)# vswitch http
create new vSwitch "http" ? (y or n) : y
```

2. Create a VLAN with Bridge Mode Load Balancing enabled.

```
sun(config-vSwitch-http vRouter-default)# vlan 10 vlanid2 bridgeMode
Load Balancing enabled
sun(config-vSwitch-http vRouter-default)#show vlan
```

Vlan Name	Vlan ID	Admin Status	Oper Status	Learning	BridgeMode LoadBalancing
10	2	enabled	down	enabled	enabled

3. Create interfaces in the VLAN.

```
sun(config-vSwitch-http vRouter-default)# vlan 10
sun(config-vSwitch-http vRouter-default vlan-10)# interface eth.1.11
sun(config-vSwitch-http vRouter-default vlan-10)# interface eth.1.12
sun(config-vSwitch-http vRouter-default vlan-10)# interface eth.1.13
sun(config-vSwitch-http vRouter-default vlan-10)# interface eth.1.14
sun(config-vSwitch-http vRouter-default vlan-10)# interface eth.1.15
sun(config-vSwitch-http vRouter-default vlan-10)# interface eth.1.16
sun(config-vSwitch-http vRouter-default vlan-10)# exit
```

4. Issue an IP for the VLAN.

```
sun(config-vSwitch-http vRouter-default)# ip
sun(config-vSwitch-http vRouter-default ip)# interface vlan.10
sun(config-vSwitch-http vRouter-default ip)# interface vlan.10
10.5.2.25.255.255.0
```

```
sun(config-vSwitch-http vRouter-default ip) # show address
```

Name	IP Address	Subnet mask	VSRP Redirect	Managed vRouter
vlan.10	10.5.2.25	255.255.255.0	disabled	N/A

```
sun(config-vSwitch-http vRouter-default ip)# end
```

5. Configure hosts in the vSwitch.

```
sun# vswitch http
sun(vSwitch-http)# loadbalance

sun(vSwitch-http loadBalance)# config
sun(config-vSwitch-http loadBalance)# host httpserver1 10.5.2.91
sun(config-vSwitch-http loadBalance)# host httpserver2 10.5.2.92
sun(config-vSwitch-http loadBalance)# host httpserver3 10.5.2.93
sun(config-vSwitch-http loadBalance)# show host
```

Name	IP Address	Admin State	Description	vRouter
httpserver1	10.5.2.91	enabled	N/A	http: default
httpserver2	10.5.2.92	enabled	N/A	http: default
httpserver3	10.5.2.93	enabled	N/A	http: default

6. Create realServices associated with the hosts.

```
sun(config-vSwitch-http loadBalance)# realservice rs1 httpserver1
protocol tcp port 80 weight 1
sun(config-vSwitch-http loadBalance)# realservice rs2 httpserver2
protocol tcp port 80 weight 1
sun(config-vSwitch-http loadBalance)# realservice rs3 httpserver3
protocol tcp port 80 weight 1
sun(config-vSwitch-http loadBalance)# show realService
```

```
Name: rs1
Host Name: httpserver1
Protocol: TCP
Port: 80
Weight In ServiceGroup: 1
Admin State: enabled
Disable Delay: 0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode: disabled
Encryption: unencrypted
Virtual Services:
Oper Status: inactive
```

```
Name: rs2
Host Name: httpserver2
Protocol: TCP
Port: 80
Weight In ServiceGroup: 1
Admin State: enabled
Disable Delay: 0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode: disabled
Encryption: unencrypted
Virtual Services:
```

```

Oper Status:                inactive

Name:                       rs3
Host Name:                  httpserver3
Protocol:                   TCP
Port:                       80
Weight In ServiceGroup:    1
Admin State:                enabled
Disable Delay:              0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode:                disabled
Encryption:                 unencrypted
Virtual Services:
Oper Status:                inactive

```

7. Enable Bridge Mode on the realServices.

```

sun(config-vSwitch-http loadBalance)# realService rs1 bridgeMode
enabled
sun(config-vSwitch-http loadBalance)# realService rs2 bridgeMode
enabled
sun(config-vSwitch-http loadBalance)# realService rs3 bridgeMode
enabled
sun(config-vSwitch-http loadBalance)# show realService
Name:                       rs1
Host Name:                  httpserver1
Protocol:                   TCP
Port:                       80
Weight In ServiceGroup:    1
Admin State:                enabled
Disable Delay:              0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode:                enabled
Encryption:                 unencrypted
Virtual Services:
Oper Status:                inactive

Name:                       rs2
Host Name:                  httpserver2
Protocol:                   TCP
Port:                       80
Weight In ServiceGroup:    1
Admin State:                enabled
Disable Delay:              0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode:                enabled
Encryption:                 unencrypted
Virtual Services:

```

```

Oper Status:                inactive

Name:                       rs3
Host Name:                  httpserver3
Protocol:                   TCP
Port:                       80
Weight In ServiceGroup:    1
Admin State:                enabled
Disable Delay:              0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode:                enabled
Encryption:                 unencrypted
Virtual Services:
Oper Status:                inactive
  
```

8. Create a serviceGroup containing the realServices.

```

sun(config-vSwitch-http loadBalance)# serviceGroup httpservers
weightedhash {rs1 rs2 rs3}
sun(config-vSwitch-http loadBalance)# show serviceGroup
Name:                        httpservers
Load Balance Type:          weightedHash
Configured Real Services:   rs1; rs2; rs3
Active Real Services:       N/A
Admin State:                enabled
Virtual Services:
Health Check Name:
In-Line Health Check:      enabled
Failover Retry Count:      1
Oper Status:                inactive
  
```

9. Create a VIP.

```

sun(config-vSwitch-http loadBalance)# virtualService httpNET L4SLB
10.5.2.100 httpservers vrouter http:default
sun(config-vSwitch-http loadBalance)# show virtualService
Name:                        httpNET
Service Type:                L4SLB
IP Address:                  10.5.2.100
Protocol:                    TCP
Port:                        80
Service Group:               httpservers
Admin State:                 enabled
Disable Delay:               0
vRouter:                     http:default
Client Source IP Address Range: 0.0.0.0-255.255.255.255
SYN Rate Limit:              unlimited
Oper Status:                 active
  
```

```

Oper Message:                    Operational; Server Health
Checking not                      configured on all ServiceGroups

Total Real Services:             3
Active Real Services:            3

```

10. Display the realServices and serviceGroup.

```

sun(config-vSwitch-http loadBalance)# show realService
Name:                             rs1
Host Name:                         httpserver1
Protocol:                           TCP
Port:                               80
Weight In ServiceGroup:             1
Admin State:                         enabled
Disable Delay:                       0
inLine SHC Failure Rate Threshold:  1
Client Address Translation:          disabled
Bridge Mode:                         enabled
Encryption:                          unencrypted
Virtual Services:                    httpNET
Oper Status:                         active

Name:                             rs2
Host Name:                         httpserver2
Protocol:                           TCP
Port:                               80
Weight In ServiceGroup:             1
Admin State:                         enabled
Disable Delay:                       0
inLine SHC Failure Rate Threshold:  1
Client Address Translation:          disabled
Bridge Mode:                         enabled
Encryption:                          unencrypted
Virtual Services:                    httpNET
Oper Status:                         active

Name:                             rs3
Host Name:                         httpserver3
Protocol:                           TCP
Port:                               80
Weight In ServiceGroup:             1
Admin State:                         enabled
Disable Delay:                       0
inLine SHC Failure Rate Threshold:  1
Client Address Translation:          disabled
Bridge Mode:                         enabled
Encryption:                          unencrypted
Virtual Services:                    httpNET
Oper Status:                         active

```



```
sun(config-vSwitch-http loadBalance)# show serviceGroup
Name: httpservers
Load Balance Type: weightedHash
Configured Real Services: rs1; rs2; rs3
Active Real Services: rs1; rs2; rs3
Admin State: enabled
Virtual Services: httpNET
Health Check Name:
In-Line Health Check: enabled
Failover Retry Count: 1
Oper Status: active
sun(config-vSwitch-http loadBalance)#
```

FTP Load Balancing

The N2000 Series supports load balancing in an FTP session. When a client sends a request, the switch will forward the request to an FTP server configured in a server group.

The File Transfer Protocol (FTP) uses two types of TCP sessions, one for control information and one for data transfer. These sessions must be jointly load-balanced to the same server. In addition, embedded TCP port information must be translated.

The N2000 Series supports the following features in an FTP Load Balancing session:

- Active FTP (FTP server initiates the data connection)
- Passive FTP (FTP client initiates the data connection)
- Client address translation (CAT)
- Outbound FTP (a server initiates the control connection)

How to configure FTP Load Balancing

To configure the N2000 Series for FTP Load Balancing, follow the procedures for Layer 4 load balancing, see “L4 load-balancing configuration steps” (page 5-4). Use the following settings:

- In the `virtualService`, set the `appServiceType` to `FTPLB`.
- Use the default port range. (The default port range is sufficient for most applications.) Configure the two timer values if appropriate.



Note: The port range is the ephemeral port range for supporting passive mode FTP. Calculate the number of ephemeral ports to support based on the number of connections you set for the server. Normally, the formula is ports = 2x connections. To support 1000 connections, set the port range to span 2000 ports and choose the port range that is convenient based on the other applications that you are supporting on the switch.

CLI Session

The following CLI session creates real services for three hosts, specifies round-robin load balancing and checks the health of each host with a full FTP login and a good response. A user password is defined in the FTP health check.

1. Create three realservices (use the default values except where noted).

```
sun(config-vSwitch-ecommerce loadBalance)# realservice rs1_ftp h1 port
21
sun(config-vSwitch-ecommerce loadBalance)# realservice rs2_ftp h2 port
21
sun(config-vSwitch-ecommerce loadBalance)# realservice rs3_ftp h3 port
21
sun(config-vSwitch-ecommerce loadBalance)# show realService *ftp
```

```
Name:                               rs1_ftp
Host Name:                           h1
Protocol:                             TCP
Port:                                  21
Weight In ServiceGroup:                1
Admin State:                           enabled
Disable delay:                          0
inLine SHC Failure Rate Threshold:     1
Client Address Translation:             disabled
Bridge Mode:                            disabled
Encryption:                             unencrypted
Virtual Services:
Oper Status:                             inactive
```

```
Name:                               rs2_ftp
Host Name:                           h2
Protocol:                             TCP
Port:                                  21
Weight In ServiceGroup:                1
Admin State:                           enabled
Disable delay:                          0
```

```

inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode: disabled
Encryption: unencrypted
Virtual Services:
Oper Status: inactive

Name: rs3_ftp
Host Name: h3
Protocol: TCP
Port: 21
Weight In ServiceGroup: 1
Admin State: enabled
Disable delay: 0
inLine SHC Failure Rate Threshold: 1
Client Address Translation: disabled
Bridge Mode: disabled
Encryption: unencrypted
Virtual Services:
Oper Status: inactive

```

2. Create the virtual service and set the appServiceType to: FTPLB

```

sun(config-vSwitch-ecommerce loadBalance)# virtualService vs_ftp FTPLB
10.10.10.1 serviceGroupName sg_ftp
sun(config-vSwitch-ecommerce loadBalance)#show virtualService *ftp
Name: vs_ftp
Service Type: FTPLB
IP Address: 10.10.10.1
Protocol: TCP
Port: 21
Service group: sg_ftp
Admin State: enabled
Disable Delay 0
vRouter: system:shared
Persist Type none
Source Address Persist Mask: 255.255.255.255
Client source IP Address Range: 0.0.0.0-255.255.255.255
SYN Rate Limit: unlimited
FTP Data Port Range: 10001-12048
FTP Data RX Timer Short Timeout Value:32_seconds
FTP Data RX Timer Long Timeout Value:64_seconds
Oper Status: misConfigured
Oper Message: Service Group not configured

```

3. Create a service group having all three defined realservices as active realservices.

```

sun(config-vSwitch-ecommerce loadBalance)# serviceGroup sg_ftp
roundRobin {rs1_ftp rs2_ftp rs3_ftp} healthName hc_ftp
sun(config-vSwitch-ecommerce loadBalance)# show serviceGroup sg_ftp

```

```

Name                               sg_ftp
Load Balance type                  roundRobin
Configured Real Services          rs1_ftp; rs2_ftp; rs3_ftp
Active Real Services              N/A
Admin State:                      enabled
Virtual Services:                 vs_ftp
Health Check Name:               hc_ftp
In-Line Health Check:            enabled
Failover Retry Count:            1
Oper Status:                      inactive

```

4. Define the healthcheck profile. Use GoodLogin as the response.

```

sun(config-vSwitch-ecommerce loadBalance)#healthCheckProfile hc_ftp
FTP Good LoginResponse userName anonymous userPassword
myname@mydomain.com
sun(config-vSwitch-ecommerce loadBalance)# show virtualService *ftp
Name:                               vs_ftp
Service Type:                       FTPLB
IP Address:                          10.10.10.1
Protocol:                            TCP
Port:                                 21
Service Group:                      sg_ftp
Admin State:                         enabled
Disable Delay:                       0
vRouter:                             system:shared
Persist Type:                        none
Source Address Persist Mask:         255.255.255.255
Client Source IP Address Range:      0.0.0.0-255.255.255.255
SYN Rate Limit:                     unlimited
FTP Data Port Range:                 10001-12048
FTP Data RX Timer Short Timeout Value:32_seconds
FTP Data RX Timer Long Timeout Value:64_seconds
Open Status:                         active
Oper Message:                       Operational
Function Card:                       1
Total Real Services:                 3
Active Real Services:                3

```

5. Verify that the probes have been sent and the backend servers are running.

```
sun(config-vSwitch-ecommerce loadBalance)#show serviceGroup slbInfo
advanced counters
```

ServiceGroup	RealService	Probes	Successful	Unsuccessful	Passive
sg_ftp	rs1_ftp	116	116	0	0
sg_ftp	rs2_ftp	116	116	0	0
sg_ftp	rs3_ftp	116	116	0	0

IMAP Load Balancing

Internet Message Access Protocol (IMAP) Load Balancing provides support for load balancing of IMAP traffic. IMAP Load Balancing is configured as a Layer 4 server load balancing application.

How to Configure IMAP Load Balancing

To configure the N2000 Series for IMAP Load Balancing, follow the procedure for Layer 4 load balancing, see “L4 load-balancing configuration steps” (page 5-4). In the `RealService`, set the port to 143.

RTSP Load Balancing

The Real Time Streaming Protocol (RTSP) is a Layer 7 protocol that controls the delivery of streaming data with real-time properties. RTSP consists of a control channel and a data channel. The control channel uses TCP on port 554, the data channel uses UDP.

RTSP Load Balancing provides support for load balancing of streaming media content (video stream, audio stream, text) stored on servers.

How to configure RTSP Load Balancing

To configure the N2000 Series for RTSP Load Balancing, follow the procedures for Layer 7 load balancing, see “L5 to L7 load-balancing configuration steps” (page 6-6). Configure the settings of the following parameters:

- Define a RequestPolicy and set the firstObjectSwitching setting to enabled.
- In the virtualService set the appServiceType to RTSP. Add the name of the request policy to requestPolicyList.



Note: It is not necessary to set the port to the RTSP port 554 because the VirtualService will use the default port 80.

CLI Session

```

sun(config-vswitch-ecommerce loadbalance)#host name hA ip Address
10.10.10.10 vRouter tst:default
sun(config-vswitch-ecommerce loadbalance)#host name hB ip Address
10.10.10.11 vRouter tst:default
sun(config-vswitch-ecommerce loadbalance)#host name hC ip Address
10.10.10.12 vRouter tst:default
sun(config-vswitch-ecommerce loadbalance)#realService name rA hostName
hA port 554
sun(config-vswitch-ecommerce loadbalance)#realService name rB hostName
hB port 554
sun(config-vswitch-ecommerce loadbalance)#realService name rC hostName
hC port 554
sun(config-vswitch-ecommerce loadbalance)#healthCheckProfile name hcSM
type RTSP
sun(config-vswitch-ecommerce loadbalance)#ServiceGroup name sgSM
loadBalanceType roundrobin cfg RealSevices {rA,rB,rC} healthName hcSM
sun(config-vswitch-ecommerce loadbalance)#requestPolicy name fpSM
serviceGroupName sgSM persistType srcAddress
sun(config-vswitch-ecommerce loadbalance)# virtualService name vsSM
ipAddress 212.2.2.2 appServiceType RTSP requestPolicyList fpSM
protocol TCP port 554

```

SMTP Load Balancing

Simple Mail Transfer Protocol (SMTP) Load Balancing provides support for load balancing of SMTP traffic.

How to configure SMTP Load Balancing

To configure the N2000 Series for SMTP Load Balancing, follow the procedures for Layer 4 load balancing, see “L4 load-balancing configuration steps” (page 5-4). In the `RealService`, set the port to 25 (the default port for SMTP).

Transparent Device Load Balancing

Transparent Device Load Balancing (TDLB) is non-terminated redirection of traffic to transparent devices such as caches. Network address translation is not applied on the server IP address and port. TDLB is supported by the Media Module NP load balancing engine on all application service types.

The N2000 Series redirects traffic addressed to the addresses specified in the `virtualservice` to selected transparent devices (caches). The N2000 Series also load balances requests from the transparent caches if so configured.

Transparent Device Load Balancing is a higher-precedence function than server load balancing. If a client request is directed to an IP address that is both a server load balance VIP and in the TDLB range, the TDLB is invoked. If the request is from a transparent device or if there are no active transparent devices, L4SLB is performed.

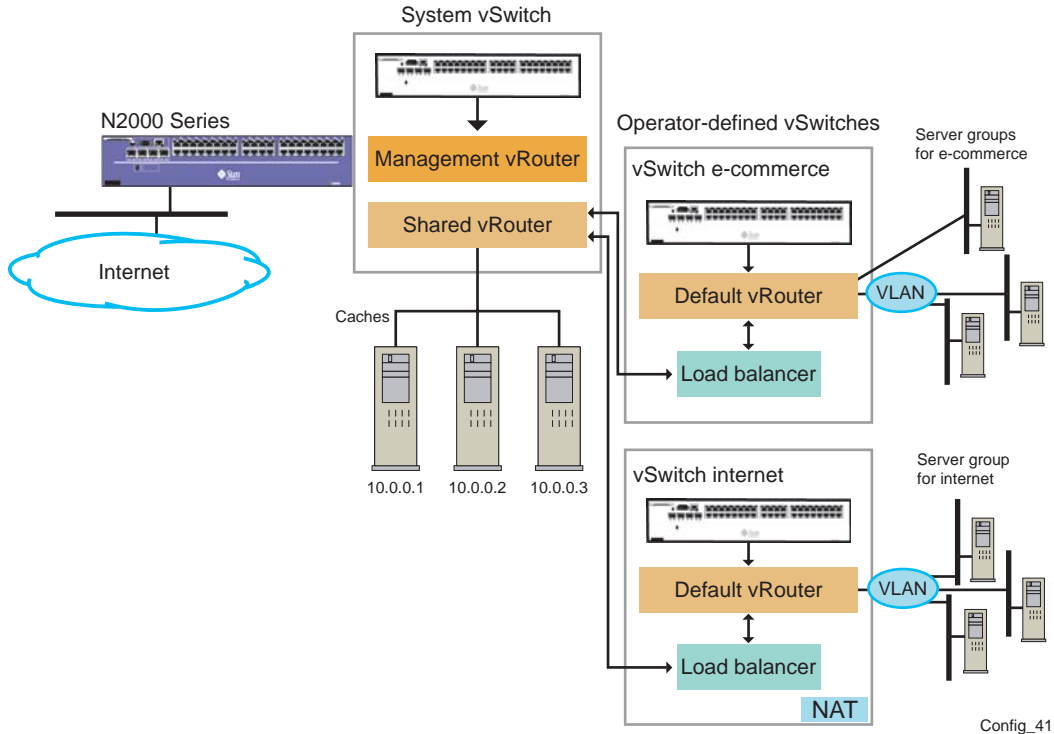
How to configure Transparent Device Load Balancing

Each transparent device is configured as a real service by providing the IP address of each device. To configure TDLB Load Balancing, follow the procedures for Layer 4 load balancing, see “L4 load-balancing configuration steps” (page 5-4). Change the settings of the following parameters:

- Add a host for each transparent device.
- In the `realService` definition, use the default TCP protocol and enter the appropriate port for the service.

- In the `virtualService`, set the `appServiceType` to `TDLB`.

Figure 7-2. Configuration for Transparent Device Load Balancing



Chapter 8. Configuring client address translation

Introduction

This chapter covers N2000 Series client address translation for both terminated and non-terminated load-balancing applications.

Topics

This chapter includes the following topics:

Topic	Page
N2000 Series client address translation overview	8-2
Comparing load balancing with and without client address translation	8-2
Client address and port translations	8-3
Client address translation configuration steps	8-4
How to create the proxy IP pool	8-4
How to enable client address translation	8-5
How to display proxy IP pool statistics	8-5

N2000 Series client address translation overview

The N2000 Series provides server load balancing (SLB) with optional client address translation (CAT). On an inbound request, 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.

Comparing load balancing with and without client address translation

By default, when a client requests a resource from a virtual service, the N2000 Series load-balancing application preserves the client's IP address and TCP port, and forwards the request to the selected server's IP address and TCP port. In the response path, the SLB application replaces the server's IP address and TCP port with the virtual IP (VIP) and virtual service port before forwarding the response to the client. This mode allows the backend servers to see the original client IP addresses.

With CAT enabled, the load-balancing application converts the client address and TCP port to a proxy address and port before forwarding the traffic to the selected server.

Typically, the N2000 Series is configured as the default gateway for each server. Typically, when the server sends traffic back to the client's global IP address, the server will send it to the N2000 Series default gateway. In some networks, the N2000 Series may not be the default gateway for the backend servers. In this case, using CAT ensures that traffic from the servers destined for clients is first sent to the N2000 Series. There are two network topologies where CAT may be useful:

- Where the N2000 Series is installed in front of a server network that already has an existing load-balancing device that is operating as the default gateway
- When the servers are located elsewhere in the Internet, (away from the N2000 Series) with possibly many intervening gateways

CAT uses proxy IP address pools (PIPs) that are created on the vRouter connected to the backend servers. The N2000 Series uses addresses from these pools to replace the client address. The IP addresses in the pool must be "owned" by the N2000 Series; no other node in the vRouter network can use these addresses.

Client address and port translations

Table 8-1 summarizes the IP address and port translations using standard SLB with CAT.

Table 8-1. Client address translation on inbound/outbound SLB traffic

Translation	Addresses as seen on the Internet (On <i>system:shared</i> vRouter)	Addresses as seen on the backend network (On <i>vSwitch:default</i> vRouter)
Without CAT		
Client request	From: Client IP/Client port To: VIP/Virtual Service port	From: Client IP/Client port To: Server IP address/Server port
Server response	From: VIP/Virtual Service port To: Client IP/Client port	From: Server IP address/Server port To: Client IP/Client port
With CAT		
Client request	From: Client IP/Client port To: VIP/Virtual Service port	From: Proxy IP pool/ephemeral port To: Server IP address/Server port
Server response	From: VIP/Virtual Service port To: Client IP/Client port	From: Server IP address/Server port To: Proxy IP pool/ephemeral port

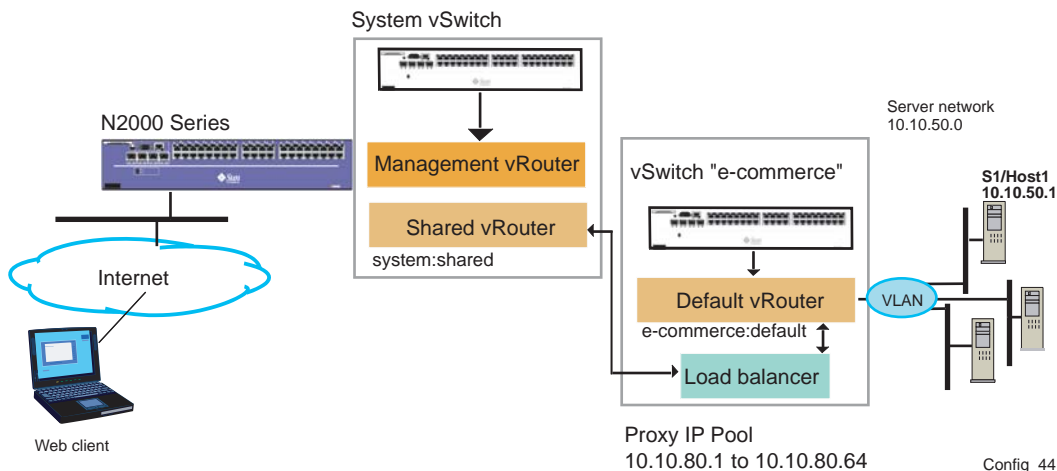
Client address translation configuration steps

To configure client address translation (CAT) on an operator-defined vSwitch, perform the following steps:

1. Create the proxy IP address pool on the vSwitch:default router (or on whatever vRouter supports the backend servers).
2. Enable the `clientAddressTranslation` parameter in the real service configuration specifying the proxy IP address pool to be used.

Figure 8-1 illustrates a sample network for this configuration example.

Figure 8-1. Sample N2000 Series CAT network



How to create the proxy IP pool

The following CLI session creates the proxy IP pool on the *e-commerce* vSwitch. The proxy IP pool is used by one or more N2000 Series applications (such as CAT and TCP pooled connections) that need to draw IP addresses from a pool when establishing TCP sessions with hosts.

The example creates the proxy IP pool named *ipPool_1* that uses an IP address range of 10.10.80.1 through 10.10.80.64 on the *e-commerce:default* vRouter, and specifies that the pool operate on L4SLB traffic between the Internet and the hosts.

The IP address pool can have up to 64 addresses specified by a combination of contiguous ranges separated by a hyphen character (-), and individual IP addresses separated by a semicolon character (;).

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# proxyIPPool ipPool_1
{10.10.80.1-10.10.80.64} e-commerce:default L4SLB
```

```
sun(config-vSwitch-e-commerce loadBalance)# show proxyIPPool
Name: ipPool_1
Proxy IP Address Pool: 10.10.80.1-10.10.80.64
vRouter: e-commerce:default
Service Type: L4SLB
```

How to enable client address translation

The following configuration session enables client address translation on the *e-commerce* vSwitch for the real service named *rs1* and the backend server named *host1*. Client address translation will take place between the Internet client and *host1*.

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# realService rs1 host1
clientAddressTranslation enabled proxyIpPool ipPool_1
sun(config-vSwitch-e-commerce loadBalance)#
```

How to display proxy IP pool statistics

The following CLI session displays proxy IP pool statistics using the `show proxyIPpool statistics` command. For detailed information on these statistics, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# show proxyIPPool
statistics
Name: ipPool_1
addressesInPool: 64
portsInPool: 64510
portsAllocated: 0
portsReleased: 0
portsInUse: 0
portsAvailable: 64510
portAllocationFailures: 0
```

Chapter 9. Configuring cookie persistence

Introduction

This chapter covers N2000 Series cookie persistence.

A *cookie* is a mechanism that a Web server uses to keep track of client requests (usually Web pages visited by the client). When a client accesses a Web site, the Web server returns a cookie to the client in the HTTP response. Subsequent client requests to that server may include the cookie, which identifies the client to the server, eliminating repeated logins, user identification, as well as information already provided by the client. Cookies can also be used to maintain persistent (or “sticky”) sessions between an HTTP client and server.

A common cookie application is the e-commerce shopping cart. As users shop and add items to the cart, they can choose to “continue shopping” to view additional Web pages for items they may wish to purchase before returning to the shopping cart to check out. Cookies keep the connection persistent until the client chooses to end the session (check out, supply payment information, and receive payment confirmation from the e-commerce Web site).

Topics

This chapter includes the following topics:

Topic	Page
N2000 Series cookie persistence overview	9-2
Creating cookie persistence rules	9-5
cookieName	9-5
cookieDomain	9-5
cookiePath	9-6
cookieExpires	9-6
secure	9-6
Cookie persistence limitations	9-6

N2000 Series cookie persistence overview

When configured, the N2000 Series load balancer inserts a cookie into HTTP server response packets returned to the client. This cookie, (called a switch-managed cookie) identifies the original server to the N2000 Series SLB application. The client stores this cookie and includes it in subsequent HTTP client requests to the same server. When the load balancer receives a request containing this cookie, the load balancer deciphers the cookie, and then uses an object rule to forward the traffic to the indicated Web server.

During the session, HTTP requests use the same cookie, keeping the connection persistent until the client closes the session.

Figure 9-1 illustrates a sample network using a `cookiepersistence` command and a sample object rule configuration session. In this example the following occurs:

1. The load balancer uses the cookie persistence rule (`cp1`) to define the cookie to be inserted into the HTTP response.
2. The HTTP client receives and stores the cookie.
3. The next HTTP request from the client includes the cookie in the URL.

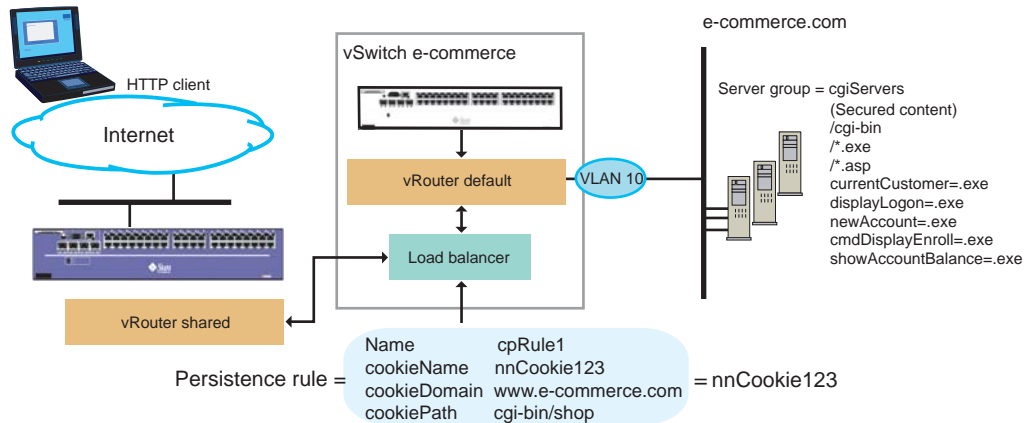
-
4. At the load balancer, a request policy that includes an object rule matches the inbound HTTP request strings (“cgi”, “jsp”, “asp”) applies the cookie persistence name to the forwarding action, and then forwards the request to the destination Web server that originated the cookie.



Note: To enable session persistence, you *must* include the `cookiePersist` forwarding action in the request policy.

Figure 9-1. N2000 Series network with cookie persistence transactions

1. Load balancer inserts cookie using the Set-Cookie header:
Set-Cookie:nnCookie123=0x5FCD285C
in HTTP response to client.
2. HTTP client receives and stores cookie: nnCookie123
3. Subsequent HTTP request includes cookie:
www.e-commerce.com/cgi-bin/shop/additem=3nnCookie123



4. Object rule and request policy to match and forward HTTP client requests;
persistence rule (cpRule1) to keep session persistent based on deciphered cookie.

```
objectRule OR1 {URI_SUFFIX matches "cgi"} or
{URI_SUFFIX matches "asp"} or
{URI_PATH matches "/cgi-bin/shop/*"}

requestpolicy qp1 forward OR1 cgiServers precedence1 cookie persist cpRule1

virtualService appServiceType HTTP forwardingPolicyList fp1
```

Config_25b

Creating cookie persistence rules

A named cookie persistence rule describes the elements that the load balancer uses to create a cookie. These elements are:

- `cookieName`
- `cookieDomain` (optional)
- `cookiePath` (optional)
- `cookieExpires` (optional)
- `secure` (optional)

cookieName

The `cookieName` is the actual string that the load balancer inserts into the HTTP response packet header. When you create a real service, the switch generates a unique 32-bit hash key based on the real service name. The load balancer inserts the hash key into the `cookieName` field to identify the client session, in the following format:

cookieNamecookieDomaincookiePath

The entire string becomes the cookie persistence rule for forwarding traffic to a host.

The default `cookieName` is `nnSessionID` and the value is a hexadecimal number. For example:

Set-Cookie: `nnSessionID=0x123456F`

The `cookieDomain` and `cookiePath` values are optional. If specified, the load balancer adds these fields to the `cookieName` to produce the full cookie string.

cookieDomain

The `cookieDomain` is an optional string for matching a fully qualified domain name (FQDN) using two period characters (.) in the string, such as `www.e-commerce.com`.

If you do not specify a `cookieDomain`, the load balancer inserts the host name of the server that generated the response.

cookiePath

The `cookiePath` is an optional string for matching a URL path. If you do not specify a path, the load balancer inserts the path of the header in the URL request.

cookieExpires

The `cookieExpires` string specifies either an exact date and time, or a dynamic time period when a cookie expires. If expired, the client no longer includes the cookie during subsequent requests to the server that originated the cookie.

If you do not specify the `cookieExpires` string, the cookie expires when the client terminates the session.

For more information on this setting, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

secure

The `secure` keyword attribute protects the confidentiality and authenticity of the cookie when the switch issues switch-managed Set-cookie response headers. When the `secure` option is set to true, 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.

Cookie persistence limitations

Cookie persistence operates with the following limitations:

- Each `virtualService` definition supports up to six unique cookie persistence definitions.
- Each unique cookie persistence rule name counts as one of the six cookies in the `virtualService`.
- Each cookie persistence rule that has a unique `cookieName` counts as one of the six cookies in the `virtualService`.

- If more than one request policy uses cookie persistence, one of two limitations exists: 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.

Chapter 10. Configuring network address translation

Introduction

This chapter covers N2000 Series outbound static and dynamic network address translation (NAT).

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. Each private IP address translates to a different public address (static NAT), or multiple private addresses translate to a single public address (dynamic NAT).

Topics

This chapter includes the following topics:

Topic	Page
N2000 Series outbound NAT overview	10-2
Static NAT	10-3
Dynamic NAT	10-3
Configuring static and dynamic NAT	10-3
How to configure static NAT	10-5
How to configure dynamic NAT	10-6
How to display NAT statistics	10-7

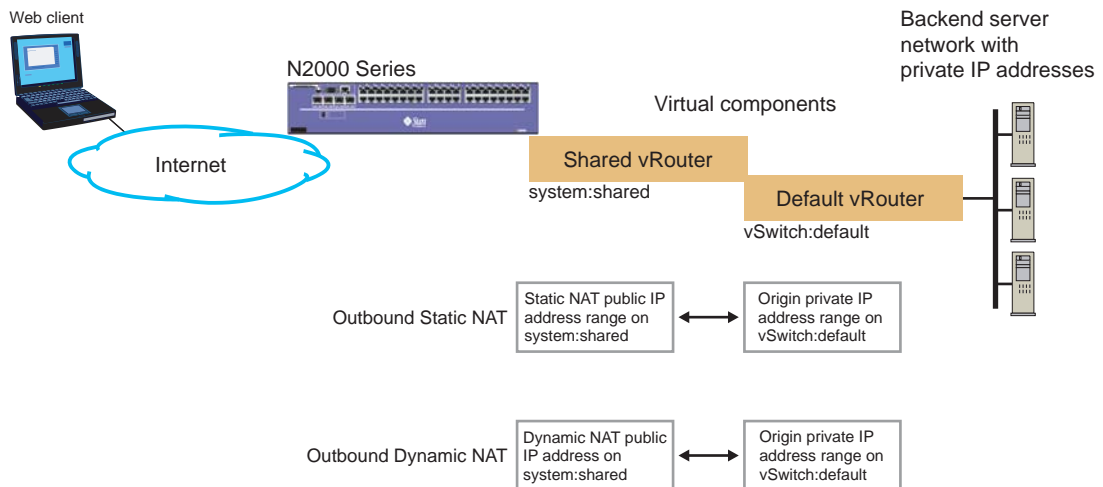
N2000 Series outbound NAT overview

Static NAT and dynamic NAT allow backend servers to access the Internet (acting as clients) while not exposing the backend server private IP addressing information to public Internet. With the N2000 Series architecture, backend server networks are typically served by a different vRouter than the Internet, which isolates them from Internet-based attacks. Backend servers use their private IP addresses for the traffic they generate. Global public addresses must be used for external traffic that is transmitted to the Internet. NAT between the private and public addresses enables backend-to-Internet communication.

NAT modifies the source or destination address in the IP header (and the affected checksums) of both inbound and outbound packets on the network. NAT may also modify the IP addresses that are carried in the payload of the packet. For example, if an Internet Control Message Protocol (ICMP) error message is received, the NAT function will translate embedded IP addresses in addition to the IP addresses in the IP packet header before transmitting the packet. Other protocols, such as FTP (active mode), also carry IP addresses within the payloads of their packets. These protocols require an additional translation function provided by the N2000 Series.

Figure 10-1 illustrates a sample N2000 Series network that shows how the private IP addresses in the backend network are translated to public IP addresses (static NAT) or to a single public IP address (dynamic NAT).

Figure 10-1. N2000 NAT network



Config_45

Static NAT

Static NAT translates individual 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, and can be used in conjunction with Access Control Lists (ACLs) to isolate and protect backend servers that require internet access.

Dynamic NAT

Dynamic NAT translates multiple private addresses to a single public address. Thus, 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 TCP/UDP port in each outbound packet and maintain state information for each TCP/UDP 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 the TCP and UDP protocols.

Configuring static and dynamic NAT

Figure 10-2 illustrates a sample network using static and dynamic NAT.

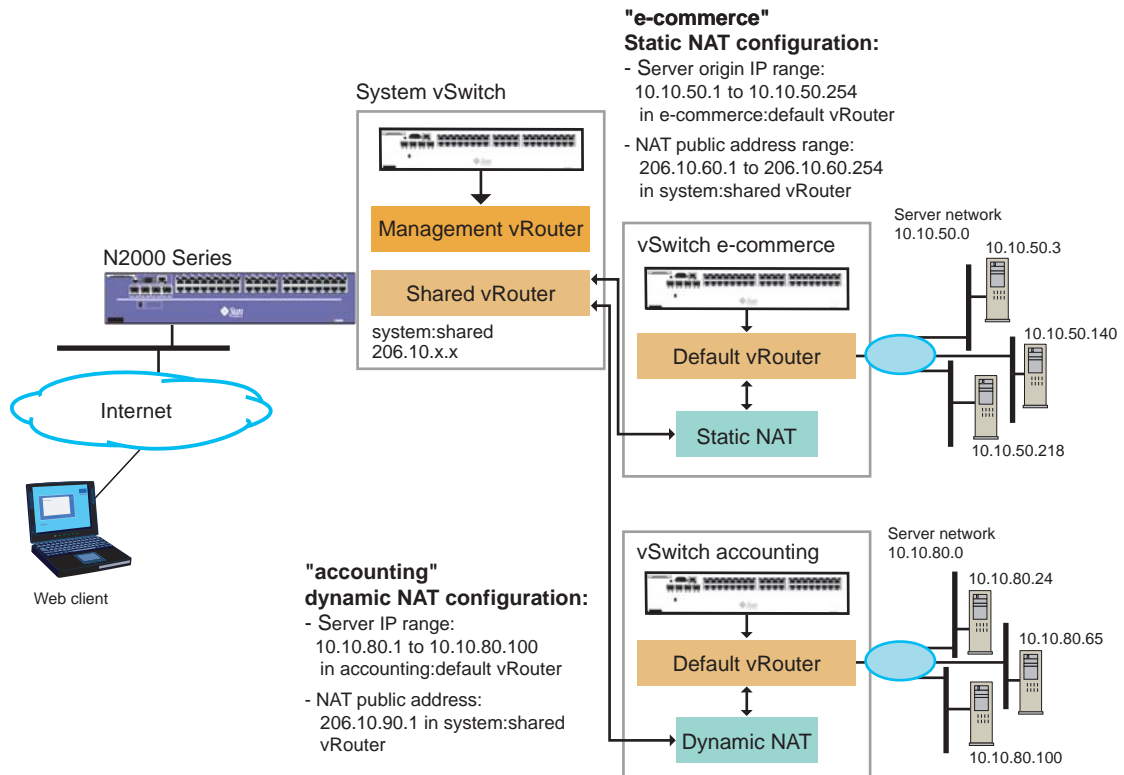
The “e-commerce” vSwitch is using a static NAT configuration where the server IP address range has a corresponding one-to-one NAT public address range for outbound traffic to the Internet.

The “accounting” vSwitch is using a dynamic NAT configuration where the servers are using a single public address.



Note: The IP addresses used in this chapter are *examples* only and should not be used to configure an actual network.

Figure 10-2. Static and dynamic NAT network



Config_43

How to configure static NAT

The following CLI session creates a static NAT configuration called *e-commerceNat* that performs the following operations:

- Specifies an origin host IP range of 10.10.50.1 to 10.10.50.254 in the *e-commerce:default* vRouter, where this range comprises the server private IP addresses
- Specifies a NAT IP range of 206.10.60.1 to 206.10.60.254 in the *system:shared* vRouter, where this range comprises the public IP addresses that shield the private IP addresses of the Web servers



Note: The number of addresses in the origin host IP range must be equal to the number of addresses in the NAT IP range.

By default, the origin network is the *e-commerce:default* vRouter, while the NAT network is the *system:shared* vRouter.

Origin host and NAT IP ranges must be contiguous ranges. For non-contiguous ranges, you may create multiple unique static NAT configurations.

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# outboundNat static  
e-commerceNat {10.10.50.1-10.10.50.254} {206.10.60.1-206.10.60.254}  
enabled
```

```
sun(config-vSwitch-e-commerce loadBalance)# show outboundNat static  
Name: nat1  
Origin Host IP Address Range: 10.10.50.1-10.10.50.254  
NAT IP Address Range: 206.10.60.1-206.10.60.254\  
Origin Host Vrouter: e-commerce:default  
NAT Vrouter: system:shared  
Admin State: enabled
```

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

How to configure dynamic NAT

The following CLI session creates a dynamic NAT configuration called *nat1* that performs the following operations:

- Specifies the global IP address of 206.10.90.1 in the *system:shared* vRouter that the backend servers will use when sending traffic to the Internet
- Specifies the range of server IP addresses (contiguous 10.10.80.1-10.10.80.100 and one non-contiguous address 10.10.80.200) for which dynamic NAT will be performed

In this example, the origin network is the *accounting:default* vRouter, while the NAT network is the *system:shared* vRouter.

The origin host IP range can be a combination of contiguous IP ranges and non-contiguous IP addresses.

CLI Session

```
sun(config-vSwitch-accounting loadBalance outboundNat)# dynamic nat1  
206.10.90.1 enabled system:shared
```

```
sun(config-vSwitch-accounting loadBalance outboundNat)# show dynamic  
Name: nat1  
NAT IP Address: 206.10.90.1  
vrouter: system:shared  
Admin State: enabled
```

```
sun(config-vSwitch-accounting loadBalance outboundNat) dynamic nat1  
sun(config-vSwitch-accounting loadBalance outboundNat dynamic-nat1)#  
hostIpRange hostvRouter accounting:default hostIPRangeList  
{10.10.80.1-10.10.80.100;10.10.80.200}
```

```
sun(config-vSwitch-e-accounting loadBalance outboundNat)# show dynamic  
hostIPRange  
Name: nat1  
Host Vrouter: accounting:default  
Origin Host IP Address Range: 10.10.80.1-10.10.80.100; 10.10.80.200
```

```
sun(config-vSwitch-accounting loadBalance outboundNat)#
```

How to display NAT statistics

Use the `show statistics` command to display information about NAT activities on the current vSwitch. The command displays packet and byte statistics associated with transmitted and received traffic.

CLI Session

```
sun(config-vSwitch-accounting loadBalance outboundNat dynamic-nat1)#  
show statistics
```

```
Name:                               nat1  
Host Bytes Received:                1000  
Host Bytes Transmitted:             2000  
Host Packets Received:              1850  
Host Packets Transmitted:           1850
```

Chapter 11. Configuring CKM and SSL acceleration

Introduction

This chapter covers the N2000 Series Certificate and Key Manager (CKM) configuration and Secure Sockets Layer (SSL) acceleration. SSL requires the optional Server Load Balancing with SSL Function Card (Fx_SSL) on the N2000 Series. You can confirm the installation of this function card by observing the LEDs on the N2000 Series front panel.

SSL acceleration uses the L4SLB_SSL and the HTTPS application types supported in the vSwitch virtual service definition. Both L4SLB_SSL and HTTPS provide SSL decryption at the vSwitch. HTTPS load balances traffic using object switching rules, while L4SLB_SSL load balances traffic using only L4 information.

See Chapters 5 and 6 for information on configuring L4 to L7 load balancing.

Topics

This chapter includes the following topics:

Topic	Page
SSL acceleration overview	11-2
Certificate and key management for SSL	11-3
For new keys and certificates	11-4
For existing keys and certificates	11-4
CKM and SSL configuration steps	11-5

Topic	Page
For new SSL deployments	11-5
For existing SSL deployments	11-6
Generating the SSL key and certificate request	11-6
How to create a cryptographic key pair	11-6
How to create the Certificate Signing Request	11-7
How to issue temporary certificates	11-8
How to submit the certificate request to the CA	11-8
How to install the certificate on the N2000 Series	11-8
Working with existing keys and certificates	11-9
SSL on a vSwitch configuration steps	11-10
How to configure SSL in the serviceGroup definition	11-10
How to configure SSL in the virtualService definition	11-11

SSL acceleration overview

N2000 Series Secure Sockets Layer (SSL) acceleration allows N2000 vSwitches to terminate and decrypt SSL traffic, off-loading SSL processing typically performed by the backend Web servers. There are two primary benefits of this feature:

- The CPU-intensive crypto processing is offloaded from the server freeing up more processing cycles for higher-volume applications.
- The N2000 Series has access to the “clear-text” client request data and so can make intelligent content-based load balancing decisions.

Figure 11-1 illustrates a sample N2000 Series network. The server group named *cgiServers* is load balancing secure requests from HTTP clients on TCP port 443. As an SSL server, the N2000 Series decrypts encrypted requests from the client Web browser before load balancing and forwarding the requests “in the clear” to the destination Web hosts. Optionally, the N2000 Series functions as an SSL client, re-encrypting the traffic enroute to the backend servers.

A separate certificate is required for each operator-defined vSwitch running SSL acceleration, regardless of how many non-SSL servers are being load balanced by the N2000 Series.



Note: The CKM utility is available on the system vSwitch and on all operator-defined vSwitches. On the system vSwitch, use CKM to create key pairs for Secure Shell (SSH) operations using the digital signature algorithm (DSA). On operator-defined vSwitches, use the CKM utility to create key pairs for SSL operations using RSA.

For new keys and certificates

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 certificate, you import it onto the system, overwriting the uncertified public key data with the new certificate data.

X.509 certificate authorities include:

- Verisign (<http://www.verisign.com>)
- Entrust (<http://www.entrust.net/>)
- GlobalSign (<http://www.globalsign.com/>)
- Baltimore Technologies Secure Certificate Management Facility (<http://www.cybertrust.gte.com/>)
- Thawte (<http://www.thawte.com/>)

While the Certificate Authorities require that you supply information on a Web page, you will need to cut and paste a CSR into their request form.

For existing keys and certificates

If you are running an SSL server and you already have keys and certificates from an authorized CA, you can import these keys and certificates to the N2000 Series using the CKM utility and then configure the virtualService to use the CKM key name.

CKM and SSL configuration steps

For new SSL deployments

To configure CKM and SSL on an operator-defined vSwitch, you need to perform the following steps:

1. **Generate an SSL key and certificate request using the Certificate and Key Manager (CKM) utility.**

CKM generates a new cryptographic key pair, mathematically related private and public 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 CA, becomes an X.509 certificate.

2. **Submit the Certificate Signing Request (CSR) to an authorized CA to obtain a valid X.509 certificate.**

An X.509 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.

3. **Install the X.509 certificate from the CA to the load balancer virtualService configuration on the vSwitch.**
4. **Configure the SSL parameters or accept the N2000 Series default settings.**

For existing SSL deployments

To configure CKM and SSL on an operator-defined vSwitch, and if you already have an X.509 certificate, you need to perform the following steps:

1. Use the CKM `export` and `import` commands to move the certificate from one system to another, or for backup purposes.
2. Configure the SSL parameters or accept the N2000 Series default settings.

Generating the SSL key and certificate request

The `generate` and `csr` commands allow you to create a cryptographic key pair and a Certificate Signing Request (CSR) for an N2000 Series vSwitch.

How to create a cryptographic key pair

The `generate` command randomly creates a new key pair bound to the specified key pair name. The keys can be of two types—digital signature algorithm (DSA) for SSH operations or RSA for SSL operations. While both types are public key algorithms used for digital signatures, RSA provides encryption. Because of the complexity of the mathematical operations, the key generation may take several moments.

The following CLI session creates a key pair named *e-commerce2*, and assigns a 1024-bit length to the RSA algorithm.

CLI Session

```
sun(config-vSwitch-e-commerce2 ckm)# generate keyID e-commerce2  
bitlength 1024 algorithm rsa  
...+++
```

In this example, the `keyID` is the name of the vSwitch for which you are accelerating SSL. The `bitlength` argument specifies the length of the key using the values 512, 1024 (default), and 2048, where larger lengths generate slower, but more powerful keys. The `algorithm` argument specifies either DSA or RSA (default). SSL requires RSA.

How to create the Certificate Signing Request

The `csr` command generates the Certificate Signing Request that you send to the Certificate Authority. The data is displayed on 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.

The `csr` command includes a password option for certificate revocation. 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, it is a best practice to fill out all that are appropriate. Check with your CA to verify which fields are expected.

The following CLI session generates a certificate signing request based on the key named `e-commerce2`. The data that is displayed can be copied from the screen and pasted into the requested field at your CA's Web site.

CLI Session

```
sun(config-vSwitch-system ckm)# csr keyId e-commerce2 fqdn
www.e-commerce2.com country US state Massachusetts locality Framingham
orgName "E-Commerce2 Network Solutions" orgUnitName "IT Admin" email
jdoe@e-commerce2.com password certTest makeTestCert true
-----BEGIN CERTIFICATE REQUEST-----
MIICWjCCAcMCAQAwwYQxCzAJBgNVBAYTAlVTMRYwFAYDVQQQIEw1N1YXNzYWNodXN1
dHRzMRMwEQYDVQQHEwpGcmFtaW5naGFTMR0wGAYDVQQKExFOYXV0aWN1cyBOZXR3
b3JrczERMA8GA1UECzMISVQgQWRtaW4xGTAXBgNVBAMTEHd3dy5uYXV0aWN1cy5j
b20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAM3qQrfpe/KQGOKtBAcdgovu
bAWStX0PE9Mar9yWR8qpF5ZG28bh1XDyHPUfr19hxOgw0BmoMtbF221b5GvJbX+2
6P+tEUuuOSUvQiQUatqC/gcOoFpsjOmFuf7XdY8aODPrwfOg+0STMflKez5yIv+a
nTO67YcS6re+tTfuZjirAgMBAAGggZQwFwYJKoZIhvcNAQkHMqoTCGN1cnRUZXN0
MCAGCSqGSIb3DQEJATETFhFqZG91QG5hdXRpY3VzLmNvbTBXBGkqhkiG9w0BCQ4x
```

```
SjBIMBsGA1UdEQQUmBKEHd3dy5uYXV0aWN1cy5jb20wHAYDVR0RBBUwE4ERamRv
ZUBuYXV0aWN1cy5jb20wCwYDVR0PBAQDAgSwMA0GCScqGSIB3DQEBBQUAA4GBAJIq
WtZjnBaOwoVKV2I5WTE6QP/eIuaelmi50gl8m0TgIVnSWyYw8GvX1FJavOHWna78d
eGMSKAo1bX2ntXMdTfPT6Y0WomeAFKkjC3t3m7AtzG6OsfHa+h+e7zciZdN+2CHF
vCu/GMybGHmItliISauBTKbZ6PWfDxTvpXm9ayyq
-----END CERTIFICATE REQUEST-----
sun(config-vSwitch-system ckm)#
```

How to issue temporary certificates

If you set the `makeTestCert` argument to `true`, the N2000 Series issues a temporary, self-signed certificate that you can use to test the configuration and SSL while you are waiting for an approved certificate from the CA.



Caution: Use this option with care, as it could delete any existing certificate material for this key.

How to submit the certificate request to the CA

Using your Web browser, go to your selected CA's Web site and follow the directions on how to submit the request. This will involve copying and pasting information from the CLI (or Web interface) to the CA's Web page where you are supplying requested information.

When you receive the certificate from the CA, you will need to install it on the N2000 Series using the `CKM import` command.

How to install the certificate on the N2000 Series

After you obtain the X.509 certificate from the CA, use the `CKM import paste` command to install the certificate in ASCII format (Private Enhanced Mail format called PEM) to an N2000 Series vSwitch. If you created a temporary certificate, this will overwrite the temporary certificate with the certificate you received from the CA.

The following CLI session overwrites the temporary certificate named *e-commerce2* by replacing it with the X.509 certificate from the CA. Be sure that you paste the `BEGIN CERTIFICATE` followed by the ASCII text, followed by the `END CERTIFICATE` line.

CLI Session

```
sun(config-vSwitch-e-commerce2 ckm)# import paste e-commerce2 pairHalf
certificate format pem
```

```
    Please enter your data ctrl-z to accept ctrl-c to cancel:
-----BEGIN CERTIFICATE -----
BIIcWjCCAcMCAQAwwYQxCzAJBgNVBAYTA1VTMRYwFAYDVQQIEw1NYXNzYWNodXN1
dHRzMRMwEQYDVQQHEwpGcmFtaW5naGFTMR0wGAYDVQQKExFOYXV0aWN1cyBOZXR3
b3JrczERMA8GA1UECXMISVQgQWRtaW4xGTAXBgNVBAMTEHd3dy5uYXV0aWN1cy5j
b20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAM3qQrfpe/KQGOKtBACDgovu
bAWStX0PE9Mar9yWR8qpF5ZG28bh1XDyHPUfr19hxOgw0BmoMtbF221b5GvJBX+2
6P+tEUuuOSUvQiQUatqC/gcOoFpsjOmFuf7Xdy8aODPrwfOg+0STMflKez5yIv+a
nTO67YcS6rE+tTfuZjirAgMBAAGggZQwFwYJKoZIhvcNAQkHMQoTCGNlcnRUZXN0
MCAGCSqGSIB3DQEBJATETfhFqZG9lQG5hdXRpY3VzLmNvbTBXBGkqhkiG9w0BCQ4x
SjBIMBsGA1UdEQQUmBKCEHd3dy5uYXV0aWN1cy5jb20wHAYDVR0RBBUe4ERamRv
ZUBuYXV0aWN1cy5jb20wCwYDVR0PBAQDAgSwMA0GCSqGSIB3DQEBBQUAA4GBAJIG
WtZjnBaOwoVKV2I5WTE6QP/eIuajmi50gl8m0TgIVnSWyYw8GvX1FJavOHWna78d
eGMSKAolbX2ntXmdTfPT6Y0WomeAFknjC3t3m7AtzG6OsfHa+h+e7zciZdN+2CHf
vCu/GMybGHmItliISauBTKbZ6PWFdxTvpXm9ayyq
-----END CERTIFICATE -----
```

Working with existing keys and certificates

You can move an existing key and certificate from one vSwitch to another using the CKM `export` and `import` commands.

The `export` command displays the data contained in a private key or certificate associated with the specified index entry. You can then copy the data and import it 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.



Note: Currently, you should use the `export` command to back up private keys and certificates that are installed on the N2000 Series.

Additionally, if you are using private keys and certificates from an existing SSL server, you can copy them and use one of the CKM `import` commands to install them on one or more N2000 Series systems. For more information on the CKM `import` and `export` commands, as well as other CKM commands, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

SSL on a vSwitch configuration steps

SSL configuration commands are contained within the load balancer on each operator-defined vSwitch.

To configure and complete an SSL load-balanced configuration, you need to perform the following steps:

- 1. Configure the vSwitch hosts, real services, service groups, request policies, and virtual services**
See “Load balancing L4 network traffic” (page 5-1) (L4SLB_SSL) and “Load balancing L5 to L7 network traffic” (page 6-1) (HTTP object rules).
- 2. Configure the SSL settings within the vSwitch serviceGroup configuration.**
- 3. Configure the virtualService definition to include the X.509 certificate that you received from the CA, TCP port 443, and the application service type L4SLB_SSL or HTTPS.**

In addition to the above steps, and in most cases, you can accept the default SSL settings in the serviceGroup definition to complete the SSL configuration.

How to configure SSL in the serviceGroup definition

Within the serviceGroup, you only need to set the `rsEncryption` parameter to `unencrypted`.



Note: Setting the `rsEncryption` parameter to `SSL` allows SSL regeneration (or reencryption) to the backend Web servers.

The following CLI session sets the realService encryption to unencrypted on the *cgiServers* serviceGroup (see Figure 11-1 on page 11-3).

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# serviceGroup cgiServers  
rsEncryption unencrypted loadBalanceType roundRobin {rs4 rs5 rs6}
```

```
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup  
cgiServers
```

```
Name:                               cgiServers  
Load Balance Type:                  roundRobin  
Configured Real Services:           rs4; rs5; rs6  
Active Real Services:               rs4; rs5; rs6  
Admin State:                         enabled  
Health Check Name:                  cgiServersHealth  
Source Address Mask:                255.255.255.255  
Oper Status:                         inactive  
Real Services Encryption:           unencrypted  
Real Services SSL Certificate Name:  N/A  
Real Services SSL Certificate Type:  CA  
Real Services SSL Protocols:        SSLv3; TLSv1  
Real Services SSL Ciphersuites:     RSA_WITH_RC4_128_MD5;  
                                     RSA_WITH_RC4_128_SHA;  
                                     RSA_WITH_3DES_EDE_CBC_SHA  
Real Service SSL Renegotiation Support: true  
Real Service SSL Resumption Support: true
```

How to configure SSL in the virtualService definition

To configure SSL in the virtualService definition, you need to do the following:

1. Specify the X.509 certificate and key name that you received from the CA.
2. Set the TCP port to 443.
3. Set the application type to L4SLB_SSL or HTTPS.

The following CLI session demonstrates this configuration.

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# virtualService e-commerce
protocol TCP port 443 appServiceType HTTPS ckmkeyName e-commerce2
sun(config-vSwitch-e-commerce loadBalance) show virtualService
e-commerce
Name:                               e-commerce
IP Address:                          10.10.50.11
Protocol:                             TCP
Port:                                 443
Service Type:                         HTTPS
Request Policy List:                  logonRedirect; enrollRedirect
Description:
Admin State:                          enabled
VRouter:                              system:shared
Client Source IP Address Range:      0.0.0.0-255.255.255.255
SYN Rate Limit:                       unlimited
Certificate And Key Name:             e-commerce2
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 SGC Support:                      false
SSL Resumption Support:               true
Function Card:                        N/A
Oper Status:                          N/A
Oper Message:                         N/A
```

Chapter 12. Configuring server health checks

Introduction

This chapter covers N2000 Series server health checks. Health checks determine server health by measuring server response time to a specific probe or a client request.

Topics

This chapter includes the following topics:

Topic	Page
Server health checks overview	12-2
Out-of-band server health checks	12-2
In-line server health checks	12-3
Passive health checks	12-4
Server health check considerations	12-4
Configuring server health checks	12-5
How to configure out-of-band server health checks	12-5
How to configure in-line server health checks	12-11
How to configure passive server health checks	12-12

Server health checks overview

The N2000 Series provides out-of-band, in-line, and passive server health checks. By using these health checks, the N2000 Series determines which servers, in a service group, are available for load-balancing and switching applications. If a server is unresponsive, the N2000 Series temporarily removes the server from the service group, leaving the remaining servers in the group to handle the traffic load. When distributing the traffic load, the N2000 Series prefers those servers with faster response times to health check probes. When a subsequent probe determines that a failed server is available, the N2000 Series places the server back into the load-balancing algorithm.

The switch reports server health check probe state changes in the event log. The `show realService slbInfo` command displays basic server health check status and server latency information. The `serviceGroup slbInfo` command displays server health check status by service group and provides more detailed health check statistics.

For more information on the `show realService slbinfo` and the `show serviceGroup slbinfo` commands, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

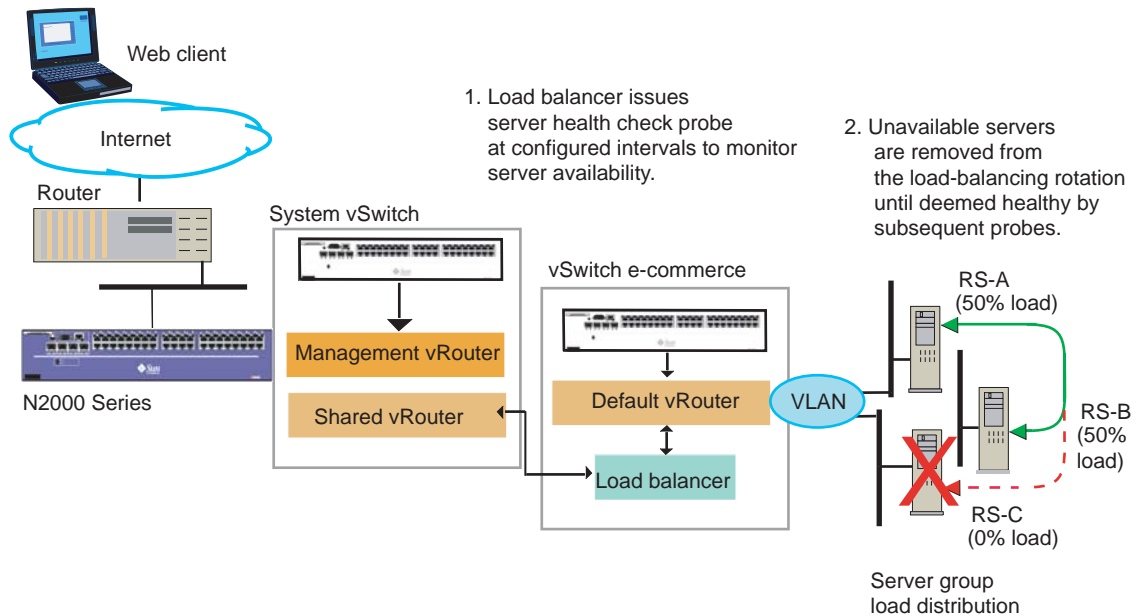
Out-of-band server health checks

Out-of-band health checks generate a specific, scheduled probe to a backend server, then determine server health by measuring the response time to that probe.

The N2000 Series allows you to configure out-of-band health checks that are application-specific or scripted. General configuration options, like intervals, retries, and counts, are available for all probe types. Advanced configuration options are available based on the probe type you select.

Figure 12-1 illustrates the out-of-band server health checks process.

Figure 12-1. N2000 Series out-of-band server health checks



Config_33

In-line server health checks

In-line server health checks monitor server responses to client requests handled through the switch. In-line server health checks are only available for terminated services.

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 removes the server from the service group, leaving the remaining active servers in the group to balance the traffic load.

Determining server health directly by listening to the response from TCP and HTTP client requests prevents multiple connection retries on an unavailable server. Therefore, a server can be deemed unavailable without waiting for the next out-of-band poll to detect the failed server. The N2000 Series places the removed server back into the load-balancing rotation when out-of-band probes determine that the server is once again available.

When using in-line server health checks, you have the option to configure passive server health check probes.

Passive health checks

Passive server health checks monitor the in-line health checks for successful activity; if successful activity is observed through the in-line health checks, the passive health check reports good server health and slows down the normal polling of the out-of-band probes.

Server health check considerations

When configuring server health check profiles, keep the following considerations in mind.

realService weight

When a `realService weight` is set to `dynamic`, the switch uses server health check response time to determine the load balance weights across the servers in the `serviceGroup`. When the `realService weight` is set to a specific value, the switch uses server health check results only to determine whether a server is available.

Server memory and resources

Server health check probes, such as HTTP and LIST, consume more memory on vSwitch and server resources than other probes such as TCP and ICMP. You should consider the server's available resources, as well as the resources that you have allocated on a vSwitch, before running probes that could impact performance.

Additionally, the mode of a health check may require more resources. For example, the `userLogin` mode on an FTP probe or an IMAP probe requires more transactions between the switch and the server. To maximize server resources, determine what level of probing is necessary to satisfy your reliability requirements and use the most efficient probes to meet those requirements.

Configuring server health checks

Out-of-band server health checks require a name, an application-specific probe type, and an associated service group. Each probe type has additional settings you can optionally configure.

In-line server health checks require a name and an associated service group. You must configure additional parameters for an in-line HTTP health check.

How to configure out-of-band server health checks

Perform the following steps to configure out-of-band server health checks on the N2000 Series:

1. **Create a named health check profile.**
2. **Define the health check to be used, including failure threshold settings.**
3. **Apply the health check profile to a service group.**

Naming the profile

Each server health check profile has a unique name that you associate with a serviceGroup. The software supports one health check profile per serviceGroup definition, but a profile can be used in more than one serviceGroup.

Using a specific probe type (and a retry setting), the load balancer runs the profile at the configured interval on the servers in the group. The probe types are HTTP, TCP, LIST (list of HTTP probes), RADIUS, DNS_TCP, DNS_UDP, RAW_TCP, RAW_UDP, FTP, RTSP, POP3, IMAP4, SMTP, and SCRIPT.

The following sample CLI session creates the health check profile named *cgiServersHealthChk* using a TCP probe with a maximum of five retries before declaring a non-responsive server unavailable.

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile
cgiServersHealthChk type TCP retries 5

...vSwitch-e-commerce loadBalance)# show healthCheckProfile
Name:                cgiServersHealthChk
Type:                TCP
Interval:            5
Retries:             5
Success Rate:       0
Timeout:            2
Count:              3
```

Defining the health check profile

A server health check profile contains two elements: a probe type and probe-specific settings.

Choose a probe type, listed in Table 12-1, that provides the information that you need to determine a server's availability in a serviceGroup. This should be based on the networking applications that are running on the server, memory and bandwidth resources, and other server considerations that are important in your data center.

Within the probe types, you can configure a mode and additional settings to determine how the probe verifies the server's availability. For example, if you want to verify server health using a POP3 probe without disclosing a valid user name and password, you can set the `pop3mode` setting to *anyLoginResponse*. Thus, when the server denies a login attempt, the probe accepts this response as valid and considers the server healthy.

Each probe supports optional settings that comprise the health check profile. The following settings are common to all probe types:

- interval (2 to 3600 seconds)
- retries (0 to 32)
- successRate (0 to 100)
- timeout (1 to 10 seconds)
- count (1 to 32)

Table 12-1 lists each probe type and the specific settings that you can modify for that probe type.

Table 12-1. Server health check probes and profile settings

Probe type	Definition	Profile settings
HTTP	Server health is determined by verifying the contents of an HTML page.	<pre>httpMode {checkResponseCodes, Hash, or searchForResponseString} requestString <i>text</i> (in HTTP format) responseString <i>text</i></pre> <p>Example: <pre>loadBalance healthCheckProfile cgiServerHealth type HTTP httpMode CheckResponseCode requestString "GET / cgi-bin/file1.exe\r\nHOST: 10.2.3.5\r\n\r\n" retries 5</pre></p>
TCP	Verifies that the Web server application is running and listening for connections.	<p>Example: <pre>healthcheckProfile name cgiServeHealth2 type TCP retries 5</pre></p>
ICMP	Uses an ICMP echo request (a ping) to verify connectivity and basic functionality.	<p>Example: <pre>healthcheckProfile name cgiServeHealth3 type ICMP retries 5</pre></p>
RADIUS	Verifies remote authentication dial-in user service using RADIUS probes.	<pre>radiusMode {anyAuthResponse, goodAuthResponse, goodActgResponse} userName <i>text</i> secret <i>text</i> optionalRequestAttrs <i>text</i></pre> <p>Example: <pre>healthcheckProfile name cgiServeHealth4 type RADIUS retries 5</pre></p>
DNS_UDP	Domain Name Server verification using a UDP probe.	<pre>dnsMode {anyHostResponse, goodHostResponse, specificHostAddress, anyMailExResponse, goodMailExResponse, specificMailExResponse, verySpecificMailExResponse}</pre> <pre>dnsReqString <i>text</i> dnsRspString <i>text</i></pre> <p>Example: <pre>healthcheckProfile name cgiServeHealth5 type DNS_UDP retries 5</pre></p>

Table 12-1. Server health check probes and profile settings (continued)

Probe type	Definition	Profile settings
DNS_TCP	Domain Name Server verification using a TCP probe.	dnsMode {anyHostResponse, goodHostResponse, specificHostAddress, anyMailExResponse, goodMailExResponse, specificMailExResponse, verySpecificMailExResponse} dnsReqString <i>text</i> dnsRspString <i>text</i> Example: healthcheckProfile name cgiServeHealth6 type DNS_TCP retries 5
RAW_UDP	Raw UDP verification. Specify the exact data stream in hex bytes for the request and response.	rawMode {anyResponse, specificResponse} rawReqString <i>text</i> rawRspString <i>text</i> profileDescription <i>text</i> Example: healthcheckProfile name cgiServeHealth8 type RAW_UDP retries 5
RAW_TCP	Raw TCP verification. Specify the exact data stream in hex bytes for the request and response.	rawMode {anyResponse, specificResponse} rawReqString <i>text</i> rawRspString <i>text</i> profileDescription <i>text</i> Example: healthcheckProfile name cgiServeHealth7 type RAW_TCP retries 5
LIST	A list of up to five HTTP health check probes.	profileList <i>"healthCheck1 healthCheck2 healthCheck3"</i> Example: healthcheckProfile name runHTTPlist profileList "http1 http2 http3" retries 5

Table 12-1. Server health check probes and profile settings (continued)

Probe type	Definition	Profile settings
FTP	Verifies connectivity to an FTP server.	ftpMode {anyServerResponse, goodServerResponse, anyUserResponse, goodUserResponse, anyLoginResponse, goodLoginResponse} ftpTerminationMode {fastDisconnect, quitThenDisconnect} userName <i>text</i> userPassword <i>text</i> Example: healthcheckProfile name cgiServeHealth01 type FTP
RTSP	Probes Real Time Streaming Protocol server availability.	rtspMode {anyOptionResponse, goodOptionResponse, anyDescribeResponse, goodDescribeResponse} rtspUrlName <i>text</i> Example: healthcheckProfile name shc-rtsp type RTSP
POP3	Mail server verification using a POP3 probe.	pop3Mode {anyServerResponse, goodServerResponse, anyUserResponse, goodUserResponse, anyLoginResponse, goodLoginResponse, anyStatResponse, goodStatResponse} pop3TerminationMode {fastDisconnect, quitThenDisconnect} userName <i>text</i> userPassword <i>text</i> Example: healthcheckProfile name healthCheckProfile ss type POP3
IMAP4	Mail server verification using an IMAP4 probe.	imap4Mode {anyServerResponse, goodServerResponse, anyCapabilityResponse, goodCapabilityResponse, anyLoginResponse, goodLoginResponse, anyListResponse, goodListResponse, anyExamineResponse, goodExamineResponse} IMAP4TerminationMode {fastDisconnect, logoutThenDisconnect} userName <i>text</i> userPassword <i>text</i> mboxName <i>text</i> Example: healthcheckProfile name healthCheckProfile sss type IMAP4

Table 12-1. Server health check probes and profile settings (continued)

Probe type	Definition	Profile settings
SMTP	Mail server verification using an SMTP probe.	smtpMode {anyServerResponse, goodServerResponse, anyHelloResponse, goodHelloResponse, anyVerifyResponse, goodVerifyResponse, anySendResponse, goodSendResponse} smtpTerminationMode {fastDisconnect, quitThenDisconnect} srcName <i>text</i> dstName <i>text</i> msgSubject <i>text</i> msgBody <i>text</i> Example: healthcheckProfile name healthCheckProfile shc-smtp type SMTP
SCRIPT	Allows custom health checks through CLI commands or TCL scripts.	script File <i>text</i> script Commands <i>text</i> Example: healthcheckProfile name shc-script type SCRIPT



Note: If you set the probe to LIST, the optional settings (interval, retries, failure, threshold, timeout, and count argument settings) supersede the values defined within an individual health check profile in the profile list.

For more detailed information on the health check profile settings, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

Applying the health check profile to a serviceGroup

The switch allows one server health check profile per serviceGroup definition, but a profile can be used in more than one serviceGroup. If you are probing HTTP servers, use the LIST probe to run multiple HTTP server health checks. The following CLI session adds a server health check profile named *cgiServersHealth* to a serviceGroup definition.

CLI Session

```
sun(config-vSwitch-e-commerce2 loadBalance)# show healthCheckProfile
Name:                cgiServersHealth
Type:                HTTP
Interval:            5
```

```
Retries:          5
Success Rate:    0
Timeout:         2
Count:           3
HTTP Mode:       Hash
Request String:  "GET /www.e-commerce.com/cgi-bin/file1.exe"
```

```
sun(config-vSwitch-e-commerce2 loadBalance)# serviceGroup cgiServers
healthName cgiServersHealth
```

```
sun(config-vSwitch-e-commerce2 loadBalance)# show serviceGroup
Name:                cgiServers
Load Balance Type:   roundRobin
Configured Real Services:  rs4; rs5; rs6
Active Real Services:  rs4; rs5; rs6
Admin State:         enabled
Virtual Service:     N/A
Health Check Name:   cgiServersHealth
Source Address Mask: 255.255.255.255
In-Line Health Check: enabled
Oper Status:         inactive
```

How to configure in-line server health checks

Perform the following steps to configure in-line server health checks on the N2000 Series:



Note: In-line server health checks are available only if the N2000 Series is running the Service Load Balancing with SSL Function Card.

- 1. Create a named health check profile.**

See “Naming the profile” on page 12-5.

- 2. In the serviceGroup, set the `inLineHealthCheck` setting to `enabled`.**

For HTTP traffic:

- 3. To enable in-line health checks for HTTP requests, set the response policy `inLineHealthCheckFailure` option to `enabled`.**

CLI Session

```
sun(config-vSwitch-e-commerce2 loadBalance)# serviceGroup cgiServers  
inlineHealthCheck enabled
```

```
sun(config-vSwitch-e-commerce2 loadBalance)# show serviceGroup  
cgiServers
```

```
Name:                cgiServers  
Load Balance Type:   roundRobin  
Configured Real Services: RS3; RS4; RS5  
Active Real Services: RS3; RS4; RS5  
Admin State:         enabled  
Virtual Service:     N/A  
Health Check Name:   cgiServersHealth  
In-Line Health Check: enabled  
Failover Retry Count 1  
Oper Status:         inactive
```

Configuring in-line server health checks for HTTP

To configure an in-line server health check., define a response policy and set the parameter: `inlineHealthCheckFailure` to **enabled**. For information about creating a response policy, see [How to create a response policy \(page 6-17\)](#).

If the traffic matches the response policy, and the response policy `inlinehealthcheckfailure` parameter is enabled, an inline health failure is triggered. If no response policies are configured, inline health checking assumes every response is successful.

How to configure passive server health checks

Passive server health checks are only available when the N2000 Series is running a load-balancing function card and in-line health checks are enabled.

By default, passive server health checks are *off*. To enable this feature, set the `SuccessiveCount` argument to a number greater than 0. All servers in the service group must be *up* and reporting good health for the passive health check feature to operate.



Note: Passive server health check settings are not available during the initial creation of the health check profile, you must configure these parameters once a health check profile exists.

CLI Session

The following CLI session displays the passive server health check settings that you can modify.

```
sun(config-vSwitch-e-commerce loadBalance)# healthCheckProfile
health_1 passive ?
  [SuccessiveCount (0..100)] Maximum passive probes allowed in a row,
                             before forcing a normal probe (0-none)
                             (default: 0)
  [ActivityThreshold (6..3600)] The maximum time, in seconds, when an
                                RS's activity reflects good status
                                (default: 15)
  [UpdateTolerance (5..60)] The max age, in seconds, that a sample
                              update (from DB Proc) is valid
                              (default: 10)
  [MinPollCycles (2..1000)] The smallest number of normal probes
                              required before passive probes are used
                              (default: 10)
  [InterProcessTimeout (10..2000)] The amount of time, in milliseconds
                                    to wait for a SHC DB response
                                    (default: 50)
```

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
```

Chapter 13. Configuring SSL regeneration

Introduction

This chapter covers the N2000 Series Secure Sockets Layer (SSL) regeneration (reencryption) to the backend servers. Like SSL acceleration, SSL regeneration uses the Server Load Balancing with SSL Function Card (Fx_SSL), which is preinstalled on the N2000 Series. You can confirm the installation of the function card by observing the LEDs on the N2000 Series front panel.

SSL regeneration can be configured on any terminated load-balanced service except FTP Load Balancing (FTPLB).

Topics

This chapter includes the following topics:

Topic	Page
SSL regeneration overview	13-2
SSL regeneration configuration steps	13-3
How to configure SSL in the serviceGroup definition	13-3

SSL regeneration overview

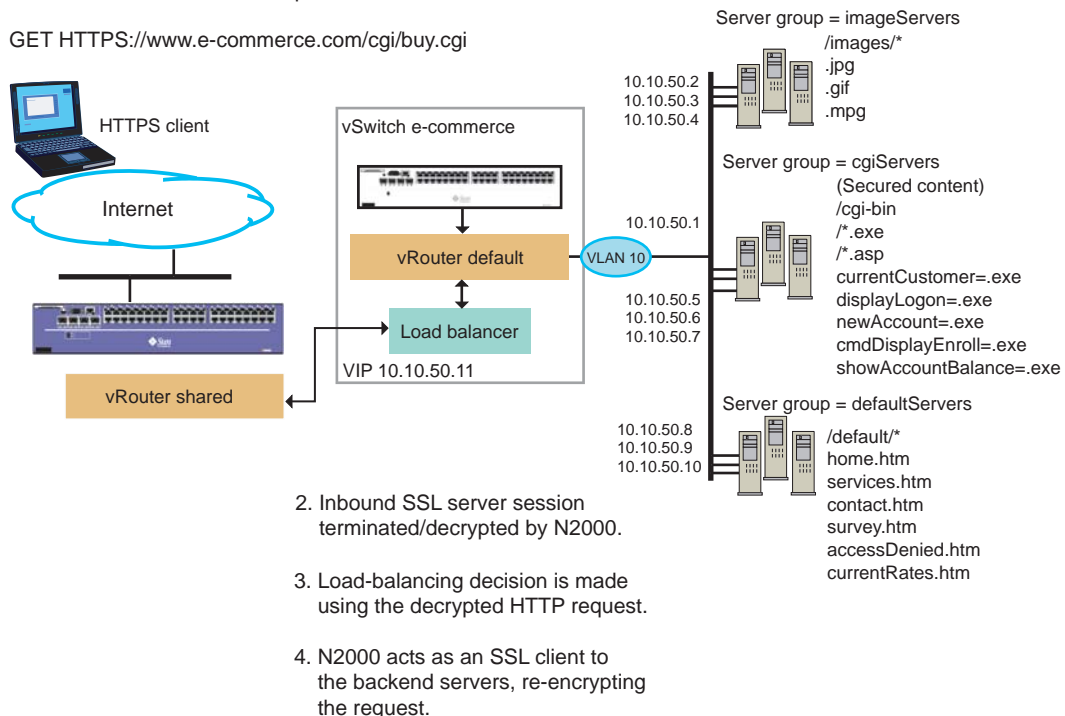
Figure 13-1 illustrates a sample network with the N2000 Series performing as an SSL client. The server group named *cgiServers* is load balancing HTTPS requests from HTTPS clients on TCP port 443. These requests, when processed by the N2000 Series, are first decrypted, and then reencrypted for transmission to the backend servers.

This provides end-to-end security while still allowing the N2000 Series to operate on the clear-text request and response stream.

Figure 13-1. N2000 Series SSL regeneration to backend servers

1. Web client initiates HTTPS request:

GET HTTPS://www.e-commerce.com/cgi/buy.cgi



2. Inbound SSL server session terminated/decrypted by N2000.
3. Load-balancing decision is made using the decrypted HTTP request.
4. N2000 acts as an SSL client to the backend servers, re-encrypting the request.

Config_34B

SSL regeneration configuration steps

SSL configuration commands are contained within the load balancer on each operator-defined vSwitch.

To configure and complete a load-balanced SSL regeneration configuration, you need to perform the following steps:

1. **Configure the vSwitch hosts, real services, service groups, request policies, and virtual services.**
See Chapter 5 and Chapter 6 for more information.
2. **In the vSwitch serviceGroup configuration, perform the following actions:**
 - **Set the `rsEncryption` parameter to SSL.**
 - **Set the name and type of certificate expected from the real service.**

In addition to the above steps, and in most cases, you can accept the default SSL settings in the serviceGroup definition to complete the SSL configuration.

How to configure SSL in the serviceGroup definition

Within the serviceGroup, you only need to set the `rsEncryption` parameter to `SSL` and set the `rsCertName` to the CKM key name where the backend certificate resides. Set the `rsCertType` to `CA` or `literal`, depending on how you set up the backend certificates.

For information on certificates as they relate to SSL, refer to Chapter 11. For detailed information on exporting and importing certificates, refer to the *Sun N2000 Series Release 2.0 – System Administration Guide*.

The following CLI session sets the real service encryption to SSL on the *cgiServers* serviceGroup (see Figure 13-1 on page 13-2).

CLI Session

```
sun(config-vSwitch-e-commerce loadBalance)# serviceGroup cgiServers  
rsEncryption SSL loadbalanceType roundRobin {rs4 rs5 rs6}
```

```
sun(config-vSwitch-e-commerce loadBalance)# serviceGroup cgiServers  
rsCertName e-commerceNet rsCertType CA
```

```
sun(config-vSwitch-e-commerce loadBalance)# show serviceGroup  
cgiServers
```

```
Name:                               cgiServers  
Load Balance Type:                  roundRobin  
Configured Real Services:           rs4; rs5; rs6  
Active Real Services:               rs4; rs5; rs6  
Admin State:                        enabled  
Health Check Name:                  cgiServersHealth  
Source Address Mask:                255.255.255.255  
Oper Status:                        inactive  
Real Services Encryption:          SSL  
Real Services SSL Certificate Name: e-commerce2  
Real Services SSL Certificate Type: CA  
Real Services SSL Protocols:        SSLv3; TLSv1  
Real Services SSL Ciphersuites:     RSA_WITH_RC4_128_MD5;  
                                     RSA_WITH_RC4_128_SHA;  
                                     RSA_WITH_3DES_EDE_CBC_SHA  
  
Real Service SSL Renegotiation Support: true  
Real Service SSL Resumption Support:  true
```

Chapter 14. Configuring access control lists and groups

Introduction

This chapter covers N2000 Series access control lists (ACLs). Access control lists provide a security mechanism that permits or denies inbound or outbound traffic on vRouter IP interfaces. An ACL consists of one or more industry-standard rules permitting or denying access based on IP protocol, source and destination IP addresses or subnets, and application ports. Inbound ACL rules permit or deny network access to resources on the backend Web servers, while outbound ACL rules permit or deny traffic from the Web servers to the external network.

Topics

This chapter includes the following topics:

Topic	Page
Access control lists (ACL) configuration steps	14-2
Basic access control list configuration	14-2
How to create and edit the ACL	14-3
Setting ACL rule precedence, actions, and protocols	14-4
Required and optional ACL settings	14-5
How to create the access group	14-6

Access control lists (ACL) configuration steps

Perform the following configuration steps to create access groups and control lists on a vRouter:

1. Determine the vSwitch and vRouter to which ACLs should apply.

You can create ACLs on the system vSwitch (on the shared vRouter), the management vRouter, or on any operator-defined vSwitch (on the default vRouter). Look at the applications you are running on the backend servers to decide what IP protocol traffic should be permitted (forwarded) or denied (blocked).

Your network should already be configured with IP interfaces and IP addresses. (See Chapter 4, “Configuring the L2/L3 network.”)

2. Create the named ACLs.

Each ACL contains the rules that either permit or deny traffic (by IP protocol or protocol number, source and destination IP address, TCP/UDP port, and physical interface). You can configure up to four ACLs per vRouter.

3. Create the access groups to which you want to apply an ACL.

An access group is identified by an N2000 Series physical interface, a named access list, and the inbound or outbound traffic direction to which the access list rules apply on the vRouter IP interfaces.

Basic access control list configuration

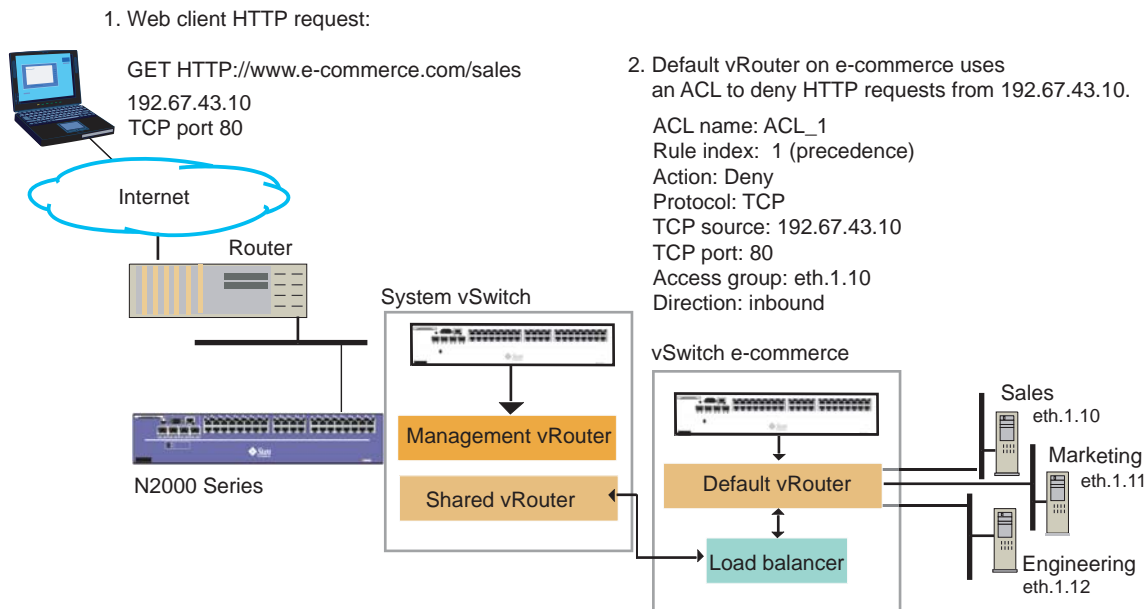
Figure 14-1 illustrates a sample e-commerce network with three Web servers: *Sales*, *Marketing*, and *Engineering*. The following configuration session shows a sample ACL named *ACL_1*. This ACL contains the rule that blocks TCP traffic from 192.67.43.10 TCP port 80 (for HTTP) at the default vRouter on interface eth.1.10 (Sales Web server).

CLI Session

```
sun(config-vSwitch-e-commerce vRouter-default ip)# accesslist ACL_1
rule 1 ruleAction deny ruleProto TCP ruleTcpSrcPort 80 ruleSrcAddr
192.67.43.10
...vRouter-default ip)# accessGroup eth.1.10 in ACL_1
...vRouter-default ip)# show accessList rule ACL_1
```

Summary
 ACL_1.1 deny tcp 192.67.43.10 any http

Figure 14-1. Network using an ACL to block an HTTP request



Config_35

How to create and edit the ACL

When you create an ACL, you can apply the ACL 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 ACL supports up to 256 rules.

To delete an ACL, you must first remove the access group to which the ACL is assigned. You can then delete the disabled ACL from the configuration.

The following CLI session creates the ACL named `ACL_2` and places the CLI in the `ip accessList ACL_2` context.

CLI Session

```
sun(config-vSwitch-e-commerce vRouter-default ip)# accessList ACL_2  
vRouter-default ip accessList-ACL_2)#
```

Setting ACL rule precedence, actions, and protocols

Precedence

When you create a rule for an ACL, you associate a “precedence” with the rule using the `ruleIndex` argument. The list is processed by first match, which means when the system receives a packet, it conceptually applies each rule, in the order of rule index against the packet until it finds a match. The `ruleIndex` argument determines the position of the rule in the list. The lower the value, the earlier the system evaluates it.

It is a good practice 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, the first match, denied ICMP.

By default, the last rule in every ACL is the implicit “deny all” rule. In cases where traffic does not match any of the configured rules, it is handled by the implicit deny, and the traffic is dropped.

Actions

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 action, as this is a required argument. If no rules are configured, all packets are forwarded. If one or more rules are configured, and the packet does not match any rules, the action is `deny`.

Protocols

Table 14-1 lists the IP protocols that you can specify for matching traffic that you declare in an ACL rule. You can use any of these IP protocols with the `ruleProto` argument to define match criteria for permitting or denying traffic. For well-known IP protocols, supply a protocol name or integer. For less-common protocols, use the protocol number, as defined by the Internet Assigned Numbers Authority at the following URL:

<http://www.iana.org/numbers.html>.

Table 14-1. Well-known protocols for ACL traffic 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)

Required and optional ACL settings

All IP protocols, with TCP, UDP and ICMP protocols being the exceptions, use the same `accessList` command settings, as follows:

- All access lists require an ACL name (specified with the `accessList` command), with the optional `adminState` and `description` arguments.
- All access list rules (specified with the `accessList rule` command) require:
 - A `ruleIndex` (rule precedence value of 1 to 4096)
 - A `ruleAction` (permit or deny)

- A ruleProto (the network protocol to which the ACL is applied)

For detailed information on the optional settings associated with access lists including source and destination IP address matching, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

The following CLI session configures the access control list named *ACL_2* with a precedence of *1*, and permits traffic from *any* protocol to be forwarded on an interface (specified with the *accessGroup* command).

CLI Session

```
sun#(config-vSwitch-e-commerce vRouter-default ip accessList-ACL_2)
rule ruleIndex 1 ruleAction permit ruleProto any
```

How to create the access group

Use the *accessGroup* command to associate ACLs with N2000 Series interfaces. For each interface, you can specify two ACLs: one for inbound traffic and one for outbound traffic.

The following CLI session creates two access groups with inbound and outbound ACLs on interface *eth.1.10*. The *show* commands display the configuration before and after creation of the access group.

CLI Session

```
sun(config-vSwitch-e-commerce2 vRouter-default)# show ip accessList
Name                State    Status    Interfaces
ACL_1               enabled  inactive
ACL_2               enabled  inactive
...default)# ip accessGroup eth.1.10 direction in aclListName ACL_1
...default)# ip accessGroup eth.1.10 direction out aclListName ACL_2

sun(config-vSwitch-e-commerce vRouter-default)# show ip accessGroup
Interface    Direction  Name      Status    Hits
eth.1.10     in         ACL_1     active    0
eth.1.10     out        ACL_2     active    0

sun(config-vSwitch-e-commerce vRouter-default)# show ip accessList
Name                State    Status    Interfaces
ACL_1               enabled  active    eth.1.10-in
ACL_2               enabled  active    eth.1.10-out
```

Chapter 15. Configuring redundancy

Introduction

This chapter covers N2000 Series redundancy using the Virtual Router Redundancy Protocol (VRRP), as described in RFC 2338, and the Sun-proprietary Virtual Service Redundancy Protocol (VSRP).

Using two N2000 Series switches, the redundancy protocols provide link-level and service-level failover capabilities between the two switches. VRRP provides continued service to a redundant N2000 Series switch when a link goes down, while VSRP guarantees continued operation for the N2000 Series virtual services during a N2000 system failure. Both VRRP and VSRP use an election protocol to decide which N2000 Series switch becomes the master for the given link or virtual service.

VRRP and VSRP run independently from each other. VRRP may be run separately on any number of IP interfaces. VSRP runs one instance to control all of the virtual services in the system.

Topics

This chapter includes the following topics:

Topic	Page
VRRP overview	15-2
VRRP configuration steps	15-4
How to configure VRRP on redundant switches	15-5
VSRP overview	15-11
VSRP configuration requirements	15-12

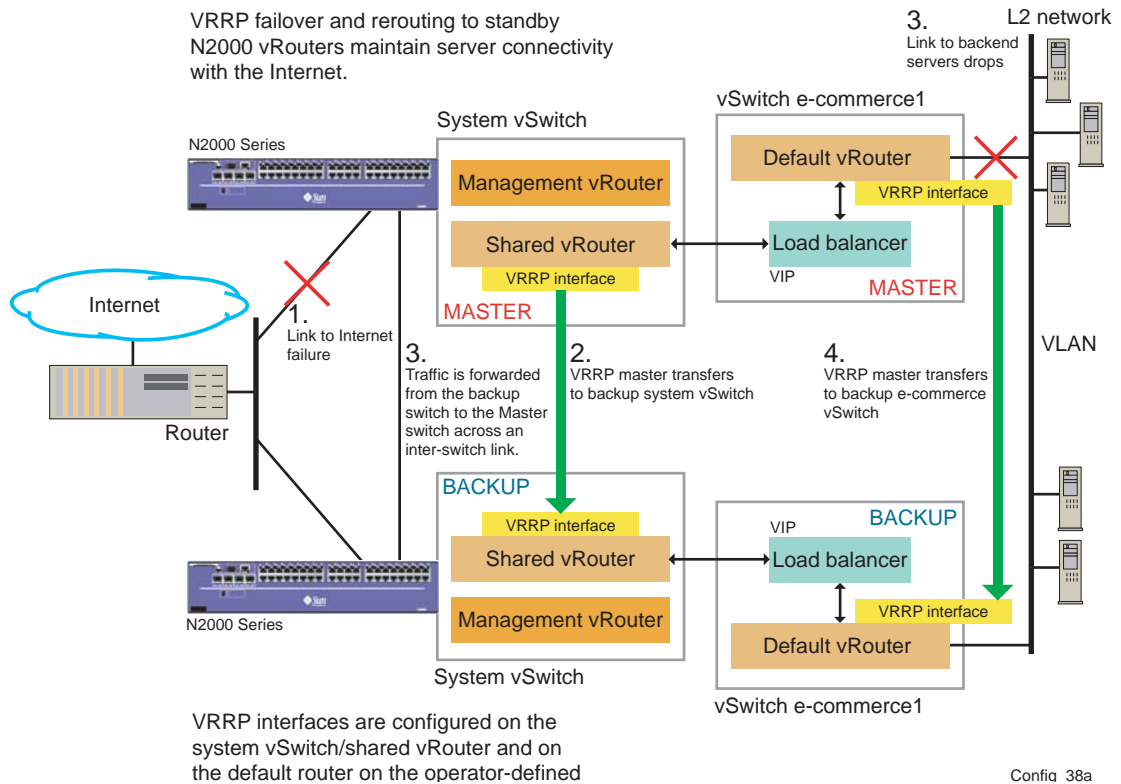
Topic	Page
VSRP configuration steps	15-12
How to configure VSRP peers	15-13
Exporting configurations to redundant switches	15-17
Redundancy configuration steps	15-17
How to export and import redundant configurations	15-17

VRRP overview

Virtual Router Redundancy Protocol (VRRP) is an IETF protocol that runs on N2000 Series vRouter interfaces. Using two physical N2000 Series application switches, full connectivity is maintained between the Internet and the backend servers in the event of a link failure. Configuring VRRP on a vRouter interface creates a VRRP virtual router.

Figure 15-1 illustrates a sample network where VRRP reroutes traffic on failed links on the master VRRP interfaces to the backup VRRP interfaces, forwarding the traffic to the destination servers.

Figure 15-1. N2000 Series VRRP network



VRRP provides access to an active default gateway when router interfaces fail, particularly for the real services running in the backend networks. N2000 Series switches also use VRRP to maintain connectivity of virtual IP addresses (VIPs) to front end networks when router interfaces fail.

VRRP runs on a per-interface basis. When you configure VRRP interfaces, VRRP uses an election protocol that dynamically assigns master and backup status to the vRouters for that network. The vRouter responsible for receiving traffic from that network interface is the master VRRP virtual router. The other vRouter in the configuration functions as a backup VRRP router for that network.

If the VRRP interface of the master VRRP virtual router becomes unavailable due to a link failure, the election process selects the backup VRRP virtual router to become the master, providing dynamic failover for IP traffic forwarding.



Note: VRRP must be configured on the frontend and backend networks for VSRP to work. Running VRRP on the frontend network ensures that a vRouter will always respond to Address Resolution Protocol (ARP) requests for VIPs. Running VRRP on the backend network maintains a default gateway for the servers. Therefore, for full redundancy, configure VRRP on the system vSwitch and on the operator-defined vSwitches (see Figure 15-1).

VRRP configuration steps

Perform the following configuration steps to create a VRRP configuration on N2000 Series vRouters:

1. **Configure the physical N2000 Series interfaces and IP addresses.**
See Chapter 2 and Chapter 4 for more information.

2. **Configure the VRRP interfaces on each N2000 Series.**

This defines the VRRP interface, a VRRP identifier that *must* match a corresponding ID on the VRRP peer, one (or more) virtual IP address(es) to be used by VRRP, and optional VRRP settings. Specify a VRRP identifier of 0 unless you are already running VRRP among other vRouters on this interface (in which case you must specify an identifier not already in use by those VRRP routers). Configuring VRRP on the shared and default vRouter interface creates the VRRP virtual router.

3. **Set the VSRP preference and enable trap generation (optional).**

In most cases, it is desirable to have the switch actively serving VIPs (for example, the VSRP-active switch) to be the master VRRP virtual router. The VSRP preference indicates the incremental increase in VRRP preference that this N2000 Series advertises when it is the VSRP master. This increases the likelihood that the master VSRP switch will also be the VRRP master.

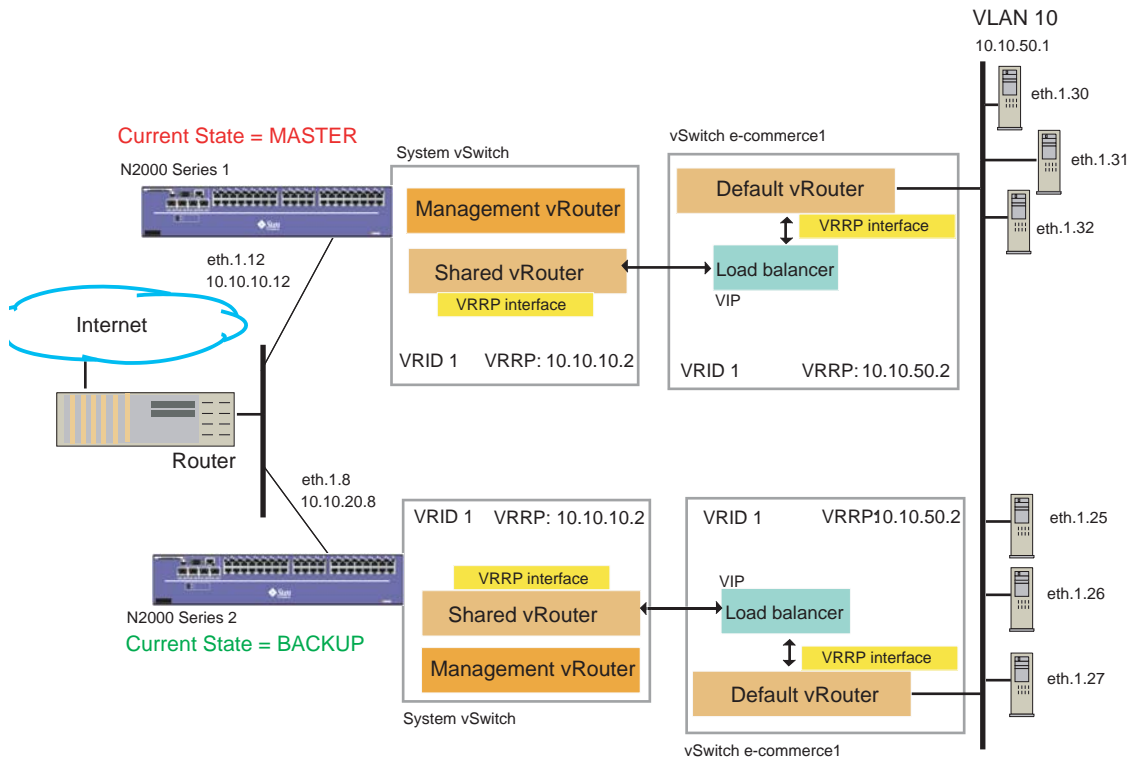
When other VRRP preferences are equal, the VSRP preference causes the VSRP master switch to win VRRP elections. Set the VSRP preference by specifying a value to add to the VRRP priority when the VSRP state is master. A value of 0 indicates that the VSRP state should not influence VRRP elections, and therefore, should not monitor VSRP state. See the section in this chapter covering VSRP for more information.

Set VRRP trap generation to `enabled` to generate traps for VRRP events if failovers occur.

How to configure VRRP on redundant switches

The following CLI session configures VRRP in the sample network illustrated in Figure 15-2. In this network, the backend servers use a single virtual LAN (VLAN 10) and the servers are connected to L2 switching equipment.

Figure 15-2. N2000 Series VRRP network configuration



Config_40

CLI Session on N2000 Series 1

1. Configure the IP interface to the Internet on system vSwitch/shared vRouter.

```

sun(config)# vSwitch system
sun(config-vSwitch-system)# vRouter shared
sun(config-vSwitch-system vRouter-shared)# ip interface eth.1.12
...vRouter-shared)# ip address eth.1.12 10.10.10.12 255.255.255.0
sun(config-vSwitch-system vRouter-shared)# show ip address
If Index      IP Address      Subnet Mask      Redirect Traffic
eth.1.12     10.10.10.12    255.255.255.0   disabled
sun(config-vSwitch-system vRouter-shared)#

```

2. Configure the VRRP interface, VRRP ID, VRRP IP addresses, and optional settings.



Note: The VRID and VRRP IP address (or addresses if more than one address is used) on this vRouter must match the VRID and VRRP IP address(es) on the corresponding redundant vRouter.

```
sun(config-vSwitch-system vRouter-shared)# vrrp
sun(config-vSwitch-system vRouter-shared vrrp)# interface eth.1.12
vrid 1 10.10.10.2
adminState enabled priority 100 interval 5 preempt true
```

```
sun(config-vSwitch-system vRouter-shared vrrp)# show interface
If Index:          eth.1.12
VRID:              1
IP Addresses:      10.10.10.2
Primary IP Address: 10.10.10.12
Admin State:       enabled
Oper State:        master
Priority:           100
Actual Priority:    100
Interval:          5
Preempt Mode:      true
MAC Address:       00:00:5e:00:01:01
IP Address Count:  3
Master's IP Address: 10.10.10.12
Up Time:           3/10/2003-11:45:11
```

3. Set the VSRP preference and enable trap generation.

```
sun(config-vSwitch-system vRouter-shared)# vrrp vsrpPreference 10
sun(config-vSwitch-system vRouter-shared)# vrrp traps enabled
```

4. Configure the VLAN interfaces on the e-commerce1/default vRouter.

```

sun(config-vSwitch-e-commerce1 vRouter-default)# vlan 10
...vRouter-default vlan-10)# interface eth.1.30
...vRouter-default vlan-10)# interface eth.1.31
...vRouter-default vlan-10)# interface eth.1.32
...vRouter-default vlan-10)# show interface

Vlan Id      Port          Interface    Admin State  Oper Status  Tagging
10           eth.1.30     vlan.10.1   up           down enabled
10           eth.1.31     vlan.10.2   up           down enabled
10           eth.1.32     vlan.10.3   up           down enabled

...vRouter-default vlan-10)# exit
sun(config-vSwitch-e-commerce1 vRouter-default) ip
...vRouter-default ip)# interface vlan.10
...vRouter-default ip)# address vlan.10 10.10.50.1 255.255.255.0

```

5. Configure the VRRP interface, VRRP ID, VRRP IP addresses, and optional settings.

```

sun(config-vSwitch-e-commerce1 vRouter-default)# vrrp
sun(config-vSwitch-e-commerce1 vRouter-default vrrp)# interface
vlan.10 vrid 1 10.10.50.2
adminState enabled priority 100 interval 5 preempt true

sun(config-vSwitch-e-commerce1 vRouter-default vrrp)# show interface
If Index:          vlan.10
VRID:              1
IP Addresses:      10.10.50.2
Primary IP Address: 10.10.50.1
Admin State:       enabled
Oper State:        master
Priority:           100
Actual Priority:    100
Interval:          5
Preempt Mode:      true
MAC Address:       00:00:5e:00:01:01
IP Address Count:  3
Master's IP Address: 10.10.50.1
Up Time:           3/10/2003-11:45:11

```

6. Set the VSRP preference and enable trap generation.

```

sun(config-vSwitch-e-commerce1 vRouter-default)# vrrp vsrpPreference
10
sun(config-vSwitch-e-commerce1 vRouter-default)# vrrp traps enabled

```

CLI Session on N2000 Series 2

1. Configure the IP interface to the Internet on system vSwitch/shared vRouter.

```
sun(config)# vSwitch system
sun(config-vSwitch-system)# vRouter shared
sun(config-vSwitch-system vRouter-shared)# ip interface eth.1.8
...vRouter-shared)# ip address eth.1.8 10.10.20.8 255.255.255.0
sun(config-vSwitch-system vRouter-shared)# show ip address
If Index      IP Address      Subnet Mask      Redirect Traffic
eth.1.8       10.10.20.8      255.255.255.0   enabled
```

2. Configure the VRRP interface, VRRP ID, VRRP IP addresses, and optional settings.

```
sun(config-vSwitch-system vRouter-shared)# vrrp
sun(config-vSwitch-system vRouter-shared vrrp)# interface eth.1.8 vrid
1 10.10.10.2 adminState enabled priority 50
interval 5 preempt true
```

```
sun(config-vSwitch-system vRouter-shared vrrp)# show interface
If Index:          eth.1.8
VRID:              1
IP Addresses:      10.10.10.2
Primary IP Address: 10.10.20.8
Admin State:       enabled
Oper State:        backup
Priority:           50
Actual Priority:    50
Interval:          5
Preempt Mode:      true
MAC Address:       00:00:5e:00:01:01
IP Address Count:  3
Master's IP Address: 10.10.10.12
Up Time:           3/10/2003-11:45:11
```

3. Set the VSRP preference and enable trap generation.

```
sun(config-vSwitch-system vRouter-shared)# vrrp vsrpPreference 1
sun(config-vSwitch-system vRouter-shared)# vrrp traps enabled
```

4. Configure the VLAN interfaces on the e-commerce1/default vRouter.

```
sun(config-vSwitch-e-commerce1 vRouter-default)# vlan 10
...vRouter-default vlan-10)# interface eth.1.25
...vRouter-default vlan-10)# interface eth.1.26
...vRouter-default vlan-10)# interface eth.1.27
...vRouter-default vlan-10)# show interface
```

Vlan Id	Port	Interface	Admin State	Oper Status	Tagging
10	eth.1.25	vlan.10.1	up	down	enabled
10	eth.1.26	vlan.10.2	up	down	enabled
10	eth.1.27	vlan.10.3	up	down	enabled

```
...vRouter-default vlan-10)# exit
sun(config-vSwitch-e-commerce1 vRouter-default) ip
...vRouter-default ip)# interface vlan.10
...vRouter-default ip)# address vlan.10 10.10.50.1 255.255.255.0
```

5. Configure the VRRP interface, VRRP ID, VRRP IP addresses, and options.

```
sun(config-vSwitch-e-commerce1 vRouter-default)# vrrp
sun(config-vSwitch-e-commerce1 vRouter-default vrrp)# interface
vlan.10 vrid 1 10.10.50.2 adminState enabled priority 50 interval 5
preempt true
```

```
sun(config-vSwitch-e-commerce1 vRouter-default)# show interface
If Index:          vlan.10
VRID:              2
IP Addresses:     10.10.50.2
Primary IP Address: 10.10.50.1
Admin State:      enabled
Oper State:       backup
Priority:          50
Actual Priority:   50
Interval:         5
Preempt Mode:     true
MAC Address:      00:00:5e:00:01:01
IP Address Count: 3
Master's IP Address: 10.10.50.1
Up Time:          3/10/2003-11:45:11
```

6. Set the VSRP preference and enable trap generation.

```
sun(config-vSwitch-e-commerce1 vRouter-default)# vrrp vsrpPreference 1
sun(config-vSwitch-e-commerce1 vRouter-default)# vrrp traps enabled
```

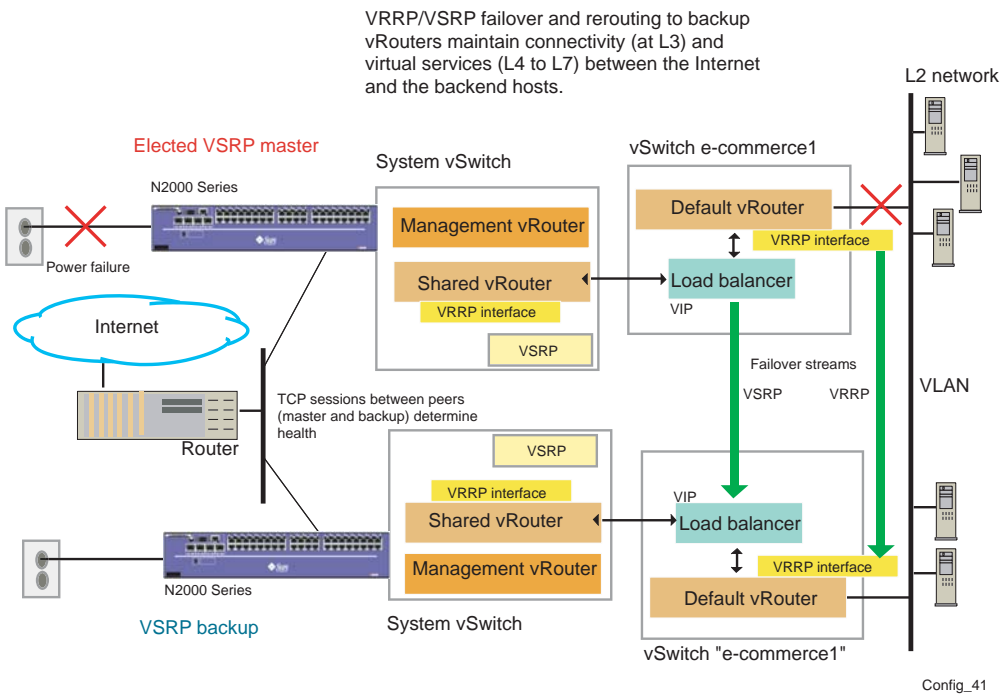
For detailed information on the optional VRRP configuration settings, refer to the *Sun N2000 Series Release 2.0 – Command Reference*.

VSRP overview

Virtual Service Redundancy Protocol (VSRP) is the Sun protocol that provides redundancy for virtual services between two configured VSRP peers. When VSRP is configured on redundant N2000 Series switches, the switches exchange state and health information (using TCP) to determine whether a switch should run in a master (active) or backup (standby) state. If the master experiences a failure, virtual service traffic switches over to the VSRP backup, or peer node.

The N2000 Series master implements all virtual service functions. The N2000 Series backup redirects service (load balance and SSL) traffic to the master. Figure 15-3 illustrates a sample network running VRRP and VSRP.

Figure 15-3. N2000 Series VSRP network



With VRRP and VSRP running on redundant N2000 Series switches, you can configure VRRP to accept “hints” from VSRP in electing a VRRP master. You do this with the `vsrpPreference` setting in the VRRP configuration. This preference setting helps guarantee the most efficient traffic routing.

When first configured, VSRP remains in a backup, or hold-down state until the first VSRP election takes place. During this period, the configured virtual services are deactivated. Once the election identifies the VSRP master, the virtual services are reactivated on the master.

VSRP configuration requirements

VSRP requires a VRRP configuration on the system vSwitch/shared vRouter (on interfaces over which a VIP should respond to ARP messages) and on one or more operator-defined vSwitches/default vRouter (on interfaces for which the N2000 Series is the default gateway). Figure 15-3 illustrates a sample network running VRRP and VSRP.

VSRP configuration steps

Perform the following configuration steps to create a VSRP peer configuration on N2000 Series vRouters:

1. Configure the VRRP interfaces, as described earlier in this chapter.

Configuring VRRP on the shared and default vRouter interfaces creates the VRRP virtual routers required for VSRP operation. VRRP is required on the following:

- Interfaces over which the VIPs are responding to ARPs and on
- Interfaces where the N2000 Series is the default gateway (interfaces to the backend networks)

2. Configure VSRP on one N2000 Series application switch in the VSRP pair.

You configure VSRP as a global protocol on each N2000 Series using vSwitch redundancy.

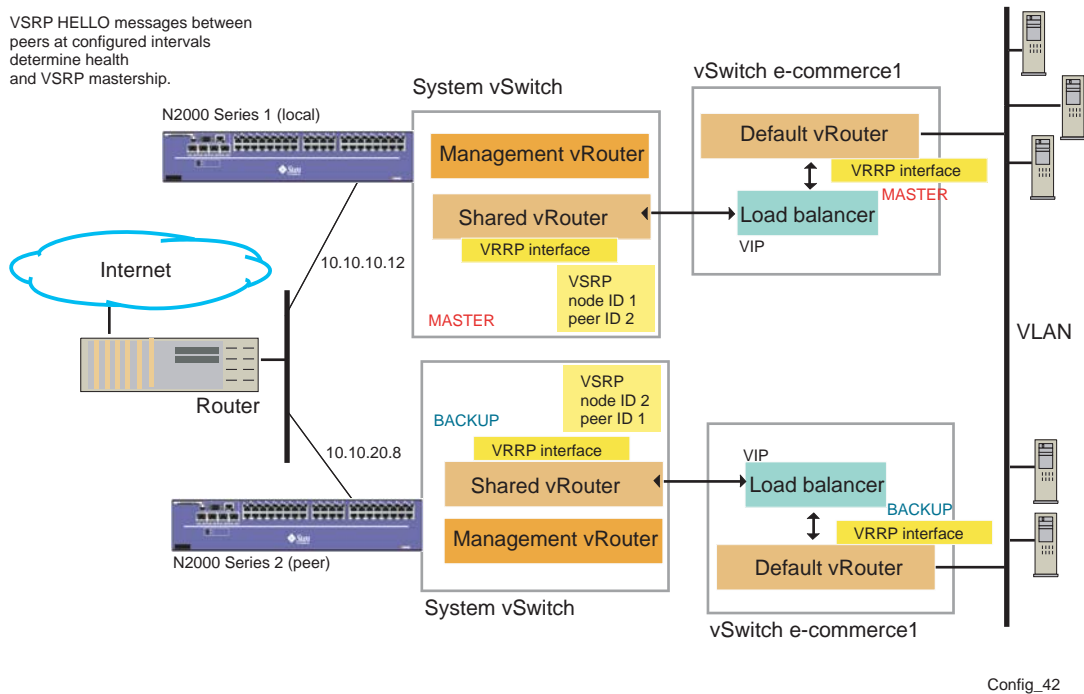
- Configure the local node using the `node` command.
- Configure the peer node for the local node using the `node peer` command.

- 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.
 - Optionally, specify an IP address to use for redirected virtual service traffic using the `IP address` command.
3. Repeat Step 2 by configuring the peer N2000 Series application switch.

How to configure VSRP peers

The following CLI session configures VSRP in the sample network illustrated in Figure 15-4. In this network, two physical N2000 Series switches are configured as VSRP peers, where N2000 Series 1 is the elected VSRP master, and N2000 Series 2 is the VSRP backup.

Figure 15-4. N2000 Series VSRP network configuration



CLI Session

1. Configure the local N2000 Series node by entering the VSRP node identifier.

```

sun(config)#
sun(config)# redundancy
sun(config-redundancy)# vsrp node ?
Usage:
  nodeId <integer>           The numeric identifier associated
                              with the local node.
  [port (1..65535)]          Port to listen on for peer
                              communication (default: 4121)
  [electionPreference (0..65535)] The preference assigned to the
                              local node. (default: 100)
  [electedIncrease (0..65535)] Increase to electionPreference
                              when a node is elected master
                              (default: 100)
  [helloTime (0..65535)]     Number of allowable missing
                              Hellos (default: 3)
  [adminState (disabled|enabled)] Administrative state of the
                              node (default: enabled)

sun(config-redundancy vsrp)# nodeID 1
sun(config-redundancy vsrp)# show node
Node ID:                1
Port:                   4121
Election Preference:    110
Elected Increase:      100
Hello Time:             2
Admin State:            enabled
Oper Status:            down
Elected State:         master
Current Election Preference: 210
Election Changes:       1

```

Options:

- **port** — The optional TCP port number that the local node listens on for communication from the peer node. Valid values are 1 through 65535. The default value is 4121.
- **electionPreference** — 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 highest electionPreference becomes the master. Valid values are 0 through 65535. The default value is 100.
- **electedIncrease** — The amount to increase the electionPreference if the local node becomes the master node. Valid values are 0 through 65535. The default value is 100.

- **helloTime** — The time, in seconds, that elapses before the system sends a HELLO message to its peer node. Valid values are 0 through 65535. The default value is 2.
2. **On the same N2000 Series switch, configure the peer node by specifying the peer numeric identifier.**

```
sun(config-redundancy vsrp)# node nodeId 1 peer peerId 2 port 4535
```

If you do not configure an `electionPreference`, the local and peer nodes use their respective Node IDs to determine which system should be the elected master. The N2000 Series with the higher `nodeId` value becomes the master node.

Option:

- **port** — If the local node ID is greater than the peer ID, the local node initiates a connection to the peer using this TCP 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.

3. **Configure the session over which the local and peer nodes exchange TCP messages.**

```
sun(config-redundancy vsrp)# node nodeId 1 peer peerId 2 session ?
vRouter <NamedIndex>          vRouter
ipAddress <IP Address>        IP address
[adminState (disabled|enabled)] administrative state (default:
                               enabled)
sun(config-redundancy vsrp)# node nodeId 1 peer peerId 2 session
system:shared 10.10.20.8
```

`vRouter` is the name of the `vSwitch:vRouter` on the local node associated with the session used for VSRP communication. The format of this argument is `vSwitch:vRouter`. The IP address is the address configured on the peer node for VSRP session communication.

You can configure multiple sessions for added resiliency. The elected switch is only considered “down” by the backup if all sessions to the elected switch fail.

Option:

- **adminState** — 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.
4. **Repeat Steps 1 through 3 on the other switch to configure the peer VSRP node.**
 5. **Optionally, specify an IP address to use for redirected VSRP traffic traffic on both the local and peer nodes.**

The backup VSRP switch may need to redirect service traffic to the master. This setting allows you to configure one IP interface per vRouter that is used to carry service traffic from the backup to the master.

```
sun(config-vSwitch-system vRouter-shared)# ip address eth.1.11  
10.10.10.11 vsrpRedirect enabled
```

If `vsrpRedirect` is set to `enabled`, the system uses this address when:

- The interface associated with the address is up, and
- The address is the lowest of all the addresses that have `redirectTraffic` set to `enabled`.

If `vsrpRedirect` is set to `disabled`, the system uses this address when:

- There are no other IP addresses with `redirectTraffic` set to `enabled`, and
- This address is the lowest address of the addresses associated with interfaces that are up.

The default value is `disabled`.

Exporting configurations to redundant switches

After you create a VRRP and VSRP configuration on an N2000 Series, you can export the configuration and import it to the appropriate redundant switch. This will save time when configuring the redundant switch with a similar running configuration.

The `show redundantConfig` command displays the parts of the current running configuration that are required to synchronize the virtual service configuration of two N2000 Series switches configured 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 for which the user has configuration access.

To configure switches for redundancy, you only copy portions of the configuration. There are some parts that are not portable, such as the Certificate and Key Manager (CKM) configuration, 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 (vRouter configuration, for example).

Redundancy configuration steps

Perform the following steps to export and import redundant configurations between N2000 Series application switches:

1. Use the `show redundantConfig` command to display and write the configuration to a user-specified text file.
2. Use the `import runningConfig` command to import the file specified in Step 1 to the redundant N2000 Series.

How to export and import redundant configurations

CLI Session

```
sun(config)# 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_3DS_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(config)# show redundantConfig saveToFile masterconfig.txt
```

On the redundant N2000 Series to which you are importing the configuration, use the `import running-config` command.

```
sun(config)# import runningConfig masterconfig.txt
```

For more information on importing running configurations, see the *Sun N2000 Series Release 2.0 – System Administration Guide* for detailed procedures.

Index

A

ACCEPT, HTTP field name 6-23
 ACCEPT_ESI, HTTP field name 6-23
 ACCEPT_LANGUAGE, HTTP field name 6-23
 access control lists
 and the N2000 Series 1-18
 configuration example (illustration) 14-2
 configuration steps 14-2
 creating 14-3
 access groups 14-6
 rules
 setting actions 14-4
 setting precedence 14-4
 setting protocols (table) 14-4 to 14-5
 defined 14-1
 deleting 14-3
 required and optional settings 14-5
 application switching
 sample N2000 Series network (illustration) 1-4

B

braces, used in predicate statements 6-31
 bridge mode load balancing 7-2
 Bridge Protocol Data Units 4-11

C

Certificate and Key Manager. *See* CKM
 chunked keyword 6-36
 CKM
 and SSL acceleration 11-1, 11-3
 configuring in existing SSL deployments 11-6
 configuring in new SSL deployments 11-5
 installing certificate 11-8

CLI

 and the N2000 Series 1-19
 client address translation (CAT)
 and proxy IP pools 8-4
 comparing load balancing with and without 8-2
 configuring 8-4
 displaying statistics 8-5
 enabling 8-5
 on inbound/outbound SLB traffic (Table) 8-3
 overview 8-2
 sample network using 8-4
 CLIENT_ADDRESS field name 6-30
 close keyword 6-36
 command line interface. *See* CLI 1-19
 configuration 4-7
 configuration steps 13-3
 configuration steps on the N2000 Series 1-21
 CONNECT keyword 6-35
 CONNECTION, HTTP field name 6-24
 CONTENT_LENGTH, HTTP field name 6-24
 cookie persistence
 overview 9-1 to 9-4
 rules 9-5
 cookieDomain 9-5
 cookieExpires 9-5, 9-6
 cookiePath 9-5, 9-6
 secure 9-5, 9-6
 sample network using (illustration) 9-2 to 9-4
 COOKIE, HTTP field name 6-24
 cookieDomain, element in cookie persistence
 rule 9-5
 cookieExpires, element in cookie persistence
 rule 9-6
 cookieName, element in cookie persistence rule 9-5
 cookiePath, element in cookie persistence rule 9-6

cookies, defined 9-1
COPY keyword 6-35
csr command
 creating a certificate signing request 11-7

D

default route
 to management vRouter 2-5
default vRouter
 configuring IP interfaces on 4-3
 on operator-defined vSwitches 3-3
 overview 1-7
DELETE keyword 6-35
designated port 4-12
DNS_TCP, server health check probe (table) 12-8
DNS_UDP, server health check probe (table) 12-7
dynamic NAT
 configuring 10-3
 overview 10-3
 sample configuration session 10-6
 sample network (illustration) 10-4

E

Ethernet interfaces
 configuring IP over 4-5
 configuring IP over LAG interfaces 4-8
 configuring IP over VLAN/LAG interfaces 4-10
 configuring IP over VLANs 4-6

F

firstObjectSwitching, forwarding action 6-37
FTP load balancing 7-9
FTP, server health check probe (table) 12-9

G

generate command
 when creating SSL key 11-6
GET enumeration 6-34
gzip keyword 6-36

H

HEAD enumeration 6-34
health checking. *See* server health checking
host
 adding 5-7
 defined 5-4, 6-3
host, defined 6-6
HOST, HTTP field name 6-24
HOST_HEADER, HTTP field name 6-25
HOST_HEADER_PORT, HTTP field name 6-25
hosts, adding to L4 to L7 network 5-8
HTTP
 field names used in object rules (table) 6-23 to 6-27
 load balancing configuration example (illustrated) 6-39 to 6-43
 overview 1-9
HTTP, server health check probe (table) 12-7
HTTP_VERSION, HTTP field name 6-26
HTTPS
 and SSL 11-1
 overview 1-9

I

ICMP, server health check probe (table) 12-7
IMAP load balancing 7-13
IMAP4, server health check probe (table) 12-9
interfaces
 See also IP interfaces, Ethernet, VLANs, LAGs
 configuring IP on a vRouter 4-3, 4-10
 configuring IP over Ethernet 4-5
 configuring L2 and L3 4-2
IP interfaces
 configuring over Ethernet 4-5
 configuring over LAGs and Ethernet 4-8
 configuring over VLAN/LAG and Ethernet 4-10
 configuring over VLANs and Ethernet 4-6
IP routing
 configuring static routes 4-15

K

- keep-alive keyword 6-36
- keywords, used in predicate statements (table) 6-34, 6-36
- keywordSets 6-36

L

- L2/L3 networks
 - configuring N2000 Series in 4-1, 4-18
 - See also* VLANs, Spanning Tree Protocol
- L4 load balancing
 - configuration hierarchy (illustration)
 - See* load balancing 5-6
- L4SLB. *See* load balancing
- L4SLB_ADV. *See* load balancing
- L4SLB_SSL. *See* load balancing
- L5 to L7 load balancing
 - configuration hierarchy 6-9
- LAGs
 - configuring over Ethernet 4-8
 - maximum number of interfaces supported on 4-8
 - maximum number per N2000 4-8
 - sample configuration (illustration) 4-9
- least connections algorithm, defined 5-14
- LIST, server health check probe (table) 12-8
- load balancer
 - assigned VIP 3-4
 - on operator-defined vSwitches 3-4
- load balancing
 - algorithms supported 5-13
 - bridge mode load balancing 7-2
 - FTP load balancing 7-9
 - hosts 5-7
 - IMAP load balancing 7-13
 - L4 configuration steps 5-4
 - L5 to L7 configuration steps 6-6
 - N2000 hardware required for 1-9, 5-1
 - object rule 6-10
 - overview 1-8, 1-8 to 1-11
 - real service 5-9
 - request policy 6-13

- request transform 6-15
- response policy 6-17
- response transform 6-19
- RTSP load balancing 7-13
- service group L5 to L7 6-5
- service groups L4 5-12
- SMTP load balancing 7-15
- sorry data 6-20
- TDLB load balancing 7-15
- virtual service L4 5-16
- virtual service L5 to L7 6-21

M

- management vRouter
 - configuring 2-3, 2-4
 - in N2000 Series network (illustration) 2-4
 - overview 1-6
 - setting default route to 2-5
- METHOD, HTTP field name 6-26
- MKCOL keyword 6-35
- MOVE keyword 6-35, 6-36

N

- N2000 Series
 - as a load balancer 1-8 to 1-11
 - as an application switch 1-4
 - as an SSL accelerator (illustration) 1-15 to 1-16
 - functionality 1-3
 - in a basic network 1-3
 - management tools 1-18 to 1-21
 - product overview 1-2
 - steps to configuring 1-21
 - N2040 (illustration) 1-2
 - N2120 (illustration) 1-2
 - network address translation
 - overview 10-2
 - See also* static NAT, dynamic NAT
- O**
- object rule
 - defined 6-3
 - object rules

- action fields 6-37
 - forward, defined (table) 6-37
- creating in L5 to L7 network 6-10
- predicate keywords 6-34
- predicate keywordSets 6-36
- predicate operators used in 6-31
- object, defined 6-3
- operator-defined vSwitches
 - allocating port and bandwidth resources 3-6
 - configuring 3-1 to 3-6
 - overview 1-7, 3-2 to 3-4
 - supported functions 3-2
- operators, used in predicate statements (table) 6-31 to 6-33
- OPTIONS enumeration 6-34
- P**
- path costs 4-12
- PIP
 - proxy IP address pool 8-2
- policy, defined 6-3
- POP3, server health check probe (table) 12-9
- port bandwidth, allocating 3-6
- PORT, HTTP field name 6-26
- POST enumeration 6-34
- predicate, defined 6-4
- product overview 1-2
- PROPFIND keyword 6-35
- PROPPATCH keyword 6-35
- protocols, for ACL traffic matching 14-5
- proxy IP pools 8-2
 - using with CAT 8-4
- PUT keyword 6-34, 6-35
- Q**
- quotes, used in predicate statements 6-31
- R**
- RADIUS, server health check probe (table) 12-7
- RANGE, HTTP field name 6-27
- RAW_TCP, server health check probe (table) 12-8
- RAW_UDP, server health check probe (table) 12-8
- real service
 - defined 5-4, 6-4
- real service, defined 6-6
- real services
 - configuring 5-9
- REFERRER, HTTP field name 6-25
- request policy
 - creating 6-13
 - defined 6-5
- request transform
 - creating 6-15
 - defined 6-5
- resources
 - allocating vSwitch port and bandwidth 3-6
- response policy
 - creating 6-17
 - defined 6-4
- response transform
 - creating 6-19
 - defined 6-4
- RESPONSE_CODE, HTTP field name 6-27
- RIP
 - configuring interfaces 4-17
 - overview 4-16
- root bridge 4-11
- root port 4-12
- round robin algorithm, defined 5-14
- route
 - setting default to management vRouter 2-5
- RTSP load balancing 7-13
- RTSP, server health check probe (table) 12-9
- S**
- SCRIPT, server health check probe (table) 12-10
- Secure Shell. *See* SSH 1-19
- Secure Socket Layer. *See* SSL
- secure, keyword in cookie persistence rule 9-6

- server cloaking 6-19
- server health checking
 - and the N2000 Series 1-17
 - out-of-band (illustration) 12-2
 - applying profile to service group 12-10
 - setting up profile 12-5 to 12-11
 - naming 12-5
 - probes
 - DNS_TCP (table) 12-8
 - DNS_UDP (table) 12-7
 - FTP (table) 12-9
 - HTTP (table) 12-7
 - ICMP (table) 12-7
 - IMAP4 (table) 12-9
 - LIST (table) 12-8
 - POP3 (table) 12-9
 - RADIUS (table) 12-7
 - RAW_TCP (table) 12-8
 - RAW_UDP (table) 12-8
 - RTSP (table) 12-9
 - SCRIPT (table) 12-10
 - SMTP (table) 12-10
 - TCP (table) 12-7
 - special considerations when configuring 12-4
- server load balancing. *See* load balancing
- service bandwidth, allocating 3-6
- service group
 - defined 5-4, 6-5
- shared vRouters
 - configuring 2-6 to 2-7
 - overview 1-6
 - single and multiple (illustrated) 2-6 to 2-7
- SMTP load balancing 7-15
- SMTP, server health check probe (table) 12-10
- SNMP
 - and the N2000 Series 1-20
 - supported versions 1-20
- sorry data
 - defined 6-5
- sorryData
 - creating 6-20
- sorryString, forwarding action 6-38
- Spanning Tree Protocol 4-13 to 4-15
 - bridge protocol data units (BPDU)s 4-11
 - configuring on a VLAN (illustration) 4-13 to 4-15
 - designated port 4-12
 - path costs 4-12
 - root bridge 4-11
 - root port 4-12
- SSH 1-19
- SSL
 - acceleration and regeneration overview
 - (illustration) 1-15 to 1-16, 11-2 to 11-3
 - and CKM 11-1, 11-3
 - configuring in existing SSL deployments 11-6
 - configuring in new SSL deployments 11-5
 - for existing keys and certificates 11-4
 - for new keys and certificates 11-4
 - generating key and certificate request 11-6 to 11-8
 - installing certificate 11-8
 - submitting certificate request to CA 11-8
 - temporary certificates 11-8
 - working with existing keys and certificates 11-9
 - and HTTPS 11-1
 - configuring on operator-defined vSwitch 11-10 to 11-12
 - N2000 Series requirements for using 11-1
 - regeneration
 - overview (illustration) 13-2
 - steps to configuring
 - serviceGroup definition 13-3
- static NAT
 - configuring 10-3
 - overview 10-3
 - sample configuration session 10-5
 - sample network (illustration) 10-4
- static routes
 - configuring 4-15
- switch-managed cookie 9-2
- system vSwitch
 - configuring management and shared vRouters
 - on 2-3
 - displaying information about 2-2
 - illustrated 2-2
 - overview 1-6

T

TCP, server health check probe (table) 12-7
TDLB load balancing 7-15
Telnet 1-19
TRACE keyword 6-35
traffic filtering 1-18
TRANSFER_ENCODING, HTTP field name 6-25

U

uniform resource identifier. *See* URI 6-28
UPGRADE, HTTP field name 6-26
URI
 predicates (table) 6-29 to 6-30
 structure (illustration) 6-28
URI field name 6-29
URI_ALLFILE field name 6-30
URI_BASENAME field name 6-30
URI_HOST field name 6-29
URI_PATH field name 6-29
URI_PORT field name 6-29
URI_QUERY field name 6-30
URI_SCHEME field name 6-29
URI_SUFFIX field name 6-30
USER_AGENT, HTTP field name 6-27

V

VIP
 and the load balancer application 3-4
virtual service
 creating 6-21
 defined 5-4, 6-6
virtual service groups, creating 5-18, 6-22
virtual switching and routing. *See* vSwitches
VLAN 4-7
VLANs
 and Spanning Tree Protocol 4-13
 configuring on Ethernet interfaces 4-6

 configuring with LAGs 4-10
 IDs associated with 4-6
 maximum number per vRouter 4-6
 sample configuration (illustration) 4-7
vRouters. *See* default vRouter, management vRouter,
 shared vRouters
VRRP
 configuration steps 15-4
 overview 15-2
 sample network (illustration) 15-5
vsGroup command, using 5-18, 6-22
VSRP
 configuration steps 15-12
 overview 15-11
vSwitches 1-5
 operator-defined 1-7
 system 1-6

W

Web interface 1-20
weighed hash algorithm, defined
 sample network (illustration) 5-16
weighted hash algorithm, defined 5-14
weighted random algorithm, defined 5-14

X

X.509 certificate authorities, commonly known 11-4